



UNIVERSIDAD NACIONAL
AUTÓNOMA DE
MÉXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA
DE LA COMPUTACIÓN

**GUÍA PARA LA DESCRIPCIÓN Y DOCUMENTACIÓN DE ARQUITECTURAS
DE SOFTWARE UTILIZANDO LENGUAJES DE DESCRIPCIÓN DE
ARQUITECTURA**

**TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRA EN INGENIERÍA (COMPUTACIÓN)**

**PRESENTA:
SANDRA LILIANA RAMÍREZ MORA**

**TUTOR:
M. EN C. MARÍA GUADALUPE ELENA IBARGÜENGOITIA GONZÁLEZ
FACULTAD DE CIENCIAS, UNAM.**

MÉXICO, D. F. ENERO 2016



Universidad Nacional
Autónoma de México

Dirección General de Bibliotecas de la UNAM

Biblioteca Central



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Agradecimientos

A la Maestra Guadalupe Ibargüengoitia, por ser mi profesora y tutora, por su gran apoyo en el transcurso de la maestría, por sus enseñanzas en las clases, y por sus consejos y dedicación durante el desarrollo de este trabajo.

A la doctora Hanna Oktaba, por todo su apoyo, por compartir su experiencia y conocimiento durante la maestría, por aceptar ser mi sinodal, por contribuir al buen desarrollo de este trabajo, y por sus consejos académicos y profesionales.

Al doctor Jorge Luis Ortega, por sus enseñanzas, por aceptar ser mi sinodal y revisar este trabajo, y por su labor en el posgrado.

Al maestro Gustavo Márquez, por su trabajo como profesor del posgrado, por aceptar ser mi sinodal, y por su disposición para revisar este trabajo.

A la doctora María del Pilar Ángeles, por aceptar ser mi sinodal y revisar este trabajo, y por su valiosa aportación para la mejora del mismo.

Al Posgrado en Ciencia e Ingeniería de la Computación, en especial a mis profesores.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT), por su apoyo para la realización de mis estudios de maestría.

A mi familia, especialmente a mis padres, por su apoyo incondicional.

A Angel Zúñiga, por ser parte fundamental de mi vida, por sus consejos, apoyo y comprensión.

A Dios, por la vida.

Contenido

INTRODUCCIÓN	1
CONTEXTO	1
PLANTEAMIENTO DEL PROBLEMA	3
TRABAJOS RELACIONADOS	3
HIPÓTESIS	4
PROPUESTA DE SOLUCIÓN	4
ESTRUCTURA DEL TRABAJO	6
PARTE 1 MARCO TEÓRICO	8
CAPÍTULO 1. CONTEXTO DE LA ARQUITECTURA DE SOFTWARE	9
1.1. ¿QUÉ ES UNA ARQUITECTURA DE SOFTWARE?	9
1.2. IMPORTANCIA DE LA ARQUITECTURA DE SOFTWARE	9
1.3. PROCESO DE ARQUITECTURA EN EL CICLO DE VIDA DEL SOFTWARE	11
1.4. ARQUITECTURA EN EL CONTEXTO DE NEGOCIO	15
1.5. ARQUITECTURA EN EL CONTEXTO PROFESIONAL	15
CAPÍTULO 2. DOCUMENTACIÓN DE ARQUITECTURAS DE SOFTWARE Y DESCRIPCIONES DE ARQUITECTURA	16
2.1. DOCUMENTACIÓN DE ARQUITECTURAS DE SOFTWARE	16
2.2. DESCRIPCIONES DE ARQUITECTURA	18
CAPÍTULO 3. MARCO CONCEPTUAL DE LAS DESCRIPCIONES DE ARQUITECTURA	22
3.1. INTERESADOS O <i>STAKEHOLDERS</i>	22

3.2.	INTERESES O <i>CONCERNS</i> ARQUITECTÓNICOS _____	22
3.3.	PUNTOS DE VISTA ARQUITECTÓNICOS _____	24
3.4.	VISTAS ARQUITECTÓNICAS _____	25
3.5.	MARCO DE TRABAJO DE ARQUITECTURA _____	25
3.6.	ESTRUCTURAS Y ELEMENTOS ARQUITECTÓNICOS _____	25
3.7.	PERSPECTIVAS ARQUITECTÓNICAS _____	26
3.8.	ESTILOS Y PATRONES ARQUITECTÓNICOS _____	27
3.9.	MODELOS DE ARQUITECTURA Y TIPOS DE MODELOS _____	27
3.10.	RAZONAMIENTO Y DECISIONES DE ARQUITECTURA _____	28
3.11.	ELEMENTOS DE LAS DESCRIPCIONES DE ARQUITECTURA, CORRESPONDENCIAS Y REGLAS DE CORRESPONDENCIA _____	29
3.12.	MODELO CONCEPTUAL DE UNA DESCRIPCIÓN DE ARQUITECTURA _____	29
 CAPÍTULO 4. LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA _____		32
4.1.	NOTACIONES DE MODELADO Y MÉTODOS FORMALS EN ARQUITECTURA DE SOFTWARE _____	32
4.2.	¿QUÉ SON LOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA? _____	33
4.3.	CARACTERÍSTICAS DE UN LENGUAJE DE DESCRIPCIÓN DE ARQUITECTURA _____	35
4.4.	DIFERENCIA ENTRE LOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA Y OTROS LENGUAJES _____	36
4.5.	VENTAJAS Y DESVENTAJAS DE LOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA _____	37
 PARTE 2 GUÍA PARA DOCUMENTAR ARQUITECTURAS DE SOFTWARE UTILIZANDO LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA _____		39
 CAPÍTULO 5. CONSIDERACIONES SOBRE LA GUÍA _____		40
5.1.	OBJETIVO _____	40
5.2.	CARACTERÍSTICAS PRINCIPALES _____	40
5.3.	¿A QUIÉN ESTÁ DIRIGIDA? _____	41

5.4.	FUNDAMENTOS PRIMORDIALES QUE DIRIGEN EL CONTENIDO	41
5.5.	ACTIVIDADES GENERALES PARA ELABORAR DESCRIPCIONES DE ARQUITECTURA (GUÍA RÁPIDA DE DOCUMENTACIÓN)	43
5.6.	CONSIDERACIONES ADICIONALES	45

CAPÍTULO 6. IDENTIFICACIÓN Y DOCUMENTACIÓN DE LA INFORMACIÓN GENERAL DE UNA DESCRIPCIÓN DE LA ARQUITECTURA 47

6.1.	IDENTIFICACIÓN Y DOCUMENTACIÓN DEL SISTEMA DE INTERÉS	47
6.2.	IDENTIFICACIÓN Y DOCUMENTACIÓN DE INFORMACIÓN COMPLEMENTARIA	49
6.3.	DOCUMENTACIÓN DE RESULTADOS DE EVALUACIONES ARQUITECTÓNICAS	54

CAPÍTULO 7. IDENTIFICACIÓN Y DOCUMENTACIÓN DE INTERESADOS E INTERESES 58

7.1.	TÉCNICAS PARA LA IDENTIFICACIÓN DE INTERESADOS	59
7.2.	TÉCNICAS PARA LA IDENTIFICACIÓN DE INTERESES	65
7.3.	DOCUMENTACIÓN DE INTERESADOS E INTERESES EN UNA DESCRIPCIÓN DE ARQUITECTURA	74

CAPÍTULO 8. CRITERIOS PARA SELECCIONAR LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA 81

8.1.	CARACTERÍSTICAS ESPECIFICADAS POR EL ESTÁNDAR ISO/IEC/IEEE 42010:2011	82
8.2.	ESTILOS ARQUITECTÓNICOS	87
8.3.	HERRAMIENTAS	88
8.4.	PLATAFORMAS Y TECNOLOGÍAS RELACIONADAS	90
8.5.	DOCUMENTACIÓN E INFORMACIÓN RELACIONADA	91
8.6.	EXTENSIBILIDAD	92
8.7.	VENTAJAS SOBRE NOTACIONES SEMIFORMALES	93

CAPÍTULO 9. IDENTIFICACIÓN Y DOCUMENTACIÓN DE PUNTOS DE VISTA ARQUITECTÓNICOS _____ 95

9.1. ACTIVIDADES PARA IDENTIFICAR Y SELECCIONAR PUNTOS DE VISTA ARQUITECTÓNICOS _____ 96

9.2. DEFINICIÓN Y DOCUMENTACIÓN DE PUNTOS DE VISTA _____ 99

CAPÍTULO 10. DESARROLLO Y DOCUMENTACIÓN DE VISTAS ARQUITECTÓNICAS _____ 105

10.1. DESARROLLO DE VISTAS ARQUITECTÓNICAS _____ 106

10.2. ANÁLISIS DE LAS VISTAS Y MODELOS ARQUITECTÓNICOS _____ 117

10.3. DOCUMENTACIÓN DE VISTAS ARQUITECTÓNICAS _____ 120

CAPÍTULO 11. MODELADO Y ESPECIFICACIÓN ARQUITECTÓNICA UTILIZANDO LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA _____ 124

11.1. ESPECIFICACIÓN ARQUITECTÓNICA UTILIZANDO UN LENGUAJE DE DESCRIPCIÓN DE ARQUITECTURA DE PRIMERA GENERACIÓN (WRIGHT) _____ 125

11.2. ESPECIFICACIÓN Y MODELADO ARQUITECTÓNICO UTILIZANDO UN LENGUAJE DE DESCRIPCIÓN DE ARQUITECTURA DE INTERCAMBIO (ACME) _____ 133

11.3. ESPECIFICACIÓN Y MODELADO UTILIZANDO UN LDA DE PROPÓSITO ESPECÍFICO (AADL) _____ 145

11.4. CONSTRUCCIÓN DE MODELOS UTILIZANDO UN LENGUAJE DE DESCRIPCIÓN DE ARQUITECTURA FLEXIBLE Y EXTENSIBLE (xADL) _____ 158

11.5. LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA Y TECNOLOGÍAS DE VANGUARDIA _____ 166

11.6. CONCLUSIONES SOBRE EL USO DE LOS LDAs _____ 170

CAPÍTULO 12. DOCUMENTACIÓN DE RELACIONES ARQUITECTÓNICAS _____ 175

12.1. DOCUMENTACIÓN DEL ANÁLISIS DE CONSISTENCIA _____ 176

12.2. DOCUMENTACIÓN DE REGLAS DE CORRESPONDENCIA _____ 176

12.3. DOCUMENTACIÓN DE CORRESPONDENCIAS _____ 179

CAPÍTULO 13. DOCUMENTACIÓN DE RAZONAMIENTO Y DECISIONES ARQUITECTÓNICAS	182
13.1. DOCUMENTACIÓN DE RAZONAMIENTO ARQUITECTÓNICO	182
13.2. DOCUMENTACIÓN DE DECISIONES ARQUITECTÓNICAS	184
CAPÍTULO 14. VALIDACIÓN Y MANTENIMIENTO DE UNA DESCRIPCIÓN DE ARQUITECTURA	185
14.1. MANTENIMIENTO	185
14.2. VALIDACIÓN	185
CAPÍTULO 15. EVALUACIÓN DE LA GUÍA	190
15.1. RESULTADOS DE LA EVALUACIÓN	190
15.2. MEJORAS REALIZADAS	192
CONCLUSIONES	193
APÉNDICES	198
A. CATÁLOGO DE INTERESADOS	199
B. CATÁLOGO DE ATRIBUTOS DE CALIDAD	207
C. CATÁLOGO DE PUNTOS DE VISTA	215
D. PLANTILLAS DE DOCUMENTACIÓN	234
E. EJEMPLO DE UNA DESCRIPCIÓN DE ARQUITECTURA	251
BIBLIOGRAFÍA	260

Lista de tablas

TABLA 4.1 ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA.	34
TABLA 5.1 ACTIVIDADES PARA ELABORAR DESCRIPCIONES DE ARQUITECTURA Y LOS CAPÍTULOS EN QUE SE PRESENTAN.	45
TABLA 6.1 EJEMPLO DE UNA LISTA DE VERIFICACIÓN CON INTERESES ARQUITECTÓNICOS.	55
TABLA 6.2 LISTA DE VERIFICACIÓN PARA UNA DESCRIPCIÓN DE ARQUITECTURA CON BASE EN LO ESPECIFICADO POR EL ESTÁNDAR ISO/IEC/IEEE 42010:2011.	56
TABLA 7.1 EJEMPLO DEL REGISTRO DE INTERESADOS.	62
TABLA 7.2 EJEMPLO DE ASIGNACIÓN DE VALORES A INTERESADOS.	63
TABLA 7.3 CATEGORÍAS DE METAS DE NEGOCIO (BASS, CLEMENTS Y KAZMAN 2012).	70
TABLA 7.4 EJEMPLO DEL REGISTRO DE INTERESADOS EN UNA DESCRIPCIÓN DE ARQUITECTURA.	74
TABLA 7.5 EJEMPLO DE REGISTRO DE LOS PROPÓSITOS DEL SISTEMA.	75
TABLA 7.6 EJEMPLO DEL REGISTRO DE INTERESES.	76
TABLA 7.7 EJEMPLO DEL REGISTRO DE INTERESES.	76
TABLA 7.8 EJEMPLO DEL REGISTRO DE INTERESES.	76
TABLA 7.9 REGISTRO DE RIESGOS.	77
TABLA 7.10 REGISTRO DE IMPACTOS POTENCIALES.	78
TABLA 7.11 REGISTRO DE ATRIBUTOS DE CALIDAD.	79
TABLA 7.12 EJEMPLO DEL REGISTRO DE LA RELACIÓN ENTRE INTERESES E INTERESADOS.	80
TABLA 8.1 ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA.	81
TABLA 8.2 INTERESES DE ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA.	83
TABLA 8.3 POSIBLES INTERESADOS SOPORTADOS POR ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA.	84
TABLA 8.4 TIPOS DE MODELOS DE ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA.	86
TABLA 8.5 POSIBLES PUNTOS DE VISTA DE ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA.	86

TABLA 8.6 ELEMENTOS DE POSIBLES REGLAS DE CORRESPONDENCIA DE ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA.	87
TABLA 8.7 ESTILOS ARQUITECTÓNICOS DE ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA.	88
TABLA 8.8 HERRAMIENTAS DE ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA.	90
TABLA 8.9 PLATAFORMAS Y TECNOLOGÍAS RELACIONADAS CON ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA.	91
TABLA 8.10 DOCUMENTACIÓN DE ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA.	92
TABLA 8.11 CARACTERÍSTICAS DE EXTENSIBILIDAD DE ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA.	93
TABLA 8.12 VENTAJAS DE ALGUNOS LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA RESPECTO A NOTACIONES MENOS FORMALES.	94
TABLA 9.1 EJEMPLO PARA ENMARCAR INTERESES EN PUNTOS DE VISTA.	97
TABLA 9.2 REGISTRO DE INTERESES Y ANTI-INTERESES EN UN PUNTO DE VISTA.	100
TABLA 9.3 REGISTRO DE INTERESADOS EN UN PUNTO DE VISTA.	100
TABLA 9.4 REGISTRO DE REGLAS DE CORRESPONDENCIA EN UN PUNTO DE VISTA.	103
TABLA 9.5 REGISTRO DE LAS OPERACIONES DE UNA VISTA EN UN PUNTO DE VISTA.	104
TABLA 11.1 NOTACIÓN GRÁFICA DE ACMESTUDIO.	133
TABLA 11.2 NOTACIÓN GRÁFICA DE AADL.	146
TABLA 11.3 NOTACIÓN GRÁFICA DE XADL.	159
TABLA 12.1 EJEMPLO DE UNA CORRESPONDENCIA.	181
TABLA 14.1 LISTA DE VERIFICACIÓN DE UNA DESCRIPCIÓN DE ARQUITECTURA (CLEMETS ET AL. 2003).	188
TABLA 14.2 LISTA DE VERIFICACIÓN DE UNA DESCRIPCIÓN DE ARQUITECTURA SEGÚN EL CONTENIDO ESPECIFICADO POR EL ESTÁNDAR ISO/IEC/IEEE 42010:2011.	189
TABLA C.1 PERSPECTIVA DE SEGURIDAD.	222
TABLA C.2 PERSPECTIVA DE EFICIENCIA DE DESEMPEÑO Y ESCALABILIDAD.	223
TABLA C.3 PERSPECTIVA DE DISPONIBILIDAD Y FLEXIBILIDAD.	224
TABLA C.4 PERSPECTIVA DE EVOLUCIÓN.	224
TABLA C.5 PERSPECTIVA DE ACCESIBILIDAD.	225

TABLA C. 6 PERSPECTIVA DE RECURSOS DE DESARROLLO. _____	225
TABLA C. 7 PERSPECTIVA DE INTERNACIONALIZACIÓN. _____	226
TABLA C. 8 PERSPECTIVA DE LOCALIZACIÓN. _____	227
TABLA C. 9 PERSPECTIVA DE REGULACIÓN. _____	227
TABLA C. 10 PERSPECTIVA DE USABILIDAD. _____	227
TABLA D. 1 RELACIÓN ENTRE SECCIONES DE LA PLANTILLA Y CAPÍTULOS. _____	234

Lista de figuras

FIGURA 3.1 MODELO CONCEPTUAL (O METAMODELO) DE UNA DESCRIPCIÓN DE ARQUITECTURA (ISO/IEC/IEEE 42010:2011), USANDO LAS CONVENCIONES DE LOS DIAGRAMAS DE CLASES. _____	30
FIGURA 5.1 DIAGRAMA DE ACTIVIDADES PARA ELABORAR DESCRIPCIONES DE ARQUITECTURA. _____	44
FIGURA 7.1 EJEMPLO DE UN ÁRBOL DE UTILIDAD (BASS, CLEMENTS Y KAZMAN 2012). _____	72
FIGURA 11.1 ACMESTUDIO. _____	138
FIGURA 11.2 MODELO DEL SISTEMA DE MONITOREO. _____	144
FIGURA 11.3 NOTACIÓN GRÁFICA DE RELACIONES Y LIGAS ENTRE COMPONENTES EN AADL. _____	147
FIGURA 11.4 NOTACIÓN GRÁFICA PARA INTERACCIONES EN AADL. _____	148
FIGURA 11.5 REPRESENTACIÓN GRÁFICA DE UN EJEMPLO CON AADL. _____	150
FIGURA 11.6 REPRESENTACIÓN GRÁFICA DEL PROCESO <i>PROCESO_DE_CONTROL</i> . _____	151
FIGURA 11.7 REPRESENTACIÓN GRÁFICA DEL SISTEMA <i>SISTEMA PROCESAMIENTO TEXTO</i> . _____	153
FIGURA 11.8 HERRAMIENTA OSATE2. _____	155
FIGURA 11.9 REPRESENTACIÓN GRÁFICA DE LOS ELEMENTOS ARQUITECTÓNICOS. _____	157
FIGURA 11.10 REPRESENTACIÓN GRÁFICA DEL PROCESO <i>CONTROL</i> . _____	157
FIGURA 11.11 REPRESENTACIÓN GRÁFICA DEL SISTEMA DE MONITOREO CON ARCHSTUDIO. _____	164
FIGURA 11.12 EJEMPLO DE UNA ARQUITECTURA EMPRESARIAL CON ARCHIMATE®. _____	167
FIGURA B.1 MODELO DE CALIDAD DEL PRODUCTO SOFTWARE _____	207
FIGURA B.2 MODELO DE CALIDAD EN USO _____	212

Lista de códigos

CÓDIGO 11.1 EJEMPLO DE LA DEFINICIÓN DE UN ESTILO CLIENTE-SERVIDOR CON WRIGHT.	131
CÓDIGO 11.2 EJEMPLO DE LA DEFINICIÓN DE UNA CONFIGURACIÓN CON WRIGHT.....	132
CÓDIGO 11.3 EJEMPLO DE LA DEFINICIÓN DE UN COMPONENTE CON ACME.	135
CÓDIGO 11.4 EJEMPLO DE LA DEFINICIÓN DE UN CONECTOR CON ACME.	135
CÓDIGO 11.5 EJEMPLO DE LA DEFINICIÓN DE UN SISTEMA CON ACME.	136
CÓDIGO 11.6 EJEMPLO DE LA ESPECIFICACIÓN DE UNA FAMILIA DE SISTEMAS CON ACME.	143
CÓDIGO 11.7 EJEMPLO DE UN SISTEMA DE MONITOREO ESPECIFICADO CON ACME.	144
CÓDIGO 11.8 EJEMPLO DE LA DEFINICIÓN DE UN COMPONENTE CON AADL.	149
CÓDIGO 11.9 EJEMPLO DE LA DEFINICIÓN DE TIPOS DE ELEMENTOS CON AADL.	150
CÓDIGO 11.10 EJEMPLO DE LA ESPECIFICACIÓN DE UNA INTERACCIÓN CON AADL.	152
CÓDIGO 11.11 EJEMPLO DE LA DEFINICIÓN DE UN MODO CON AADL.	153
CÓDIGO 11.12 EJEMPLO DE LA DEFINICIÓN DE FLUJOS CON AADL.	154
CÓDIGO 11.13 EJEMPLO DE LA DEFINICIÓN DE UNA PROPIEDAD CON AADL.	154
CÓDIGO 11.14 EJEMPLO DE UN SISTEMA DE CONTROL DE VELOCIDAD ESPECIFICADO CON AADL.....	157
CÓDIGO 11.15 EJEMPLO UN ESQUEMA CON xADL.....	162
CÓDIGO 11.16 EJEMPLO UN ESQUEMA QUE IMPLEMENTA UNA EXTENSIÓN CON xADL.....	163
CÓDIGO 11.17 EJEMPLO DE LA DEFINICIÓN DE UN COMPONENTE CON xADL.	165

Introducción

Contexto

En la actualidad, los sistemas de software pueden implicar cierta complejidad o demandar características precisas, lo que ha producido nuevos desafíos para las organizaciones que crean y utilizan tales sistemas. Poder visualizar un sistema como una abstracción de alto nivel, con sus principales componentes y características, puede resultar útil para realizar actividades de análisis, diseño, implementación, pruebas y mantenimiento. Debido a esto, son aplicados conceptos, principios y técnicas de Arquitectura de Software en el desarrollo de sistemas.

La arquitectura de un sistema de software es el conjunto de estructuras requeridas para razonar sobre el sistema, que abarca elementos de software, las relaciones entre ellos, y las propiedades de ambos (Bass, Clements y Kazman 2012). Es el conjunto de conceptos y propiedades fundamentales de un sistema en su entorno, plasmadas en sus elementos, relaciones y en los principios de su diseño y evolución (ISO/IEC/IEEE 42010:2011).

La arquitectura de un sistema es importante debido a que permite inhibir o habilitar atributos de calidad, facilita el razonamiento sobre el sistema, mejora la comunicación entre los interesados, y permite razonar sobre el costo y calendarización del sistema (Bass, Clements y Kazman, 2012). Por otra parte se puede decir que todo sistema, ya sea grande o pequeño, tiene una arquitectura, ya que tiene un comportamiento y está compuesto por piezas que están relacionadas entre sí. Por tal motivo, la arquitectura es una propiedad intrínseca y fundamental de todo sistema (Rozansky y Woods 2005).

La arquitectura de un sistema de software implica un proceso, que involucra actividades de concepción, definición, comunicación, implementación, documentación, mantenimiento y mejora de la arquitectura a través del ciclo de vida del sistema (ISO/IEC/IEEE 42010:2011).

En particular, la documentación arquitectónica es la tarea de producir documentos sobre la arquitectura de un sistema. La documentación arquitectónica es una tarea amplia, que puede incluir actividades de especificación, representación y descripción de la arquitectura. Debido a esto, la documentación de una arquitectura puede ser formal o no, puede contener modelos o no, y los documentos pueden describir o especificar (Clements et al. 2003).

Para documentar una arquitectura existen distintos enfoques y técnicas, entre los que se encuentran el enfoque 4+1 (Kruchten 1995), y el enfoque "*Views and Beyond*" (Clements et al. 2003). La mayoría de dichos enfoques utilizan vistas arquitectónicas como principal medio de documentación, sin embargo, es importante mencionar que para que una documentación sea útil, debe ser detallada, no ambigua, organizada, accesible, concreta, descriptiva y preceptiva, y debe soportar todas las necesidades relevantes del sistema (Bass, Clements y Kazman 2012).

Al realizar una adecuada documentación de la arquitectura de un sistema, los documentos de arquitectura generados pueden servir como medio de educación, como un medio fundamental de comunicación entre interesados, y como base en el análisis y

construcción del sistema (Clements et al. 2003). Además, una arquitectura bien documentada hace más útil la información que incluye. Según Bass, Clements y Kazman (2012), la documentación debe visualizarse como algo esencial para producir un producto de software de alta calidad.

Debido a la importancia de la documentación arquitectónica, han surgido estándares de documentación cuyo objetivo es especificar las disposiciones que debe cumplir la documentación de la arquitectura de un sistema. Tal es el caso del estándar internacional ISO/IEC/IEEE 42010:2011 “*Systems and software engineering - Architecture description*”, que proporciona una ontología principal para la documentación de arquitecturas. Especifica la manera en que las descripciones de arquitectura de un sistema son organizadas y expresadas.

Las descripciones de arquitectura son productos de trabajo usados para expresar arquitecturas. Son un medio útil y completo para documentar arquitecturas de software, y se conforman de distintos elementos, tales como puntos de vista, vistas y modelos arquitectónicos. Éstos últimos son parte fundamental de una descripción de arquitectura, ya que son representaciones de aspectos arquitectónicos del sistema, que permiten, entre otras cosas, la comunicación entre los interesados del sistema (Rozansky y Woods 2005).

Un modelo arquitectónico puede tomar varias formas (incluir texto, dibujos informales, diagramas u otras características más formales) y usa convenciones de modelado apropiadas para los intereses arquitectónicos que representa.

En ocasiones, la práctica del diseño y modelado arquitectónico ha tendido a ser *ad-hoc*¹ e informal, y como resultado, los modelos arquitectónicos han sido entendidos pobremente por los desarrolladores; los modelos no han podido ser analizados en cuanto a su consistencia y completitud; y las restricciones asumidas en el diseño inicial de una arquitectura no son mantenidas a medida que un sistema evoluciona. En respuesta a estos problemas, investigadores de la industria y académicos han propuesto notaciones formales para representar y analizar diseños arquitectónicos, generalmente referidas como *Lenguajes de Descripción de Arquitectura* (LDAs). Estas notaciones usualmente proporcionan, tanto un *framework* conceptual, como una sintaxis concreta para representar arquitecturas de software. Por lo general, los Lenguajes de Descripción de Arquitectura proporcionan herramientas para compilar, analizar o simular modelos arquitectónicos (Garlan, Monroe y Wile 1997).

Los Lenguajes de Descripción de Arquitectura son usados en diversas actividades durante el proceso de arquitectura, incluyendo la documentación, análisis y evaluación de características arquitectónicas.

En la actualidad existen muchos Lenguajes de Descripción de Arquitectura, los cuales han surgido desde la década de los 90's. Algunos de éstos lenguajes son AADL (Feiler, Gluch y Hudak 2006), Acme (Garlan, Monroe y Wile 1997), Aesop (Garlan, Allen y Ockerbloom 1994), ByADL (Ruscio et al. 2010), Darwin (Magee et al. 1995), Pi-ADL (Oquendo 2004),

¹ Que es apropiado, adecuado, hecho especialmente para un fin determinado, o que fue pensado para una situación concreta.

Rapide (Luckham y Vera 1995), Unicon (Shaw et al. 1995), Wright (Allen 1997), y xADL (Dashofy, Hoek y Taylor 2001).

Planteamiento del problema

Aunque las actividades de documentación arquitectónica son una parte fundamental del proceso de arquitectura, no siempre se realizan adecuadamente. Para solucionar esto, se han creado estándares internacionales como el ANSI/IEEE 1471:2000 y su sucesor el ISO/IEC/IEEE 42010:2011. Sin embargo, no existe una guía suficientemente detallada que permita llevarlos a la práctica, que incluya técnicas de arquitectura específicas, consejos útiles, y plantillas de documentación que representen una forma práctica de generar productos de trabajo de la arquitectura de un sistema de software.

Por otra parte, el uso de técnicas informales y *ad-hoc* en la documentación de una arquitectura ocasiona que ésta sea mal entendida, ambigua, poco detallada, poco precisa, inconsistente, y en general, insuficiente para propósitos de análisis, mantenimiento, y desarrollo de sistemas.

Aunque existe una gran variedad de Lenguajes de Descripción de Arquitectura (LDAs) que pueden ser utilizados para especificar, modelar o analizar arquitecturas de software de manera precisa, detallada y no ambigua, dichos lenguajes no son tan utilizados como otras notaciones menos formales o de propósito general (como UML). Algunas de las razones del escaso uso de los LDAs, sobre todo en la industria, son: el poco conocimiento sobre su existencia, la falta de documentación que facilite su uso en el proceso de arquitectura, la dificultad para seleccionar un lenguaje debido a la gran cantidad de LDAs que existen, y la complejidad que implican. Por otra parte, la documentación de algunos LDAs no cuenta con ejemplos reales y no especifica cómo utilizarlos en el proceso de arquitectura, sobre todo para realizar descripciones de arquitectura y para utilizarlos en conjunto con estándares internacionales de documentación.

En resumen, el problema identificado es que no existe una guía de documentación suficientemente completa y detallada, que permita la implementación del estándar internacional ISO/IEC/IEEE 42010:2011 para desarrollar descripciones de arquitectura como forma de documentación arquitectónica, que incluya el uso de Lenguajes de Descripción de Arquitectura en el proceso de documentación, que cuente con técnicas de arquitectura específicas que permitan realizar las tareas de documentación, y que sea compatible con diferentes enfoques de documentación, como el de Soni et al. (1995), el de Kruchten (1995), y el de Clements et al. (2003).

Trabajos relacionados

Los trabajos relacionados en torno a la documentación de arquitecturas de software, son otros enfoques de documentación que han surgido en las últimas décadas, sin embargo, ninguno está fuertemente relacionado con los Lenguajes de Descripción de Arquitectura. Algunos de los enfoques de documentación más significativos son el enfoque de Soni et al. (1995), el enfoque 4+1 (Kruchten 1995), el enfoque del *Rational Unified Process* (RUP) (Kruchten 2000), y el enfoque "*Views and Beyond*" (Clements et al. 2003) creado por el *Software Engineering Institute* (SEI).

En cuanto a los Lenguajes de Descripción de Arquitectura, existen múltiples tutoriales, manuales de referencia, artículos, libros y otras fuentes de información, que guían en el modelado, especificación y análisis arquitectónico. Sin embargo, tales fuentes de información están limitadas en cuanto a la incorporación de dichos Lenguajes de Descripción de Arquitectura en el proceso de desarrollo de descripciones de arquitectura considerando un estándar internacional de documentación, como el ISO/IEC/IEEE 42010:2011.

En cuanto al estándar ISO/IEC/IEEE 42010:2011, existen documentos, algunos ejemplos, repositorios, e información adicional para su uso, sin embargo, no existe una guía detallada que permita implementarlo en conjunto con Lenguajes de Descripción de Arquitectura específicos. Por otra parte, en el repositorio del estándar, existe un documento con “formas” poco refinadas, derivadas directamente del estándar, las cuales únicamente replican el contenido del estándar en secciones, no presentan un formato adecuado para generar productos de trabajo de software finales, ni las consideraciones ni técnicas necesarias para elaborar un documento arquitectónico útil.

Hipótesis

Contar con una guía de documentación arquitectónica, basada en un estándar internacional de documentación (ISO/IEC/IEEE 42010:2011), que incluya el uso de Lenguajes de Descripción de Arquitectura, y que cuente con técnicas de arquitectura específicas y plantillas de documentación, permitirá realizar las tareas de documentación, y permitirá elaborar descripciones de arquitectura útiles, precisas, y entendibles.

Propuesta de solución

Objetivo general

El objetivo general de este trabajo es desarrollar y presentar una guía que permita realizar las tareas de documentación arquitectónica, y que permita elaborar descripciones de arquitectura útiles, precisas, y entendibles.

Objetivos particulares

Los objetivos particulares de este trabajo representan las características específicas de la guía a desarrollar y son:

1. La guía estará basada en las especificaciones del estándar internacional ISO/IEC/IEEE 42010:2011.
2. La guía presentará un conjunto de actividades y técnicas detalladas que permitan realizar la documentación de arquitecturas de software a través de descripciones de arquitectura.
3. La guía contará con las consideraciones necesarias para el uso y selección de Lenguajes de Descripción de Arquitectura para realizar modelos y especificaciones arquitectónicas.
4. La guía incluirá plantillas detalladas para permitir elaborar la documentación arquitectónica generando productos de trabajo de software de manera sencilla.

5. La guía incluirá ejemplos prácticos que apoyen en la documentación arquitectónica, en especial, ejemplos que permitan el modelado y la especificación arquitectónica mediante el uso de LDAs.
6. La guía incluirá consejos y recomendaciones para realizar la documentación de arquitecturas de software.

Justificación y contribución

La guía estará basada en el estándar ISO/IEC/IEEE 42010:2011 debido a que es el estándar internacional para documentar arquitecturas de software más reciente (generado a partir de las revisiones y mejoras del ANSI/IEEE 1471:2000) de dos de las organizaciones que tienen mayor reconocimiento internacional: la Organización Internacional de Normalización (ISO), y la Comisión Electrotécnica Internacional (IEC). En comparación con su antecesor, el estándar ISO/IEC/IEEE 42010:2011 considera conceptos adicionales, tales como los tipos de modelos (ver sección 3.9), que proporcionan grandes ventajas para elaborar y validar la documentación arquitectónica.

El estándar ISO/IEC/IEEE 42010:2011 especifica la manera en la que las descripciones de arquitectura son organizadas y expresadas, por lo que seguir este estándar enfocado en la documentación arquitectónica permite guiar la creación de descripciones de arquitectura, y al mismo tiempo es flexible al permitir incluir métodos, notaciones, técnicas, consejos, prácticas, y otros elementos específicos de arquitectura según sea conveniente para un proyecto u organización. Además, al ser un estándar internacional, permite sobrepasar algunas barreras técnicas en el ámbito de la Ingeniería de Software.

Por otra parte, el uso del estándar ISO/IEC/IEEE 42010:2011 tiene como consecuencia prevista, que una descripción de arquitectura sea bien entendida, lo que debe contribuir en una mejor arquitectura, y por lo tanto, en un mejor sistema. Además, tener una buena arquitectura es crítico para el éxito de un sistema, y el estándar refleja las mejores prácticas para describir arquitecturas de sistemas de software.

El uso del estándar ISO/IEC/IEEE 42010:2011 implica que la documentación de una arquitectura se haga a través de la elaboración de descripciones de arquitectura, las cuales son una forma detallada de documentar arquitecturas de software. Este tipo de productos de trabajo son un medio útil y completo para documentar arquitecturas, ya que sirven como base para actividades de diseño y desarrollo de sistemas, para analizar y evaluar alternativas de implementación de una arquitectura, y como entrada a herramientas automatizadas para simulación, generación y análisis de sistemas. Además, las descripciones de arquitectura son elementos de documentación de desarrollo y mantenimiento, permiten documentar aspectos esenciales de un sistema, permiten especificar un grupo de sistemas que comparten características en común, facilitan la comunicación entre interesados, permiten compartir lecciones aprendidas y reusar conocimiento arquitectónico, y permiten entrenar y educar a los interesados de un sistema (ISO/IEC/IEEE 42010:2011).

Como parte de una descripción de arquitectura se deben generar modelos de arquitectura, los cuales son sumamente importantes debido a que son representaciones de las características fundamentales de la arquitectura, pueden proporcionar información clara sobre la misma, y son usados para diversos fines en el ciclo de vida de un sistema.

En esta guía, se propone el uso de Lenguajes de Descripción de Arquitectura como notación para elaborar modelos y especificaciones arquitectónicas debido a que su uso proporciona considerables ventajas: no son lenguajes simplemente de modelado, ya que permiten especificar la arquitectura de un sistema de manera formal mediante bases matemáticas; están destinados a ser entendibles tanto por máquinas como por humanos; permiten describir un sistema al nivel de abstracción más alto posible; permiten analizar aspectos arquitectónicos fundamentales; pueden soportar la generación automática de elementos de software; permiten presentar con mayor detalle los modelos arquitectónicos; reducen la ambigüedad; y algunos de ellos permiten extender ciertas propiedades para poder modelar características arquitectónicas. Algunas de las características de los Lenguajes de Descripción de Arquitectura antes mencionadas, no son soportadas por notaciones menos formales o de propósito general como UML.

Por otra parte, debido a la gran cantidad de Lenguajes de Descripción de Arquitectura que existen, la guía contará con las consideraciones necesarias para el uso y selección de Lenguajes de Descripción de Arquitectura específicos, lo cual apoyará en las actividades de modelado y especificación arquitectónica.

Como resultado de la documentación de una arquitectura, se generan productos de trabajo de software que pueden ser usados durante distintas fases en el ciclo de vida de un sistema, por lo que es importante contar con herramientas que faciliten dicha tarea. Por lo anterior, se propone incluir en la guía un conjunto de plantillas que apoyen en la generación de la documentación arquitectónica.

Debido a que la documentación de una arquitectura involucra diversas actividades, tales como la identificación y desarrollo de vistas arquitectónicas, en la guía, se presentarán dichas actividades detalladamente, incluyendo técnicas de arquitectura que permitan su realización.

Adicionalmente, la guía será lo suficientemente abierta como para permitir utilizar distintos enfoques de documentación en conjunto con la guía, tales como el método 4+1 (Kruchten 1995), o el enfoque de Soni et al. (1995).

Finalmente, es importante mencionar que no existe una guía de documentación con las características antes mencionadas, por lo que la contribución de este trabajo es el desarrollo de dicha guía.

Estructura del trabajo

El trabajo se divide en dos partes principales: el marco teórico, y la guía para documentación de arquitecturas de software. La primera parte (Marco teórico), incluye los conceptos, definiciones y teoría sobre la documentación de arquitecturas de software. Se compone de cuatro capítulos:

- El capítulo 1, incluye el marco teórico principal de la arquitectura de software.
- El capítulo 2, incluye el marco teórico sobre la documentación arquitectónica y las descripciones de arquitectura.
- El capítulo 3, incluye los conceptos fundamentales en torno a las descripciones de arquitectura, tales como intereses, interesados, vistas y puntos de vista arquitectónicos.

- El capítulo 4, incluye el marco teórico sobre los Lenguajes de Descripción de Arquitectura.

La segunda parte (Guía para documentar arquitecturas de software utilizando Lenguajes de Arquitectura), incluye los pasos a seguir para realizar descripciones de arquitectura según lo establecido por el estándar ISO/IEC/IEEE 42010:2011 y utilizando Lenguajes de Descripción de Arquitectura. Se compone de 11 capítulos:

- El capítulo 5, presenta las consideraciones generales de la guía, tales como su alcance y audiencia.
- El capítulo 6, presenta las consideraciones necesarias para identificar y documentar la información general de una descripción de arquitectura.
- El capítulo 7, presenta una guía para identificar y documentar los interesados e intereses principales de la arquitectura de un sistema.
- El capítulo 8, presenta criterios para seleccionar Lenguajes de Descripción de Arquitectura que pueden utilizarse para elaborar una descripción de arquitectura.
- El capítulo 9, presenta las consideraciones necesarias para identificar y documentar los puntos de vista arquitectónicos de un sistema.
- El capítulo 10, presenta una guía para desarrollar y documentar las vistas arquitectónicas de un sistema.
- El capítulo 11, presenta las consideraciones necesarias para realizar modelos y especificaciones arquitectónicas utilizando Lenguajes de Descripción de Arquitectura específicos.
- El capítulo 12, presenta las consideraciones necesarias para documentar las relaciones entre los elementos de una descripción de arquitectura.
- El capítulo 13, presenta las consideraciones necesarias para documentar el razonamiento y las decisiones arquitectónicas de un sistema.
- El capítulo 14, presenta las consideraciones necesarias para validar y dar mantenimiento a una descripción de arquitectura.
- El capítulo 15 presenta los resultados de la evaluación de la guía.

Adicionalmente, en los apéndices de este trabajo se presenta un catálogo de interesados, uno de atributos de calidad, uno de puntos de vista, un conjunto de plantillas para documentar arquitecturas de software mediante descripciones de arquitectura, y ejemplos de descripciones de arquitectura.

PARTE 1 MARCO TEÓRICO

Esta parte tiene como objetivo presentar los conceptos esenciales sobre documentación de arquitecturas de software que permitan al lector entender y utilizar puntualmente la guía para documentar arquitecturas de software utilizando Lenguajes de Descripción de Arquitectura presentada en la segunda parte de este trabajo.

Esta primera parte incluye los siguientes capítulos:

CAPÍTULO 1 Contexto de la Arquitectura de Software. Presenta conceptos fundamentales sobre la Arquitectura de Software tales como su definición, importancia y los diferentes contextos en que se desarrolla.

CAPÍTULO 2 Documentación de arquitecturas de software y descripciones de arquitectura. Presenta conceptos esenciales en torno a la documentación de arquitecturas de software, e incluye los fundamentos sobre las descripciones de arquitectura con base en el estándar ISO/IEC/IEEE 42010:2011.

CAPÍTULO 3 Marco conceptual de las descripciones de arquitectura. Presenta el marco conceptual en torno a las descripciones de arquitectura con base en el estándar ISO/IEC/IEEE 42010:2011. Incluye conceptos tales como intereses, vistas y puntos de vista arquitectónicos.

CAPÍTULO 4 Lenguajes de Descripción de Arquitectura. Presenta el marco teórico sobre los Lenguajes de Descripción de Arquitectura, incluyendo su definición, características principales, y ventajas y desventajas.

Capítulo 1. Contexto de la Arquitectura de Software

La Arquitectura de Software ha emergido como una importante subdisciplina de la Ingeniería de Software, particularmente en el desarrollo de grandes sistemas (Clements et al. 2003). En el amplio marco teórico de la Arquitectura de Software, resulta fundamental saber qué es una arquitectura de software y el proceso que implica, ya que éste incluye distintas actividades entre las que se encuentra la documentación arquitectónica, tema principal de este trabajo.

1.1. ¿Qué es una arquitectura de software?

No existe una definición universal de lo que es una arquitectura de software, existen muchas que han surgido en las últimas décadas, por lo que a continuación, se presentan algunas de las más citadas con el fin de distinguir las características fundamentales de una arquitectura de software:

1. *“Una especificación abstracta de un sistema que consiste principalmente de componentes funcionales descritos en términos de sus comportamientos e interfaces y conexiones entre componentes”* (Hayes-Roth 1994).
2. *“La estructura de los componentes de un programa/sistema, sus interrelaciones, y principios y directrices que gobiernen su diseño y evolución en el tiempo”* (Garlan 1995).
3. *“Un conjunto de decisiones significativas acerca de la organización de un sistema de software, la selección de los elementos estructurales que compondrán el sistema y sus interfaces, junto con su comportamiento, tal y como se especifican en las colaboraciones entre estos elementos, la composición de los elementos estructurales y del comportamiento en subsistemas progresivamente más grandes, y el estilo estructural que guía esta organización: los elementos y sus interfaces, sus colaboraciones, y su composición”* (Jacobson, Booch y Rumbaugh 1999).
4. *“Conceptos o propiedades fundamentales de un sistema en su ambiente, expresados en sus elementos, relaciones, y en los principios de su diseño y evolución”* (ISO/IEC/IEEE 42010:2011).
5. *“Es el conjunto de estructuras necesarias para razonar sobre un sistema, que comprende elementos de software, las relaciones entre ellos y propiedades de ambos”* (Bass, Clements y Kazman 2012). Esta es una definición muy citada en la Ingeniería de Software.

Aunque existen, además de las presentadas, otras definiciones de arquitectura de software, la mayoría coinciden en que una arquitectura de software está compuesta de elementos y las conexiones entre ellos, y que una arquitectura de software es el conjunto de los conceptos y propiedades primordiales de un sistema de software en su entorno, incluyendo su comportamiento, y los principios y directrices de su evolución y diseño.

1.2. Importancia de la Arquitectura de Software

En la actualidad, los sistemas de software pueden implicar cierta complejidad o demandar características precisas, lo que ha producido nuevos desafíos para las organizaciones que

crean y utilizan dichos sistemas. Cuando se realizan actividades sobre este tipo de sistemas de software, como diseño o análisis, puede resultar muy útil poder visualizarlos como una abstracción de alto nivel, que permita identificar sus principales componentes y características de manera sencilla. Es por lo anterior que actualmente son aplicados con mayor frecuencia conceptos, principios y procedimientos de Arquitectura de Software.

Por otra parte se puede decir que todo sistema, ya sea grande o pequeño, tiene una arquitectura, ya que tiene un comportamiento y está compuesto por piezas que están relacionadas entre sí. Tal arquitectura, es una propiedad intrínseca y fundamental de todo sistema (Rozansky y Woods 2005).

Existen distintas razones por las que la arquitectura de un sistema de software es importante. A continuación, se presentan las siguientes tomadas de "*Software Architecture in Practice*" (Bass, Clements y Kazman 2012):

- La arquitectura inhibe o habilita los atributos de calidad² (*Quality Attributes*) del sistema, ya que si un sistema será capaz de presentar sus atributos deseados o requeridos, esto es fundamentalmente determinado por su arquitectura.
- Las decisiones plasmadas en la arquitectura permiten razonar sobre el sistema, y administrar cambios a medida que el sistema evoluciona.
- El análisis de una arquitectura permite la predicción temprana de cualidades del sistema.
- La arquitectura documentada mejora la comunicación entre los interesados del sistema (*stakeholders*), ya que representa una abstracción del sistema que puede ser entendida por sus interesados en un lenguaje común.
- La arquitectura es un portador de las decisiones más tempranas y, por lo tanto, de las decisiones de diseño más fundamentales de un sistema. Una arquitectura puede verse como un conjunto de decisiones, las cuales pueden tener mucho peso en actividades de desarrollo, implementación y mantenimiento del sistema.
- La arquitectura define un conjunto de restricciones para la implementación del sistema, debido a que el sistema debe implementarse con base en un conjunto de elementos prescritos en su arquitectura.
- La arquitectura establece la estructura de una organización, o viceversa. La arquitectura prescribe la estructura del sistema a desarrollar, pero la estructura se incrusta en la estructura del proyecto, y algunas veces, en la estructura de la organización.
- La arquitectura puede proporcionar las bases para un prototipo evolutivo. Una vez que la arquitectura ha sido definida, puede ser analizada y usada como prototipo de un sistema "esqueleto"³.
- La arquitectura es el artefacto clave que permite al arquitecto y al administrador del proyecto razonar sobre costo y calendarización. La arquitectura permite entender las estructuras que conforman el sistema, y así, estimar costos y asignar recursos de trabajo.

² Un atributo de calidad es una propiedad medible y comprobable de un sistema, que es usada para indicar qué tan bien el sistema satisface las necesidades de sus interesados (Bass, Clements y Kazman 2012).

³ Sistema que cuenta con gran parte de su estructura, antes de que se cree gran parte de su funcionalidad (Bass, Clements y Kazman 2012).

- La arquitectura puede ser creada como un modelo transferible y reusable que forme el corazón de una línea de productos. Una línea de productos es un conjunto de sistemas de software que son construidos usando el mismo conjunto de activos reutilizables.
- El desarrollo de un sistema basado en la arquitectura, enfoca la atención en el ensamblado de componentes, no simplemente en su creación.
- Al restringir las alternativas de diseño, la arquitectura canaliza la creatividad de desarrolladores, reduciendo la complejidad del diseño del sistema. El uso de patrones arquitectónicos también ayuda a reducir la complejidad del sistema.
- La arquitectura puede ser el fundamento para entrenar a un nuevo miembro del equipo de trabajo u otro interesado en la arquitectura del sistema.

Debido a su importancia, se puede considerar a la arquitectura parte esencial en el éxito de un sistema de software, sobre todo si sus conceptos y principios son aplicados correctamente.

1.3. Proceso de arquitectura en el ciclo de vida del software

En el contexto de la Ingeniería de Software, un proceso es un conjunto de actividades y resultados asociados que producen un producto de software (Sommerville 2005). Durante el ciclo de vida de desarrollo de un sistema de software, se realizan distintos procesos, que pueden ser guiados por un modelo de procesos⁴.

Un modelo de procesos puede incluir distintas actividades de arquitectura. El proceso que engloba estas actividades, es llamado “*architecting*”, o proceso de arquitectura. Este proceso toma lugar en el contexto de una organización y/o proyecto (ISO/IEC/IEEE 42010:2011) e incluye distintas actividades, entre las que se encuentran:

- Definición, que consiste en crear o seleccionar un diseño de la arquitectura que satisfaga los requerimientos del sistema.
- Descripción, que es un conjunto de prácticas y técnicas para representar, expresar, y comunicar arquitecturas de software. El resultado de aplicar tales prácticas, es un producto de trabajo que expresa una arquitectura de software.
- Evaluación, que consiste en asegurar que el sistema construido a partir de la arquitectura, satisface las necesidades de sus interesados (Bass, Clements y Kazman 2012). Permite comprobar que se realizaron las decisiones de arquitectura correctas durante la definición de arquitectura, para que dicha arquitectura cubra las necesidades de los interesados del sistema.
- Documentación, que implica la creación de un documento sobre la arquitectura de un sistema. Debido a que la documentación es tema fundamental en este trabajo, es descrita con mayor detalle en el capítulo 2.
- Comunicación: Actividad para dar a conocer información sobre la arquitectura del sistema entre los interesados de la misma.
- Certificación: Actividades para cerciorar la buena implementación de la arquitectura.

⁴ Un modelo de procesos de software es una descripción simplificada del proceso de software. Un modelo puede incluir actividades que son parte de los procesos y productos de software, y el papel de las personas involucradas en la ingeniería de software (Sommerville 2005).

- **Mantenimiento:** Actividades realizadas sobre la arquitectura para corregir defectos y mejorarla.

El proceso de arquitectura, con todas las actividades que implica, es realizado durante todo del ciclo de vida de desarrollo de un sistema, no solo en una etapa, por lo que influencia potencialmente otros procesos como el diseño, desarrollo, y mantenimiento del sistema.

A continuación, se describe como el proceso de arquitectura influencia, y está relacionado con otros procesos y actividades durante el ciclo de vida de desarrollo de un sistema, y como se desarrolla en distintos modelos de procesos.

1.3.1. Arquitectura en distintos procesos durante el desarrollo de software

1.3.1.1. Arquitectura y requerimientos

Dentro de las primeras actividades que se realizan durante el desarrollo de un software, se encuentra la obtención y análisis de requerimientos, en las que se determina el dominio del sistema, los servicios que debe proporcionar, el rendimiento requerido, las restricciones, etc. (Sommerville 2005).

La arquitectura de un sistema se genera para satisfacer los requerimientos del sistema, sin embargo, existen requerimientos que tienen un efecto profundo en la arquitectura, es decir, la arquitectura puede ser dramáticamente diferente en su ausencia. Estos requerimientos son llamados “requerimientos arquitectónicamente significativos” (Bass, Clements y Kazman 2012). Para diseñar una arquitectura exitosa es indispensable conocer los requerimientos arquitectónicamente significativos, que a veces toman forma de atributos de calidad, o son metas de negocio. Es importante conocer tales requerimientos ya que pueden afectar las decisiones tempranas de diseño de la arquitectura.

El análisis de requerimientos proporciona el contexto para la definición de una arquitectura, ya que define el alcance y las propiedades de calidad y funcionalidad deseadas por el sistema. La definición de una arquitectura a veces da a conocer requerimientos inconsistentes y faltantes, y ayuda a los interesados a entender costos y la complejidad de satisfacer sus intereses (Rozansky y Woods 2005).

1.3.1.2. Arquitectura y diseño

El diseño de software se ubica en el área técnica de la Ingeniería de Software y comienza una vez que se han analizado y modelado los requerimientos, los cuales proporcionan información necesaria para crear un modelo de diseño del sistema, como el de datos, interfaz o de arquitectura (Pressman 2010). El diseño de la arquitectura define, entre otras cosas, los elementos estructurales de software, su relación y comportamiento, todo con el fin de satisfacer los requerimientos del sistema. Por otra parte, el diseño de la arquitectura es un diseño de alto nivel de abstracción del sistema, que sirve como base para diseños más específicos.

1.3.1.3. *Arquitectura e implementación*

Cuando se realiza la definición de una arquitectura que satisface un conjunto aceptable de requerimientos, se puede planear la construcción del sistema (Rozansky y Woods 2005).

La arquitectura sirve como “arquetipo” para la implementación de un sistema, ya que define las estructuras a ser implementadas, así como las cualidades con que debe contar el sistema y las restricciones a considerar. Además, la arquitectura permite definir equipos de trabajo y actividades de implementación.

La arquitectura puede resultar muy útil para la implementación de un sistema, sobre todo si está bien documentada y puede ser entendida y comunicada detalladamente y sin ambigüedades.

1.3.1.4. *Arquitectura y pruebas*

La arquitectura representa un papel importante en las pruebas de un sistema, sobre todo si se cuenta con la documentación suficiente, ya que propicia la realización de pruebas menos costosas y más efectivas. A continuación, se menciona la importancia de la arquitectura en distintos tipos de pruebas:

- Las pruebas unitarias se realizan sobre piezas específicas de software, estas unidades o módulos, pueden corresponder a un elemento de la arquitectura, la cual puede definir también sus responsabilidades y los requerimientos que satisfacen. Por esto, una arquitectura bien definida y documentada es de gran utilidad en la realización de pruebas unitarias.
- Debido a que la arquitectura establece las interfaces⁵ de los elementos que conforman el sistema, una arquitectura bien documentada puede ser útil para realizar pruebas de integración, ya que puede incluir la interacción entre los elementos del sistema.
- La documentación de una arquitectura permite conocer los requerimientos de cada pieza del sistema, que son sumamente útiles para realizar las pruebas de caja negra.
- Una arquitectura bien documentada, puede informar sobre los riesgos del sistema, lo que es útil para realizar pruebas basadas en riesgos.

1.3.2. **Arquitectura en algunos modelos de procesos**

1.3.2.1. *Arquitectura en el modelo lineal secuencial*

Este modelo, también llamado “modelo en cascada”, es visto como una secuencia lineal de tareas, donde cada tarea utiliza las salidas de la tarea previa como entradas (Rozansky y Woods 2005). El proceso inicia con la especificación requerimientos, seguido por el diseño, la implementación, la integración, las pruebas, la instalación, y todo seguido por el mantenimiento. El proceso de arquitectura puede ser integrado dentro de la etapa de diseño.

⁵ Las interfaces comprenden cada interacción de un elemento de la arquitectura con su entorno (Bachmann et al. 2002).

1.3.2.2. *Arquitectura en modelos iterativos*

En estos modelos se realiza una serie de pequeños ciclos llamados iteraciones, en los que se reproduce un ciclo de vida en cascada a menor escala. En cada una de las iteraciones debe ser entregado algo útil y funcional. El objetivo del enfoque iterativo es reducir riesgos por medio de entregas tempranas de funcionalidades parciales del sistema. En este tipo de modelos, las actividades de arquitectura, como la definición o concepción, podrían formar parte de las fases de análisis y diseño de cada iteración.

1.3.2.3. *Arquitectura en el modelo incremental*

Este modelo combina elementos del modelo en cascada con la construcción de prototipos⁶. Aplica secuencias lineales de actividades de forma escalonada que producen un incremento del software de acuerdo a un prototipo. Durante cada incremento, el sistema se evalúa con respecto al desarrollo de versiones futuras.

En este modelo hay un diseño inicial de la arquitectura del sistema completo, seguido de sucesivos incrementos funcionales. Debido que el diseño de la arquitectura se realiza en la etapa inicial, es necesario conocer los requerimientos de manera temprana.

1.3.2.4. *Arquitectura en métodos ágiles*

Estas metodologías son iterativas e incrementales, y se enfocan en rápidas y continuas entregas de software a usuarios finales, fomentando la constante interacción entre cliente y desarrolladores, e intentando minimizar gastos generales del proceso de desarrollo.

Debido a la naturaleza de estos métodos, comúnmente no se realizan muchas actividades de arquitectura, ya que en ocasiones son vistas como procedimientos burocráticos y se tiende a dar mayor importancia a la implementación. Sin embargo, como se mencionó anteriormente en este trabajo, todo sistema de software tiene una arquitectura, por lo que en algún momento del ciclo de vida de desarrollo debe considerarse. Es por esto que existen algunas técnicas y prácticas de arquitectura que pueden ser implementadas en metodologías ágiles. Algunos ejemplos de éstas son:

- YANGI (*you ain't gonna need it*), es un principio que dicta que solo se escriba para el lector, es decir, se requiere entender quién va a leer y usar la documentación, y si no hay audiencia, no hay necesidad de producir documentación (Bass, Clements y Kazman 2012). Esta técnica puede ser implementada en actividades de documentación de la arquitectura.
- El *Architecture Tradeoff Analysis Method* (ATAM), es un método de evaluación de la arquitectura en el que solo se consideran los intereses más importantes del sistema (Bass, Clements y Kazman 2012).

⁶ Un prototipo es un modelo de cierta funcionalidad del sistema con base en los requerimientos del mismo.

1.4. Arquitectura en el contexto de negocio

Cuando se construye un sistema, se deben considerar las metas de negocio de la organización o proyecto, ya que los sistemas a veces son creados para satisfacer los objetivos o metas de negocio de una o más organizaciones. Algunos de estos objetivos tienen una influencia profunda en la arquitectura del sistema, y son manifestados en forma de atributos de calidad. De hecho, cada atributo de calidad debe originarse a partir de un propósito más alto que puede ser descrito en términos de valor agregado (Bass, Clements y Kazman 2012). Es por lo anterior, que en el proceso de arquitectura debe entenderse como está establecida la organización y cuáles son sus metas de negocio.

Por otra parte, una organización puede desear hacer inversiones de negocio a largo plazo en una infraestructura para seguir metas estratégicas, y puede ver los sistemas propuestos como uno de los medios de financiamiento y extensión de esa infraestructura. Además, la estructura organizacional puede dar forma a una arquitectura de software, y viceversa (Bass, Clements y Kazman 2012).

1.5. Arquitectura en el contexto profesional

En el contexto profesional de la Arquitectura de Software, lo más importante es el papel que desempeña el arquitecto de software, quién tiene más relación con la arquitectura de un sistema de software.

El arquitecto de software es, en general, el responsable de desarrollar la arquitectura y su documentación. Sus principales intereses son negociar y realizar compensaciones entre los requerimientos del sistema y realizar aproximaciones de la arquitectura que plasmen las decisiones de diseño. Por otra parte, un arquitecto debe proporcionar evidencia de que la arquitectura satisface los requerimientos del sistema (Bass, Clements y Kazman 2012).

Un arquitecto requiere tener habilidades técnicas, pero además, requiere habilidades diplomáticas, de negociación y de comunicación. Esto debido a que entre sus actividades, no solo se encuentra la producción de un diseño de la arquitectura, si no que requiere comunicar claramente sus ideas, analizar y negociar aspectos sobre la arquitectura, y tener conocimientos actualizados sobre temas tecnológicos como patrones, plataformas tecnológicas, y estándares de servicios web.

Capítulo 2. Documentación de arquitecturas de software y descripciones de arquitectura

Como se mencionó en el capítulo 1, una de las actividades del proceso de arquitectura es la documentación, que puede aportar grandes beneficios a un sistema, proyecto u organización. La documentación de una arquitectura de software está relacionada con otras actividades, puede ser realizada siguiendo estándares y aplicando diversas técnicas y métodos, e implica la elaboración distintos productos de trabajo de software, como es el caso de las descripciones de arquitectura.

2.1. Documentación de arquitecturas de software

2.1.1. ¿Qué es documentar una arquitectura de software?

La documentación de una arquitectura de software, es la tarea de producir un documento de arquitectura de un sistema (Clements et al. 2003).

Comúnmente, la documentación se asocia con otros conceptos, como la especificación, representación o descripción arquitectónica. En ocasiones, estos términos pueden llegar a resultar confusos debido a que se refieren a cuestiones similares y tienen cierta relación, por lo que a modo de aclaración, se presenta una pequeña definición de los mismos:

- La “especificación” de una arquitectura, comúnmente tiende a referirse a la actividad de traducir una arquitectura a un lenguaje formal. Es un término utilizado para una actividad concreta en la que se utiliza una notación formal.
- La “representación”, es el modelado o abstracción de la arquitectura de un sistema. Éste es un término más general que el anterior, ya que puede realizarse utilizando distintas técnicas y notaciones.
- La “descripción” de una arquitectura, define las prácticas, técnicas y tipos de representaciones usadas para registrar una arquitectura de software. La actividad de describir una arquitectura es más general que las anteriores, ya que puede incluir la representación, modelado y especificación.

La “documentación” de una arquitectura es un término más amplio que todos los anteriores, ya que al realizar una actividad de documentación, se puede especificar, representar y describir. Debido a esto, una documentación puede ser formal o no, puede contener modelos o no, y los documentos pueden describir o especificar (Clements et al. 2003).

2.1.2. ¿Por qué es importante documentar una arquitectura de software?

Al realizar actividades adecuadas de documentación de una arquitectura, se generan documentos, que pueden tener muchos usos y ventajas, entre los que se encuentran (Clements et al. 2003):

- Sirven como medio de educación, para introducir personas al sistema, tales como nuevos miembros del equipo de trabajo o analistas externos.

- Sirven como principal medio de comunicación entre los interesados en la arquitectura, tales como arquitectos, analistas, implementadores, personal de pruebas y administradores.
- Sirven como base para el análisis y construcción del sistema.

También es importante mencionar que una arquitectura bien documentada, hace más útil la información que incluye.

A pesar de que existen muchas razones para realizar actividades de documentación, en la actualidad no siempre se realizan adecuadamente, se hacen porque “se tienen que hacer” o por contrato o demanda del cliente, pero nada obliga a hacer una documentación de calidad. Sin embargo, la documentación debe verse como algo esencial para producir un producto de software de alta calidad (Bass, Clements y Kazman 2012).

2.1.3. Características de los documentos de arquitectura

Para que sea útil y de calidad, la documentación arquitectónica de un sistema debe contar ciertas características. A continuación se presentan algunas de las más importantes:

- Debe ser descrita detalladamente, asegurando que la información más importante sea puntualizada.
- No debe ser ambigua, es decir, no debe poder interpretarse o entenderse de diversas maneras, únicamente de la forma adecuada.
- Debe estar organizada de tal forma que permita encontrar fácilmente la información necesaria (Bass, Clements y Kazman 2012). Esto permite no tener que recurrir siempre al arquitecto para su explicación.
- Debe ser suficientemente transparente y accesible para ser entendida rápidamente por los interesados.
- Debe ser suficientemente concreta para guiar correctamente la construcción de cada parte del sistema.
- Debe contar con suficiente información para ser base del análisis del sistema.
- Debe ser preceptiva y descriptiva, para algunas audiencias prescribe que debe ser hecho, para otras audiencias, describe lo que está hecho, haciendo un recuento de las decisiones ya tomadas sobre el diseño del sistema.
- Debe soportar todas las necesidades relevantes del sistema. (Bass, Clements y Kazman 2012).

2.1.4. Algunos enfoques de documentación

Existen diferentes enfoques que pueden ser utilizados para documentar una arquitectura de software, algunos de ellos son los siguientes:

- Enfoque de Soni et al. (1995): este enfoque define vistas en las siguientes categorías: conceptual, que representa la arquitectura en términos de componentes y conectores; modular, que representa la descomposición funcional; de ejecución, que incluye elementos de tiempo de ejecución; y de código, que mapea archivos de código.
- Enfoque 4+1 (Kruchten 1995): es un modelo basado en el uso de múltiples vistas utilizado para describir la arquitectura de un sistema. Las vistas describen el

sistema desde el punto de vista de diferentes interesados. Las cuatro vistas del modelo son: lógica, de desarrollo, de proceso y física. Además los casos de uso o escenarios seleccionados son usados para ilustrar la arquitectura, lo que toma el papel de la vista "+1", de ahí el nombre del enfoque.

- Enfoque del *Rational Unified Process* (RUP) (Kruchten 2000): está basado en el enfoque "4+1" y utiliza modelos creados con UML para representar los siguientes tipos de vistas: lógica, de procesos, de despliegue y física.
- Enfoque "*Views and Beyond*" (Clements et al. 2003): creado por el *Software Engineering Institute* (SEI), y cuyo enfoque es documentar las vistas relevantes de una arquitectura y documentar la información que se aplique a múltiples vistas. Las vistas, que son representaciones de varias estructuras de un sistema, pueden ser de tres tipos: modulares, de componentes y conectores, y de distribución.

Adicionalmente, existen algunos estándares de documentación que pueden ser usados para documentar una arquitectura, tal es el caso del estándar ANSI/IEEE 1471:2000 y su versión actual, el estándar ISO/IEC/IEEE 42010:2011, que será presentado con mayor detalle en futuras secciones en este trabajo.

2.2. Descripciones de arquitectura

Existen distintas técnicas y enfoques para generar documentación arquitectónica, sin embargo, lo importante es que ésta sea de calidad y útil para los interesados del sistema, tal como se mencionó en la sección anterior.

El tipo y tamaño de la documentación de una arquitectura varía en cada proyecto u organización, sin embargo, resulta útil contar con un estándar o norma que guíe la creación de la documentación de la arquitectura del sistema, especificando el tipo y contenido de los documentos con base en fundamentos conceptuales consistentes.

2.2.1. El estándar ISO/IEC/IEEE 42010:2011 "Systems and software engineering, Architecture description"

El estándar ISO/IEC/IEEE 42010:2011 "*Systems and software engineering, Architecture description*", al que se ha hecho referencia en secciones anteriores en este trabajo, es un estándar internacional que prescribe la forma en que debe ser realizada la documentación de una arquitectura de software a través de la generación de descripciones de arquitectura.

El estándar ISO/IEC/IEEE 42010:2011 proporciona una ontología principal para la descripción de arquitecturas, y especifica la manera en que las descripciones de arquitectura de un sistema son organizadas y expresadas.

El estándar fue publicado en 2011, y es resultado de una revisión al estándar ANSI/IEEE 1471:2000 (*IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*).

El estándar ISO/IEC/IEEE 42010:2011 también especifica las propiedades con que deben contar los “*frameworks de arquitectura*”⁷ y los Lenguajes de Descripción de Arquitectura (de los que se hablará a detalle en los siguientes capítulos), con el fin de aportar una contribución útil al desarrollo y uso de las descripciones de arquitectura.

Los conceptos relacionados con el estándar ISO/IEC/IEEE 42010:2011, tales como vistas o puntos de vista arquitectónicos, son descritos en secciones futuras (ver capítulo 3).

2.2.2. ¿Que son las descripciones de arquitectura?

El término “descripción de arquitectura” ampliamente utilizado en este trabajo, es establecido por el estándar ISO/IEC/IEEE 42010:2011, que define una descripción de arquitectura como un producto de trabajo usado para expresar una arquitectura de un sistema de interés.

Las descripciones de arquitectura son resultado de la ejecución de diversas actividades durante proceso de arquitectura en el ciclo de vida de un sistema (ISO/IEC/IEEE 42010:2011). Aun cuando las descripciones de arquitectura resulten de una sola actividad, su contenido es el resultado de múltiples actividades como análisis, modelado o especificación.

2.2.3. ¿Porque usar descripciones de arquitectura como forma de documentación?

Las descripciones de arquitectura, con las características definidas por el estándar ISO/IEC/IEEE 42010:2011, tienen varios usos para una gran cantidad de interesados del sistema ya que se pueden emplear:

- Como base para analizar y evaluar alternativas de implementación de una arquitectura.
- Como documentación de desarrollo y mantenimiento.
- Para documentar aspectos esenciales de un sistema tales como:
 - Ambiente y uso.
 - Principios, supuestos y restricciones para guiar futuros cambios.
 - Puntos de flexibilidad, o limitaciones del sistema con respecto a futuros cambios.
 - Decisiones de arquitectura, sus razonamientos e implicaciones.
- Como entrada a herramientas automatizadas para simulación, generación de sistemas y análisis.
- Para especificar un grupo de sistemas que comparten características en común (como estilos arquitectónicos y líneas de productos).
- Como medio de comunicación entre las partes involucradas en el desarrollo, producción, implementación, operación y mantenimiento del sistema.
- Como base para la preparación de documentos de adquisición (tales como solicitudes de propuesta y declaraciones de trabajo).
- Como medio de comunicación entre clientes, adquiridores, proveedores y desarrolladores, como parte de negociaciones de contrato.

⁷ Convenciones, principios y prácticas para la descripción de arquitecturas establecidas en un dominio específico de la aplicación y/o comunidad de interesados.

- Para documentar las características, rasgos y diseño de un sistema para clientes potenciales, adquiridores, propietarios, operadores e integradores.
- Para planear la transición de una arquitectura creada, a una nueva arquitectura.
- Como una guía para soporte operacional y de infraestructura, y administración de la configuración.
- Como soporte para la planeación, calendarización y actividades de presupuesto de un sistema.
- Para establecer criterios para certificar implementaciones de una arquitectura.
- Como mecanismo de conformidad para políticas externas e internas de proyectos y organizaciones.
- Como base para revisión, análisis, y evaluación de sistemas a través de su ciclo de vida.
- Para compartir lecciones aprendidas y reusar conocimiento de arquitectura a través de puntos de vista, patrones y estilos arquitectónicos.
- Para entrenar y educar a los interesados y otras partes en mejores prácticas en arquitectura y evolución de sistemas.

Las descripciones de arquitectura proporcionan muchas ventajas y usos, por lo que son una manera óptima y muy completa para documentar arquitecturas de software. Sin embargo, para que proporcionen dichos beneficios, deben ser elaboradas adecuada y profesionalmente, como lo indica el estándar ISO/IEC/IEEE 42010:2011.

2.2.4. Características de una descripción de arquitectura según el estándar ISO/IEC/IEEE 42010:2011

De manera general, el estándar ISO/IEC/IEEE 42010:2011 especifica que una descripción de arquitectura debe incluir lo siguiente:

- a. La identificación del sistema de interés, y debe incluir información complementaria determinada por la organización y/o proyecto.
- b. La identificación de los interesados del sistema que tienen intereses considerados fundamentales para la arquitectura del sistema, tales como usuarios, operadores y desarrolladores. Además, la descripción de arquitectura debe identificar los intereses considerados fundamentales para el sistema.
- c. Una definición para cada punto de vista usado en la descripción de arquitectura. Un punto de vista debe especificar:
 - Uno o más intereses enmarcados por el punto de vista.
 - Los interesados típicos para los intereses enmarcados por el punto de vista.
 - Uno o más tipos de modelo usados en el punto de vista.
 - Para cada tipo de modelo identificado, los lenguajes, notaciones, convenciones, técnicas de modelado, métodos analíticos y/o algunas otras operaciones que sean usadas en los modelos de este tipo.
 - Referencias fuentes de información.

Además, un punto de vista debe incluir información sobre técnicas de arquitectura usadas para crear, interpretar o analizar una vista gobernada por el punto de vista.
- d. Una vista arquitectónica y los modelos de arquitectura para cada punto de vista definido. Se debe incluir una vista para cada punto de vista definido. Cada vista debe:

- Identificar la información complementaria según lo especifique la organización y/o proyecto.
 - Identificar el punto de vista que la gobierna.
 - Contener modelos de arquitectura que aborden todos los intereses enmarcados por el punto de vista que la gobierna.
 - Registrar cualquier cuestión con respecto al punto de vista que la gobierna.
- e. Correspondencias y reglas de correspondencia aplicables a la descripción de arquitectura, y un registro de las inconsistencias conocidas en los contenidos de la descripción de arquitectura.
- f. Razonamiento de las decisiones de arquitectura tomadas.

Para poder elaborar una descripción de arquitectura con base en las características mencionadas, es necesario conocer algunos conceptos a detalle, tales como puntos de vista, vistas, y modelos arquitectónicos. Por esta razón, en el capítulo 3 de este trabajo, se presenta el marco conceptual en torno a las descripciones de arquitectura. Por otra parte, en los capítulos 6 al 14 se incluye una guía para elaborar descripciones de arquitectura con las características antes mencionadas.

Capítulo 3. Marco conceptual de las descripciones de arquitectura

Para elaborar y utilizar descripciones de arquitectura adecuadamente, es conveniente conocer conceptos fundamentales en torno a las mismas, tales como interesados, vistas, puntos de vistas, modelos, y otros conceptos que definen el contexto de una descripción de arquitectura.

En este capítulo, se presenta un conjunto de conceptos relacionados a las descripciones de arquitectura con base en el modelo conceptual de una descripción de arquitectura del estándar ISO/IEC/IEEE 42010:2011. Además, se incluyen algunos conceptos arquitectónicos adicionales que serán de utilidad para entender futuras secciones de este trabajo.

3.1. Interesados o *stakeholders*

En los diferentes contextos de la Arquitectura de Software (técnico, de negocio y del ciclo de vida), existen interesados en el sistema que tienen una participación fundamental en el éxito del mismo. Estos interesados, también llamados *stakeholders*, pueden ser individuos, organizaciones, o grupos de los mismos.

Algunos de los interesados más comunes y que pueden ser fundamentales para la arquitectura de un sistema son: asesores, analistas, desarrolladores, personal de mantenimiento, administradores del sistema, personal de pruebas y usuarios. En el apéndice “A” de este trabajo, se presenta un catálogo de interesados más completo.

Es importante que los principales interesados de un sistema sean identificados, ya que tienen intereses fundamentales que requieren ser conocidos con precisión para elaborar adecuadamente una descripción de arquitectura.

3.2. Intereses o *concerns* arquitectónicos

Un *concern* o interés arquitectónico, es un requerimiento, objetivo, intención, o aspiración, que uno o más interesados tienen por la arquitectura. Los intereses surgen de las necesidades y requerimientos del sistema, de opciones de diseño, metas de negocio, y consideraciones de implementación y operación del sistema durante su ciclo de vida.

Un interés arquitectónico puede pertenecer a cualquier influencia del sistema en su ambiente, incluyendo influencias de desarrollo, tecnológicas, de negocio, operacionales, organizacionales, políticas, económicas, legales, regulatorias, ecológicas y sociales (ISO/IEC/IEEE 42010:2011).

Un interés arquitectónico puede ser manifestado en diferentes formas, tales como necesidades, metas, expectativas, responsabilidades, requerimientos, restricciones de diseño, suposiciones, dependencias, atributos de calidad, decisiones de arquitectura, propósitos, riesgos, o alguna otra cuestión sobre el sistema.

Los intereses arquitectónicos deben ser considerados en todas las actividades del proceso de arquitectura de un sistema. Por ejemplo, durante la definición de la arquitectura, algunos intereses como requerimientos no funcionales, atributos de calidad y metas de negocio, dan forma a la arquitectura y son llamados *drivers* arquitectónicos (“Glosario” del Software Engineering Institute).

Por otra parte, un interés arquitectónico debe ser trazable⁸, específico, no ambiguo, comprobable⁹ y medible.

A continuación, se presentan algunas de las diferentes formas en que pueden ser manifestados los intereses arquitectónicos de un sistema, tales como atributos de calidad y metas de negocio.

3.2.1. Requerimientos arquitectónicamente significativos

Como se mencionó en el capítulo 1, la arquitectura existe para satisfacer los requerimientos de un sistema. También se mencionó sobre los requerimientos arquitectónicamente significativos, que son requerimientos que tienen un efecto profundo en la arquitectura, es decir, la arquitectura puede ser dramáticamente diferente en su ausencia. Típicamente, los requerimientos arquitectónicamente significativos son requerimientos técnicos o requerimientos importantes relacionados con los objetivos del sistema.

La importancia de los requerimientos arquitectónicamente significativos, es que se deben conocer para diseñar exitosamente una arquitectura y generar la documentación adecuada de la misma. En ocasiones, estos intereses toman forma de atributos de calidad o son metas de negocio.

3.2.2. Restricciones

Una restricción es una limitación para realizar alguna actividad con respecto a un sistema de software durante su ciclo de vida.

Las restricciones pueden ser intereses de carácter técnico o de negocio. Algunos ejemplos de restricciones son: estándares tecnológicos o de negocio, estrategias tecnológicas, y políticas de negocio.

3.2.3. Atributos o propiedades de calidad

Un atributo o propiedad de calidad es una característica medible y comprobable de un sistema, que es usada para indicar qué tan bien el sistema satisface las necesidades de sus interesados (Bass, Clements y Kazman 2012). Algunos ejemplos de atributos de calidad son seguridad y usabilidad.

⁸ Debe ser justificado con base en principios y metas, y debe ser trazado hacia características arquitectónicas y de diseño.

⁹ Debe poder demostrarse si el interés arquitectónico es cumplido.

La norma ISO/IEC 25010 “*System and software quality models*”, describe el modelo de calidad para un producto de software. Esta norma define ciertas características y subcaracterísticas de calidad para evaluar un producto de software. Tales características son presentadas como un catálogo de atributos de calidad en el apéndice “B” de este trabajo.

3.2.4. Metas de negocio

Las metas de negocio son objetivos específicos que tiene una organización, establecen el contexto de un proyecto, y son la razón fundamental para su existencia (Rozansky y Woods 2005). Las metas de negocio son la base sobre la que se justifica, construye y analiza un sistema de software.

Un sistema de software es construido para cumplir las metas de negocio, las cuales son intereses que frecuentemente conducen a requerimientos arquitectónicamente significativos.

3.2.5. Principios arquitectónicos

Un principio arquitectónico es una creencia, enfoque o intención, que guía la definición de una arquitectura. Un principio arquitectónico convierte supuestos implícitos de interesados en explícitos, y puede referirse a circunstancias actuales o a un estado futuro deseado. Un buen principio es constructivo, razonado, bien articulado, comprobable y significativo (Rozansky y Woods 2005).

3.2.6. Otros intereses arquitectónicos

Otros intereses que pueden resultar fundamentales para un sistema, son los propósitos del sistema, la conveniencia o idoneidad de la arquitectura para lograr los propósitos del sistema, la factibilidad de construir y desplegar el sistema, los riesgos e impactos potenciales del sistema para sus interesados a lo largo de su ciclo de vida, y las expectativas y suposiciones del sistema.

3.3. Puntos de vista arquitectónicos

Un punto de vista arquitectónico es un conjunto de convenciones para construir, interpretar y analizar una vista arquitectónica (ISO/IEC/IEEE 42010:2011).

Un punto de vista enmarca un conjunto de intereses, los cuales son reflejados en el punto de vista, y en la guía, principios y modelos para construir las vistas arquitectónicas gobernadas por ese punto de vista.

Un punto de vista incluye ciertas convenciones, tales como lenguajes, notaciones, tipos de modelos, reglas de diseño, métodos de modelado, técnicas de análisis y otras operaciones que son aplicadas a las vistas arquitectónicas (ISO/IEC/IEEE 42010:2011).

En el apéndice “C” de este trabajo, se presenta un catálogo de puntos de vista considerando los enfoques de documentación de Rozansky y Woods (2005), Kruchten (1995) y del SEI (Clements et al. 2003).

3.4. Vistas arquitectónicas

Una vista arquitectónica es probablemente el concepto más importante asociado con la documentación de una arquitectura de software. Una vista arquitectónica es la representación de un conjunto de elementos del sistema y las relaciones entre ellos desde la perspectiva de intereses específicos del sistema (ISO/IEC/IEEE 42010:2011).

Las vistas permiten dividir una arquitectura, que es una entidad compleja que no puede ser descrita de una manera unidimensional, en un número manejable de representaciones del sistema.

Una descripción de arquitectura puede incluir una o más vistas arquitectónicas, las cuales abordan uno o más intereses del sistema. Una vista expresa la arquitectura de un sistema de acuerdo con un punto de vista que la gobierna.

3.4.1. Ventajas de usar vistas y puntos de vista

Algunas de las ventajas de utilizar vistas y puntos de vista arquitectónicos para documentar una arquitectura son (Rozansky y Woods 2005):

- Permiten la separación de intereses arquitectónicos, lo cual ayuda a realizar actividades de diseño, análisis y comunicación, permitiendo enfocarse en aspectos por separado.
- Facilitan la comunicación con distintos grupos de interesados, ya que el utilizar diferentes puntos de vista, se puede guiar a los diferentes grupos de interesados con base en sus intereses particulares. Además, cada vista puede ser representada usando un lenguaje o notación apropiada para cada grupo de interesados.
- Ayudan a manejar la complejidad del sistema al tratar cada aspecto significativo del sistema por separado.
- Las diferentes vistas del sistema incluyen aspectos que son importantes para el equipo de desarrollo, lo que ayuda a asegurar que se construya el sistema correcto.

3.5. Marco de trabajo de arquitectura

Un marco de trabajo o “*framework*” de arquitectura, establece una práctica común para crear, interpretar, analizar y usar descripciones de arquitectura en un dominio o aplicación en particular, o en una comunidad de interesados.

Algunos de los usos de los *frameworks* de arquitectura incluyen crear descripciones de arquitectura, desarrollar herramientas de modelado arquitectónico y métodos de arquitectura, y establecer procesos para facilitar la comunicación, entrega e interoperación a través de múltiples organizaciones y/o proyectos (ISO/IEC/IEEE 42010:2011).

3.6. Estructuras y elementos arquitectónicos

Una vista arquitectónica representa una arquitectura por medio de estructuras, que son conjuntos de elementos y sus relaciones. Las estructuras arquitectónicas proporcionan una perspectiva de la arquitectura de un sistema.

En cuanto al tipo de elementos que definen, las estructuras arquitectónicas pueden ser de dos tipos:

- Estructuras estáticas, que definen los elementos de diseño interno y su estructura. Algunos ejemplos son módulos, clases, paquetes, procedimientos almacenados, servicios, archivos de datos, cables o enrutadores¹⁰.
- Estructuras dinámicas, que definen elementos en tiempo de ejecución y sus interacciones. Estas interacciones pueden ser, por ejemplo, flujos de información entre elementos.

Por otra parte, un elemento arquitectónico es una pieza fundamental a partir de la cual un sistema puede ser construido. Un elemento cuenta con los siguientes atributos (Rozansky y Woods 2005):

- Un conjunto de responsabilidades claramente definido.
- Una frontera o límite claramente definido.
- Un conjunto de interfaces claramente definidas, que especifica los servicios que el elemento proporciona a otros elementos de la arquitectura.

Es importante mencionar que los distintos tipos de estructuras arquitectónicas que existen, permiten definir tipos de vistas o puntos de vista arquitectónicos.

3.7. Perspectivas arquitectónicas

Una perspectiva arquitectónica es una colección de actividades, tácticas y guías que son usadas para asegurar que el sistema exhibe un conjunto particular de atributos de calidad que requieren consideración a través de un número de vistas arquitectónicas del sistema (Rozansky y Woods 2005).

Si un atributo de calidad es muy importante para un sistema, se pueden tener varios puntos de vista que consideren este atributo de calidad. En este caso, crear un gran número de vistas que exhiban tal atributo de calidad no es recomendable, se requiere algo “ortogonal” a los puntos de vista, es por esto que se aplican las perspectivas.

Una perspectiva puede visualizarse como un almacén de conocimiento útil que ayuda a analizar rápidamente los modelos de arquitectura para una propiedad de calidad en particular. Es una guía efectiva cuando se trabaja en un área nueva en la que no son familiares algunos intereses, problemas y soluciones arquitectónicas.

Adicionalmente, una perspectiva puede implementarse en una o más vistas arquitectónicas, y puede abordar uno o más atributos de calidad.

¹⁰ Los enrutadores o *routers* son dispositivos de interconexión de redes que aseguran el enrutamiento de paquetes y determinan las rutas que deben tomar dichos paquetes.

3.8. Estilos y patrones arquitectónicos

Un estilo arquitectónico define un conjunto de elementos particulares, especifica sus responsabilidades, e incluye reglas para organizar las relaciones entre ellos. Proporciona un conjunto de principios organizacionales para el sistema como un todo, además, expresa la arquitectura en un sentido más formal y teórico, y proporciona un marco abstracto para una familia de sistemas.

Por otra parte, el propósito de un patrón arquitectónico es proporcionar una solución probada y ampliamente aplicada a un problema particular en una forma estándar que permita ser fácilmente usada. Un patrón es más específico que un estilo arquitectónico, cuenta con un nombre que lo identifica, y define:

- El contexto en el que se aplica.
- L problema particular que soluciona.
- La descripción de la solución al problema que soluciona, la cual es usualmente un modelo que explica los elementos de diseño que incluye y la forma en que trabajan para solucionar el problema.
- Las consecuencias, tales como resultados y *tradeoffs*¹¹, que resulten de su aplicación y que ayuden a decidir si el patrón es una buena solución a un problema.

Los patrones arquitectónicos pueden ser utilizados en diferentes maneras (Rozansky y Woods 2005):

- Como fuente de conocimiento.
- Como ejemplos de buenas prácticas.
- Para crear y compartir un lenguaje común para discutir problemas de diseño.
- Para lograr una estandarización.
- Como fuente de mejora continua.
- Para fomento de la generalidad, ya que los patrones son genéricos flexibles y reusables.

Un ejemplo de un estilo arquitectónico puede ser la implementación por capas, mientras que un patrón arquitectónico basado en ese estilo, puede ser una arquitectura en 3 capas o el modelo de red *Open System Interconnection* (OSI).

Otros ejemplos de estilos arquitectónicos son filtros y tuberías (*pipes and filters*), cliente-servidor, *peer-to-peer*, implementación por capas y publicador-suscriptor.

3.9. Modelos de arquitectura y tipos de modelos

Un modelo de arquitectura es una representación abstracta o simplificada de algunos aspectos de la arquitectura de un sistema, y su propósito es comunicar esos aspectos a uno o más interesados (Rozansky y Woods 2005). Por otra parte, los tipos de modelo son las convenciones utilizadas para realizar una especie de modelo.

¹¹ Compensaciones entre los intereses de un sistema.

Un modelo puede ayudar a manejar la complejidad de un sistema, y si es efectivo, puede representar únicamente los aspectos importantes de la arquitectura, ocultando aquellos que no lo son.

Los modelos de arquitectura pueden incluir texto, dibujos informales, diagramas u otros formalismos, y usan convenciones de modelado apropiadas para representar ciertos intereses arquitectónicos. Dichas convenciones son determinadas por un tipo de modelo específico.

En el proceso de arquitectura, los modelos arquitectónicos son importantes por las siguientes razones (Rozansky y Woods 2005):

- Ayudan a entender las situaciones que se modelan. Un modelo proporciona precisión a una descripción de arquitectura, y se enfoca en los elementos más importantes de la misma.
- Son medios de comunicación que ayudan a explicar ciertas ideas o conceptos sobre la arquitectura. Los modelos reducen la cantidad de información que el lector necesita entender, y sus estructuras guían al lector a través de la información.
- Ayudan a analizar situaciones permitiendo aislar los elementos más importantes y entender sus interrelaciones. Permiten razonar sobre algún aspecto de la arquitectura que es modelada, y hacer conclusiones sobre sus propiedades.
- Ayudan a organizar procesos, equipos de trabajo, y entregables como resultado de las estructuras que describen.

Existen distintos tipos de modelos arquitectónicos (Rozansky y Woods 2005):

- Modelos cualitativos: ilustran los elementos estructurales o de comportamiento, y las características o atributos clave de la arquitectura. Algunos de estos modelos pueden elaborarse usando Lenguajes de Descripción de Arquitectura, de los que se hablará a detalle en secciones posteriores.
- Modelos cuantitativos: representan propiedades medibles de una arquitectura, tales como eficiencia de desempeño (*performance*). Algunos de estos modelos pueden elaborarse usando Lenguajes de Descripción de Arquitectura.
- Modelos cualitativos informales: son modelos gráficos informales, creados para comunicar los aspectos más importantes de una arquitectura a una audiencia no técnica.

3.10. Razonamiento y decisiones de arquitectura

El razonamiento de la arquitectura registra la explicación, justificación o lógica sobre las decisiones de arquitectura que han sido tomadas. El razonamiento de una decisión, puede incluir las bases, alternativas y *tradeoffs* considerados, y las consecuencias potenciales de la decisión.

Las decisiones pueden afectar a la arquitectura de un sistema en distintas formas, lo que se puede ver reflejado en las descripciones de arquitectura de la siguiente manera:

- Requiriendo la existencia de elementos en la descripción de arquitectura.

- Cambiando las propiedades de los elementos de la descripción de arquitectura.
- Provocando el análisis en algunos elementos de la descripción de arquitectura, incluyendo otras decisiones e intereses arquitectónicos.
- Generando nuevos intereses arquitectónicos.

3.11. Elementos de las descripciones de arquitectura, correspondencias y reglas de correspondencia

Los elementos de descripción de arquitectura son elementos primitivos tales como interesados (*stakeholders*), intereses (*concerns*), puntos de vista, vistas, tipo de modelo, modelos de arquitectura, decisiones de arquitectura y razonamiento arquitectónico.

Cuando los puntos de vista y tipos de modelo son definidos y sus modelos son construidos, pueden surgir nuevos elementos de descripción de arquitectura.

Las correspondencias definen una relación entre dos o más elementos de descripción de arquitectura. Son guiadas por una o más reglas de correspondencia, y las reglas de correspondencia pueden guiar una o más correspondencias.

Las reglas de correspondencia son usadas para hacer cumplir relaciones en las descripciones de arquitectura, o entre descripciones de arquitectura (ISO/IEC/IEEE 42010:2011).

Algunos ejemplos de relaciones que expresan correspondencias y reglas de correspondencias son: composición, refinamiento, consistencia, trazabilidad, dependencia, restricción y obligación.

3.12. Modelo conceptual de una descripción de arquitectura

Además de conocer los conceptos en torno a las descripciones de arquitectura, es importante conocer su relación, por esto, se presenta a continuación el modelo conceptual de una descripción de arquitectura que proporciona el estándar ISO/IEC/IEEE 42010:2011, junto con su explicación.

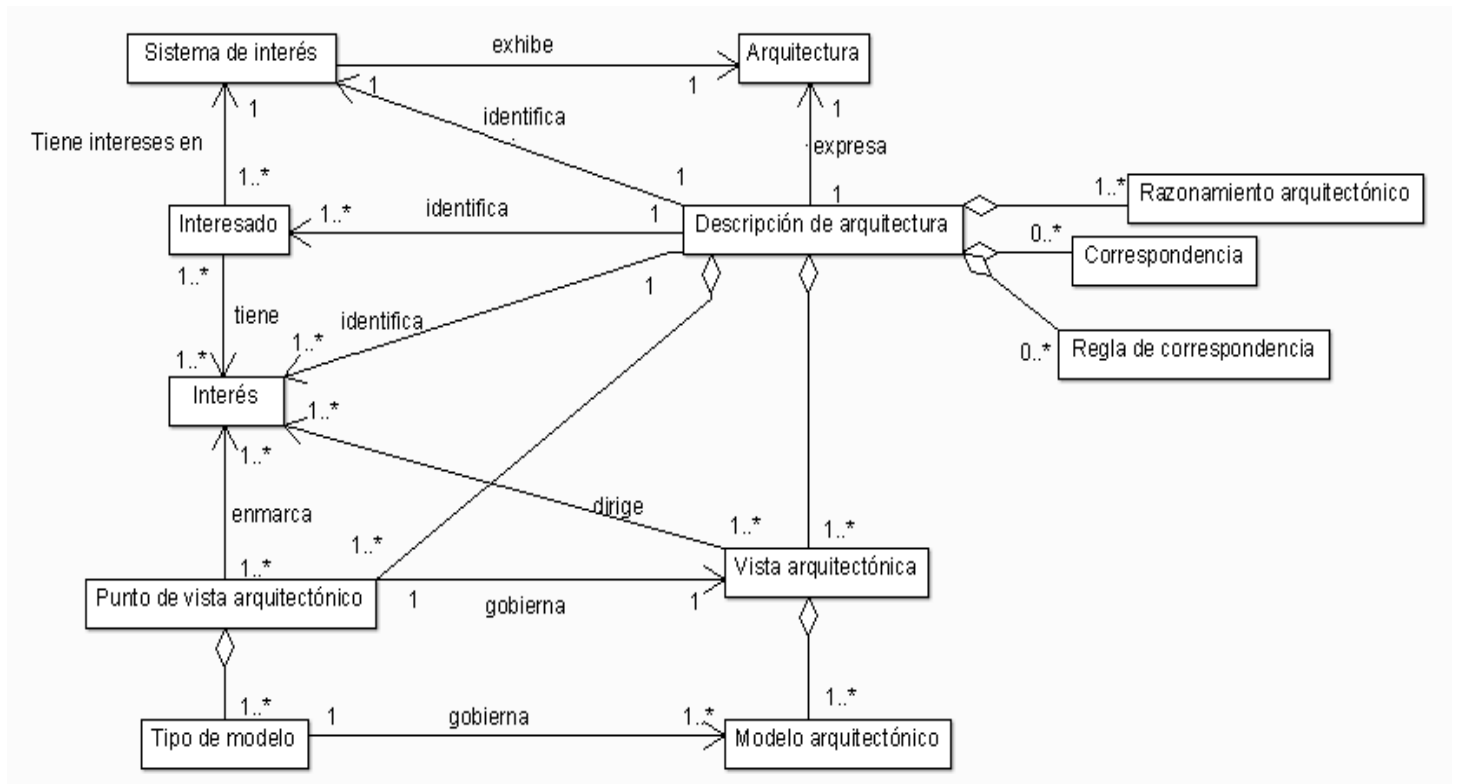


Figura 3.1 Modelo conceptual (o metamodelo) de una descripción de arquitectura (ISO/IEC/IEEE 42010:2011), usando las convenciones de los diagramas de clases.

Las relaciones del diagrama presentado anteriormente, son las siguientes:

- Un sistema presenta una arquitectura.
- Un sistema tiene uno o más interesados con intereses en el mismo.
- Un interesado tiene uno o más intereses, y un interés puede pertenecer a uno o más interesados.
- Una descripción de arquitectura expresa una arquitectura.
- Una descripción de arquitectura identifica un sistema de interés.
- Una descripción de arquitectura identifica uno o más interesados.
- Una descripción de arquitectura identifica uno o más intereses.
- Una descripción de arquitectura tienen uno o más puntos de vista arquitectónicos.
- Una descripción de arquitectura tiene una o más vistas arquitectónicas.
- Una descripción de arquitectura tiene uno o más razonamientos arquitectónicos.
- Una descripción de arquitectura puede tener una o más correspondencias.
- Una descripción de arquitectura puede tener una o más reglas de correspondencia.
- Un punto de vista arquitectónico enmarca uno o más intereses, y un interés puede ser enmarcado por uno o más puntos de vista.
- Un punto de vista arquitectónico contiene uno o más tipos de modelo.
- Un tipo de modelo gobierna un modelo arquitectónico.
- Un punto de vista arquitectónico gobierna una vista arquitectónica.

- Una vista arquitectónica aborda uno o más intereses, y un interés puede ser abordado por uno o más puntos de vista.
- Una vista arquitectónica contiene uno o más modelos arquitectónicos y un modelo de arquitectura puede ser parte de más de una vista.

Capítulo 4. Lenguajes de Descripción de Arquitectura

Una descripción de arquitectura debe incluir modelos que representen la arquitectura del sistema. Para elaborar un modelo arquitectónico se pueden utilizar distintas notaciones, sin embargo, es importante que permitan representar los aspectos más importantes de un sistema de manera clara, detallada y sin ambigüedades.

4.1. Notaciones de modelado y métodos formales en Arquitectura de Software

Las distintas notaciones que pueden ser utilizadas para el modelado arquitectónico difieren en su grado de formalidad, de acuerdo a esto, se pueden identificar 3 tipos (Bass, Clements y Kazman 2012):

- Notaciones informales: son notaciones comúnmente gráficas, que usan diagramas de propósito general, herramientas de edición y convenciones visuales comunes. Sus semánticas¹² se caracterizan en lenguaje natural, y no pueden ser analizadas formalmente. Un ejemplo de este tipo de notaciones es un diagrama en *PowerPoint*.
- Notaciones semiformales: son notaciones de modelado estándar, que prescriben elementos gráficos y reglas de construcción, pero no proporciona un trato de semántica completa para el manejo de estos elementos. Los modelos elaborados con estas notaciones pueden ser analizados rudimentariamente para determinar si un modelo satisface ciertas propiedades sintácticas¹³. Un ejemplo de este tipo de notaciones es el *Unified Modeling Language* (UML).
- Notaciones formales: son notaciones que cuentan con una semántica precisa (usualmente con una base matemática). Permiten el análisis formal, tanto sintáctico como semánticamente. Existen muchas notaciones formales que proporcionan un vocabulario gráfico y la semántica subyacente para la representación de una arquitectura. Los Lenguajes de Descripción de Arquitectura comúnmente forman parte de este tipo de notaciones.

La práctica del diseño y modelado arquitectónico ha tendido a ser *ad-hoc* e informal. Como resultado, los modelos arquitectónicos han sido entendidos pobremente por los desarrolladores; no han podido ser analizados en cuanto su consistencia y completitud; y las restricciones asumidas en el diseño inicial no son mantenidas a medida que un sistema evoluciona. En respuesta a estos problemas, investigadores de la industria y académicos han propuesto notaciones formales para representar y analizar diseños arquitectónicos, generalmente referidas como Lenguajes de Descripción de Arquitectura (LDAs). Estas notaciones usualmente proporcionan tanto un *framework* conceptual como una sintaxis concreta para representar arquitecturas de software. Por lo general,

¹² La semántica es la parte de la lingüística que estudia los aspectos del significado, sentido o interpretación de los elementos de un lenguaje.

¹³ La sintaxis es la parte de la gramática que estudia la forma en la que se deben combinar y relacionar los elementos de un lenguaje para formar expresiones.

proporcionan herramientas para realizar validaciones sintácticas, compilar, analizar o simular descripciones de arquitectura (Garlan, Monroe y Wile 1997).

Los métodos formales aplicados al modelado y análisis de la arquitectura y de los estilos arquitectónicos surgieron como una posible solución al estado *ad-hoc* de la práctica de arquitectura (Allen 1997).

Al proporcionar semánticas precisas para un sistema a nivel de arquitectura, un modelo formal puede proporcionar bases para el análisis riguroso y justificable de propiedades críticas del sistema. Si el modelado formal es aplicado a nivel de un estilo arquitectónico, los análisis se pueden extender a cualquier sistema que conforme el estilo. Además, si las restricciones establecidas para un sistema son definidas precisamente, es posible determinar si un sistema conforma una arquitectura, y si una arquitectura dada conforma un estilo arquitectónico (Allen 1997).

Por otra parte, un modelo arquitectónico formal puede exponer solo las abstracciones importantes, y puede ser usado como base para la verificación de una implementación. Además, la descripción formal de una arquitectura tiene el potencial de aumentar la eficacia de la arquitectura como un vehículo de comunicación sobre el diseño de un sistema (Allen). Los Lenguajes de Descripción de Arquitectura (LDAs), en general, aplican los métodos formales antes mencionados a la Arquitectura de Software.

4.2. ¿Qué son los Lenguajes de Descripción de Arquitectura?

Un Lenguaje de Descripción de Arquitectura (LDA) es comúnmente un lenguaje formal, una notación de propósito especial que permite, entre otras cosas, realizar modelos de arquitectura de un sistema que pueden ser incluidos en una descripción de arquitectura. Los Lenguajes de Descripción de Arquitectura pueden ser usados en diversas actividades durante el proceso de arquitectura, incluyendo actividades de documentación, análisis y evaluación de ciertas características arquitectónicas.

Al elaborar una descripción de arquitectura, un Lenguaje de Descripción de Arquitectura puede permitir elaborar uno o más tipos de modelos que enmarquen intereses particulares para una audiencia de interesados.

4.2.1. Breve historia de los Lenguajes de Descripción de Arquitectura

Los Lenguajes de Descripción de Arquitectura surgieron en la década de los 90's, desde entonces, ha surgido una gran cantidad de ellos, incluyendo tanto lenguajes de propósito arquitectónico general como de propósito específico.

La primera generación de Lenguajes de Descripción de Arquitectura que surgió, incluye lenguajes como Wright (Allen 1997), Rapide (Luckham y Vera 1995), Darwin (Magee et al. 1995), Aesop (Garlan, Allen y Ockerbloom 1994), y Unicon (Shaw et al. 1995). Tales lenguajes están caracterizados por una sintaxis propietaria del lenguaje, soportada por compiladores, herramientas de análisis estático y dinámico, y máquinas virtuales o simuladores (Dashofy, Hoek y Taylor 2001). Estos lenguajes también se caracterizan por haber surgido en la academia, y estar enfocados en gran parte en la estructura de la arquitectura y en el análisis formal (Garlan, Monroe y Wile 2000).

Aunque muchos de los Lenguajes de Descripción de Arquitectura de primera generación mencionados comparten características en común, también han sido destinados para propósitos específicos, lo que ha dado como resultado una proliferación de Lenguajes de Descripción de Arquitectura que son difíciles de usar en conjunto. Con el objetivo de solucionar tal problema, se han desarrollado lenguajes como Acme, un lenguaje de intercambio que permite la integración de diferentes LDAs, proporcionando una forma de intercambiar modelos arquitectónicos. La esencia de Acme es proporcionar un vocabulario u ontología fija para representar estructuras arquitectónicas, y proporcionar un *framework* de semántica abierta para que las estructuras arquitectónicas puedan incluir propiedades específicas de otros LDAs (Garlan, Monroe y Wile 1997).

Aunque lenguajes como Acme soportan un mecanismo de intercambio para explotar características de distintos Lenguajes de Descripción de Arquitectura, con el tiempo ha surgido la necesidad de desarrollar lenguajes que sean ampliamente extensibles. Como resultado, se han desarrollado LDAs y *frameworks* arquitectónicos enfocados en la extensibilidad, tal es el caso de ByADL (Ruscio et al. 2010) y xADL (Dashofy, Hoek y Taylor 2001). Este último es un LDA flexible, basado en XML, y cuyo objetivo es enfrentar los problemas de extensibilidad de los Lenguajes de Descripción de Arquitectura convencionales.

Actualmente existe una gran cantidad de Lenguajes de Descripción de Arquitectura, algunos de ellos enfocados en dominios específicos. Algunos de los más comunes en la actualidad son los sistemas embebidos, sistemas de tiempo real, y sistemas críticos. Un ejemplo de un LDA enfocado en dichos sistemas es AADL (Feiler, Gluch y Hudak 2006).

Algunos de los elementos y conceptos más comunes de los LDAs son componentes, conectores, configuraciones o sistemas, restricciones, propiedades funcionales y no funcionales, estilos arquitectónicos, dinamismo, comunicación, abstracción y alternativas de implementación. En la tabla 4.1 se presentan algunos de los Lenguajes de Descripción de Arquitectura más comunes.

LDA	Desarrollador
AADL (Feiler, Gluch y Hudak 2006)	Society of Automotive Engineers (SAE)
Acme (Garlan, Monroe y Wile 1997)	Universidad Carnegie Mellon
Aesop (Garlan, Allen y Ockerbloom 1994)	Universidad Carnegie Mellon
ALI (Bashroush et al. 2008)	Queen's University Belfast
ByADL (Ruscio et al. 2010)	Universidad de L'Aquila
EADL (Li et al. 2009)	Universidad Tsinghua
Darwin (Magee et al. 1995)	Imperial College
MADL (Smeda, Oussalah y Khammaci 2005)	Universidad de Nantes
Pi-ADL (Oquendo 2004)	Universidad de Savoie
Rapide (Luckham y Vera 1995)	Universidad Princeton
SADL (Ricks, Weir y Wells 1995)	Universidad de Alabama en Huntsville
Unicon (Shaw et al. 1995)	Universidad Carnegie Mellon
Wright (Allen 1997)	Universidad Carnegie Mellon
xADL (Dashofy, Hoek y Taylor 2001)	Universidad de California, Irvine

Tabla 4.1 Algunos Lenguajes de Descripción de Arquitectura.

4.3. Características de un Lenguaje de Descripción de Arquitectura

4.3.1. Características que especifica el estándar ISO/IEC/IEEE 42010:2011

Según el estándar “*Systems and software engineering, Architecture description*” (ISO/IEC/IEEE 42010:2011), un Lenguaje de Descripción de Arquitectura debe especificar lo siguiente:

- La identificación de uno o más intereses arquitectónicos para ser expresados por el lenguaje.
- La identificación de uno o más interesados que tienen dichos intereses.
- Los tipos de modelo implementados por el lenguaje que enmarcan esos intereses.
- Cualquier punto de vista arquitectónico.
- Reglas de correspondencia relacionadas a sus tipos de modelo.

Se puede notar que las características especificadas por el estándar ISO/IEC/IEEE 42010:2011 para un Lenguaje de Descripción de Arquitectura son muy generales, ya que únicamente especifica las disposiciones que hacen cumplir las propiedades deseadas para los mismos.

Es importante mencionar que, de acuerdo las especificaciones del estándar ISO/IEC/IEEE 42010:2011, UML puede considerarse como un LDA, sin embargo, este trabajo se enfoca en aquellos LDAs más formales es decir, aquellos que cuentan con una sintaxis y semántica bien definidas, con capacidades de análisis, o que cuentan con características propiamente arquitectónicas, por lo que se excluirán notaciones de modelado poco formal o general.

4.3.2. Características según otros autores

Desde el origen de los primeros Lenguajes de Descripción de Arquitectura, algunos autores establecieron ciertas características con que éstos debían contar. Shaw et al. (1994), por ejemplo, especifica que un lenguaje de descripción de arquitectura debe contar con propiedades para representar componentes, interfaces, conectores y protocolos¹⁴. Luckham y Vera (1995), por su parte, definen que algunos de los requerimientos esenciales para un Lenguaje de Descripción de Arquitectura son la abstracción de componentes y la habilidad para modelar arquitecturas dinámicas. Por otro lado, Allen (1997) establece que un LDA debe soportar la descripción de configuraciones y estilos arquitectónicos, el análisis de propiedades de interés, y la aplicación de problemas prácticos en sistemas reales.

Clements (1996), con base en un estudio sobre Lenguajes de Descripción de Arquitectura, define el siguiente conjunto mínimo de requerimientos para los mismos:

¹⁴ Un protocolo de comunicación es un conjunto de reglas que intervienen en la comunicación entre dos elementos en un sistema de comunicaciones.

- Deben soportar las tareas de creación, refinamiento, y validación de una arquitectura. Deben incluir reglas acerca de lo que constituye una arquitectura completa o consistente.
- Deben permitir representar estilos arquitectónicos comunes.
- Deben permitir elaborar vistas del sistema que expresen su información arquitectónica.
- Deben soportar el análisis de la arquitectura, o la capacidad para generar rápidamente implementaciones prototipo.

Se puede notar que existen muchas características deseadas para un Lenguaje de Descripción de Arquitectura, sin embargo, en la actualidad muchos LDAs pueden variar en diversos aspectos tales como: la habilidad para manejar arquitecturas en tiempo real, el soporte para la especificación de estilos arquitectónicos, la habilidad para permitir la definición de nuevas sentencias en el lenguaje, el soporte para el análisis de ciertas características como la consistencia, y el manejo de diferentes ejemplificaciones de la misma arquitectura.

Debido a las diferencias entre los Lenguajes de Descripción de Arquitectura, se requiere tener cuidado al seleccionar alguno, ya que se debe asegurar que se adapte a las necesidades de la organización y al tipo de sistema cuya arquitectura será modelada.

4.4. Diferencia entre los Lenguajes de Descripción de Arquitectura y otros lenguajes

Muchas veces puede confundirse un Lenguaje de Descripción de Arquitectura con otro tipo de lenguajes, como los de requerimientos, de programación, o de modelado.

En principio, los Lenguajes de Descripción de Arquitectura difieren de los lenguajes de requerimientos en que estos últimos describen un problema, mientras que los lenguajes de Descripción de Arquitectura Describen una solución.

Por otra parte, los Lenguajes de Descripción de Arquitectura se diferencian de los lenguajes de programación ya que estos últimos dirigen la arquitectura de un sistema a soluciones puntuales y específicas, mientras que los Lenguajes de Descripción de Arquitectura describen una solución a un nivel de abstracción más alto.

Los Lenguajes de Descripción de Arquitectura son diferentes a los lenguajes de modelado de propósito general (como UML), ya que estos últimos están más enfocados en la esencia del sistema como un todo más que en las partes del mismo. Los Lenguajes de Descripción de Arquitectura se concentran en la representación de cada componente del sistema (Clements, 1996) y no se enfocan únicamente en el modelado, si no que consideran el análisis y la validación.

Finalmente, es importante destacar la existencia de otras notaciones, que aunque cuentan con facilidades para la especificación y el modelado, no pueden ser usadas para especificar, modelar o analizar una arquitectura de software. Dichas notaciones son:

- Los Lenguajes de Interconexión de Módulos (Module Interconnection Languages o MILs), que han sido desarrollados para soportar la descripción de estructuras de

programas de gran escala independientemente de cualquier lenguaje de programación o sistema. Estos lenguajes incrementan la independencia de los componentes del sistema, sin embargo, no proporcionan un medio para especificar patrones abstractos de interacción o para definir familias de sistemas (estilos arquitectónicos). Además, no se enfocan en la especificación y análisis de información.

- Los Formalismos de Interconexión de Módulos (Module Interconnection Formalisms o MIFs), que extienden a los Lenguajes de Interconexión de Módulos con semánticas de interconexión. Sin embargo, estos formalismos no pueden ser usados para construir sistemas que usen otros patrones de comunicación, por lo que no son mecanismos generales para describir las interacciones entre arquitecturas de software.
- Los Lenguajes de Definición de Interfaz (Interface Definition Languages o IDLs), desarrollados para enfrentar el problema de la inconsistencia en las representaciones de datos derivada de la integración de sistemas. Estos lenguajes permiten definir interfaces y mapeos abstractos de construcciones de lenguajes de programación en representaciones intermedias que median la transformación entre formatos de bajo nivel incompatibles. El inconveniente de estos lenguajes, es que no proporcionan bases suficientes para la descripción y análisis de interacciones en arquitecturas de software.

4.5. Ventajas y desventajas de los Lenguajes de Descripción de Arquitectura

Los Lenguajes de Descripción de Arquitectura pueden tener muchas fortalezas y ventajas sobre otros tipos de lenguajes y notaciones menos formales o de propósito general. A continuación se presentan las más significativas:

- No son lenguajes simplemente de modelado, permiten especificar la arquitectura de un sistema de manera formal, ya que proporcionan bases matemáticas como método formal, que permiten controlar el sistema con mayor precisión.
- Están destinados a ser entendidos tanto por máquinas como por humanos.
- Pueden describir un sistema al nivel de abstracción más alto posible.
- Permiten analizar aspectos fundamentales de la arquitectura, tales como consistencia y rendimiento. Estas características no son propias de otras notaciones menos formales como UML.
- Pueden soportar la generación automática de elementos de software.
- Permiten presentar con mayor detalle los modelos de una arquitectura, por lo que pueden ser más útiles para la comunicación entre los interesados en la arquitectura.
- Son una forma no ambigua de representar una arquitectura, una fortaleza sobre UML y notaciones informales, cuyas construcciones carecen de semánticas formales, y por lo tanto pueden ser una fuente de ambigüedad e inconsistencia.
- Algunos de los Lenguajes de Descripción de Arquitectura permiten extender ciertas propiedades para poder modelar características arquitectónicas adicionales.

A pesar de las ventajas que proporcionan los Lenguajes de Descripción de Arquitectura, también implican ciertas desventajas, entre las cuales se encuentran:

- El uso de las notaciones formales de los Lenguajes de Descripción de Arquitectura puede implicar más esfuerzo para la creación y comprensión de modelos arquitectónicos.
- En la actualidad, no son demasiado conocidos o utilizados como otras notaciones de modelado de propósito general.
- Muchos Lenguajes de Descripción de Arquitectura son de dominio específico, por lo que para modelar la arquitectura de cierto sistema, se podría requerir el uso de más de un LDA.
- En su mayoría, las herramientas que los soportan son propietarias, lo que implica la necesidad de su uso. En contraste, existen muchas herramientas para utilizar otras notaciones menos formales y de propósito general como UML.
- Muchos Lenguajes de Descripción de Arquitectura no soportan la elaboración de múltiples vistas demandadas por los arquitectos de software.
- Muchos Lenguajes de Descripción de Arquitectura son restrictivos e imponen modelos de arquitectura particulares, lo puede resultar poco apropiado.
- Muchos Lenguajes de Descripción de Arquitectura son muy genéricos, con una falta de especialización de dominio, lo que significa que no sirven para necesidades específicas.

PARTE 2 GUÍA PARA DOCUMENTAR ARQUITECTURAS DE SOFTWARE UTILIZANDO LENGUAJES DE DESCRIPCIÓN DE ARQUITECTURA

En esta parte, se presenta la guía para documentar arquitecturas de software utilizando Lenguajes de Descripción de Arquitectura. Se compone de los siguientes capítulos:

CAPÍTULO 5 Consideraciones sobre la guía. Presenta las consideraciones y principios fundamentales de la guía.

CAPÍTULO 6 Identificación y documentación de la información general de una descripción de la arquitectura. Presenta las consideraciones para identificar y documentar la información general de una descripción de arquitectura.

CAPÍTULO 7 Identificación y documentación de interesados e intereses. Presenta las consideraciones para identificar y documentar los interesados e intereses fundamentales para arquitectura de un sistema.

CAPÍTULO 8 Criterios para seleccionar Lenguajes de Descripción de Arquitectura. Presenta las consideraciones para la selección de los Lenguajes de Descripción de Arquitectura que serán utilizados en la elaboración de modelos arquitectónicos.

CAPÍTULO 9 Identificación y documentación de puntos de vista arquitectónicos. Presenta los criterios para identificar y documentar puntos de vista arquitectónicos.

CAPÍTULO 10 Desarrollo y documentación de vistas arquitectónicas. Presenta las consideraciones para el desarrollo de vistas arquitectónicas.

CAPÍTULO 11 Modelado y especificación arquitectónica utilizando Lenguajes de Descripción de Arquitectura. Presenta una guía para elaborar modelos y especificaciones arquitectónicas utilizando Lenguajes de Descripción de Arquitectura específicos.

CAPÍTULO 12 Documentación de relaciones arquitectónicas. Presenta las consideraciones para documentar relaciones entre elementos de una descripción de arquitectura.

CAPÍTULO 13 Documentación de razonamiento y decisiones arquitectónicas. Presenta las consideraciones para documentar razonamiento y decisiones arquitectónicas.

CAPÍTULO 14 Validación y mantenimiento de una descripción de arquitectura. Presenta las consideraciones para la validación y mantenimiento de descripciones de arquitectura.

CAPÍTULO 15 Evaluación de la guía. Presenta la evaluación de la guía.

Capítulo 5. Consideraciones sobre la guía

Para que la guía de documentación arquitectónica pueda utilizarse adecuadamente, se requieren tomar en cuenta ciertos aspectos relevantes sobre la misma, tales como su objetivo, características principales, y los fundamentos primordiales que dirigen su contenido y estructura.

5.1. Objetivo

El objetivo principal de la guía que se presenta en los siguientes capítulos, es proporcionar un conjunto de facilidades, consejos y recomendaciones que conduzcan la documentación de arquitecturas de software a través de descripciones de arquitectura, utilizando Lenguajes de Descripción de Arquitectura, y con base en las consideraciones del estándar ISO/IEC/IEEE 42010:2011.

5.2. Características principales

Las características principales de la guía son las siguientes:

- Está basada en las especificaciones del estándar ISO/IEC/IEEE 42010:2011.
- Se enfoca en la documentación de arquitecturas de software a través de descripciones de arquitectura.
- Incluye las consideraciones necesarias para utilizar Lenguajes de Descripción de Arquitectura como medio para realizar especificaciones y modelos arquitectónicos. Además, presenta una guía para realizar modelos y especificaciones arquitectónicas utilizando los siguientes LDAs: Acme, AADL, Wright y xADL.
- Presenta una guía para identificar y documentar cada uno de los elementos de una descripción de arquitectura, tales como vistas, puntos de vista y modelos arquitectónicos.
- Presenta un conjunto de plantillas detalladas para apoyar en la documentación de elementos específicos de una descripción de arquitectura, tales como puntos de vista y vistas arquitectónicas.
- Presenta un conjunto de técnicas, consejos y recomendaciones para elaborar descripciones de arquitectura que resulten de utilidad para su audiencia.
- Presenta catálogos de algunos elementos de las descripciones de arquitectura, tales como interesados y puntos de vista arquitectónicos.
- Presenta ejemplos prácticos con el fin de ser más comprensible por su audiencia.
- La guía puede utilizarse para documentar aspectos de distintas arquitecturas de software, independientemente de los interesados, intereses, comportamiento, tamaño o tipo del sistema, y en diversos modelos de procesos de desarrollo de software.
- La guía cuenta con cierta flexibilidad, que permite realizar diversas actividades de documentación de arquitectura con base en las necesidades de una organización y/o proyecto. Tales actividades incluyen la selección de Lenguajes de Descripción de Arquitectura, y la definición de las vistas arquitectónicas.
- En conjunto con la guía, se pueden utilizar aspectos de otros enfoques de documentación tales como el método 4+1 (Kruchten 1995) o el enfoque de Soni et al. (1995).

5.3. ¿A quién está dirigida?

Debido a que existen muchos interesados en un sistema de software, y en particular su arquitectura, esta guía puede ser de interés para las siguientes audiencias:

- Arquitectos u otros encargados de documentar, especificar, diseñar, evaluar o modelar una arquitectura, ya que aunque esta guía se centra en la documentación arquitectónica, dicha actividad se relaciona con las antes mencionadas.
- Encargados del mantenimiento o implementación de un sistema o parte de él, ya que podrían requerir utilizar y entender descripciones de arquitectura para realizar sus actividades.
- Interesados principales en la arquitectura de un sistema, como programadores o personal de pruebas, que requieran que sus intereses sean identificados y considerados al elaborar una descripción de arquitectura.
- Para aquellos interesados en conocer más sobre la documentación de arquitecturas de software a través de descripciones de arquitectura, con base en el estándar ISO/IEC/IEEE 42010:2011, y haciendo uso de Lenguajes de Descripción de Arquitectura.

5.4. Fundamentos primordiales que dirigen el contenido

La guía se fundamenta principalmente en el estándar internacional ISO/IEC/IEEE 42010:2011 como lineamiento de documentación, el uso de las descripciones de arquitectura como producto de trabajo de documentación arquitectónica, el uso de Lenguajes de Descripción de Arquitectura como notación de especificación y modelado arquitectónico, y técnicas y prácticas específicas que ayuden a realizar actividades de documentación.

5.4.1. Uso del estándar ISO/IEC/IEEE 42010:2011

En ocasiones, resulta útil seguir estándares internacionales para realizar actividades de Ingeniería de Software, ya que permiten sobrepasar algunas barreras técnicas en el ámbito de dicha disciplina.

El estándar “*Systems and software engineering, Architecture description*” (ISO/IEC/IEEE 42010:2011), dirige la creación, análisis y mantenimiento de arquitecturas de software a través del uso de descripciones de arquitectura, y especifica la manera en la que éstas son organizadas y expresadas.

Es importante mencionar que el estándar no especifica un formato o medio para registrar descripciones de arquitectura, y puede ser usado en distintos enfoques, tales como técnicas basadas en modelos y en repositorios. El estándar tampoco prescribe métodos, modelos, notaciones o técnicas de arquitectura específicas para producir descripciones de arquitectura.

El estándar ISO/IEC/IEEE 42010:2011 permite guiar de manera general la creación de descripciones de arquitectura, pero es flexible, ya que permite incluir métodos, modelos, notaciones, técnicas, consejos, prácticas, y otros elementos específicos de arquitectura.

Aunque el estándar no prescribe la forma que debe tomar una descripción de arquitectura (documentos, hipertextos, repositorios, etc.), permite definir una serie de pasos para elaborar descripciones de arquitectura y especificar su contenido detallado.

Adicionalmente, es importante mencionar que al seguir el estándar, se pueden construir mejores sistemas, aunque debe considerarse que la calidad de un sistema no es una consecuencia automática del uso del estándar, si no que la consecuencia prevista es que una descripción de arquitectura sea entendida de mejor manera, lo que debería contribuir a mejores arquitecturas y por lo tanto, a mejores sistemas (página ISO/IEC/IEEE 42010:2011).

5.4.2. Uso de descripciones de arquitectura

Las descripciones de arquitectura son una forma detallada y completa de documentar arquitecturas de software. Son productos de trabajo más específicos que las vistas arquitectónicas, ya que permiten documentar más aspectos de una arquitectura. Además, su elaboración puede basarse en las características que especifica el estándar internacional ISO/IEC/IEEE 42010:2011 (ver sección 2.2.4.), utilizado como lineamiento de documentación en esta guía.

Una de las razones para utilizar descripciones de arquitectura como forma de documentación, es debido a que ofrecen muchas ventajas, como las listadas en la sección 2.2.3 de este trabajo.

Una descripción de arquitectura puede visualizarse como un producto de trabajo individual, pero está conformada por un conjunto de elementos que permiten documentar aspectos de una arquitectura independientemente. Tales elementos incluyen puntos de vista y vistas arquitectónicas.

5.4.3. Uso de Lenguajes de Descripción de Arquitectura

Debido a que el estándar ISO/IEC/IEEE 42010:2011 no prescribe notaciones específicas para elaborar modelos arquitectónicos, en esta guía, se propone el uso de Lenguajes de Descripción de Arquitectura como notación de modelado arquitectónico debido a las ventajas que proporcionan (ver sección 4.4).

La producción de modelos arquitectónicos es sólo una de las actividades para elaborar descripciones de arquitectura, sin embargo, los modelos arquitectónicos son una de las partes más importantes de la documentación de una arquitectura, ya que son representaciones de las características fundamentales de la arquitectura, pueden proporcionar información clara sobre la misma, y son usados para diversos fines en el ciclo de vida de un sistema. Por tal motivo, se optó por los Lenguajes de Descripción de Arquitectura como notación de modelado, ya que permiten representar y especificar arquitecturas de software de manera clara, detallada y sin ambigüedades.

Es importante mencionar que la guía es flexible para utilizar cualquier Lenguaje de Descripción de Arquitectura, pero debido a la gran cantidad de lenguajes que existe, en este trabajo únicamente serán considerados a detalle los siguientes: Acme, AADL, Wright y xADL.

5.4.4. Uso de técnicas arquitectónicas

Además del uso de Lenguajes de Descripción de Arquitectura, en la guía son incluidas algunas técnicas que apoyen en la realización de las actividades de documentación.

5.5. Actividades generales para elaborar descripciones de arquitectura (guía rápida de documentación)

En general, el estándar ISO/IEC/IEEE 42010:2011 es neutral en cuanto a procesos y métodos, sin embargo, en esta guía se propone un conjunto de actividades específicas para elaborar descripciones de arquitectura con la intención de apoyar al lector en el desarrollo de dicha documentación.

Considerando el estándar ISO/IEC/IEEE 42010:2011 y el uso de Lenguajes de Descripción de Arquitectura, las actividades para elaborar una descripción de arquitectura que se proponen y desarrollan en este trabajo son las siguientes:

- Identificar y documentar la información general de una descripción de arquitectura que incluye:
 - La identificación y documentación del sistema de interés.
 - La identificación y documentación de información complementaria.
 - La documentación de los resultados de las evaluaciones de la arquitectura y de la descripción de arquitectura.

Con esta información se puede identificar y documentar más fácilmente otros elementos de la descripción de arquitectura.

- Identificar y documentar los interesados del sistema que tienen una participación en la arquitectura. La identificación de los interesados de la arquitectura permite generar una idea más clara sobre la naturaleza de arquitectura, además, proporciona información útil para seleccionar algunos elementos específicos de la descripción de la arquitectura y tomar de decisiones futuras sobre la misma.
- Identificar los intereses arquitectónicos relevantes. La identificación de tales intereses puede resultar más sencilla si se ha identificado con anterioridad a los interesados de dicha arquitectura.
- Seleccionar uno o más Lenguajes de Descripción de Arquitectura que permitan generar modelos arquitectónicos que reflejen los intereses del sistema. Con tal selección, se puede proceder a identificar, seleccionar y documentar los puntos de vista arquitectónicos que cubran los intereses ya identificados.
- Seleccionar o definir, y documentar uno o más puntos de vista arquitectónicos para expresar la arquitectura del sistema, de tal forma que cada interés sea enmarcado por al menos un punto de vista.
- Desarrollar y documentar las vistas arquitectónicas del sistema con base en los puntos de vista seleccionados. El desarrollo de una vista arquitectónica incluye:
 - La identificación de los puntos de vista que gobiernan a las vistas.
 - La elaboración y documentación de los modelos arquitectónicos del sistema utilizando Lenguajes de Descripción de Arquitectura.
 - Realizar el análisis de consistencia entre vistas arquitectónicas.
 - Documentar las cuestiones de cada vista con respecto al punto de vista que la gobierna.

- Documentar las relaciones entre los elementos de la descripción de arquitectura que incluye:
 - La documentación del análisis de consistencia.
 - La documentación de reglas de correspondencia.
 - La documentación de las correspondencias.
- Registrar el razonamiento y las decisiones arquitectónicas del sistema, y proporcionar evidencia de la consideración de múltiples alternativas arquitectónicas.
- Realizar una validación de la descripción de arquitectura elaborada.

Aunque las actividades mencionadas pueden realizarse secuencialmente, algunas de ellas también pueden realizarse simultáneamente. En el caso de la identificación de intereses e interesados, ambas actividades se podrían realizar simultáneamente debido a su naturaleza. Por otra parte, la selección de Lenguajes de Descripción de Arquitectura y la selección o definición de puntos de vista pueden realizarse al mismo tiempo. Asimismo, la documentación del razonamiento arquitectónico puede realizarse en conjunto con otras actividades.

En la figura 5.1 se presenta un diagrama de actividades que ilustra una manera en que pueden elaborarse las descripciones de arquitectura considerando las actividades antes mencionadas.

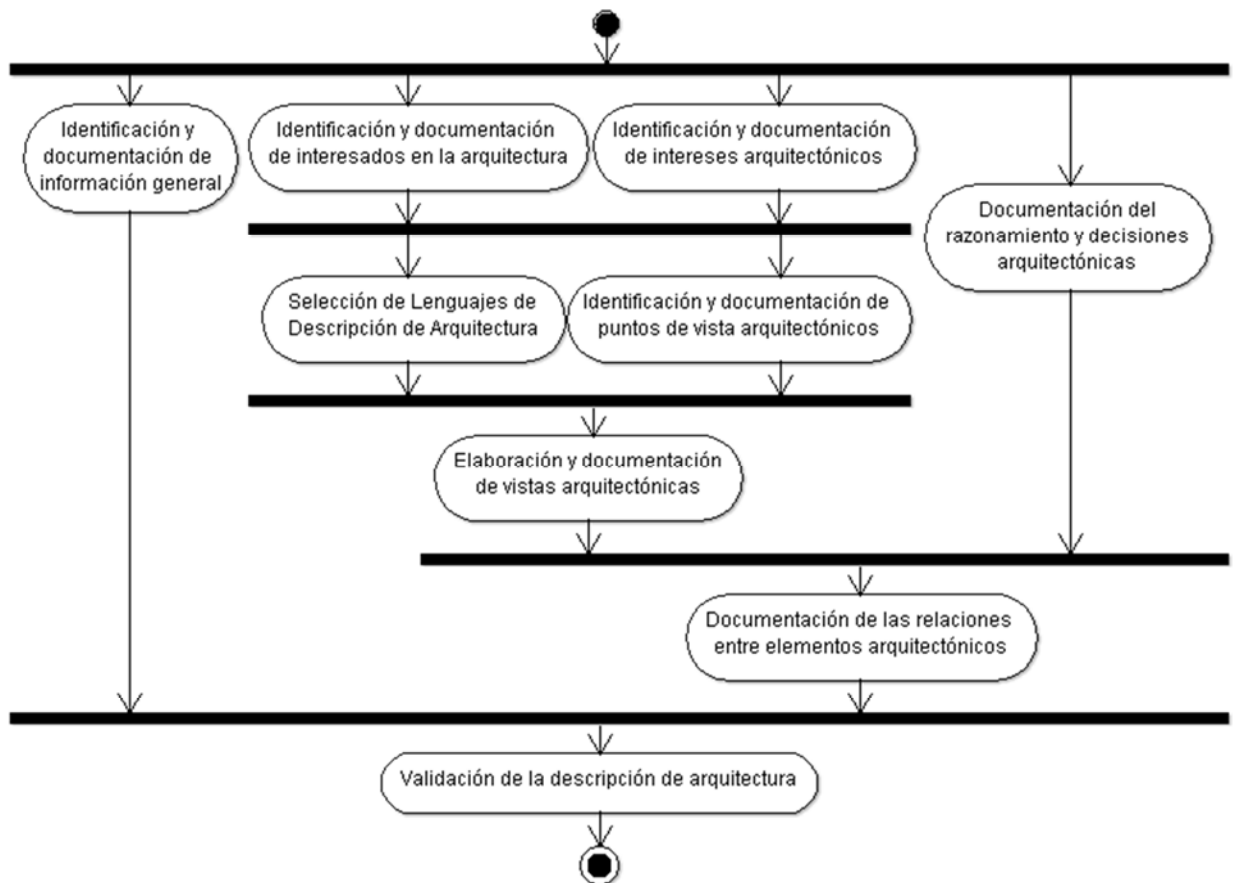


Figura 5.1 Diagrama de actividades para elaborar descripciones de arquitectura.

Las actividades para elaborar descripciones de arquitectura son presentadas a detalle en los siguientes capítulos de este trabajo, tal como se ilustra en la tabla 5.1.

Actividad	Capítulo
Identificación y documentación de información general de una descripción de arquitectura.	6
Identificación y documentación de interesados e intereses en la arquitectura.	7
Selección de Lenguajes de Descripción de Arquitectura.	8
Identificación y documentación de puntos de vista arquitectónicos.	9
Desarrollo y documentación de vistas arquitectónicas, incluyendo la elaboración de modelos arquitectónicos utilizando Lenguajes de Descripción de Arquitectura.	10 y 11
Documentación de relaciones entre elementos arquitectónicos.	12
Documentación del razonamiento y decisiones arquitectónicas	13
Validación de una descripción de arquitectura.	14

Tabla 5.1 Actividades para elaborar descripciones de arquitectura y los capítulos en que se presentan.

5.6. Consideraciones adicionales

El estándar ISO/IEC/IEEE 42010:2011 está destinado a ser usado en un ciclo de vida y/o en el contexto de un método o *framework* de arquitectura usado para desarrollar descripciones de arquitectura.

En particular, esta guía puede ser usada junto con cualquier modelo de procesos de desarrollo de software, sin embargo, su uso implica cierto tiempo y trabajo con el que probablemente no se cuente si se sigue alguna metodología ágil. Por otra parte, aunque la guía está destinada a ser utilizada durante el proceso de arquitectura, éste se relaciona fuertemente con otros procesos como el análisis, diseño, desarrollo y pruebas del sistema.

Por otra parte, el estándar ISO/IEC/IEEE 42010:2011 puede ser usado para estructurar una descripción de arquitectura aplicando algún *framework* de arquitectura, como el *framework* de arquitectura de sistemas de información de Zachman (Zachman 1987), el “*Ministry of Defence Architecture Framework*” (MODAF) , el *framework* de arquitectura del Open Group (TOGAF), el modelo de vistas “4 +1” (Kruchten, 1995), el método de vistas de Siemens (Hofmeister, Nord y Soni 1999), y el “*Reference Model for Open Distributed Processing*” (RM-ODP).

Por otra parte, la guía presentada en este trabajo puede ser usada junto con elementos de otros enfoques de documentación, tales como el enfoque del SEI (Clements et al. 2003), el de Rozansky y Woods (2005) y de Kruchten (1995). Tal facilidad es posible debido a que muchos de los enfoques de documentación están basados en la generación de vistas arquitectónicas, parte fundamental de esta guía.

En cuanto a los sistemas cuya arquitectura puede ser documentada usando esta guía, el estándar ISO/IEC/IEEE 42010:2011 recomienda que sean sistemas intensivos de software, es decir, sistemas en los que el desarrollo y/o integración de software son las consideraciones dominantes. Esto incluye sistemas que van desde aplicaciones de software individuales, sistemas de información, sistemas embebidos, líneas de productos de software, familias de productos de software, y sistemas de sistemas. En general, la guía puede ser usada para desarrollar descripciones de arquitectura para cualquier tipo de sistema de software.

Capítulo 6. Identificación y documentación de la información general de una descripción de la arquitectura

El estándar ISO/IEC/IEEE 42010:2011 señala que una descripción de arquitectura debe incluir la siguiente información general:

- La identificación del sistema de interés.
- Información complementaria que determine la organización y/o proyecto.
- Los resultados de las evaluaciones de la arquitectura o de las descripciones de arquitectura.

Es importante mencionar que el estándar ISO/IEC/IEEE 42010:2011 no especifica un formato o contenido detallado para tal información, sin embargo, como parte de este capítulo se propone una serie de consideraciones para identificar y documentar la información general de una descripción de arquitectura.

6.1. Identificación y documentación del sistema de interés

Dentro de la información general de una descripción de arquitectura, es importante identificar el sistema de interés, es decir, la entidad de la que nos interesa documentar su arquitectura.

Un sistema está situado en un ambiente o entorno, que es el contexto que determina los escenarios y circunstancias del sistema. Es importante que al identificar el sistema de interés, se identifique el ambiente en el que se sitúa, el cual puede contener a su vez otros sistemas.

Lo relevante de identificar el sistema de interés y el ambiente en el que se sitúa, es que proporcionan información indispensable requerida durante el proceso de arquitectura. Por ejemplo, en actividades como el diseño y la documentación arquitectónica, se requiere conocer información indispensable sobre los interesados del sistema y sus intereses, los cuales pueden ser definidos por el ambiente del sistema.

Al realizar el proceso de arquitectura de un sistema, resulta indispensable que se tenga identificado el sistema de interés, sin embargo, es probable que no se haya identificado totalmente el ambiente en el que se sitúa.

Identificar el sistema de interés, consiste en conocer información esencial sobre el mismo, por lo que se propone identificar la siguiente:

- Nombre con el que se identifica el sistema en el proyecto u organización.
- Los objetivos generales del sistema.
- El alcance del sistema.

El sistema de interés puede ser parte de un proyecto, el cual puede consistir en la creación, actualización, mejora o alguna otra actividad sobre el sistema. Por esto, es

común que la información esencial sobre el sistema de interés puede encontrarse en documentos tales como el “Plan de proyecto”¹⁵.

Es común que el sistema de interés sea parte de otro sistema, es decir, que sea un subsistema. También es frecuente que el sistema de interés sea un conjunto de subsistemas. Por estas razones, se recomienda identificar los límites del sistema de interés con el fin de no confundirlo con otros sistemas que pueden formar parte de su entorno.

Identificar el sistema de interés consiste también en determinar el ambiente o entorno donde se sitúa, por lo que para identificarlo, se propone considerar las influencias de desarrollo, técnicas, operacionales, organizacionales, políticas, económicas, legales, regulatorias, ecológicas y sociales de dicho entorno. Para identificar tales influencias, se requieren identificar las circunstancias que se producen en torno a cada una de ellas, las que además, deben estar comprobadas por escrito, a través de personas u otros medios fiables. Se propone que para obtener dicha información sean consultados documentos o fuentes de información sobre:

- Principios y políticas de la organización.
- Objetivos, misión y metas de la organización y/o del sistema.
- Normas que regulan el sistema y la organización.
- Estado social o económico de la organización.
- Sistemas relacionados al sistema de interés.

No es necesario incluir toda la información sobre el entorno del sistema como parte de la descripción de arquitectura, sin embargo, es recomendable conocerla e identificarla para documentar otros aspectos arquitectónicos fundamentales. Por ejemplo, al identificar el entorno de un sistema y las influencias sobre el mismo, se puede reconocer más fácilmente a los intereses e interesados clave del sistema, tema principal del capítulo 7.

Se recomienda que la información sobre el sistema de interés sea incluida al inicio de la descripción de arquitectura. Tal información puede ser una descripción breve del sistema y el entorno donde se sitúa. Por otra parte, se recomienda agregar las referencias a documentos o fuentes de información relacionadas con el sistema y su entorno, en la sección de referencias (ver sección 6.2.15.). Lo anterior es con el fin de reducir la duplicación de información y propiciar que la descripción de arquitectura sea concreta.

En el apéndice “D” de este trabajo se presenta una plantilla (plantilla 1) para documentar descripciones de arquitectura, que cuenta con un apartado (sección 2) para documentar el sistema de interés.

¹⁵ Documento formal y aprobado usado para guiar la ejecución y control del proyecto.

6.2. Identificación y documentación de información complementaria

La información complementaria en una descripción de arquitectura permite tener un mayor control sobre la misma, ya que incluye aspectos sobre su elaboración, revisión, actualización, y en general, sobre su administración.

La información complementaria en una descripción de arquitectura es importante ya que proporciona ayuda al lector (interesados en la arquitectura del sistema) para entender y utilizar más eficientemente dicho producto de trabajo.

El estándar ISO/IEC/IEEE 42010:2011 especifica que la información complementaria de una descripción de arquitectura debe ser especificada por la organización y/o proyecto. Debido a lo anterior, como parte de esta guía no se impone un conjunto particular de elementos de información complementaria, si no que se presenta un conjunto amplio de elementos que pueden ser considerados dependiendo de la organización y/o proyecto en cuestión. Tales elementos son presentados y explicados en las siguientes secciones (6.2.1 a 6.2.17), y es importante mencionar que algunos de ellos son especificados por el estándar “*Systems and software engineering, Content of life-cycle information products (documentation)*” (ISO/IEC/IEEE 15289:2011), que define el contenido general para documentos de descripción.

También es importante mencionar que en el apéndice “D” de este trabajo se presenta una plantilla (plantilla 1) para documentar descripciones de arquitectura, que cuenta con apartados (secciones 1 y 9) para documentar su información complementaria.

6.2.1. Fecha de emisión y estado

Como parte de la información complementaria, se puede incluir la fecha en la que fue emitida la descripción de arquitectura, y cuyo formato, puede ser especificado por la organización y/o proyecto.

Se propone también, incluir el estado en el que se encuentra la descripción de arquitectura. Algunos estados que pueden ser considerados son los siguientes:

- En proceso: no concluida.
- Concluida: concluida pero no revisada ni aprobada.
- Revisada: concluida y revisada, pero no aprobada.
- Aprobada: concluida, revisada y aprobada.

6.2.2. Autor(es)

Los autores que intervienen en la elaboración de la descripción de arquitectura también pueden ser incluidos como parte de la información complementaria. Es importante que los datos de tales autores puedan ser identificados fácilmente, ya que se puede requerir contactarlos para consultar algún aspecto sobre la descripción de arquitectura. Debido a esto, se recomienda incluir los siguientes datos:

- Nombre completo del autor.

- Rol o puesto en la organización y/o proyecto.
- Alguna forma de contacto (correo electrónico, página web, número telefónico, etc.)

6.2.3. Revisor(es)

La información del personal que realiza la revisión de la descripción de arquitectura también puede ser considerada como parte de la información complementaria. Se recomienda incluir los siguientes datos para cada revisor:

- Nombre completo del revisor.
- Rol o puesto en la organización y/o proyecto.
- Alguna forma de contacto (correo electrónico, página web, número telefónico, etc.).

6.2.4. Autoridad aprobatoria

La “autoridad aprobatoria” es el personal u organización que acredita la descripción de arquitectura, y puede ser considerada como información complementaria en una descripción de arquitectura. Si es el caso, se recomienda agregar los siguientes datos:

- Nombre completo de la autoridad aprobatoria.
- Rol o puesto en la organización y/o proyecto, u organización si es el caso.
- Alguna forma de contacto (correo electrónico, página web, número telefónico, etc.).

6.2.5. Organización que emite

La organización que emite la descripción de arquitectura también puede ser parte de la información complementaria a considerar. Si es el caso, se recomienda incluir los siguientes datos:

- Nombre de la organización.
- Contacto (página web, correo electrónico, número telefónico, etc.).

6.2.6. Historial de cambios

El historial de cambios permite llevar un control sobre los cambios realizados a la descripción de arquitectura, por lo que se recomienda ampliamente incluirlo como elemento de información complementaria. Se recomienda agregar los siguientes datos:

- Nombre de quién o quienes realizaron algún cambio en la descripción de arquitectura.
- Fecha en la que el cambio fue realizado.
- Una descripción sobre el cambio realizado.

6.2.7. Tabla de contenido

Una tabla de contenido puede ser parte importante de una descripción de arquitectura, ya que da a conocer al lector lo que presenta dicho documento, facilitando la búsqueda de

información. Se recomienda que la tabla de contenido incluya cada sección de la documentación, la página donde se encuentra, y una pequeña sinopsis de la información que puede ser encontrada en ella.

6.2.8. Introducción

La introducción en una descripción de arquitectura permite preparar o introducir al lector a dicho documento, dando un panorama general del mismo. Este elemento de información complementaria es recomendable, y se propone que incluya:

- El objetivo de la descripción de arquitectura.
- Una pequeña sinopsis de cómo está organizada la información.
- Un panorama general de la descripción de arquitectura.

6.2.9. Resumen

Un resumen que extraiga brevemente el contenido de la descripción de arquitectura puede resultar muy útil y es altamente recomendable como elemento de información complementaria ya que permite ahorrar esfuerzo a su audiencia.

Un resumen también puede explicar la relación que existe entre la descripción de arquitectura y otros documentos, tales como documentos de requerimientos, de análisis, de diseño, y el Plan del proyecto.

El resumen puede ser generado al finalizar la elaboración de la descripción de arquitectura, cuando se conozca el contenido exacto de la misma.

6.2.10. Alcance

El alcance en una descripción de arquitectura permite especificar los límites de la misma, aclarando lo que incluye, y lo que no, por lo que se recomienda incluirlo como parte de la información complementaria.

6.2.11. Contexto

El contexto permite definir el conjunto de circunstancias en las que se encuentra la descripción de arquitectura. Puede incluirse como parte de la información complementaria en forma de texto, en el que se mencionen las influencias principales sobre la descripción de arquitectura en aspectos técnicos, de desarrollo, políticos, organizacionales, económicos, legales, regulatorios o sociales.

6.2.12. Información sobre control de versiones y administración de la configuración

Puede resultar útil incluir información sobre las versiones existentes de la descripción de arquitectura, las cuales surgen de los diversos cambios que experimenta dicho documento.

Es probable que una de las actividades realizadas durante el ciclo de vida de un sistema sea la administración de la configuración, en la cual se establecen las pautas para

gestionar los cambios realizados sobre los productos de trabajo de software, tales como las descripciones de arquitectura. Debido a esto, durante las actividades de administración de la configuración pudo haber sido definida la siguiente información:

- Lugar o repositorio, físico o electrónico, donde se almacenan las descripciones de arquitectura y se lleva a cabo el control de sus versiones. Es importante, que tal repositorio sea accesible para la audiencia de las descripciones de arquitectura.
- Mecanismo de nombrado para las descripciones de arquitectura, que puede incluir el formato de su nombre y versión.
- Definición de líneas base, es decir, las especificaciones de una descripción de arquitectura que sirvan como base para actividades posteriores.
- La definición del proceso para la solicitud de cambios y sus procedimientos.
- Definición del proceso para la incorporación de cambios a las descripciones de arquitectura.
- Definición de reportes sobre cambios en las descripciones de arquitectura.
- Definición de una herramienta para controlar los cambios de las descripciones de arquitectura.

En una descripción de arquitectura, como parte de su información complementaria, se propone incluir los siguientes datos sobre administración de la configuración:

- Versión de la descripción de arquitectura.
- Referencias a los repositorios donde se administran las versiones de las descripciones de arquitectura.
- Referencias a documentos de administración de la configuración, las cuales pueden ser incluidas en la sección de referencias de una descripción de arquitectura (ver sección 6.2.15).

Se recomienda la revisión de los documentos sobre administración de configuración para saber cómo realizar el nombrado, almacenamiento físico o digital, el proceso de cambios, o el uso de herramientas para el control de las versiones de una descripción de arquitectura.

6.2.13. Uso

Las descripciones de arquitectura deben ser usadas correctamente por sus interesados, por esta razón, se propone incluir las especificaciones sobre su uso. Se propone incluir la siguiente información:

- Los interesados para los que fue elaborada la descripción de la arquitectura.
- Una relación de las secciones que pueden ser de interés para cada interesado o tipo de interesado.
- Las secciones que pueden ser consultadas primero para entender mejor la descripción de arquitectura.
- Escenarios que ejemplifiquen como los interesados pueden hacer uso de la descripción de arquitectura según sus intereses. Un ejemplo es el siguiente: “Si una persona de mantenimiento requiere conocer las unidades de software a modificar, debe consultar las vistas modulares, sección 3.5”.

6.2.14. Glosario

Un glosario puede facilitar el uso de una descripción de arquitectura, ya que es un compendio de términos poco conocidos, difíciles de interpretar o poco comunes, los cuales requieren alguna explicación. Por esta razón, se recomienda altamente incluirlo como elemento de información complementaria. El tipo de elementos que se propone agregar al glosario son los siguientes:

- Términos técnicos difíciles de inferir.
- Siglas poco comunes.
- Palabras cuyo significado puede confundirse en el contexto de una descripción de arquitectura.

El glosario de un documento comúnmente es agregado al final del mismo, por lo que se recomienda situarlo al final de una descripción de arquitectura.

6.2.15. Referencias

En una descripción de arquitectura se recomienda agregar las referencias a documentos o fuentes de información que sean de utilidad para entender mejor el contenido de la descripción de arquitectura. Algunos ejemplos de documentos a los que se puede hacer referencia son los siguientes:

- Documentos técnicos u organizacionales.
- Documentos de administración de la configuración.
- Políticas y normas.
- Documentos de requerimientos.
- Documentos de diseño.

En distintas secciones de una descripción de arquitectura, puede resultar más conveniente hacer referencia a documentos o fuentes de información en lugar de agregar su contenido explícito, esto con el fin de evitar la duplicidad de información. Por esta razón, es altamente recomendable agregar un apartado de referencias en una descripción de arquitectura, que puede ser situado al final de dicho documento.

6.2.16. Notaciones

En una descripción de arquitectura se puede agregar una definición de las notaciones que utiliza, es decir, de las convenciones empleadas para elaborar cada uno de sus elementos, tales como vistas y puntos de vista arquitectónicos.

6.2.17. Apéndices

En una descripción de arquitectura se puede incluir un apartado de apéndices para presentar información adicional, tal como información sobre estilos o patrones arquitectónicos, políticas, estándares, guías, revisiones, u otra documentación relacionada con la descripción de arquitectura.

6.3. Documentación de resultados de evaluaciones arquitectónicas

El estándar ISO/IEC/IEEE 42010:2011 establece que como parte de la información general de una descripción de arquitectura, se deben incluir los resultados de las evaluaciones de la arquitectura y/o de la descripción de arquitectura.

El objetivo de incluir esta información, es indicar si la arquitectura plasmada satisface los requerimientos de sus interesados, y si la descripción de arquitectura está bien elaborada y cuenta con el contenido necesario para entender y reflejar tal arquitectura.

Esta información puede ser especificada y generada al finalizar el proceso de documentación arquitectónica, cuando la descripción de arquitectura pueda ser evaluada.

6.3.1. Resultados de la evaluación de la arquitectura

Evaluar una arquitectura es asegurar que el sistema construido a partir de la misma satisface las necesidades de sus interesados.

La evaluación de la arquitectura está relacionada con la descripción y documentación de la misma, y puede ser realizada desde etapas tempranas en el desarrollo de un sistema.

La evaluación de una arquitectura produce un reporte, cuya forma varía de acuerdo al método usado, sin embargo, comúnmente incluye información que responde dos tipos de preguntas:

- ¿La arquitectura es adecuada para el sistema de interés? Para responder esta pregunta, se pueden considerar dos criterios:
 - Si satisface las metas de calidad, es decir, si el sistema al que representa la arquitectura es suficientemente rápido, con restricciones de seguridad, etc.
 - Si el sistema puede ser construido usando los recursos disponibles, tales como presupuesto y personal.
- En caso de que existan varias propuestas de arquitectura, ¿cuál de ellas es la más adecuada para el sistema en cuestión?

Es importante que los resultados de una evaluación arquitectónica sean registrados clara y formalmente para evitar malos entendidos sobre los problemas encontrados y las decisiones resultantes o cambios realizados. No hacerlo, puede llevar a ignorar mejoras y reducir la confianza de los interesados en la arquitectura y en el arquitecto. Algunos enfoques para registrar los resultados de las evaluaciones pueden ser los siguientes (Rozansky y Woods 2005):

- Minutas de reuniones: proporcionan información sobre las discusiones realizadas en torno a la arquitectura y sus resultados.
- Registros de decisión: registran las decisiones de arquitectura hechas según cierto razonamiento, e identifican las partes que están de acuerdo.

- Registros de revisión: utilizan un enfoque bien definido para registrar revisiones de arquitectura, permitiendo la documentación estándar de problemas y soluciones.
- Reportes de evaluación: permiten documentar las salidas de ejercicios de evaluación de la arquitectura en una forma estándar y estructurada para ser más útil para sus interesados. Permiten identificar y conservar los beneficios de los ejercicios y las lecciones aprendidas del proceso de evaluación.
- Documentos de cierre de sesión: registran acuerdos de decisiones particulares.

Al realizar una descripción de arquitectura, probablemente ya se haya realizado alguna evaluación de la arquitectura, o tal vez, la descripción de la arquitectura se utilice para realizarla. Sin embargo, lo que debe ser incluido en una descripción de arquitectura, es el resultado de la evaluación.

En esta guía se propone agregar, como parte de la información general de una descripción de arquitectura, la siguiente información sobre resultados de evaluaciones arquitectónicas:

- Respuesta a la pregunta: ¿La arquitectura plasmada en la descripción de arquitectura es adecuada para el sistema al que representa?.
- Una lista de verificación¹⁶ que permita identificar los requerimientos o intereses de los interesados del sistema que son cubiertos por la arquitectura del mismo. En la tabla 6.1 se muestra un pequeño ejemplo.

Requerimiento	Cubierto por la arquitectura (Si / No)
Alto rendimiento	Si
Alta disponibilidad de información	No
Escalable	Si
Módulo de administración de datos personales	Si

Tabla 6.1 Ejemplo de una lista de verificación con intereses arquitectónicos.

- Indicar si el sistema es viable, es decir, si puede ser construido con los recursos disponibles.
- Referencias a reportes o documentos más detallados sobre resultados de validaciones o evaluaciones de la arquitectura. Estas referencias pueden ser incluidas en la sección de referencias de la descripción de arquitectura (ver sección 6.2.15).

6.3.2. Resultados de evaluación de la descripción de arquitectura

Además de agregar los resultados de la evaluación de la arquitectura, se deben incluir los resultados de evaluar la descripción de arquitectura.

¹⁶ Método que permite valorar el estado de algún proceso o elemento mediante la verificación de una serie de requisitos.

Evaluar una descripción de arquitectura permite validar que esté elaborada correctamente y verificar que satisface lo prescrito por el estándar ISO/IEC/IEEE 42010:2011. Además, la evaluación de una descripción de arquitectura previene la propagación de defectos en los documentos creados subsecuentemente y facilita la corrección de errores.

Debido a que esta guía se basa en las especificaciones del estándar ISO/IEC/IEEE 42010:2011, uno de sus principales intereses es que la descripción de arquitectura cuente con el contenido especificado por dicho estándar, por lo que se recomienda agregar, como parte de la validación de una descripción de arquitectura, una lista de verificación como la presentada en la tabla 6.2.

Sección de la descripción de arquitectura	Contenido de la sección	¿Documentado? (si/no)
Información general	Información sobre el sistema de interés.	
	Información complementaria (fecha, resumen, etc.).	
	Resultados de evaluaciones arquitectónicas.	
Interesados	Interesados clave del sistema.	
Intereses	Intereses clave del sistema.	
Relación entre interesados e intereses	Relación entre interesados e intereses.	
Puntos de vista	Intereses enmarcados por el punto de vista.	
	Los interesados típicos para los intereses enmarcados por el punto de vista.	
	Tipos de modelo usados por el punto de vista.	
	Para cada tipo de modelo identificado, los lenguajes, notaciones, convenciones, técnicas de modelado, métodos analíticos y otras operaciones que pueden ser usadas en los modelos de ese tipo.	
	Referencias a fuentes de información.	
Vistas arquitectónicas	Información complementaria que es especificada por la organización y/o proyecto.	
	Punto de vista que la gobierna.	
	Modelos arquitectónicos.	
	Asuntos conocidos en la vista con respecto al punto de vista que la gobierna.	
Reglas de correspondencia	Análisis de consistencia de las vistas y modelos arquitectónicos.	
	Correspondencias entre los elementos de la descripción de arquitectura.	
	Reglas de correspondencia aplicables a los elementos de la descripción de arquitectura.	
Razonamiento y decisiones arquitectónicas	Razonamiento de las decisiones tomadas.	
	Decisiones tomadas.	

Tabla 6.2 Lista de verificación para una descripción de arquitectura con base en lo especificado por el estándar ISO/IEC/IEEE 42010:2011.

Además de la lista de verificación propuesta, se recomienda agregar el resultado de la evaluación de otras características de la descripción de arquitectura, tales como su consistencia.

Se propone incluir la respuesta a las siguientes preguntas tomadas de “*Documenting Software Architectures*” (Clements et al. 2003) (ver sección 14.2):

- ¿La descripción de arquitectura es consistente con los interesados que la usarán?
 - ¿Se Identificó el conjunto correcto de interesados clave y sus intereses?
 - ¿Se seleccionó el conjunto correcto de vistas y puntos de vista?
 - ¿Se incluyó la información adecuada sobre su organización para encontrar fácilmente cierta información?
- ¿La descripción de arquitectura es consistente consigo misma? Lo que significa estar libre de errores de ambigüedad y contradicción.
 - ¿Se incluyeron mapeos entre vistas?
 - ¿Se Incluyeron las inconsistencias entre las vistas?
 - ¿Se incluyeron las afirmaciones tales como hechos, heurísticas, requerimientos, decisiones, etc.?
 - ¿Se aseguró que no existan dos términos que signifiquen lo mismo o que un término se use para dos cosas diferentes?
 - ¿Se aseguró que no se repitiera información innecesaria en lugares diferentes en una descripción de arquitectura?
- ¿La descripción de arquitectura es consistente con una buena forma?
 - ¿La estructura y secciones de la documentación están definidas a través de plantillas o un estándar?
 - Para cada vista, ¿se proporcionó su definición?, es decir, elementos, relaciones, y propiedades.
 - ¿Contiene razonamiento, restricciones, entorno y alternativas rechazadas que puedan ser útiles?
 - ¿Incluye rastros de requerimientos incluidos?
 - ¿Incluye como ejercer variabilidades?
- ¿La descripción de arquitectura es consistente con la arquitectura que describe?
 - ¿La descripción de arquitectura es precisa?
 - ¿La descripción de arquitectura está actualizada?
 - ¿La descripción de arquitectura está completa?

Finalmente, se recomienda agregar las referencias a documentos o fuentes más detalladas sobre la evaluación de la descripción de arquitectura, las cuales pueden incluirse en la sección de referencias de la descripción de arquitectura.

Es importante mencionar que en el apéndice “D” de este trabajo se presenta una plantilla (plantilla 1) para documentar descripciones de arquitectura, que cuenta con un apartado (sección 8) para documentar las evaluaciones arquitectónicas. Por otra parte, se puede consultar el capítulo 14 de este trabajo para obtener más información sobre la evaluación de las descripciones de arquitectura.

Capítulo 7. Identificación y documentación de interesados e intereses

La arquitectura de un sistema debe ser creada para satisfacer los intereses del sistema, y por otra parte, la documentación de una arquitectura requiere ser elaborada de tal forma que resulte útil para sus interesados, es decir, que pueda ser entendida y empleada correctamente por ellos. Es por lo anterior, que los interesados e intereses que deben ser considerados durante el proceso de arquitectura son aquellos que resultan fundamentales para la arquitectura del sistema de interés, además, dichos interesados e intereses deben ser incluidos como parte de la descripción de arquitectura (ISO/IEC/IEEE 42010:2011).

Para elaborar descripciones de arquitectura, resulta fundamental conocer los interesados e intereses fundamentales para la arquitectura que es descrita, ya que en dicho proceso, se realizan algunas actividades que se basan en conocerlos (como selección de Lenguajes de Descripción de Arquitectura y definición de puntos de vista).

El estándar ISO/IEC/IEEE 42010:2011 especifica que en una descripción de arquitectura deben ser considerados, y si es el caso, identificados los siguientes interesados: usuarios, operadores, adquiridores, propietarios, proveedores, desarrolladores, constructores y mantenedores.

El estándar ISO/IEC/IEEE 42010:2011, también especifica que deben ser considerados, y si es el caso, identificados los siguientes intereses: propósitos del sistema; la idoneidad de la arquitectura para lograr los propósitos del sistema; la factibilidad de construir y desplegar el sistema; los riesgos e impactos potenciales del sistema para sus interesados a lo largo de su ciclo de vida; y la mantenibilidad y capacidad de evolución del sistema. Además, el estándar especifica que una descripción de arquitectura debe asociar cada interés identificado con los interesados que tienen tales intereses.

Las especificaciones del estándar ISO/IEC/IEEE 42010:2011 para la identificación y documentación intereses e interesados son generales, y no definen aspectos concretos como la granularidad de los intereses ni la manera en cómo éstos se relacionan con otras cuestiones del sistema. Además, el estándar proporciona un conjunto de intereses e interesados a considerar poco detallado, y no especifica técnicas para identificarlos, analizarlos o documentarlos. Por tal motivo, en este capítulo se presenta una guía para realizar dichas actividades.

Es importante mencionar que aunque la identificación de intereses e interesados comúnmente es parte de actividades de requerimientos, se requiere realizar la identificación de aquellos interesados e intereses considerados fundamentales para la arquitectura de un sistema, ya que en ocasiones éstos no son identificados completamente en actividades de requerimientos. En las próximas secciones de este capítulo se presentan técnicas para identificar dichos intereses e interesados, y una propuesta para su documentación. También es importante considerar que en el apéndice "D" de este trabajo se presenta una plantilla (plantilla 1) para documentar descripciones de arquitectura, que cuenta con un apartado (sección 3) para documentar intereses e interesados.

7.1. Técnicas para la identificación de interesados

Es importante que durante el proceso de arquitectura, se identifique puntual y específicamente a todas aquellas personas, organizaciones, o grupos de ellas, que tengan intereses considerados fundamentales para la arquitectura del sistema de interés, es decir, identificar a aquellos cuyos intereses estén fuertemente relacionados con el diseño, la descripción, y la evaluación de la arquitectura, y para los cuales se realice la documentación de la misma.

La identificación de interesados consiste en saber quiénes son, y el rol que desempeñan dentro del proceso arquitectura.

Se debe entender que no hay una lista general de los interesados de una arquitectura, esto depende del negocio y la evolución del sistema, y de otros factores como los cambios de opinión de los interesados con respecto a sus intereses. Por otra parte, es importante señalar que no existe un procedimiento claro y sistemático para la identificación de interesados clave de la arquitectura un sistema, si no que existen distintas estrategias. A continuación se presentan dos de ellas: la técnica de exploración y la lluvia de ideas.

7.1.1. Técnica de exploración

Consiste en hacer una exploración del panorama de los interesados en el entorno del sistema, considerando las dimensiones “vertical”, “horizontal”, y “fuera de la organización” (Majumdar, Rahman y Rahman 2013):

- **Dimensión vertical.** El tamaño de la mayoría de los equipos que intervienen en el desarrollo de un proyecto a gran escala, comprende desde los líderes de alto rango de la organización, a los usuarios finales del sistema. En esta dimensión, es importante explorar los principales actores en cada nivel de la organización y/o proyecto, y entender la manera en cómo cada uno de ellos se relaciona con el sistema de interés, y en particular con su arquitectura. Esta exploración es un paso fundamental para la identificación de interesados.
- **Dimensión horizontal.** A través de una organización y/o proyecto, es probable que haya muchos actores cuyas funciones se relacionen de diferentes maneras con el sistema de interés, y en específico con su arquitectura. En esta dimensión, se debe explorar e identificar cada interesado "funcional" que represente una perspectiva y tipo de experiencia diferente para el sistema, y en particular para su arquitectura.
- **Dimensión “fuera de la organización”.** Además de los interesados que colaboran en una organización y/o proyecto para el desarrollo de un sistema, se debe considerar a aquellos que representan entidades externas al mismo, y que puedan estar relacionados con su arquitectura, por ejemplo, los proveedores.

7.1.2. Lluvia de ideas

Una de las técnicas que se puede aplicar para identificar los interesados en la arquitectura de un sistema, es la lluvia de ideas. Esta técnica es útil cuando se requiere generar un gran número de ideas en relación a un problema a resolver (Dybá, Dingsøy y Moe 2004).

La lluvia de ideas consta de 7 pasos (Dybá, Dingsøy y Moe 2004), y plantea una participación activa de todos los presentes. Una de las ventajas de esta técnica, es que es una forma rápida y efectiva para la obtención de un gran número de ideas. A continuación se presentan las 7 actividades que conforman esta técnica, y se proponen consejos prácticos para identificar a los interesados principales de la arquitectura de un sistema:

Paso 1

Presentar la técnica y asegurarse que todos acepten sus reglas.

Paso 2

Establecer el tiempo de la sesión, que se sugiere sea de 5 a 15 minutos.

Paso 3

Realizar las preguntas para la generación de ideas, en este caso, para la identificación de interesados. Se propone considerar personas, organizaciones, o grupos de éstas, que representen respuesta a las siguientes preguntas:

- *¿Quién controla los cambios del sistema?*
Pensar en quién es responsable de los cambios de un sistema, los cuales se ven fuertemente reflejados en su arquitectura.
- *¿Quién diseña el sistema de interés y los sistemas relacionados con el mismo?*
Una de las actividades que se realiza durante el proceso de arquitectura es su definición o diseño, que está relacionado con otros tipos de diseño del sistema (base de datos, interfaz de usuario), por lo que se requiere pensar en las personas que realizan tales diseños como posibles interesados en la arquitectura.
- *¿Quién controla los recursos destinados para el sistema?*
Conocer quién controla los recursos destinados para el sistema permite conocer quiénes son los que controlan los recursos monetarios, tecnológicos o humanos, destinados para realizar distintas actividades, tales como las arquitectónicas.
- *¿Quién tiene habilidades especializadas para realizar actividades sobre el sistema?*
Si existen personas con habilidades especializadas para desarrollar, implementar, evaluar, o mantener el sistema, éstos deben considerarse como posibles interesados, ya que pueden tener intereses fundamentales a considerar para diseñar la arquitectura. Además, al conocer tales interesados, se puede generar documentación útil, especializada y adecuada de la arquitectura para los mismos. Por ejemplo, es útil saber si los interesados tienen conocimiento o experiencia con Lenguajes de Descripción de Arquitectura o descripciones de arquitectura.

- *¿Quién usa el sistema?*
Es importante identificar las personas de las que surgen los requerimientos del sistema, ya que éstos serán cubiertos por la arquitectura. Por otra parte, es útil conocer la audiencia de la documentación arquitectónica, la cual puede ser empleada para entender o usar el sistema.
- *¿Quién está directamente envuelto en los procesos del sistema?*
Se debe identificar a las personas relacionadas con los procesos de análisis, diseño, desarrollo, implementación, evaluación, despliegue o mantenimiento del sistema, ya que dichas personas están fuertemente relacionadas con las actividades de arquitectura, y en especial con su documentación. Para conocer más sobre estos posibles interesados, se puede consultar el catálogo de interesados presentado en el apéndice “A”.
- *¿Quién debe administrar, introducir, operar, o mantener el sistema después de su despliegue?*
Saber quiénes son los interesados que deben implementar, evaluar o mantener el sistema, permite generar documentación arquitectónica útil, adecuada y especializada para ellos.
- *¿Quién es responsable de los negocios o procesos que soporta el sistema?*
Las metas de negocio y los objetivos específicos del sistema son intereses que impactan directa y fuertemente la arquitectura de un sistema, por lo que conocer a las personas relacionadas con tales intereses resulta fundamental para actividades de diseño, documentación y evaluación de la arquitectura.
- *¿Quién tiene un interés financiero en el sistema?*
Una arquitectura bien documentada puede proporcionar un valor agregado al cliente o negocio del sistema, por lo que se deben considerar como interesados en la arquitectura aquellos con intereses financieros en el sistema.
- *¿Quién restringe el sistema?*
Las restricciones del sistema se ven reflejadas directamente en su arquitectura, por lo que conocer quiénes las establecen resulta importante para el diseño de la arquitectura.
- *¿Quién está interesado potencialmente en nuevos mercados, nuevas tecnologías, nuevos clientes?*
Se debe considerar aquellos interesados en nuevos mercados y tecnologías, y considerar también a clientes que tengan alta influencia con el sistema y cuyos intereses puedan considerarse para el proceso de arquitectura.

Algunas cuestiones adicionales que se propone considerar son las siguientes:

- Pensar en las actividades que se realizan en la organización, en sus objetivos, en quiénes las realizan, y si están relacionadas con la arquitectura del sistema.
- Asegurarse de considerar una amplia diversidad de interesados. Se puede tomar como referencia el catálogo de interesados que se propone en este trabajo (ver apéndice A: “Interesados”).

- Considerar los entornos tecnológicos, de desarrollo, de negocio, operacionales, organizacionales, políticos, económicos, legales, regulatorios, ecológicos y sociales para la identificación de interesados.
- Considerar que un buen interesado está informado (cuenta con información, experiencia y el entendimiento necesario para tomar buenas decisiones), es comprometido (habilitado para participar en el proceso y preparado para tomar decisiones difíciles), autorizado (las decisiones tomadas no serán posteriormente revertidas), y es representativo (si existe un grupo de interesados similares, tomar uno representativo) (Rozansky y Woods 2005).

Paso 4

Realizar la colecta de ideas (interesados) de todos los participantes. Puede ser de una forma estructurada (todos presentan sus ideas en turno) o no estructurada (se expresan las ideas a medida que éstas se presentan). Para este paso, se puede realizar una de las preguntas presentadas en el paso anterior a la vez, o permitir responderlas abiertamente, colocando las preguntas a la vista de todos.

Paso 5

Documentar las ideas, que en este caso son los interesados identificados, los cuales se pueden anotar en una pizarra o colocarlos en tarjetas adhesivas, lo importante es que estén a la vista de todos los participantes.

Paso 6

Consiste en clarificar las ideas para asegurar que todos las entiendan adecuadamente. Para este paso se propone hacer un repaso rápido de todos los interesados identificados para asegurar que se entiendan claramente y sin ambigüedades.

Paso 7

Consiste en eliminar las ideas duplicadas, si es el caso. Se propone descartar los interesados duplicados, o aquellos que tengan un rol idéntico, es decir, las mismas responsabilidades e intereses.

Finalmente, tanto los resultados de la lluvia de ideas como los de la técnica de exploración pueden ser registrados de manera que se visualicen fácilmente, y si se requiere, se puede realizar su análisis o priorización. La tabla 7.1 muestra un ejemplo de la manera en que los interesados obtenidos de aplicar alguna técnica pueden ser registrados.

Identificador	Rol	Nombre
I01	Analista	Margarita Pérez
I02	Desarrollador	Agustín López

Tabla 7.1 Ejemplo del registro de interesados.

Es importante mencionar que ambas técnicas (exploración y lluvia de ideas) se pueden combinar, y también se puede usar alguna técnica adicional, lo importante es entender quiénes son los interesados, sus intereses, y cómo se relacionan con la arquitectura del sistema.

7.1.3. Priorización de interesados

Una vez identificados los interesados de la arquitectura del sistema, es útil hacer un análisis más profundo para entender mejor su relevancia y la perspectiva que ofrecen, y para establecer su prioridad en función de sus intereses.

La priorización de interesados consiste en saber cuáles de los interesados identificados son los más importantes y esenciales, es decir, cuáles son los interesados clave que tienen intereses considerados fundamentales para la arquitectura del sistema.

Para la priorización de interesados se pueden aplicar diversas técnicas de priorización, tales como el “*Analytical Hierarchy Process*” (Avesani et al 2005), el “*Case-Based Ranking*” (Berander y Jonsson 2006) y el “*Hierarchical Cumulative Voting*” (Mulla y Girase 2012), las cuales no son técnicas usadas únicamente para la priorización de interesados, pero pueden resultar útiles para dicha tarea.

Una de las cuestiones esenciales para realizar la priorización de interesados, es establecer criterios adecuados, mediante los cuales se establezca cuáles interesados son los más importantes. Algunos de los criterios que se propone considerar son:

- a) Contribución o valor: indica si el interesado tiene información, consejo o experiencia, que pueda ser de utilidad para el proceso de arquitectura del sistema.
- b) Participación: indica qué tanta participación tiene el interesado en actividades de arquitectura.
- c) Influencia: indica cuánta influencia tiene el interesado en la arquitectura.
- d) Riesgo de omisión: indica el impacto que tiene para la arquitectura la omisión del interesado.

De acuerdo con los criterios antes mencionados, los interesados se pueden priorizar asignando valores, o utilizando clasificaciones previamente establecidas. Ambas técnicas se presentan a continuación.

7.1.3.1. Asignación de valores

Una forma de priorizar interesados es asignando valores numéricos (entre 1 y n) o relativos (alto, medio, bajo), considerando algunos criterios como los antes mencionados. De esta forma, se puede identificar a aquellos interesados clave para la arquitectura del sistema. En la tabla 7.2 se muestra un ejemplo de la asignación de valores a interesados.

Interesado	Contribución	Valor	Influencia
I01	Alto	Alto	Medio
I02	Bajo	Medio	Bajo
I03	Alto	Medio	Medio

Tabla 7.2 Ejemplo de asignación de valores a interesados.

En la tabla 7.2 se puede notar que el interesado I01 es el más relevante.

7.1.3.2. Clasificación

La priorización de interesados también se puede realizar utilizando ciertas clasificaciones establecidas basadas en ciertos criterios. Algunas clasificaciones que se propone considerar son las siguientes (Majumdar, Rahman y Rahman 2013):

- Dependiendo del nivel de participación de los interesados en actividades de arquitectura, se pueden identificar tres tipos:
 - Obligatorios: deben ser incluidos, o en caso contrario, el éxito de la arquitectura se ve amenazado.
 - Opcionales: sus necesidades no amenazan al éxito de la arquitectura. No necesariamente deben ser considerados.
 - Bueno tener: no influyen en la arquitectura si no son considerados.
- Dependiendo de qué tan importantes son los interesados, o qué tanto son afectados por el sistema y su arquitectura, se pueden considerar dos tipos:
 - Primarios: aquellos que son directamente significativos o potencialmente afectados por las actividades de arquitectura.
 - Secundarios: aquellos que son indirectamente afectados por la arquitectura del sistema, o para quienes el impacto de ésta no es tan relevante.
- Dependiendo el riesgo que causa la omisión de los interesados, se pueden identificar tres tipos:
 - Críticos: si la omisión del interesado y sus intereses causa la inutilización de la arquitectura, o causa un gran impacto en actividades de la misma.
 - Mayores: si la omisión causa un impacto significativo y negativo en la arquitectura.
 - Menores: si la omisión tiene un impacto marginal en la arquitectura del sistema.

Finalmente, se recomienda registrar el razonamiento y decisiones arquitectónicas sobre la identificación y priorización de los interesados para realizar la documentación del razonamiento arquitectónico (ver capítulo 13).

7.2. Técnicas para la identificación de intereses

Muchos de los intereses de un sistema son identificados en actividades de requerimientos, sin embargo, existe la probabilidad de que no todos los intereses fundamentales para la arquitectura del sistema hayan sido identificados al llegar a actividades de documentación arquitectónica. Por esta razón, el objetivo de esta sección es presentar técnicas de apoyo para identificar aquellos intereses que sean considerados fundamentales para la arquitectura del sistema de interés (requerimientos arquitectónicamente significativos). En las técnicas presentadas, se consideran las siguientes formas en que se manifiestan los intereses:

- Metas y estrategias de negocio, que dirigen a atributos de calidad y otros intereses de impacto para la arquitectura del sistema.
- Atributos de calidad (como mantenibilidad y capacidad de evolución), que requieren ser exhibidos por el sistema y por lo tanto, considerados en su arquitectura.
- Propósitos y expectativas del sistema, los cuales deben ser cubiertos por su arquitectura.
- Riesgos e impactos potenciales del sistema para sus interesados a lo largo de su ciclo de vida.
- Restricciones que deben ser consideradas para desarrollar el sistema, y que impactan su arquitectura.

7.2.1. Identificación de intereses arquitectónicos a través de documentos

La forma más obvia de reunir los intereses arquitectónicos es a través de los documentos de requerimientos o de historias de usuario. Sin embargo, no siempre hay información suficiente por el tipo y nivel de documentación que se maneja en cada proyecto, o porque en ocasiones el diseño de la arquitectura debe comenzar cuando los requerimientos aún están en proceso de ser analizados, o están “en el aire” (Bass, Clements y Kazman 2012).

Cuando existen documentos de requerimientos, se pueden extraer de ellos algunos intereses arquitectónicos. A continuación se proponen algunas consideraciones para su identificación:

- Considerar los requerimientos que puedan afectar el diseño de alto nivel del sistema, ya que la arquitectura es un diseño de este tipo.
- Considerar los requerimientos que puedan ser cubiertos definiendo estructuras principales en la arquitectura, tales como módulos deseados por el sistema.
- Considerar los requerimientos que tengan que ver con el comportamiento principal del sistema, lo que es parte fundamental de su arquitectura.
- Considerar los requerimientos relacionados con patrones o estilos arquitectónicos, tecnologías, o alguna otra característica técnica, ya que éstas pueden determinar características importantes de la arquitectura del sistema.
- Considerar los objetivos principales del sistema, los cuales deben ser cubiertos por la arquitectura.
- Considerar las características del sistema que puedan ser reflejadas como atributos de calidad, por ejemplo, si hay un requerimiento que indique que el

sistema debe ser rápido, esto se puede interpretar como el rendimiento requerido por el sistema.

- Considerar los requerimientos que tengan gran relación con el ambiente en el que se desplegará el sistema, ya que pueden indicar las características deseadas por el sistema, y por tanto, aquellas que deben ser cubiertas por su arquitectura. Además, estos requerimientos pueden determinar restricciones tecnológicas u otras cuestiones a las que se requiera adaptar el sistema.
- Considerar los requerimientos que tengan relación con los objetivos o metas de negocio, ya que pueden determinar parte del alcance del sistema y características de su arquitectura.
- Considerar los requerimientos relacionados con restricciones del sistema, ya que la arquitectura será diseñada bajo éstas limitaciones.
- Considerar los requerimientos relacionados con una posible evolución del sistema, ya que la arquitectura debe diseñarse para soportar tal característica.
- Considerar los requerimientos relacionados con respuestas esperadas por el sistema, ya que estos pueden conducir a identificar atributos de calidad.

Se debe tomar en cuenta que aunque existan, los documentos de requerimientos usualmente fallan en dos formas: no todo lo que está plasmado en los documentos afecta a la arquitectura; y lo que puede resultar útil para la arquitectura puede no estar ni siquiera en el mejor documento de requerimientos (tal es el caso de los atributos de calidad u otros requerimientos no funcionales) (Bass, Clements y Kazman 2012).

7.2.2. Entrevistas con interesados

Hablar con los interesados es otra forma de identificar los intereses fundamentales de la arquitectura del sistema. Esta es la manera más segura de saber qué necesitan los interesados, ya que la información se obtiene de la fuente principal.

Si anteriormente se ha realizado la identificación y priorización de los interesados de la arquitectura, se cuenta con una lista de interesados clave que pueden proporcionar los intereses fundamentales para la arquitectura.

Las cuestiones que se propone abordar en una entrevista con los interesados son las siguientes:

- Las características más importantes deseadas por el sistema, tales como estructura y comportamiento, ya que serán cubiertas por su arquitectura.
- Las restricciones (tecnológicas, económicas, regulatorias, etc.) del sistema, ya que estas serán consideradas por su arquitectura.
- Los posibles cambios que podría experimentar el sistema, es decir, su posible evolución, y con ella, la evolución de su arquitectura.
- El ambiente tecnológico en el que el sistema será desplegado, lo que determina las características del diseño de su arquitectura, y las restricciones que serán consideradas por la misma.
- Aspectos de diseño, desarrollo, despliegue, evaluación, o mantenimiento del sistema, lo que puede generar intereses fundamentales para su arquitectura.
- Las principales metas de negocio, las cuales pueden conducir a intereses fundamentales para la arquitectura del sistema.
- Los objetivos sistema, que deben ser cubiertos por su arquitectura.

7.2.3. El “Quality Attribute Workshop”

Es un método para determinar atributos de calidad que consulta a los interesados del sistema. Este método complementa al método *Architecture Tradeoff Analysis Method* (ATAM) (Kazman, Klein y Clements 2000), y proporciona una manera de identificar los atributos de calidad importantes y clarificar los requerimientos del sistema antes que sea creada su arquitectura. El método cuenta con 8 pasos, y se recomienda una participación de 5 a 30 interesados.

PASO 1: Presentación del método e introducción

Los facilitadores describen la motivación y cada paso del método. Posteriormente los facilitadores y los interesados se presentan, indicando brevemente sus antecedentes, su rol en la organización, y su relación con el sistema que se está construyendo.

PASO 2: Presentación del negocio/misión

Un interesado representativo presenta los “*drivers*” de negocio y misión para el sistema. Los *drivers* arquitectónicos a menudo incluyen requerimientos de alto nivel, intereses de negocio/misión, metas, objetivos, y atributos de calidad.

Durante la presentación, los facilitadores escuchan cuidadosamente y capturan la información relevante que puede clarificar los atributos de calidad, los cuales serán refinados en pasos posteriores.

PASO 3: Presentación del plano arquitectónico

Un interesado técnico presenta, si existen, descripciones, dibujos, o artefactos de alto nivel del sistema. La información en esta presentación debe incluir los planes y estrategias para cubrir los requerimientos de negocio/misión; los principales requerimientos técnicos y restricciones; y diagramas de contexto, diagramas de alto nivel del sistema, y otras descripciones escritas existentes.

PASO 4: Identificación de drivers arquitectónicos

Los facilitadores consolidan las notas tomadas en los pasos 2 y 3, y posteriormente comparten su lista de los principales *drivers* arquitectónicos con los interesados, los cuales dan lugar a aclaraciones, eliminaciones o correcciones. El objetivo es llegar a un consenso de una lista “refinada” de *drivers* arquitectónicos que incluyan requerimientos de alto nivel, *drivers* de negocio, restricciones y atributos de calidad. La lista final de los *drivers* arquitectónicos ayudará a realizar una lluvia de ideas de escenarios.

PASO 5: Lluvia de ideas de escenarios

Los facilitadores revisan las partes de un buen escenario (estimulo, ambiente, y respuesta), y aseguran que cada escenario sea formado adecuadamente durante el proceso.

Cada interesado expresa un escenario, representando sus intereses con respecto al sistema en una forma *round-robin*¹⁷, con al menos 2 rondas. Los facilitadores se aseguran que al menos un escenario representativo exista para cada *driver* arquitectónico listado en el paso 4.

PASO 6: Consolidación de escenarios

Después de la lluvia de ideas de escenarios, los escenarios similares deben consolidarse cuando sea razonable. Para esto, los facilitadores piden a los interesados que identifiquen aquellos escenarios que tienen un contenido muy similar. Los escenarios similares se fusionan, siempre y cuando las personas que los propusieron estén de acuerdo.

PASO 7: Priorización de escenarios

Se asigna a cada uno de los interesados, un número de votos igual al 30% del número total de escenarios generados después de la consolidación. El número real de votos asignados a las partes interesadas se redondea a un número par de votos a discreción de los facilitadores. La votación se realiza de manera *round-robin* en dos rondas. Durante cada ronda, las partes interesadas asignan la mitad de sus votos. Las partes interesadas pueden asignar cualquier número de sus votos a cualquier escenario o combinación de escenarios. Los votos son contados, y los escenarios son priorizados.

PASO 8: Refinamiento de escenarios

Después de la priorización, y dependiendo de la cantidad de tiempo restante, los 4 o 5 escenarios prioritarios deben ser detallados, documentando lo siguiente:

- Los seis elementos de un escenario: estímulo, respuesta, fuente del estímulo, ambiente, artefacto estimulado, y medida de respuesta.
- Los objetivos de negocio/misión que se ven afectados por el escenario.
- Los atributos de calidad relevantes asociados con el escenario.

Además, se debe permitir a los interesados formular preguntas y plantear cuestiones relevantes al escenario. Las preguntas deben concentrarse en los aspectos de los atributos de calidad del escenario, y cualquier interés relacionado con lograr la respuesta establecida por el escenario.

Los resultados que se obtienen al realizar el *Quality Attribute Workshops* incluyen: una lista de *drivers* arquitectónicos, los escenarios en “crudo” (escenarios no refinados), la lista priorizada de los escenarios en “crudo”, y los escenarios refinados.

Esta información puede ser usada para diseñar la arquitectura de un sistema, y después de crear la arquitectura, los escenarios pueden ser utilizados como parte de una evaluación de la arquitectura de software. Además, los atributos de calidad identificados pueden ser documentados claramente en una descripción de arquitectura.

¹⁷ Método para seleccionar todos los elementos en un grupo de manera equitativa y en un orden racional.

Para más información sobre este método consultar el “*Quality Attribute Workshops (QAWs)*” (Barbacci et al. 2003).

7.2.4. Método PALM

El método “*Pedigreed Attribute eLicitation Method*” (PALM) (Bass, Clements y Kazman 2012), permite identificar y documentar metas de negocio. Puede ser usado para identificar requerimientos faltantes, o para descubrir información adicional de los requerimientos existentes. También puede ser usado para examinar atributos de calidad particularmente difíciles.

Este método utiliza una lista estándar de metas de negocio y un formato de escenarios para metas de negocio, y se lleva a cabo con la participación de arquitectos e interesados que puedan hablar sobre las metas de negocio de la organización en cuestión. A continuación se describen los siete pasos que lo conforman.

Paso 1: Presentación del método

Se presenta el método, el problema que soluciona, sus pasos, y los resultados esperados.

Paso 2: Presentación de los drivers de negocio

Sesión informativa sobre los *drivers* de negocio por parte de la administración del proyecto. Es normalmente una discusión prolongada que permite a los participantes formular preguntas acerca de las metas de negocio presentadas por la administración del proyecto.

Paso 3: Presentación de drivers arquitectónicos

Sesión de información sobre los *drivers* arquitectónicos por parte del arquitecto.

Paso 4: Obtención de las metas de negocio

Usando la lista estándar de metas de negocio presentada en la tabla 7.3 para guiar la discusión, se captura el conjunto de metas de negocio importantes para el sistema.

Categoría de meta de negocio	Descripción
Contribuir al crecimiento y continuidad de la organización.	En esta categoría, el sistema se desarrolla solo para la existencia de la organización.
Satisfacer objetivos financieros.	Incluye los ingresos generados o ahorrados por el sistema. El sistema es creado para ventas, o para proporcionar un servicio que genere un ahorro. Puede incluir objetivos financieros o individuales.

Satisfacer objetivos personales.	Las metas personales asociadas con la construcción del sistema, tales como mejorar la reputación o ganar experiencia.
Cumplir las responsabilidades con los empleados.	Los empleados son los relacionados con el desarrollo u operación del sistema. Las responsabilidades con ellos incluyen seguridad, capacitación, oportunidades, o carga de trabajo.
Cumplir las responsabilidades con la sociedad.	Cuando la organización sirve a la sociedad como parte del negocio.
Satisfacer responsabilidades con el estado.	Cuando el sistema es gubernamental, o tiene responsabilidades con el estado o país.
Satisfacer responsabilidades con accionistas.	Obligaciones de protección, y algunos tipos de conformidad regulatoria.
Administrar la posición del mercado.	Estrategias para incrementar o mantener las cuotas de mercado, varios tipos de protección de propiedad intelectual, o tiempo de venta.
Mejorar procesos de negocio.	Habilitar nuevos mercados, nuevos productos, o mejorar soporte para el cliente.
Administrar la calidad y reputación de productos.	Incluyen avisos, tipos de usuarios potenciales, calidad de productos existentes, y evaluación de soporte y estrategias.
Administrar cambios en factores del entorno.	Alentar a los interesados a considerar lo que puede cambiar en las metas de negocio para el sistema.

Tabla 7.3 Categorías de metas de negocio (Bass, Clements y Kazman 2012).

Las metas de negocio son elaboradas y presentadas como escenarios con los siguientes elementos:

- La fuente de la meta, es decir, las personas o documentos que proporcionan la meta de negocio.
- El sujeto de la meta, que es cualquier interesado en la meta.
- El objeto de la meta, es decir, cualquier entidad a la que se aplica la meta (individuos, organizaciones, sociedad).
- El ambiente o contexto para la meta de negocio, que puede ser legal, social, competitivo, del cliente, o tecnológico.
- Cualquier meta de negocio articulada por la fuente de la meta (ver tabla 7.1).
- La medida de la meta, que determinar si la meta fue lograda. Usualmente incluye una medida de tiempo, con el tiempo en el cual la meta debe ser lograda.

- *Pedigree* y valor. El *pedigree* indica el grado de confianza que la persona que declaró la meta tiene en ella. El valor puede ser expresado en qué tanto su propietario está dispuesto a invertir para lograrla, o su importancia relativa comparada con otras metas. La importancia relativa puede ser dada por un rango de 1 (más importante) a n (menos importante); en escala de 0 a 10; o con valores de “alto”, “medio” y “bajo”.

Posteriormente, se deben consolidar las metas de negocio parecidas o casi iguales para eliminar duplicación. Los participantes priorizan el conjunto resultante para identificar las metas más importantes.

Paso 5: Identificación de atributos de calidad potenciales de las metas de negocio

Para cada escenario de meta de negocio, se realiza una descripción de los atributos de calidad que pueden ayudar a lograrla.

Paso 6: Examinar la existencia de drivers arquitectónicos

Este paso da a los atributos de calidad su “*pedigree*”. Para cada *driver* arquitectónico mencionado en el paso 3, se identifican la(s) meta(s) de negocio que soporta. Se establece su “*pedigree*”, preguntando por la parte cuantitativa de la fuente. Por ejemplo ¿Por qué el tiempo de respuesta es “40 ms” y no “80 ms”?

Paso 7: Conclusión

Se realiza la revisión de resultados y la retroalimentación de los participantes. Se obtiene información sobre el ejercicio, y se escribe el informe final.

Los beneficios del uso de este método incluyen la clarificación de los atributos de calidad y la mejora de la documentación de arquitectura.

Para más información del método PLAM, consultar “*Software Architecture in Practice*” (Bass, Clements y Kazman 2012).

7.2.5. Árbol de utilidad

Esta técnica permite la traducción directa y eficiente de las metas de negocio de un sistema, en escenarios concretos de atributos de calidad. Los árboles de utilidad, ayudan a concretar, perfeccionar y priorizar las propiedades de calidad.

Para crear un árbol de utilidad, se comienza colocando la palabra “utilidad” en el nodo raíz (la utilidad es una expresión de la “bondad” general del sistema). Después, se listan los atributos de calidad que se requiere exhiba el sistema, y debajo de cada de ellos, se registra el “refinamiento” o detalle de los atributos de calidad. Estos “refinamientos” deben ser relevantes para el sistema.

Debajo de cada refinamiento, se coloca un Requerimiento Arquitectónicamente Significativo apropiado (usualmente expresado como escenario de atributo de calidad).

Algunos Requerimientos Arquitectónicamente Significativos expresan más de un atributo de calidad, y pueden aparecer en más de un lugar en el árbol.

Después del refinamiento, se realiza una priorización, la cual puede establecerse en una escala de 1 a 10, o ser una priorización relativa (alta, media o baja). La priorización se hace en 2 dimensiones: por la importancia de cada nodo para el éxito del sistema, y el grado de riesgos percibidos que supone el logro de este nodo (que tan fácil será logrado). Se pueden colocar las dos medidas en una tupla (Valor, Valor), tal como se muestra en la figura 7.1.

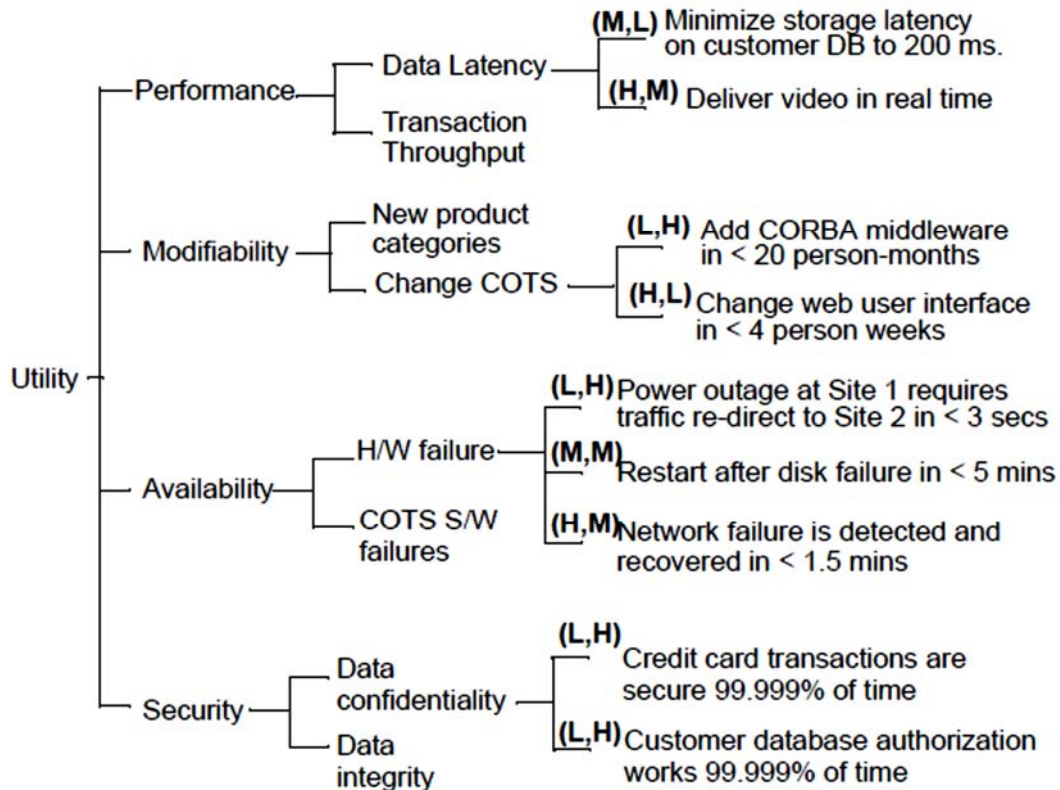


Figura 7.1 Ejemplo de un árbol de utilidad (Bass, Clements y Kazman 2012).

La creación de un árbol de utilidad sirve como guía para la priorización de los intereses arquitectónicos. Además, su elaboración ayuda a revisar si un atributo de calidad no se deriva de un Requerimiento Arquitectónicamente Significativo, y permite considerar aquellos atributos de calidad que después de la priorización, sean más importantes y requieran más atención.

Para mayor información sobre los árboles de utilidad, se puede consultar el “ATAM: Method for Architecture Evaluation” (Kazman, Klein y Clements 2000).

Adicionalmente, es importante mencionar que la lista de intereses derivados de aplicar cualquiera de las técnicas antes mencionadas, se puede proporcionar a los interesados con el fin de validarla.

Finalmente, se recomienda registrar el razonamiento y decisiones arquitectónicas sobre la identificación de los intereses para realizar la documentación del razonamiento arquitectónico (ver capítulo 13).

7.3. Documentación de interesados e intereses en una descripción de arquitectura

El objetivo de esta sección es presentar consideraciones para documentar intereses e interesados en una descripción de arquitectura con base en lo especificado por el estándar ISO/IEC/IEEE 42010:2011. Incluye la definición de elementos específicos de documentación que pueden ser de utilidad para todo el proceso de arquitectura.

Es importante considerar que en el apéndice “D” de este trabajo se presenta una plantilla (plantilla 1) para documentar descripciones de arquitectura que cuenta con un apartado (sección 3) para documentar intereses e interesados, y que es compatible con la forma en que se propone en esta sección.

7.3.1. Documentación de interesados

Los interesados que deben ser documentados en una descripción de arquitectura según el estándar ISO/IEC/IEEE 42010:2011 son aquellos que tienen intereses considerados fundamentales para la arquitectura del sistema de interés. Si al momento de documentar la arquitectura de software, éstos no han sido identificados, se pueden aplicar distintas técnicas, como las presentadas en la sección 7.1 de este trabajo.

La forma en que se propone documentar cada uno de los interesados fundamentales para la arquitectura, es registrando sus características principales. Se propone incluir algunas de las siguientes:

- **Identificador.** Clave específica para el interesado, la cual permita referirse al mismo en otras secciones de la descripción de arquitectura.
- **Nombre.** Nombre del individuo, organización, o grupo de los anteriores que represente a un interesado.
- **Tipo.** Se refiere a alguna clasificación de interesados que sea conveniente para el sistema. Por ejemplo, si el interesado es interno o externo a la organización o proyecto, o si es un interesado especializado técnicamente o no.
- **Rol.** Papel que desempeña el interesado en el proyecto o sistema, por ejemplo, si es un desarrollador, analista, cliente o usuario final. Se puede tomar como referencia el catálogo propuesto en el apéndice “A”.
- **Intereses.** Una breve descripción de los intereses del interesado, y que son fundamentales para la arquitectura del sistema.
- **Contacto.** Referencia para contactar al interesado.

La información anterior puede ser presentada en una tabla o estructura que resulte fácilmente leída y comprendida. En la tabla 7.4 se presenta un ejemplo.

Id	Nombre	Tipo	Rol	Intereses	Contacto
S01	Amalia Rosas	Interno	Desarrollador	Estructura y comportamiento arquitectónico	arosas@mcia.com
S02	Banco MMXS	Externo	Usuario	Desempeño	admin@mmxs.com

Tabla 7.4 Ejemplo del registro de interesados en una descripción de arquitectura.

7.3.2. Documentación de intereses

Los intereses que deben ser documentados en una descripción de arquitectura según el estándar ISO/IEC/IEEE 42010:2011, son aquellos que resultan fundamentales para la arquitectura del sistema de interés. A pesar de que algunos intereses, como las metas de negocio, no son requeridos como parte de una descripción de arquitectura según el estándar, es de suma importancia conocerlos para realizar otras actividades de arquitectura, tales como el diseño o evaluación, por esta razón se recomienda incluirlos como parte de la documentación arquitectónica.

A continuación, se presenta un conjunto de consejos y propuestas para documentar los distintos tipos de intereses que forman parte de una descripción de arquitectura.

7.3.2.1. *Propósitos del sistema*

Los propósitos del sistema son algunos de los intereses identificados más tempranamente en el ciclo de vida de desarrollo del sistema, y se pueden encontrar en documentos como el “Plan de proyecto” o documentos de requerimientos. Si este no es el caso, se puede recurrir a algunas técnicas para su identificación (ver sección 7.2).

Para documentar cada uno de estos intereses, se recomienda incluir un identificador o clave única, y la descripción del propósito. Además se recomienda citar la fuente de la que se obtuvo la información, si es que existe. Tal información puede ser incluida en una estructura, como el ejemplo de la tabla 7.5.

Id	Propósito
C01	El sistema debe permitir el alta de pólizas.
C02	El sistema implementará la generación automática de reportes financieros.

Tabla 7.5 Ejemplo de registro de los propósitos del sistema.

7.3.2.2. *Idoneidad o conveniencia de la arquitectura para lograr los propósitos del sistema*

Para documentar estos intereses se da por hecho que se tiene algún diseño o concepción de la arquitectura con la que se pueda validar si la misma es idónea para lograr los propósitos del sistema. Si la documentación de la arquitectura se hace al mismo tiempo que el diseño de la misma, se puede esperar para documentar estos intereses hasta que se tenga algún diseño de la arquitectura que se pueda evaluar. Lo ideal, es que la información requerida pueda ser obtenida de documentos que resulten de actividades de la evaluación de la arquitectura del sistema.

Para registrar estos intereses, se propone presentar una lista de verificación en forma de tabla, con los propósitos del sistema, definiendo si son cubiertos por la arquitectura, y si es el caso, una breve descripción de como la arquitectura los logra. Además, se propone incluir un identificador para cada interés. Esta información puede ser presentada en una estructura, como el ejemplo de la tabla 7.6.

Id	Propósito del sistema	Logrados por la arquitectura	Descripción
C11	Ser un portal web	Si	Patrón MVC
C12	Administrar seguridad de clientes	Si	Módulo de seguridad
C13	Apoyar en la generación de reportes	No	-----

Tabla 7.6 Ejemplo del registro de intereses.

Otra forma de documentar estos intereses, es únicamente colocar una sentencia que resuma sí, en general, la arquitectura es idónea o no para lograr los propósitos del sistema, tal como el ejemplo de la tabla 7.7.

Identificador	Interés
C14	La arquitectura es idónea para lograr los propósitos del sistema.

Tabla 7.7 Ejemplo del registro de intereses.

Si existen, pueden citarse los documentos relacionados con la evaluación de la arquitectura de donde se obtuvo la información.

7.3.2.3. Viabilidad o factibilidad de construir y desplegar el sistema

El determinar si un sistema es viable para ser construido y desplegado, comúnmente se realiza durante fases tempranas del sistema, y al llegar a las actividades de documentación de la arquitectura, esa información ya fue determinada. Sin embargo, en ocasiones puede recurrirse a la misma arquitectura para determinar si el sistema puede ser construido y desplegado viablemente. La información relacionada con la viabilidad de construir y desplegar el sistema puede ser presentada en forma de texto, o a través de una tabla como la 7.8.

Identificador	Interés
C21	Es viable construir el sistema.
C22	Es viable desplegar el sistema.

Tabla 7.8 Ejemplo del registro de intereses.

7.3.2.4. Riesgos e impactos potenciales del sistema para sus interesados a lo largo de su ciclo de vida

Los riesgos e impactos potenciales del sistema pueden haberse determinado antes de realizar la documentación de la arquitectura de actividades de administración de riesgos, y pueden encontrarse en documentos como el “Plan de proyecto”, o los “Planes de iteraciones”.

Algunas técnicas para identificar riesgos son el “*Mission Risk Diagnostic (MRD)*” (Alberts y Dorofee 2012), las lluvias de ideas, o la misma experiencia de expertos.

Los riesgos e impactos a considerar deben ser aquellos que resulten fundamentales para la arquitectura del sistema. Para documentarlos, se propone registrar los principales componentes de los mismos (Alberts y Dorofee 2012):

- El evento potencial que desencadena el riesgo.
- La condición o conjunto de circunstancias que propician el riesgo.
- La consecuencia o pérdida que se produce cuando ocurre el riesgo.

También se recomienda incluir algunas de las medidas para un riesgo (Alberts y Dorofee 2012):

- Probabilidad de que se produzca el riesgo.
- Impacto o pérdida que se produce cuando ocurre el riesgo.
- Exposición al riesgo, es decir, la magnitud del riesgo con base en los valores de la probabilidad e impacto.

Adicionalmente, puede incluirse las medidas de contención y/o contingencia para cada riesgo.

Los riesgos pueden registrarse indicando algunos de sus componentes y medidas, y un identificador para cada uno de ellos. Tal información puede presentarse en una tabla como la 7.9.

Id	Riesgo	Evento	Condición	Consecuencia	Probabilidad	Impacto	Exposición	Contención	Contingencia

Tabla 7.9 Registro de riesgos.

Para los impactos potenciales, ya sean positivos o negativos, se propone incluir la siguiente información:

- Identificador del impacto.
- Impacto potencial.
- Fase durante la cual se lleva a cabo el impacto en el ciclo de vida del sistema (implementación, pruebas, etc.).
- Naturaleza del impacto (positivo, negativo, directo o indirecto).
- Los aspectos o interesados del sistema que son impactados.
- Magnitud del impacto, que puede ser expresada en una escala de 1 a n, o ser una medida relativa (alta, media, baja).

La información sobre los impactos potenciales del sistema puede ser presentada en una tabla como la 7.10.

Id	Impacto	Fase	Naturaleza	Objeto	Magnitud
C41					
C42					

Tabla 7.10 Registro de impactos potenciales.

Por último, se recomienda incluir la fuente de los documentos de los que se obtienen los riesgos e impactos potenciales del sistema si es que estos existen.

7.3.2.5. *Atributos de calidad*

A pesar de que el estándar ISO/IEC/IEEE 42010:2011 únicamente especifica que sean consideradas la mantenibilidad y la capacidad de evolución del sistema, se propone documentar todos los atributos de calidad que sean fundamentales para la arquitectura del sistema de interés.

Es importante mencionar que uno de los objetivos primordiales de una arquitectura es lograrlos atributos de calidad, y uno de los principales usos de la documentación de la arquitectura es servir como base para el análisis, para asegurar que la arquitectura logra dichos atributos de calidad.

En una documentación de arquitectura, los atributos de calidad pueden estar presentes en muchas formas: como atributos de calidad incorporados a algún patrón o estilo arquitectónico usado, en elementos arquitectónicos que proporcionan un servicio y que comúnmente tienen atributos de calidad asociados, mapeos a requerimientos que muestran cómo se satisfacen los atributos de calidad, o como parte de los intereses de los interesados del sistema. Sin embargo, es importante documentarlos explícitamente para que sean más claros para la audiencia de una descripción de arquitectura.

Los atributos de calidad pueden obtenerse al aplicar diversas técnicas como las presentadas en la sección 7.2, y se propone documentarlos en una descripción de arquitectura incluyendo algunos de los siguientes datos:

- Identificador.
- Nombre (se puede usar el catálogo de atributos de calidad del apéndice “B”).
- Refinación o detalle del atributo de calidad. Pueden ser algunas de las subcategorías del atributo de calidad, o el interés detallado, por ejemplo, “confidencialidad de datos requerida del 100%”.
- Opcionalmente, se puede incluir algún escenario que aclare el atributo de calidad. Los seis elementos de un escenario son (Bass, Clements y Kazman 2012):
 - Fuente del estímulo, es decir, la entidad que genera el estímulo.
 - Estímulo, que es la condición que afecta al sistema.
 - Ambiente, o conjunto de las condiciones bajo las cuales se produce el estímulo.
 - Artefacto estimulado.
 - Respuesta, que es la actividad que resulta del estímulo.
 - Medida por la cual la respuesta del sistema será evaluada.

La información de cada atributo de calidad puede ser documentada mediante una tabla como la 7.11.

Id	Atributo de calidad	Detalle	Escenario
C51	Seguridad	Integridad de datos	<i>Fuente del estímulo:</i> <i>Estímulo:</i> <i>Ambiente:</i> <i>Artefacto:</i> <i>Respuesta:</i> <i>Medida:</i>
C52	Seguridad	Confidencialidad 100%	<i>Fuente del estímulo:</i> <i>Estímulo:</i> <i>Ambiente:</i> <i>Artefacto:</i> <i>Respuesta:</i> <i>Medida:</i>

Tabla 7.11 Registro de atributos de calidad.

7.3.2.6. Otros intereses

Si existen otros intereses que sean fundamentales para la arquitectura del sistema (metas de negocio, requerimientos funcionales, etc.), y no hayan sido considerados en las categorías anteriores, se recomienda ampliamente incluirlos.

NOTA: se propone que para asignar los identificadores de los intereses, se utilice la letra “C” (de *concern*) seguida de un número único para cada interés. De igual forma para los identificadores de los interesados, pero anteponiendo la letra “S” (de *stakeholder*).

NOTA: la empresa o proyecto puede determinar la información específica para documentar interesados e intereses, y puede ser diferente a la propuesta en este trabajo, sin embargo, lo importante es que se documente clara y detalladamente.

7.3.3. Documentación de la relación entre intereses e interesados

Especificar la relación entre intereses e interesados, además de ser requisito del estándar ISO/IEC/IEEE 42010:2011, sirve para realizar otras actividades de documentación.

La relación entre interesados e intereses es en general, de “muchos a muchos”, es decir, cada interesado puede tener varios intereses, y cada interés puede pertenecer a más de un interesado.

Se propone registrar la relación entre interesados e intereses en una tabla como la 7.12, indicando únicamente los identificadores de cada interesado e interés.

Interesados/Intereses	C01	C02	C03	C04	C05	C06
S01	X				X	
S02			X			
S03				X	X	
..	X		X			
Snn			X	X		

Tabla 7.12 Ejemplo del registro de la relación entre intereses e interesados.

Capítulo 8. Criterios para seleccionar Lenguajes de Descripción de Arquitectura

Una parte fundamental de las descripciones de arquitectura son los modelos arquitectónicos, los cuales son creados utilizando ciertas convenciones de modelado. En particular, este trabajo se enfoca en el uso de Lenguajes de Descripción de Arquitectura (LDAs) con notaciones formales como medio de modelado.

El estándar ISO/IEC/IEEE 42010:2011 no prescribe el uso de un Lenguaje de Descripción de Arquitectura en particular, debido a que depende de las características de la arquitectura del sistema en cuestión. La única regla que dicta el estándar es seleccionar las notaciones apropiadas para enmarcar los intereses relevantes que surgen del sistema.

Debido a que existe un gran número de Lenguajes de Descripción de Arquitectura que pueden ser utilizados para crear modelos arquitectónicos, es esencial contar con información que permita seleccionar los LDAs convenientes para modelar la arquitectura del sistema del interés. Por dicha razón, el objetivo de este capítulo es presentar información útil sobre criterios para seleccionar un LDA, y presentar información específica sobre algunos LDAs con el fin de apoyar en la selección de los mismos.

Los LDAs que se incluyen y consideran a detalle en este capítulo son los presentados en la tabla 8.1, y fueron seleccionados considerando lo siguiente:

- Cumplen con las especificaciones establecidas por el estándar ISO/IEC/IEEE 42010:2011 (ver sección 4.3).
- Cuentan con información suficiente para su uso, tal como manuales de referencia y tutoriales.
- Cuentan con una herramienta de software que facilita su uso.

LDA	Descripción general
AADL	El <i>Architecture Analysis and Design Language</i> (AADL) es un estándar internacional de la <i>Society of Automotive Engineers</i> (SAE), que permite el análisis y diseño de sistemas (sitio de AADL).
ACME	Es un LDA genérico y simple que puede ser usado como un formato de intercambio común para herramientas de diseño de arquitectura, y/o como base para desarrollar nuevas herramientas de análisis y diseño arquitectónico (sitio de ACME).
ByADL	Su nombre surge del acrónimo “Build Your ADL” o “construye tu ADL”. Es un <i>framework</i> que permite desarrollar nuevos LDAs, o extender LDAs existentes (sitio de ByADL).
Wright	LDA que proporciona una base para la descripción formal de arquitecturas de sistemas y familias de sistemas (sitio de Wright).
XADL	LDA para arquitecturas de software y sistemas, es un LDA estándar, extensible y basado en XML (sitio de xADL).

Tabla 8.1 Algunos Lenguajes de Descripción de Arquitectura.

Aunque el conjunto de lenguajes que se presenta a detalle en este capítulo es pequeño, los LDAs seleccionados permiten modelar una amplia variedad de arquitecturas de software con características y dominios particulares. Además, en este capítulo se incluyen los criterios que pueden ser tomados en cuenta para seleccionar cualquier Lenguaje de Descripción de Arquitectura. Tales criterios incluyen las características especificadas por el estándar ISO/IEC/IEEE 42010:2011, estilos arquitectónicos, herramientas, tecnologías relacionadas a los lenguajes, documentación y extensibilidad.

8.1. Características especificadas por el estándar ISO/IEC/IEEE 42010:2011

En las siguientes secciones se presenta información detallada sobre las características propias de cada LDA listado en la tabla 8.1, considerando las especificaciones del estándar ISO/IEC/IEEE 42010:2011 (ver sección 4.3.1). Además, se presentan las consideraciones para seleccionar cualquier LDA tomando en cuenta tales características.

8.1.1. Intereses arquitectónicos

Es importante que los LDAs seleccionados, permitan modelar la arquitectura del sistema considerando todos los intereses que sean fundamentales para dicha arquitectura. Estos intereses deben ser identificados con anterioridad de proceder con la selección de los LDAs (ver capítulo 7).

Algunos de los intereses considerados fundamentales para la arquitectura de un sistema, y que pueden ser modelados por un LDA, pueden incluir: propósitos del sistema, requerimientos funcionales y atributos de calidad. En la tabla 8.2 se presentan los intereses más significativos considerados por algunos LDAs.

LDA	Intereses
AADL	<p>Permite modelar la arquitectura de sistemas complejos y embebidos, y sus plataformas destino (hardware) mediante arquitecturas basadas en componentes. También permite modelar interacciones entre componentes arquitectónicos.</p> <p>Es útil en el modelado de sistemas con características especiales de eficiencia de desempeño (<i>performance</i>).</p> <p>Soporta la especificación de requerimientos de tiempo, el análisis de confiabilidad y seguridad, y la predicción y validación de características de tiempo de ejecución como disponibilidad y seguridad.</p>
ACME	<p>Permite describir la estructura arquitectónica de un sistema, así como estilos, tipos de elementos, y propiedades de elementos arquitectónicos. Aunque no es apropiado para todas las aplicaciones, permite describir arquitecturas de software fácilmente.</p> <p>Soporta la descripción jerárquica de arquitecturas, y la descripción de familias de arquitecturas. No soporta la descripción de comportamiento ni propiedades funcionales.</p>

ByADL	Permite extender un LDA con intereses de dominio específico, lo que permite modelar arquitecturas con cualquier interés requerido.
Wright	Se enfoca en la comunicación, comportamiento y estructura de arquitecturas de software. Permite analizar tanto sistemas, como familias de sistemas.
XADL	Soporta: el modelado para elementos de tiempo de diseño y tiempo de ejecución del sistema; tipos arquitectónicos; conceptos de administración de configuración avanzados, tales como versiones, opciones y variantes; arquitecturas de familias de productos; y “diff” ¹⁸ arquitectónico. Puede ser base para el desarrollo de LDAs de dominio específico, y permite crear esquemas para soportar intereses de dominio específico.

Tabla 8.2 Intereses de algunos Lenguajes de Descripción de Arquitectura.

8.1.2. Interesados

Un Lenguaje de Descripción de Arquitectura no define un conjunto específico de interesados como tal, sino que éstos pueden ser identificados a partir de los intereses particulares que son soportados por el lenguaje. Por ejemplo, si un LDA soporta la especificación de la plataforma física del sistema, los modelos generados con el LDA pueden ser útiles para algún encargado de desplegar el sistema.

En la tabla 8.3 se presentan algunos de los interesados más “obvios” que pueden ser considerados por algunos LDAs con base en los intereses que permiten modelar, sin embargo, se debe tomar en cuenta que tales interesados pueden variar dependiendo de cada organización o sistema. Se puede consultar el apéndice “A” que cuenta con un catálogo de posibles interesados.

LDA	Interesados
AADL	<ul style="list-style-type: none"> • Analistas, ya que AADL es un lenguaje que permite realizar distintos tipos de análisis, incluyendo rendimiento y seguridad. • Desarrolladores o implementadores interesados en conocer los principales componentes del sistema y sus interacciones. • Arquitectos de software. • Administradores de bases de datos ya que AADL es un lenguaje que cuenta con componentes que permiten modelar datos con distintos niveles de detalle. • Clientes o usuarios que requieran visualizar las principales características del sistema, tales como su plataforma destino. • Evaluadores, personal de mantenimiento, o personal de pruebas a los que les pueda interesar un modelo arquitectónico con características de seguridad, rendimiento, tiempo de ejecución o estructura.

¹⁸ *Diff* es una utilidad que permite la comparación de dos archivos a causa de las diferencias o cambios realizados a los mismos.

	<ul style="list-style-type: none"> • Ingenieros en sistemas cuyos intereses incluyan características estructurales y de tiempo de ejecución del sistema.
ACME	<ul style="list-style-type: none"> • Administradores de bases de datos cuyos intereses incluyan el modelado de datos en la arquitectura del sistema. • Administradores de líneas de productos ya que ACME permite la definición de familias de sistemas. • Administradores de sistemas con intereses en la estructura de un sistema. • Arquitectos de software. • Analistas con intereses arquitectónicos estructurales. • Clientes y usuarios interesados en la estructura de un sistema o familia de sistemas. • Comunicadores interesados en representaciones estructurales de la arquitectura de un sistema o familia de sistemas. • Desarrolladores, equipo de soporte, ingenieros en sistemas y responsables de integración interesados en los componentes principales de un sistema y en sus interacciones.
ByADL	Cualquier interesado que tenga intereses soportados por el LDA generado o extendido utilizando ByADL.
Wright	<ul style="list-style-type: none"> • Administradores de bases de datos con intereses en la representación de datos en la arquitectura del sistema. • Administradores de líneas de productos ya que Wright permite la especificación de estilos arquitectónicos que facilitan la definición de familias de sistemas. • Administradores de sistemas con intereses en la operación del sistema. • Arquitectos de software. • Analistas con intereses en el análisis estructural o de comportamiento del sistema. • Clientes y usuarios interesados en la estructura de un sistema o familia de sistemas y en su comportamiento. • Comunicadores interesados en representaciones estructurales y de comportamiento de la arquitectura de un sistema o familia de sistemas. • Desarrolladores, equipo de soporte, ingenieros en sistemas y responsables de integración interesados en los componentes principales de un sistema y en sus interacciones, y en el comportamiento del mismo.
XADL	Cualquier interesado cuyos intereses sean soportados por el LDA o por extensiones hechas al LDA. Algunos de los más significativos son administradores de líneas de productos, operadores, analistas y arquitectos.

Tabla 8.3 Posibles interesados soportados por algunos Lenguajes de Descripción de Arquitectura.

8.1.3. Tipos de modelo

Es importante que para seleccionar adecuadamente un LDA, se conozcan las convenciones, notaciones, técnicas de modelado y métodos analíticos que soporta, ya que se debe asegurar que el LDA sea adecuado para modelar los intereses fundamentales del sistema. Por esto, se recomienda consultar los documentos que definan la sintaxis, semántica, restricciones y demás consideraciones importantes sobre las notaciones del lenguaje que se considere seleccionar.

Además, es importante considerar que algunos LDAs cuentan con notaciones gráficas que facilitan su uso, y que existen LDAs cuyas notaciones pueden resultar complejas sobre todo para audiencias poco técnicas.

En la tabla 8.4 se presentan los tipos de modelo que soportan algunos LDAs.

LDA	Tipos de modelo
AADL	<p>Cuenta con una notación gráfica y una notación textual definidas por el estándar SAE AADL y sus extensiones. Además, cuenta con un formato de intercambio XML/XMI para soportar el intercambio de modelos creados con AADL.</p> <p>Las notaciones cuentan con sintaxis y semántica precisas.</p>
ACME	<p>El lenguaje cuenta con las siguientes características:</p> <ul style="list-style-type: none">• Una ontología arquitectónica de siete elementos básicos: componentes, conectores, sistemas, puertos, roles, representaciones, y rep-maps (mapas de representación).• Un mecanismo que soporta la asociación de información no estructural, usando sublenguajes definidos externamente.• Un mecanismo de tipos para abstraer estilos y modismos arquitectónicos reusables.• Un <i>framework</i> de semántica abierta que permite razonar sobre descripciones arquitectónicas. <p>Cuenta con una especificación en la forma <i>Backus-Naur</i> (BNF) para el <i>parser</i>¹⁹ Armani, que es un lenguaje que extiende ACME basándose en lógica de predicados de primer orden.</p>
ByADL	<p>La especificación de ByADL consiste principalmente de un metamodelo cuya semántica está dada en términos del núcleo de semántica DUALLy, un <i>framework</i> que proporciona interoperabilidad entre LDAs. ByADL está integrado y construido sobre DUALLy.</p> <p>El <i>framework</i> de ByADL permite extender un LDA por medio de “Mecanismos de Composición”, los cuales usan operadores de composición específicos que son aplicados a metamodelos extraídos de</p>

¹⁹ Analizador sintáctico que forma parte de un compilador.

	un repositorio de metamodelos. El lenguaje obtenido, es un lenguaje de modelado que consiste en: una sintaxis abstracta (por ejemplo, el metamodelo obtenido de la composición), un conjunto de sintaxis concretas (por ejemplo, notaciones gráficas y textuales para visualizar y editar modelos que conforman el metamodelo compuesto), y semánticas que describen el significado de las construcciones del lenguaje.
Wright	<p>Para especificar la estructura de una arquitectura, cuenta con una notación basada en componentes, conectores y configuraciones, que está especificada en la forma <i>Backus-Naur</i> (BNF).</p> <p>Para especificar el comportamiento arquitectónico, utiliza la notación "<i>Communicating Sequential Processes</i>" (CSP).</p> <p>Para especificar estilos arquitectónicos, hace uso de una notación basada en la lógica de predicados de primer orden.</p>
XADL	<p>XADL está definido como un conjunto de esquemas de XML (<i>eXtensible Markup Language</i>).</p> <p>XADL es una aplicación de xArchc, que es una herramienta que proporciona una notación de XML de núcleo común para arquitecturas de software.</p>

Tabla 8.4 Tipos de modelos de algunos Lenguajes de Descripción de Arquitectura.

8.1.4. Puntos de vista

Si un Lenguaje de Descripción de Arquitectura soporta uno o más intereses, puede permitir definir uno o más puntos de vista arquitectónicos. Es importante identificar los posibles puntos de vista soportados por el LDA que será utilizado. En la tabla 8.5 se presentan algunos de los puntos de vista más significativos que pueden permitir definir algunos LDAs con base en los intereses que soportan.

LDA	Puntos de vista
AADL	Puntos de vista relacionados intereses de tiempo real, sistemas embebidos, eficiencia de desempeño, despliegue, componentes e interacciones, ejecución de tareas, y concurrencia.
ACME	Estructural, y de componentes y conectores.
ByADL	Permite definir cualquier punto de vista que enmarque los intereses que son soportados por el LDA extendido o generado con ByADL.
Wright	Funcional, operacional, de comunicación, concurrencia, comportamiento.
XADL	Permite definir cualquier punto de vista que enmarque intereses soportados por el LDA, o aquellos que pueda soportar como resultado de una extensión al LDA. En especial, hace una separación entre una arquitectura de tiempo de diseño y de ejecución.

Tabla 8.5 Posibles puntos de vista de algunos Lenguajes de Descripción de Arquitectura.

8.1.5. Reglas de correspondencia

Un Lenguaje de Descripción de Arquitectura tiene asociadas reglas de correspondencia que relacionen algunos de sus elementos, tales como los componentes y conectores definidos por su ontología principal, los estilos que permite definir, y los intereses que permiten modelar.

En la tabla 8.6 se incluyen los elementos más significativos que pueden formar parte de las reglas de correspondencia de algunos LDAs.

LDA	Elementos de reglas de correspondencias
AADL	<ul style="list-style-type: none">• Tipos de componentes• Intereses• Componentes
ACME	<ul style="list-style-type: none">• Componentes• Familias de sistemas• Conectores• Diferentes LDAs
ByADL	<ul style="list-style-type: none">• Metamodelos• Elementos de modelos <p>En general, se puede establecer cualquier regla de correspondencia entre elementos del LDA extendido o del nuevo LDA generado.</p>
Wright	<ul style="list-style-type: none">• Estilos arquitectónicos• Componentes• Conectores• Estructura y comportamiento
XADL	<ul style="list-style-type: none">• Esquemas• Elementos arquitectónicos como componentes y conectores <p>En general, se puede establecer cualquier regla de correspondencia entre elementos del LDA.</p>

Tabla 8.6 Elementos de posibles reglas de correspondencia de algunos Lenguajes de Descripción de Arquitectura.

8.2. Estilos arquitectónicos

Es importante conocer los estilos arquitectónicos definidos por cada LDA, ya que esto ayuda a seleccionar LDAs que permitan modelar la arquitectura del sistema con base en sus características principales.

Si la arquitectura de un sistema está diseñada de acuerdo a un estilo o patrón arquitectónico, es importante asegurar que los LDAs seleccionados permitan modelarlos,

ya que los estilos arquitectónicos pueden permitir satisfacer los atributos de calidad del sistema y son parte fundamental de la arquitectura.

En la tabla 8.7 se presentan los estilos arquitectónicos que pueden ser definidos por algunos LDAs, así como sus elementos y características principales.

LDA	Estilos arquitectónicos
AADL	<p>Cuenta con abstracciones de componentes que se dividen en tres categorías: software de aplicación (hilos, grupos de hilos, procesos, datos, y subprogramas); plataforma de ejecución o hardware (procesadores, procesadores virtuales memorias, dispositivos, <i>buses</i> y <i>buses</i> virtuales); y composición (sistemas).</p> <p>Los componentes interactúan a través de interfaces definidas. Las conexiones establecen una de las siguientes interacciones: conexiones de puerto, conexiones de acceso a componentes, llamadas a subprogramas y conexiones de parámetro.</p>
ACME	<p>Una familia en ACME, incluye una colección de tipos de componentes y conectores que incorporan el vocabulario de un estilo arquitectónico. Los estilos incluyen reglas y restricciones específicas.</p> <p>ACME describe estructuras arquitectónicas, estilos y tipos arquitectónicos, y propiedades de los elementos arquitectónicos.</p> <p>Los siete elementos básicos que especifica ACME son: componentes, conectores, sistemas, puertos, roles, representaciones, y mapas de representación.</p>
ByADL	Permite definir cualquier estilo arquitectónico que sea definido por el LDA extendido o el nuevo LDA.
Wright	<p>Permite definir estilos específicos, como el "<i>Pipe and Filter</i>", y permite definir nuevos estilos arquitectónicos como subestilos de otro estilo.</p> <p>Las abstracciones que constituyen un estilo arquitectónico son: componentes, conectores y configuraciones.</p>
XADL	Debido a que es un lenguaje extensible, se pueden crear elementos que conformen un estilo específico a través de la generación de esquemas XML para intereses de dominio específico.

Tabla 8.7 Estilos arquitectónicos de algunos Lenguajes de Descripción de Arquitectura.

8.3. Herramientas

Es importante conocer las herramientas tecnológicas con que cuenta cada LDA, ya que determinan aspectos que pueden ser relevantes para el sistema y/o proyecto, tales como el soporte para análisis y modelado. En la tabla 8.8 se presentan las características de las herramientas tecnológicas de algunos LDAs.

LDA	Herramientas
AADL	<p>Algunas de las herramientas que soportan AADL son las siguientes:</p> <ul style="list-style-type: none"> • <i>Osate2</i>: herramienta de uso libre que soporta AADL versión 2. Tiene un editor textual completo para AADL y un conjunto de herramientas de análisis. Tiene soporte para meta-modelos en Eclipse. • <i>Ocarina</i>: herramienta <i>standalone</i>²⁰ para análisis sintáctico, generación de código, verificación de modelos, y análisis de planificación (<i>schedulability</i>). • Existe una herramienta comercial para software críticamente seguro que proporciona editores gráficos. <p>En particular, la herramienta <i>Osate2</i> soporta múltiples tipos de análisis (confiabilidad, eficiencia de desempeño (<i>performance</i>), seguridad, tiempo) y el seguimiento de modelado y análisis. Además, permite el análisis de la estructura del sistema y su comportamiento en tiempo de ejecución.</p>
ACME	<p>Cuenta con la herramienta <i>AcmeStudio</i>, que es un <i>plug-in</i>²¹ de Eclipse que proporciona un ambiente gráfico para editar y revisar descripciones en ACME. La versión <i>standalone</i> de <i>AcmeStudio</i> más reciente es 3.5.8 Eclipse 3.7.2., y cuenta con soporte para Windows 32 y 64, Linux GTK 64Bits y MacOS X 64 bits. La versión con características desplegadas más reciente de <i>AcmeStudio</i>, requiere Java 1.5, Eclipse 3.7.x, GEF 3.7.</p> <p><i>AcmeStudio</i> cuenta con un editor gráfico que permite editar diseños de familias existentes, o crear nuevas familias y tipos de elementos arquitectónicos. Permite crear nuevos estilos de diagramas basados en la convenciones de visualización que el usuario requiera definir. Integra un verificador de restricciones llamado “Armani” para revisar las reglas de diseño arquitectónico.</p> <p>Cuenta con “<i>AcmeLib Runtime</i>”, que es una biblioteca Java que proporciona un <i>parser</i> de línea de comandos y un verificador de tipos para ACME. Es usada por quienes no están interesados en trabajar con Eclipse pero necesitan usar Acme. Se requiere Java 1.5 para la versión más reciente.</p>
ByADL	<p>ByADL está implementado como un conjunto de <i>plug-ins</i> de Eclipse, que explotan el AMMA (<i>ATLAS Model Management Architecture</i>) y que pueden ser integrados con otras tecnologías MDE ya disponibles en la comunidad de Eclipse.</p> <p>Con ByADL, las composiciones de metamodelos son definidos usando el <i>ATLAS Model Weaver (AMW)</i>. AMW permite la definición de</p>

²⁰ Herramienta de software que puede trabajar sin conexión a la red.

²¹ Complemento de una herramienta tecnológica.

	<p>correspondencias entre metamodelos y ligas entre elementos de modelos. Los modelos y metamodelos son administrados por medio de EMF, que es un <i>framework</i> de modelado para generar código para construir herramientas y otras aplicaciones basadas en metamodelos.</p>
Wright	<p>Existe una versión preliminar de una herramienta que cuenta con un <i>parser</i> para descripciones en Wright, que permite traducir una descripción de Wright a la notación CSP (Communicating Sequential Processes), traducir descripciones de Wright a ACME y viceversa.</p> <p>Una vez que una descripción de Wright es traducida a CSP, se pueden usar verificadores comerciales de CSP (como FDR) para verificar consistencia y completitud. La herramienta es para la plataforma Linux, y la herramienta para traducir de Wright a Acme está disponible para Linux y Sun OS.</p>
XADL	<p>El ambiente más completo de xADL es ArchStudio, una herramienta de uso libre. En su versión 5, ArchStudio incluye herramientas para modelar, visualizar, analizar e implementar arquitecturas de software y sistemas. Está basado en la plataforma Eclipse bajo una licencia estilo BSD²². Los tres componentes de ArchStudio son: Java 2 Standard Edition (J2SE); Eclipse versión 4.3 o más; y ArchStudio en sí mismo.</p> <p>Incluye distribuciones binarias para Windows 32-/64-bit, Mac OS X 32-/64-bit y Linux 32-/64-bit</p> <p>ArchStudio incorpora Apigen, una herramienta usada para generar bibliotecas para el esquema núcleo de xADL.</p>

Tabla 8.8 Herramientas de algunos Lenguajes de Descripción de Arquitectura.

8.4. Plataformas y tecnologías relacionadas

Es importante identificar y considerar las plataformas o sistemas operativos que pueden soportar las herramientas del LDA que se seleccione. También se debe tomar en cuenta que existen LDAs que pueden ser utilizados en conjunto con otros *frameworks* o ambientes de desarrollo integrado (IDEs), lo que puede facilitar tareas de diseño y desarrollo.

En la tabla 8.9 se presentan las tecnologías y plataformas relacionadas con las herramientas de algunos LDAs.

²² Licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Es una licencia de software libre permisiva.

LDA	Plataformas y tecnologías relacionadas
AADL	Tiene soporte para meta-modelos en Eclipse, y hay soporte de Git ²³ para Osate2. La herramienta Osate2 puede ser usada en plataformas Linux, Windows y MacOS.
ACME	AcmeStudio es un <i>plug-in</i> de Eclipse. La versión <i>standalone</i> es para Windows 32 y 64, Linux GTK 64Bits y MacOS X 64 bits. La versión con características desplegables más reciente, requiere Java 1.5, Eclipse 3.7.x, GEF 3.7.
ByADL	Está implementado como un conjunto de <i>plug-ins</i> de Eclipse que pueden ser integrados con otras tecnologías MDE ya disponibles en la comunidad de Eclipse.
Wright	La herramienta está disponible para Linux y Sun OS.
XADL	Cuenta con soporte para herramientas comerciales de XML. ArchStudio está basado en la plataforma Eclipse bajo una licencia estilo BSD. Los tres componentes de ArchStudio son: Java 2 Standard Edition (J2SE), y la versión 1.7 de la máquina virtual de Java; Eclipse versión 4.3 o mayor; y ArchStudio en sí mismo. Incluye distribuciones binarias para Windows 32-/64-bit, Mac OS X 32-/64-bit y Linux 32-/64-bit.

Tabla 8.9 Plataformas y tecnologías relacionadas con algunos Lenguajes de Descripción de Arquitectura.

8.5. Documentación e información relacionada

Es importante conocer el tipo de documentación que existe para cada Lenguaje de Descripción de Arquitectura, ya que puede resultar de gran utilidad para comprender, usar, y ejecutar el lenguaje y sus herramientas tecnológicas.

La documentación de un LDA puede incluir cartas sintácticas, tutoriales, repositorios de ejemplos, y libros.

En la tabla 8.10 se presenta información sobre la documentación existente para algunos LDA y sus herramientas.

LDA	Documentación
AADL	El Software Engineering Institute (SEI) proporciona entrenamiento y asistencia técnica para el uso del lenguaje. Existen manuales de introducción y uso de AADL. Existe un tutorial para crear modelos con Osate2, y guías para instalar Git para Osate2. Cuenta con ayuda <i>online</i> para el uso de Osate2, que es de

²³ Software controlador de versiones.

	<p>uso libre.</p> <p>El SEI cuenta con un libro llamado “<i>Model-Based Engineering with AADL</i>”. Además, existe una “carta sintáctica” con la sintaxis precisa de AADL, documentación de los anexos del lenguaje (como el de error y el gráfico), múltiples artículos, y ejemplos de modelado. Para mayor información se puede consultar la página de AADL</p>
ACME	<p>El sitio de ACME ofrece una introducción a ACME a través de un repositorio de especificaciones de ACME, artículos y literatura técnica. Ofrece ejemplos de ACME, la biblioteca de desarrollo de herramientas de ACME (AcmeLib’s), herramientas de uso libre disponibles, y contribuciones de usuarios de ACME.</p> <p>Se puede descargar AcmeStudio desde la página de descargas de ACME, solo se requiere el registro por parte del usuario.</p>
ByADL	<p>La página del lenguaje proporciona información detallada sobre el <i>framework</i> y la herramienta de ByADL. También ofrece diversos artículos sobre ByADL. Permite descargar las versiones 1 y 2 de ByADL, el paquete de Eclipse con ByADL con las dependencias requeridas, los <i>plug-ins</i> para Eclipse, el código fuente de ByADL, un conjunto de proyectos de Eclipse de un caso de estudio, y el archivo “<i>readme</i>” de ByADL. Todo es de dominio libre.</p>
Wright	<p>En la página de Wright hay artículos y reportes técnicos relacionados al lenguaje. Hay una versión preliminar de la herramienta. Para descargarla se requiere el registro por parte del usuario.</p> <p>Cuenta con un tutorial para el análisis de descripciones en Wright que se encuentra en formato zip.</p>
XADL	<p>ArchStudio 5 es de uso libre, está basado en la plataforma Eclipse bajo una licencia estilo BSD. Además xADL cuenta con manuales para usar ArchStudio, y documentos sobre el lenguaje. También cuenta con información para crear esquemas en xADL.</p>

Tabla 8.10 Documentación de algunos Lenguajes de Descripción de Arquitectura.

8.6. Extensibilidad

Es importante considerar si el o los LDAs seleccionados cuentan con soporte para extender las características del lenguaje, ya que la arquitectura de un sistema puede evolucionar.

En la tabla 8.11 se presentan las características de extensibilidad de algunos LDA.

LDA	Extensibilidad
AADL	AADL permite declarar nuevas propiedades, componentes, u otros elementos de modelado. Utilizando las extensiones del lenguaje, pueden ser incluidos modelos y propiedades adicionales.
ACME	Se puede codificar un modelo de semántica en ACME para definir una

	familia, la cual permita definir nuevos estilos arquitectónicos.
ByADL	Proporciona mecanismos de composición que permiten crear un LDA personalizando, o extender LDAs existentes. Con ByADL un LDA puede ser extendido con: intereses de dominio específico, notaciones de análisis, nuevas vistas arquitectónicas, y metodologías y procesos que soporten otras actividades del ciclo de vida.
Wright	Se pueden extender estilos, o definir nuevos a partir de otros ya existentes.
XADL	Puede ser base para el desarrollo de LDAs de dominio específico. Es extensible, ya que permite crear nuevos esquemas para arquitecturas de dominio específico.

Tabla 8.11 Características de extensibilidad de algunos Lenguajes de Descripción de Arquitectura.

8.7. Ventajas sobre notaciones semiformales

Es importante estimar las ventajas de los Lenguajes de Descripción de Arquitectura seleccionados en comparación con notaciones semiformales considerando el costo de aprender y explicar a los interesados algunos de estos lenguajes contra el beneficio de su uso.

En la tabla 8.12 se presentan las ventajas de utilizar algunos LDA en comparación con otras notaciones semiformales o informales.

LDA	Ventajas
AADL	<p>AADL cuenta con una notación gráfica de modelado muy parecida a UML, por lo que su uso y aprendizaje no implicaría demasiado esfuerzo si se ha utilizado UML con anterioridad. Además, AADL cuenta con herramientas que permiten modelar gráficamente arquitecturas por lo que su uso no necesariamente implica el aprendizaje de su notación textual.</p> <p>Por otra parte, AADL es un lenguaje con grandes facilidades de análisis, lo que es una ventaja sobre notaciones menos formales. AADL permite especificar características de seguridad y rendimiento, y la estructura de un sistema de manera más detallada.</p> <p>Además, al contar con una notación textual formal, AADL permite especificar con mayor precisión una arquitectura, lo que reduce riesgos de problemas de comunicación e interpretación.</p>
ACME	<p>Cuenta con una notación gráfica sencilla, fácil de aprender, y parecida a UML. Además, cuenta con AcmeStudio, una herramienta que facilita el modelado gráfico y genera automáticamente el código correspondiente de un modelo y viceversa.</p> <p>Por otra parte, Acme permite la definición de familias de sistemas, que facilita especificación de restricciones y características propias de un</p>

	<p>conjunto de sistemas, una ventaja sobre notaciones menos formales.</p> <p>Además, Acme reduce la posibilidad de modelar un sistema con características incorrectas si dicho sistema pertenece a una familia de sistemas.</p>
ByADL	<p>Cuenta con características de extensibilidad que permiten crear un LDA personalizando, o extender LDAs existentes. Aunque su aprendizaje puede implicar mayor tiempo que una notación menos formal, uno de sus beneficios es poder modelar ciertas características según se requiera.</p>
Wright	<p>Es un lenguaje que en particular permite especificar de manera formal el comportamiento de un sistema, una gran ventaja sobre notaciones menos formales.</p> <p>Además, permite definir estilos arquitectónicos, lo que facilita la definición de un conjunto de sistemas con características en común.</p>
XADL	<p>Es extensible, lo que permite modelar características arquitectónicas según se requiera.</p> <p>Al estar basado en XML, puede ser utilizado con otras herramientas que soporten dicha notación.</p> <p>Cuenta con una notación gráfica fácil de aprender y con herramientas que permiten generar una especificación textual a partir de un modelo gráfico y viceversa.</p> <p>Permite realizar distintos tipos de análisis.</p>

Tabla 8.12 Ventajas de algunos Lenguajes de Descripción de Arquitectura respecto a notaciones menos formales.

Finalmente, se recomienda identificar y registrar el razonamiento y las decisiones sobre la selección de los Lenguajes de Arquitectura para realizar la documentación del razonamiento arquitectónico (ver capítulo 13).

Capítulo 9. Identificación y documentación de puntos de vista arquitectónicos

Los puntos de vista arquitectónicos son elementos importantes de las descripciones de arquitectura ya que representan una manera de “mirar” la arquitectura de un sistema. Por otra parte, los puntos de vista proporcionan un marco de trabajo para capturar conocimiento arquitectónico reusable, que puede ser utilizado para guiar la creación de un tipo particular de descripciones de arquitectura, o para para capturar conocimiento arquitectónico estratégico en una organización o en una comunidad arquitectónica.

El estándar ISO/IEC/IEEE 42010:2011 establece que un punto de vista arquitectónico debe especificar lo siguiente:

- Uno o más intereses que son enmarcados por el punto de vista. Un punto de vista “enmarca” un interés, cuando el punto de vista proporciona los medios para expresar dicho interés.
- Los interesados típicos para los intereses enmarcados por el punto de vista.
- Uno o más tipos de modelo usados por el punto de vista.
- Para cada tipo de modelo identificado, los lenguajes, notaciones, convenciones, técnicas de modelado, métodos analíticos y otras operaciones que pueden ser usadas en los modelos de ese tipo.
- Referencias a fuentes de información.

El estándar también especifica que un punto de vista arquitectónico debe incluir información sobre técnicas de arquitectura usadas para crear, interpretar o analizar una vista gobernada por ese punto de vista.

Es importante mencionar que el estándar no establece puntos de vista particulares para una descripción de arquitectura, sino que son seleccionados o definidos dependiendo de las características específicas de cada sistema (intereses, interesados, etc.).

Este capítulo tiene como objetivo proponer y presentar una guía para la selección y definición de puntos de vista arquitectónicos, actividad fundamental para la elaboración de una descripción de arquitectura. Por otra parte, es importante considerar que en el apéndice “D” de este trabajo se presenta una plantilla (plantilla 1) para documentar descripciones de arquitectura, que cuenta con un apartado (sección 4) para documentar los puntos de vista arquitectónicos. Además, en el mismo apéndice se presenta otra plantilla (plantilla 2) para documentar puntos de vista arquitectónicos como productos de trabajo independientes.

9.1. Actividades para identificar y seleccionar puntos de vista arquitectónicos

A continuación, se propone y presenta un conjunto actividades y consideraciones para la identificación y selección de puntos de vista arquitectónicos tomando en cuenta las especificaciones del estándar ISO/IEC/IEEE 42010:2011, el uso de Lenguajes de Descripción de Arquitectura y las características particulares de algún sistema de interés.

9.1.1. Identificar interesados e intereses

El primer paso para seleccionar o identificar los puntos de vista arquitectónicos, es contar con una lista de los interesados e intereses considerados fundamentales para la arquitectura del sistema. Tales intereses e interesados son esenciales debido a que los puntos de vista deben enmarcar uno o más intereses, y se requiere identificar los interesados a quienes pertenecen dichos intereses. Además, los interesados representan la audiencia de los puntos de vista y de las vistas elaboradas a partir de ellos.

En el capítulo 7 de este trabajo se proponen y presentan técnicas, consejos y consideraciones para identificar los intereses e interesados fundamentales para la arquitectura del sistema.

9.1.2. Seleccionar puntos de vista “candidatos”

Existen puntos de vista específicos que han sido propuestos y definidos por diversos enfoques arquitectónicos, y que podrían ser seleccionados para incluirse en la descripción de arquitectura. Entre los enfoques que han propuesto puntos de vista se encuentran el modelo de vistas de Kruchten 4+1 (Kruchten 1995), el modelo de puntos de vista del SEI (Clements et al. 2003), el “*Reference Model of Open Distributed Processing (RM-ODP)*”, y el modelo de cuatro vistas de Siemens (Soni et al. 1995).

Puede ser de gran utilidad considerar el uso de puntos de vista ya definidos, debido a que éstos pueden ajustarse a las características y aspectos de la arquitectura del sistema (intereses, tipos de modelo, etc.), permitiendo minimizar el trabajo de documentación de una arquitectura y reusar conocimiento arquitectónico.

En el apéndice “C” de este trabajo se presenta un catálogo que concentra puntos de vista definidos por algunos de los enfoques ya mencionados, y que pueden ser de ayuda para seleccionar e identificar puntos de vista arquitectónicos. Dicho catálogo presenta los intereses que son enmarcados por cada punto de vista, los modelos que utiliza cada punto de vista, y otras características esenciales que pueden ayudar a definir si el alcance de dichos puntos de vista es suficiente.

Algunos aspectos importantes a considerar para seleccionar los puntos de vista “candidatos”, a partir de algunos ya definidos, son los siguientes:

- **Incluir todos los intereses del sistema.** Se deben seleccionar puntos de vista que enmarquen los intereses considerados fundamentales para la arquitectura del sistema (ver sección 9.1.3).

- **Considerar la audiencia de cada punto de vista.** Debido a que los interesados del sistema son quienes hacen uso de la descripción de arquitectura, que incluye vistas y puntos de vista, se debe conocer para que qué clase de interesados está dirigido cada punto de vista. Además, se deben identificar los interesados del sistema (ver capítulo 7) para saber qué tanto conocimiento técnico tienen, y qué tanto requieren conocer sobre algún aspecto de la arquitectura, y así, seleccionar los puntos de vista adecuados. Se debe considerar que los modelos, que serán desarrollados a partir de los punto de vista, deben ser adecuados para cada interesado (Clements et al. 2003).
- **Considerar los Lenguajes de Descripción de Arquitectura seleccionados.** Es fundamental asegurarse que los puntos de vista definidos o elegidos, sean soportados por los Lenguajes de Descripción de Arquitectura que han sido seleccionados para modelar la arquitectura del sistema, ya que tales lenguajes serán utilizados para realizar el modelado arquitectónico del sistema. El capítulo 8 de este trabajo presenta, entre otras cosas, los puntos de vista, intereses, interesados y tipos de modelo que son soportados por algunos Lenguajes de Descripción de Arquitectura.

Adicionalmente, además de considerar los puntos de vista definidos por enfoques de documentación, se recomienda tomar en cuenta el conocimiento y experiencia que ha resultado de la documentación de otras arquitecturas. Esto es, considerar aquellos puntos de vista que han sido usados por la organización, o aquellos que han sido aplicados a sistemas con características similares al sistema de interés.

9.1.3. Asegurar incluir todos los intereses arquitectónicos

Al tener un conjunto de puntos de vista “candidatos”, se requiere asegurar que todos los intereses fundamentales de la arquitectura del sistema sean enmarcados por alguno de ellos. Para realizar esto, se propone utilizar una tabla cuyas filas sean los intereses del sistema, y sus columnas los posibles puntos de vista. En dicha tabla, se puede señalar cada celda cuyo interés (fila) sea enmarcado por un punto de vista (columna), de esta manera, se puede visualizar qué puntos de vista enmarcan qué intereses. En la tabla 9.1 se presenta un ejemplo.

	Punto de vista 1	Punto de vista 2	Punto de vista 3
Interés 1			X
Interés 2	X		
Interés 3		X	X

Tabla 9.1 Ejemplo para enmarcar intereses en puntos de vista.

Es importante asegurar que cada interés sea enmarcado por al menos un punto de vista, de tal modo que se cubran todos los intereses del sistema. Por otra parte, un punto de vista puede enmarcar uno o más intereses, y también es importante identificar los interesados a los que pertenecen dichos intereses.

Si al relacionar los intereses y puntos de vista se identifica que un interés no fue enmarcado por alguno de los puntos de vista, se propone considerar alguna de las siguientes opciones:

- Si es posible, se recomienda agregar a alguno de los puntos de vista que enmarcan otros intereses del sistema, características adicionales a sus notaciones de tal forma que permitan expresar el interés en cuestión.
- Si lo anterior no es posible o conveniente, probablemente se deban considerar otros puntos de vista diferentes a los propuestos inicialmente, que permitan enmarcar el interés en cuestión.
- Si no existe la manera de enmarcar el interés con algún punto de vista existente, tal vez se deba considerar definir un nuevo punto de vista (ver sección 9.2), aunque podría no ser muy recomendable debido a la carga de trabajo y complejidad que podría implicar.

9.1.4. Minimizar el conjunto de puntos de vista

Cuando se cuente con un conjunto de puntos de vista candidatos, es decir, un conjunto de puntos de vista que enmarquen todos los intereses del sistema, se debe analizar y considerar la posibilidad de minimizar el número de puntos de vista. Esto es conveniente debido a que cada punto de vista implica la elaboración de una vista arquitectónica, por lo que un número pequeño de puntos de vista minimiza el trabajo de documentación. Además, un número pequeño de puntos de vista puede reducir la complejidad e inconsistencia de la documentación de la arquitectura.

Para minimizar el número de puntos de vista arquitectónicos se propone realizar lo siguiente:

- Eliminar los puntos de vista cuyos intereses sean enmarcados por otros puntos de vista, asegurando que todos los intereses sean enmarcados por al menos un punto de vista y que los puntos de vista no resulten ser demasiado complejos o estén sobrecargados.
- Consolidar o fusionar dos puntos de vista arquitectónicos si se cumplen las siguientes condiciones:
 - Si ambos puntos de vista enmarcan un número pequeño o manejable de intereses, ya que un punto de vista sobrecargado puede aumentar la complejidad de las vistas arquitectónicas.
 - Si los intereses de ambos puntos de vista son compatibles, es decir, si los intereses de un punto de vista no son “anti-intereses” del otro.
 - Si los intereses de ambos puntos de vista pueden ser expresados utilizando notaciones compatibles, y esto no implique la elaboración de vistas demasiado complejas.

Según el estándar ISO/IEC/IEEE 42010:2011, un número típico de puntos de vista para muchos sistemas es entre 5 y 9, aunque se recomienda que se procure minimizar dicho número tanto como sea posible debido a la carga de trabajo y complejidad que puede implicar definir un número grande de puntos de vista.

9.2. Definición y documentación de puntos de vista

Un punto de vista puede ser definido y documentado como parte de una descripción de arquitectura, como parte de un *framework* de arquitectura, o individualmente (ISO/IEC/IEEE 42010:2011).

Definir y documentar un punto de vista arquitectónico como producto de trabajo independiente a una descripción de arquitectura, permite que dicho punto de vista sea usado en muchas descripciones de arquitectura.

En las siguientes secciones (9.2.1 a 9.2.10) se proponen consideraciones para definir y documentar cada uno de los elementos que componen a un punto de vista (tanto como producto de trabajo independiente, como parte de una descripción de arquitectura) con base en lo especificado por el estándar ISO/IEC/IEEE 42010:2011. Tales elementos incluyen el nombre, resumen, intereses, interesados, tipos de modelo, reglas de correspondencia, operaciones, ejemplos, y notas adicionales de un punto de vista.

Además es importante mencionar que en el apéndice “D” se presenta una plantilla (plantilla 1) para documentar descripciones de arquitectura, que cuenta con un apartado (sección 4) para documentar los puntos de vista arquitectónicos. Además, en el mismo apéndice se presenta otra plantilla (plantilla 2) para documentar puntos de vista arquitectónicos como productos de trabajo independientes. Ambas plantillas son compatibles con la forma en que se propone documentar un punto de vista en esta sección.

9.2.1. Nombre

El nombre de un punto de vista permite identificarlo de otros.

Cuando se define un punto de vista, se debe asignar un nombre adecuado considerando los intereses enmarcados por el punto de vista, los tipos de modelo usados, u otros aspectos arquitectónicos que sean relevantes para el punto de vista. Se recomienda asignar nombres pequeños, significativos, y que permitan identificar fácilmente al punto de vista.

Cuando se incluye un punto de vista ya definido como parte de una descripción de arquitectura (como los presentados en el apéndice C), se debe incluir el nombre común por el cual se identifica al punto de vista. Si es el caso, se pueden incluir todos los sinónimos por los que se conoce al punto de vista en cuestión. Esto permitirá identificarlo correctamente y no confundirlo con otros puntos de vista.

9.2.2. Resumen

Un resumen puede ser incluido en un punto de vista cuando éste se define y documenta como un producto de trabajo independiente a una descripción de arquitectura.

El resumen puede contener una breve visión del punto de vista, e incluir sus principales características, tales como los intereses generales que enmarca, interesados típicos, y a qué tipos de sistemas puede ser aplicado.

9.2.3. Intereses y anti-intereses

La documentación de un punto de vista, tanto como producto de trabajo independiente como parte de una descripción de arquitectura, debe incluir los intereses detallados que enmarca, los cuales pueden ser atributos de calidad u otros requerimientos del sistema.

Por otra parte, puede resultar útil documentar los intereses que no es apropiado incluir en un punto de vista, es decir, los “*anti-intereses*”. Por ejemplo, si un punto de vista está enfocado en intereses sobre estructura del sistema, podría ser inconveniente incluir intereses sobre algunas características de ejecución.

Para documentar tales intereses y anti-intereses de manera clara, se recomienda incluir su nombre y una descripción. Se puede hacer uso de una tabla como la 9.2.

Interés o anti-interés	Descripción

Tabla 9.2 Registro de intereses y anti-intereses en un punto de vista.

9.2.4. Interesados

La definición y documentación de un punto de vista debe incluir sus interesados típicos, es decir, aquellos a quienes pertenecen los intereses que enmarca el punto de vista. En otras palabras, se deben incluir los interesados del sistema que se espera sean usuarios o audiencias para las vistas elaboradas usando el punto de vista.

El estándar ISO/IEC/IEEE 42010:2011 establece que en una descripción de arquitectura se debe documentar la asociación entre los interesados del sistema y los intereses enmarcados por cada punto de vista. Para esto, se propone utilizar una tabla como la 9.3.

Nombre del punto de vista	
Interesado	Intereses

Tabla 9.3 Registro de interesados en un punto de vista.

9.2.5. Tipos de modelos

La definición y documentación de un punto de vista, tanto como producto de trabajo independiente como parte de una descripción de arquitectura, debe incluir los tipos de modelo usados para expresar los intereses que enmarca. Los tipos de modelo son importantes ya que determinan el vocabulario para construir vistas arquitectónicas.

Para cada tipo de modelo se deben definir los lenguajes, notaciones, convenciones, técnicas de modelado, métodos analíticos y/o otras operaciones para usarse en los modelos de este tipo (ISO/IEC/IEEE 42010:2011).

El estándar ISO/IEC/IEEE 42010:2011 no especifica un estilo particular para documentar los tipos de modelo de un punto de vista, sin embargo, indica que se pueden utilizar las siguientes formas:

- Mediante la especificación de un metamodelo que defina las construcciones base del tipo de modelo.
- Proporcionar una plantilla de modelo para ser llenada por los usuarios.
- Mediante la definición o referencia a un lenguaje de modelado.
- Mediante las operaciones que pueden ser aplicadas a los modelos de ese tipo.
- Una combinación de los anteriores.

Debido a que en este trabajo se presenta una guía para documentar arquitecturas de software utilizando Lenguajes de Descripción de Arquitectura, éstos definen gran parte de los tipos de modelo de un punto de vista. Por lo anterior, a continuación se describen algunas técnicas para documentar y definir tipos de modelos considerando que éstos son definidos por Lenguajes de Descripción de Arquitectura.

9.2.5.1. *Metamodelos*

Un metamodelo presenta los elementos que forman parte del vocabulario de un tipo de modelo y en ocasiones, puede visualizarse como un modelo conceptual. Aunque hay diferentes formas de representarlos, los metamodelos deben incluir (ISO/IEC/IEEE 42010: 2011):

- Entidades: los principales tipos de elementos que están presente en los modelos de este tipo.
- Atributos: las propiedades con que cuentan las entidades de los modelos de este tipo.
- Relaciones: las relaciones definidas entre las entidades de los modelos de este tipo.
- Restricciones: los tipos de restricciones que hay en las entidades, atributos o relaciones en los modelos de este tipo.

Es importante mencionar que las entidades, atributos, relaciones y restricciones, son considerados como elementos de una descripción de arquitectura.

Cuando un punto de vista especifica múltiples tipos de modelos, en ocasiones puede resultar útil especificar un único metamodelo del punto de vista, unificando la definición de los tipos de modelos. También puede resultar útil usar un único metamodelo para expresar múltiples puntos de vista relacionados.

Aunque un metamodelo es una manera poco formal de describir un tipo de modelo implementado por un Lenguaje de Descripción de Arquitectura, algunos LDAs cuentan con un metamodelo que los describe. Tal es el caso de xADL (Dashofy, Hoek y Taylor 2001), cuya definición se basa en un metamodelo de esquemas XML. Otros LDAs cuentan con un metamodelo que define los elementos que constituyen el vocabulario del LDA, tales como componentes, conectores, sistemas, estilos arquitectónicos, etc. Los metamodelos permiten definir las construcciones sintácticas válidas de un LDA, y definen cómo dichas construcciones pueden ser compuestas para producir modelos válidos.

9.2.5.2. Plantillas

Las plantillas proporcionan una forma para especificar el formato y/o contenido de los modelos de un tipo.

Esta opción podría resultar útil si una vista arquitectónica propicia la elaboración de modelos con características similares, de los cuales se pueda definir una plantilla. Si por ejemplo, se utiliza un Lenguaje de Descripción de Arquitectura para elaborar distintos modelos, se podría definir una plantilla basada en la estructura que especifica el LDA para un modelo.

9.2.5.3. Definición del lenguaje

Otra forma de documentar un tipo de modelo de un punto de vista, es identificando o definiendo el lenguaje de modelado que puede ser usado por los modelos de ese tipo.

Para definir y documentar los tipos de modelos implementados por un Lenguaje de Descripción de Arquitectura, se propone incluir la siguiente información, o referencias a la misma:

- Los tipos de notaciones que incluye el LDA. Dichas notaciones pueden ser textuales, gráficas, o ambas. Por ejemplo, en el caso de Acme, AADL, y xADL, cuentan con notaciones tanto gráficas como textuales soportadas por dichos lenguajes o por las herramientas que los soportan, por otra parte, lenguajes como Wright cuentan únicamente con una notación textual.
- Los elementos que constituyen el vocabulario del LDA tales como componentes, conectores, sistemas, estilos arquitectónicos, etc.
- La sintaxis precisa del LDA, la cual se puede encontrar en cartas sintácticas, documentos de definición del lenguaje, u otras fuentes de información relacionada. En el caso de Acme, AADL, Wright y xADL, existen documentos que contienen información de tallada sobre el lenguaje (artículos, libros, página web), y en el caso de Acme, AADL y Wright, existen cartas sintácticas del lenguaje en la forma *Backus-Naur*²⁴ (BNF). En el caso de xADL, existe un conjunto de esquemas que representan el núcleo del lenguaje y definen la estructura de sus modelos.
- La semántica del LDA, que especifica el significado ligado a las entidades y relaciones capturadas en un modelo elaborado con el LDA.
- Las restricciones del lenguaje.

9.2.5.4. Operaciones

Otra forma de documentar los tipos de modelo, es definiendo las operaciones disponibles para los modelos de este tipo. Para la definición y documentación de los tipos de modelo implementados por un Lenguaje de Descripción de Arquitectura se puede incluir:

²⁴ Metalenguaje o notación usada para expresar gramáticas libres de contexto.

- Métodos analíticos soportados por el LDA, que pueden incluir métodos de análisis sintáctico y semántico, y análisis de propiedades arquitectónicas.
- Las técnicas o métodos que establece cada LDA para generar un modelo o especificación arquitectónica. Es importante considerar que de acuerdo a las características de cada LDA, dichos métodos pueden variar ya que no es lo mismo hacer una especificación en un LDA de dominio específico como AADL, que un modelo con un LDA extensible como xADL.

9.2.6. Reglas de correspondencia

Este elemento de información puede ser incluido cuando el punto de vista se define y documenta como un producto de trabajo independiente a una descripción de arquitectura.

Para documentar y definir un punto de vista, se debe incluir cualquier regla de correspondencia definida por el punto de vista o por sus tipos de modelo (ISO/IEC/IEEE 42010:2011). Las reglas de correspondencia asocian elementos de un punto de vista tales como interesados, intereses, y tipos de modelo.

También es importante incluir las reglas de correspondencia entre los elementos de los tipos de modelo implementados por Lenguajes de Descripción de Arquitectura. Para esto se pueden incluir las relaciones y restricciones entre tales elementos (componentes, conectores, etc.).

Para documentar cualquier regla de correspondencia, se propone utilizar una tabla como la 9.4.

Elemento 1	Elemento 2	Correspondencia

Tabla 9.4 Registro de reglas de correspondencia en un punto de vista.

9.2.7. Operaciones entre vistas

Las operaciones entre vistas pueden ser incluidas como parte de la definición y documentación de un punto de vista como producto de trabajo independiente a una descripción de arquitectura. Tales operaciones definen los métodos a ser aplicados a las vistas o modelos elaborados a partir de un punto de vista. Las operaciones pueden ser divididas en las siguientes categorías (ISO/IEC/IEEE 42010:2011):

- Métodos de creación: son los medios por los cuales las vistas son elaboradas usando un punto de vista. Pueden estar en forma de una guía de procesos (cómo comenzar, qué hacer después), de una guía de producto de trabajo (plantillas para vistas de este tipo), heurísticas, estilos o patrones.
- Métodos interpretativos: son los medios por los cuales las vistas son entendidas por el lector y los interesados del sistema.
- Métodos de análisis: son usados para revisar, transformar, predecir, aplicar, evaluar, y razonar sobre los resultados arquitectónicos de la vista. En particular,

los LDAs permiten realizar distintos tipos de análisis de propiedades arquitectónicas tales como atributos de calidad.

- Métodos de diseño o implementación: son usados para realizar o construir sistemas usando la información de la vista arquitectónica.

Las operaciones de una vista pueden ser documentadas en una tabla como la 9.5.

Nombre de la operación	Tipo	Descripción

Tabla 9.5 Registro de las operaciones de una vista en un punto de vista.

9.2.8. Ejemplos

Cuando el punto de vista se define y documenta como un producto de trabajo independiente, es importante incluir ejemplos que proporcionen muestras útiles al lector sobre el uso del punto de vista.

Se recomienda incluir ejemplos sobre la construcción de una vista, o alguna otra operación sobre las mismas que se considere de utilidad para la audiencia del punto de vista.

9.2.9. Notas

Cuando un punto de vista se define y documenta como un producto de trabajo independiente, se debe incluir cualquier información adicional considerada útil. Se propone considerar notas para usuarios especializados, o información sobre el sistema de interés.

9.2.10. Referencias

En un punto de vista, tanto como producto de trabajo independiente como parte de una descripción de arquitectura, se deben incluir las referencias a las fuentes de información. Se propone considerar las siguientes:

- Autores del punto de vista.
- Referencias a literatura, trabajos previos y otros documentos que especifiquen características del punto de vista, o los tipos de modelo del punto de vista.

La información definida y documentada como parte de un punto de vista, permite ayudar al arquitecto en la creación y síntesis de vistas arquitectónicas.

Finalmente, es importante la identificación y registro del razonamiento y decisiones sobre la definición y documentación de puntos de vista como parte de una descripción de arquitectura (ver capítulo 13).

Capítulo 10. Desarrollo y documentación de vistas arquitectónicas

Una descripción de arquitectura debe incluir una o más vistas arquitectónicas, las cuales permiten expresar la arquitectura del sistema desde la perspectiva de intereses específicos, y son gobernadas por puntos de vista arquitectónicos (ISO/IEC/IEEE 42010:2011).

Para los interesados, las vistas arquitectónicas son un medio para verificar que el sistema cubre sus intereses, por tal razón, el desarrollo de las vistas arquitectónicas es una actividad fundamental en la elaboración de una descripción de arquitectura.

Una vista arquitectónica se compone de uno o más modelos arquitectónicos, los cuales son usados para expresar gran parte de la arquitectura del sistema, y son regidos por tipos de modelos específicos (ISO/IEC/IEEE 42010:2011).

El estándar ISO/IEC/IEEE 42010:2011 establece que una descripción de arquitectura debe incluir exactamente una vista arquitectónica por cada punto de vista definido, y que cada vista arquitectónica debe incluir:

- Información complementaria que es especificada por la organización y/o proyecto.
- La identificación del punto de vista que la gobierna.
- Uno o más modelos arquitectónicos que abordan todos los intereses enmarcados por el punto de vista que la gobierna, y que cubren todo el sistema desde ese punto de vista.
- Un registro de asuntos conocidos en la vista con respecto al punto de vista que la gobierna.

Como parte de este trabajo, además de considerar la elaboración de modelos arquitectónicos (requisitos de una vista arquitectónica según lo establecido por el estándar ISO/IEC/IEEE 42010:2011), también se considerarán algunos aspectos sobre la especificación y análisis de arquitecturas de software debido a que este trabajo está enfocado en el uso de Lenguajes de Descripción de Arquitectura formales, los cuales proporcionan las facilidades para realizar dicho análisis y especificación.

En las siguientes secciones de este capítulo, se presenta una propuesta para el desarrollo, análisis y documentación de vistas arquitectónicas (incluyendo sus modelos arquitectónicos) utilizando Lenguajes de Descripción de Arquitectura (LDAs) de forma general, es decir, sin hacer énfasis en un LDA en específico. El capítulo 11 de este trabajo está enfocado en el desarrollo y análisis de modelos arquitectónicos utilizando algunos Lenguajes de Descripción de Arquitectura específicos.

Además, es importante considerar que en el apéndice “D” de este trabajo se presenta una plantilla (plantilla 1) para documentar descripciones de arquitectura, que cuenta con un apartado (sección 5) para documentar las vistas arquitectónicas.

10.1. Desarrollo de vistas arquitectónicas

En este trabajo, el desarrollo de las vistas arquitectónicas de un sistema se fundamenta en los requerimientos que establece el estándar ISO/IEC/IEEE 42010:2011 y en el uso de Lenguajes de Descripción de Arquitectura. Por lo anterior, la propuesta presentada en este trabajo para el desarrollo de vistas arquitectónicas consiste en la realización de las siguientes actividades:

- La primera actividad que se propone realizar, es identificar el conjunto de puntos de vista de la arquitectura del sistema (Ver capítulo 9), ya que la creación de cada una de las vistas arquitectónicas debe ser guiada por el contenido de un punto de vista específico y previamente definido.
- Una vez identificado el conjunto de puntos de vista de la arquitectura, es importante definir un método a seguir para el desarrollo de las vistas arquitectónicas, el cual incluya el orden y el enfoque para su elaboración.
- Una vez definido el método de desarrollo, se propone proceder con la elaboración de los modelos arquitectónicos para cada una de las vistas. Adicionalmente, se debe identificar la información complementaria y relevante de cada una de las vistas arquitectónicas y sus modelos.

Los pasos antes mencionados, son descritos a detalle en las siguientes secciones de este capítulo.

NOTA: en esta sección (10.1) únicamente se presenta una guía para el desarrollo de vistas arquitectónicas, las consideraciones para su documentación son presentadas en la sección 10.3.

10.1.1. Identificación del conjunto de puntos de vista

Para realizar el desarrollo de las vistas arquitectónicas se requiere contar con un conjunto de puntos de vista que guiarán la creación de cada una de las vistas arquitectónicas, y que especifiquen los intereses que serán expresados en sus modelos arquitectónicos, los tipos de modelo que se usarán, y la audiencia a la que estarán dirigidas. Por tales motivos, antes de la elaboración de las vistas arquitectónicas, se requiere la identificación de los puntos de vista arquitectónicos del sistema (ver capítulo 9).

10.1.2. Definición del método de desarrollo de las vistas arquitectónicas

Debido a que una descripción de arquitectura puede incluir más de una vista arquitectónica (cada punto de vista definido implica el desarrollo de una vista), es conveniente seleccionar un enfoque de desarrollo y definir el orden en que serán creadas dichas vistas.

10.1.2.1. *Enfoques para el desarrollo de vistas arquitectónicas*

Es importante considerar que existen 2 tipos de enfoques principales para la construcción de vistas arquitectónicas, los cuales pueden ser usados en conjunto con el estándar ISO/IEC/IEEE 42010:2011. Estos tipos de enfoques son el sintético y el proyectivo, y son descritos a continuación.

Enfoques sintéticos

En los enfoques sintéticos, las vistas del sistema son construidas como metamodelos distintos o heterogéneos, y el sistema se obtiene como síntesis de la información de las diferentes vistas.

El conjunto de los distintos modelos o metamodelos, es usado para describir diferentes características del sistema. Por ejemplo, un lenguaje de modelado basado en el patrón Modelo-Vista-Controlador (MVC), permite especificar aplicaciones web considerando tres intereses principales (datos, lógica de negocio e interfaz de usuario). Dichos intereses deben ser “sintetizados” para obtener el sistema (Cicchetti, Ciccozzi y Leveque 2012).

En los enfoques sintéticos se debe especificar la manera en que los diferentes puntos de vista, con sus intereses particulares, pueden ser fusionados, y se deben definir las semánticas de unión, es decir, las correspondencias entre entidades de modelos diferentes (Cicchetti, Ciccozzi y Leveque 2012).

El principal aspecto relacionado con los enfoques sintéticos es el manejo de la consistencia: mientras que la semántica está involucrada con la relación entre modelos, las interconexiones entre vistas deben ser definidas cuidadosamente, una tarea que crece con el número de vistas. Por otra parte, añadir o actualizar vistas, especialmente si no son ortogonales a las existentes, demanda una revisión de las reglas de consistencia actuales así como los mecanismos de síntesis (Cicchetti, Ciccozzi y Leveque 2012).

Enfoques proyectivos

Los enfoques proyectivos afrontan el problema de las interdependencias entre los tipos de vistas, proporcionando un formalismo común capaz de cubrir todos los puntos de vista, y al mismo tiempo proporcionando compatibilidad con formalismos existentes (Burger 2014).

En estos enfoques, las vistas se construyen sobre un metamodelo único o central, de esta manera, se puede proporcionar un conjunto de vistas que especifiquen el sistema desde diferentes perspectivas, y al mismo tiempo, el manejo de la consistencia resulta “gratis” al realizar la construcción de las vistas, ya que todos los cambios se reducen a manipulaciones del mismo modelo central. A pesar de contar con una gestión de consistencia más fácil que los enfoques sintéticos, los enfoques proyectivos exigen una semántica del lenguaje base bien definida. Por ejemplo, la sincronización de diagramas UML plantea varios problemas incluso si los desarrolladores están operando sobre el mismo modelo debido a las ambigüedades en la formalización de lenguaje (Cicchetti, Ciccozzi y Leveque 2012).

En general, las soluciones proyectivas sufren una personalización limitada debido al lenguaje base fijo y a un conjunto predefinido de vistas.

Otras propuestas

Existen algunas propuestas para el desarrollo de vistas arquitectónicas que combinan las fortalezas de los enfoques proyectivos y sintéticos. Por ejemplo, Burger (2014) propone el uso del enfoque *Vitruvius*, que está basado en el desarrollo de software dirigido por

modelos, y combina las ventajas de los enfoques proyectivo y sintético. Otro enfoque multi-vistas híbrido, es decir, que fusiona el enfoque sintético y el proyectivo es el de Cicchetti, Ciccozzi y Leveque (2012), el cual propone comenzar con el desarrollo de las vistas a partir de un metamodelo, y permitir a los desarrolladores crear vistas a través de un extensivo conjunto de facilidades de personalización.

¿Qué enfoque seleccionar?

El enfoque que se seleccione para la elaboración de las vistas arquitectónicas del sistema, depende en gran medida de los Lenguajes de Descripción de Arquitectura que se utilicen para elaborar los modelos arquitectónicos, ya que éstos pueden permitir elaborar un metamodelo central que facilite implementar un enfoque proyectivo, o pueden permitir definir distintos tipos de modelos que involucren el uso de un enfoque sintético.

Por ejemplo, el *Architecture Analysis & Design Language* (AADL), permite fácilmente utilizar un enfoque proyectivo, ya que todos los modelos creados con él, pueden estar basados en el mismo metamodelo central.

Puede resultar conveniente utilizar un enfoque específico para la elaboración de las vistas arquitectónicas del sistema, sin embargo, es también fundamental estar consciente de las consecuencias que implica. Si se usa un enfoque sintético, se debe ser consciente que el análisis de consistencia puede ser más complicado que usando un enfoque proyectivo, y por otra parte, el uso de un enfoque proyectivo puede requerir más formalidad para elaborar un metamodelo central que si se utiliza un enfoque sintético.

10.1.2.2. Definir el orden de desarrollo

Debido a que una descripción de arquitectura puede incluir más de una vista arquitectónica, es importante definir el orden en que éstas serán creadas. A continuación se presentan algunos consejos y consideraciones para esto.

Considerar recursos disponibles

Las vistas arquitectónicas de un sistema son productos de trabajo que requieren una gran cantidad de recursos (esfuerzo, tiempo), y en algunas ocasiones, los recursos disponibles del proyecto para elaborar dicha documentación pueden ser limitados, por lo que probablemente no se puedan elaborar todas las vistas arquitectónicas del sistema. Por esta razón, se recomienda elaborar las vistas más relevantes, es decir, aquellas que tienen mayor impacto para el sistema y sus interesados. Para identificar las vistas más relevantes, se propone considerar lo siguiente:

- Tomar en cuenta los beneficios que puede aportar cada vista a la organización y/o sistema, ya que debido a que las vistas expresan uno o más intereses a través de sus modelos arquitectónicos, pueden beneficiar a distintos interesados en varios niveles, por lo que se deben considerar aquellas vistas que aporten más beneficios. Por ejemplo, una vista arquitectónica que aporta información fundamental para realizar las pruebas del sistema puede ser relevante, ya que gran parte de los recursos de un sistema son asignados a actividades de pruebas.
- Un criterio para determinar si una vista arquitectónica es relevante, es considerar si el punto de vista que la gobierna enmarca metas de negocio de suma

importancia para la organización, ya que la elaboración de dicha vista puede ser vital para la organización. Por ejemplo, si una vista arquitectónica será usada como parte de negociaciones, su elaboración es fundamental.

- Una vista arquitectónica con mayor audiencia, puede tener mayor alcance, y por tanto ser más relevante que otra.

Para los puntos de vista establecidos por el SEI (Clements et al. 2003), por ejemplo, se considera que las vistas modulares pueden contener información que puede ser de utilidad para más interesados, y pueden presentar información más detallada, por lo que pueden ser más relevantes que otras vistas.

Considerar el modelo de procesos de desarrollo del sistema

Debido a que un sistema puede ser desarrollado utilizando distintos modelos de procesos, y una descripción de arquitectura puede ser creada y utilizada en distintas fases durante el ciclo de vida de un sistema, se recomienda considerar cuáles de las vistas arquitectónicas pueden ser usadas antes que otras, por lo que puede convenir elaborarlas primero. Por ejemplo, si una vista arquitectónica está enfocada en guiar actividades de implementación o despliegue, puede ser conveniente elaborarla antes que vistas que apoyen tareas de mantenimiento del sistema.

Identificar vistas “base”

Existen vistas arquitectónicas que puedan ser base de otras, es decir, que puedan ser requeridas para elaborar o entender otras, por lo que se recomienda identificarlas y considerar si es conveniente elaborarlas primero.

Por ejemplo, una vista gobernada por el punto de vista funcional puede ser elaborada antes que las demás, ya que representa la parte principal de una descripción de arquitectura y frecuentemente es la primera que los interesados tratan de leer (Rozansky y Woods 2005).

10.1.2.3. Algunos consejos adicionales

A continuación se presentan algunos consejos útiles para el desarrollo de las vistas arquitectónicas:

- Mientras se desarrolla una vista arquitectónica, es importante capturar el razonamiento de las decisiones clave, las cuales pueden ser parte de la descripción de arquitectura (ISO/IEC/IEEE 42010:2011) (ver capítulo 13).
- Se recomienda que al realizar el desarrollo de una vista arquitectónica, se identifiquen las cuestiones importantes con respecto al punto de vista que la gobierna, tales como asuntos sin resolver, conflictos conocidos, y excepciones y desviaciones de las convenciones, lo que debe ser documentado como parte de la vista arquitectónica (ver sección 10.3).
- Es importante considerar el enfoque de desarrollo de vistas que se utilice, ya que si se sigue un enfoque proyectivo por ejemplo, se requiere contar primero con el metamodelo base, es decir, del que se derivarán las vistas arquitectónicas.
- Bass, Clements y Kazman (2012) proponen que no se tiene que completar la elaboración de una vista para comenzar con la siguiente, y que tampoco se tiene

que satisfacer toda la información requerida por los interesados, el 80% de información puede ser aceptable y suficiente, sin embargo, se recomienda que esto se considere de acuerdo a las características del sistema y/o proyecto.

10.1.3. Creación de modelos arquitectónicos

Una vez identificado el orden y método a seguir para el desarrollo de las vistas arquitectónicas, se puede proceder con la creación de los modelos arquitectónicos para cada una de ellas.

Una vista puede incluir uno o más modelos arquitectónicos, cuya elaboración se basa en gran parte en las especificaciones del punto de vista que gobierna a la vista. En esta guía, los tipos de modelo empleados para elaborar los modelos arquitectónicos son definidos por Lenguajes de Descripción de Arquitectura.

Debido a que el estándar ISO/IEC/IEEE 42010:2011 no prescribe la forma en que son creados los modelos arquitectónicos, se propone seguir los siguientes pasos para el desarrollo de los modelos de una vista arquitectónica:

- Definir los modelos de la vista que serán desarrollados, precisando sus propósitos, audiencia y nombre específico.
- Para la elaboración de cada modelo arquitectónico se propone realizar lo siguiente:
 - Identificar y entender el tipo de modelo a utilizar (LDAs).
 - Abstractar la información relevante que será incluida como parte del modelo, es decir, identificar los intereses específicos que serán representados.
 - Construir el modelo, incluyendo los elementos que permitan expresar los intereses requeridos, y con base en el tipo de modelo que se utilice. Debido a que para elaborar los modelos arquitectónicos serán utilizados Lenguajes de Descripción de Arquitectura, se puede generar la descripción textual de cada modelo además de su representación gráfica dependiendo del LDA usado.
 - Si es necesario, se puede simplificar el modelo de tal forma que no resulte complejo y que incluya únicamente la información necesaria.

En las siguientes secciones se presentan a detalle las actividades antes mencionadas, y se presentan también algunos consejos y consideraciones para la creación de modelos arquitectónicos efectivos utilizando Lenguajes de Descripción de Arquitectura.

10.1.3.1. Definición de los modelos de una vista arquitectónica

Debido a que una vista arquitectónica puede incluir más de un modelo, es importante definir cuántos y cuáles modelos serán elaborados. Para definir cuantos modelos crear como parte de una vista arquitectónica, se propone considerar lo siguiente:

- Una vista debe incluir uno o más modelos arquitectónicos con el fin de expresar todos los intereses enmarcados por el punto de vista que la gobierna, y que cubren todo el sistema desde ese punto de vista. Este requerimiento es esencial para la asignación completa de intereses en una descripción de arquitectura.

Cada modelo arquitectónico debe expresar una cantidad adecuada de intereses de tal forma que el modelo no resulte demasiado complejo. Si el punto de vista que gobierna a la vista enmarca muchos intereses, y si éstos son incluidos en un solo modelo, se podría generar un modelo muy complejo, por lo que se recomienda crear varios modelos más sencillos de tal forma que se cubran todos los intereses enmarcados por el punto de vista.

- Cuando se requiera crear modelos que puedan ser de interés para diferentes interesados (como los modelos de las vistas funcionales que pueden tener valor para diferentes audiencias), se requiere considerar si es posible crear un único modelo que sea para uso de todos los interesados, o crear varios modelos para los diferentes interesados.

Es conveniente señalar que al compartir modelos arquitectónicos entre las vistas, se reduce la redundancia o repetición de información, y las posibilidades de inconsistencia. Compartir modelos arquitectónicos entre vistas, también permite elaborar una descripción de arquitectura con un estilo orientado a aspectos, ya que los modelos arquitectónicos compartidos a través de vistas arquitectónicas pueden ser usados para expresar perspectivas arquitectónicas (ISO/IEC/IEEE 42010:2011).

Para definir concretamente los modelos que compondrán una vista arquitectónica, se deben identificar sus características principales, por lo que se propone precisar las siguientes particularidades para cada modelo:

- **Propósito:** debido a que los modelos pueden ser artefactos costosos en los que se invierta gran cantidad de tiempo y esfuerzo, se debe asegurar que los modelos tengan un propósito bien definido y sean efectivos para su uso. Se recomienda no comenzar con la elaboración de un modelo si éste no tiene un propósito claro, ya que sus características principales podrían ser confusas. Para definir o identificar el propósito se recomienda considerar las actividades para las que será usado el modelo, y los intereses específicos que son representados por el mismo.
- **Audiencia:** son los diferentes interesados en el modelo, los cuales pueden ser identificados fácilmente con base en el punto de vista que gobierna la vista a la que pertenece el modelo, y específicamente, son aquellos interesados cuyos intereses serán expresados por el modelo. Identificar la audiencia del modelo permite que los modelos sean útiles y accesibles.
- **Nombre:** el nombre de un modelo arquitectónico debe ser adecuado considerando su propósito y audiencia, ya que puede ser confuso asignarle un nombre que no tenga relación con sus características principales.

10.1.3.2. *Comprensión de los tipos de modelo*

Se debe asegurar que para cada modelo definido, se identifique y entienda claramente el tipo de modelo que lo gobierna, es decir, el conjunto de convenciones que se utilizará para su elaboración. El tipo de modelo está definido por el punto de vista que dirige la vista de la que es parte el modelo, y es importante conocer los lenguajes, notaciones, convenciones, técnicas de modelado, métodos analíticos y otras operaciones para usarse en los modelos de este tipo.

Este trabajo está enfocado en el uso de Lenguajes de Descripción de Arquitectura, por lo que las notaciones utilizadas para elaborar cada uno de los modelos, son las convenciones y notaciones de los lenguajes seleccionados (ver capítulo 8).

Es importante conocer claramente las características del Lenguaje de Descripción de Arquitectura que se usará para elaborar cada modelo, ya que si una notación no es comprendida claramente, los desarrolladores y audiencia del modelo pueden confiar en su intuición para definir o entender el contenido del modelo, lo cual puede traer problemas de comunicación e interpretación.

Se recomienda que para cada Lenguaje de Descripción de Arquitectura que sea usado para elaborar modelos arquitectónicos, se comprenda claramente lo siguiente:

- Los fundamentos del lenguaje, es decir, sus principios, objetivos y alcance.
- Los elementos básicos del lenguaje, sus características principales y sus notaciones textuales y gráficas (si es el caso). Algunos de los elementos más comunes en un Lenguaje de Descripción de Arquitectura con componentes, conectores y configuraciones o sistemas.
- Es fundamental conocer la sintaxis del lenguaje, ya que determina las reglas y principios para relacionar sus elementos básicos. Para lo anterior, se puede acudir a cartas sintácticas u otros documentos que especifiquen la sintaxis precisa del lenguaje en cuestión.
- Es necesario que se conozca la semántica del lenguaje, es decir las reglas y restricciones requeridas para cuidar el significado de los modelos arquitectónicos.
- Debido a que el modelado y la especificación utilizando Lenguajes de Descripción de Arquitectura comúnmente se realizan a través de herramientas específicas, es importante conocer sus características particulares, las cuales pueden incluir editores gráficos, editores de texto, analizadores sintácticos o semánticos, extensiones de ayuda, barras de herramientas, o alguna otra característica.

Para identificar la información propia de cada Lenguaje de Descripción de Arquitectura, se puede hacer uso de su documentación particular. Por ejemplo, para el lenguaje AADL se puede recurrir a su carta sintáctica (AADL V2 2011), o documentos para el manejo de sus herramientas gráficas.

10.1.3.3. Identificación de la información relevante del modelo

Es importante identificar la información primordial que contendrá un modelo, es decir, los intereses específicos que serán representados por dicho modelo (atributos de calidad, requerimientos funcionales, comportamiento del sistema, etc.) y el propósito del modelo.

En un modelo debe incluirse la información primordial y necesaria, de tal manera que el modelo no resulte demasiado complejo. Por ejemplo, si se elabora un modelo de concurrencia, podría ser necesario identificar la estructura de las tareas, la comunicación inter-procesos, la sincronización de tareas, o algún otro interés del sistema que requiera ser expresado por el modelo.

10.1.3.4. Construcción del modelo

Una vez que son definidos los modelos para cada vista arquitectónica, y se conocen sus objetivos, los tipos de modelo que los guiarán, y la información que representarán, se puede proceder con su construcción

En las siguientes secciones, se presentan algunas consideraciones y actividades para la construcción de modelos arquitectónicos.

Uso de la abstracción

Para elaborar un modelo es importante utilizar la abstracción, que es la técnica de omitir los detalles no significantes con el fin de comunicar solo las ideas más importantes. La clave para lograr el correcto nivel de abstracción en un modelo es la habilidad de detectar los elementos esenciales en contraposición de los detalles irrelevantes. Lo que es relevante en un modelo varía de acuerdo a las circunstancias, el propósito del modelo, y a las necesidades u habilidades de los interesados, así que determinar lo que se debe incluir en un modelo puede ser una decisión subjetiva (Rozansky y Woods 2005).

Procurar la simplicidad

Es importante tomar en cuenta que un modelo más simple es más fácil de usar, y probablemente sea más efectivo para su audiencia que un modelo complejo. Sin embargo, si un modelo es demasiado simple podría no representar las características esenciales que son de interés para su audiencia. Lo adecuado es tratar de llegar a un balance entre la simplicidad excesiva y la complejidad del modelo.

Muchos modelos comienzan siendo simples y bien estructurados, pero al agregárseles mayor detalle y al considerar más casos específicos su complejidad puede crecer significativamente. Si la complejidad de un modelo aumenta rápidamente, reduce su eficiencia para ser comunicado y analizado. Se debe cuidar que los modelos arquitectónicos no se conviertan en modelos de diseño detallado.

En sistemas grandes los modelos de software pueden llegar a crecer mucho, de forma que los desarrolladores tengan problemas en entenderlo por completo y la navegación a través de los modelos se pueda convertir en una tarea complicada.

Si un modelo es muy complejo y no puede ser usado fácilmente, se puede considerar reemplazarlo con varios modelos más simples, que contengan la misma información pero en una forma más accesible (Rozansky y Woods 2005).

Consideración de estilos y patrones arquitectónicos

La arquitectura del sistema de interés podría estar basada en estilos o patrones arquitectónicos, por lo que para cada modelo a elaborar, se tiene que considerar si algún estilo o patrón arquitectónico será usado con base en los intereses que representa cada modelo.

Es importante identificar los elementos del Lenguaje de Descripción de Arquitectura usado que permitan modelar los estilos o patrones requeridos.

Algunos Lenguajes de Descripción de Arquitectura permiten definir fácilmente estilos arquitectónicos, como es el caso de ACME mediante su herramienta “AcmeStudio”.

Definición de los elementos de un modelo

Un problema particular con las notaciones del modelado gráfico es la tendencia a dibujar el diagrama que representa la estructura del modelo y considerar el modelo completo. Por supuesto, el modelo no está completo debido a que ninguno de sus símbolos en el diagrama ha sido realmente definido, y el modelo está abierto a malas interpretaciones. Este problema no está limitado solo a los modelos creados con notaciones gráficas, es común encontrar modelos cuantitativos (como modelos de eficiencia de desempeño) que son muy difíciles de interpretar por la falta de definición de los elementos y las relaciones capturadas en el modelo.

Mientras se realiza el desarrollo del modelo, se debe asegurar invertir suficiente tiempo para definir cuidadosamente todos sus elementos de tal forma que sus significados, roles, y mapeos al mundo real sean claros y no estén abiertos a diferentes interpretaciones (Rozansky y Woods 2005).

Para definir correctamente los elementos de un modelo arquitectónico, se requiere conocer a detalle las notaciones usadas. En este trabajo dichas notaciones son los Lenguajes de Descripción de Arquitectura, los cuales determinan las restricciones, sintaxis, y principios para definir elementos que pueden ser parte de un modelo, tales como componentes de hardware y software, procesos, y subsistemas. En AADL por ejemplo, se pueden definir componentes tipo “hilos” (*threads*), para lo que se requiere conocer las restricciones para su definición, y las notaciones textuales y gráficas para su representación.

Los nombres asignados a los elementos de un modelo pueden tener un impacto significativo en la comunicación, por tal motivo, es importante elegir nombres adecuados, ya que una vez que los nombres han sido entendidos y discutidos, se convierten en una parte del lenguaje común para un proyecto y puede ser difícil cambiarlos.

Cuando se crean inicialmente los modelos, es fácil asignar nombres ambiguos o engañosos a sus elementos debido a que se está tratando de entender su rol y responsabilidades. Es importante elegir nombres exactos y significativos, lo cual ayuda a los lectores de un modelo a comprender fácilmente su estructura fundamental y el rol de cada elemento en él (Rozansky y Woods 2005).

Definición de las interacciones, comportamiento e interfaces de los componentes

Además de definir los componentes o elementos de un modelo arquitectónico correctamente, se requiere definir las interacciones entre los mismos, es decir, la forma o mecanismos para relacionarse.

Sin suficiente información sobre las interfaces e interacciones entre los componentes de un modelo, los implementadores podrían estar forzados a adivinar las intenciones del arquitecto o a consultarlo continuamente (Allen 1997).

Para definir las interacciones entre elementos de un modelo, es importante conocer la forma en que el Lenguaje de Descripción de Arquitectura permite especificarlas. Por ejemplo, en el lenguaje AADL, se pueden especificar interacciones mediante conexiones de puertos, llamadas a subprogramas o conexiones por parámetros, y para especificar cualquiera de éstas, se requieren declarar como subcláusulas de la especificación de los componentes relacionados. En contraste, lenguajes como Acme y Wright, soportan la especificación de interacciones a través de elementos tipo “conector”.

Las interacciones entre elementos deben ser especificadas y modeladas correctamente debido a que muchas de ellas representan aspectos del comportamiento del sistema. Un ejemplo de lo anterior son las llamadas a subprogramas, que pueden representar parte del comportamiento de un sistema y que pueden ser modeladas como interacciones entre elementos.

El comportamiento de un sistema describe la forma en que las interacciones entre elementos pueden afectar a éstos, en cualquier punto de tiempo o en un estado del sistema dado (Clements et al. 2003).

Considerar el comportamiento de un sistema al realizar los modelos arquitectónicos es una forma de agregar información semántica a los elementos y a sus interacciones con características relacionadas al tiempo. Otras de las razones para considerar el comportamiento del sistema al elaborar los modelos arquitectónicos, son las siguientes (Clements et al. 2003):

- La documentación del comportamiento de un sistema permite razonar sobre la completitud, correctitud, atributos de calidad, y otros requerimientos del sistema.
- La documentación del comportamiento puede ser un vehículo de comunicación importante entre los interesados del sistema.

El comportamiento de un sistema que es expresado en un modelo arquitectónico, debe apegarse a las características del lenguaje utilizado, tales como sintaxis y semántica.

Además de las interacciones entre componentes y su comportamiento, se requiere definir correctamente sus interfaces, es decir, los medios para realizar las interacciones. Las interfaces son importantes para realizar algunos tipos de análisis arquitectónicos, por tal motivo es importante su definición.

Las características propias de cada interfaz dependen del tipo de modelo que se elabore, sin embargo, Clements et al. (2003) establece los siguientes principios para una interfaz:

- Todos los elementos que interactúen con su ambiente tienen interfaces.
- La interfaz de un elemento contiene información específica de la vista a la que pertenece el modelo.
- Las interfaces se constituyen por lo que es proporcionado y requerido por el elemento al que pertenecen.
- Un elemento puede tener múltiples interfaces, lo cual proporciona la separación de intereses y brinda beneficios.
- Un elemento puede interactuar con más de un elemento a través de la misma interfaz.

- Algunas veces puede ser útil tener distintos tipos de interfaces (que pueden actuar como plantillas de interfaces), así como instancias de interfaces con características propias.

Para definir las interfaces de los elementos, se debe identificar las interacciones de dichos elementos con su ambiente.

De la misma manera que para los componentes, para las interacciones e interfaces de los mismos, se debe cuidar incluir únicamente la información requerida y necesaria por los interesados de manera que los modelos no resulten complejos.

Por otra parte, es importante conocer con precisión los tipos de interfaces que pueden ser definidas con el Lenguaje de Descripción de Arquitectura que se utiliza, y la forma en que son definidas (sintaxis y semántica). Por ejemplo, AADL permite definir puertos, y grupos de puertos como interfaces de elementos. Otros lenguajes como Acme y Wright, utilizan puertos y roles para definir las interfaces de componentes y conectores respectivamente.

Habilitación de atributos de calidad

Es importante identificar si el modelo a elaborar expresará atributos de calidad, ya que se debe identificar de qué forma el Lenguaje de Descripción de Arquitectura permitirá expresarlo. Por ejemplo, en el caso del lenguaje AADL, se pueden expresar características de eficiencia de desempeño mediante la asignación de propiedades a algunos elementos.

Por otra parte, si se hace uso de perspectivas para habilitar atributos de calidad, éstas deben ser consideradas en la elaboración del modelo arquitectónico.

Creación de estilos y patrones arquitectónicos

Los modelos arquitectónicos pueden ser utilizados como "contenedores" para la aplicación de patrones o estilos arquitectónicos en vistas arquitectónicas. Por tal motivo se debe considerar si un estilo arquitectónico puede ser obtenido o derivado del modelo elaborado. Para definir un estilo arquitectónico, Clements et al. (2003) propone que se debe considerar: el vocabulario del estilo (tipos de elementos, sus relaciones, propiedades y reglas de composición que determinan como puede ser usado el vocabulario), semánticas para definir el modelo computacional que los elementos soportan, formas de análisis que son soportadas, y las estrategias de implementación que permiten producir un sistema ejecutable. Clements et al. (2003) menciona que los arquitectos a menudo utilizan tres técnicas para desarrollar estilos arquitectónicos:

- Combinación de estilos, que es la mezcla de elementos de varios estilos existentes para producir uno, lo que implica la ventaja del reúso.
- Especialización de un estilo, que refina un estilo existente.
- Estilos dirigidos por análisis, que es la selección de un estilo que permita una forma particular de análisis.

10.2. Análisis de las vistas y modelos arquitectónicos

Debido a que una descripción de arquitectura puede incluir diversas vistas y modelos arquitectónicos, es importante que cuenten con ciertas características que son importantes para asegurar la calidad de la descripción de arquitectura.

Algunas de las propiedades de los modelos que resulta importante considerar y validar son las siguientes:

- **Completitud:** el grado en que todos los requerimientos han sido implementados y verificados en el modelo.
- **Consistencia:** el grado en que el modelo contiene requerimientos, afirmaciones, restricciones, componentes o descripciones de componentes no conflictivos entre ellos.
- **Correctitud:** el grado en que el modelo satisface sus requerimientos y especificaciones de diseño, y está libre de defectos.

Por otra parte, el estándar ISO/IEC/IEEE 42010:2011 introduce correspondencias y reglas de correspondencia para expresar las relaciones entre los elementos en una descripción de arquitectura (ver capítulo 12). Algunas de las correspondencias que pueden ser usadas para capturar y forzar las relaciones entre elementos de la arquitectura son la composición, refinamiento, consistencia, trazabilidad, dependencia, restricción y obligación.

Los métodos formales, como los Lenguajes de Descripción de Arquitectura, son usados para especificar, desarrollar y verificar la arquitectura de software a través de la aplicación de notaciones con bases matemáticas. En estos métodos, la verificación de consistencia, completitud y correctitud puede realizarse de una manera sistemática y automatizada o semi-automatizada.

En las siguientes secciones se presentan algunos métodos para el análisis de diferentes aspectos de vistas y modelos arquitectónicos, sin embargo, no se presentan con gran profundidad debido a que está fuera del alcance de este trabajo.

10.2.1. Análisis de consistencia

Debido a que un sistema es descrito usando diferentes vistas, es necesario que haya consistencia entre ellas, ya que aunque expresan cualidades diferentes del sistema, expresan el mismo sistema. Las múltiples vistas y modelos arquitectónicos de un sistema pueden ser usados para lidiar con la complejidad del sistema, pero la información puede estar esparcida en múltiples modelos heterogéneos, lo que causa inconsistencia.

Las incoherencias entre los diferentes modelos o vistas pueden surgir si éstos se modifican de forma independiente y no son sincronizados.

La redundancia de las vistas también puede afectar la consistencia, ya que ésta se propicia cuando la misma pieza de información está representada en múltiples artefactos con formalismos heterogéneos. Aunque la redundancia puede ser una propiedad deseada

en algunos escenarios de desarrollo, es una de las razones de inconsistencia en la descripción de un sistema (Burger 2014).

¿Cómo mantener la consistencia?

Típicamente, la revisión de consistencia puede ser realizada mediante una herramienta que implemente funciones de análisis automatizado, sin embargo, también se puede verificar manualmente, usando inspecciones o técnicas de revisión.

Para asegurar la consistencia entre modelos y vistas arquitectónicas se propone lo siguiente:

- Al definir el conjunto de modelos, vistas y puntos de vista de la arquitectura de un sistema, deben ser definidas las reglas de correspondencia y las correspondencias que permitan expresar, registrar, hacer cumplir y analizar la consistencia entre modelos y vistas arquitectónicas.
Por otra parte, se deben establecer y verificar las reglas de correspondencia entre los elementos de las diferentes vistas arquitectónicas, y entre los elementos de un modelo arquitectónico.
- Se debe revisar la coherencia entre vistas con distintos niveles de abstracción. Por ejemplo, si existe una vista que sea un refinamiento de otra, se debe revisar la consistencia entre dichas vistas.

Por otra parte, algunos Lenguajes de Descripción de Arquitectura permiten analizar características de consistencia en modelos arquitectónicos. Por ejemplo, AADL permite realizar un análisis de consistencia de las conexiones entre elementos de un modelo. Wright por su parte, permite la validación de consistencia utilizando la herramienta comercial FDR.

10.2.2. Análisis de completitud

La completitud es el grado en que todos los requerimientos han sido implementados y verificados en un modelo o vista. Dicha característica puede ser analizada con una herramienta de modelado que use técnicas tales como análisis estructural y "*state-space reachability analysis*", que asegura que todas las rutas en los modelos de estados son alcanzadas con entradas correctas. Por otra parte, la completitud puede ser verificada manualmente usando inspecciones o técnicas de revisión.

10.2.3. Análisis de correctitud

La correctitud es el grado en que el modelo satisface sus requerimientos y especificaciones de diseño, y está libre de defectos.

El análisis de correctitud incluye:

- La verificación de la correctitud de la sintaxis, es decir, el correcto uso de la gramática y las construcciones del lenguaje del modelo.

- La verificación de la correctitud de la semántica, que es uso de las construcciones del lenguaje para representar correctamente el significado de lo que está siendo modelado.

Para analizar los modelos en cuanto a la correctitud sintáctica y semántica, se puede realizar automáticamente (usando alguna herramienta), o de forma manual buscando defectos.

Comúnmente las herramientas de los Lenguajes de Descripción de Arquitectura formales permiten el análisis sintáctico y semántico de forma automática, tal es el caso de AADL y ACME.

10.2.4. Análisis de interacción o comportamiento

El análisis de la interacción se enfoca en las comunicaciones y relaciones entre los elementos para hacer una tarea en un modelo. Este análisis examina el comportamiento dinámico de las interacciones entre diferentes porciones o elementos de un modelo arquitectónico.

Algunos Lenguajes de Descripción de Arquitectura, como AADL, permiten realizar un análisis de las conexiones entre los elementos de un modelo.

10.2.5. Análisis de atributos de calidad

Cuando un modelo arquitectónico representa atributos de calidad, puede ser conveniente realizar análisis de dichos intereses para asegurar que el sistema cuanta con las propiedades de calidad requeridas.

Para analizar los atributos de calidad de un modelo, se puede recurrir a las herramientas automatizadas de los Lenguajes de Descripción de Arquitectura usados. Por ejemplo, la herramienta Osate de AADL, permite realizar análisis de latencia, seguridad y tolerancia a fallas.

10.3. Documentación de vistas arquitectónicas

El estándar ISO/IEC/IEEE 42010:2011 establece que una vista arquitectónica debe incluir lo siguiente:

- Información complementaria que es especificada por la organización y/o proyecto.
- La identificación del punto de vista que la gobierna.
- Uno o más modelos arquitectónicos que abordan todos los intereses enmarcados por el punto de vista que la gobierna, y que cubren todo el sistema desde ese punto de vista.
- Un registro de asuntos conocidos en la vista con respecto al punto de vista que la gobierna.

En las siguientes secciones (10.3.1 a 10.3.4) se presentan algunas consideraciones para documentar los elementos que conforman una vista arquitectónica. Además, es importante considerar que en el apéndice “D” se presenta una plantilla (plantilla 1) para documentar descripciones de arquitectura que cuenta con un apartado (sección 5) para documentar las vistas arquitectónicas, y que es compatible con la forma en que se propone en este apartado.

10.3.1. Información complementaria

En una vista arquitectónica, puede ser requerida información complementaria que resulte de utilidad para su audiencia. La información complementaria debe ser definida por la organización y/o proyecto, y puede incluir elementos tales como fecha de emisión, autores, organización, resumen, alcance, contexto y referencias. Sin embargo, mucha de esta información pudo haber sido considerada como parte de la información complementaria de la descripción de arquitectura (ver sección 6.2), por lo que no vale la pena repetir dicha información, sino que es conveniente incluir únicamente la información necesaria. La información que se propone considerar, y si es el caso incluir, es la siguiente:

- Resumen: extracto breve del contenido de la vista arquitectónica, que puede ser de utilidad debido a que ofrece un panorama general de la misma. El resumen puede ser generado al finalizar la elaboración de la vista, cuando se conozca el contenido preciso de la misma.
- Alcance: especifica los límites de la vista arquitectónica, aclarando lo que incluye, y lo que no.
- Uso: se propone incluir una descripción de la forma en que la vista arquitectónica puede ser usada correctamente por sus interesados. Se propone incluir:
 - Los interesados para los que fue elaborada la vista.
 - Una relación de los modelos que pueden ser de interés para cada interesado o tipo de interesado.
 - Las secciones que pueden ser consultadas primero, y que permitan entender mejor la vista arquitectónica. En muchos casos, es conveniente definir claramente cuáles son los modelos que son fuente primaria de información, y aquellos que contienen información derivada.

10.3.2. Punto de vista

En una vista se debe incluir el punto de vista que la gobierna. No es necesario incluir toda la definición del punto de vista, únicamente puede incluirse el nombre, y las referencias al mismo si es que será definido y documentado como un producto de trabajo independiente.

10.3.3. Modelos arquitectónicos

Se debe documentar cada uno de los modelos arquitectónicos que pertenecen a la vista.

Según el estándar ISO/IEC/IEEE 42010:2011, cada modelo arquitectónico debe incluir una versión, y el tipo de modelo que lo gobierna.

Aunque no lo especifica el estándar, también se propone considerar otros elementos que permitan entender mejor los modelos arquitectónicos, tales como su nombre, propósito, términos empleados, catálogo de elementos, y sus diagramas y código respectivo. Dichos elementos se detallan a continuación.

10.3.3.1. Nombre del modelo

Se puede incluir el nombre que identifica al modelo, el cual debe ser adecuado para el mismo (ver sección 10.1.1).

10.3.3.2. Versión

Es importante incluir la versión que identifique al modelo tal como lo especifique la organización y/o proyecto.

10.3.3.3. Tipo de modelo

Se debe identificar el tipo de modelo que gobierna al modelo, es decir, el conjunto de convenciones a las que se adhiere.

Debido a que esta información puede estar especificada en el punto de vista que gobierna a la vista arquitectónica a la que pertenece el modelo, no es necesario incluirla completamente, basta con hacer una referencia a ella o mencionarla brevemente. Por ejemplo, si se hace uso del lenguaje AADL, se puede citar su carta de sintaxis o documento de especificación gráfica y/o textual.

Si para la elaboración del modelo se utilizó información específica sobre el tipo de modelo que no esté incluida en el punto de vista, y que resulte relevante para entender el modelo, dicha información puede ser incluida ya sea explícitamente o mediante una referencia.

10.3.3.4. Propósito del modelo

Se propone incluir una breve descripción del propósito de cada modelo, el cual permita identificar si el modelo será útil para los interesados.

10.3.3.5. Términos

Debido a que un modelo es una parte que puede expresar mucha información en una descripción de arquitectura, se pueden incluir los términos que permitan entender fácilmente cada modelo. Ejemplos de éstos pueden ser términos técnicos, o términos relacionados a las notaciones utilizadas.

10.3.3.6. Catálogo de elementos

Un catálogo de elementos puede ser de gran utilidad para documentar un modelo arquitectónico, ya que detalla aquellos elementos representados en el modelo. El catálogo de elementos puede incluir (Bass, Clements y Kazman 2012):

- Los elementos del modelo y sus propiedades. Sobre todo en las notaciones gráficas, puede ser útil incluir una tabla con los elementos gráficos que incluye el modelo, y su significado.
- Las relaciones entre los elementos del modelo.
- Las interfaces de los elementos. Clements et al. (2003) propone que para cada interfaz se documente lo siguiente:
 - La identidad de la interfaz, es decir, si un elemento tiene múltiples interfaces, lo que identifique claramente a cada una de ellas.
 - Recursos proporcionados a elementos. Estos recursos pueden ser definidos mediante su sintaxis, semántica (puede incluir asignación de valores a datos, cambios en el estado del elemento, eventos o mensajes, resultados humanos observables), y restricciones de uso.
 - Tipos de datos definidos localmente.
 - Manejo de errores, es decir, condiciones de error que puedan ser alzadas por los recursos en la interfaz.
 - Cualquier variabilidad proporcionada por la interfaz, es decir, si la interfaz permite a los elementos ser configurados de alguna manera.
 - Características en cuanto a los atributos de calidad de la interfaz, tales como eficiencia de desempeño o confiabilidad.
 - Lo requerido por el elemento, es decir, los recursos proporcionados por otros elementos.
 - Razonamiento del diseño de las interfaces de los elementos, que puede incluir restricciones, compromisos alternativas de diseño, si éstas fueron consideradas, y porqué.
 - Guía de uso para entender las interacciones y comportamiento del modelo.
- El comportamiento de los elementos que no sea tan obvio (el comportamiento también puede ser documentado como parte de la documentación de las interfaces, o en la documentación del análisis de la arquitectura). Como parte de la documentación del comportamiento se puede incluir lo siguiente:
 - Tipos de comunicación, por ejemplo si es un flujo de datos o de control.
 - Restricciones, por ejemplo, en una comunicación síncrona, se puede incluir el elemento que inicia y el elemento que termina la comunicación.
 - Simulaciones activadas por reloj, es decir, si hay actividades que se llevarán a cabo en cierto momento o después de cierto intervalo de tiempo, se debe incluir cierta noción del tiempo.

10.3.3.7. *Diagramas y código*

Debido a que los modelos serán creados con Lenguajes de Descripción de Arquitectura, serán generados modelos gráficos y especificaciones textuales (código) de dichos modelos.

Dependiendo del nivel técnico de los interesados para los que está dirigido cada modelo, se puede incluir su representación gráfica o textual.

Es difícil mantener una documentación con imágenes o código que cambia constantemente, por lo que podría ser conveniente citar la fuente del recurso, es decir, la referencia al modelo o código que pueden ser parte de algún repositorio de información de la organización.

10.3.3.8. *Estilos arquitectónicos*

Si como parte de la elaboración fue definido un estilo o patrón arquitectónico, éste puede ser documentado especificando su vocabulario, semántica, tipos de análisis y estrategias de implementación como propone Clements et al. (2003).

10.3.4. Cuestiones con respecto al punto de vista

El estándar ISO/IEC/IEEE 42010:2011 establece que se debe registrar cualquier cuestión en la vista con respecto al punto de vista que la gobierna. Se recomienda considerar los siguientes aspectos:

- Asuntos sin resolver: cualquier asunto relacionado con la elaboración de las vistas arquitectónicas que no se resolvió, por ejemplo, intereses que no fueron considerados en las vistas arquitectónicas.
- Conflictos conocidos: problemas que resultan de la elaboración de las vistas arquitectónicas, tales como dificultades para expresar intereses, o inconvenientes en el análisis de las mismas.
- Excepciones y desviaciones de las convenciones, las cuales pueden ser documentadas como salidas de decisiones y razonamiento. Ejemplos de estas excepciones y desviaciones pueden ser omisiones de reglas de los lenguajes utilizados.

Capítulo 11. Modelado y especificación arquitectónica utilizando Lenguajes de Descripción de Arquitectura

En este capítulo se presentan las consideraciones generales para realizar modelos y especificaciones arquitectónicas utilizando los siguientes Lenguajes de Descripción de Arquitectura: AADL, Acme, Wright y xADL.

Los lenguajes presentados en este capítulo pertenecen a diferentes generaciones de LDAs y están enfocados en cuestiones arquitectónicas específicas. Se incluye un Lenguaje de Descripción de Arquitectura de primera generación (Wright), uno de intercambio (Acme), uno de dominio específico con capacidades de análisis (AADL), y uno con características de extensibilidad (xADL).

El objetivo de presentar diferentes LDAs en este trabajo, es exponer las características que tienen en común, presentar sus enfoques y rasgos particulares, identificar los conceptos arquitectónicos que han evolucionado con el tiempo, y facilitar al lector la selección de uno o más LDAs.

Por otra parte, es importante mencionar que dentro del proceso de la elaboración de una descripción arquitectónica, el modelado arquitectónico es parte del desarrollo de las vistas arquitectónicas, por lo que este capítulo puede concebirse como un complemento al capítulo 10 (Desarrollo de vistas arquitectónicas). La razón por la que se dedica un capítulo completo para abordar el modelado y especificación arquitectónica utilizando LDAs, es debido al tipo y extensión del contenido.

Adicionalmente, se recomienda leer con anterioridad el capítulo 8, dónde se presentan las características generales de cada LDA, y el capítulo 10, que aborda el desarrollo de vistas arquitectónicas.

11.1. Especificación arquitectónica utilizando un Lenguaje de Descripción de Arquitectura de primera generación (Wright)

Wright es un Lenguaje de Descripción de Arquitectura formal, que permite describir arquitecturas, aplicar procesos formales (álgebras de proceso y refinamiento de procesos), y realizar verificaciones automatizadas de propiedades arquitectónicas.

Wright es un LDA de primera generación que se distingue por el uso de tipos de componentes y conectores independientes y explícitos, la habilidad de describir el comportamiento abstracto de componentes usando la notación CSP, la caracterización de estilos usando predicados sobre instancias de sistemas, y una colección de verificaciones estáticas para determinar la consistencia y completitud de una especificación arquitectónica (Allen 1997).

En las siguientes secciones se presenta una guía para especificar la estructura y comportamiento de arquitecturas de software, y definir estilos arquitectónicos utilizando Wright. También se presenta un ejemplo que incluye la especificación de un estilo arquitectónico, y la especificación de la estructura y el comportamiento de una arquitectura de software basada en ese estilo.

11.1.1. Especificación estructural de la arquitectura

Wright está construido sobre las siguientes abstracciones arquitectónicas básicas (Allen 1997):

- **Componente:** describe un cómputo independiente.
- **Conector:** representa interacciones entre componentes.
- **Configuración:** es una colección de instancias de componentes conectadas a través de conectores. Wright utiliza "instancias" para distinguir cada componente y conector por separado en una configuración.

Para especificar la estructura de una arquitectura, se debe identificar y especificar los tipos de componentes y conectores que serán utilizados, y posteriormente, definir una configuración con las instancias de dichos componentes y conectores, indicando las relaciones entre ambos.

Para definir los tipos de componentes de una arquitectura, se debe identificar cuáles son los elementos que serán parte de la arquitectura, y definir su funcionamiento e interfaces de comunicación. Algunos ejemplos de componentes que pueden ser especificados con Wright son: clientes, servidores, depósitos de datos, procedimientos, programas, clases y funciones.

Para definir los conectores de la arquitectura del sistema, se debe identificar cuáles son las interacciones que habrá entre los componentes que conformarán la arquitectura.

A continuación se presentan las consideraciones para especificar componentes, conectores y configuraciones utilizando Wright.

11.1.1.1. Especificación de tipos de componentes

La especificación de un tipo de componente cuenta con dos partes importantes: la interfaz y el cómputo. La interfaz se conforma de la especificación de un conjunto de puertos, y la especificación del cómputo describe lo que el componente hace, es decir, lleva a cabo las interacciones descritas por los puertos y muestra cómo están vinculados en conjunto para formar un “todo coherente” (Allen 1997).

A continuación se presenta un ejemplo de la especificación estructural de un tipo de componente llamado “Filtro”, que cuenta con un puerto de entrada y uno de salida, y cuyo objetivo es leer datos desde puerto de entrada y transmitirlos al puerto de salida.

Component Filtro

Port Entrada [*lee datos hasta el final de los datos*]

Port Salida [*transmite los datos repetidamente*]

Computation [*lee repetidamente los datos del puerto “Entrada” y los emite al puerto “Salida”*]

Puede notarse que el comportamiento del tipo de componente “Filtro”, es definido de manera informal en este ejemplo (se presenta como texto dentro de corchetes), ya que en esta sección únicamente se aborda la especificación estructural, en futuras secciones se presenta la especificación de comportamiento de manera formal utilizando Wright.

11.1.1.2. Especificación de tipos de conectores

Para realizar la especificación de un tipo de conector, se requiere definir un conjunto de roles y el “glue” o “pegamento” del conector. Cada rol especifica el comportamiento de un participante en una interacción, mientras que el “glue” del conector describe la forma en que los participantes trabajan en conjunto para crear una interacción, y representa la especificación completa de comportamiento (Allen 1997).

A continuación se presenta un ejemplo de la especificación estructural de un tipo de conector llamado “Conducto”, que representa la interacción entre un componente “Fuente” y uno “Destino”:

Connector Conducto

Role Fuente [*entrega datos repetidamente, señalando la terminación por medio del cierre*]

Role Destino [*lee datos repetidamente, cerrando al final de los datos*]

Glue [*“Destino” recibe los datos en el mismo orden entregados por “Fuente”*]

Al igual que el ejemplo anterior, el comportamiento del tipo de conector “Conducto” es definido de manera informal, especificando de manera formal solo la parte estructural.

Es importante mencionar que el hecho de que un Lenguaje de Descripción de Arquitectura permita definir explícitamente un conector, es una ventaja sobre notaciones menos formales en las cuales los conectores se representan únicamente con líneas y no se puede definir a los participantes en una interacción.

11.1.1.3. Especificación de configuraciones

Para especificar una configuración (conjunto de componentes y conectores que forman una arquitectura), se debe especificar los tipos de componentes y conectores que conforman la arquitectura de un sistema, las instancias de dichos componentes y conectores, y los “*attachments*” o asociaciones entre ellos (Allen 1997).

Las instancias de los tipos de componentes y conectores que forman parte de una configuración, deben ser nombradas única y explícitamente. El siguiente ejemplo es la definición de una instancia del tipo de componente “Filtro” llamada “Analizador”:

Analizador: **Filtro**

Los “*attachments*” o asociaciones de una configuración o sistema, definen la topología de la configuración, mostrando los componentes que participan en cada interacción. Para especificar un *attachment*, se deben asociar los puertos de los componentes con los roles de los conectores que participan en la interacción. A continuación se presenta un ejemplo de una asociación entre el puerto “Salida” de un componente “Comp” con el rol “Fuente” de un conector “Cnn”:

Comp. Salida **as** Cnn. Fuente

Finalmente, es importante mencionar que Wright soporta descripciones jerárquicas, por lo que se pueden definir subsistemas dentro de un sistema. Para esto, se puede definir en el cómputo de un componente, una subestructura o subsistema con un conjunto de relaciones (*bindings*) asociadas, que definen la forma en que los puertos en el subsistema son asociados con los puertos del sistema.

En la sección 11.1.4 se presenta un ejemplo completo de una configuración, que incluye la definición de tipos de componentes y conectores, instancias de los mismos, y sus respectivas asociaciones o *attachments*.

11.1.2. Especificación de estilos arquitectónicos

Además de arquitecturas de sistemas, Wright permite definir familias de sistemas a través de la especificación de estilos arquitectónicos, los cuales definen un conjunto de propiedades que son compartidas por configuraciones o sistemas. Estas propiedades pueden incluir un vocabulario común (que se conforma de un conjunto de tipos de componentes y conectores), y un conjunto de restricciones (Allen 1997).

Cada una de las restricciones declaradas por un estilo, representa un predicado que debe ser cumplido por cualquier configuración que sea miembro de dicho estilo. La notación para especificar las restricciones se basa en la lógica de predicados de primer orden, y dichas restricciones se pueden referir a los siguientes conjuntos y operadores (Allen 1997):

- El conjunto de componentes en una configuración.
- El conjunto de conectores en una configuración.
- El conjunto de *attachments* o asociaciones en una configuración.
- *Name(e)*: el nombre del elemento *e*, donde *e* es un componente, conector, puerto, o rol.
- *Type(e)*: el tipo del elemento *e*.
- *Ports(c)*: el conjunto de puertos del componente *c*.
- *Computation(c)*: el cómputo del componente *c*.
- *Roles(c)*: el conjunto de roles del conector *c*.
- *Glue(c)*: el “*glue*” del conector *c*.

A continuación se presenta un ejemplo de una restricción que indica que todos los componentes de cierto estilo deben ser del tipo “Filtro”:

$$\forall c : \mathbf{Components} \cdot \mathbf{Type}(c) = \mathbf{Filtro}$$

Además de las restricciones y los tipos de componentes y conectores, para definir un estilo arquitectónico se puede utilizar la parametrización, que permite dejar “agujeros” en la especificación de un tipo de componente o conector, y que pueden ser llenados cuando dicho tipo sea instanciado (Allen 1997). Por ejemplo, la siguiente definición es de un tipo de componente “FiltroParametrizado” que permite parametrizar su cómputo.

Component FiltroParametrizado (*C* : **Computation**)
Port Entrada . . .
Computation = *C*

Además de estilos, Wright permite definir sub-estilos. En Wright, un estilo es sub-estilo de otro si hereda todas sus restricciones (Allen 1997).

En la sección 11.1.4 se presenta un ejemplo de la definición de un estilo arquitectónico completo.

11.1.3. Especificación de comportamiento

Wright hace posible la especificación de comportamiento arquitectónico utilizando la notación “*Communicating Sequential Processes*” (CSP) (Hoare 1978), que permite describir patrones de interacción en sistemas.

En la notación CSP hay 2 clases de primitivas: los eventos y los procesos. Los eventos son la unidad básica de comportamiento, representan una acción importante, y son indivisibles e instantáneos. Los procesos son combinaciones de eventos y otros procesos más simples, y representan comportamientos fundamentales.

Dentro de los operadores algebraicos más comunes de CSP se encuentran:

- Prefijo \rightarrow : combina un evento y un proceso para producir un nuevo proceso.

- Elección determinística $[\]$: también llamada elección externa, y permite definir un proceso como una elección entre dos procesos, y permite al ambiente resolver la elección.
- Elección no determinística Π : elección interna que permite definir un proceso como una elección entre dos procesos, pero no permite al ambiente ningún control sobre la selección.
- Intercalación $\|$: representa una actividad de concurrencia completamente independiente.
- Interfaz paralela $\{ \}$: representa una actividad concurrente que requiere sincronización entre los procesos.
- Ocultación \backslash : proporciona una forma de abstraer procesos, haciendo algunos eventos no observables.

Más detalles sobre la notación CSP puede consultarse en “*Communicating Sequential Processes*” (Hoare 1978).

Es importante mencionar que para la especificación de comportamiento, Wright añade a la notación CSP ciertos elementos que permiten definir distintos tipos de eventos (Allen 1997):

- Los eventos iniciados o señalados por un proceso, se denotan con una barra encima: *evento*.
- Los eventos observados por un proceso se denotan sin barra: *evento*.
- Un evento especial que indica la terminación exitosa del sistema (representa fin de comunicación) se denota como: \checkmark
- Los procesos que proporcionan datos son considerados salidas, son iniciados, y se escriben con un símbolo de exclamación: *evento! dato*.
- Los procesos que reciben datos se consideran entradas, son observados y se escriben con un símbolo de interrogación: *evento? dato*.

Además de la notación CSP y los elementos adicionales que incluye Wright, se puede hacer uso de técnicas como la secuenciación para especificar el comportamiento de una arquitectura. La secuenciación es la forma más simple de construir un proceso y hace uso del operador “ \rightarrow ” y del operador “;”.

Por ejemplo, la especificación: $e \rightarrow P$, es el proceso en el que primero participa el evento e , y después se comporta como el proceso P . El operador “;” también permite combinar dos procesos en una secuencia. Por ejemplo: $P; Q$, es el proceso que se comporta como P y después se comporta como Q . Si el proceso P no termina, $P; Q$ actúa como P para siempre.

11.1.4. Ejemplo usando Wright

El ejemplo que se presenta a continuación, es la especificación de un estilo arquitectónico Cliente-Servidor, en el que se define la estructura y comportamiento de sus tipos de componentes y conectores a través de Wright y la notación CSP. También se definen las

restricciones de dicho estilo a través de lógica de predicados de primer orden, y se presenta un ejemplo de arquitectura de un sistema basado en el estilo definido.

11.1.4.1. El estilo Cliente-Servidor

Dentro de los estilos arquitectónicos que son definidos mediante componentes y conectores se encuentran: *Pipes and Filters*, *Shared-Data*, *Publish-Suscribe*, Cliente-Servidor, *Peer-to-Peer*, y *Communicating-Processes*. En particular, el estilo Cliente-Servidor cuenta con las siguientes características (Clements et al. 2003):

- Los tipos de componentes son: clientes (que solicitan servicios) y servidores (que proporcionan servicios). Los servidores tienen interfaces que describen los servicios que proporcionan, sin embargo, los servidores pueden también solicitar servicios de otros servidores formando una jerarquía de invocación de servicios.
- El tipo de conector del estilo Cliente-Servidor es *request/reply* (solicitar/responder), que es la invocación asimétrica (unidireccional) de servicios por parte de los clientes, y cuentan con un puerto “solicitar” y uno “responder”. Los conectores relacionan a los clientes con el rol “solicitar”, y a servidores con el rol de “responder”, y determinan qué servicios son petición de qué clientes.
- En el modelo computacional, las interacciones o comunicaciones son iniciadas por los clientes, quienes solicitan servicios a los servidores y esperan por el resultado de esas solicitudes. Los servidores proporcionan un conjunto de servicios a través de una o más interfaces, y los clientes usan servicios proporcionados por los servidores en el sistema. Un cliente debe saber la identidad del servicio que invoca, en contraste, los servidores no saben la identidad de los clientes antes de la solicitud de un servicio.
- En cuanto a la topología, por lo general no hay restricciones.

11.1.4.2. Definición de un estilo Cliente-Servidor en Wright

El estilo Cliente-Servidor presentado a continuación (código 11.1), es tomado de “*Specifying Dynamism in Software Architectures*” (Allen, Douence y Garlan 1998), utiliza la notación de Wright para la especificación estructural, la notación CSP para la especificación del comportamiento, y la lógica de predicados de primer orden para la especificación de las restricciones del estilo.

//Definición del estilo “Client-Server”

Style Client-Server

//Definición de tipos de componentes (“Client” y “Server”)

//Definición del tipo de componente “Client”. Cuenta con un puerto “p” que inicia una petición, y espera por una respuesta

Component Client

Port p = $\overline{\text{request}}$ → reply → p Π §

Computation = internal Compute → $\overline{p.\text{request}}$ → p.reply → **Computation** Π §

//Definición del tipo de componente “Server”. Cuenta con un puerto “p” que espera por peticiones y emite respuestas

Component Server

```

Port p = request →  $\overline{\text{reply}}$  p [] §
Computation = p.request → internalCompute →  $\overline{p.\text{reply}}$  Computation []
§

//Definición del tipo de conector "Link"
Connector Link
  Role c =  $\overline{\text{request}}$  →  $\overline{\text{reply}}$  → c [] §           //Rol que actúa como cliente
  Role s = request →  $\overline{\text{reply}}$  → s [] §           //Rol que actúa como servidor

  //El "glue" describe cómo interactúan los roles entre sí.
  //La solicitud del cliente (c.request) debe ser transmitida al
  //servidor (s. $\overline{\text{request}}$ ), mientras que,
  //la respuesta del servidor (s.reply) debe ser transmitida al
  //cliente (c. $\overline{\text{reply}}$ )

  Glue = c.request →  $\overline{s.\text{request}}$  → Glue
  [] s.reply →  $\overline{c.\text{reply}}$  → Glue
  [] §

//Definición de restricciones
Constraints
  //Existe un componente s del tipo "Server", y todos los componentes
  //del tipo "Client" deben estar conectados al componente tipo
  //"Server"
  ∃! s ∈ Component, ∀ c ∈ Component : TypeServer(s) ∧ TypeClient(c)
  ⇒ connected(c, s)
EndStyle

```

Código 11.1 Ejemplo de la definición de un estilo Cliente-Servidor con Wright.

11.1.4.3. Especificación de una configuración

La configuración "Simple" presentada a continuación (código 11.2), está basada en el estilo Client-Server definido en el código 11.1, y cuenta con un componente tipo "Client", uno tipo "Server", y un conector tipo "Link".

```

//Definición de la configuración "Simple"
Configuration Simple

  //Estilo del que forma parte la configuración: "Client-Server"
  Style Client-Server

  //Instancias de los componentes y conectores del estilo
  Instances
  C : Client ;
  L : Link ;
  S : Server

  //Attachments o relaciones entre los componentes y conectores
  Attachments
  C.p as L.c ;
  S.p as L.s

```


EndConfiguration

Código 11.2 Ejemplo de la definición de una configuración con Wright.

11.2. Especificación y modelado arquitectónico utilizando un Lenguaje de Descripción de Arquitectura de intercambio (ACME)

Acme es un lenguaje de intercambio arquitectónico que permite utilizar distintos LDAs a través de un mecanismo de extensión (Garlan, Monroe y Wile 1997). Permite especificar la estructura arquitectónica de sistemas y estilos arquitectónicos por medio de una notación textual, y permite elaborar su representación o modelo gráfico a través de una notación gráfica implementada por la herramienta *AcmeStudio*.

Para la especificación estructural de una arquitectura, Acme cuenta con una ontología de siete tipos de elementos arquitectónicos (Garlan, Monroe y Wile 1997):

- **Componente:** representa un elemento computacional primario o un almacén de datos de un sistema.
- **Conector:** representa las interacciones entre componentes.
- **Sistema:** representa un conjunto o configuración de componentes y conectores.
- **Puerto:** es una interfaz de un componente, es decir, un punto de interacción entre un componente y su entorno.
- **Rol:** es una interfaz de un conector. Un rol define un participante en una interacción representada por un conector.
- **Representación:** permite especificar explícitamente refinamiento estructural. Cualquier componente o conector puede ser representado por una o más descripciones de más bajo nivel, lo que permite elaborar descripciones jerárquicas.
- **Rep-map (*Representation map*):** permite asociar la descripción de un componente abstracto con representaciones detalladas a través de una lista de enlaces.

En la tabla 11.1 se presenta la notación gráfica que incorpora *AcmeStudio* para los elementos de Acme.

Elemento	Representación gráfica	Elemento	Representación gráfica
Componente		Puerto	
Conector		Rol	

Tabla 11.1 Notación gráfica de AcmeStudio.

Además de los elementos estructurales, Acme permite definir propiedades para añadir una amplia variedad de información auxiliar a una especificación arquitectónica, tal como semántica de tiempo de ejecución, información detallada de tipado (como tipos de datos comunicados entre componentes), protocolos de interacción, e información sobre consumo de recursos.

Acme también permite definir familias de sistemas (conjuntos de elementos que constituyen el vocabulario de un estilo arquitectónico) mediante la definición de tipos de elementos (componentes, conectores, roles, etc.) y restricciones para los sistemas de la familia. Para especificar dichas restricciones, Acme hace uso de un lenguaje de restricciones basado en la lógica de predicados de primer orden llamado *Armani*.

En las siguientes secciones se presenta una guía general para especificar estructuras y estilos arquitectónicos con Acme.

11.2.1. Especificación arquitectónica

El primer paso para realizar la especificación y modelado de una arquitectura en Acme, es identificar los elementos arquitectónicos (sistemas, componentes, conectores, puertos, roles y representaciones) que conformarán la arquitectura.

Es importante considerar que los componentes pueden ser usados para modelar hardware y software, o para modelar una abstracción que pueda ser mapeada a software, hardware, o ambos. Además, los componentes pueden ser usados para describir elementos en diferentes niveles de abstracción, por lo que pueden representar una parte de un sistema complejo. Algunos ejemplos de componentes son procesos, procesadores, recursos de hardware, dispositivos externos, servidores, sistemas de archivos, bases de datos, programas y filtros.

Por otra parte, se debe identificar si los elementos que conformarán la arquitectura son parte de algún estilo arquitectónico ya definido. Esto puede resultar útil debido a que se podrían reusar dichos elementos.

Es importante también identificar si se hará uso de alguna propiedad que permita incorporar información adicional a la arquitectura. Dichas propiedades pueden pertenecer a otros Lenguajes de Descripción de Arquitectura como Wright, UniCon o Aesop.

11.2.1.1. Especificación de componentes

Para definir un componente, se debe saber cuál es el trabajo que realizará, sus puertos, y sus propiedades específicas que definan aspectos de su estructura o comportamiento computacional.

Los componentes exponen su funcionalidad a través de sus puertos, los cuales son puntos de contacto entre el componente y su ambiente. Ejemplos de puertos son interfaces, fuentes de peticiones, servicios proporcionados, y fuentes o destinos de datos.

La especificación de un componente se compone de un nombre, la definición de sus puertos, y la definición de sus propiedades (Garlan, Monroe y Wile 1997).

Las propiedades representan información semántica sobre un sistema y los elementos que lo componen. Cada propiedad es definida mediante un nombre, un tipo (opcional), y un valor. Es importante mencionar que los siete tipos de elementos que conforman la ontología de Acme (componentes, conectores, puertos, roles, etc.) pueden incluir en su definición propiedades.

El código 11.3 presenta un ejemplo de la especificación de un componente llamado "ProcesadorTexto" con un puerto de entrada y uno de salida, y cuya función (definida como una propiedad llamada "implementación") es leer datos del puerto de entrada, hacer algún cómputo sobre dichos datos, y presentarlos en el puerto de salida.

```

Component ProcesadorTexto = {
    Port entrada;
    Port salida;
    Property implementacion : String = "while (!entrada.eof) {
        entrada.read; entrada.read; compute; salida.write}";
}

```

Código 11.3 Ejemplo de la definición de un componente con Acme.

11.2.1.2. Especificación de conectores

Un conector captura la naturaleza de interacción entre componentes. Para especificar un conector, se debe saber qué tipo, y cuántos elementos conectará. Ejemplos de conectores son: modelos de sincronización, protocolos de comunicación, llamadas a procedimientos, o características de un canal de comunicación como ancho de banda u otras interacciones de tiempo de ejecución entre componentes más complejos.

Para especificar un conector, se debe definir un conjunto de interfaces o roles que puntualicen los participantes en una interacción. Además de los roles, los conectores también pueden hacer uso de las propiedades para describir información adicional del conector (Garlan, Monroe y Wile 1997).

El código 11.4 presenta un ejemplo de una especificación de un conector llamado "colaboración", con tres roles, y una propiedad usada para definir la forma en que las tareas derivadas de la solicitud de un "solicitante" son distribuidas entre dos "esclavos".

```

Connector colaboracion = {
    Role solicitante;
    Role esclavo1;
    Role esclavo2;
    Property distribucion =
        "solicitante.solicitudA -> esclavo1.hacerX,
        esclavo2.hacerY;
        solicitante.solicitudB -> esclavo.hacerU |
        esclavo2.hacerV" }

```

Código 11.4 Ejemplo de la definición de un conector con Acme.

11.2.1.3. Especificación de arquitecturas de sistemas

La arquitectura de un sistema se conforma por un conjunto de elementos, como componentes y conectores, y la forma en que interactúan para lograr ciertos objetivos.

De la misma manera que para otros elementos arquitectónicos, en los sistemas pueden definirse propiedades, las cuales son derivadas de los componentes y conectores que conforman el sistema, y que representan alguna propiedad emergente del mismo. Las propiedades en los sistemas también pueden usarse para representar características del ambiente en el que el sistema opera, o propiedades “globales” que aplican a todos los elementos del sistema.

Al igual que en Wright, la topología de un sistema en Acme se define mediante un conjunto de “*attachments*” o asociaciones, las cuales representan las interacciones entre componentes y conectores, y se definen relacionando los puertos de los componentes con los roles de los conectores.

El código 11.5 presenta un ejemplo de un sistema que está conformado por un componente “cliente” y uno “servidor”, los cuales interactúan a través de un conector llamado “petición”.

```
System SistemaClienteServidor = {
    Component servidor = {
        Port respuesta;
    };
    Component cliente = {
        Port hacePetición;
    };
    Connector petición = {
        Role solicitante;
        Role solicitado;
    };
    Attachments {
        servidor.respuesta to petición.solicitante;
        cliente.hacePetición to petición.solicitado;
    }
}
```

Código 11.5 Ejemplo de la definición de un sistema con Acme.

11.2.2. Especificación de familias de sistemas

Una familia de sistemas incluye una colección de tipos de componentes y conectores que conforman el vocabulario de un estilo arquitectónico. Además, una familia de sistemas puede incluir un conjunto de tipos de propiedades, y restricciones para los sistemas que pertenezcan a dicha familia.

11.2.2.1. Especificación de tipos de elementos

Al definir arquitecturas de sistemas, puede resultar más sencillo instanciar tipos de elementos previamente definidos en un estilo arquitectónico, que definir elementos nuevos. Por esta razón, en la definición de un estilo arquitectónico resulta fundamental la especificación de tipos de elementos arquitectónicos que puedan ser instanciados por sistemas que pertenezcan a dicho estilo.

Para definir un tipo de elemento, como un componente, conector, o puerto, únicamente se debe incluir “type” antes del nombre del elemento. Por ejemplo, la siguiente, es la especificación de un tipo de componente “Dispositivo” que puede ser instanciado y complementado en la especificación de la arquitectura de un sistema:

```

Component Type Dispositivo = {
    . . . . .
}

```

11.2.2.2. Especificación de restricciones

AcmeStudio incorpora un lenguaje de restricciones llamado “*Armani*” (Monroe 2000), el cual puede ser usado para especificar restricciones en sistemas o estilos arquitectónicos. El lenguaje de restricciones incluye un conjunto de construcciones basadas en la lógica de predicados de primer orden, y en un conjunto de funciones especiales que se refieren a aspectos específicos de arquitectura. Dichas funciones especiales pueden determinar, por ejemplo, si dos componentes están conectados, o si un componente tiene una propiedad particular.

Las funciones de arquitectura especiales que incorpora *Armani* para especificar restricciones, incluyen funciones de tipo, funciones de conectividad, y funciones sobre conjuntos (Monroe 2000).

Por otra parte, es importante mencionar que las restricciones pueden ser invariantes o heurísticas. Las invariantes son reglas que no pueden ser violadas, mientras que las heurísticas son reglas que deben ser observadas, pero pueden ser selectivamente violadas.

A continuación se presenta el ejemplo de una restricción invariante que especifica que todos los sistemas de un estilo deben tener al menos un componente del tipo “Servidor”:

```

rule reglaUnServidor = invariant size({select c in self.COMPONENTS
| satisfiesType(c, Servidor)}) >= 1;

```

11.2.3. La herramienta AcmeStudio

AcmeStudio, además de permitir realizar especificaciones textuales utilizando Acme, permite definir gráficamente arquitecturas de sistemas, como se muestra en la figura 11.1.

En general, las facilidades que proporciona *AcmeStudio* son las siguientes:

- Permite definir de manera gráfica y textual familias de sistemas, incluyendo los tipos de elementos, tipos de propiedades, funciones, y patrones de conexión que las conforman.
- Permite definir gráfica y textualmente sistemas, asociándolos a una familia de sistemas previamente definida. La definición de sistemas de manera gráfica es sencilla debido a que simplemente se requiere “arrastrar” de la paleta de elementos, aquellos que conformarán el sistema, y posteriormente definir sus propiedades y características fácilmente. Además, *AcmeStudio* permite visualizar

los elementos que conforman cada familia de sistemas para facilitar el modelado arquitectónico.

- Permite realizar de manera automática el análisis de eficiencia de desempeño y de planificación²⁵ en sistemas del estilo arquitectónico “tiempo real”. Muestra información sobre dichos análisis tales como la utilización de recursos (red, CPU), y el conjunto de información de tareas incluyendo sus propiedades de tiempo (ejecución, respuesta, etc.) y resultados del análisis de planificación (Choi 2003).
- Debido a que es una herramienta basada en Eclipse, AcmeStudio puede descargarse y ser usada inmediatamente.

Para una mayor información sobre el uso de *AcmeStudio*, se puede consultar la página de Acme (<http://www.cs.cmu.edu/~acme/>), que incluye tutoriales para utilizar *AcmeStudio*.

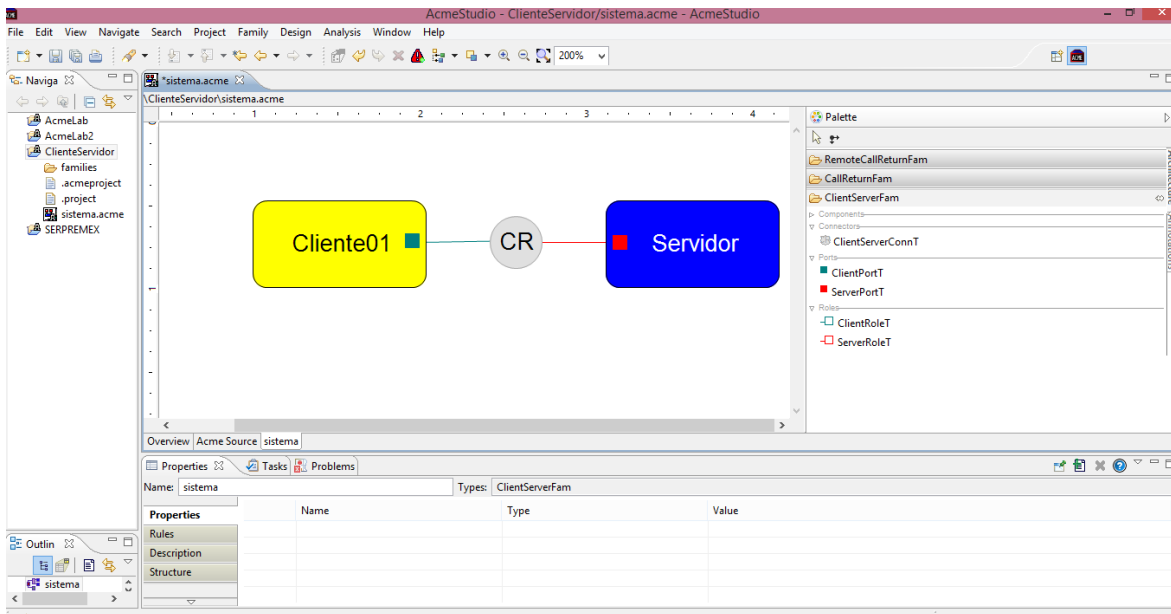


Figura 11.1 AcmeStudio.

11.2.4. Ejemplo de una especificación utilizando Acme

El ejemplo presentado en esta sección, es la especificación de la arquitectura de un pequeño sistema de monitoreo cuya integración a un sistema tecnológico puede mejorar su disponibilidad.

11.2.4.1. Contexto

La fiabilidad es la capacidad de un sistema o componente para desempeñar sus funciones especificadas, cuando se usa bajo condiciones y un periodo de tiempo

²⁵ El análisis de planificación en sistemas de tiempo real consiste en realizar el análisis y prueba de los procesos y tareas en cuanto a su cumplimiento, y los métodos de planificación usados en las aplicaciones, tales como algoritmos de planificación.

determinados. Una de las subdivisiones de la fiabilidad es la disponibilidad, que es la capacidad del sistema o componente de estar operativo y accesible para su uso cuando sea requerido (ISO/IEC 25010).

La disponibilidad está relacionada con la seguridad y la eficiencia de desempeño, que se refiere al tiempo y la habilidad del sistema de satisfacer sus requerimientos de tiempo (Bass, Clements y Kazman 2012).

Para que un sistema cuente con una adecuada disponibilidad, requiere estar libre de defectos y no fallar, es decir, que el sistema no se desvíe de su especificación debido a defectos internos o externos (Bass, Clements y Kazman 2012).

Existen diversos tipos de tácticas que se pueden implementar para lograr la disponibilidad en un sistema, entre éstas se encuentran las tácticas para detectar fallas, tácticas para la recuperación ante fallas, tácticas para la reintroducción de componentes fallidos tras su corrección, y tácticas para la prevención de fallas. (Bass, Clements y Kazman 2012).

Una de las tácticas de detección de fallas es el uso de un monitor, que es un componente que permite monitorear el “estado de salud” de varias partes de un sistema (procesadores, memorias, etc.) y proporcionar información sobre la congestión en una red u otros recursos compartidos, la negación de accesos a servicios, o características de la eficiencia de desempeño.

11.2.4.2. Descripción de la arquitectura del sistema

El sistema presentado, tiene como objetivo monitorear características de la eficiencia de desempeño de componentes en un sistema tecnológico para detectar posibles fallas o estados que afecten su disponibilidad. La arquitectura de dicho sistema será definida mediante elementos de una familia de sistemas que será previamente especificada.

El sistema de monitoreo que se presenta, debe recabar datos sobre la eficiencia de desempeño de los componentes que lo conforman utilizando programas “proveedores de datos”. Dichos programas entregan los datos a un programa “monitor” que se encarga de reunir, analizar y administrar la información, y de generar alertas sobre posibles fallas o estados que afecten la disponibilidad del sistema. La información generada por el monitor es entregada a programas “consumidores de datos”, encargados de desplegar la información para su análisis, visualización, generación de reportes. A continuación, se describe a detalle los elementos que conforman la arquitectura del sistema:

- Proveedores de datos: recaban datos sobre la eficiencia de desempeño de un componente individual del sistema del que forman parte, y los entregan al “monitor”. Los datos son manejados por medio de objetos de datos, que cuentan con atributos que representan datos específicos de la eficiencia de desempeño. Se consideran objetos de datos con los siguientes atributos:
 - Momento inicial: momento en que se inicia la recopilación de datos.
 - Momento final: momento en que se termina la recopilación de datos.
 - Latencia: tiempo que toma al componente responder a un evento.
 - Rendimiento: cantidad de eventos que pueden ser respondidos en un intervalo de tiempo dado (tasa de procesamiento).

- Capacidad: cantidad de demanda que puede ser colocada en el sistema mientras continúa satisfaciendo la latencia y el rendimiento.
- Monitor: proporciona una infraestructura para reunir y administrar la información del sistema. Compara constantemente los valores reportados por los proveedores de datos sobre los componentes monitoreados, con los valores de ciertos umbrales, y despliega una alerta si un valor excede o está por debajo de dichos umbrales. La información sobre las alertas es transferida a consumidores de datos.
- Consumidores de datos: programas que leen los datos proporcionados por el monitor, y presentan (despliegan) la información en forma de alertas relacionadas a posibles fallas producidas o propensas a producirse en el sistema. Permiten generar reportes confiables y rápidos sobre errores percibidos, y sobre valores que exceden o caen debajo de un valor de umbral particular. Las alertas contienen una descripción, y un indicador de estado con un valor numérico.

A continuación se presenta la definición del sistema antes descrito en Acme. Se define una familia de sistemas con los tipos de componentes, conectores y propiedades, y las restricciones necesarias para definir un sistema con las características mencionadas. Posteriormente se define un sistema basado en esa familia.

11.2.4.3. Especificación de una familia de sistemas

Los tipos de componentes de la familia son: proveedores de datos, monitores, y consumidores de datos.

Los tipos de conectores de la familia son:

- ProveedorMonitor: componente que conecta la salida de datos de un componente tipo proveedor de datos y la entrada de datos de uno tipo monitor. Requiere un rol de lectura y uno de escritura.
- MonitorConsumidor: componente que conecta la salida de un componente tipo monitor y uno tipo consumidor de datos. Requiere un rol de lectura y uno de escritura.

Los tipos de puertos que requieren los componentes de la familia son:

- SalidaProveedor: puerto de salida de componentes tipo "ProveedorDatos".
- EntradaMonitor: puerto de entrada de componentes tipo "Monitor".
- SalidaMonitor: puerto de salida de componentes tipo "Monitor".
- EntradaConsumidor: puerto de entrada de componentes tipo "ConsumidorDatos".

Los tipos de roles que requieren los conectores de la familia son:

- "FuenteProveedorMonitor: rol de fuente de datos para el conector "ProveedorMonitor".
- EscapeProveedorMonitor: rol de escape de datos para el conector "ProveedorMonitor".
- FuenteMonitorConsumidor: rol de fuente de datos para el conector "MonitorConsumidor".

- EscapeMonitorConsumidor: rol de escape de datos para el conector "MonitorConsumidor".

Los tipos de propiedades de la familia de sistemas son:

- Objetos de datos: la información sobre la eficiencia de desempeño de componentes del sistema.
- Alertas sobre el estado de la eficiencia de desempeño.

Las restricciones de la familia de sistemas son:

- Todos los componentes deben ser de los 3 tipos de la familia: proveedores de datos, monitores, o consumidores de datos.
- Todos los tipos de conectores de un sistema deben ser ProveedoresMonitor, o MonitorConsumidores.
- Solo debe haber un monitor en cada sistema.
- Debe haber al menos un proveedor de datos en cada sistema.
- Debe haber al menos un consumidor de datos en cada sistema.
- Deben estar conectados todos los proveedores con el monitor.
- Deben estar conectados todos los consumidores con el monitor.
- Para conectar un proveedor de datos con monitor, se debe usar un conector tipo "ProveedorMonitor".
- Para conectar un monitor con un consumidor de datos, se debe usar un conector tipo "MonitorConsumidor".

El código 11.6 presenta la especificación textual en Acme de la familia de sistemas antes descrita.

//Especificación de la familia "MonitoresAlertas"

Family MonitoresAlertas = {

//Especificación de los tipos de propiedades

Property Type ObjetoDatos = **Record** [fechaHoraIni : string;
 fechaHoraFin : string; latencia : float;
 redimiento : float; capacidad : float;];

Property Type Alerta = **Record** [descripcion : string;
 importancia : int; tipo : string;];

//Especificación de los tipos de puertos

Port Type salidaProveedor = {}

Port Type entradaMonitor = {}

Port Type salidaMonitor = {}

Port Type entradaConsumidor = {}

//Especificación de los tipos de componentes

Component Type Monitor = {

Port SalidaDatosMonitor : salidaMonitor = {}

Port EntradaDatosMonitor : entradaMonitor = {}

Property entradaProveedor : **Sequence** <ObjetoDatos>;

Property salidaConsumidor : **Sequence** <Alerta>;

```

Property maxClientesConcurrentes : int = 10;
Property tasa : int;
}

Component Type ConsumidorDatos = {
  Port EntradaDatos : entradaConsumidor = {}
  Property entradaDatos : Sequence <Alerta>;
}

Component Type ProveedorDatos = {
  Port SalidaDatosProveedor : salidaProveedor = {}
  Property iniciadoAutomatico : boolean = true;
  Property tipoDatoGenerado : ObjetoDatos;
  Property intervaloSalida : float = 60;
}

  //Especificación de los tipos de roles
Role Type fuenteProveedorMonitor = {}
Role Type escapeProveedorMonitor = {}
Role Type fuenteMonitorConsumidor = {}
Role Type escapeMonitorConsumidor = {}

  //Especificación de los tipos de conectores
Connector Type ProveedorMonitor = {
  Role entrada : fuenteProveedorMonitor = {}
  Role salida : escapeProveedorMonitor = {}
  Property tamMaxBuffer : int = 10;
}

Connector Type MonitorConsumidor = {
  Role entrada : fuenteMonitorConsumidor = {}
  Role salida : escapeMonitorConsumidor = {}
  Property tamMaxBuffer : int = 10;
}

  //Especificación de algunas reglas de la familia

  //Regla para especificar los tipos de componentes de la familia
rule tiposComponentes = invariant forall c in self.COMPONENTS |
  satisfiesType(c, Monitor) OR satisfiesType(c, ProveedorDatos) OR
  satisfiesType(c, ConsumidorDatos);
  //Regla para especificar los tipos de conectores de la familia
rule tiposConectores = invariant forall cnn in self.CONNECTORS |
  satisfiesType(cnn, ProveedorMonitor) OR satisfiesType(cnn,
  MonitorConsumidor);

  //Regla para especificar la cantidad de componentes tipo "Monitor"
rule unMonitor = invariant size({select c in self.COMPONENTS |
  satisfiesType(c, Monitor)}) == 1;

  //Regla para especificar la cantidad de componentes tipo "Proveedor"

```

```

rule numProveedoresDatos = invariant size({select cp in
  self.COMPONENTS | satisfiesType(cp, ProveedorDatos)}) >= 1;

  //Regla para especificar la relación entre un componente "ProveedorDatos"
  //y un conector "ProveedorMonitor"
rule conexionProveedoresMonitorFuente = invariant exists cp in
  self.COMPONENTS | exists c in self.CONNECTORS | (satisfiesType(cp,
  ProveedorDatos) AND satisfiesType(c, ProveedorMonitor) AND
  attached(cp, c) AND (forall pcp : Port in cp.PORTS | forall rc :
  Role in c.ROLES | attached(pcp, rc) -> (satisfiesType(pcp,
  salidaProveedor) AND satisfiesType(rc, fuenteProveedorMonitor)))));

```

Código 11.6 Ejemplo de la especificación de una familia de sistemas con Acme.

11.2.4.4. Especificación de un sistema

El sistema que será definido, se basa en la familia especificada anteriormente, y cuenta con los siguientes elementos:

- Componentes: ProveedorDatos0, ProveedorDatos1, Monitor y ConsumidorDatos0.
- Conectores: MonitorConsumidor0, ProveedorMonitor0, ProveedorMonitor1.
- Asociaciones entre:
 - ConsumidorDatos0.EntradaDatos y MonitorConsumidor0.salida.
 - Monitor0.SalidaDatosMonitor y MonitorConsumidor0.entrada.
 - Monitor0.EntradaDatosMonitor y ProveedorMonitor0.salida.
 - ProveedorDatos0.SalidaDatosProveedor y ProveedorMonitor0.entrada.
 - Monitor0.EntradaDatosMonitor y ProveedorMonitor1.salida.
 - ProveedorDatos1.SalidaDatosProveedor y ProveedorMonitor1.entrada.

En el código 11.7 se presenta la especificación del sistema.

```

import familias/MonitoresAlertas.acme;
System SystemaMonitor : MonitoresAlertas = new MonitoresAlertas extended
with {

  //Especificación de las instancias de los componentes del sistema
  Component ProveedorDatos0 : ProveedorDatos = new ProveedorDatos
  extended with {}
  Component Monitor0 : Monitor = new Monitor extended with {}
  Component ConsumidorDatos0 : ConsumidorDatos = new ConsumidorDatos
  extended with {}
  Component ProveedorDatos1 : ProveedorDatos = new ProveedorDatos
  extended with {}

  //Especificación de las instancias de los conectores del sistema
  Connector MonitorConsumidor0 : MonitorConsumidor = new
  MonitorConsumidor extended with {}
  Connector ProveedorMonitor0 : ProveedorMonitor = new ProveedorMonitor

```

```

        extended with {}
Connector ProveedorMonitor1 : ProveedorMonitor = new ProveedorMonitor
        extended with {}

        //Especificación de los attachments entre componentes y conectores
Attachment ConsumidorDatos0. EntradaDatos to
    MonitorConsumidor0. salida;
Attachment Monitor0. SalidaDatosMonitor to MonitorConsumidor0. entrada;
Attachment Monitor0. EntradaDatosMonitor to ProveedorMonitor0. salida;
Attachment ProveedorDatos0. SalidaDatosProveedor to
    ProveedorMonitor0. entrada;
Attachment Monitor0. EntradaDatosMonitor to ProveedorMonitor1. salida;
Attachment ProveedorDatos1. SalidaDatosProveedor to
    ProveedorMonitor1. entrada;
}

```

Código 11.7 Ejemplo de un sistema de monitoreo especificado con Acme.

El modelo gráfico del sistema de monitoreo se presenta en la figura 11.2.

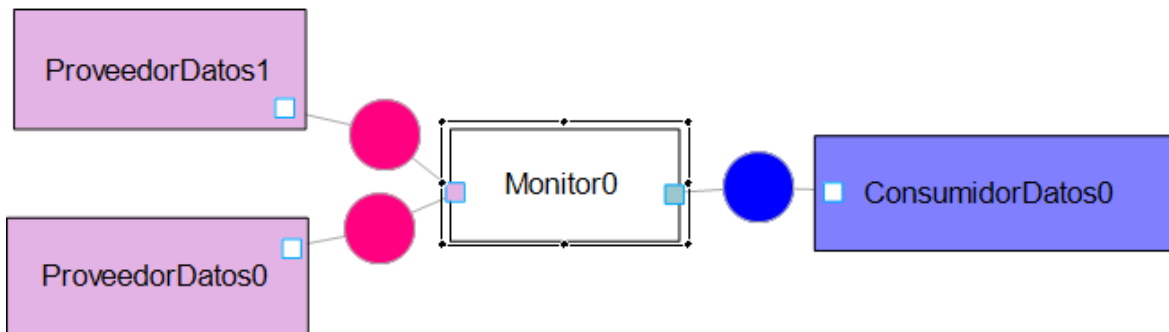


Figura 11.2 Modelo del sistema de monitoreo.

11.3. Especificación y modelado utilizando un LDA de propósito específico (AADL)

El *Architecture Analysis & Design Language* (AADL), es un Lenguaje de Descripción de Arquitectura que proporciona conceptos de modelado formal para la descripción y el análisis de arquitecturas de sistemas en términos de componentes y sus interacciones. Incluye facilidades para modelar hardware y software, y cuenta con abstracciones para (Feiler et al. 2006):

- Especificar y analizar sistemas embebidos de tiempo real, sistemas de sistemas complejos, y sistemas con capacidades especiales de eficiencia de desempeño.
- Mapear elementos de software en elementos de hardware.

Además, AADL cuenta con soporte para la especificación de requerimientos de tiempo, el análisis de confiabilidad y seguridad, y la predicción y validación de características de tiempo de ejecución, tales como disponibilidad.

Los elementos básicos de modelado que incluye AADL son los componentes y las interacciones entre ellos. Los componentes que soporta AADL en su versión 2.0 (Feiler et al. 2010) son los siguientes:

1. Componentes de software:
 - a. Hilo: unidad de ejecución concurrente que puede organizarse en grupos.
 - b. Grupo de hilos: abstracción que permite organizar lógicamente hilos, datos, y grupos de hilos en un proceso.
 - c. Proceso: espacio de direcciones protegido, cuya protección es proporcionada por otros componentes que acceden al proceso.
 - d. Datos: representa tipos de datos, y datos estáticos (como datos numéricos o texto fuente) en un sistema.
 - e. Subprograma: representa texto fuente ejecutable secuencialmente, es un componente que puede ser llamado con o sin parámetros, que opera en datos, o que proporciona funciones de servidor a los componentes que lo solicitan.
 - f. Grupo de subprogramas: representa bibliotecas de subprogramas que pueden ser instanciadas (por ejemplo, declaradas como subcomponentes).
2. Componentes de plataforma de ejecución (hardware)
 - a. Procesador: componente que programa y ejecuta hilos.
 - b. Procesador virtual: representa un recurso lógico que es capaz de programar y ejecutar hilos.
 - c. Memoria: componente que almacena código y datos.
 - d. Dispositivo: representa entidades que interactúan con el ambiente exterior de un sistema, tal es el caso de sensores o sistemas *stand-alone* como el *Global Positioning System* (GPS).

- e. Bus: componente que representa hardware y protocolos de comunicación asociados, y que habilita las interacciones entre otros componentes de plataforma de ejecución (procesadores, memorias y dispositivos).
 - f. Bus virtual: representa un bus lógico tal como un canal virtual o un protocolo de comunicación
3. Componentes de composición
 - a. Sistema: elemento de diseño que habilita la integración de otros componentes en distintas unidades en la arquitectura. Pueden consistir en otros sistemas, así como de componentes de hardware y software.
 4. Componentes abstractos: representan modelos de componentes. Permiten el modelado arquitectónico conceptual, para después realizar un refinamiento en una arquitectura de tiempo de ejecución.

La tabla 11.3 incluye la representación gráfica de los elementos de AADL antes descritos.



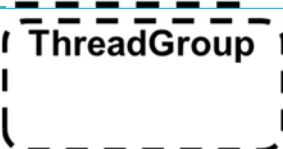
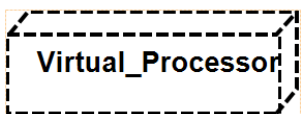
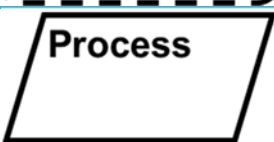
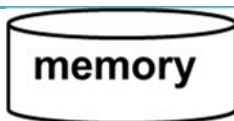



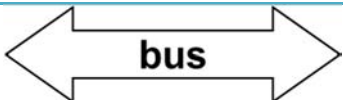

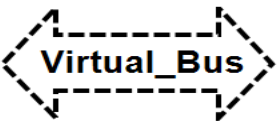
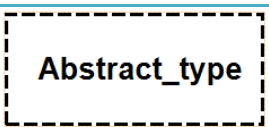
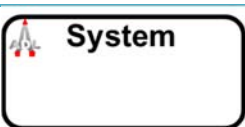

Componente	Representación gráfica	Componente	Representación gráfica
Hilo		Procesador	
Grupo de hilos		Procesador virtual	
Proceso		Memoria	
Datos		Dispositivo	
Subprogramas		Bus de datos	
Grupo de subprogramas		Bus de datos virtual	
Abstracción		Sistema	

Tabla 11.2 Notación gráfica de AADL.

Nótese que la mayoría de las representaciones gráficas de los elementos son similares a las usadas en UML, a excepción de aquellas que son propias de AADL y que contienen el

símbolo . Por tal motivo, puede resultar sencillo familiarizarse con la notación de AADL si antes se ha utilizado UML.

Por otra parte, AADL cuenta con una notación gráfica, mostrada en la figura 11.3, para representar las relaciones y ligas entre componentes.

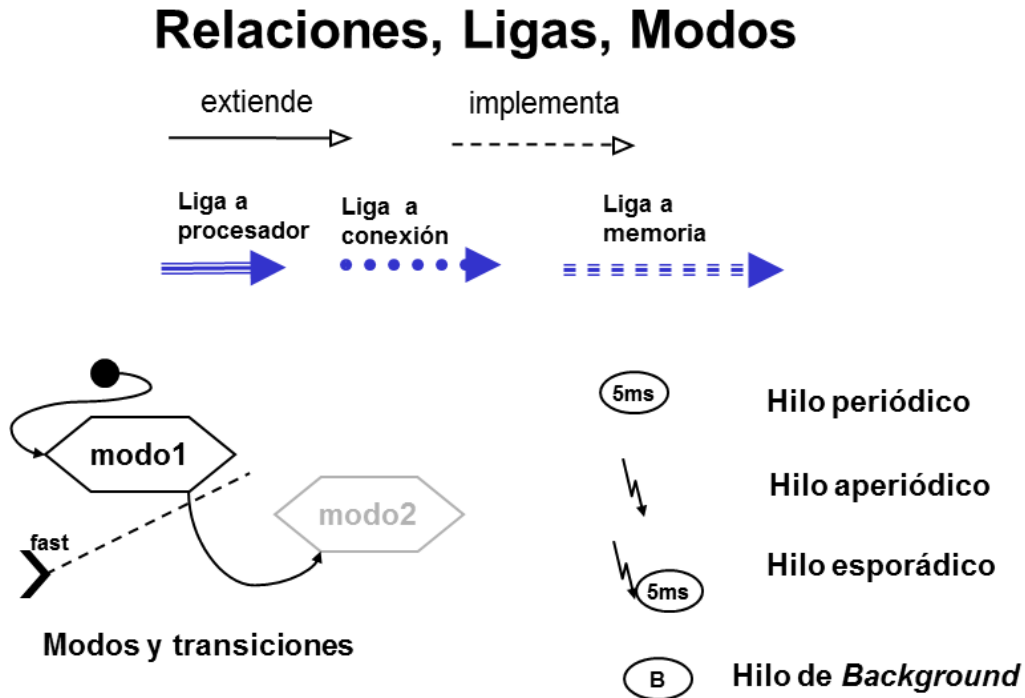


Figura 11.3 Notación gráfica de relaciones y ligas entre componentes en AADL.

En cuanto a las interacciones entre componentes, AADL soporta las siguientes:

- Conexiones de puerto: son relaciones explícitas declaradas entre puertos o entre grupos de puertos, y que habilitan el intercambio direccional de datos y eventos entre los componentes.
- Llamadas a subprogramas: son declaraciones explícitas en implementaciones de componentes, que habilitan el acceso “*call/return*” a subprogramas.
- Conexiones de parámetros: relaciones entre elementos de datos asociados con llamadas a subprogramas.

Para soportar las interacciones entre componentes, AADL cuenta con los siguientes elementos: puertos, grupos de puertos, accesos a subcomponentes, llamadas a subprogramas, y el intercambio y compartición de datos en subprogramas.

La figura 11.4 incluye las convenciones gráficas para los elementos de interacción entre componentes de AADL.

Puertos, Subprogramas, Conexiones

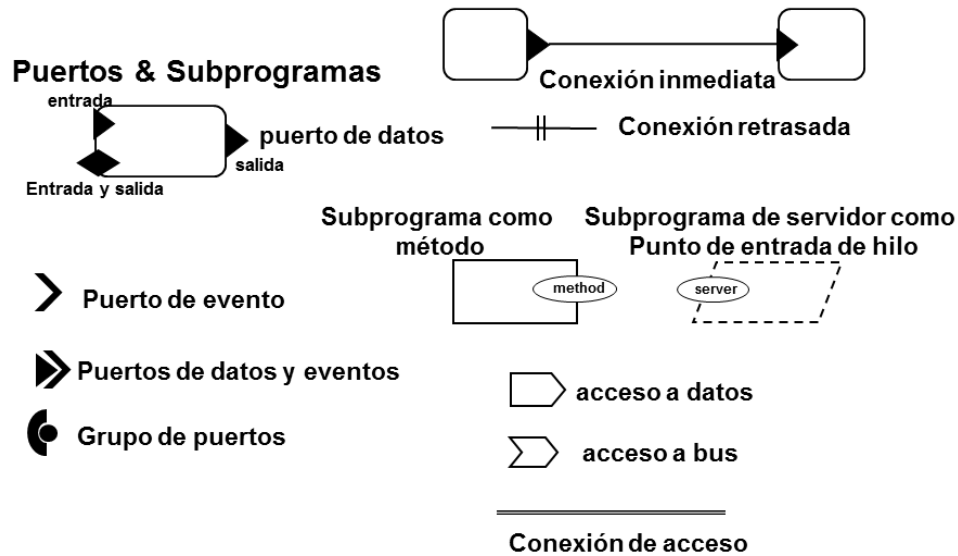


Figura 11.4 Notación gráfica para interacciones en AADL.

11.3.1. Especificación arquitectónica

En las siguientes secciones, se presentan las consideraciones para realizar especificaciones arquitectónicas utilizando AADL mediante componentes y sus interacciones.

11.3.1.1. Especificación de componentes

Para especificar un componente, en AADL existen los siguientes tres niveles principales de descripción (Rosen y Tilman 2005):

- Tipo: especifica las interfaces externas del componente.
- Implementación: especifica el contenido del componente, su estructura interna.
- Instanciación de un tipo de componente.

Una declaración de “tipo” de un componente, se conforma de una cláusula de definición y las siguientes sub-clausulas descriptivas:

- Características (*features*), que representan los puntos de interacción con otros componentes.
- Flujos (*flows*), que especifican distintos canales abstractos de transferencia de información.
- Propiedades (*properties*) del componente que aplican a todas las instancias de ese componente.
- Extensiones (*extends*), que permiten extender un tipo de componente a partir de otro.

Por otra parte, una declaración de “implementación” se conforma de las siguientes subclausulas:

- Subcomponentes (*subcomponents*), llamadas (*calls*), y conexiones (*connections*), que especifican la composición de un componente como una colección de componentes y sus interacciones.
- Flujos (*flows*), que representan implementaciones de especificaciones de flujo en un tipo de componente o flujos *end-to-end* para ser analizados.
- Modos (*modes*), que representan modos operacionales alternativos que pueden manifestarse como configuraciones alternas de subcomponentes, secuencias de llamadas, conexiones, secuencias de flujo, y propiedades.
- Propiedades (*properties*), que definen características intrínsecas de un componente. Hay propiedades predefinidas para cada implementación de un componente.
- Extensiones (*extends*), que permiten heredar características de la implementación original de un componente y de todos sus predecesores. Permiten extender otras implementaciones.
- Refinamiento (*refines type*), que permite refinar otras implementaciones.

El código 11.8 presenta un ejemplo de la especificación de un componente de tipo “process” llamado “proceso_de_control”, que cuenta con un puerto de entrada y uno de salida, y que está constituido por dos hilos: “control_entrada” y “control_salida”. El ejemplo se basa en el presentado por Feiler, Gluch y Hudack (2006), y se incluyen sus especificaciones de tipo e implementación.

```

//Especificación del componente "proceso_de_control" del tipo "process"
process proceso_de_control
    //Especificación de los puntos de interacción con otros componentes
    //Cuenta con un puerto de entrada de datos del tipo "datos_sensor",
    //y con un puerto de salida de datos del tipo "datos_comando"
    features
        input: in data port datos_sensor;
        output: out data port datos_comando;
end proceso_de_control;

//Especificación de la implementación del componente
process implementation proceso_de_control . control_velocidad
    //Estructura interna del componente. Se compone de un hilo llamado
    //"control_de_entrada" del tipo "control_entrada" y de un hilo
    //llamado "control_de_salida" del tipo "control_salida".
    subcomponents
        control_de_entrada: thread
            control_entrada. entrada_procesamiento_01;
        control_de_salida: thread
            control_salida. salida_procesamiento_01;
end proceso_de_control . control_velocidad;

```

Código 11.8 Ejemplo de la definición de un componente con AADL.

Adicionalmente, los tipos de datos e hilos utilizados en el ejemplo anterior (código 11.8), pueden ser especificados como se muestra en el siguiente código (11.9).

```

//Especificación del tipo e implementación del hilo "control_entrada"
thread control_entrada
end control_entrada;
thread implementation control_entrada.entrada_procesamiento_01
end control_entrada.entrada_procesamiento_01;

//Especificación del tipo e implementación del hilo "control_salida"
thread control_salida
end control_salida;
thread implementation control_salida.salida_procesamiento_01
end control_salida.salida_procesamiento_01;

//Especificación del tipo de datos "datos_sensor"
data datos_sensor
end datos_sensor;

//Especificación del tipo de datos "datos_comando"
data datos_comando
end datos_comando;

```

Código 11.9 Ejemplo de la definición de tipos de elementos con AADL.

Puede observarse, que en el caso de los datos, no se requiere su especificación de implementación para poder ser usados. En la figura 11.5 se presenta la representación gráfica del ejemplo anterior generada con la herramienta Osate.

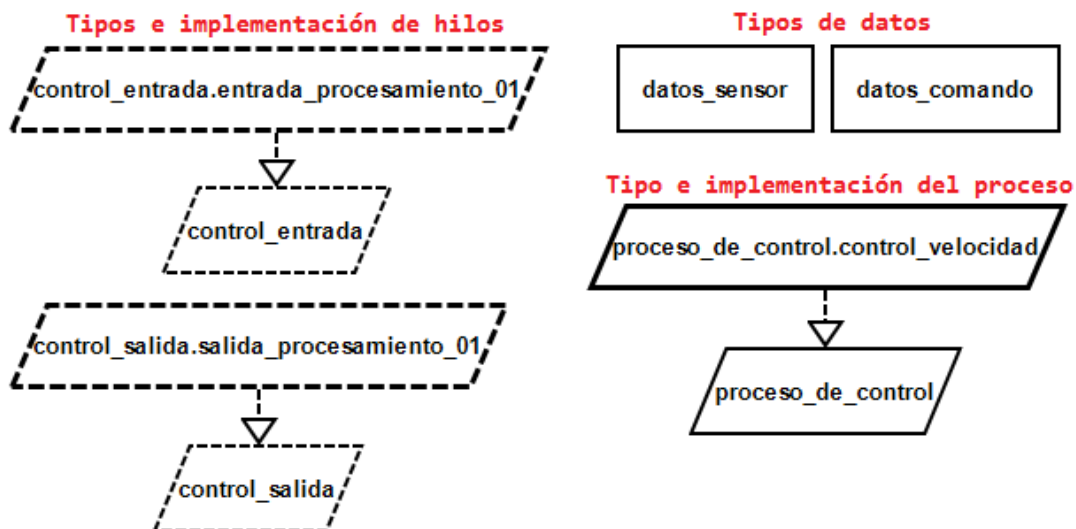


Figura 11.5 Representación gráfica de un ejemplo con AADL.

La figura 11.6 muestra las representaciones gráficas de tipo e implementación del proceso “proceso_de_control”.

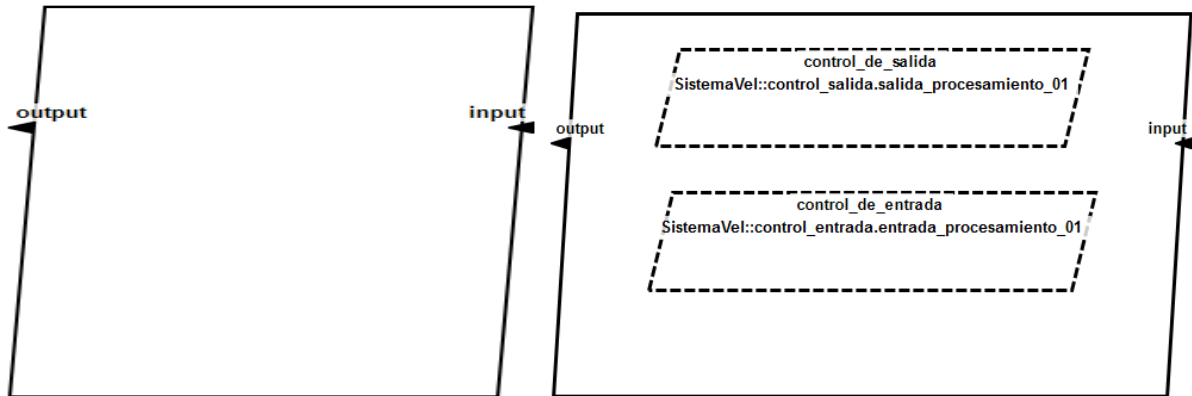


Figura 11.6 Representación gráfica del proceso *proceso_de_control*.

En la representación gráfica de la implementación del proceso (gráfica de la derecha) se puede apreciar su estructura interna.

Adicionalmente, es importante mencionar que cada tipo de elemento tiene restricciones asociadas, las cuales permiten modelar y especificar una arquitectura correctamente. Por ejemplo, una restricción en un procesador, es que solo puede ser subcomponente de un sistema. Para especificar correctamente una arquitectura, se recomienda conocer a detalle estas restricciones. La sintaxis específica de AADL y sus restricciones, puede ser consultada en documentos como la carta sintáctica de AADL V2 (2011).

11.3.1.2. Especificación de interacciones

Para especificar las interacciones entre componentes, AADL soporta elementos de interfaz que incluyen puertos, grupos de puertos, acceso a subcomponentes, llamadas a subprogramas, e intercambio y compartición de datos en subprogramas. Dichos elementos deben ser declarados en la sección de “*features*” de un componente en su declaración de tipo (*type*) (Feiler, Gluch y Hudack 2006).

En el código 11.10 se presenta un ejemplo de cómo se puede especificar una interacción entre puertos. El ejemplo presentado consiste en un sistema que recibe datos de dos puertos de entrada, los procesa e imprime a la salida texto procesado. El sistema cuenta con dos subsistemas, uno para hacer una fusión o “*merge*” de los datos provenientes de las dos fuentes de entrada (puertos del sistema), y uno para procesar el texto resultante de la fusión. Los puertos de los subsistemas están conectados, de modo que la salida de uno se conecta con la entrada del otro, y a su vez, los puertos de entrada del subsistema “*merge*” se conectan con los puertos de entrada del sistema, de la misma forma, el puerto de salida del subsistema de procesamiento, interactúa con el puerto de salida del sistema.

```
//Especificación del tipo e implementación del sistema
system sistemaProcesamientoTexto
features
    Fuente_datos_01: in data port texto;
```

```

    Fuente_datos_02: in data port texto;
    Salida_datos_01: out data port texto;
end sistemaProcesamientoTexto;

system implementation sistemaProcesamientoTexto. implementation
subcomponents
    SubsistemaMerge: system merge. implementation;
    SubsistemaProcesa: system procesa. implementation;
Connections
    //Especificación de las conexiones entre puertos

    entradaMerge01: port Fuente_datos_01 ->
        SubsistemaMerge. Entrada_merge_01;
    entradaMerge02: port Fuente_datos_02 ->
        SubsistemaMerge. Entrada_merge_02;
    conexionSubsistemas: port
        SubsistemaMerge. Salida_merge_01 ->
        SubsistemaProcesa. Entrada_procesamiento_01;
    salidaProcesamiento01: port
        SubsistemaProcesa. Salida_procesamiento_01->
        Salida_datos_01 ;
end sistemaProcesamientoTexto. implementation;

//Especificación del tipo e implementación de los subsistemas "merge" y
"procesa"
//Especificación del subsistema "merge"
system merge
    features
        Entrada_merge_01: in data port texto;
        Entrada_merge_02: in data port texto;
        Salida_merge_01: out data port texto;
end merge;

system implementation merge. implementation
end merge. implementation;

//Especificación del subsistema "procesa"
system procesa
    features
        Entrada_procesamiento_01: in data port texto;
        Salida_procesamiento_01: out data port texto;
end procesa;

system implementation procesa. implementation
end procesa. implementation;

//Especificación del tipo de datos "texto"
data texto
end texto;

```

Código 11.10 Ejemplo de la especificación de una interacción con AADL.

En la figura 11.7 se presenta la representación gráfica del sistema, incluyendo los dos subsistemas y las interacciones entre ellos a través de sus puertos.

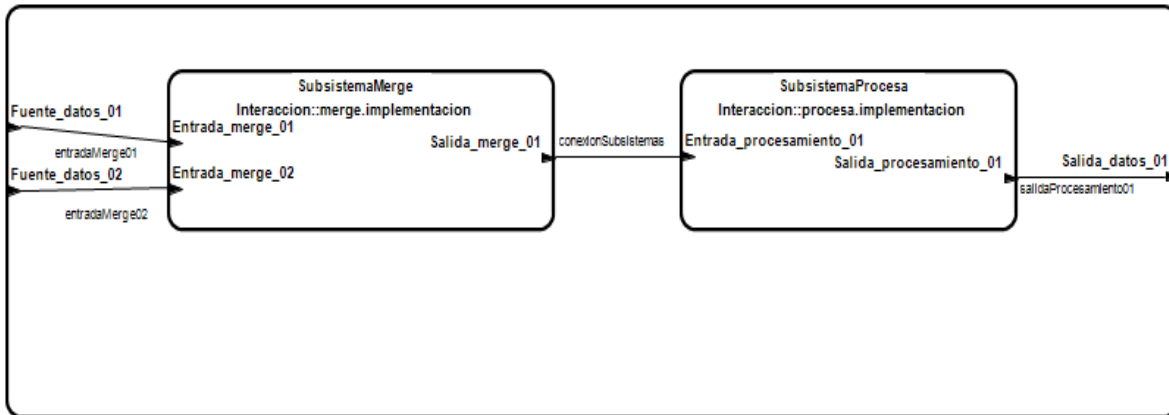


Figura 11.7 Representación gráfica del sistema *SistemaProcesamientoTexto*.

Puede notarse que, a diferencia de otros lenguajes como Acme y Wright, las interacciones entre componentes en AADL se especifican explícitamente en los componentes, mientras que en Acme y Wright, se especifican por medio de elementos “conectores”. Sin embargo, cabe resaltar que al construir un sistema con una arquitectura basada en componentes y conectores, estos últimos no siempre son construidos como elementos de software. Por ejemplo, un conector que representa una llamada a un programa, no representa un elemento de software real en un sistema, sino que dicha interacción (la llamada a un programa) debe ser implementada en un componente (otro programa por ejemplo). Por lo anterior, AADL podría proporcionar una forma más “real” de especificar las interacciones entre componentes.

11.3.1.3. Especificación de otras características de componentes en AADL

Además de los componentes y sus interacciones, AADL permite especificar otras características como modos, flujos y propiedades en los componentes.

Los modos representan estados de operación alternativos de un sistema o componente. Los modos pueden especificar diferentes secuencias de llamadas para ser usadas en un hilo o subprograma. Pueden también representar diferentes estados lógicos de cualquier componente, tales como hilos o subprogramas (Feiler, Gluch y Hudak 2006).

Los modos se especifican en la implementación de los componentes. El código 11.11 presenta un ejemplo de la especificación de modos.

```

thread implementation control.implementacion
  modes
    inicial : initial mode;
    control : mode;
end control.implementacion;

```

Código 11.11 Ejemplo de la definición de un modo con AADL.

Por otra parte, los flujos habilitan la descripción y análisis detallado de una ruta de información abstracta a través de un sistema. Los flujos son direccionales, y para especificarlos, se requieren las declaraciones de tipos e implementaciones de los componentes. Para un tipo de componente, una declaración de flujo incluye una fuente, un vertedero (*sink*) y una ruta de flujo (Feiler, Gluch y Hudak 2006).

Un ejemplo de la especificación de flujos es el presentado en el código 11.12.

```
device sensor
  features
    salida: out data port tipo_dato;
  flows
    flujo01: flow source evento;
end sensor;
```

Código 11.12 Ejemplo de la definición de flujos con AADL.

Por su parte, las propiedades son usadas para proporcionar información descriptiva sobre componentes, subcomponentes, características, conexiones, flujos, modos, y llamadas a subprogramas. Las propiedades son especificadas mediante un nombre, tipo, y un valor asociado (Feiler, Gluch y Hudak 2006). Un ejemplo de la especificación de una propiedad es el presentado en el código 11.13.

```
thread implementation procesamiento.implementacion
  properties
    Periodo => 200 ms;
end procesamiento.implementacion;
```

Código 11.13 Ejemplo de la definición de una propiedad con AADL.

11.3.2. La herramienta OSATE

Una de las herramientas para especificación y modelado con AADL, es la *Open Source AADL Tool Environment* (OSATE). Actualmente se encuentra disponible en la versión 2 y está basada en Eclipse. Esta herramienta soporta las tres representaciones de AADL: textual, gráfica y XML. Es una herramienta que puede ser descargada e instantáneamente está lista para usarse.

OSATE cuenta con una paleta con la representación gráfica de los elementos de AADL, y permite arrastrarlos hacia el área de trabajo y modelar una arquitectura gráficamente (figura 11.8). Por otra parte, permite especificar de manera textual una arquitectura mediante un editor gráfico.

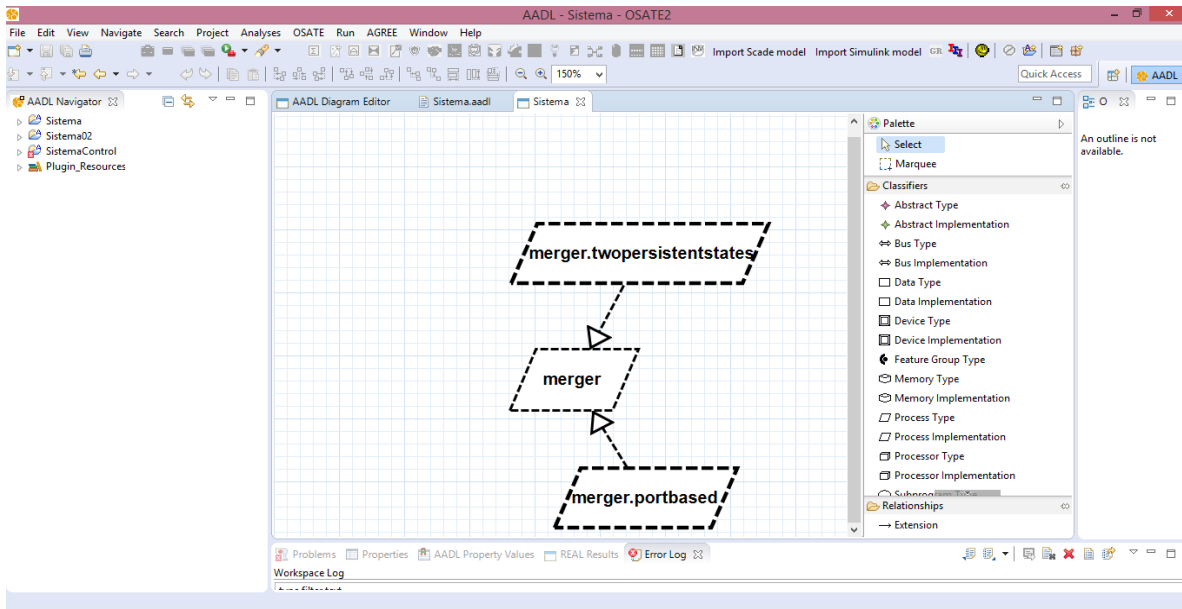


Figura 11.8 Herramienta Osate2.

11.3.3. Ejemplo

A continuación se presenta un ejemplo (código 11.14) sobre un sistema para controlar velocidad en un automóvil. Dicho sistema está compuesto por un proceso “control”, que a su vez está compuesto por dos hilos:

- “control_entrada”: hilo que cuenta con un puerto de entrada de datos de velocidad, uno para entrada de eventos relacionados con frenos, un puerto de entrada de datos y eventos relacionados con velocidades preestablecidas, y un puerto de salida de datos procesados.
- “control_salida”: cuenta con un puerto de datos de entrada, y uno de datos de salida.

El proceso “control”, que incluye ambos hilos, cuenta con un puerto de entrada de datos de velocidad, un puerto de entrada de eventos relacionados con la velocidad, y un puerto de salida de datos de comandos.

El ejemplo está basado en Feiler, Gluch y Hudack (2006), sin embargo, en este trabajo se adaptó para que esté especificado de acuerdo a la versión 2.0 de AADL.

```
//Especificación del tipo e implementación de los hilos
//Especificación del tipo de hilo "control_entrada"
thread control_entrada
  features
    //Especificación de los puertos del hilo
    dato_velocidad_entrada: in data port velocidad_pura;
    evento_freno: in event port;
    dato_velocidad_establecida: in event data port
      velocidad_pura_establecida;
```



```

        salida_datos_c: out data port datos_procesados;
    end control_entrada;

    //Especificación de la implementación del hilo "control_entrada"
    thread implementation control_entrada.entrada_procesamiento_01
    end control_entrada.entrada_procesamiento_01;

    //Especificación del tipo de hilo "control_salida"
    thread control_salida
        features
        //Especificación de los puertos del hilo
        c_entrada_datos: in data port datos_procesados;
        c_comando_salida: out data port datos_comando;
    end control_salida;

    //Especificación de la implementación del hilo "control_salida"
    thread implementation control_salida.salida_procesamiento_01
    end control_salida.salida_procesamiento_01;

    //Especificación del proceso "control"
    process control
        features
        //Especificación de los puertos del proceso
        velocidad: in data port velocidad_pura;
        frenos: in event port;
        velocidad_establecida: in event data port
            velocidad_pura_establecida;
        comando_aceleracion: out data port datos_comando;
    end control;

    //Especificación de la implementación del proceso "control"
    process implementation control.control_velocidad
        //Especificación de los subcomponentes del proceso
        subcomponents
        input: thread control_entrada.entrada_procesamiento_01;
        control_plus_output: thread
            control_salida.salida_procesamiento_01;

        //Especificación de las conexiones del proceso
        connections
        speed_in: port velocidad ->
            input.dato_velocidad_entrada;
        brake_in: port frenos -> input.evento_freno;
        set_speed_in: port velocidad_establecida ->
            input.dato_velocidad_establecida;
        c_data_transfer: port input.salida_datos_c ->
            control_plus_output.c_entrada_datos;
        a: port control_plus_output.c_comando_salida ->
            comando_aceleracion;
    end control.control_velocidad;

    //Especificación de los tipos de datos

```

```

data velocidad_pura
end velocidad_pura;
data velocidad_pura_establecida
end velocidad_pura_establecida;
data datos_comando
end datos_comando;
data datos_procesados
end datos_procesados;

```

Código 11.14 Ejemplo de un sistema de control de velocidad especificado con AADL.

La figura 11.9 presenta la representación gráfica de los componentes del sistema antes descrito, y la figura 11.10 la representación de la implementación del proceso principal del sistema.

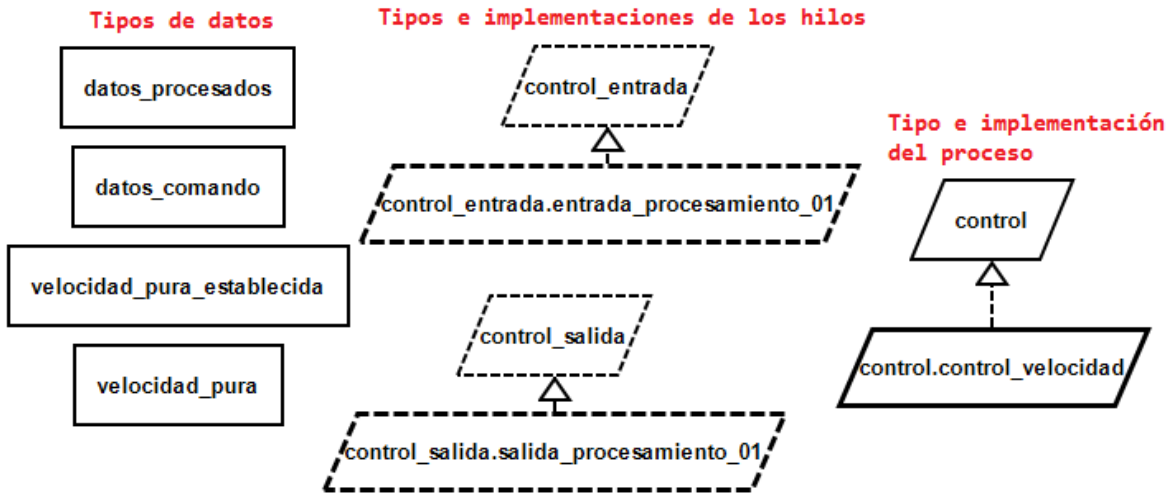


Figura 11.9 Representación gráfica de los elementos arquitectónicos.

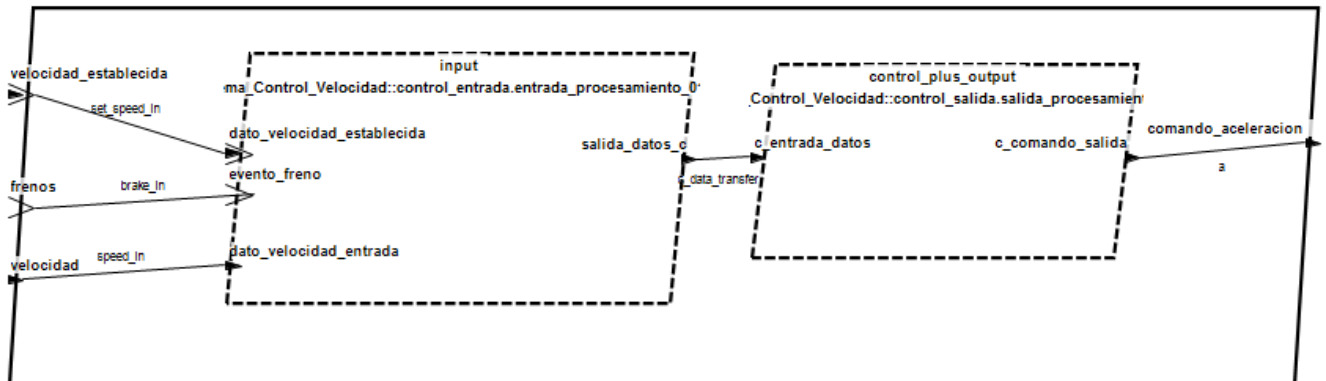


Figura 11.10 Representación gráfica del proceso *control*.

11.4. Construcción de modelos utilizando un Lenguaje de Descripción de Arquitectura flexible y extensible (xADL)

XADL es un Lenguaje de Descripción de Arquitectura extensible y flexible, basado en XML, y diseñado para enfrentar los problemas de extensibilidad de los Lenguajes de Descripción de Arquitectura convencionales. Además, xADL proporciona facilidades para explorar nuevas posibilidades y técnicas de modelado (Dashofy, Hoek y Taylor 2001).

XADL está basado en xArch, que es un estándar que proporciona extensibilidad para modelar arquitecturas de software. XArch cuenta con una notación basada en XML, y un núcleo común que puede servir para la representación de arquitecturas *stand-alone*, como un punto inicial para notaciones arquitectónicas basadas en XML más avanzadas, y como un mecanismo de intercambio para descripciones arquitectónicas.

Además de las características que xADL adquiere del núcleo de xArch, proporciona soporte para la prescripción arquitectónica (la “receta” para instanciar una arquitectura en un sistema en ejecución), un modelo de tipos e instancias, conceptos de administración de configuración a nivel arquitectura (versiones, opciones y variantes), y el mapeo de tipos de elementos a implementaciones de esos tipos (Dashofy, Hoek y Taylor 2001).

Es importante mencionar que xADL hace una clara distinción entre una prescripción arquitectónica (la plantilla de tiempo de ejecución que es usada para instanciar la arquitectura), y la descripción arquitectónica (que describe el estado en tiempo de ejecución del sistema) (Dashofy, Hoek y Taylor 2001).

XADL está definido como un conjunto de esquemas de XML, y un documento de modelado de xADL es un documento en XML que se ajusta a la sintaxis del núcleo de xArch y a las extensiones de xADL.

Es útil entender como están organizados los diferentes esquemas de XML que comprenden xADL en términos de sus dependencias. El conjunto actual de esquemas de xADL (en su versión 3.0) incluye:

- Soporte para modelar elementos de tiempo de ejecución y de tiempo de diseño de un sistema (componentes, conectores, interfaces y conexiones o *links*).
- Soporte para modelar tipos arquitectónicos.
- Conceptos de administración de configuración avanzados como versiones, opciones y variantes.
- Arquitecturas de familias de productos.

Los esquemas de xADL están organizados en los siguientes grupos de acuerdo a sus propósitos (Dashofy, Hoek y Taylor 2001):

- Un conjunto de esquemas establece la descripción y la prescripción de arquitecturas.
- Un conjunto de esquemas proporciona mapeos de componentes y conectores a sus implementaciones.

- Un conjunto de esquemas proporcionan soporte para la administración de la configuración arquitectónica.

Es importante mencionar que los elementos arquitectónicos estructurales con que cuenta xADL son componentes, conectores, conexiones entre componentes y conectores (*links*), e interfaces, las cuales representan un puerto en un componente, o un rol en un conector.

ArchStudio, es una herramienta que permite modelar gráfica y textualmente una arquitectura basada en xADL. *ArchStudio* cuenta con representaciones gráficas para los elementos estructurales de xADL, tal como se muestra en la tabla 11.3.



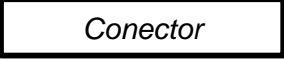

Elemento	Representación gráfica	Elemento	Representación gráfica
Componente		Interfaz	
Conector		Conexiones o links	

Tabla 11.3 Notación gráfica de xADL.

11.4.1. La herramienta ArchStudio

La herramienta que se usa para generar modelos en xADL es *ArchStudio*, y juega dos roles importantes en el desarrollo de arquitecturas: primero, es un ambiente de desarrollo de arquitecturas que permite modelar la estructura jerárquica de sistemas complejos, los tipos componentes, conectores, interfaces, y líneas de productos. En segundo lugar, es un ambiente de meta-modelado arquitectónico, ya que está basado en xADL, y permite definir y redefinir características de la sintaxis y semántica del lenguaje. Además, proporciona una plataforma ideal para expertos del dominio e investigadores de la arquitectura, que quieren investigar nuevos conceptos de modelado arquitectónico.

Las facilidades que proporciona *ArchStudio* son:

- Modelado arquitectónico con xADL.
- Visualización y edición arquitectónica través de la herramienta gráfica *Archipelago*. Además incluye otros editores textuales como *ArchEdit*.
- Análisis arquitectónico, ya que soporta diseño exploratorio en todos los editores, y permite evaluar la correctitud y consistencia de un modelo de arquitectura. El *framework Archlight* permite analizar las descripciones de arquitectura automáticamente utilizando diferentes criterios.

ArchStudio está basado Eclipse, y su instalación es sencilla. Para más información se puede consultar la página de *ArchStudio*.

11.4.2. Modelado arquitectónico

A diferencia de otras notaciones, la sintaxis y herramientas de xADL están destinadas a ser ampliadas para apoyar las necesidades específicas de un proyecto, por lo que se puede modelar cualquier arquitectura utilizando los esquemas actuales de xADL más las extensiones que requieran elaborarse. Para lo anterior, se requiere seguir ciertos pasos que son descritos a continuación.

11.4.2.1. Identificar el alcance de xADL y el alcance del sistema a desarrollar

Para comenzar a realizar un modelo arquitectónico, se requiere identificar las discrepancias entre lo que xADL puede modelar con su conjunto actual de esquemas, y lo que se quiere modelar para un sistema en particular.

Es importante conocer que los elementos con que cuenta xADL incluyen componentes, conectores, interfaces y conexiones. Por otra parte, es importante mencionar que existe un estilo predefinido basado en estos elementos, llamado “Myx” (página de *ArchStudio*).

Por otra parte, es muy útil conocer exactamente lo que será modelado, es decir, conocer las características de la arquitectura del sistema a modelar (sus componentes, conectores, comportamiento, etc.), ya que es importante identificar qué puede y qué no puede ser modelado con los esquemas actuales de xADL, los cuales pueden ser obtenidos del repositorio de *ArchStudio*.

11.4.2.2. Identificar extensiones a elaborar

Después de saber que se puede modelar con los esquemas actuales de xADL, y qué se requiere modelar, es importante decidir qué elementos de xADL se extenderán para soportar las necesidades de modelado que no son soportadas por los esquemas actuales de xADL. Se debe considerar si se requiere agregar nuevas características a los elementos estructurales (componentes, conectores, etc.), o si se requieren nuevos tipos de elementos.

11.4.2.3. Definir sintaxis de las extensiones

Se debe decidir cómo codificar sintácticamente las nuevas extensiones (esquemas XML) de xADL que serán usadas.

Un esquema XML permite definir la estructura de documentos XML, por lo que para generar un esquema se debe especificar:

- Los elementos y atributos que pueden aparecer en un documento.
- Los elementos que son elementos “hijos”.
- El número de elementos “hijo” que puede tener un elemento.
- Los tipos de elementos y atributos.
- Los valores por default o fijos para elementos y atributos.

11.4.2.4. Desarrollar extensiones

Los esquemas XML que sean elaborados para extender xADL, deben incorporar las nuevas construcciones de modelado que sean requeridas. También es conveniente validar los nuevos esquemas con herramientas como el “XML Schema Validator” (XSV), que es una herramienta de uso libre para validar esquemas XML (Thompson y Tobin 2007).

En *ArchStudio*, los esquemas XML son almacenados en *plug-ins* individuales de Eclipse. La herramienta “*Apigen*”, incluida en *AcmeStudio*, permite generar automáticamente bibliotecas de vinculación de datos para los esquemas alojados en esos *plug-ins*. Para mayor información se puede consultar la página de *ArchStudio*.

En el código 11.15 se presenta un pequeño esquema XML (basado en el esquema estructural de xADL 3.0), que permite modelar interfaces, las cuales cuentan con un id, un nombre, una dirección de información (entrada, salida, etc.).

```
//Definición del tipo de elemento "Interfaz"
<xs:complexType name="Interfaz">
  <xs:annotation>
    <xs:documentation>
      Una interfaz es un punto a través del cual,
      componente o conector se comunica con el mundo
      exterior.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="ext" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  // Definición de atributos
  <xs:attribute name="id" type="xs:ID"/>
  <xs:attribute name="nombre" type="xs:string"/>
  <xs:attribute name="direccion" type="Direccion"/>
</xs:complexType>

//Definición del tipo de elemento "Direccion" que define
//direcciones de interfaces
<xs:simpleType name="Direccion">
  <xs:annotation>
    <xs:documentation>
      Define las direcciones de interfaz permitidas.
      Éstas son:
      - ninguna: No hay dirección implicada.
      - entrada: Interfaz de entrada.
      - salida: Interfaz de salida.
      - entradasalida: Interfaz de entrada y salida.
    </xs:documentation>
  </xs:annotation>
  //Definición de los valores permitidos
  <xs:restriction base="xs:string">
    <xs:enumeration value="ninguna"/>
    <xs:enumeration value="entrada"/>
```

```

        <xs:enumeration value="salida"/>
        <xs:enumeration value="entradasalida"/>
    </xs:restriction>
</xs:simpleType>

```

Código 11.15 Ejemplo un esquema con xADL.

A continuación se presenta el esquema anterior (código 11.15), agregando una extensión para que, además de que las interfaces cuenten con un atributo sobre su dirección (entrada, salida, etc.), cuenten con un atributo para definir si la información relacionada a ellos son datos o eventos.

Para definir tales características adicionales, se requiere agregar un atributo al tipo de elemento "Interfaz", y definir un nuevo tipo de elemento "TipoInformacion" para definir los diferentes tipos de información relacionada a un puerto. A continuación (código 11.16) se presenta el esquema resaltando las extensiones hechas para soportar las características mencionadas.

```

//Definición del tipo de elemento "Interfaz"
<xs:complexType name="Interfaz">
  <xs:annotation>
    <xs:documentation>
      Una interfaz es un punto a través del cual,
      componente o conector se comunica con el mundo
      exterior.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="ext" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  // Definición de atributos
  <xs:attribute name="id" type="xs:ID"/>
  <xs:attribute name="nombre" type="xs:string"/>
  <xs:attribute name="direccion" type="Direccion"/>
  // Definición de un atributo adicional para soportar
  //tipos de información
  <xs:attribute name="tipo" type="TipoInformacion"/>
</xs:complexType>

//Definición del tipo de elemento "Direccion" que define
//direcciones de interfaces
<xs:simpleType name="Direccion">
  <xs:annotation>
    <xs:documentation>
      Define las direcciones de interfaz permitidas.
      Éstas son:
      - ninguna: No hay dirección implicada.
      - entrada: Interfaz de entrada.
      - salida: Interfaz de salida.
      - entradasalida: Interfaz de entrada y salida.
    </xs:documentation>
  </xs:annotation>
</xs:simpleType>

```

```

//Definición de los valores permitidos
<xs:restriction base="xs:string">
  <xs:enumeration value="ninguna"/>
  <xs:enumeration value="entrada"/>
  <xs:enumeration value="salida"/>
  <xs:enumeration value="entradasalida"/>
</xs:restriction>
</xs:simpleType>

// Definición del tipo de elemento TipoInformacion
<xs:simpleType name="TipoInformacion">
  <xs:annotation>
    <xs:documentation>
      Define los tipos de información permitidos en
      la interfaz.
      Éstos son:
      -ninguno: No hay tipo de información implicado.
      -datos: Es interfaz de datos.
      -eventos: Es interfaz de eventos.
      -datoevento: Es interfaz de datos y eventos.
    </xs:documentation>
  </xs:annotation>
  // Definición de valores permitidos
  <xs:restriction base="xs:string">
    <xs:enumeration value="ninguno"/>
    <xs:enumeration value="datos"/>
    <xs:enumeration value="eventos"/>
    <xs:enumeration value="datoeventos"/>
  </xs:restriction>
</xs:simpleType>

```

Código 11.16 Ejemplo un esquema que implementa una extensión con xADL.

11.4.2.5. Elaborar un modelo arquitectónico

Para elaborar un modelo arquitectónico basado en xADL, o extender uno existente, se pueden utilizar el editor gráfico *Archipelago*, o el editor de texto *ArchEdit*, ambos incluidos en *ArchStudio*.

Para la elaboración de un modelo, se debe especificar el conjunto de esquemas a los que se adhiere y asegurarse de seguir sus convenciones.

La sintaxis para elaborar un modelo de manera textual, es la sintaxis de XML, la cual se basa en la definición de bloques a través de etiquetas (*tags*). En la siguiente sección se presenta un ejemplo de un modelo elaborado en xADL.

11.4.3. Ejemplo

El ejemplo presentado en esta sección, es sobre el sistema de monitoreo de la sección 11.2.4 modelado con xADL 3.0. Se presenta su representación gráfica, y parte de su descripción textual.

La representación gráfica o modelo generado con *Archipelago*, en *ArchStudio*, se muestra en la figura 11.11.

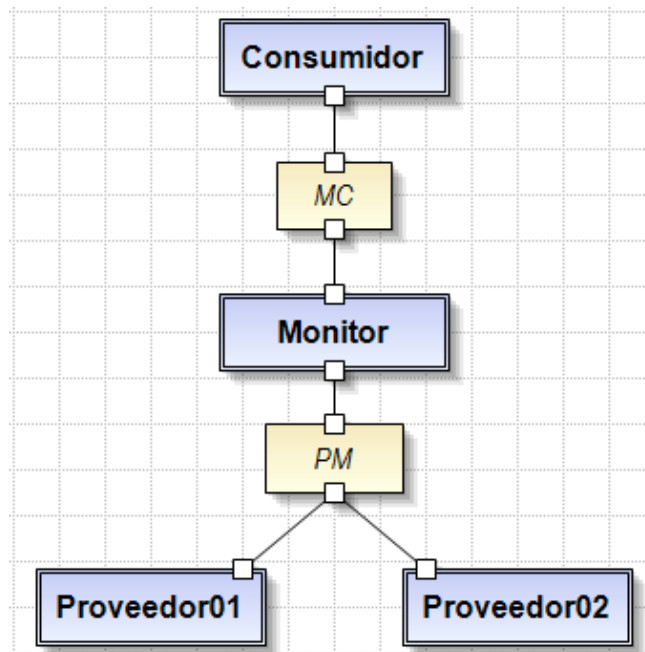


Figura 11.11 Representación gráfica del sistema de monitoreo con ArchStudio.

En el código 11.17 se presenta la representación textual del componente “Monitor” basada en los esquemas de xADL.

```

<structure_3_0:structure structure_3_0: id="s01" structure_3_0: name="Sistema">
  <structure_3_0:component structure_3_0: id="m01"
    structure_3_0: name="Monitor">
    <structure_3_0:interface structure_3_0: direction="none"
      structure_3_0: id="iMonitor01"
      structure_3_0: name="iM01">
      <structure_3_0:ext xsi:type="hints_3_0:HintsExtension">
        <hints_3_0:hint
          hints_3_0: hint="java.awt.geom.Point2D: 1, 46446185907025
            34656, 4640255728563519488"
          hints_3_0: name="locacion"/>
        </structure_3_0:ext>
      </structure_3_0:interface>
      <structure_3_0:interface structure_3_0: direction="none"
        structure_3_0: id="iMonitor02"
        structure_3_0: name="iM02">
        <structure_3_0:ext xsi:type="hints_3_0:HintsExtension">
          <hints_3_0:hint
            hints_3_0: hint="java.awt.geom.Point2D: 1, 46446185907025
              34656, 4641663103447072768"
            hints_3_0: name="locacion"/>
          </structure_3_0:ext>
        </structure_3_0:interface>
      </structure_3_0:ext xsi:type="hints_3_0:HintsExtension">

```

```
<hints_3_0: hint
    hints_3_0: hint="org.eclipse.swt.graphics.Rectangle: 276, 172, 12
    0, 40"
    hints_3_0: name="bounds"/>
<hints_3_0: hint
    hints_3_0: hint="org.eclipse.swt.graphics.RGB: 197, 203, 245"
    hints_3_0: name="color"/>
</structure_3_0: ext>
</structure_3_0: component>
</structure_3_0: structure>
```

Código 11.17 Ejemplo de la definición de un componente con xADL.

11.5. Lenguajes de Descripción de Arquitectura y tecnologías de vanguardia

Los Lenguajes de Descripción de Arquitectura surgieron en los años 90, sin embargo, la tecnología ha cambiado desde entonces, por lo que se podría cuestionar si tales lenguajes pueden soportar el modelado y la especificación de arquitecturas que involucren tecnologías actuales, tales como cómputo en la nube (*cloud computing*), *big data*, cómputo móvil o sistemas orientados a servicios.

En principio, muchos Lenguajes de Descripción de Arquitectura se desarrollaron considerando tecnologías o estilos arquitectónicos de cierto momento, sin embargo, muchos de ellos han evolucionado con el tiempo para seguir estando vigentes, algunos otros han llegado a ser obsoletos. Por otra parte, muchos de los conceptos y principios de Arquitectura de Software a partir de los cuales se desarrollaron algunos lenguajes siguen estando vigentes.

En las siguientes secciones se presenta un panorama general del uso de Lenguajes de Descripción de Arquitectura para modelar arquitecturas de software con tecnologías actuales.

11.5.1. Arquitecturas Empresariales

Una arquitectura empresarial es el conjunto de componentes, procesos y políticas de una empresa. Las arquitecturas empresariales se relacionan con las arquitecturas de software en cuanto a que las primeras pueden incluir una solución de arquitectura de TI que permite describir, estructurar y optimizar la arquitectura de los sistemas de información, lo cual permite a la organización obtener un conocimiento global sobre sus activos de TI, establecer principios eficaces de gobierno de TI, y desarrollar una arquitectura de TI específica.

Muchas organizaciones de tecnología, como IBM y Oracle, ofrecen servicios de Arquitectura Empresarial, por lo que resulta útil contar con notaciones que permitan modelar soluciones de dicha índole. Una de las herramientas creadas para tal fin es ArchiMate®, un estándar que proporciona un lenguaje gráfico para la representación de arquitecturas empresariales, así como su motivación y razonamiento. Permite realizar la descripción, construcción y operación de procesos de negocio, estructuras organizacionales, flujos de información, sistemas de TI, e infraestructuras técnicas.

En la figura 11.12 se presenta un ejemplo de una arquitectura empresarial modelada con ArchiMate®, en el que se puede ver la integración de una capa de tecnología, una de aplicación y una de negocios.

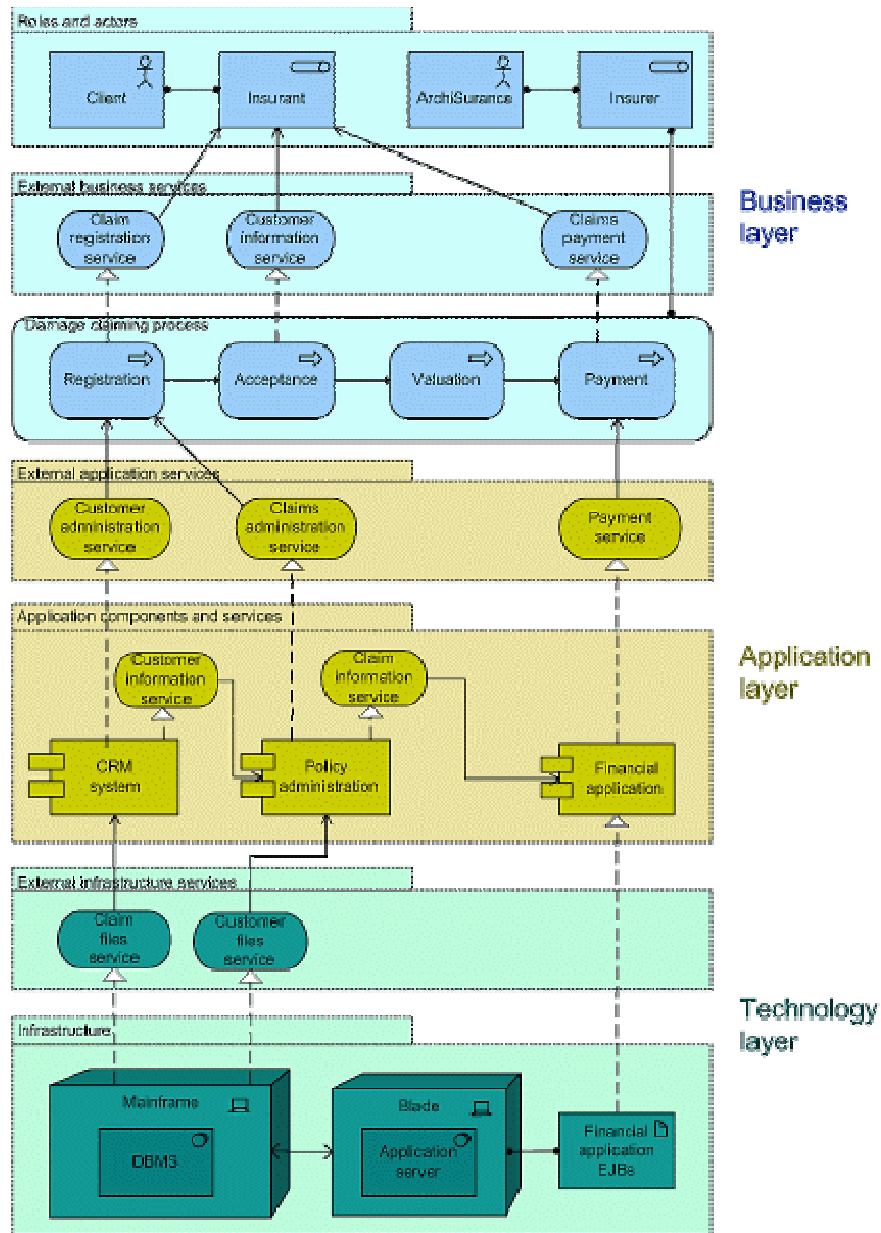


Figura 11.12 Ejemplo de una arquitectura empresarial con ArchiMate®.

11.5.2. Arquitecturas Orientadas a Servicios

La Arquitectura Orientada a Servicios (*Service Oriented Architecture* o SOA) es un paradigma de arquitectura para sistemas distribuidos, que facilita la integración de sistemas, la alineación de los procesos de negocio, la innovación de servicios y la adaptación ágil ante cambios. Además, permite crear sistemas altamente escalables que reflejan el negocio de la organización.

Una arquitectura orientada a servicios permite controlar un problema de forma general, facilita la independencia y reutilización de plataformas y tecnologías, facilita la toma de

decisiones, y permite la reducción de costos y la disminución de riesgos al migrar componentes tecnológicos.

Debido a sus ventajas, muchas empresas de tecnología implementan arquitecturas orientadas a servicios para ofrecer soluciones, servicios o productos, tal es el caso de IBM, Oracle y Accenture.

Debido a que una arquitectura orientada a servicios cuenta con servicios (principales elementos arquitectónicos), los cuales tienen interfaces bien definidas, pueden ser modeladas con algunos Lenguajes de Descripción de Arquitectura. En particular, SOADL (Jia et al. 2007) es un Lenguaje de Descripción de Arquitectura que permite especificar las interfaces, comportamiento, semánticas y propiedades de calidad de servicios, además proporciona un mecanismo para modelar y analizar arquitecturas dinámicas, y soporta la composición de servicios. Otros lenguajes como AADL (Feiler et al. 2006), permiten especificar sistemas con arquitecturas orientadas a servicios, tales como los sistemas distribuidos.

11.5.3. Cómputo en la nube

El cómputo en la nube o "*Cloud Computing*", es un paradigma que permite ofrecer servicios de cómputo a través de una red, que usualmente es Internet. Algunas de sus características son la agilidad para proporcionar recursos tecnológicos, la reducción de costos, la escalabilidad y elasticidad, y la independencia de dispositivos y de ubicación.

Existen distintos modelos de servicios en la nube, entre los que se encuentran el Software como Servicio (SaaS o *Software as a Service*), Plataforma como Servicio (PaaS o *Platform as a Service*), Infraestructura como servicio (IaaS o *Infrastructure as a Service*) y el Desarrollo como Servicio (DaaS o *Development as a Service*).

Algunos ejemplos de servicios proporcionados a través de la nube son Dropbox, Google drive, iCloud y Microsoft Azure. Además, muchas organizaciones como IBM y Oracle ofrecen servicios en la nube.

Algunos Lenguajes de Descripción de Arquitectura como AADL (Feiler et al. 2006) permiten especificar arquitecturas de sistemas que pueden operar en la nube, tales como los sistemas distribuidos.

11.5.4. Almacenes de datos

Un almacén de datos o *data warehouse* es una colección de datos orientado a un ámbito en particular (empresa u organización), es integrado, no volátil y variable en el tiempo, y ayuda en la toma de decisiones. Un almacén de datos se compone de datos; medios para extraerlos, transformarlos y cargarlos; técnicas para analizarlos y generar información; y formas de gestión de datos.

Algunas empresas de tecnología como IBM, Oracle y Microsoft ofrecen soluciones para el almacenamiento, administración y mejora en el uso de datos a través de software,

dispositivos, herramientas y modelos de almacenamiento de datos. Algunos ejemplos son bases de datos, modelos de datos, conectores para integrar bases de datos con otras tecnologías, y SQL para *big data*.

El proceso para generar un almacén de datos incluye tareas de estructuración y modelado de datos. Aunque en particular, para el modelado de datos se utilizan notaciones especiales para dicha tarea, algunos Lenguajes de Descripción de Arquitectura como AADL (Feiler et al. 2006) permiten modelar aspectos de datos y las plataformas físicas que pueden contenerlos.

11.5.5. Sistemas móviles y sistemas distribuidos

En la actualidad existe una tendencia en el desarrollo de aplicaciones o sistemas distribuidos que operen en dispositivos móviles.

En los sistemas distribuidos es importante considerar la localización y la transparencia, ya que un sistema distribuido debe proporcionar las mismas funciones como si fuera un sistema centralizado, lo que implica que los clientes no están al tanto de dónde se localizan los servicios, que los recursos son accesibles aún si están distribuidos, y que los elementos cuentan con movilidad.

Una arquitectura de software puede contar con componentes y conectores, los componentes proporcionan una funcionalidad y los conectores los coordinan, por lo que se puede separar la funcionalidad de la coordinación en una arquitectura. Esto es importante para definir elementos con comunicación distribuida y movilidad.

Aunque existen muchos Lenguajes de Descripción de Arquitectura, no todos proporcionan primitivas para describir arquitecturas de software de sistemas distribuidos y móviles (Ali, Solís y Ramos 2008). Algunos de los lenguajes que lo permiten son: Darwin (Magee et al. 1995), C2Sadel (Medvidovic y Rakic 2001), Community (Lopes, Fiadeiro y Wermelinger 2002), MobiS (Ciancarini y Mascolo 1998), LAM Model (Xu, Yin y Deng 2003), pi-ADL (Oquendo 2004), Con-Moto (Gruhn y Schafer 2004) y Ambient-PRISMA (Ali, Millan y Ramos 2006). Se puede consultar el trabajo de Ali, Solís y Ramos (2008), el cual proporciona una comparación detallada de los Lenguajes de Descripción de Arquitectura que pueden ser usados para modelar sistemas de software móviles.

11.5.6. Sistemas en tiempo real y sistemas embebidos

Los sistemas en tiempo real son sistemas que interactúan con su ambiente físico y responden a los estímulos del entorno dentro de un plazo de tiempo determinado.

Uno de los lenguajes presentados en este trabajo (sección 11.3) es AADL (Feiler et al. 2006), el cual permite especificar y analizar sistemas embebidos de tiempo real.

11.6. Conclusiones sobre el uso de los LDAs

11.6.1. Sobre las ventajas de los LDAs

Una de las razones por las que se incluyó en este trabajo el uso de Lenguajes de Descripción de Arquitectura (LDAs), es debido a las ventajas que proporcionan en el proceso de arquitectura, y en particular, en las actividades de documentación arquitectónica. Las ventajas del uso de dichos lenguajes, se pueden apreciar al compararlos con notaciones menos formales, o con notaciones de modelado general como UML.

En primera instancia, es importante mencionar que una de las características de una arquitectura de software, es que incluye el comportamiento del sistema al que pertenece, por lo que poder modelar o especificar dicha característica es una cuestión importante. Algunos de los Lenguajes de Descripción de Arquitectura que se presentan en este trabajo (como Wright), permiten especificar el comportamiento de un sistema en términos de eventos y procesos, e integrar dicho comportamiento con la especificación estructural. Poder especificar el comportamiento es una ventaja sobre otras notaciones menos formales que no permiten su especificación o modelado, o lo soportan de una forma menos formal, como en el caso de UML, que sólo permite modelarlo a través de diagramas de propósito general, como con diagramas de actividades, y cuyas notaciones no permiten una especificación más precisa, ni permiten integrarlos con una especificación estructural.

Otra característica que es importante en una arquitectura, es que debe incluir las relaciones o interacciones entre los elementos que la componen, por lo que especificar y modelar dichas interacciones, es algo fundamental. Dentro de los Lenguajes de Descripción de Arquitectura presentados, Acme, Wright y AADL permiten especificar dichas interacciones, los primeros dos en forma de conectores, y AADL explícitamente como parte de los componentes. Especificar las interacciones de manera precisa, es una ventaja de los LDAs sobre otras notaciones menos formales, que únicamente permiten modelar interacciones como líneas y no cuentan con mayor precisión. Por otra parte, los LDAs permiten especificar características propias de las interacciones entre elementos arquitectónicos, tales como interfaces de conexión (como puertos), o la forma en que interactúan dos elementos. Por ejemplo, AADL permite especificar puertos con características importantes, como su dirección (que define si son puertos de entrada o salida), o definir si son puertos de datos o de eventos.

Una de las razones de la importancia de la arquitectura de un sistema, es debido a que puede ser creada como un modelo transferible y reusable que forme el corazón de una línea de productos (Bass, Clements y Kazman, 2012). Los Lenguajes de Descripción de Arquitectura, como Acme, xADL y Wright, permiten definir familias de sistemas o estilos arquitectónicos, los cuales pueden definir los lineamientos para construir cierto tipo de sistemas con características en común. En el caso de Acme y Wright, se pueden definir estilos arquitectónicos definiendo un conjunto de tipos de componentes y conectores, y especificando restricciones para los sistemas que serán parte de un estilo arquitectónico. En ambos lenguajes (Acme y Wright), se puede especificar las restricciones de los estilos arquitectónicos mediante notaciones basadas en la lógica de predicados de primer orden, lo que representa una forma precisa de definir dichas restricciones. Lo anterior es una ventaja sobre notaciones menos formales o de propósito general.

En relación con los estilos arquitectónicos, algunos LDAs como Acme, Wright y AADL, permiten especificar tipos de elementos (como componentes y conectores) que posteriormente pueden ser instanciados e incluidos como parte de una arquitectura de software. Esto es una ventaja sobre otras notaciones que incluyen sólo ciertos tipos predefinidos de elementos que no pueden incluir características adicionales.

Algunos LDAs como Acme, AADL y xADL, cuentan con notaciones tanto gráficas como textuales para realizar modelos y especificaciones arquitectónicas, lo que permite generar automáticamente, mediante ciertas herramientas, una representación gráfica a partir de una especificación textual o viceversa. Lo anterior es una ventaja sobre notaciones únicamente gráficas y menos formales que no permiten generar modelos a partir de especificaciones textuales o viceversa. Otra ventaja de que los LDAs cuenten con distintas notaciones, es que facilitan que los modelos y especificaciones elaboradas con ellos puedan ser entendidas por distintos interesados con diferentes grados de conocimiento arquitectónico, y además, las especificaciones textuales pueden ser entrada para diversas herramientas computacionales. Adicionalmente, se debe destacar la similitud entre la notación gráfica de AADL con UML, lo que permite comprender más fácilmente la notación gráfica de dicho LDA si se ha usado UML con anterioridad.

Otra de las ventajas de los LDAs con respecto a notaciones menos formales o de propósito general, es que permiten realizar ciertos tipos de análisis de características arquitectónicas. Por ejemplo, con la herramienta *AcmeStudio* de Acme, se pueden realizar análisis de la eficiencia de desempeño (*performance*) y correctitud. Por otra parte, AADL cuenta con características y extensiones para analizar aspectos como latencia, consistencia y seguridad.

Algunos LDAs como AADL, Wright, Acme y xADL, permiten definir arquitecturas de manera jerárquica, permitiendo definir desde el nivel de abstracción más alto, hasta el más bajo que se requiera, a través de distintos tipos de elementos arquitectónicos. Dicho modelado y especificación jerárquica puede realizarse de una forma más precisa utilizando LDA, ya que por ejemplo, se pueden especificar las relaciones entre los subcomponentes del sistema y el sistema como tal, permitiendo definir con mayor claridad la estructura interna de un componente o sistema.

Por último, es importante resaltar las facilidades que proporcionan los LDAs para representar aspectos específicos y propios de Arquitectura de Software, tales como atributos de calidad y comportamiento, y la manera en que permiten especificarlos de una manera no ambigua.

11.6.2. Sobre las desventajas de los LDAs

Si bien los Lenguajes de Descripción de Arquitectura tienen ventajas sobre notaciones menos formales o de propósito general, también su uso presenta ciertas desventajas.

Una de las desventajas más notables de los LDAs, es que sus notaciones textuales pueden llegar a ser más complejas que las notaciones de UML o notaciones de modelado gráfico, por lo que su aprendizaje, uso, e interpretación, pueden resultar complicados, sobre todo para aquellos que tienen bajo dominio en temas arquitectónicos o computacionales. Por ejemplo, Wright cuenta con una notación propia para especificar arquitecturas estructuralmente, integra la notación CSP para especificar comportamiento,

y emplea una notación basada en lógica de predicados de primer orden para especificar restricciones en estilos arquitectónicos y arquitecturas de sistemas, lo cual puede resultar complejo para ciertas personas. La buena noticia es que muchos LDAs incluyen notaciones gráficas que pueden compensar el problema de las notaciones textuales complejas.

Por otra parte, la complejidad y tamaño de las notaciones de los LDAs, puede implicar que se requiera mayor tiempo para elaborar un modelo o especificación arquitectónica de manera textual, como en el caso de AADL, que es un lenguaje extenso que requiere cierto tiempo para realizar una especificación arquitectónica.

Otra desventaja de usar LDAs, es que muchos de ellos siguen en proceso de crecimiento y refinamiento, por lo que muchas de sus características aún están en proceso de ser implementadas. Por ejemplo, Acme es un LDA de intercambio que incluye propiedades para incluir características de otros LDAs, sin embargo, aún se requiere implementar herramientas que validen la sintaxis de dichas propiedades con características de otros LDAs.

11.6.3. Sobre las similitudes y diferencias entre los LDAs

Los Lenguajes de Descripción de Arquitectura incluidos en este trabajo, cuentan con algunas características en común derivadas de los conceptos arquitectónicos a partir de los cuales fueron definidos.

En el caso de Acme y Wright, ambos lenguajes comparten una ontología de elementos similar, ya que ambos cuentan con las facilidades para especificar componentes, conectores, roles, puertos, estilos arquitectónicos, restricciones, y representaciones jerárquicas.

Por otra parte, los cuatro LDAs presentados (Acme, Wright, AADL y xADL) incorporan facilidades para utilizar, especificar o modelar distintos tipos de elementos, es decir, incorporan un sistema de tipos de elementos arquitectónicos.

Una de las razones de las características en común entre algunos LDAs, es debido a que algunos LDAs han sido derivados, inspirados o basados en otros. Tal es el caso de Acme, que incluye un núcleo conceptual que fue derivado de distintos LDAs (Unicon, Aesop, etc.) con el objetivo de ser un LDA de intercambio.

En cuanto a las diferencias entre los LDAs presentados en este trabajo, se encuentran los propósitos para los que fueron destinados. Por ejemplo, mientras que Acme y Wright permiten especificar arquitecturas de sistemas en general basados componentes y conectores, AADL se enfoca en sistemas distribuidos, complejos, y con características especiales de eficiencia de desempeño (*performance*).

Por otra parte, es importante mencionar que aunque algunos LDAs cuentan con mayor formalidad que otras notaciones, los LDAs también varían entre sí en cuanto a la formalidad de sus especificaciones. Por ejemplo, la especificación del núcleo de xADL es más un meta-modelo que una definición formal de lenguaje, en contraste, AADL está definido con mayor formalidad.

Es también importante mencionar que los LDAs presentados, varían en cuanto a sus características de extensibilidad. Mientras que xADL está destinado completamente a la flexibilidad y extensibilidad, otros lenguajes como Wright o AADL, proporcionan características más limitadas o nulas en cuanto a extensibilidad.

11.6.4. Sobre las herramientas de los LDAs

En general, las herramientas de los distintos LDAs están implementadas de acuerdo a las especificaciones de los lenguajes que soportan. Sin embargo, existen algunas herramientas que aún requieren implementar características fundamentales. Tal es el caso de *AcmeStudio*, que aunque soporta adecuadamente el lenguaje Acme, no tiene soporte para validar la sintaxis de otros LDAs que, en teoría, pueden ser incluidos como parte de una especificación en Acme.

Es importante mencionar que muchas de las herramientas de los LDAs son propietarias, es decir, fueron desarrolladas únicamente para soportar las características de un LDA en particular. En el caso de xADL, que está basado en XML, se pueden usar herramientas convencionales que soportan XML para escribir un modelo en xADL, sin embargo, existe una herramienta específica para el análisis y modelado con xADL (*ArchStudio*).

Es importante mencionar que algunas herramientas que soportan cierto LDA, están disponibles a la comunidad para ser extendidas y mejoradas. Tal es en el caso de *ArchStudio*, que puede ser modificado por cualquier persona ya que es utilizado para desarrollarse a sí mismo.

Es importante resaltar que la mayoría de los lenguajes presentados (Acme, AADL y xADL), a excepción de Wright, cuentan con herramientas que incluyen facilidades para modelado gráfico, especificación, y análisis, y además son multiplataforma al estar basadas en Eclipse.

En general, las herramientas con que cuentan los LDAs más actuales son fáciles de usar, y existen tutoriales y documentación suficiente para su uso.

11.6.5. Sobre las tendencias actuales de los LDAs

Las tendencias actuales de los Lenguajes de Descripción de Arquitectura son la incorporación de características para modelar y especificar atributos de calidad, análisis de características arquitectónicas, extensibilidad, y separación de intereses. Un ejemplo de esto último, es la facilidad que proporciona xADL para separar los elementos de tiempo de ejecución y de tiempo de diseño en una arquitectura.

Cabe resaltar que existen características innovadoras en algunos LDAs, por ejemplo, xADL, incorpora facilidades para especificar características sobre administración de la configuración.

11.6.6. Sobre el futuro y desafíos de los LDAs

Aún existe mucho trabajo en cuanto al desarrollo de Lenguajes de Descripción de Arquitectura, ya que hace falta extender sus capacidades de análisis, extensibilidad, y

comportamiento, características que pueden ser cruciales para especificar la arquitectura de un sistema.

Por otra parte, hace falta trabajar en las características semánticas de los LDAs, ya que por ejemplo, en algunos casos (como Acme) la definición sintáctica está bien definida, pero no su semántica. En el caso de Acme, esto último se debe a que contiene un *framework* de semántica abierta para la realización de razonamiento automatizado, sin embargo esto puede ser una desventaja para dicho LDA.

Debido a que la consistencia es una propiedad necesaria para una descripción de arquitectura de calidad, es necesario que las herramientas que soportan los diferentes LDAs incluyan análisis de consistencia.

Por otra parte, cada vez se vuelve más necesario que los LDAs cuenten con herramientas que permitan elaborar modelos gráficos, realizar distintos tipos de análisis de atributos de calidad (que están presentes en la mayoría de los sistemas), y cuyo uso sea sencillo.

Capítulo 12. Documentación de relaciones arquitectónicas

El estándar ISO/IEC/IEEE 42010:2011 establece que una descripción de arquitectura debe incluir información sobre las relaciones que existen entre sus elementos. Dichas relaciones deben ser documentadas mediante:

- Un análisis de consistencia de las vistas y modelos arquitectónicos.
- Las correspondencias entre los elementos de la descripción de arquitectura.
- Las reglas de correspondencia aplicables a los elementos de la descripción de arquitectura.

Los elementos de una descripción de arquitectura más primitivos son los interesados, intereses, puntos de vista, vistas, tipos de modelo, modelos arquitectónicos, y decisiones y razonamiento de la arquitectura. Sin embargo, cuando los puntos de vista y los tipos de modelo son definidos (ver capítulo 9), y sus modelos son construidos, se generan nuevos elementos de descripción de arquitectura cuyas relaciones también deben ser documentadas (ISO/IEC/IEEE 42010:2011).

En las siguientes secciones (12.1 a 12.3) se presenta una propuesta y consejos para documentar las relaciones entre los elementos de una descripción de arquitectura. Además, es importante mencionar que en el apéndice “D” se presenta una plantilla (plantilla 1) para documentar descripciones de arquitectura, que cuenta con un apartado (sección 6) para documentar tales relaciones arquitectónicas, y que es compatible con la forma en que se propone en este capítulo.

12.1. Documentación del análisis de consistencia

En las descripciones de arquitectura, una de las consecuencias del empleo de múltiples vistas es la necesidad de expresar y mantener la consistencia entre dichas vistas.

Una descripción de arquitectura debe incluir un análisis de consistencia de los modelos y las vistas arquitectónicas que debe contener el registro de cualquier inconsistencia conocida a través de dichos modelos y vistas.

Las descripciones de arquitectura consistentes siempre son preferibles, sin embargo, a veces es imposible o impráctico resolver todas las inconsistencias.

En la sección 10.2 de este trabajo se presentan algunas consideraciones para el análisis de vistas y modelos arquitectónicos, incluyendo el análisis de consistencia.

Para documentar las inconsistencias encontradas en los modelos y vistas arquitectónicas se propone incluir lo siguiente:

- Los modelos o vistas donde fueron encontradas las inconsistencias.
- El tipo de inconsistencia encontrada (correspondencia, refinamiento, de requerimientos no funcionales).
- Posible solución a las inconsistencias encontradas y razones para su solución o no solución.

12.2. Documentación de reglas de correspondencia

Una descripción de arquitectura debe incluir cada regla de correspondencia aplicada a ella. Una regla de correspondencia expresa una restricción para ser impuesta en correspondencias.

Las reglas de correspondencia pueden ser originadas en la descripción de arquitectura, en un punto de vista, o en un Lenguaje de Descripción de Arquitectura. Para cada regla de correspondencia identificada, se debe registrar lo siguiente (ISO/IEC/IEEE 42010:2011):

- Si la regla de correspondencia se cumple o no. Una regla se cumple si se demuestra que la correspondencia asociada satisface la regla, y es violada si se demuestra que la correspondencia asociada no satisface la regla o cuando la correspondencia asociada no existe.
- Todas las violaciones a la regla, si es que existen.

12.2.1. Reglas de correspondencia comunes y ejemplos

A continuación se presentan algunas de las reglas de correspondencia más comunes entre los elementos de una descripción de arquitectura.

Interesados - intereses

Por lo general, la relación entre intereses e interesados es de muchos a muchos, es decir, cada interesado puede tener uno más intereses y cada interés puede pertenecer a más de un interesado (ISO/IEC/IEEE 42010:2011). De esto se derivan las siguientes reglas:

- Cada interesado debe tener uno o más intereses.
- Cada interés debe pertenecer a uno o más interesados.

Las correspondencias relacionadas a estas reglas de correspondencia debieron haber sido generadas como parte de la documentación de intereses e interesados (sección 7.3).

Interesado - interesado

Se pueden establecer reglas de correspondencia entre interesados. Por ejemplo, se puede definir la siguiente: “cada responsable de pruebas debe tener asociado uno o más usuarios que los ayuden a validar los requerimientos del sistema”.

Interés - interés

Se pueden establecer reglas de correspondencia entre intereses. Por ejemplo, se podría definir una regla como la siguiente: “Cada atributo de calidad se deriva de una meta de negocio”.

Intereses - puntos de vista

Un punto de vista debe enmarcar uno o más intereses, y un interés puede ser enmarcado por uno o más puntos de vista. Esto significa que la relación entre ambos elementos es de muchos a muchos (ISO/IEC/IEEE 42010:2011).

Las reglas de correspondencia que se pueden derivar de lo anterior son:

- Cada punto de vista debe enmarcar uno o más intereses.
- Cada interés debe ser enmarcado por uno o más puntos de vista.

Las correspondencias relacionadas a estas reglas de correspondencia debieron haber sido generadas como parte de la documentación de los puntos de vista (ver sección 9.2.3).

Punto de vista - punto de vista

Se pueden establecer reglas de correspondencia entre puntos de vista arquitectónicos dependiendo de las características de la arquitectura del sistema en cuestión. Un ejemplo de una regla de correspondencia es el siguiente: “Un punto de vista computacional debe ser un refinamiento de un punto de vista empresarial, donde las acciones en la vista computacional son las acciones detalladas de la vista empresarial”.

Punto de vista - vista

La relación entre puntos de vista y vistas de 1 a 1, es decir, cada vista arquitectónica es gobernada por un único punto de vista y cada punto de vista gobierna solo a una vista en una descripción de arquitectura (ISO/IEC/IEEE 42010:2011).

La información para establecer las correspondencias basadas en esta regla, pudo haber sido identificada al realizar la definición de puntos de vista y vistas arquitectónicas (ver secciones 9.1 y 10.1).

Vista - vista

Se pueden establecer reglas de correspondencia entre vistas arquitectónicas en una descripción de arquitectura. Por ejemplo, se puede definir la siguiente: “una vista computacional debe ser un refinamiento de una vista empresarial”.

Las relaciones entre un conjunto de vistas arquitectónicas debidamente definidas, pueden ayudar a manejar la inconsistencia en una descripción de arquitectura.

Vistas - intereses

El estándar ISO/IEC/IEEE 42010:2011 establece que una vista arquitectónica debe abordar uno o más intereses arquitectónicos, y que un interés arquitectónico puede ser abordado por una o más vistas arquitectónicas. Esto resulta en una relación de muchos a muchos.

Las reglas de correspondencia que se pueden derivar de lo anterior son:

- Cada vista arquitectónica debe abordar uno o más intereses.
- Cada interés debe ser abordado por una o más vistas arquitectónicas.

Las correspondencias relacionadas a estas reglas de correspondencia debieron haber sido identificadas como parte del desarrollo de las vistas arquitectónicas (sección 10.1).

Punto de vista - tipos de modelo

El estándar ISO/IEC/IEEE 42010:2011 establece que un punto de vista debe incluir uno o más tipos de modelo.

La información requerida para establecer las correspondencias basadas en esta regla, pudo haber sido identificada al realizar la definición de puntos de vista arquitectónicos (ver sección 9.2.5).

Tipo de modelo - tipo de modelo

Cuando se utilizan distintos tipos de modelos para crear los modelos arquitectónicos, es útil establecer reglas de correspondencia entre los lenguajes, notaciones, restricciones o convenciones entre los tipos de modelo. Por ejemplo, si se utiliza el lenguaje ACME en

conjunto con AADL, se requerirá establecer reglas de correspondencia entre los elementos que soporta cada lenguaje.

Tipo de modelo - modelos

El estándar ISO/IEC/IEEE 42010:2011 establece que un modelo debe ser gobernado por un tipo de modelo, y que un tipo de modelo puede gobernar uno o más modelos.

Las correspondencias relacionadas a esta regla de correspondencia debieron haber sido identificadas como parte de la definición de puntos de vista (9.2.5) y el desarrollo de las vistas arquitectónicas (ver sección 10.1).

Modelo - modelo

Se pueden incluir las relaciones entre modelos arquitectónicos, ya sea que sean parte de la misma vista arquitectónica o no.

De la misma manera se pueden definir reglas de correspondencia entre los elementos particulares de un modelo arquitectónico. Un ejemplo es la siguiente regla: “cada proceso debe estar asignado a un procesador específico”.

Vistas – modelos

El estándar ISO/IEC/IEEE 42010:2011 establece que una vista debe incluir uno o más modelos arquitectónicos que expresen todos los intereses enmarcados por el punto de vista que gobierna a la vista arquitectónica.

Las correspondencias relacionadas a esta regla de correspondencia debieron haber sido identificadas como parte del desarrollo de las vistas arquitectónicas (ver sección 10.1).

Reglas de correspondencia en los Lenguajes de Descripción de Arquitectura

Los Lenguajes de Descripción de Arquitectura incluyen reglas de correspondencia o restricciones que se deben seguir para realizar un modelo o especificación de la arquitectura. Estas reglas de correspondencia pueden ser documentadas si se considera necesario.

Un ejemplo de una regla de correspondencia entre elementos del lenguaje AADL es la siguiente: “Un componente del tipo memoria únicamente puede ser contenido en elementos del tipo memoria, procesador, o sistema”.

12.3. Documentación de correspondencias

Una correspondencia define una relación entre elementos de una descripción de arquitectura (interesados, intereses, puntos de vista, vistas, tipos de modelos, modelos arquitectónicos, y decisiones y razonamiento de la arquitectura), sin embargo, cuando los puntos de vista y los tipos de modelo son definidos, se generan nuevos elementos de descripción de arquitectura. Las correspondencias pueden ser usadas para expresar

relaciones de la arquitectura en una descripción de arquitectura (o entre descripciones de arquitectura) (ISO/IEC/IEEE 42010:2011). Por otra parte, una correspondencia puede ser definida entre elementos de la descripción de arquitectura y entre ella misma.

Matemáticamente, una correspondencia es una relación n -aria, y una regla de correspondencia es una definición intencional de una relación n -aria. Las relaciones tienen propiedades útiles que facilitan la composición y el razonamiento, y permiten la representación y manipulación eficiente. Muchas reglas de correspondencia son binarias, pero esto no es necesario, una correspondencia puede relacionar cualquier número arbitrario de elementos.

Las correspondencias pueden (no es necesario) ser gobernadas por reglas de correspondencia, y tanto las correspondencias como las reglas de correspondencia son usadas para expresar, registrar y hacer cumplir relaciones de composición, refinamiento, consistencia, trazabilidad, dependencia, restricción y obligación entre modelos y vistas arquitectónicas.

En una descripción de arquitectura, las correspondencias deben ser documentadas incluyendo (ISO/IEC/IEEE 42010:2011):

- La identificación de cada correspondencia, es decir, un nombre o clave único que permita identificarla del resto de las correspondencias.
- La identificación de los elementos participantes en cada correspondencia, es decir, los interesados, intereses, vistas, puntos de vista, tipos de modelo, modelos arquitectónicos, y decisiones y razonamiento arquitectónico que son parte de la correspondencia. Los elementos que forman parte de una correspondencia no necesariamente deben ser distintos.
- La identificación de la reglas de correspondencia que gobierna cada correspondencia, si es el caso.

Cuando las vistas de la descripción de arquitectura son heterogéneas, es decir, que cada vista cuenta con modelos que utilizan distintos lenguajes de modelado, es útil declarar correspondencias entre los modelos en diferentes lenguajes y no sólo entre las vistas.

Algunas correspondencias pueden ser expresadas en términos de los elementos que forman parte de los modelos arquitectónicos, aunque esto no es requerido.

12.3.1. Ejemplo de una correspondencia gobernada por una regla de correspondencia

Si una regla de correspondencia ha sido definida, ésta debe tener asociadas correspondencias con información específica de las relaciones entre elementos de una descripción de arquitectura. A continuación se presenta un ejemplo de una regla de correspondencia y correspondencias entre elementos de un modelo arquitectónico:

Sea la regla de correspondencia **R1**: “Un proceso debe tener asignado uno o más hilos de ejecución”.

Sean los procesos p_1 , p_2 , p_3 y p_4 y sean los hilos h_1 , h_2 , h_3 , ..., h_{10} .

Un ejemplo de una correspondencia asociada a la regla de correspondencia **R1**, y considerando los procesos e hilos mencionados, es se presenta en la tabla 12.1.

Proceso (TIENE ASIGNADOS) hilos de procesamiento	
p1	h1, h2
p2	h3, h4, h5, h6, h7
p3	h8, h9, h10

Tabla 12.1 Ejemplo de una correspondencia.

La correspondencia presentada identifica la regla de correspondencia que la gobierna (R1) y los elementos participantes en la correspondencia (procesos e hilos), y es una correspondencia que se puede identificar de las demás (“TIENE ASIGNADOS”).

Puede observarse que la correspondencia del ejemplo viola la regla de correspondencia ya que el proceso p4 no tiene asignado ningún hilo de ejecución.

Capítulo 13. Documentación de razonamiento y decisiones arquitectónicas

Un requisito de una descripción de arquitectura según el estándar ISO/IEC/IEEE 42010:2011, es registrar el razonamiento de las decisiones de arquitectura hechas. Dichas decisiones permiten reusar conocimiento y entender mejor la arquitectura de un sistema.

En las siguientes secciones (13.1 y 13.2) se presentan las consideraciones para documentar tanto el razonamiento arquitectónico, como las decisiones tomadas en el proceso de arquitectura. Además, es importante mencionar que en el apéndice “D” se presenta una plantilla (plantilla 1) para documentar descripciones de arquitectura, que cuenta con un apartado (sección 7) para documentar tal razonamiento y decisiones arquitectónicas, y que es compatible con la forma en que se propone en este capítulo.

13.1. Documentación de razonamiento arquitectónico

El estándar ISO/IEC/IEEE 42010:2011 establece que como parte del razonamiento que surge del proceso de arquitectura, se debe registrar en una descripción de arquitectura:

- El razonamiento para cada punto de vista incluido y usado en términos de sus interesados, intereses, tipos de modelo, notaciones y métodos.
- El razonamiento para cada decisión considerada como una decisión de arquitectura “clave”.
- La evidencia de las consideraciones de alternativas, y su razonamiento para las decisiones hechas.

A continuación se presentan las consideraciones para documentar el razonamiento arquitectónico en una descripción de arquitectura de acuerdo a lo establecido por el estándar ISO/IEC/IEEE 42010:2011.

13.1.1. Razonamiento sobre puntos de vista

En el capítulo 9 de este trabajo, se presenta una guía para la definición, selección y documentación de puntos de vista arquitectónicos de un sistema. El razonamiento que surge de la realización de esas actividades debe ser documentado, para esto, se propone:

- Incluir el razonamiento relacionado a la identificación y selección de los interesados clave, que condujeron a la definición y/o selección de los puntos de vista arquitectónicos.
- Incluir el razonamiento relacionado a la identificación y selección de los intereses clave, que condujeron a la definición y/o selección de los puntos de vista. También se debe pensar en el razonamiento relacionado a la forma en que los intereses que fueron enmarcados por cada punto de vista arquitectónico.
- Incluir el razonamiento relacionado a la identificación y selección de los tipos de modelo que influyeron en la selección y/o definición de los puntos de vista. También se recomienda incluir el razonamiento relacionado con la forma en que

los tipos de modelos fueron seleccionados o definidos en cada uno de los puntos de vista.

Debido a que los tipos de modelo utilizados en este trabajo son los Lenguajes de Descripción de Arquitectura, se debe registrar el razonamiento que se siguió para la selección de los mismos.

- Si es el caso, incluir el razonamiento relacionado con la selección de los métodos incluidos en los puntos de vista.

13.1.2. Razonamiento sobre decisiones de arquitectura clave

El proceso de arquitectura incluye diversas actividades como la definición, especificación, documentación, evaluación y mantenimiento, por lo que se debe registrar el razonamiento relacionado a las decisiones clave hechas en cada una de estas actividades. Para lo anterior se propone:

- Incluir el razonamiento relacionado a la identificación de los interesados clave en la arquitectura, así como el razonamiento del uso de métodos o técnicas para identificarlos.
- Incluir el razonamiento relacionado a la identificación de los intereses clave en la arquitectura, así como el razonamiento del uso de métodos o técnicas para identificarlos.
- Si es el caso, incluir el razonamiento relacionado a la selección de los patrones o estilos arquitectónicos de la arquitectura.
- Incluir el razonamiento sobre técnicas utilizadas en la evaluación y/o análisis de la arquitectura.
- Incluir el razonamiento relacionado a la información incluida y presentada en la documentación de la arquitectura (vistas, modelos, etc.).

13.1.3. Evidencias de alternativas y su razonamiento

En una descripción de arquitectura se puede incluir información o fuentes de información, que proporcionen evidencia sobre el razonamiento arquitectónico hecho. Se propone considerar las siguientes fuentes de información:

- Minutas o documentos sobre reuniones, que contengan información que permitió realizar algún razonamiento relacionado a las actividades del proceso de arquitectura.
- Documentación de arquitecturas de sistemas similares, que hayan permitido reusar conocimiento y razonar sobre aspectos de la arquitectura del sistema.
- Documentos con información sobre los métodos empleados para identificar intereses e interesados claves para la arquitectura.
- Documentos con información proporcionada por personal externo a la organización, tales como usuarios futuros, proveedores o evaluadores, que proporcionen evidencia sobre alternativas y razonamiento de la arquitectura.

13.2. Documentación de decisiones arquitectónicas

El estándar ISO/IEC/IEEE 42010:2011 establece que en una descripción de arquitectura se deben registrar las decisiones consideradas clave en el proceso de arquitectura del sistema. Probablemente no sea práctico registrar cada decisión hecha en torno a la arquitectura del sistema, así que debe aplicarse un criterio establecido por la organización y/o proyecto para seleccionar las más importantes. Los criterios que se puede considerar son los siguientes (ISO/IEC/IEEE 42010:2011):

- Decisiones respecto a requerimientos arquitectónicamente significativos. Dichos requerimientos pueden incluir metas de negocio, atributos de calidad, restricciones, requerimientos funcionales, etc.
- Decisiones que puedan requerir una mayor inversión de esfuerzo o tiempo. Ejemplos de dichas decisiones pueden ser la selección de los Lenguajes de Descripción de Arquitectura, selección de tecnologías, o definición de puntos de vista arquitectónicos.
- Decisiones que afecten a los interesados, o a un conjunto de interesados clave. Ejemplos de dichos interesados pueden ser clientes, o integrantes del equipo de implementación.
- Decisiones que requieren un razonamiento no obvio, o complicado.
- Decisiones que son altamente sensibles a cambios.
- Decisiones cuyo cambio puede ser costoso.
- Decisiones que forman una base para la planeación y administración del proyecto.
- Decisiones que resultan en gastos de capital o costos indirectos.

Cuando se registren las decisiones de un sistema, se debe considerar lo siguiente:

- La decisión es identificada como única.
- La decisión está declarada.
- La decisión está vinculada a los intereses del sistema a los que pertenece.
- Identificar al propietario de cada decisión.
- La decisión está vinculada a elementos de la descripción de arquitectura (intereses, interesados, etc.) afectados por dicha decisión.
- Existe razonamiento vinculado a una decisión.
- Se debe identificar las restricciones y supuestos que influyen la decisión.
- Se deben registrar las alternativas que han sido consideradas y sus consecuencias potenciales.
- Se deben registrar las consecuencias de la decisión (relacionada a otras decisiones)
- Se pueden considerar registros del momento en que la decisión fue hecha, cuando fue aprobada y cuando fue cambiada.
- Se deben proporcionar citas a fuentes de información adicional.

Nota: puede ser útil registrar las relaciones entre las decisiones arquitectónicas. Ejemplos de tipos de relaciones entre decisiones son: restricciones, influencias, habilita, desencadena, refina, está en conflicto con, es compatible con.

Capítulo 14. Validación y mantenimiento de una descripción de arquitectura

En capítulos previos de este trabajo (del 5 al 13), se presentó una guía para documentar arquitecturas de software a través de descripciones de arquitectura utilizando Lenguajes de Descripción de Arquitectura y con base en el estándar ISO/IEC/IEEE 42010:2011, sin embargo, es de suma importancia presentar algunas consideraciones para validar y dar mantenimiento a dicha documentación.

14.1. Mantenimiento

Durante el desarrollo de un sistema pueden cambiar los requerimientos, propiedades o restricciones del mismo. Esto implica que haya un cambio continuo en el sistema (y por lo tanto en su arquitectura) con el que se tiene que lidiar para que el sistema siga cubriendo las necesidades de sus interesados.

El desafío que implican los cambios en la arquitectura de un sistema, es que si la descripción de la arquitectura deja de reflejar la realidad, es decir, comienza a divergir de la arquitectura que describe, pronto dicha documentación dejará de ser usada porque ya no será útil y perderá su valor. Con el objetivo de evitar esto, es importante actualizar la descripción de arquitectura regularmente para que siga siendo relevante para sus interesados.

Una descripción de arquitectura concisa y relativamente pequeña puede ser más fácil de actualizar, por lo que puede ser adecuado mantenerla de un tamaño manejable.

Para realizar la actualización o revisión de una descripción de arquitectura, se debe poder capturar y establecer sus líneas base. Cuando los documentos que integran la descripción de arquitectura se manejan en papel puede resultar sencillo, pero cuando los elementos que la conforman (como vistas o modelos arquitectónicos) se encuentran en uno o más repositorios, puede llegar a ser más difícil establecer una línea base a través de los repositorios y herramientas, las cuales tal vez no cuenten con mecanismos para versionar o administrar la configuración (página ISO/IEC/IEEE 42010:2011).

En la sección 6.2.12 de este trabajo, se presentan algunas consideraciones para incluir en una descripción de arquitectura información sobre control de versiones y administración de la configuración, lo que ayuda en el mantenimiento y evaluación de dicha documentación.

14.2. Validación

La validación permite asegurar que la documentación de la arquitectura será útil para quienes está destinada, es decir, que podrá ser entendida y usada por sus interesados.

Para validar una descripción de arquitectura se debe asegurar que cumpla con ciertas propiedades que la hagan útil. Por ejemplo, para Rozansky y Woods (2005) una descripción de arquitectura efectiva cuenta con las siguientes características:

- Correctitud o exactitud: la arquitectura se ha documentado adecuadamente, representando correctamente las necesidades de sus interesados.
- Suficiencia: presenta suficiente detalle para responder las preguntas importantes de la arquitectura, para lo que se debe seleccionar las vistas y puntos de vista apropiados.
- Concisión: se debe centrar en los aspectos importantes.
- Claridad: debe ser entendida por su audiencia.
- Actualizada: debe reflejar la evolución de la arquitectura.
- Precisión: debe describir la arquitectura exactamente, con suficiente detalle para permitir al sistema ser diseñado e implementado pero sin dejar de ser concisa.

Es importante mencionar que el estándar ISO/IEC/IEEE 42010:2011 es un conjunto de buenas prácticas, que al ser consideradas, permiten generar descripciones de arquitectura útiles. En esta guía se incluyen técnicas, consejos, ejemplos y consideraciones con base en dicho estándar para lograr tal objetivo en torno a las descripciones de arquitectura. Por ejemplo, considerando las características señaladas por Rozansky y Woods (2005), esta guía las cubre de la siguiente manera:

- Para lograr la correctitud o exactitud se deben conocer claramente cuáles son los interesados y sus intereses específicos que serán presentados en la descripción de arquitectura, parte que fue cubierta en el capítulo 7.
- La selección y elaboración adecuada de puntos de vista y vistas arquitectónicas, presentada en los capítulos 9 y 10, son parte fundamental para que la descripción de arquitectura cuente con la suficiencia adecuada.
- En los capítulos 10 y 11 se incluye una guía para representar los intereses clave de la arquitectura a través de vistas y sus modelos arquitectónicos, incluyendo consejos para que la documentación se centre en los aspectos más importantes. Tales consejos incluyen el uso de la abstracción y la aspiración a la simplicidad, lo que permiten hacer que la descripción de arquitectura sea concisa, y al mismo tiempo, lo suficientemente detallada para que sea precisa y clara. Adicionalmente, los capítulos 12 y 13 presentan las pautas para registrar en una descripción de arquitectura, las relaciones entre elementos arquitectónicos, y el razonamiento y decisiones arquitectónicas que la hacen clara y entendible.
- Como parte de este capítulo (14) se presentan consideraciones para mantener una descripción de arquitectura actualizada.
- En el capítulo 6, se presentan consejos y consideraciones para incluir en una documentación de arquitectura información sobre control de versiones, glosarios, y otro tipo de información general que resulte útil para la audiencia de la documentación.

Aunque seguir las consideraciones que se presentan en esta guía debería resultar en la elaboración de una descripción de arquitectura útil, es también adecuado contar con una lista de verificación que permita guiar la verificación de la descripción de arquitectura, por lo que a continuación se presentan dos listas de verificación que pueden ser de utilidad para el lector.

La lista de verificación presentada en la tabla 14.1, está basada en las especificaciones de validación de Clements et al. (2003) para ayudar a validar si una descripción de arquitectura está documentada lo suficientemente bien para ser entendida y usada. Además, dicha lista incluye las secciones de esta guía que cubren las validaciones de cada uno de los puntos citados en ella.

Validación	Capítulos
¿La descripción de arquitectura es consistente con los interesados que la usarán?	Capítulos 7, 9 y 10.
¿Se Identificó el conjunto correcto de interesados clave y sus intereses?	Capítulo 7, sobre la identificación de interesados e intereses.
¿Se seleccionó el conjunto correcto de vistas y puntos de vista?	Capítulos 9 y 10 sobre identificación, desarrollo y documentación de vistas y puntos de vista arquitectónicos.
¿Se incluyó la información adecuada sobre su organización para encontrar fácilmente cierta información?	Capítulo 6, sobre información general de una descripción de arquitectura.
¿La descripción de arquitectura es consistente consigo misma? Lo que significa estar libre de errores de ambigüedad y contradicción.	Capítulos 10, 11, 12 y 13, y apéndice “D”.
¿Se incluyeron mapeos entre vistas?	Capítulo 10, en la sección sobre análisis de vistas y modelos arquitectónicos.
¿Se Incluyeron las inconsistencias entre las vistas?	Capítulo 12, en el análisis de consistencia.
¿Se incluyeron las afirmaciones tales como hechos, heurísticas, requerimientos, decisiones, etc.?	Capítulo 13, sobre documentación de razonamiento y decisiones arquitectónicas.
¿Se aseguró que no existan dos términos que signifiquen lo mismo o que un término se use para dos cosas diferentes?	Cubierto implícitamente en distintos capítulos de este trabajo, por ejemplo, en el capítulo 11 que incluye modelado y especificación utilizando Lenguajes de Descripción de Arquitectura.
¿Se aseguró que no se repitiera información innecesaria en lugares diferentes en una descripción de arquitectura?	Considerado en la forma en que se documenta cada parte de una vista arquitectónica, y en las plantillas de la misma, presentadas en el apéndice “D”, en las que se propone incluir una única vez cada elemento de información, y en otras secciones únicamente hacer referencia a ellas.
¿La descripción de arquitectura es consistente con una buena forma?	Todos los capítulos del 6 al 14.
¿La estructura y secciones de la documentación están definidas a través de	Todos los capítulos de la guía presentada en este trabajo, incluyen

plantillas o un estándar?	las consideraciones y plantillas para realizar descripciones con base en la norma ISO/IEC/IEEE 42010:2011.
Para cada vista, ¿se proporcionó su definición?, es decir, elementos, relaciones, y propiedades.	Considerado como parte de los capítulos 10 y 11, que abordan la elaboración de vistas arquitectónicas y sus modelos, incluyendo sus elementos, relaciones, y propiedades.
¿Contiene razonamiento, restricciones, entorno, y alternativas rechazadas que puedan ser útiles?	Incluido como parte de la documentación del razonamiento y las decisiones arquitectónicas del capítulo 13.
¿Incluye rastros de requerimientos incluidos?	Incluido en el capítulo 7.
¿Incluye como ejercer variabilidades?	Incluido como parte del capítulo 13.
¿La descripción de arquitectura es consistente con la arquitectura que describe?	Capítulos, 7, 9, 10, 11 y 14
¿La descripción de arquitectura es precisa?	En distintos capítulos como en el 7, 9, 10 y 11, se incluyen consejos para que la documentación se centre en los aspectos importantes de la arquitectura.
¿La descripción de arquitectura está actualizada?	En este capítulo (14) se incluyen consideraciones para el mantenimiento y actualización de la descripción de arquitectura.
¿La descripción de arquitectura está completa?	En este capítulo (14) se presenta una lista de verificación para asegurar que la descripción de arquitectura cumple con lo especificado por el estándar ISO/IEC/IEEE 42010:2011.

Tabla 14.1 Lista de verificación de una descripción de arquitectura (Clements et al. 2003).

Adicionalmente se presenta en forma de lista de verificación, las consideraciones establecidas por el estándar ISO/IEC/IEEE 42010:2011 para asegurarse que la descripción de arquitectura esté completa según dicho estándar.

Sección de la descripción de arquitectura	Contenido de la sección	Secciones que abordan la sección
Información general	Información sobre el sistema de interés.	Capítulo 6
	Información complementaria (referencias, glosarios, etc.).	Capítulo 6
	Resultados de evaluaciones de la arquitectura y de la descripción de arquitectura.	Capítulos 6 y 14
Interesados	Interesados clave del sistema.	Capítulo 7
Intereses	Intereses clave del sistema.	Capítulo 7

Relación entre interesados e intereses	Relación entre interesados e intereses.	Capítulo 7
Puntos de vista	Intereses enmarcados por el punto de vista.	Capítulo 9
	Los interesados típicos para los intereses enmarcados por el punto de vista.	Capítulo 9
	Tipos de modelo usados por el punto de vista.	Capítulo 9
	Para cada tipo de modelo identificado, los lenguajes, notaciones, convenciones, técnicas de modelado, métodos analíticos y otras operaciones que pueden ser usadas en los modelos de ese tipo.	Capítulo 9
	Referencias a fuentes de información.	Capítulo 9
Vistas arquitectónicas	Información complementaria que es especificada por la organización y/o proyecto.	Capítulo 10
	Punto de vista que la gobierna.	Capítulo 10
	Modelos arquitectónicos.	Capítulos 10 y 11
	Asuntos conocidos en la vista con respecto al punto de vista que la gobierna.	Capítulos 10
Reglas de correspondencia	Análisis de consistencia de las vistas y modelos arquitectónicos.	Capítulos 10 y 12
	Correspondencias entre los elementos de la descripción de arquitectura.	Capítulo 12
	Reglas de correspondencia aplicables a los elementos de la descripción de arquitectura.	Capítulo 12
Razonamiento y decisiones arquitectónicas	Razonamiento de las decisiones tomadas.	Capítulo 13
	Decisiones tomadas.	Capítulo 13

Tabla 14.2 Lista de verificación de una descripción de arquitectura según el contenido especificado por el estándar ISO/IEC/IEEE 42010:2011.

Capítulo 15. Evaluación de la guía

El objetivo de realizar una evaluación a la guía presentada en este trabajo, fue validar su utilidad, claridad y detalle. Además, la evaluación permitió identificar las principales fortalezas, debilidades, y mejoras que podrían aplicarse a la guía.

La evaluación de la guía se realizó considerando el ámbito profesional de Arquitectura de Software y fue dirigida por un experto en el área de Arquitectura de Software. Dicho experto es el M. en I. Alejandro Alberto Ramírez Ramos, director de Gobernanza, Arquitectura, Seguridad y Pruebas en la empresa Ultrasist, y cuya experiencia en la Ingeniería de Software abarca más de 10 años. Dentro de sus funciones como director, se encuentran la definición y revisión de los modelos arquitectónicos de clientes internos y externos. Además, tiene experiencia en la definición, diseño e implementación de arquitecturas empresariales y orientadas a servicios (SOA) en proyectos de seguridad nacional.

15.1. Resultados de la evaluación

Como resultado de la evaluación, se obtuvo un conjunto de observaciones, recomendaciones y propuestas de mejora a la guía, las cuales son descritas a continuación.

15.1.1. Evaluación de las características principales

El resultado de la evaluación concluye que la guía es útil en cuanto a que permite conducir de forma adecuada la documentación de arquitecturas de software. Esto significa que puede ser empleada como un medio de dirigir la realización de actividades de modelado, especificación, y en general, de documentación arquitectónica a través de descripciones de arquitectura.

La utilidad de la guía se debe a que explica detallada y claramente los pasos a seguir para documentar una arquitectura de software mediante descripciones de arquitectura, aunque podría parecer que se presenta de manera muy extensa y en algunos aspectos es repetitiva.

Por otra parte, la guía podría ser utilizada para documentar cualquier arquitectura de software, en cualquier proyecto u organización, sin embargo, como resultado de la evaluación se concluyó que se podría limitar el uso de Lenguajes de Descripción de Arquitectura para “escenarios más simples”.

En cuanto a los ejemplos presentados, se concluyó que son claros, y aunque algunos de ellos podrían parecer simples, son entendibles y pueden servir como base para elaborar algunos más complejos.

Por otra parte, el experto que dirigió la evaluación, concluyó que los anexos de la guía son “bastante útiles”, es decir, son de gran ayuda para generar descripciones de arquitectura. También afirmó la utilidad de las plantillas de documentación presentadas en el apéndice “D”.

En cuanto a los Lenguajes de Descripción de Arquitectura, es posible utilizarlos en las organizaciones que desarrollan software, sin embargo, es necesario identificar muy bien en qué escenarios resultarían más útiles.

15.1.2. Principales aportaciones

Como resultado de la evaluación, el experto en Arquitectura de Software certificó que las principales fortalezas y aportaciones de la guía son las siguientes:

- Representa un compendio muy completo de cómo documentar una arquitectura de software.
- Contiene explicaciones detalladas y muy claras de los pasos, conceptos y técnicas utilizadas.
- El enfoque es lo suficientemente genérico para soportar diversas arquitecturas de software.

15.1.3. Debilidades identificadas

Como resultado de la evaluación, se identificaron las siguientes posibles debilidades de la guía:

- La guía se puede utilizar para documentar cualquier arquitectura de software, sin embargo se debe considerar que el uso de los Lenguajes de Descripción de Arquitectura podría resultar innecesario en algunos escenarios, por ejemplo, en arquitecturas de sistemas sumamente simples, o en la documentación de arquitecturas de sistemas que son desarrollados siguiendo metodologías ágiles.
- La guía podría requerir una vista más concreta respecto a los pasos específicos que dirigen a la guía, ya que puede parecer repetitiva en algunos conceptos.

15.1.4. Propuestas de mejora

Como resultado de la evaluación, surgieron las siguientes propuestas para mejorar la guía:

- Generar una guía rápida, *top-down*, que resalte la simplicidad de los pasos principales y que a su vez permita profundizar en cada una de los temas de manera estructurada.
- Incluir un modelo de madurez en la documentación de arquitecturas de software que permita ajustar el nivel de detalle y formalidad de la descripción arquitectónica con base en:
 - El tiempo y recursos asignados para la documentación arquitectónica.
 - El tamaño, complejidad y criticidad del sistema que se está describiendo.
 - El nivel de madurez y conocimiento arquitectónico de la organización.
- Incluir ejemplos de punta a punta del uso de la plantilla para describir la arquitectura de un sistema real de:
 - complejidad alta
 - complejidad media
 - complejidad baja

- Incluir en el apéndice B los atributos de calidad en uso de acuerdo a la norma ISO/IEC 25010.
- Agregar claridad al capítulo 8 sobre los criterios para seleccionar Lenguajes de Descripción de Arquitectura. En específico:
 - Incluir un mapeo específico entre los intereses e interesados de los Lenguajes de Descripción de Arquitectura presentados.
 - Especificar en qué escenario conviene emplear un Lenguaje de Descripción de Arquitectura contra un modelado con alguna notación semiformal, considerando el costo de aprender y explicar a los interesados alguno de estos lenguajes contra el beneficio.

15.2. Mejoras realizadas

Las mejoras realizadas a la guía después de su evaluación e identificación de sus debilidades y propuestas de perfeccionamiento, fueron las siguientes:

- Se generó un tipo de “guía rápida” que resalta la simplicidad de los pasos principales, y que a su vez, permite profundizar en cada una de los temas de manera estructurada. Dicha guía es presentada en la sección 5.5.
- Se incluyeron ejemplos de punta a punta del uso de la plantilla para describir la arquitectura de sistemas reales. Tales ejemplos se presentan en el apéndice “E”.
- Se incluyeron en el apéndice “B”, los atributos de calidad en uso de acuerdo a la norma ISO/IEC 25010.
- Se incrementó la claridad del capítulo 8 sobre los criterios para seleccionar Lenguajes de Descripción de Arquitectura. En específico:
 - Se Incluyó un mapeo específico entre los intereses e interesados de los Lenguajes de Descripción de Arquitectura presentados. Esto se puede visualizar en la tabla 8.3.
 - Se especificó los escenarios en que conviene emplear un Lenguaje de Descripción de Arquitectura contra un modelado con alguna notación semiformal, considerando el costo de aprender y explicar a los interesados algunos de estos lenguajes contra el beneficio. Esto es presentado en la sección 8.7.

Conclusiones

Sobre el cumplimiento de los objetivos

El objetivo general de este trabajo: *desarrollar y presentar una guía que permita realizar las tareas de documentación arquitectónica, y que permita elaborar descripciones de arquitectura útiles, precisas, y entendibles*, fue cumplido, ya que después de realizar la evaluación de la guía, se concluyó que es útil para documentar cualquier arquitectura de software de manera precisa y clara.

Los objetivos particulares, presentados en la introducción de este trabajo, también fueron cumplidos:

- El objetivo particular #1 se cumplió, ya que la guía desarrollada está basada en las especificaciones de la norma ISO/IEC/IEEE 42010:2011. Tal guía es presentada en los capítulos 5 a 13 de este trabajo.
- El objetivo particular #2 se cumplió, ya que en la guía se presenta un conjunto de actividades y técnicas detalladas que permitan realizar la documentación de arquitecturas de software a través de descripciones de arquitectura. Tales actividades y técnicas son presentadas en los capítulos 6 a 14.
- El objetivo particular #3 se cumplió, ya que la guía se enfoca en la selección y uso de Lenguajes de Descripción de Arquitectura como notación de modelado y especificación arquitectónica. La incorporación de tales lenguajes se puede apreciar en la mayoría de los capítulos que constituyen la guía, principalmente en los capítulos 8, 10, y 11.
- El objetivo particular #4 se cumplió, ya que la guía elaborada incluye plantillas que permiten elaborar la documentación arquitectónica al permitir generar productos de trabajo de software de manera sencilla. Dichas plantillas se presentan en el apéndice D de este trabajo.
- El objetivo particular #5 se cumplió, ya que la guía incluye ejemplos prácticos que apoyan en la documentación arquitectónica. También incluye ejemplos que permiten el modelado y la especificación arquitectónica mediante el uso de algunos LDAs (Acme, Wright, AADL y xADL), tal como se aprecia en el capítulo 11.
- El objetivo particular #6 se cumplió, ya que la guía incluye consejos y recomendaciones para realizar la documentación de arquitecturas de software.

Sobre el desarrollo del trabajo

Para el cumplimiento de los objetivos planteados (ver introducción), se realizaron las siguientes actividades:

- La identificación y análisis de los conceptos fundamentales definidos por el estándar ISO/IEC/IEEE 42010:2011 involucrados con el contenido y desarrollo de las descripciones de arquitectura, tales conceptos incluyen interesados, intereses puntos de vista, vistas y modelos arquitectónicos. Esta actividad resultó fundamental ya que permitió establecer un marco conceptual a partir del cual se elaboró la guía para documentar arquitecturas de software, y permitió definir el contenido y forma que deben tomar las descripciones de arquitectura. Dicho marco

conceptual está plasmado en los primeros cuatro capítulos de este trabajo y es esencial para entender la guía.

- La identificación y análisis de las actividades que implica el desarrollo de una descripción de arquitectura con base en el estándar ISO/IEC/IEEE 42010:2011. Esta actividad se realizó con el fin de identificar los pasos a seguir para realizar la documentación arquitectónica y está plasmada en los capítulos 5 al 14.
- Debido a que la guía incluye el uso de Lenguajes de Descripción de Arquitectura como notación de modelado arquitectónico, se requirió identificar y analizar las actividades de documentación que involucran el uso de dichos lenguajes.
- Se realizó el análisis de distintos tipos de Lenguajes de Descripción de Arquitectura que existen con el fin de identificar aquellos que serían incluidos en secciones específicas de la guía, y con el objetivo de presentar las consideraciones necesarias para su uso. Además, se requirió el estudio minucioso de algunos Lenguajes de Descripción de Arquitectura (AADL, Acme, Wright y xADL) para elaborar secciones específicas de la guía, tales como el capítulo de modelado arquitectónico (capítulo 11). El resultado de esta actividad está plasmada en capítulos como el 4, el 8 y el 12.
- Se definieron, desarrollaron y documentaron las actividades requeridas para elaborar descripciones de arquitectura, y que conforman la guía. Tales actividades son:
 - Identificación y documentación de información general de una descripción de arquitectura.
 - Identificación y documentación de interesados e intereses.
 - Selección de Lenguajes de Descripción de Arquitectura.
 - Identificación y documentación de puntos de vista arquitectónicos.
 - Desarrollo y documentación de vistas arquitectónicas.
 - Modelado y especificación arquitectónica utilizando Lenguajes de Descripción de Arquitectura.
 - Documentación de relaciones arquitectónicas.
 - Documentación de razonamiento y decisiones arquitectónicas.
 - Validación y mantenimiento de una descripción de arquitectura.

Para la definición y desarrollo de las actividades antes mencionadas se consideraron e incluyeron diversas técnicas, prácticas y consejos de Arquitectura de Software. El resultado de desarrollar tales actividades es la elaboración de los capítulos 5 a 14.

- Se elaboraron plantillas de documentación para cada uno de los elementos que conforman una descripción de arquitectura, tales como puntos de vista y vistas arquitectónicas. Tales plantillas se presentan en el apéndice “D”.
- Se realizó la evaluación de la guía, que fue dirigida por un experto del área de Arquitectura de Software. El resultado de la evaluación se presenta en el capítulo 15.

Sobre aspectos de Arquitectura de Software relacionados con el trabajo

El estándar ISO/IEC/IEEE 42010:2011 es un conjunto de buenas prácticas para la documentación de arquitecturas de software y otras actividades del proceso de arquitectura, y facilita la comprensión y elaboración de descripciones de arquitectura entre diferentes personas, equipos de trabajo, u organizaciones que tengan conocimiento sobre el estándar. Esto quiere decir que si se conoce el estándar y las disposiciones que deben cumplir las descripciones de arquitectura, resulta más fácil poder entender y trabajar

sobre los mismos documentos arquitectónicos, lo cual permite eliminar algunas barreras técnicas.

Se apreció que la implementación del estándar ISO/IEC/IEEE 42010:2011 implica trabajo, ya que para obtener una descripción de arquitectura deben realizarse diversas actividades como la identificación de intereses e interesados, la definición de puntos de vista, y el desarrollo de vistas y modelos arquitectónicos. Además, tales actividades demandan un conocimiento sólido sobre distintos conceptos, técnicas y prácticas de Arquitectura de Software.

El estándar ISO/IEC/IEEE 42010:2011 define las características que deben cumplir las descripciones de arquitectura, pero al mismo tiempo es suficientemente abierto para ser implementado en distintas organizaciones, proyectos y sistemas, e incorporar las prácticas y técnicas que convengan en cada caso, por lo que se requiere un buen dominio de Arquitectura de Software y la capacidad para tomar decisiones adecuadas.

Muchos de los diversos enfoques de documentación arquitectónica que existen comparten conceptos en común, los cuales son también considerados y normalizados por el estándar ISO/IEC/IEEE 42010:2011. Por tal razón, es fácil utilizar el estándar en conjunto con otros enfoques de documentación, además, el uso del estándar no implica un cambio total en el proceso de documentación arquitectónica de una organización, si no que puede enriquecerlo.

Una descripción de arquitectura elaborada a partir del estándar ISO/IEC/IEEE 42010:2011, debe demostrar cómo una arquitectura satisface las necesidades de sus interesados. Además, los intereses arquitectónicos de los diversos interesados son dirigidos por una descripción de arquitectura construida con múltiples vistas arquitectónicas del sistema, donde cada vista cubre un conjunto de intereses. Por esto, la identificación de intereses e interesados son actividades cruciales en la documentación arquitectónica, y en general, en el proceso de arquitectura. Sin embargo, durante la elaboración de este trabajo se pudo apreciar que no existen muchos procedimientos o técnicas sistemáticas para la identificación o priorización de los interesados considerados fundamentales para la arquitectura de un sistema.

Los puntos de vista son elementos indispensables de una descripción de arquitectura ya que permiten evaluar aspectos de la arquitectura y su documentación, y elaborar las vistas y modelos arquitectónicos. Una de sus principales ventajas es que son productos de trabajo reusables, y aunque su elaboración implica trabajo, deben ser definidos de manera correcta para poder elaborar y analizar vistas y modelos arquitectónicos adecuadamente. Muchos de los distintos enfoques de documentación arquitectónica que existen incluyen puntos de vista específicos que pueden ser útiles para realizar una descripción de arquitectura, sin embargo, se debe tener cuidado al seleccionar alguno ya que éstos pueden ser incompatibles con algunas notaciones que se pretenda utilizar en el modelado arquitectónico.

Las vistas arquitectónicas son una parte esencial de una descripción de arquitectura ya que incluyen modelos arquitectónicos, por lo que su elaboración y análisis requiere ser sustentado a partir de puntos de vista bien definidos. Un aspecto de las vistas arquitectónicas que resulta crucial es su consistencia, tanto como entre sus modelos, como entre las diferentes vistas de una descripción de arquitectura. Además, hay que

considerar que muchos enfoques de documentación se centran en vistas y modelos arquitectónicos, y éstos últimos pueden usarse para realizar el desarrollo de software basado en modelos.

Se pudo notar que la elaboración de las vistas arquitectónicas es una de las actividades que más requieren trabajo en el desarrollo de una descripción de arquitectura, sin embargo, dicho trabajo aporta grandes beneficios ya que las vistas son ampliamente utilizadas en el análisis, diseño detallado, implementación, despliegue o pruebas del sistema.

Se pudo apreciar que muchos de los enfoques de documentación arquitectónica que existen no hacen énfasis en documentar las correspondencias y las reglas de correspondencias de una descripción de arquitectura, sin embargo, tales elementos son esenciales para realizar análisis de consistencia de una arquitectura y de su documentación.

La documentación del razonamiento arquitectónico es una actividad que permite plasmar y guardar lecciones aprendidas y mejores prácticas, y entender mejor la arquitectura de un sistema, por lo que puede aportar grandes beneficios a una organización o proyecto.

Se puede concluir que aunque los Lenguajes de Descripción de Arquitectura no son un medio muy utilizado para modelar arquitecturas de software, su uso proporciona grandes ventajas. Se pudo apreciar que en la actualidad existe un gran número de Lenguajes de Descripción de Arquitectura, y tal variedad puede implicar un desafío para su uso, por lo que se debe tener un buen conocimiento de los distintos lenguajes que existen y saber realizar una buena selección.

Por otra parte, se analizaron y utilizaron algunos Lenguajes de Descripción de Arquitectura específicos, y se pudieron identificar sus ventajas y desventajas, tal como se presenta a detalle en la sección 11.5.

Sobre los resultados de evaluar de la guía

Después de realizar la evaluación de la guía se pudo concluir que aunque es extensa, es una herramienta detallada, entendible y completa, que en general, puede dirigir la documentación de cualquier arquitectura de software mediante descripciones de arquitectura realizando una serie de pasos.

Aunque la guía es genérica para documentar cualquier arquitectura de software, el uso de Lenguajes de Descripción de Arquitectura debe evaluarse para cierto tipo de escenarios.

Se puede afirmar que las plantillas y los ejemplos presentados en la guía son muy útiles para generar descripciones de arquitectura.

Trabajo futuro

Se puede incluir un modelo de madurez en la documentación de arquitecturas de software que permita ajustar el nivel de detalle y formalidad de la descripción arquitectónica con base en:

- El tiempo y recursos asignados para la documentación arquitectónica.
- El tamaño, complejidad y criticidad del sistema que se está describiendo.
- El nivel de madurez y conocimiento arquitectónico de la organización.

Por otra parte, se puede implementar la guía en diferentes tipos de sistemas y organizaciones, realizando una evaluación más exhaustiva, y asegurar que es útil en tales escenarios.

Apéndices

Los apéndices presentados en esta sección son los siguientes:

- Apéndice A *Catálogo de interesados*: presenta un catálogo de posibles interesados en la arquitectura de un sistema.
- Apéndice B *Catálogo de atributos de calidad*: presenta un catálogo de atributos de calidad con base en el estándar ISO/IEC 25010.
- Apéndice C *Catálogo de puntos de vista*: presenta un catálogo de puntos de vista considerando distintos enfoques de documentación.
- Apéndice D *Plantillas de documentación*: presenta un conjunto de plantillas para documentar los elementos de una descripción de arquitectura.
- Apéndice E *Ejemplo de una descripción de arquitectura*.

A. Catálogo de interesados

Este catálogo incluye algunos de los interesados que pueden resultar fundamentales para la arquitectura de un sistema.

Administrador de base de datos

Sus actividades incluyen el diseño de bases de datos; el análisis, modelado, almacenamiento, recuperación, procesamiento y optimización de datos; la instalación de software de base de datos; y el monitoreo y administración de la seguridad de datos.

Sus intereses respecto a la arquitectura del sistema están relacionados en entender cómo los elementos de la arquitectura crean, almacenan, procesan y usan los datos. Por otra parte, un administrador de base de datos puede interesarse en las propiedades que requieren los datos o la base de datos para lograr las metas de negocio del sistema, y que son reflejadas en su arquitectura.

Administrador de línea de productos

Un administrador de línea de productos es responsable del desarrollo de una familia entera de productos, utilizando para todos los mismos activos fundamentales, los cuales incluyen aspectos arquitectónicos.

Algunos de sus intereses en la arquitectura, se relacionan con determinar si un potencial miembro de la familia o línea de productos, está o no fuera del alcance, y si está fuera, qué tanto (Bass, Clements y Kazman 2012).

Administrador de proyectos

Es el responsable de planear, dar secuencia, programar, y asignar los recursos requeridos para desarrollar y entregar los componentes de software del sistema. También establece el presupuesto y calendarización del sistema, estima su progreso, y resuelve la contención de recursos en tiempo de desarrollo.

Sus intereses en la arquitectura se enfocan en administrar los recursos (monetarios, humanos, tecnológicos, tiempo, etc.) destinados para realizar el proceso de arquitectura. Ejemplos de estos intereses son las restricciones tecnológicas o de tiempo consideradas para realizar el diseño o documentación de la arquitectura.

Administrador/operador de sistema

Un administrador de sistemas puede realizar tareas de configuración, monitoreo, operación, ejecución, documentación, y aseguramiento del correcto funcionamiento del sistema una vez que éste ha sido desplegado. En ambientes comerciales de gran escala, juega un papel clave debido a que la operación del sistema puede ser esencial para la

continuidad de la empresa. En algunos casos, como en los sistemas destinados a uso doméstico, los administradores del sistema pueden ser también usuarios. Los administradores de sistema pueden desempeñar el papel de operadores.

Algunos de sus intereses se centran en la documentación de la arquitectura del sistema, ya que ésta puede ser de gran utilidad para el monitoreo, operación, y en general, para la adecuada administración del sistema. Por otra parte, algunos de sus intereses se relacionan con los atributos de calidad que el sistema debe presentar al momento de su ejecución, tal es el caso de la recuperación de desastres, la disponibilidad, la capacidad de adaptación, y la escalabilidad del sistema.

Administrador de red

Es el responsable del despliegue, la configuración, el mantenimiento, y el monitoreo o vigilancia de los componentes de hardware y software que componen la red.

Sus intereses se enfocan en determinar la carga de red durante varios perfiles de uso, y entender el uso de la red (Bass, Clements y Kazman 2012). Además, un administrador de red se interesa por la relación que existe entre los componentes arquitectónicos del sistema, y los componentes de la red, por lo que una adecuada documentación de la arquitectura del sistema puede ser de gran utilidad.

Adquiridor

Supervisa las adquisiciones realizadas para el sistema. Típicamente, los adquiridores incluyen personas de la “alta dirección”, quienes proporcionan o autorizan los fondos destinados para el desarrollo del sistema. De la misma manera, los departamentos adquisitivos y legales que representan los intereses comerciales de los usuarios en negociaciones con los proveedores pueden ser también adquiridores.

Los adquiridores pueden ser referidos a menudo como patrocinadores de negocio o patrocinadores empresariales, y para el desarrollo de un sistema, tienden a ser altos ejecutivos de ventas, marketing y tecnología (Rozansky y Woods 2005).

Los intereses de los adquiridores típicamente se relacionan con las metas estratégicas; el retorno de inversión; y los costos, plazos, planes y recursos involucrados en la construcción y funcionamiento del sistema. Sus intereses se enfocan también en la calidad y en el gasto eficiente de recursos durante la entrega y operación del sistema. (Rozansky y Woods 2005). Estos intereses son fundamentales para la arquitectura de un sistema, ya que pueden guiar el diseño de la misma.

Analista

Algunas de las actividades que realiza un analista, son analizar la viabilidad de desarrollar el sistema; y analizar si se satisfacen los requerimientos del sistema con base en la arquitectura del mismo.

Los analistas a menudo son interesados especializados, es decir, tienen habilidades específicas para el desarrollo de un sistema (por ejemplo, existen analistas de la eficiencia de desempeño y analistas de seguridad), y pueden tener posiciones bien definidas en un proyecto.

Sus intereses se relacionan con analizar la satisfacción de los requerimientos del sistema con base en la arquitectura del mismo. Por esta razón, una documentación adecuada de la arquitectura resulta fundamental para un analista.

Arquitecto

Es el responsable del proceso de arquitectura, que incluye la definición, diseño, documentación, evaluación, y mantenimiento de la arquitectura del sistema. Un arquitecto es un “recipiente” para registrar las decisiones de diseño y proporciona evidencia de que la arquitectura satisface los requerimientos del sistema (Bass, Clements y Kazman 2012).

Sus intereses principales en la arquitectura, se relacionan con la negociación y compensación entre los requerimientos que compiten, y con los distintos enfoques de diseño (Bass, Clements y Kazman 2012). Asimismo, uno de los intereses fundamentales de un arquitecto es la documentación de la arquitectura, la cual permite realizar sus actividades.

Asesor

Supervisa la conformidad del sistema con normas legales y reglamentarias. Los asesores pueden provenir de los propios departamentos internos de control de calidad o conformidad de la organización, o pueden ser personas externas (Rozansky y Woods 2005).

Sus intereses se centran en torno a las pruebas (para demostrar la conformidad de los requisitos), y el cumplimiento demostrable formal (Rozansky y Woods 2005).

Cliente

Puede ser cualquier persona u organización que paga por el sistema y se asegura de su entrega. A menudo, un cliente habla en lugar de un usuario final, o es representante del mismo, pero también puede darse el caso en que los clientes no representen usuarios y requieran el sistema para obtenerlos.

Sus intereses se centran en asegurar que la funcionalidad y calidad requeridas por sistema sean entregadas según lo establecido, para lo cual pueden hacer uso de la arquitectura del sistema y de la documentación de la misma.

Comprobador de conformidad

Es el responsable de asegurar la conformidad de estándares y procesos para proporcionar confianza al sistema.

Sus intereses en la arquitectura se relacionan con las bases para comprobar la conformidad del sistema y para asegurar que las implementaciones han sido fiables a las prescripciones de la arquitectura (Bass, Clements y Kazman 2012).

Comunicadores

Son quienes explican el sistema a otros interesados por medio de documentos o material de entrenamiento. Pueden ser entrenadores públicos o internos que proporcionan capacitación para el personal soporte, desarrolladores, o personal de mantenimiento. Un comunicador puede crear manuales para los interesados del sistema.

Sus intereses se centran en entender la arquitectura a detalle y el razonamiento detrás de ella. Los comunicadores se interesan en explicar la arquitectura a audiencias técnicas y público no especializado (Rozansky y Woods 2005). Un arquitecto, puede asumir el rol de un comunicador. La documentación arquitectónica es fundamental para este tipo de interesados.

Desarrollador/implementador/programador

Construyen el sistema con base en las especificaciones establecidas. Son los responsables de desarrollar elementos específicos de software con base en diseños, requerimientos, y la arquitectura del sistema.

Los gerentes de desarrollo, quienes planean las actividades de desarrollo y conducen los equipos para hacer el trabajo, pueden considerarse también desarrolladores (Rozansky y Woods 2005).

Los intereses de los desarrolladores están relacionados con entender la arquitectura del sistema en general. Algunos de los intereses más específicos relacionados con el desarrollo del sistema son: normas de construcción; selección de la plataforma, lenguajes de programación y otras herramientas de desarrollo; y aspectos sobre mantenibilidad, flexibilidad, y preservación de conocimiento a través del tiempo.

Los desarrolladores se interesan también en entender las restricciones inviolables y las libertades explotables del sistema en actividades de desarrollo, características consideradas por la arquitectura del sistema (Bass, Clements y Kazman 2012).

Diseñador

Es el responsable del diseño del sistema, y por tanto, está fuertemente relacionado con la arquitectura del mismo, ya que hace uso de la arquitectura para diseñar detalladamente el sistema de tal forma que logre sus requerimientos específicos. Un diseñador también puede establecer las estimaciones de consumo de recursos.

Sus intereses en la arquitectura se relacionan con resolver los conflictos de recursos; establecer la eficiencia de desempeño y otros tipos de presupuestos de consumo de recursos en tiempo de ejecución; y entender como las partes del sistema se comunican e interactúan unas con otras, lo que es definido por la arquitectura (Bass, Clements y Kazman 2012). La documentación de la arquitectura es fundamental para estos interesados, ya que permite entender los módulos que conforman el sistema, y estimar el consumo de recursos.

Evaluador

Es el responsable de conducir las evaluaciones formales de la arquitectura y de su documentación considerando algunos criterios claramente definidos.

Sus intereses en la arquitectura se centran en evaluar las habilidades de la arquitectura para cumplir con el comportamiento y los atributos de calidad requeridos por el sistema (Bass, Clements y Kazman 2012). La documentación arquitectónica es fundamental para este tipo de interesados ya que puede guiar sus actividades.

Gerente de negocios

Es el responsable del funcionamiento de la “entidad de negocios” organizacional relacionada con el sistema. Posee responsabilidades gerenciales/ejecutivas, y responsabilidades para definir procesos de negocio.

Sus intereses en la arquitectura se centran en entender la habilidad de la misma para lograr las metas de negocio del sistema (Bass, Clements y Kazman 2012).

Ingeniero de sistemas

Es el responsable de diseñar y desarrollar el sistema o los componentes del mismo en los que el software juega un rol.

Su interés en la arquitectura se centra en asegurar que el entorno del sistema proporcionado por el software es suficiente (Bass, Clements y Kazman 2012).

Personal de mantenimiento

Administra la evolución del sistema una vez que éste está en funcionamiento. Es responsable de corregir errores y proporcionar mejoras al sistema a través de su ciclo de vida, incluyendo la adaptación del sistema para usuarios no previstos.

Sus intereses se enfocan en aspectos de documentación (incluyendo la documentación de la arquitectura), desarrollo, instrumentación (instalaciones para el monitoreo operacional del sistema), entornos de depuración, control de cambios de producción, y la preservación del conocimiento a través del tiempo (Rozansky y Woods).

Otro de sus intereses, es entender las ramificaciones de un cambio (Bass, Clements y Kazman 2012).

Personal de soporte

Proporciona soporte a usuarios del sistema cuando éste se encuentra en ejecución. Ejemplos de estos interesados son: personal de soporte técnico, y departamentos de servicio al cliente.

Sus intereses se centran en tener la información necesaria para resolver los problemas que se presenten con los usuarios, quienes pueden comunicarse vía telefónica, e mail, internet, o en persona (Rozansky y Woods). La documentación de la arquitectura del sistema puede ser de gran utilidad para lidiar con sus actividades, ya que proporciona una descripción de las características fundamentales del sistema.

Propietarios

Los propietarios del sistema pueden incluir clientes, usuarios, o la misma organización que desarrolla el sistema.

Proveedores

Construyen y/o suministran el hardware, software, y la infraestructura requerida para ejecutar el sistema. Pueden proporcionar un equipo especializado para la operación o desarrollo del sistema.

Aunque no se involucran mucho en el desarrollo, uso, o mantenimiento del sistema, pueden imponer restricciones debido a las limitaciones o requerimientos de los productos que suministran.

Sus intereses comúnmente están relacionados con restricciones del sistema, y por tanto de su arquitectura. Por ejemplo, una aplicación puede requerir una versión particular de sistema operativo para funcionar, puede ejecutarse únicamente en determinadas configuraciones de hardware, o puede imponer limitaciones en el número de conexiones

simultáneas. Es esencial que se tomen en cuenta tales limitaciones en el diseño de la arquitectura del sistema (Rozansky y Woods).

Responsable de desplegar el sistema

Son los responsables de aceptar y desplegar el sistema desarrollado, haciéndolo operacional, y haciendo cumplir su función de negocio.

Sus intereses en la arquitectura se relacionan con entender los elementos arquitectónicos que son entregados y que serán instalados en el sitio del cliente o usuario (Bass, Clements y Kazman 2012). . La documentación arquitectónica es fundamental para este tipo de interesados ya que puede permitir guiar sus actividades.

Responsable de integración

Es el responsable de integrar los componentes individuales del sistema considerando el diseño de la arquitectura y los diseños específicos del sistema.

Sus intereses en la arquitectura se relacionan con producir planes y procedimientos de integración, y localizar el origen de las fallas de integración (Bass, Clements y Kazman 2012). Una documentación adecuada de la arquitectura puede resultar bastante útil para un responsable de integración, ya que puede incluir vistas arquitectónicas que representen los módulos del sistema que requieran ser integrados.

Responsables de pruebas

Es el responsable de probar y verificar tanto el sistema entero como los componentes del mismo para asegurar que el sistema es adecuado para su desarrollo o uso. Estas actividades comúnmente se realizan una vez que son definidos los requerimientos y la arquitectura del sistema.

El personal de pruebas puede ser parte del equipo de desarrollo, de otra unidad organizacional, o incluso de una organización distinta.

Sus intereses se centran en diseñar pruebas para probar si los requerimientos del sistema son cumplidos, y en construir sistemas en los cuales se puedan ejecutar sus pruebas. También se interesan en crear pruebas basadas en el comportamiento e interacción de los elementos de software, lo que es definido en el diseño arquitectónico del sistema, y se puede encontrar en documentos de arquitectura (Bass, Clements y Kazman 2012).

Representante de sistemas externos

Es el responsable de administrar un sistema externo con el que debe operar el sistema de interés, y también es responsable de administrar la interfaz entre ambos sistemas.

Sus intereses se enfocan en definir un conjunto de acuerdos entre los sistemas. (Bass, Clements y Kazman 2012). Por otra parte, la arquitectura de un sistema está fuertemente influenciada por el entorno del mismo, por lo que los intereses relacionados con la interacción del sistema con su entorno (incluyendo sistemas externos) son fundamentales para definir la arquitectura.

Usuarios finales

Definen las funcionalidades requeridas por el sistema, y en última instancia, hacen uso de él. En el caso de sistemas internos, los usuarios pueden ser parte de la organización. En caso que el sistema sea un producto de software, los usuarios son compradores eventuales del producto.

Los intereses de los usuarios se centran en el alcance y funcionalidad del sistema, sin embargo, pueden tener también intereses operacionales, tales como la eficiencia de desempeño y la seguridad, que son fundamentales para definir la arquitectura del sistema. Por esto, es crucial para la arquitectura de un sistema identificar puntal y específicamente a todos los usuarios del mismo, ya que sus intereses definen en gran parte la arquitectura.

Sus intereses, en el rol de revisores, se centran en el uso de la documentación de la arquitectura para revisar si la funcionalidad deseada por el sistema está siendo entregada. También pueden usar la documentación de la arquitectura para entender los componentes principales del sistema, que pueden ayudarlos para realizar un mantenimiento de emergencia (Bass, Clements y Kazman 2012).

B. Catálogo de atributos de calidad

En este apéndice se presentan los atributos de calidad definidos por el estándar ISO/IEC 25010 “*System and software quality models*”. Incluye el modelo de calidad del producto de software, y el modelo de calidad en uso.

Modelo de calidad del producto de software

Categoriza las propiedades de calidad de un producto de software en ocho características, que a su vez, están compuestas por un conjunto de subcaracterísticas tal como se muestra en la figura B.1.



Figura B.1 Modelo de calidad del producto software

Adecuación funcional

Representa la capacidad del producto de software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas. Se divide en completitud funcional, corrección funcional, y pertinencia funcional.

Completitud funcional

Grado en el cual, el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario.

Corrección funcional

Capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.

Pertinencia funcional

Capacidad del producto de software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario específicos.

Eficiencia de desempeño

Representa el desempeño relativo de la cantidad de recursos utilizados bajo determinadas condiciones. Se divide en comportamiento temporal, utilización de recursos, y capacidad.

Comportamiento temporal

Los tiempos de respuesta y procesamiento, y los ratios de rendimiento de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un punto de referencia establecido.

Utilización de recursos

Las cantidades y tipos de recursos utilizados cuando el software lleva a cabo su función bajo características determinadas.

Capacidad

Grado en que los límites máximos de un parámetro de un producto o sistema de software cumplen con los requisitos.

Compatibilidad

Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software. Se divide en coexistencia e interoperabilidad.

Coexistencia

Capacidad del producto para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes sin pérdida.

Interoperabilidad

Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

Es sobre el grado en el que dos o más sistemas pueden útilmente intercambiar información significativa por medio de interfaces en un contexto particular. También incluye la capacidad de interpretar correctamente los datos intercambiados (Bass, Clements y Kazman 2012).

Usabilidad

Capacidad del producto software para ser entendido, aprendido, usado, y resultar atractivo para el usuario cuando se usa bajo determinadas condiciones. Se divide en capacidad para reconocer su adecuación, capacidad de aprendizaje, capacidad para ser

usado, protección contra errores de usuario, estética de la interfaz de usuario, y accesibilidad.

Capacidad para reconocer su adecuación.

Capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades.

Capacidad de aprendizaje

Capacidad del producto que permite al usuario aprender su aplicación.

Capacidad para ser usado

Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.

Protección contra errores de usuario

Capacidad del sistema para proteger a los usuarios de cometer errores.

Estética de la interfaz de usuario

Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.

Accesibilidad

Capacidad del producto que permite que sea utilizado por usuarios con determinadas características y discapacidades.

Fiabilidad

Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados. Se divide en madurez, disponibilidad, tolerancia a fallos, y capacidad de recuperación.

Madurez

Capacidad del sistema para satisfacer las necesidades de fiabilidad en condiciones normales.

Disponibilidad

Capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiera.

Tolerancia a fallos

Capacidad del sistema o componente para operar según lo previsto en presencia de fallos de hardware o software.

Capacidad de recuperación

Capacidad del producto software para recuperar los datos directamente afectados y reestablecer el estado deseado del sistema en caso de interrupción o fallo.

Seguridad

Capacidad de protección de la información y los datos, de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos. Se divide en confidencialidad, integridad, no repudio, responsabilidad, y autenticidad.

Confidencialidad

Capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente.

Integridad

Capacidad del sistema o componente para prevenir accesos o modificaciones no autorizados a datos o programas de ordenador.

No repudio

Capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente.

Responsabilidad

Capacidad de rastrear de forma inequívoca las acciones de una entidad.

Autenticidad

Capacidad de demostrar la identidad de un sujeto o un recurso.

Mantenibilidad

Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas. Se divide en modularidad, reusabilidad, analizabilidad, capacidad para ser modificado, y capacidad para ser probado.

Modularidad

Capacidad de un sistema o programa de ordenador (compuesto de componentes discretos) que permite que un cambio en un componente tenga un impacto mínimo en los demás.

Reusabilidad

Capacidad de un activo que permite que sea utilizado en más de un sistema software o en la construcción de otros activos.

Analizabilidad

Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.

Capacidad para ser modificado

Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.

Capacidad para ser probado

Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente, y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.

Portabilidad

Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno de hardware, software, operacional o de utilización, a otro. Se divide en adaptabilidad, capacidad para ser instalado, y capacidad para ser reemplazado.

Adaptabilidad

Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.

Capacidad para ser instalado

Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.

Capacidad para ser reemplazado

Capacidad del producto para ser utilizado en lugar de otro producto software determinado con el mismo propósito y en el mismo entorno.

Modelo de calidad en uso

La calidad en uso es el grado en el que un producto o sistema puede ser utilizado por usuarios específicos para satisfacer sus necesidades para lograr objetivos específicos con efectividad, eficiencia, mitigación de riesgos, satisfacción, y cobertura en contextos

específicos. Las propiedades de calidad en uso se dividen en cinco categorías, tal como se muestra en la figura B.2.



Figura B.2 Modelo de calidad en uso

Efectividad

Exactitud y completitud con los que los usuarios logran objetivos específicos.

Eficiencia

Recursos gastados en relación con la exactitud y completitud con los que los usuarios logran objetivos.

Satisfacción

Grado en el que las necesidades de los usuarios son logradas cuando un producto o sistema es usado en un contexto de uso especificado. Se divide en utilidad, confianza, agrado y comodidad.

Utilidad

Grado en el que un usuario es complacido con su percepción de logro de metas pragmáticas, incluyendo los resultados de uso y las consecuencias de uso.

Confianza

Grado en el cual un usuario o interesado tiene confianza que un producto o sistema va a comportarse según lo previsto.

Agrado

Grado en el que el usuario está complacido de cumplir sus necesidades personales (como la adquisición de nuevo conocimiento o habilidades).

Comodidad

Grado en el que los usuarios son satisfechos con confort físico.

Mitigación de riesgos

Grado en el que un producto o sistema mitiga los riesgos potenciales de estados económicos, vida humana, salud, o el ambiente. Se divide en mitigación de riesgos económicos, mitigación de riesgos de salud, y mitigación de riesgos ambientales.

Mitigación de riesgos económicos

Grado en el que un producto o sistema mitiga los riesgos potenciales de estados financieros, operación eficiente, propiedades comerciales, reputación u otros recursos en los contextos de uso pretendidos.

Mitigación de riesgos de salud

Grado en el que un producto o sistema mitiga los riesgos potenciales de personas en los contextos de uso pretendidos.

Mitigación de riesgos ambientales

Grado en el que un producto o sistema mitiga los riesgos potenciales para los bienes o el ambiente en los contextos de uso pretendidos.

Cobertura de contexto

Grado en el que un producto o sistema puede ser usado con efectividad, eficiencia, mitigación de riesgos y satisfacción en todos los contextos de uso.

Complejidad de contexto

Grado en el que un producto o sistema puede ser usado con efectividad, eficiencia, mitigación de riesgos y satisfacción tanto en contextos de uso especificados como en contextos más allá de esos identificados explícita e inicialmente.

Flexibilidad

Grado en el que un producto o sistema puede ser usado con efectividad, eficiencia, mitigación de riesgos y satisfacción en contextos más allá de esos identificados explícita e inicialmente.

C. Catálogo de puntos de vista

El catálogo de puntos de vista presentado, es un compendio de los puntos de vista propuestos y definidos por los siguientes enfoques de documentación:

- Rozansky y Woods (2005) define un catálogo con los siguientes puntos de vista: contexto, funcional, información, concurrencia, desarrollo, despliegue y operacional. También define un conjunto de perspectivas para ser aplicadas a los puntos de vista.
- Enfoque del SEI (Clements et al. 2003), que define 3 categorías de puntos de vista o tipos de vistas arquitectónicas: de módulos, componentes y conectores, de distribución.
- Kruchten (1995), que con su modelo “4+1” define los puntos de vista: lógico, de desarrollo, de procesos, y físico.

El catálogo presentado a continuación incluye información que puede ser de utilidad para considerar si los puntos de vista pueden ser tomados en cuenta para elaborar descripciones de arquitectura. Para mayor detalle sobre los puntos de vista y los enfoques a los que pertenecen, revisar las fuentes principales.

Enfoque de Rozansky y Woods

Contexto

Describe las relaciones, dependencias, e interacciones entre el sistema y su ambiente, el cual puede incluir personas, sistemas y otras entidades externas.

Aplicabilidad: todos los sistemas.

Intereses

- Principales responsabilidades y alcance del sistema.
- Entidades externas con las que el sistema interactúa.
- Naturaleza y características de las entidades externas (localización física, calidad, etc.).
- Responsabilidades de las interfaces externas.
- Características de interfaces externas.
- Impacto del sistema en su ambiente.
- Completitud, consistencia y coherencia general del sistema.

Tipos de modelos

- Modelo de contexto: explica lo que el sistema hace y lo que no hace, presenta un panorama general de las interacciones del sistema con el mundo exterior, y resume los roles y responsabilidades de los participantes en esas interacciones. Típicamente incluye: el sistema en sí mismo, las entidades externas, y las interfaces entre el sistema y las entidades externas.

Notaciones: la notación más común para generar este tipo de modelos es UML a través de diagramas de líneas y cajas. Sin embargo, muchos Lenguajes de Descripción de Arquitectura permiten modelar sistemas en su ambiente, incluyendo sus interfaces internas y externas, y proporcionando la definición de los roles de cada participante en esas interacciones. Algunos de estos LDAs incluyen a Acme, Wright, xADL y Aesop.

- Escenarios de interacción del sistema con su ambiente.

Notaciones: la notación para este tipo de modelos es informal y puede ser textual.

Funcional

Describe los elementos funcionales de tiempo de ejecución del sistema, sus responsabilidades, interfaces e interacciones primarias.

Aplicabilidad: todos los sistemas.

Intereses

- Capacidades funcionales del sistema.
- Interfaces externas entre el sistema de interés y otros sistemas.
- Estructura interna del sistema.
- Filosofía de diseño funcional, que incluye la separación de intereses, cohesión, consistencia, acoplamiento, extensibilidad, flexibilidad funcional y simplicidad.

Tipos de modelos

- Modelo de estructura funcional: comúnmente cuentan con elementos funcionales, interfaces, conectores, y entidades externas.

Notaciones: algunas de las notaciones para este tipo de modelos son UML y algunos Lenguajes de Descripción de Arquitectura como AADL, que permite especificar características de tiempo de ejecución del sistema, o Wright y Acme, que permiten representar la estructura arquitectónica de un sistema.

Información

Describe a nivel general, la forma en que la arquitectura almacena, manipula, administra y distribuye información.

Aplicabilidad: cualquier sistema que tenga requerimientos no triviales sobre administración de información.

Intereses

- Estructura y contenido general de la información que maneja el sistema.
- Propósito y uso de la información.
- Propiedades de la información cuando se encuentra físicamente distribuida.

- Información que es propiedad de la empresa.
- Referencias a datos específicos, es decir, los identificadores únicos de entidades u objetos de datos.
- Volatilidad de la información, es decir, los cambios en la sintaxis, semántica y relaciones entre la información.
- Flujo de información, es decir, la manera en la que la información se mueve en el sistema, y cómo es accedida, creada y modificada.
- Consistencia de la información.
- Calidad de la información.
- Puntualidad, latencia²⁶ y “edad” de la información.
- Retención de datos y archivos.

Tipos de modelos principales

- Modelos de estructura de datos estáticos: analizan la estructura estática de los datos, los elementos de datos importantes y la relación entre ellos.

Notaciones: Aunque una de las notaciones por defecto para estos modelos es UML, muchos Lenguajes de Descripción de Arquitectura permiten modelar tipos de datos complejos. Tal es el caso de AADL, que cuenta con un tipo de componente de datos para realizar tal modelado.

- Modelos de flujo de información: analizan el movimiento dinámico de la información entre elementos del sistema con el mundo exterior.

Notaciones: Aunque algunas de las notaciones más comunes para estos modelos son los diagramas de flujo de datos de Gane and Sarson (1979), algunos Lenguajes de Descripción de Arquitectura permiten modelar y especificar tales características. Tal es el caso de AADL.

- Modelos de ciclo de vida de información: analizan la forma en que los valores de la información cambian en el tiempo.

Notaciones: modelos de transición de estados (UML), o algún tipo de estructura de árbol. Además, algunos LDAs como AADL permiten definir características de tiempo.

Concurrencia

Describe la estructura de concurrencia del sistema y mapea elementos funcionales a unidades de concurrencia para identificar claramente las partes del sistema que pueden ejecutarse concurrentemente, y la forma en que esto es coordinado y controlado.

Aplicabilidad: todos los sistemas con hilos²⁷ de ejecución concurrentes.

²⁶ Retardo en la propagación de la información.

²⁷ Unidad de procesamiento más pequeña que puede ser planificada.

Intereses

- Estructura de tareas, es decir, estructura de procesos del sistema.
- Mapeo de elementos funcionales en tareas.
- Comunicación inter-procesos.
- Administración del estado de tiempo de ejecución.
- Sincronización e integridad entre actividades concurrentes.
- Inicio y cierre del sistema.
- Fallos en tareas concurrentes.
- Reentrada, es decir, la habilidad de un elemento de software para operar correctamente cuando funciona concurrentemente.

Tipos de modelos

- Modelos de concurrencia a nivel de sistema: mapea elementos funcionales en entidades de tiempo de ejecución. Este modelo típicamente contiene procesos del sistema operativo, grupos de procesos, y comunicación inter-procesos

Notaciones: las notaciones para estos modelos incluyen UML, notaciones formales especializadas como CSP, y notaciones informales. En particular, Wright, que es un LDA con características para especificar comportamiento arquitectónico y concurrencia, incluye la notación CSP para especificar características de ejecución tales como procesos y eventos. Por otra parte AADL, otro LDA, permite especificar hilos y procesos de ejecución, y características de comunicación.

- Modelos de estados: describen el conjunto de estados en que los elementos de tiempo de ejecución del sistema pueden encontrarse, y las transiciones válidas entre los estados. Puede contar con los siguientes tipos de entidades: estados, transiciones, eventos, acciones, y otras notaciones de modelado más sofisticadas.

Notaciones: incluyen UML, notaciones gráficas como redes de Petri, y notaciones no gráficas. En cuanto a los LDAs, Wright permite especificar eventos y procesos como parte del comportamiento arquitectónico.

Desarrollo

Describe la arquitectura que soporta el proceso de desarrollo de software. Las vistas de desarrollo comunican los aspectos de interés de la arquitectura a aquellos interesados involucrados en la construcción, pruebas, mantenimiento y mejora del sistema.

Aplicabilidad: todos los sistemas con desarrollo significativo de software.

Intereses

- Organización de los módulos del sistema.
- Procesamiento común.
- Estandarización de diseño y pruebas.
- Prácticas para insertar código especial que ayude a obtener información sobre la ejecución, estado del sistema, uso de recursos del sistema, etc.
- Organización de líneas de código.

Tipos de modelos

- Modelos de estructura de módulos: definen la organización del código fuente del sistema en términos de los módulos en los cuales los archivos fuente individuales son colectados y las dependencias entre estos módulos.

Notaciones: diagramas de componentes de UML y Lenguajes de Descripción de Arquitectura entre los que se encuentran AADL y Acme.

- Modelos de diseño común: permiten maximizar los aspectos comunes a través de implementaciones de elementos. Un modelo de diseño común tiene las siguientes tres piezas importantes: una definición de procesamiento común requerido a través de elementos, una definición de enfoques de diseño estándar que debería ser usado cuando se diseñan los elementos del sistema, una definición del software común que debe ser usado y cómo ser usado.

Notaciones: por lo general son documentos de diseño que pueden combinar texto y algún diagrama poco formal.

- Modelos de líneas de código: indican la manera en que el código es organizado en archivos fuente, cómo los archivos son agrupados en módulos, y las estructuras de directorios a utilizar.

Notaciones: notaciones estructuradas como UML, pero puede ser un enfoque basado en texto y tablas con diagramas claros para explicar las convenciones usadas. Además, algunos LDAs extensibles como xADL podrán permitir definir este tipo de modelos.

Despliegue o distribución

Describe el ambiente en el que el sistema va a ser desplegado (nodos, interconexiones de red, etc.), los requerimientos del ambiente técnico para cada elemento, y el mapeo de los elementos de software al ambiente de ejecución.

Aplicabilidad: sistemas con ambientes de despliegue complejos o no familiares.

Intereses

- Tipos de hardware requerido por el sistema, y el rol que cada uno desempeña.
- Especificación y cantidad de hardware requerido para desplegar el sistema.
- Requisitos de software de terceros.
- Compatibilidad de tecnologías.
- Requerimientos de red.
- Capacidad de red requerida.
- Restricciones de hardware.

Tipos de modelos

- Modelos de plataforma de ejecución: definen el conjunto de nodos de hardware que son requeridos, los nodos que requieren ser conectados a través de la red, y

los elementos de software que son alojados en los nodos de red. Este modelo tiene los siguientes elementos: nodos de procesamiento, nodos cliente, hardware de almacenamiento *online*, hardware de almacenamiento *offline*, conexiones de red, otros componentes de hardware especial, y mapeo de elementos de tiempo de ejecución a nodos.

Notaciones: aunque las notaciones para este tipo de modelos incluyen UML, diagramas de líneas y cajas tradicionales, y notaciones textuales, existen Lenguajes de Descripción de Arquitectura que permiten especificar formalmente y representar la plataforma de ejecución de un sistema. Tal es el caso de AADL, que cuenta con diversos componentes, como *buses*, memorias y procesadores, que permiten modelar el hardware en que se despliega un sistema.

- Modelos de red: modelos detallados para redes complejas. Son comúnmente dirigidos a especialistas en redes. Estos modelos especifican los nodos que requieren ser conectados, cualquier hardware de red específico (*firewalls*²⁸ o enrutadores), y los requerimientos de ancho de banda y propiedades de calidad requeridas para cada parte de la red. Los elementos principales del modelo son: nodos de procesamiento, nodos de red y conexiones de red.

Notaciones: aunque se puede usar UML y otras notaciones tradicionales de líneas y cajas, Lenguajes de Descripción de Arquitectura como AADL permiten especificar los dispositivos de una red, sus interfaces y las capacidades de red requeridas, tal es el caso de AADL.

- Modelos de dependencias tecnológicas: permiten modelar las dependencias en el ambiente de desarrollo o de pruebas.

Notaciones: puede ser un enfoque basado e texto, pero se pueden usar también notaciones gráficas.

Operacional

Describe cómo será operado, administrado y soportado el sistema cuando esté en su ambiente de producción.

Aplicabilidad: cualquier sistema que sea desplegado en un ambiente operacional crítico o complejo.

Intereses

- Instalación y mejora.
- Migración funcional, es decir, el proceso de reemplazar capacidades existentes, con las proporcionadas por el sistema.
- Migración de datos.
- Control y monitoreo operacional.

²⁸ Componente de red que actúa como un filtro de los paquetes de datos que entran o salen de una red (Tanenbaum y Wetherall 2011).

- Administración de la configuración.
- Monitoreo de la eficiencia de desempeño.
- Soporte, que incluye quién va a proporcionar el soporte, y los medios por los que va a ser proporcionado.
- Respaldo y restauración de la información.

Tipos de modelos

- Modelos de instalación: muestran cómo se instalará el sistema en su ambiente de producción. Pueden ayudar a entender lo que requiere ser instalado o mejorado en el sistema, las dependencias que existen entre los distintos grupos de elementos, las restricciones que existen en el proceso de instalación o actualización del sistema.

Notaciones: se pueden utilizar algunos Lenguajes de Descripción de Arquitectura, como AADL, para modelar parte del ambiente de producción del sistema.

- Modelos de migración: Ilustran estrategias de migración.

Notaciones: textos y tablas, notaciones gráficas, diagramas informales para ilustrar la migración y sincronización, y la migración compleja.

- Modelos de administración de configuración: plasman la reconfiguración regular compleja. El modelo debe explicar: los grupos de elementos de configuración en el sistema y la manera en que cada uno es manejado; las dependencias que existen entre los grupos de configuración; los diferentes conjuntos de valores de configuración requeridos por la rutina de operación del sistema; y los diferentes conjuntos de valores de configuración que serán aplicados al sistema.

Notaciones: aunque se pueden utilizar notaciones informales como textos y tablas, algunos LDAs como AADL permiten especificar características de operación del sistema. Por otra parte, xADL permite modelar características sobre administración de la configuración.

- Modelos de administración: permiten plasmar la información que resulta de la ejecución del sistema en su ambiente de producción. Define los requerimientos y restricciones operacionales de la arquitectura, y las facilidades que proporciona para administradores. El modelo debe definir: facilidades de control y monitoreo, procedimientos de rutina requeridos, probables condiciones de error, y facilidades para monitorear la eficiencia de desempeño.

Notaciones: aunque se pueden utilizar notaciones informales como textos y tablas, algunos LDAs como AADL permiten especificar características de operación del sistema y de tiempo de ejecución tales como la eficiencia de desempeño.

- Modelos de soporte: permiten plasmar el soporte que se requiere dar al sistema o las facilidades para usarlo u operarlo. Estos modelos deben proporcionar una abstracción clara del soporte que va a ser proporcionado, quién va a proporcionar el soporte, y la forma en que los problemas puedan ser escalados cuando se busque una solución. El modelo debe definir:

- Los grupos de interesados que requieren soporte, la naturaleza del soporte requerido y los mecanismos apropiados para entregar ese soporte.
- Clases de incidentes de soporte.
- Personal que proporciona el soporte y sus responsabilidades.
- Proceso de escalamiento.
- *Notaciones:* aunque se pueden utilizar notaciones informales como textos, tablas, y diagramas de flujo, algunos LDAs como AADL permiten especificar algunas características de operación del sistema.

Perspectivas

A continuación se presentan las perspectivas que pueden ser aplicadas a los puntos de vista antes mencionados, y que permiten incorporar atributos de calidad a ellos.

Seguridad

Se puede aplicar a cualquier sistema con interfaces accesibles públicamente, con múltiples usuarios, sistemas donde la identidad del usuario sea significativa, o donde el acceso a operaciones o información requiera ser controlado. La forma en que se aplica la perspectiva de seguridad a las diferentes vistas se muestra en la tabla C.1.

Vista	Aplicabilidad
Funcional	La vista funcional permite ver claramente cuáles de los elementos funcionales del sistema requieren ser protegidos.
Información	La vista de información ayuda a visualizar lo que debe ser protegido, en este caso, los datos vulnerables en el sistema.
Concurrencia	La vista de concurrencia define la manera en que los elementos funcionales son empaquetados en elementos de ejecución. El diseño de seguridad debe indicar el aislamiento de diferentes piezas del sistema en diferentes elementos de ejecución.
Desarrollo	Se deben identificar las guías o restricciones que los desarrolladores de software requieren para asegurar que la política de seguridad sea implementada.
Despliegue	El diseño de seguridad debe tener un mayor impacto en el ambiente de despliegue del sistema, tanto en el hardware como en el software.
Operacional	Implementar políticas de seguridad no es únicamente agregar características tecnológicas avanzadas. Debido a que el sistema es operado una vez que está en producción, tendrá mayor efecto en su seguridad. La vista operacional requiere hacer los supuestos de seguridad extremadamente claros, de tal forma que puedan ser reflejados en el proceso operacional.

Tabla C.1 Perspectiva de seguridad.

Eficiencia de desempeño y escalabilidad

La perspectiva puede aplicarse a cualquier sistema con requerimientos de eficiencia de desempeño (*performance*) complejos, no claros o ambiciosos; en sistemas cuya arquitectura incluya elementos con eficiencia de desempeño desconocida; y en sistemas

donde la expansión futura sea probablemente significativa. La forma en que se aplica la perspectiva de eficiencia de desempeño y escalabilidad a las diferentes vistas se muestra en la tabla C.2.

Vista	Aplicabilidad
Funcional	Aplicar esta perspectiva puede indicar la necesidad de cambios en la estructura funcional, con el objetivo de lograr los requerimientos de eficiencia de desempeño del sistema.
Información	La vista de información proporciona una entrada útil a modelos de eficiencia de desempeño, ya que identifica los recursos compartidos y los requerimientos transaccionales de cada uno. Considerar la escalabilidad puede sugerir la replicación o distribución de elementos en la vista de información.
Concurrencia	Aplicar esta perspectiva puede producir cambios en el diseño de concurrencia debido a la identificación de problemas tales como la contención excesiva de recursos. Alternativamente, considerar la escalabilidad y la eficiencia de desempeño puede resultar en favorecer la concurrencia de un elemento de diseño para satisfacer estos requerimientos. Los elementos de la vista de concurrencia pueden también proporcionar métricas de calibración para modelos de eficiencia de desempeño (<i>performance</i>).
Desarrollo	Al aplicar esta perspectiva se puede obtener un conjunto de guías relacionadas a la eficiencia de desempeño y la escalabilidad que podría ser seguido durante el desarrollo del software.
Despliegue	La vista de despliegue es entrada crucial para el proceso de eficiencia de desempeño y escalabilidad. Muchas partes de los modelos de eficiencia de desempeño del sistema son derivados del contenido de esta vista, lo que también proporciona métricas de calibración críticas. Aplicar esta perspectiva puede sugerir cambios y refinamientos en el ambiente de despliegue para permitir soportar las necesidades de eficiencia de desempeño y escalabilidad del sistema.
Operacional	La aplicación de esta perspectiva puede reflejar la necesidad de monitorear la eficiencia de desempeño y capacidades de administración del sistema.

Tabla C.2 Perspectiva de eficiencia de desempeño y escalabilidad.

Disponibilidad y flexibilidad

Se puede aplicar a cualquier sistema que tenga requerimientos de disponibilidad amplios o complejos, procesos de recuperación complejos, o un alto perfil. La forma en que se aplica la perspectiva de disponibilidad y flexibilidad a las diferentes vistas se muestra en la tabla C.3.

Vista	Aplicabilidad
Funcional	La disponibilidad es el interés clave de usuarios y adquiridores debido a puede impactar la forma en que opera el negocio. Los cambios funcionales pueden ser algunas veces requeridos para soportar requerimientos de disponibilidad.
Información	Una consideración clave de la disponibilidad, es el conjunto de procesos

	y sistemas que se requiere respaldar y recuperar. Los sistemas deben ser respaldados de tal forma que puedan ser recuperados en una cantidad de tiempo razonable si ocurre un desastre.
Concurrencia	Características como la replicación de hardware y la disponibilidad en el sistema, pueden implicar cambios o mejoras en el modelo de concurrencia.
Desarrollo	El enfoque para lograr la disponibilidad puede imponer restricciones de diseño en los módulos de software. Por ejemplo, todos los subsistemas podrían requerir el soporte de comandos de inicio, alto, pausa, y reinicio para alinearse con la estrategia de disponibilidad.
Despliegue	La disponibilidad y la flexibilidad pueden tener un gran impacto en el ambiente de despliegue. Los requerimientos de disponibilidad pueden requerir un ambiente de producción tolerante a fallas o un sitio de recuperación de desastres por separado, y que pueda ser rápidamente activado si el sitio de producción falla. También se puede requerir software especial para soportar el <i>clustering</i> ²⁹ de hardware.
Operacional	Pueden ser requeridos procesos y mecanismos que permiten la identificación y recuperación de problemas en el ambiente de producción. Pueden requerirse facilidades de recuperación de desastres en localidades por separado.

Tabla C.3 Perspectiva de disponibilidad y flexibilidad.

Evolución

Esta perspectiva puede ser importante para todos los sistemas, per más importante para los que tienen larga vida y para sistemas ampliamente utilizados. La forma en que se aplica la perspectiva de evolución a las diferentes vistas se muestra en la tabla C.4.

Vista	Aplicabilidad
Funcional	Si la evolución requerida es significativa, la estructura funcional requiere reflejarlo.
Información	Si se requiere la evolución del ambiente o información, un modelo de información flexible puede ser requerido.
Concurrencia	Las necesidades de evolución pueden requerir el empaquetamiento de elementos particulares, o alguna restricción en la estructura de concurrencia.
Desarrollo	Los requerimientos de evolución deben tener un impacto significativo en el ambiente de desarrollo que requiera ser definido.
Despliegue	Esta perspectiva raramente tiene un impacto significativo en la vista de despliegue porque la evolución del sistema usualmente afecta estructuras descritas en otras vistas.
Operacional	Esta perspectiva tiene poco impacto en la vista operacional.

Tabla C.4 Perspectiva de evolución.

²⁹ Agrupación de componentes de hardware para actuar como uno sólo.

Accesibilidad

Puede ser aplicada a cualquier sistema que pueda ser usado u operado por gente con discapacidades. La forma en que se aplica la perspectiva de accesibilidad a las diferentes vistas se muestra en la tabla C.5.

Vista	Aplicabilidad
Funcional	En teoría, la estructura funcional no debería ser realmente afectada por las consideraciones de accesibilidad.
Información	La estructura de información es improbablemente afectada, aunque puede ser necesario, por ejemplo, mantener información sobre discapacidades de clientes o usuarios.
Concurrencia	El impacto en esta vista es mínimo.
Desarrollo	La vista de desarrollo requiere crear conciencia en cuanto a la importancia de los problemas de accesibilidad.
Despliegue	El ambiente de despliegue es probablemente la vista más afectada por la accesibilidad, ya que se puede requerir hardware especial para soportar usuarios discapacitados.
Operacional	La vista operacional puede tener que tomar en cuenta las necesidades del equipo de soporte con discapacidades.

Tabla C.5 Perspectiva de accesibilidad.

Recursos de desarrollo

Puede ser aplicada a cualquier sistema para el cual el tiempo de desarrollo es limitado, o cuando se requiere hardware o software inusual o no familiar. La forma en que se aplica la perspectiva de recursos de desarrollo a las diferentes vistas se muestra en la tabla C.6.

Vista	Aplicabilidad
Funcional	Las restricciones de recursos, como cortos periodos de tiempo o limitaciones en habilidades, pueden imponer restricciones en funcionalidad.
Información	Los modelos de información particularmente sofisticados o complejos pueden requerir equipo especializado para su implementación.
Concurrencia	La implementación de arquitecturas concurrentes es usualmente compleja, y se requiere considerar el tiempo de desarrollo y pruebas, y las habilidades de los desarrolladores cuando se diseñe la arquitectura.
Desarrollo	Las restricciones de costos pueden limitar el desarrollo y los ambientes de pruebas disponibles.
Despliegue	Las restricciones de costo pueden limitar las opciones de despliegue del sistema.
Operacional	Se requiere tomar en cuenta las implicaciones de costo de la arquitectura de operación propuesta.

Tabla C.6 Perspectiva de recursos de desarrollo.

Internacionalización

Puede ser aplicado a cualquier sistema que requiera ser accedido por usuarios o equipo operacional de diferentes culturas o partes del mundo, o en múltiples lenguajes. La forma en que se aplica la perspectiva de internacionalización a las diferentes vistas se muestra en la tabla C.7.

Vista	Aplicabilidad
Funcional	La estructura funcional puede requerir reflejar la forma en la que la presentación es separada del contenido. La funcionalidad general debe ser independiente de la localización.
Información	La vista de información define la información almacenada que requiere ser internacionalizada y la forma en que esto va a ser logrado.
Concurrencia	Esta perspectiva tiene impacto mínimo en la vista de concurrencia.
Desarrollo	La vista de desarrollo puede requerir reflejar el impacto de estos factores en el ambiente de desarrollo. Por ejemplo, pueden requerirse datos de prueba internacionalizados.
Despliegue	El ambiente de despliegue puede requerir tomar en consideración los dispositivos de presentación. No olvidar que el hardware y software requieren soportar los lenguajes con los que se trabaja. La plataforma de despliegue en general, requiere ser internacionalizada.
Operacional	La vista operacional puede requerir considerar las funcionalidades proporcionadas para soportar el mantenimiento y administración de servicios e información, y la forma en que el soporte va a ser proporcionado para diferentes localidades.

Tabla C.7 Perspectiva de internacionalización.

Localización

Puede aplicarse a cualquier sistema cuyos elementos estén, o puedan estar físicamente lejos unos de otros. La forma en que se aplica la perspectiva de localización a las diferentes vistas se muestra en la tabla C.8.

Vista	Aplicabilidad
Funcional	La vista funcional es a veces presentada independientemente de la localidad del mundo real, típicamente esto se modela en la vista de despliegue.
Información	Si los datos están distribuidos, la vista de información debe describir como la información es mantenida y sincronizada, así como las latencias de actualización esperadas, la forma en que las discrepancias temporales son manejadas, y la forma en que la información es transformada entre las localidades.
Concurrencia	El procesamiento concurrente en diferentes localidades puede ser problemático por razones de seguridad y latencia. El enfoque de concurrencia seleccionado puede cambiar para ajustar las realidades de localización.
Desarrollo	Si el desarrollo del sistema es realizado en múltiples lugares, la vista de desarrollo requiere explicar la forma en que el software será manejado, integrado y probado.
Despliegue	La vista de despliegue debe considerar la forma en que los sistemas

	son físicamente desplegados en diferentes lugares de una manera sincronizada y controlada. Se deben considerar aspectos significativos como latencia y costos.
Operacional	La vista operacional requiere considerar la forma en que los sistemas ampliamente distribuidos son monitoreados, administrados y reparados.

Tabla C.8 Perspectiva de localización.

Regulación

Se puede aplicar a cualquier sistema que pueda ser objeto de leyes o regulaciones. La forma en que se aplica la perspectiva de regulación a las diferentes vistas se muestra en la tabla C.9.

Vista	Aplicabilidad
Funcional	Las regulaciones pueden tener un impacto significativo en lo que el sistema hace y como trabaja.
Información	El impacto en la vista de información puede incluir privacidad, control de acceso, retención y archivamiento, auditoria, disponibilidad y distribución de información.
Concurrencia	La perspectiva tiene poco impacto en esta vista.
Desarrollo	La perspectiva tiene poco impacto en esta vista.
Despliegue	La perspectiva tiene poco impacto en esta vista, aunque puede haber legislaciones para uso de hardware.
Operacional	La perspectiva tiene poco impacto en esta vista.

Tabla C.9 Perspectiva de regulación.

Usabilidad

Puede ser aplicada a cualquier sistema que tenga una interacción significativa con humanos, o sea expuesto al público. La forma en que se aplica la perspectiva de usabilidad a las diferentes vistas se muestra en la tabla C.10.

Vista	Aplicabilidad
Funcional	La estructura funcional indica donde están las interfaces externas del sistema, por lo que la usabilidad requiere ser considerada.
Información	La calidad de información puede tener un gran impacto en la usabilidad.
Concurrencia	La perspectiva tiene poco o nulo impacto en esta vista.
Desarrollo	Los resultados de aplicar la perspectiva de usabilidad impactan la vista de desarrollo en términos de guías, estándares, y patrones para asegurar la creación de un conjunto de interfaces de usuario para el sistema, que sean consistentes y apropiadas.
Despliegue	La perspectiva tiene impacto mínimo o nulo en la vista de despliegue.
Operacional	La perspectiva de usabilidad debe considerar las necesidades de usabilidad de los administradores del sistema.

Tabla C.10 Perspectiva de usabilidad.

Enfoque del SEI

Módulos

Describe los principales elementos del sistema que implementan un conjunto de responsabilidades.

Intereses

- Construcción del sistema e información de la estructura del código.
- Análisis de la forma en que los requerimientos funcionales son soportados por las responsabilidades de los módulos.
- Comunicación del sistema.

Tipos de modelo

El modelo del punto de vista modular incluye las siguientes características:

- Los elementos son los módulos, que son unidades de implementación de software que proporciona una coherente unidad de funcionalidad.
- Las relaciones entre elementos son: “parte de”, “depende de”, “es un”.
- Las propiedades de los elementos son: nombre, responsabilidades, información de implementación.

Notaciones

Se pueden utilizar notaciones informales, UML y algunos Lenguajes de Descripción de Arquitectura que permitan modelar arquitecturas de sistemas en términos de módulos. Uno de estos LDAs es xADL ya que cuenta con características de extensibilidad que pueden permitir modelar tales características.

Estilos del punto de vista

Descomposición, usos, generalización, y capas.

Componentes y conectores

Muestra los elementos que tienen alguna presencia en tiempo de ejecución, como objetos, clientes, servidores y almacenes de datos.

Intereses

- Almacenes de datos compartidos.
- Componentes de ejecución.
- Elementos del sistema replicados.
- Comportamiento de datos.
- Protocolos de interacción de comunicación.
- Elementos del sistema que corren en paralelo.

- Cambio de la estructura del sistema durante la ejecución.

Tipos de modelo

El modelo del punto de vista “componentes y conectores” incluye las siguientes características:

- Los elementos del modelo son los tipos de componentes, es decir, las principales unidades de procesamiento y almacenes de datos, y los tipos de conectores o mecanismos de interacción.
- Las relaciones entre los elementos están definidas por los roles de los conectores.
- Las propiedades de los elementos (componentes y conectores) son: nombre, tipo y otras propiedades dependiendo del tipo de componente o conector.

Notaciones

Incluye notaciones informales, UML, y sobre todo Lenguajes de Descripción de Arquitectura, ya que muchos de estos lenguajes permiten especificar y modelar una arquitectura en términos de componentes y conectores. Tal es el caso de AADL, Wright, Acme y xADL.

Estilos

Pipes and Filters, datos compartidos, publicar-suscribir, cliente-servidor, *peer-to-peer*, procesos de comunicación.

Distribución

Describe el mapeo de unidades de software a elementos del ambiente en el que el software es desarrollado o ejecutado.

Intereses

- Análisis de eficiencia de desempeño (*performance*), seguridad y disponibilidad.
- Volumen y frecuencia de comunicación entre los elementos de despliegue.
- Costo de despliegue del sistema.
- Mapeo de componentes y conectores en hardware.
- Mapeo de módulos del sistema en sistemas de archivos.
- Mapeo de módulos en personas, grupos o equipos.

Tipos de modelo

El modelo del punto de vista de distribución incluye las siguientes características:

- Elementos de software y elementos del ambiente.
- La relación principal entre los elementos es “localizado en”.

Notaciones

Aunque las notaciones para este tipo de modelos son informales, o UML, algunos Lenguajes de Descripción de Arquitectura permiten modelar y especificar la plataforma en que un sistema es desplegado. Tal es el caso de AADL, el cual además, permite agregar y analizar características de tiempo de ejecución, tales como eficiencia de desempeño (*performance*).

Estilos

Despliegue, implementación, asignación de trabajo.

Enfoque 4+1

Lógico

El sistema es descompuesto en un conjunto de abstracciones clave, tomadas del dominio del problema, en forma de objetos o clases de objetos. Éstos explotan los principios de la abstracción, encapsulación y herencia. Esta descomposición, además del análisis funcional, permite identificar mecanismos comunes y elementos de diseño a través de las partes del sistema.

Intereses

- Requerimientos funcionales.

Tipos de modelo

El modelo del punto de vista lógico tiene las siguientes características:

- Tiene como componentes objetos y clases de objetos.
- Los conectores son: asociación, herencia, contención.
- Los contenedores de elementos son las categorías de clases.

Notación

La notación por defecto son los diagramas de clases derivada de la notación de Booch (1993).

Estilos de la vista lógica

Estilo orientado a objetos.

Procesos

Toma en cuenta algunos requerimientos no funcionales, como la eficiencia de desempeño y la disponibilidad. Dirige asuntos de concurrencia y distribución, de integridad de sistema, de tolerancia a fallas, y la manera en que las principales abstracciones de la vista lógica se colocan dentro del proceso de arquitectura.

Intereses

- Eficiencia de desempeño (*performance*).
- Confiabilidad.
- Tolerancia a fallas.
- Integridad.

Tipos de modelo

El modelo del punto de vista de procesos tiene las siguientes características:

- Los componentes son tareas.
- Los conectores pueden incluir mensajes, *broadcasts*³⁰, o llamadas a procedimientos remotos (RPCs por sus siglas en inglés "*Remote Procedure Calls*").
- Los contenedores de elementos son los procesos.

Notación

Aunque la notación por defecto es expandida de la notación original propuesta por Booch (1993), se pueden utilizar Lenguajes de Descripción de Arquitectura como AADL, el cual permite definir características de tiempo de ejecución como la eficiencia de desempeño y la seguridad, además de permitir definir componentes, procesos e hilos de ejecución. Otro LDA que puede ser útil es Wright, que incluye la notación CSP para especificar el comportamiento arquitectónico, características de concurrencia mediante eventos y procesos.

Estilo para la vista de procesos

Múltiples estilos, entre los que se encuentran *Pipes and Filters* y cliente-servidor.

De desarrollo

Se enfoca en la organización modular de software en el ambiente de desarrollo de software. El software es empaquetado en pequeños trozos (bibliotecas de programa o subsistemas) que pueden ser desarrollados por una o un pequeño número de desarrolladores. Los subsistemas son organizados en una jerarquía de capas, donde cada capa proporciona interfaces bien definidas y estrechas para las capas sobre ella.

Intereses

- Organización.
- Reúso.
- Portabilidad.

³⁰ Forma de transmisión donde un paquete de información es enviado simultáneamente a todos los nodos de una red, los cuales lo reciben y procesan (Tanenbaum y Wetherall 2011).

- Líneas de productos.

Tipos de modelo

El modelo del punto de vista de desarrollo tiene las siguientes características:

- Los componentes son módulos y subsistemas.
- Los conectores son: dependencias, clausula “con”, clausula “incluye”.
- Los contenedores de componentes son subsistema (bibliotecas).

Notación

Una variación de la notación de Booch (1993), limitándose a los elementos arquitectónicamente significativos. Se puede extender algunos LDAs como xADL para modelar las características de este tipo de modelos o utilizar algún LDA que lo permita especificar.

Estilos

Estilos por capas.

Físico

Mapea el software en hardware. Toma en cuenta principalmente los requerimientos no funcionales del sistema, tales como disponibilidad, tolerancia a fallas o confiabilidad, eficiencia de desempeño (*performance*), y escalabilidad.

Intereses

- Escalabilidad.
- Eficiencia de desempeño.
- Disponibilidad.

Tipos de modelo

El modelo del punto de vista físico tiene las siguientes características:

- Los componentes son nodos.
- Los conectores son medios de comunicación, LAN, WAN, *buses*, etc.
- Los contenedores de componentes son subsistemas físicos.

Notación

Se pueden utilizar Lenguajes de Descripción de Arquitectura. Uno de los más recomendables es AADL, ya que permite especificar la plataforma de ejecución (hardware) de un sistema mediante una amplia variedad de componentes, y permite definir sus características de tiempo de ejecución tales como eficiencia de desempeño (*performance*) y disponibilidad.

Escenarios (el +1)

Además de los cuatro puntos de vista del enfoque 4+1, se deben incluir los escenarios que pueden ser representados mediante casos de uso utilizando UML.

D. Plantillas de documentación

El estándar ISO/IEC/IEEE 42010:2011 no prescribe la forma que toman las descripciones de arquitectura, éstas pueden tomar la forma de un documento, hipertexto, una colección de modelos, un repositorio, o algún otro medio. En este trabajo, se presenta un conjunto de plantillas que pueden utilizarse para generar una descripción de arquitectura mediante documentos.

Las plantillas presentadas son las siguientes:

- Plantilla 1 (descripción de arquitectura): es una plantilla para documentar descripciones de arquitectura que incluye lo siguiente:
 - Una plantilla para documentar los intereses e interesados de la arquitectura del sistema.
 - Una plantilla para documentar los puntos de vista arquitectónicos.
 - Una plantilla para documentar las vistas arquitectónicas.
 - Una plantilla para documentar las relaciones entre elementos arquitectónicos.
 - Una plantilla para documentar el razonamiento y las decisiones arquitectónicas.
 - Consideraciones para agregar información complementaria.
- Plantilla 2 (punto de vista): es una plantilla para documentar un punto de vista como producto de trabajo independiente a una descripción de arquitectura.

Las plantillas presentadas en este trabajo permiten elaborar descripciones de arquitectura según lo considerado en los capítulos 5 a 14.

La plantilla 1 se divide en 9 secciones, cada una de las cuales permite documentar una parte de una descripción de arquitectura. En la tabla D.1 se presenta una relación entre las secciones de la plantilla 1 (descripción de arquitectura), y los capítulos en que se explica su uso.

Sección de la plantilla 1 (descripción de arquitectura)	Capítulo
Sección 1: información complementaria	6 (sección 6.2)
Sección 2: sistema de interés	6 (sección 6.1)
Sección 3: intereses e interesados	7
Sección 4: puntos de vista arquitectónicos	9
Sección 5: vistas arquitectónicas	10 y 11
Sección 6: relaciones arquitectónicas	12
Sección 7: razonamiento y decisiones arquitectónicas	13
Sección 8: evaluación arquitectónica	6 (sección 6.3) y 14
Sección 9: información complementaria	6 (sección 6.2)

Tabla D.1 Relación entre secciones de la plantilla y capítulos.

La plantilla 2 (punto de vista) puede ser utilizada siguiendo el contenido del capítulo 9.

Plantilla 1: Descripción de arquitectura

Sección 1: información complementaria

[En esta sección se pueden incluir algunos de los siguientes elementos de información complementaria, según lo establezca la organización y/o proyecto:

- *Fecha de emisión y estado*
- *Autores*
- *Revisores*
- *Autoridad aprobatoria*
- *Organización que emite*
- *Historial de cambios*
- *Tabla de contenido*
- *Introducción*
- *Resumen*
- *Alcance*
- *Contexto del documento*
- *Información sobre control de versiones y administración de la configuración*
- *Uso del documento*
- *Notaciones utilizadas en el documento*

NOTA: En el capítulo 6 de este trabajo (sección 6.2) se aborda a detalle la documentación e identificación de la información complementaria de una descripción de arquitectura, por lo que es recomendable revisar dicho capítulo para el llenado adecuado de esta sección.]

Sección 2: sistema de interés

[En esta sección se debe incluir una pequeña descripción del sistema de interés, es decir, el sistema cuya arquitectura es documentada. Se puede incluir el nombre, objetivos, alcance o funciones principales del sistema. Además, se pueden incluir en la sección de referencias, las referencias a documentos con mayor información sobre el sistema.]

NOTA: En el capítulo 6 de este trabajo (sección 6.1) se aborda a detalle la documentación e identificación del sistema de interés en una descripción de arquitectura, por lo que es recomendable revisar dicho capítulo para el llenado adecuado de esta sección.]

Sección 3: intereses e interesados

[Para el adecuado llenado de esta sección se recomienda revisar el capítulo 7, que aborda a detalle la identificación y documentación de los intereses e interesados de la arquitectura de un sistema.]

3.1. Interesados

[Mencionar cuáles de los interesados propuestos por el estándar ISO/IEC/IEEE 42010:2011 (usuarios, operadores, adquiridores, propietarios, proveedores, desarrolladores, constructores y mantenedores) fueron considerados y cuáles no, y si se consideró un interesado no especificado por el estándar. Mencionar también si se siguió alguna técnica o método de identificación o priorización para obtener los interesados fundamentales para la arquitectura del sistema.]

[Se puede colocar en una tabla como la siguiente la lista de interesados relevantes para la arquitectura del sistema.]

Id	Nombre	Tipo	Rol	Intereses	Contacto

[Se pueden incluir referencias a documentos explícitos sobre los interesados, si es que éstos existen.]

3.2. Intereses

[Mencionar si se siguió alguna técnica o método para identificar algunos de los intereses considerados fundamentales para la arquitectura del sistema de interés.]

3.2.1. Propósitos del sistema

[Se puede colocar en una tabla como la siguiente la lista de los propósitos del sistema de interés que son considerados fundamentales para su arquitectura.]

Id	Propósito

[Citar, si existen, documentos o fuentes de donde se obtuvieron los intereses.]

3.2.2. Idoneidad o conveniencia de la arquitectura para lograr los propósitos del sistema

[Para documentar la idoneidad de la arquitectura para lograr los propósitos del sistema se puede utilizar alguna de las siguientes opciones:

Opción 1: mediante una tabla que exprese cómo la arquitectura logra los propósitos del sistema.]

Id	Propósito del sistema	Logrado (si/no)	Descripción

[Opción 2: citar en una sola sentencia si la arquitectura logra los propósitos del sistema.]

[Si es el caso, citar documentos o fuentes de donde se obtuvieron los intereses, tales como documentos de evaluación de la arquitectura.]

3.2.3. Viabilidad o factibilidad de construir y desplegar el sistema

[Estos intereses pueden ser presentados en forma de texto, o en una tabla como la siguiente, con sentencias que indiquen si es viable construir y desplegar el sistema.]

Id	Interés	¿Es viable?

3.2.4. Riesgos e impactos potenciales del sistema para sus interesados a lo largo de su ciclo de vida

[Los riesgos e impactos potenciales del sistema se pueden describir en forma de texto, o presentarse en las siguientes tablas.]

Riesgos:

Id	Riesgo	Evento	Condición	Consecuencia	Probabilidad	Impacto	Exposición	Contención	Contingencia

Impactos:

Identificador	Impacto	Fase	Naturaleza	Objeto	Magnitud

[Citar, si existen, documentos o fuentes de dónde se obtuvieron los intereses, tales como documentos de análisis de riesgos, el “Plan de proyecto” o algún “Plan de iteración”.]

3.2.5. Atributos de calidad

[Colocar en la siguiente tabla, los atributos de calidad del sistema que son considerados fundamentales para su arquitectura. Mencionar también si la mantenibilidad y la capacidad de evolución fueron considerados o no. El escenario para cada atributo de calidad es información opcional.]

Id	Atributo de calidad	Detalle	Escenario
			<i>Fuente del estímulo:</i> <i>Estímulo:</i> <i>Ambiente:</i> <i>Artefacto:</i> <i>Respuesta:</i> <i>Medida:</i>
			<i>Fuente del estímulo:</i> <i>Estímulo:</i> <i>Ambiente:</i> <i>Artefacto:</i> <i>Respuesta:</i> <i>Medida:</i>

[Citar, si existen, documentos o fuentes de dónde se obtuvieron los intereses.]

3.2.6. Otros intereses

[Colocar en la siguiente tabla, aquellos intereses fundamentales para la arquitectura del sistema de interés que no hayan sido considerados en clasificaciones anteriores. Justificar porque son considerados y agregar su descripción.]

Id	Descripción del interés

[Citar, si existen, documentos o fuentes de dónde se obtuvieron los intereses.]

3.3. Relación entre interesados e intereses

[Colocar en una tabla como la siguiente, los identificadores o nombres de los intereses (columnas) e interesados (filas), y marcar la relación entre ellos (con una "X", por ejemplo).]

	Intereses				
Interesados					

Sección 4: puntos de vista arquitectónicos

[Para el adecuado llenado de esta sección se recomienda revisar el capítulo 9, que aborda a detalle la identificación y documentación de los puntos de vista de la arquitectura de un sistema.]

Para cada punto de vista de la arquitectura incluir las siguientes secciones.]

[Nombre del punto de vista]

4.1. Intereses

[Se puede incluir una descripción de los intereses que enmarca el punto de vista, o colocarlos en una tabla como la siguiente.]

Interés	Descripción

4.2. Interesados

[Se puede colocar en una tabla como la siguiente, la relación entre los interesados del sistema y los intereses que enmarca el punto de vista.]

Interesado	Intereses

4.3. Tipos de modelo

[En esta sección se deben incluir los tipos de modelo usados por el punto de vista para expresar los intereses que enmarca. Se puede emplear alguna de las siguientes cuatro opciones.]

Opción 1: Metamodelo

[Incluir un metamodelo con los siguientes elementos:

- *Entidades: los principales tipos de elementos que están presentes en los modelos.*
- *Atributos: propiedades que poseen las entidades en los modelos.*
- *Las relaciones que son definidas entre entidades en los modelos.*
- *Tipos de restricciones que hay en las entidades, atributos o relaciones en los modelos.*

Dichos elementos pueden ser definidos explícitamente además de ser incorporados como parte del metamodelo.]

Opción 2: Plantillas

[Se puede proporcionar una plantilla para especificar el formato y/o contenido de modelos.]

Opción 3: Definición del lenguaje

[Incluir información, o referencias a la misma, sobre las notaciones, elementos, sintaxis, semántica, herramienta de soporte y restricciones del lenguaje.]

Opción 4: Operaciones

[Definir las operaciones disponibles en los modelos. Se puede incluir los métodos analíticos soportados por el tipo de modelo, o las técnicas para generar, o interpretar los modelos.]

4.4. Referencias

[Incluir referencias a fuentes de información que pueden ayudar a entender el punto de vista.]

Sección 5: vistas arquitectónicas

[Para el adecuado llenado de esta sección se recomienda revisar el capítulo 10, que aborda a detalle el desarrollo y documentación de las vistas arquitectónicas de un sistema.

[Para cada vista arquitectónica incluir las siguientes secciones.]

[Nombre de la vista]

[El nombre de la vista puede ser el mismo que el nombre del punto de vista que la gobierna.

5.1. [Información complementaria]

[En esta sección se puede agregar la información complementaria que especifique la organización y/o proyecto. Puede incluir:

- *Resumen: breve extracto del contenido de la vista arquitectónica.*
- *Alcance: especificación de los límites de la vista arquitectónica, aclarando lo que incluye, y lo que no.*
- *Uso: incluir cualquier información de utilidad para los interesados que hagan uso de la vista. Se pueden incluir los interesados para los que fue elaborada la vista arquitectónica, una relación de los modelos que pueden ser de interés para cada interesado o tipo de interesado, y las secciones que pueden ser consultadas primero y que permitan entender claramente la vista arquitectónica.]*

5.2. Punto de vista

[Incluir el nombre y las referencias de la definición del punto de vista que gobierna a la vista.]

5.3. Modelos arquitectónicos

[Para cada modelo de la vista, se debe incluir la siguiente información considerando de lo que especifique la organización y/o proyecto.]

[Nombre del modelo y versión]

[Incluir el nombre y la versión que identifica al modelo según lo especifique la organización y/o proyecto.]

[Información complementaria]

[En esta sección se puede agregar información complementaria sobre los modelos, la cual puede incluir:

- *Propósito del modelo.*
- *Los términos que puedan ser de utilidad para entender el modelo, tales como términos técnicos o relacionados a las notaciones utilizadas. Se puede usar una tabla como la siguiente:*

Término	Descripción

- *Elementos: se pueden colocar en una tabla como la siguiente, los elementos del modelo con la información que se crea conveniente (componentes, interfaces, conectores, etc.).*

Catálogo de elementos	
Elementos	
<i>Elemento</i>	<i>Descripción</i>
Interfaces	
<i>Interfaz</i>	<i>Descripción</i>

- *Comportamiento: se puede incluir el comportamiento de los elementos que no sea tan obvio, como tipos de comunicación y restricciones.]*

5.3.1. Tipo de modelo

[Identificar el tipo de modelo que gobierna al modelo. Puede ser una referencia al tipo de modelo identificado en el punto de vista que gobierna a la vista.]

5.3.2. Diagramas y código

[Colocar los diagramas o código de cada modelo, o las referencias a los mismos si es que se encuentran en algún tipo de repositorio.]

5.4. Cuestiones con respecto al punto de vista

5.4.1. Asuntos sin resolver

[Incluir cualquier asunto relacionado con la elaboración de las vistas arquitectónicas que no se resolvió.]

5.4.2. Conflictos conocidos

[Incluir los problemas que resultaron de la elaboración de las vistas arquitectónicas.]

5.4.3. Excepciones y desviaciones de las convenciones

[Incluir las excepciones y desviaciones de las convenciones usadas para la creación de la vista arquitectónica.]

Sección 6: relaciones arquitectónicas

[Para el adecuado llenado de esta sección se recomienda revisar el capítulo 12, que aborda a detalle la documentación de las relaciones entre elementos de una descripción de arquitectura.]

6.1. Análisis de consistencia

[Para documentar las inconsistencias encontradas, se propone incluir:

- Los modelos o vistas donde fueron encontradas las inconsistencias.*
- El tipo de inconsistencia encontrada (correspondencia, refinamiento, de requerimientos no funcionales).*
- Solución a las inconsistencias encontradas y razón para solucionarlas o no.*

Tal información se puede incluir una tabla como la siguiente:]

Modelo o vista	Tipo de inconsistencia	Solución

6.2. Reglas de correspondencia

[Para cada regla de correspondencia identificada, se debe registrar

- Si la regla de correspondencia se cumple o no.
- Todas las violaciones a la regla, si existen.

Para documentar la información anterior se puede hacer uso de una tabla como la siguiente.]

Regla de correspondencia	Se cumple (si/no)	Violaciones

6.3. Correspondencias

[Las correspondencias deben ser documentadas incluyendo:

- La identificación de cada correspondencia.
- La identificación de los elementos participantes en cada correspondencia.
- La identificación de la reglas de correspondencia que gobierna cada correspondencia (si es el caso).

Se puede hacer uso de una tabla como la siguiente:]

Regla de correspondencia:	
Correspondencia:	
Elemento 1	Elemento 2

Sección 7: razonamiento y decisiones arquitectónicas

[Para el adecuado llenado de esta sección se recomienda revisar el capítulo 13, que aborda a detalle la documentación del razonamiento y las decisiones arquitectónicas en una descripción de arquitectura.]

7.1. Razonamiento

[Registrar en las siguientes secciones el razonamiento relacionado a los puntos de vista y decisiones arquitectónicas clave, y las evidencias de dichos razonamientos.]

7.1.1. Razonamiento sobre puntos de vista

[El razonamiento sobre los puntos de vista se puede documentar en forma de texto, o registrarlo en una tabla como la siguiente:]

Id	Punto de vista	Razonamiento

7.1.2. Razonamiento sobre decisiones clave de la arquitectura

[El razonamiento sobre las decisiones de arquitectura se puede documentar en forma de texto, o registrarlo en una tabla como la siguiente:]

Id	Decisión	Razonamiento

7.1.3. Evidencias

[Registrar en forma de texto, o por medio de una lista como la siguiente, las evidencias del razonamiento hecho para puntos de vista y decisiones clave:

- Evidencia o fuente de la misma
- ...
- ...
- ...
- ...]

7.2. Decisiones arquitectónicas

[Cada una de las decisiones arquitectónicas se puede registrar en forma de texto o en una tabla como la siguiente.]

Identificador (Id):			
Decisión	[Breve descripción de la decisión]		
Intereses (Ids)	[intereses involucrados]		
Propietario (Ids)	[interesados involucrados]	Razonamiento (id)	
Elementos de la DA			
Restricciones y supuestos		<ul style="list-style-type: none">•••	

Alternativas consideradas	<ul style="list-style-type: none"> • • • 		
Consecuencias			
Toma de la decisión			
Fecha	[DD/MM/AAAA]	Hora	[HH:MM]
Aprobación			
Fecha	[DD/MM/AAAA]	Hora	[HH:MM]
Último cambio			
Fecha	[DD/MM/AAAA]	Hora	[HH:MM]
Información adicional	<ul style="list-style-type: none"> • • • 		

Sección 8: evaluación arquitectónica

[Para el adecuado llenado de esta sección se recomienda revisar el capítulo 6 (sección 6.3), que aborda a detalle la documentación de las evaluaciones arquitectónicas.

En esta sección se deben incluir los resultados de las evaluaciones de la arquitectura o de la descripción de arquitectura.

La información sobre la evaluación de la arquitectura puede ser presentada de las siguientes formas:

- Respondiendo a la pregunta: ¿La arquitectura plasmada en la descripción de arquitectura es adecuada para el sistema al que representa?
- Una lista de verificación que permita identificar los requerimientos o intereses de los interesados del sistema que son cubiertos por la arquitectura del mismo:

Interés	Cubierto por la arquitectura (Si / No)

- Indicar si el sistema es viable, es decir, si puede ser construido con los recursos disponibles.
- Referencias a reportes o documentos más detallados sobre resultados de validaciones o evaluaciones de la arquitectura. Estas referencias pueden ser incluidas en la sección de referencias de la descripción de arquitectura.

La información sobre la evaluación de la descripción de arquitectura puede ser presentada a través de listas de verificación. Por ejemplo se puede usar la siguiente que valida que la descripción de arquitectura contenga lo especificado por el estándar ISO/IEC/IEEE 42010:2011:]

Sección de la descripción de arquitectura	Contenido de la sección	Cubierto en la descripción de arquitectura (si o no)
Información general	Información sobre el sistema de interés.	
	Información complementaria (referencias, glosarios, etc.).	
	Resultados de evaluaciones de la arquitectura y de la descripción de arquitectura.	
Interesados	Interesados clave del sistema.	
Intereses	Intereses clave del sistema.	
Relación entre interesados e intereses	Relación entre interesados e intereses.	
Puntos de vista	Intereses enmarcados por el punto de vista.	
	Los interesados típicos para los intereses enmarcados por el punto de vista.	
	Tipos de modelo usados por el punto de vista.	
	Para cada tipo de modelo identificado, los lenguajes, notaciones, convenciones, técnicas de modelado, métodos analíticos y otras operaciones que pueden ser usadas en los modelos de ese tipo.	
	Referencias a fuentes de información.	
Vistas arquitectónicas	Información complementaria que es especificada por la organización y/o proyecto.	
	Punto de vista que la gobierna.	
	Modelos arquitectónicos.	
	Asuntos conocidos en la vista con respecto al punto de vista que la gobierna.	
Reglas de correspondencia	Análisis de consistencia de las vistas y modelos arquitectónicos.	
	Correspondencias entre los elementos de la descripción de arquitectura.	
	Reglas de correspondencia aplicables a los elementos de la descripción de arquitectura.	
Razonamiento y decisiones arquitectónicas	Razonamiento de las decisiones tomadas.	
	Decisiones tomadas.	

[También se puede incluir respuesta a las siguientes preguntas que plantea Clements et al. (2003).

- *¿La descripción de arquitectura es consistente con los interesados que la usarán?*
 - *¿Se Identificó el conjunto correcto de interesados clave y sus intereses?*
 - *¿Se seleccionó el conjunto correcto de vistas y puntos de vista?*
 - *¿Se incluyó la información adecuada sobre su organización para encontrar fácilmente cierta información?*
- *¿La descripción de arquitectura es consistente consigo misma? Lo que significa estar libre de errores de ambigüedad y contradicción.*
 - *¿Se incluyeron mapeos entre vistas?*
 - *¿Se Incluyeron las inconsistencias entre las vistas?*
 - *¿Se incluyeron las afirmaciones tales como hechos, heurísticas, requerimientos, decisiones, etc.?*
 - *¿Se aseguró que no existan dos términos que signifiquen lo mismo o que un término se use para dos cosas diferentes?*
 - *¿Se aseguró que no se repitiera información innecesaria en lugares diferentes en una descripción de arquitectura?*
- *¿La descripción de arquitectura es consistente con una buena forma?*
 - *¿La estructura y secciones de la documentación están definidas a través de plantillas o un estándar?*
 - *Para cada vista, ¿se proporcionó su definición?, es decir, elementos, relaciones, y propiedades.*
 - *¿Contiene razonamiento, restricciones, entorno y alternativas rechazadas que puedan ser útiles?*
 - *¿Incluye rastros de requerimientos incluidos?*
 - *¿Incluye como ejercer variabilidades?*
- *¿La descripción de arquitectura es consistente con la arquitectura que describe?*
 - *¿La descripción de arquitectura es precisa?*
 - *¿La descripción de arquitectura está actualizada?*
 - *¿La descripción de arquitectura está completa?]*

Sección 9: información complementaria

[En esta sección se pueden incluir algunos de los siguientes elementos, de acuerdo a lo que establezca la organización y/o proyecto:

- *Glosario*
- *Referencias*
- *Apéndices*

NOTA: En el capítulo 6 de este trabajo (sección 6.2) se aborda a detalle la documentación e identificación de la información complementaria de una descripción de arquitectura, por lo que es recomendable revisar dicho capítulo para el llenado adecuado de esta sección.]

Plantilla 2: Punto de vista

[Nombre del punto de vista]

[Para el adecuado llenado de esta plantilla se recomienda revisar el capítulo 9, que aborda a detalle la identificación y documentación de los puntos de vista de la arquitectura de un sistema.]

1. Resumen

[El resumen puede contener una breve visión del punto de vista, e incluir sus principales características (intereses que enmarca, interesados típicos, y los sistemas a los que puede ser aplicado).]

2. Intereses

[Se puede incluir una descripción de los intereses que enmarca el punto de vista, o colocarlos en una tabla como la siguiente.]

Interés	Descripción

3. Anti-intereses

[Para los anti-intereses, se puede incluir la razón por la que no son convenientes para el punto de vista y se puede colocarlos en una tabla como la siguiente.]

Anti- interés	Descripción

4. Interesados

[Se puede colocar en la siguiente tabla los interesados a quienes pueden pertenecer los intereses que enmarca el punto de vista.]

Interesado	Intereses

5. Tipos de modelo

[En esta sección se deben incluir los tipos de modelo usados por el punto de vista para expresar los intereses que enmarca. Se puede emplear alguna de las siguientes cuatro opciones.]

Opción 1: Metamodelo

[Incluir un metamodelo con los siguientes elementos:

- *Entidades: los principales tipos de elementos que están presentes en los modelos.*
- *Atributos: propiedades que poseen las entidades en los modelos.*
- *Las relaciones que son definidas entre entidades en los modelos.*
- *Tipos de restricciones que hay en las entidades, atributos o relaciones en los modelos.*

Dichos elementos pueden ser definidos explícitamente además de ser incorporados como parte del metamodelo.]

Opción 2: Plantillas

[Se puede proporcionar una plantilla para especificar el formato y/o contenido de modelos.]

Opción 3: Definición del lenguaje

[Incluir información, o referencias a la misma, sobre las notaciones, elementos, sintaxis, semántica, herramienta de soporte y restricciones del lenguaje.]

Opción 4: Operaciones

[Definir las operaciones disponibles en los modelos. Se puede incluir los métodos analíticos soportados por el tipo de modelo, o las técnicas para generar, o interpretar los modelos.]

6. Reglas de correspondencia

[Colocar en la siguiente tabla las reglas de correspondencias que relacionan elementos del punto de vista.]

Elemento 1	Elemento 2	Correspondencia

[Colocar en la siguiente tabla las reglas de correspondencias que relacionan elementos de los tipos de modelo del punto de vista.]

Elemento 1	Elemento 2	Correspondencia

7. Operaciones entre vistas

[Colocar la descripción de las operaciones aplicados a las vistas o a los modelos generados a partir de un punto de vista. Pueden incluir métodos de creación, interpretativos, de análisis, diseño e implementación.]

Operación	Tipo	Descripción

8. Ejemplos

[Colocar los ejemplos que serán de ayuda al lector para el uso del punto de vista.]

9. Notas

[Incluir información adicional que puede ser de utilidad para el lector.]

10. Referencias

[Incluir referencias a fuentes de información que pueden ayudar a entender el punto de vista.]

E. Ejemplo de una descripción de arquitectura

Descripción de arquitectura

Nombre: MonitoreoInformación/SysMonitor/DA_V1.0

Fecha de emisión: 12/10/2004

Estado: concluida

Autor: Sandra Ramírez (sramirez@unam.mx)

Organización que emite: UNAM

Introducción

Este documento contiene la descripción arquitectónica del sistema de monitoreo “SysMonitor” que tiene como objetivo apoyar en la disponibilidad de sistemas de información.

Información sobre control de versiones y administración de la configuración

- Repositorio: “Dropbox/SysMonitoreo/DAs/”.
- Mecanismo de nombrado para las descripciones de arquitectura: *nombreproyecto/nombre sistema/“DA”_ VN.N*

1. Sistema de interés

El sistema tiene como objetivo monitorear características de la eficiencia de desempeño de componentes en un sistema tecnológico con dos módulos para detectar posibles fallas o estados que afecten su disponibilidad.

El sistema de monitoreo que se presenta, debe recabar datos sobre la eficiencia de desempeño de dos componentes que lo conforman utilizando programas “proveedores de datos”. Dichos programas entregan los datos a un programa “monitor” que se encarga de reunir, analizar y administrar la información, y de generar alertas sobre posibles fallas o estados que afecten la disponibilidad del sistema. La información generada por el monitor es entregada a un programa “consumidor de datos”, encargado de desplegar la información para su análisis, visualización, generación de reportes. A continuación, se describe a detalle los elementos que conforman la arquitectura del sistema:

- Proveedores de datos: recaban datos sobre la eficiencia de desempeño de un componente individual del sistema del que forman parte, y los entregan al “monitor”. Los datos son manejados por medio de objetos de datos, que cuentan con atributos que representan datos específicos de la eficiencia de desempeño.

- Monitor: proporciona una infraestructura para reunir y administrar la información del sistema. Compara constantemente los valores reportados por los proveedores de datos sobre los componentes monitoreados, con los valores de ciertos umbrales, y despliega una alerta si un valor excede o está por debajo de dichos umbrales. La información sobre las alertas es transferida a consumidores de datos.
- Consumidor de datos: programa que lee los datos proporcionados por el monitor, y presenta (despliega) la información en forma de alertas relacionadas a posibles fallas producidas o propensas a producirse en el sistema. Permite generar reportes confiables y rápidos sobre errores percibidos, y sobre valores que exceden o caen debajo de un valor de umbral particular.

2. Intereses e interesados

2.1. Interesados

De los interesados propuestos por el estándar ISO/IEC/IEEE 42010:2011, se consideran operadores, desarrolladores, personal de mantenimiento. Además se considera el personal de pruebas y arquitectos. La técnica que se siguió para su identificación fue una lluvia de ideas realizada con el equipo de desarrollo del sistema.

Id	Nombre	Tipo	Rol	Contacto
S01	Operador	Interno	Operador	adop@unisys.com
S02	Desarrolladores del sistema	Interno	Desarrollador	addev@unisys.com
S03	Área de mantenimiento	Interno	Mantenimiento	admai@unisys.com
S04	Área de pruebas	Externo	Pruebas	admin@prinfo.com
S05	Arquitecto	Interno	Arquitecto	ararch@prinfo.com

2.2. Intereses

2.2.1. Propósitos del sistema

Id	Propósito
C01	Monitorear características de la eficiencia de desempeño de componentes en un sistema tecnológico para detectar posibles fallas o estados que afecten su disponibilidad.

2.2.2. Idoneidad o conveniencia de la arquitectura para lograr los propósitos del sistema

Id	Propósito del sistema	Logrado (si/no)	Descripción
C02	C01	Si	Arquitectura de componentes y conectores

2.2.3. Viabilidad o factibilidad de construir y desplegar el sistema

Id	Interés	¿Es viable?
C03	Construcción del sistema	Si
C04	Despliegue del sistema	Si

2.2.4. Otros intereses

Id	Detalle
C05	Los módulos principales del sistema son dos sistemas proveedores de datos, sistema monitor, y sistema consumidor de datos.

2.3. Relación entre interesados e intereses

	Intereses							
	C01	C02	C03	C04	C05			
Interesados	S01	X	X					
	S02	X	X	X	X	X		
	S03	X	X					
	S04	X	X			X		
	S05	X	X	X	X	X		

3. Puntos de vista arquitectónicos

Punto de vista PV01 “componentes y conectores”

3.1. Intereses

Interés	Descripción
C01	Ver sección 2.2
C02	Ver sección 2.2
C03	Ver sección 2.2
C05	Ver sección 2.2
C05	Ver sección 2.2

3.2. Interesados

Interesado	Intereses
S01	C01, C02
S02	C01, C02, C03, C04, C05
S03	C01 C02
S04	C01, C02, C05
S05	C01, C02, C03, C04, C05

3.3. Tipos de modelo

El tipo de modelo son las notaciones gráficas del lenguaje Acme (Garlan, Monroe y Wile 1997).

3.4. Referencias

Ver punto de vista de componentes y conectores del SEI (Clements et al 2003).

4. Vistas arquitectónicas

Vista de componentes y conectores

Contiene las principales estructuras del sistema, muestra cómo se cumplen los objetivos del sistema por medio de su arquitectura, y la viabilidad para ser desarrollado y desplegado.

Esta vista es útil para arquitectos, desarrolladores, personal de mantenimiento, personal de pruebas, y operadores.

4.1. Punto de vista

El punto de vista en que se basa la vista es el de componentes y conectores.

4.2. Modelos arquitectónicos

Modelo de componentes y conectores versión 1.0

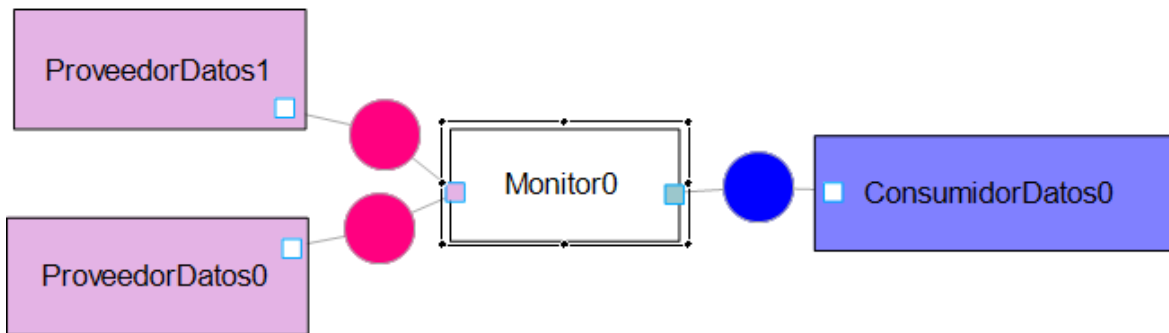
Propósito del modelo: mostrar los principales componentes del sistema, la forma en que se cumplen los objetivos del sistema por medio de su arquitectura, y la viabilidad para ser desarrollado y desplegado.

Catálogo de elementos	
Elemento	Descripción
ProveedorDatos0	Proporciona datos de un componente del sistema monitoreado.
ProveedorDatos1	Proporciona datos de un componente del sistema monitoreado.
Monitor0	Monitorea los datos proporcionados por los componentes proveedores de datos.
ConsumidorDatos0	Despliega el resultado del monitoreo.

4.2.1. Tipo de modelo

Se utiliza la notación gráfica del lenguaje Acme (Garlan, Monroe y Wile 1997).

4.2.2. Diagrama



(El código del diagrama anterior se puede encontrar en la sección 11.2.4.4 de la guía.)

4.3. Cuestiones con respecto al punto de vista

3.6.1. Asuntos sin resolver

4.3.1. Conflictos conocidos

No fueron identificados problemas.

4.3.2. Excepciones y desviaciones de las convenciones

No se identificaron excepciones y desviaciones de las convenciones usadas para la creación de la vista arquitectónica.

5. Relaciones arquitectónicas

[Solo se presentan algunos ejemplos de las relaciones asociadas a los modelos de la arquitectura para fines de simplicidad del ejemplo en este trabajo.]

5.1. Análisis de consistencia

Debido a que la descripción de arquitectura cuenta con un a única vista y modelo, no se encontraron inconsistencias entre vistas o modelos arquitectónicos.

5.2. Reglas de correspondencia

Regla de correspondencia	Se cumple (si/no)	Violaciones
R01: en un modelo, cada elemento “proveedor de datos” debe interactuar con el componente “monitor”	SI	Ninguna
R02: en un modelo, el componente “monitor” debe interactuar con el componente “consumidor de datos”	SI	Ninguna

5.3. Correspondencias

A continuación se presentan algunas correspondencias derivadas de las reglas de correspondencias (sección 5.2) del modelo arquitectónico presentado en la sección 4.2.

Regla de correspondencia: *“cada elemento proveedor de datos debe interactuar con el monitor”*

Correspondencia: R01

Elemento 1	Elemento 2
ProveedorDatos0	Monitor0
ProveedorDatos1	Monitor0

Regla de correspondencia: *“el monitor debe interactuar con el consumidor de datos”*

Correspondencia: R02

Elemento 1	Elemento 2
Monitor0	ConsumidorDatos0

6. Razonamiento y decisiones arquitectónicas

6.1. Razonamiento

6.1.1. Razonamiento sobre puntos de vista

Id	Punto de vista	Razonamiento
R01	PV01	Debido a que los intereses e interesados del sistema son pocos, y tales intereses pueden ser cubiertos con un punto de vista del tipo “componentes y conectores” del SEI (Clements et al. 2003).

6.1.2. Razonamiento sobre decisiones clave de la arquitectura

Id	Decisión	Razonamiento
D01	Selección de un punto de vista tipo “componentes y conectores”.	Enmarca todos los intereses del sistema.
D02	Selección de Acme como LDA.	Cuenta con soporte para modelar los intereses del sistema y cuenta con una notación gráfica sencilla.
D03	Identificación de interesados clave.	Son los interesados que resultaron de mayor relevancia para el sistema después de realizar una lluvia de ideas.

6.1.3. Evidencias

Las evidencias de los razonamientos antes presentados son:

- Minuta del 01/02/2004 sobre reunión inicial del proyecto.
- Documento de requerimientos REQ/SYS/MON01/V1.
- Minuta de reunión entre arquitectos del 06/04/2004

6.2. Decisiones arquitectónicas

(Se presenta el ejemplo de una única decisión arquitectónica para fines de simplicidad en este trabajo.)

Identificador (Id):	D01		
Decisión	<i>Selección de punto de vista</i>		
Intereses (Ids)	<i>C01, C02, C03, C04, C05</i>		
Propietario (Ids)	<i>S01, S02, S03, S04, S05]</i>	Razonamiento (id)	R01
Elementos de la DA	Intereses, interesados y punto de vista.		
Restricciones supuestos	y	-----	
Alternativas consideradas	Punto de vista modular		
Consecuencias			

Toma de la decisión			
Fecha	06/04/2004	Hora	14:00
Aprobación			
Fecha	06/04/2004	Hora	17:00
Último cambio			
Fecha	06/04/2004	Hora	17:00
Información adicional	-----		

7. Evaluación arquitectónica

La arquitectura plasmada en la descripción de arquitectura es adecuada para el sistema al que representa.

Interés	Cubierto por la arquitectura (Si / No)
C01	SI
C02	SI
C03	SI
C04	SI
C05	SI

La descripción de arquitectura contiene lo especificado por el estándar ISO/IEC/IEEE 42010:2011:

Sección de la descripción de arquitectura	Contenido de la sección	Cubierto en la descripción de arquitectura (si o no)
Información general	Información sobre el sistema de interés.	SI
	Información complementaria (referencias, glosarios, etc.).	SI
	Resultados de evaluaciones de la arquitectura y de la descripción de arquitectura.	SI
Interesados	Interesados clave del sistema.	SI
Intereses	Intereses clave del sistema.	SI
Relación entre interesados e intereses	Relación entre interesados e intereses.	SI
Puntos de vista	Intereses enmarcados por el punto de vista.	SI
	Los interesados típicos para los intereses enmarcados por el punto de vista.	SI
	Tipos de modelo usados por el punto de vista.	SI
	Para cada tipo de modelo identificado, los lenguajes, notaciones, convenciones, técnicas de modelado, métodos analíticos y	SI

	otras operaciones que pueden ser usadas en los modelos de ese tipo.	
	Referencias a fuentes de información.	SI
Vistas arquitectónicas	Información complementaria que es especificada por la organización y/o proyecto.	SI
	Punto de vista que la gobierna.	SI
	Modelos arquitectónicos.	SI
	Asuntos conocidos en la vista con respecto al punto de vista que la gobierna.	SI
Reglas de correspondencia	Análisis de consistencia de las vistas y modelos arquitectónicos.	SI
	Correspondencias entre los elementos de la descripción de arquitectura.	SI
	Reglas de correspondencia aplicables a los elementos de la descripción de arquitectura.	SI
Razonamiento y decisiones arquitectónicas	Razonamiento de las decisiones tomadas.	SI
	Decisiones tomadas.	SI

Bibliografía

Alberts, C., Dorofee, A. (2012). *Mission Risk Diagnostic (MRD) Method Description* (Nota técnica). CMU/SEI-2012-TN-005.

Ali, N., Millan, C., Ramos, C. (2006). *Developing Mobile Ambients Using an Aspect-Oriented Software Architectural Model*. LNCS 4276, 1633-1649.

Ali, N., Solís, C., Ramos, I. (2008). *Comparing Architecture Description Languages for Mobile Software Systems*. Polytechnic University of Valencia.

Allen Robert J. (1997). *A formal Approach to Software Architecture* (Tesis doctoral). Carnegie Mellon University. Pittsburgh, PA 15213.

Allen, R., Douence, R., y Garlan, D. (1998) *Specifying and Analyzing Dynamic Software Architectures*. Proceedings of 1998 Conference on Fundamental Approaches to Software Engineering Lisbon, Portugal.

ANSI/IEEE 1471:2000. *IEEE Recommended Practice for Architectural Description for Software-Intensive Systems*.2000.

Avesani, P., Bazzanella, C., Perini, A., Susi, A. (2005). *Facing scalability issues in requirements prioritization with machine learning techniques*. Proceedings of 13th IEEE International Conference on Requirements Engineering, 297 – 305.

Bachmann, F., Bass, L., Clements, P., Garlan, D.,, Stafford, J. (2002). *Documenting Software Architecture: Documenting Interfaces* (nota técnica). CMU/SEI-2002-TN-015. Pittsburgh, PA 15213-3890

Barbacci, M. R., Ellison, R., Lattanze, A. J., Stafford, J. A., Weinstock, C. B., Wood, W. G. (2003). *Quality Attribute Workshops (QAWs)*, Tercera edición (Reporte técnico). CMU/SEI-2003-TR-016, Pittsburgh, PA 15213-3890.

Bashroush, R., Spence, I., Kilpatrick, P., Brown, T. J., Gilani, W., Fritzsche, M. (2008). *ALI: An Extensible Architecture Description Language for Industrial Applications*. 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems.

Bass, L., Clements, P., y Kazman, R. (2012). *Software Architecture in Practice*, Tercera edición, Reading MA: Addison-Wesley, 2012, ISBN: 0-321-81573-4.

Berander, P., Jonsson, P. (2006). *Hierarchical cumulative voting - prioritization of requirements in hierarchies*, International Journal of Software Engineering and Knowledge Engineering (IJSEKE). Vol. 16, No. 6, 819-850.

Booch, G. (1993). *Object-Oriented Analysis and Design with Applications*. Segunda edición. Benjamin-Cummings Pub. Co., Redwood City, California, 589 págs.

Burger, E. (2014). *Flexible Views for View-based Model-driven Development*. Institut für Programmstrukturen und Datenorganisation (IPD)

Buch, D. (2012). *The Karlsruhe Series on Software Design and Quality*. ISBN: 978-3-7315-0276-0.

Choi, Y. H. (2003). *Users Manual. Schedulability Analyzer for AcmeStudio*. Institute for Software Research International, School of Computer Science, Carnegie Mellon University.

Ciancarini, P., Mascolo, C. (1998). *Software architecture and mobility*. In Proceedings of 3rd International Software Architecture Workshop (Orlando, FL, November). ISAW-3. In D. Perry and J. Magee, editors. ACM SIGSOFT Software Engineering Notes, 21-24.

Cicchetti, A., Ciccozzi, F., Leveque, T. (2012). *Supporting Incremental Synchronization in Hybrid Multi-view*. Modelling, Models in Software Engineering, Workshops and Symposia at MODELS 2011, Wellington, New Zealand, 89-103, 2012, ISBN: 978-3-642-29644-4.

Clements, P. (1996). *A survey of Architecture Description Languages*. Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA 1521, Eight International Workshop on Software Specification and Design, Germany.

Clements, P. Bachmann, F., Bass, L., ... Stafford, J. (2003). *Documenting Software Architectures, Views and Beyond*. Addison-Wesley 3rd Printing. ISBN: 0-201-70372-6.

Dashofy, E. M., Hoek, E., Taylor, R. N. (2001). *A Highly-Extensible, XML-Based Architecture Description Language*. Department of Information and Computer Science University of California, Irvine. Irvine, CA 92697-3425, U.S.A.

Dybá, T., Dingsøyr, T., Moe, N. (2004). *Process Improvement in Practice, A handbook for IT Companies*. Springer US. Primera edición. ISBN: 978-1-4020-7869-9. Págs. 114,

Feiler, P. H., Gluch, D. P., Hudak, J. J. (2006). *The Architecture Analysis & Design Language (AADL): An Introduction, Performance-Critical Systems* (Nota técnica). CMU/SEI-2006-TN-011

Gane, C., Sarson, T. (1979). *Structured Systems Analysis: Tools and Techniques*. Englewood Cliffs, NJ: Prentice-Hall.

Garlan, D., Allen, R., Ockerbloom, J. (1994). *Exploiting Style in Architectural Design Environments*. Proc. SIGSOFT '94: Foundations of Software Eng. Págs. 175-188.

Garlan, D., Monroe, R. T., Wile, D. (1997). *Acme: An Architecture Description Interchange Language*. Toronto, Ontario.

Garlan, D., Monroe, R. T., Wile, D. (2000). *Acme: Architectural Description of Component-Based Systems*, in Foundations of Component-Based Systems. University Press.

Glosario, Software Engineering Institute, Consultado el 23 de febrero de 2015, de <http://www.sei.cmu.edu/architecture/start/glossary/>

Gruhn, V., Schafer, C. (2004). *An Architecture Description Language for Mobile Distributed Systems*. In Proceedings of the First European Workshop on Software Architecture (EWSA 2004), Springer-Verlag, 212-218.

Hayes-Roth, F. (1994). *Architecture-Based Acquisition and Development of Software: Guidelines and Recommendations from the ARPA Domain-Specific Software Architecture (DSSA) Program*. Teknowledge Federal Systems. Version 1.01.

Hoare, C. A. R. (1978). *Communicating Sequential Processes*. The Queen's University Belfast, Northern Ireland. ACM 001-0782/78/0800-0666.

Hofmeister, C., Nord, H., Soni, D. (1999). *Applied Software Architecture*. Reading, MA: Addison-Wesley.

ISO/IEC 25010. *Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. 2011.

ISO/IEC/IEEE 15289:2011. *Systems and software engineering — Content of life-cycle information products*. Primera edición, 2011.

ISO/IEC/IEEE 42010:2011(E). *Systems and software engineering — Architecture description*. 2011.

Jacobson, I., Booch, G., Rumbaugh, J. (1999). *El proceso Unificado de Desarrollo de Software*. Addison-Wesley.

Jia, X., Ying, S., Zhang, T., Cao, H., Xie, D. (2007). *A New Architecture Description Language for Service-Oriented Architecture*. Wuhan University. Sixth International Conference on Grid and Cooperative Computing. ISBN: 0-7695-2871-6.

Kazman, R., Klein, M., Paul Clements, P. (2000). *ATAM: Method for Architecture Evaluation* (Reporte técnico). CMU/SEI-2000-TR-004 ESC-TR-2000-004 Pittsburgh, PA 15213-3890.

Kruchten, P. (1995). *Architectural Blueprints - The "4+1" View Model of Software Architecture*. IEEE Software 12(6). Págs42-50.

Kuchten, P. (2000). *The Rational Unified Process: An Introduction*. Second Edition. Addison-Wesley.

Li, J., Pilkington, N. T., Xie, F., Liu, Q. (2009). *Embedded architecture description language*. The Journal of Systems and Software 83 (2010) 235–252.

Lopes, A., Fiadeiro, J.L., Wermelinger, M. (2002). *Architectural Primitives for Distribution and Mobility*. In Proceedings of 10th Symposium on Foundations of Software Engineering, ACM Press, 41-50.

Luckham, D., Vera, J. (1995). *An Event-Based Architecture Definition Language*. IEEE Transactionson Software Engineering.

Magee, J., Dulay, N., Eisenbach, S., Kramer, J. (1995). *Specifying Distributed Software Architectures*. Proc. of 5th European Software Engineering Conference (ESEC '95).

Majumdar, S. I., Rahman, S., Rahman, M. (2013). *Thorny Issues of Stakeholder Identification and Prioritization in Requirement Engineering Process*. IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727 Volume 15. Department of Software Engineering, Daffodil International University, Dhanmondi, Dhaka, Bangladesh

Medvidovic, N., Rakic, M. (2001). *Exploiting Software Architecture Implementation Infrastructure in Facilitating Component Mobility*. In Proceedings of the Software Engineering and Mobility Workshop (Toronto, Canada, May).

Ministry of Defence Architecture Framework (MODAF), <http://www.modaf.org.uk/>. Consultado Septiembre 2015.

Mulla, N., Girase, S. (2012). *A new approach to requirement elicitation on stakeholder recommendation and collaborative filtering*. International Journal of Software Engineering and Applications, Vol.3, No.3.

Oquendo, F. (2004). π -ADL: *An Architecture Description Language based on the Higher-Order Typed π -Calculus for Specifying Dynamic and Mobile Software Architectures*. University of Savoie at Annecy. ACM Software Engineering Notes Volume 29, Issue 3.

Pressman, R. S. (2012). *Ingeniería de Software, Un enfoque práctico*. Séptima edición. New York. Mc Graw Hill. Págs. 777. ISBN: 978-607-15-0314-5.

Ricks, K. G., Weir, J. M., Wells, B E. (1995). *SADL: Simulation Architecture Description Language*.

RM-ODP. (1994). ISO. *Reference Model of Open Distributed Processing*. International Organization for Standardization. Reporte técnico 10746.

Rosen, J. P., Tilman, J. F. (2005). *Overview of AADL Syntax*. AADL Workshop.

Rozansky, N., Eoin, W. (2005). *Software Systems Architecture, Working with stakeholders using viewpoints and perspectives*. Addison-Wesley. Págs 529. ISBN: 0-321-11229-6.

Ruscio, D., Malavolta, I., Muccini, H., Pelliccione, P., Pierantonio, A. (2010). *ByADL: An MDE Framework for Building Extensible Architecture Description Languages*. University of L'Aquila, Dipartimento di Informatica. LNCS 6285, pp. 527–531, 2010.

Shaw, M., DeLine, R., Klein, D. V., Ross, T. L., Young, D. M., Zelesnik, G. (1994). *Abstractions for Software Architectures and Tools to Support Them*. Carnegie Mellon University.

Shaw, M., DeLine, R., Klein, D. V., Ross, T. L., Young, D. M., Zelesnik, G. (1995). *Abstractions for Software Architecture and Tools to Support Them*. IEEE Trans. Software Eng., vol. 21, no. 4, pp. 314- 335.

Sitio de AADL. <http://www.aadl.info/aadl/currentsite/index.html>. Consultado el 7 de septiembre de 2015.

Sitio de ACME. <http://www.cs.cmu.edu/~acme/>. Consultado el 7 de septiembre de 2015.

Sitio de ArchiMate® <http://www.archimate.nl/en/>. Consultado el 7 de septiembre de 2015.

Sitio de Wright. <http://www.cs.cmu.edu/~able/wright/>. Consultado el 7 de septiembre de 2015.

Sitio de xADL. <http://isr.uci.edu/projects/xarchuci/>. Consultado el 7 de septiembre de 2015.

Smeda, A., Oussalah, M., Khammaci, T. (2005). *MADL: Meta Architectura Description Language*. LINA, University of Nantes 2, Rue de la Houssinière, BP 92208.

Sommerville, Ian. (2005). *Ingeniería de software*. Séptima edición. Madrid. Pearson Addison-Wesley. ISBN: 84-7829-074-5.

Soni, D., Nord, R. L., Hofmeister, C. (1995). *Software, architecture in industrial applications*. In Proceedings of the 17th International Conference on Software Engineering. Páginas 196-207. Seattle, Washington, USA. ACM Press.

Tanenbaum, A. S., Wetherall, D. J. (2011). *Computer networks*. Quinta edición. Prentice Hall. ISBN -10: 0-13-212695-8.

Thompson, H. S., Tobin, R. (2007). *Current Status of XSV: Coverage, Known Bugs, etc.*

TOGAF. *The Open Group Architecture Framework (TOGAF)*. <http://www.opengroup.org/togaf/>.

Xu, D., Yin, J., Deng, Y., Ding, J. (2003). *A Formal Architectural Model for Logical Agent Mobility*. IEEE Transactions on Software Engineering, Vol. 29, N° 1, January, 2003.

Zachman, J.A. (1987). *A Framework for Information Systems Architecture*. IBM Systems Journal, 26(3).