Enl@ce: Revista Venezolana de Información, Tecnología y Conocimiento
ISSN: 1690-7515
Depósito legal pp 200402ZU1624
Año 10: No. 2, Mayo-Agosto 2013, pp. 11-27

Cómo citar el articulo (Normas APA)
Guzmán, J., Losavio, F. y Matteo, A. (2013). Comparación de métodos para el diseño arquitectónico del software que consideran las orientaciones de metas y aspectos. Enl@ce Revista Venezolana de Información, Tecnología y Conocimiento, 10 (2), 11-27.

Comparación de métodos para el diseño arquitectónico del software que consideran las orientaciones de metas y aspectos

Jean Carlos Guzmán[1]
Francisca Losavio[2]
Alfredo Matteo[3]

Resumen

Se presenta como objetivo realizar un marco de comparación aplicado a métodos de diseño arquitectónico del software que incorporen metas, aspectos o estándares de calidad para especificar Requisitos no Funcionales RNF, para identificar un conjunto de características adecuadas, y poder así definir un método general de diseño arquitectónico, al cual denominaremos *Diseño Arquitectónico Orientado a Metas, Aspectos y Calidad (DAOMAC).* Los resultados principales son el marco de comparación de estos métodos y el conjunto de características que debe ser considerado en un método general de diseño arquitectónico basado en metas, aspectos y estándares de calidad. Actualmente, existe un consenso en considerar *metas no funcionales* (MNF) en etapas tempranas del ciclo de vida del software, en vista que su postergación causa dispersión en el código resultante, dificultando la evolución del sistema. Se prevé, que una *meta* es considerada un objetivo de alto nivel de abstracción de la organización del sistema o de las personas involucradas o actores. A nivel del sistema de software, las *metas funcionales (MF)* representan la intencionalidad del actor; las MNF corresponden a *RNF* o *incumbencias no funcionales* que no son directamente percibidas por el actor; cuando éstas entrecruzan a otras incumbencias en el sistema, corresponden a *incumbencias transversales*. Un *aspecto* es una estructura que encapsula una incumbencia transversal y su origen es a nivel de implementación, sin embargo se considera ahora también en etapas de modelado del negocio, ingeniería de requisitos y diseño arquitectónico. En líneas generales, se hace referencia a que la *arquitectura* del sistema de software es considerada un enlace entre requisitos y código.

Palabras clave: arquitectura del software, métodos de diseño arquitectónico, métodos de evaluación arquitectónica, estándares de calidad del software.

Recibido: 21/6/13 Aceptado: 30/7/13

1 Caribbean International University, Willemstad, Curaçao.
Correo electrónico: jeancguzman@outlook.com.
2 Laboratorio MoST, Centro Laboratorio MoST, Centro ISYS, Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela; Caracas, Venezuela.
Correo electrónico: francislosavio@gmail.com.
3 Laboratorio MoST, Centro ISYS, Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela; Caracas, Venezuela.
Correo electrónico: alfredojose.matteo@gmail.com.

Comparison of Methods for Software Architectural Design what considering Goals and Aspects Orientations.

Abstract

It is presented as objective to carry out a comparative framework applied to methods of architectural design of the software incorporating goals, aspects and/or quality standards to specify non-functional requirements RNF, to identify a set of appropriate features, and thus to be able to define a general method for architectural design, which we call Architectural Design goal-oriented, and quality aspects (*DAOMAC*). The main results are the framework of comparison of these methods and the set of features that must be considered in a general method for architectural design based on goals, aspects and quality standards. Currently, there is a consensus to consider non-functional goals (MNF) in early stages of the life cycle of the software, view that its postponement cause dispersion in the resulting code, making the evolution of the system. It is expected, that a goal is regarded as an objective of high- level abstraction of the organization of the system or of the persons involved or actors. At the level of the software system, the functional goals (MF) represent the intent of the actor; the MNF correspond to RNF or non-functional roles that are not directly perceived by the actor; when these intertwined to other roles in the system, correspond to transverse incumbencies. One aspect is a structure that encapsulates a competence cross and its origin is at the level of implementation, however, it is considered now also in stages of the business modeling, requirements engineering and architectural design. In general, reference is made to the architecture of the software system is considered a link between requirements and code.

Keywords: Software architecture, Methods of architectural design, Architectural assessment methods, Standards of software quality.

Introducción

El objetivo principal de este trabajo es presentar un marco de comparación aplicado a métodos de diseño arquitectónico del software que incorporen metas, aspectos y/o estándares de calidad para especificar requisitos no funcionales (RNF), con el fin de identificar un conjunto de características adecuadas, y poder así definir un método general de diseño arquitectónico, al cual denominaremos *Diseño Arquitectónico Orientado a Metas, Aspectos y Calidad (DAOMAC)*

En el contexto de la Ingeniería de Requisitos Orientada a Metas o "Goal Oriented Requirement Engineering (GORE)", las metas representan los objetivos de alto nivel de la organización, del sistema y de los participantes o stakeholders.

Para los efectos, se plantean dos clases de metas, las *funcionales* (MF) y las *no funcionales* (MNF).

Las Metas Funcionales MF, representan las intenciones deseadas por un actor, los detalles específicos de cómo la meta va ser satisfecha no son descritos en la especificación de la meta: por ejemplo, cual es el grado de "seguridad" que el actor puede conseguir al cumplirse la meta funcional "acceso"; este requisito implícito de seguridad constituye la meta no funcional MNF "seguridad", que también debe ser satisfecha al cumplirse la meta funcional "acceso".

Las MNF se pueden especificar como criterios o restricciones que no dependen explícitamente del punto de vista del actor (ITU- T, 2012); pueden ser globales, es decir válidas para todo el sistema. Son conocidas coloquialmente como "ilities", "softgoals" o "non- functional requirements (NFR)".

En general en el contexto de GORE, los softgoals son representados como un grafo denominado diagrama SIG *(Softgoals Interdependency Graph)* (Chung y Do Prado Leite, 2009). Estas características pueden ser especificadas a través del estándar de calidad ISO/IEC 25010 (ISO/IEC, 2011), el cual se establece como norma o convención internacional que deben cumplir los productos de software para su consumo satisfactorio por el cliente final.

Para los efectos, se define la *calidad del software* como la capacidad que deben poseer estos para satisfacer necesidades establecidas o implícitas y es representada mediante dos modelos a través de una estructura jerárquica: el *Modelo de Calidad del Producto (MC)*, o vista de calidad del producto de software, incluye la calidad de los subproductos o artefactos construidos durante todo el proceso de desarrollo, y el *Modelo de Calidad en Uso* o vista de calidad del usuario, que aplica al sistema ya desarrollado en ejecución en su entorno final.

El MC propone ocho (8) características de calidad de alto nivel tales como: adecuación funcional, eficiencia en rendimiento, compatibilidad, usabilidad, confiabilidad, seguridad, mantenimiento y portabilidad, que se subdividen en sub-características, hasta llegar a los atributos o elementos medibles del software. Un producto de software, puede ser también un producto intermediario (por ejemplo modelo de casos de uso, modelo arquitectónico, etc.) obtenido durante el proceso de desarrollo.

Debe observarse que cuando se habla de RNF, es en el contexto del cliente quien proporciona la descripción de los requisitos del sistema; cuando se habla de requisitos de calidad, es en el contexto del ingeniero o arquitecto de software. Un requisito de calidad es especificado por una característica de calidad en el estándar ISO/IEC 25010. De igual forma se acota que el uso de estándares es una práctica recomendada por la ingeniería del software porque facilita el entendimiento y la comunicación entre los diferentes grupos de trabajos.

En particular, la terminología acerca de la calidad del software varía mucho de acuerdo al contexto en donde es utilizada; en este trabajo, para evitar ambigüedades, se utilizará el MC propuesto por ISO/IEC 25010.

Por otra parte, actualmente existe un interés en considerar las MNF en etapas tempranas del ciclo de vida del software, en vista que su postergación causa dispersión o enmarañamiento en el código resultante, dificultando el mantenimiento del sistema en producción.

Desde la perspectiva del enfoque denominado *Orientación a Aspectos* o "*Aspect-Oriented Software Development (AOSD")* (Sutton y Tarr, 2002), las *incumbencias* son cualquier preocupación o propiedad de interés de un sistema; en este contexto las incumbencias que entrecruzan otras *incumbencias funcionales (IF) o no funcionales* (INF), se les denomina *incumbencias transversales* (IT): por ejemplo en un sistema de Banca Electrónica, la s*eguridad* es una IT debido que entrecruza las IF, *Pago de Servicios, Consulta de Cuentas*, *Transferencias entre Cuentas* y *Control de Acceso*.

Es a nivel de la especificación de la *arquitectura del software* o estructura de componentes, conectores y relaciones de comportamiento (Garlan y Shaw, 1996), donde se manifiestan las IT que van a corresponder con los componentes del sistema; son candidatas a ser tratadas como aspectos en la etapa de implementación por implementación.

Un aspecto es una estructura que encapsula una incumbencia transversal (Kiczales al et., 1997) y su origen es a nivel de implantación por implementación, sin embargo está siendo considerado ahora también en las etapas tempranas del modelado de negocio (Cappelli et al., 2009), ingeniería de requisitos (Tekinerdogan et al., 2004) y *diseño arquitectónico* (Krechetov et al., 2006), el cual es una disciplina de gran importancia para el proceso de desarrollo de software, en virtud de que a través de este, con base en el espacio del problema se produce una estructura arquitectónica o estilo arquitectónico del sistema por donde se ubican las MF.

Y por otra parte, se satisfacen las MNF y claro está, las IT más significativas en el espacio de la solución para seleccionar alternativas de diseño conforme al dominio de aplicación.

La arquitectura del software es considerada un enlace entre los requisitos y el código del sistema. Se plantean dos grandes clases de métodos de diseño arquitectónico, los de diseño y los de evaluación y con frecuencia estos métodos se combinan. Los *métodos de diseño arquitectónico* admiten la construcción de la arquitectura del software con el fin de satisfacer los requisitos de calidad que afectan el comportamiento global del sistema (Maurya y Hora, 2010).

En cambio los *métodos de evaluación arquitectónica* están dirigidos a determinar, si la arquitectura seleccionada conducirá a los atributos de calidad deseados (Babar, Zhu y Jeffery, 2004) y así seleccionar una arquitectura adecuada en función del valor de estos atributos; también pueden considerarse los objetivos del negocio (Husnain y Ahmed, 2012). Sin embargo, diversos métodos de diseño arquitectónico incluyen también una etapa de evaluación arquitectónica.

En Grimán et al. (2006), se plantea que el uso de métodos apropiados para el diseño y la evaluación de arquitecturas puede ayudar a reducir el riesgo de una arquitectura de baja calidad. En la literatura se han propuesto diversos métodos tanto de diseño como de evaluación, cada uno con sus fortalezas y debilidades, pero fundamentados en una conceptualización que puede ser equivalente, complementaria o alternativa.

Este artículo, además de la introducción y las conclusiones, consta de seis secciones principales: una segunda sección que presenta los métodos de diseño y de evaluación de arquitecturas de software que consideran la orientación a metas, aspectos y además utilizan estándares de calidad del software para especificar los RNF. Una tercera sección que presenta el marco de referencia para la comparación de los métodos.

Una cuarta sección que muestra los resultados de la comparación de los métodos estudiados. Una quinta sección que describe las características generales que deben estar presentes en la definición del método DAOMAC. Finalmente, una sexta sección discute los trabajos relacionados.

Métodos de diseño de arquitecturas de software orientados a metas y/o aspectos y que utilizan estándares de calidad para especificar los RNF

Losavio y Guillen (2010) consideran como *Métodos de Diseño Arquitectónico,* a los que involucran la concepción de un sistema de software en un alto nivel de abstracción.

La gran mayoría de estos métodos tienen como objetivo la generación de una arquitectura que responda a un conjunto de requisitos iniciales del sistema, tanto funcional como no funcional. Por otra parte, los *Métodos de Evaluación Arquitectónica,* buscan predecir la calidad o adecuación de la arquitectura antes de que el sistema de software sea construido (Scholten, 2007). Dependiendo de los resultados de estas evaluaciones, la arquitectura del software puede transformase de acuerdo con los requisitos de calidad exigidos por el sistema en construcción.

Estos métodos están dirigidos a evaluar *configuraciones arquitectónicas* (conjunto de componentes y conectores que describen una solución arquitectónica) ya existentes, para escoger la mejor de acuerdo a un conjunto de requisitos. Sin embargo, todos estos enfoques son subjetivos porque dependen de la experticia del arquitecto y se ubican en las fases tempranas del proceso de desarrollo de software (Losavio y Guillen, 2010).

Para este estudio, se han considerado nueve (9) métodos de diseño de arquitectura de software que, de acuerdo al interés de este trabajo han sido seleccionados por considerar la orientación a metas y/o la orientación aspectos como enfoques principales de diseño. Debe notarse que todo método de diseño arquitectónico considera RNF y los correspondientes requisitos de calidad, por ser estos esenciales en la toma de decisiones del diseño arquitectónico o en la evaluación de las arquitecturas.

Sin embargo, se quiere indagar explícitamente si el método posee alguna técnica para especificar estos RNF mediante algún estándar de calidad. También se quieren señalar en esta revisión, aquellos métodos que consideran metas de negocio, por ser este un objetivo importante a tomar en cuenta, puesto que la tendencia actual prevé considerar el modelo de negocio como punto de partida de la especificación de un sistema de software.

En lo que sigue se presentan brevemente los métodos considerados:

1) *Preskriptor* (Brandozzi, 2003), propone un método para el **diseño de arquitecturas** de sistemas de software de alto nivel que garantiza la satisfacción de los requisitos o necesidades iniciales. Proporciona una visión general **orientada a metas** de especificación de requisitos y configuraciones arquitectónicas. Sigue el enfoque de KAOS (*Knowledge Acquisition in autOmated Specification*) (Van Lamswerdee, 2001). Los RF y RNF se especifican primero en una notación gráfica no estándar de árbol, luego son transformados formalmente a configuraciones arquitectónicas utilizando Preskriptor (*Prescription Specification Language*) como un lenguaje ADL (*Architecture Description Language*). No utilizan estándares para esta especificación.

2) *Proteus* (Chung, Cooper y Yi, 2002), es un marco conceptual que sigue la **orientación a metas** y que tiene por objetivo apoyar el **diseño de arquitecturas de software** adaptables utilizando patrones de diseño. Está centrado en el Framework NFR, el cual trata los requisitos no funcionales (NFR) como "softgoals" a ser satisfechos. Los requisitos de calidad se especifican como softgoals, mediante el grafo SIG, sin utilizar ningún estándar.

3) *Goal and Scenario Modelling Design (GSMD) method* (Liu y Yu, 2004), es un **método de diseño arquitectónico** fundamentado en el enfoque **GORE** para la derivación de arquitecturas de software a partir de un modelo de metas expresado en **GRL** que incorpora metas tanto del negocio como del sistema utilizando **escenarios** expresados en **Use Case Maps** (UCM), usados para describir procesos de negocios definidos en GRL (ITUT, 2012).

Los RF y RNF, se especifican como **metas (funcionales o no)** primero en una notación gráfica no estándar de árbol, luego son transformados a escenarios orientados al diseño, que se ocupa del **refinamiento de los requisitos del sistema**.

La calidad de la arquitectura correspondiente a estos escenarios se evalúa en base al conocimiento experto como producto del diseño utilizando SAAM (*Software Architecture Analysis Method*) (Kazman y Otros, 1994). No utilizan estándares para esta especificación.

4) *Aspectual Software Architecture Analysis Method* (ASAAM) (Tekinerdogan, 2004); es un método de **análisis y evaluación de arquitecturas**, basado en el conocido SAAM (*Software Architecture Analysis Method*) (Kazman y Otros, 1994) y ATAM (*Attribute Trade-offs Analysis Method*) (Barbacci y Otros, 1998). Propone un enfoque para la **evaluación de aspectos** a nivel arquitectural, mediante la identificación y especificación explícita de incumbencias transversales en las etapas tempranas del ciclo de vida. Finalmente, los requisitos se especifican a través de escenarios directos e indirectos, como en SAAM. No utilizan estándares para la especificación de requisitos.

5) *Vanderveken, Van Lamsweerde, Perry y Ponsard* (2005), proponen un método formal fundamentado en el enfoque de **orientación a metas (**KAOS**)** para la derivación de arquitecturas de software basado en un **diseño arquitectónico por transformaciones**, a partir de un modelo de requisitos, aplicándose patrones de refinamiento.

El lenguaje de origen y destino son el lenguaje KAOS y el ADL *Wright* (Garlan y Shaw, 1996) respectivamente. Finalmente, los requisitos en general se derivan de una notación gráfica no estándar de árbol, luego pasan a ser clases UML (OMG, 2005) y finalmente, son descritos formalmente en *Wright*. No utiliza estándares de calidad.

6) *Concern-Oriented Software Architecture Analysis Method* (COSAAM) (Scholten, 2007), es un método iterativo de **evaluación y diseño por transformación** de arquitecturas de software. Utiliza y extiende diversos enfoques existentes para crear un método para transformar sistemáticamente arquitecturas de software basado en reglas. Es un enfoque inspirado en ASAAM (Tekinerdogan, 2004), que define un método de análisis y evaluación basado en escenarios para identificar aspectos arquitectónicos. Finalmente, los requisitos de calidad se especifican a través de escenarios como en SAAM y ASAAM. No utiliza estándares de calidad.

7) Aspect-oriented Model-Driven Software Product Line Engineering (AMPLE) (Rashid, Royer y Rummler, 2011); este enfoque aborda el tema del **diseño de Arquitecturas de Líneas de Productos** de Software o *Software Product Line Architecture* (PLA) **basado en incumbencias**.

Es un "toolkit" de técnicas y explota la sinergia entre la orientación a aspectos y MDE (*Model Driven Engineering*) (Kent, 2002) (Schmidt, 2006), hacia un tratamiento sistemático de la variabilidad de los productos en la línea de productos.

Utiliza la herramienta DiVA (*Dynamic Variability in Adaptive System*) para el manejo de la variabilidad dinámica, para especificar requisitos en general; utiliza la **orientación a metas** para identificar la intencionalidad de los requisitos en general. Finalmente, los requisitos de calidad se especifican como softgoals. No se utilizan estándares para esta especificación.

8) Strategy for Transition between REquirements models and Architectural Models (STREAM) (Castro, 2011), es un enfoque sistemático de **diseño arquitectónico orientado a metas**, basado en el enfoque de MDE y por ende, en transformación de modelos para la generación de arquitecturas de software a partir de los modelos de *i** (Yu y Mylopoulos, 1993): incluye reglas de transformación para derivar especificaciones arquitectónicas a partir de metas del negocio y del sistema. Los lenguajes de origen y destino son, respectivamente, el lenguaje de modelado *i** y el ADL *Acme* (Garlan, Monroe y Wile, 2010). Los requisitos de calidad se especifican como softgoals. No se utilizan estándares para esta especificación.

9) Strategy for Transition between Requirements and Architectural Models with Architectural Decisions Documentation (STREAM-ADD) (Dermeval, 2012) es un método de **diseño arquitectónico orientado a metas,** basado en transformaciones (refinamiento) que extiende STREAM (Castro et al., 2011). Este enfoque refina metas del negocio y del sistema de software, las metas no funcionales del Modelo de Razonamiento Estratégico de *i** (Yu y Mylopoulos, 1993) en mecanismos o soluciones arquitectónicas, utilizando reglas de transformación. Finalmente, los requisitos de calidad se especifican como softgoals. No se utilizan estándares para esta especificación

Marco de referencia para la comparación de los métodos

La revisión de los distintos enfoques mencionados han permitido definir un marco de comparación, el cual es una adaptación de los trabajos de Thiel (2005) y Losavio y Guillen (2010). Los métodos objetos de este estudio se comparan desde **siete puntos de vistas o criterios,** seleccionados después de aplicar el método de análisis de características propuesto por DESMET (Kitchenham, 1996): *orientación metas, metas de negocio, orientación a aspectos, evaluación arquitectónica, especificación de requisitos de calidad, notación arquitectónica y herramientas de soporte.*

A continuación, en el Cuadro 1 a continuación presenta el marco de referencia definido, en donde se indican los criterios adoptados, con sus respectivas definiciones y su uso al examinar un método de diseño arquitectónico:

Cuadro 1
Marco de referencia de los criterios comparativos

Ítem	Criterio	Definición	Uso del criterio en el método de diseño arquitectónico
1	Orientación a Metas	Condición o estado de un asunto que a los stakeholders les gustaría lograr. Existen dos clases de metas: las Metas Funcionales (MF). Las MNF o "softgoals" son criterios o restricciones que no dependen explícitamente del punto de vista del actor; pueden ser globales, es decir válidas para todo sistema (ITU-T, 2012). Están directamente relacionadas con los requisitos de calidad. Se especifican por un grafo, el diagrama SIG (Chung, Cooper y Yi, 2002).	Indica si el método es **orientado a metas**, es decir, si emplea MNF derivadas del modelo de metas o del modelo de negocio.
2	Metas de Negocio	Las metas de negocio corresponden y conducen a los requisitos de los diferentes procesos de negocio (Behnam, Amyot y Mussbacher, 2010).	Indica si el método considera **metas de negocio**

Fuente: Elaboración propia

Continuación

Cuadro 1

Marco de referencia de los criterios comparativos

Ítem	Criterio	Definición	Uso del criterio en el método de diseño arquitectónico
3	Orientación a Aspectos	Estructuras que encapsulan las incumbencias transversales (Kiczales et al., 1997). Una incumbencia es un asunto de interés. Las incumbencias transversales son incumbencias que entrecruzan diferentes módulos de un sistema.	Indica si el método trata las incumbencias transversales.
4	Evaluación Arquitectónica	Comparan y/o valoran diferentes mecanismos/soluciones arquitectónicas y ayudan al arquitecto en la toma de decisiones.	Indica si el método **incorpora técnicas para comparar mecanismos/soluciones arquitectónicas** y facilidades para la toma de decisiones.
5	Especificación de requisitos de calidad	Los requisitos de calidad representan RNF dados por el cliente como una propiedad del software; se sitúan al mismo nivel que los RF (ISO/IEC, 2011).	Indica si el método contempla técnicas para la especificación de RNF; se quiere identificar si se utilizan estándares para especificar los requisitos de calidad, como por ejemplo el estándar ISO/IEC 25010.
6	Notación arquitectónica	Es un conjunto de reglas gráficas o textuales que se deben seguir, para describir componentes de un sistema. Definen un lenguaje conformado por una sintaxis y semántica, por ejemplo UML (OMG, 2005).	La notación textual o gráfica utilizada en el proceso de diseño arquitectónico.
7	Herramientas de Soporte	Son sistemas o aplicaciones de software de soporte al modelado y al proceso de un método específico.	Indica si el método considera herramientas de soporte para el proceso de diseño.

Fuente: elaboración propia

Resultados Obtenidos

El cuadro 2 presenta la comparación de nueve métodos de diseño arquitectónico considerados en este trabajo, siguiendo el marco de referencia definido en el cuadro 1.

Cuadro 2

Comparación de los métodos estudiados

	Método/Año	Criterios de Comparación									
		1. Orientación a Metas	2. Metas de Negocio	3. Orientación a Aspectos	4. Evaluación Arquitectónica	5. Especificación de Requisitos de Calidad				6. Notación Arquitectónica	7. Herramientas de Soporte
						Escenarios	Softgoals	Clases UML	Estándares de calidad		
1	PRESKRIPTOR (2001)	√								√	
2	Proteus (2002)	√					√				√
3	GSMD (2004)	√	√		√	√	√				√
4	ASAAM (2004)			√	√	√		√			√
5	Vanderveken, Van Lamsweerde, Perry y Ponsard (2005)	√						√		√	√
6	COSAAM (2007)			√	√	√					
7	AMPLE (2011)	√		√	√	√				√	√
8	STREAM (2011)	√	√		√		√			√	√
9	STREAM-ADD (2012)	√	√		√		√			√	√

Fuente: Elaboración propia.

Se observa que desde el año 2001 hasta la fecha, se han presentado en la literatura siete (7) métodos que consideran la orientación a metas: - *PRESKRIPTOR* (2001); - *Proteus* (2002); - *GSMD* (2004); - *Vanderveken, Van Lamsweerde, Perry y Ponsard* (2005); *AMPLE (2011)*; - *STREAM* (2011); y - *STREAM-ADD* (2012) incorporan la orientación a metas. Asimismo, solo tres (3) de los métodos estudiados, *GSMD* (2004), *STREAM* (2011) y *STREAM-ADD* (2012), consideran además metas del negocio a nivel de Ingeniería de Requisitos. Por otra parte, solo dos (2) métodos enfatizan la orientación a aspectos: - *ASAAM* (2004) y - *COSAAM* (2007). El único método que trata ambos enfoques, integrando la orientación a metas con la orientación a aspectos, es *AMPLE* (2011).

Es importante señalar, que seis (6) de los métodos estudiados incorporan técnicas de evaluación de mecanismos o soluciones arquitectónicas. Por otra parte, ninguno de los métodos objetos de estudio considera la especificación de requisitos de calidad a través de un estándar de calidad, como por ejemplo la norma ISO/IEC 25010 o el estándar IEEE.

Más aún, ninguno considera la integración de las tres técnicas de nuestro interés, la especificación de requisitos de calidad por estándares, la orientación a aspectos y a metas. Este resultado es alentador, en el sentido que la presente investigación es un campo aún abierto e indica que la definición de un método de diseño arquitectónico que integre las tres técnicas es una idea novedosa.

La posterior validación y experimentación con casos de estudio realistas, por ejemplo de sistemas de software complejos a ser desarrollados en el país para prestar servicios a la ciudadanía, se anuncia como un campo prometedor. Como conclusión del estudio realizado, la siguiente sección presentan las directrices generales para la definición de un método de diseño arquitectónico que incorpore metas, aspectos y estándares de calidad:

Directrices para la definición del método DAOMAC

Se ha precisado un conjunto de directrices (incluyendo técnicas y artefactos) que deben ser consideradas para la definición de un método de diseño arquitectónico que incorpore metas, aspectos y estándares de calidad, con bases en el estudio realizado; estas directrices se describen y justifican a continuación:

a) Modelo de negocio (MN)*:* debe definirse un modelo conceptual del negocio, el MN describe todos los procesos de una organización y de estos se derivan los sistemas computacionales de apoyo a esos procesos. Por lo tanto, desde el punto de vista de la arquitectura del futuro sistema y los requisitos a los que debe responder a este modelo, ofrece la información más completa para el ingeniero de requisitos y el arquitecto de software: esta información será parte del conocimiento del dominio de la organización y podrá ser reutilizada en futuros sistemas.

El MN debe ser expresado en una notación lo más estándar posible, por ejemplo BPMN, para facilitar la comunicación entre los *stakeholders. Justificación:* este modelo no se desprende del estudio aquí realizado, el nivel de abstracción más alto que consideran los métodos estudiados es el Modelo de Metas expresado por *KAOS*, *i* o GRL*.

Sin embargo, el método *Unified Process* (UP) Jacobson, (1999), se define como un método completo de desarrollo de software, centrado en la arquitectura, en su fase inicial considera importante pero no obligatorio, definir el MN para obtener la arquitectura inicial. No hemos reseñado este método en nuestro estudio comparativo, porque no considera la orientación a metas ni aspectos.

En tal sentido, es importante destacar que las nuevas tendencias, según Pa, Jusoh y Sani (2012), indican que el modelo de metas es considerado un nivel estratégico de importancia para la ingeniería de requisitos. Se tiende ahora a derivar de manera automática el sistema de software a partir de la especificación del proceso de negocio.

Un ejemplo actual son las aplicaciones basadas en servicios web: BPMN v. 2.0 incluye elementos para este propósito (OMG, 2011). Es por estas razones que hemos considerado trascendental la inclusión del MN en un método de diseño arquitectónico que responda a las exigencias tecnológicas actuales.

b) Modelo de metas (MM): debe definirse el MM a partir del MN utilizando reglas para la correspondencia semántica entre elementos de los modelos derivados de los lenguajes de MN (BPMN) y MM (GRL) tales como actores, roles, tareas, metas entre otros, así como el refinamiento de elementos transversales son imprescindibles para el buen desempeño de la construcción de la arquitectura.

El MM es uno de los primeros modelos que considera los elementos de alto nivel del sistema de software, y para ello debe establecerse una transición adecuada entre los elementos del MN y el MM, sobre todo respecto a los RNF que no están contemplados en el MN. Un lenguaje con una sintaxis y semántica bien definida debe ser utilizado para expresar el MM, por ejemplo *i** (Yu y Mylopoulos, 1993) o *GRL* (ITU-T, 2012). En este nivel de abstracción aún no hay un lenguaje estándar aceptado.

Justificación: cuatro de los nueve métodos estudiados consideran metas este nivel de abstracción y tres métodos refinan metas en mecanismos (patrones de diseño) o descripciones arquitectónicas en un ADL.

c) Especificación de requisitos de calidad mediante estándares: todos los métodos estudiados tratan alguna técnica para la especificación de los requisitos de calidad, porque estos requisitos son los que determinan las decisiones de diseño para la construcción de la arquitectura, sin embargo ninguno propone utilizar estándares de calidad para ello, lo cual parece una deficiencia general, ya que la terminología sobre calidad de software depende mucho del dominio y se dificulta la comunicación y la reutilización entre los stakeholders.

Justificación: se propone el uso del estándar ISO/IEC 25010 (ISO/IEC, 2011), internacionalmente reconocido y utilizado, para la especificación del modelo de calidad, considerando a la arquitectura como un producto intermedio del desarrollo.

d) Modelo de arquitectura inicial (MA): la arquitectura del sistema debe ser definida mediante un modelo preferiblemente estándar, para facilitar la comunicación entre los grupos de trabajo y en un lenguaje cuya sintaxis y semántica estén bien definidas, por ejemplo UML 2.0 (OMG, 2005). El paso del MM al MA debe ser realizado a través de técnicas de GORE, utilizando por ejemplo el SIG (Chung, Cooper y Yi, 2002), en el cual las MNF corresponderán a los softgoals, que son refinados hasta llegar a las operacionalizaciones.

Justificación: todos los métodos estudiados incluyen este nivel de abstracción.

e) Orientación a aspectos*:* el enfoque AOSD (Rashid, Royer y Rummler, 2011) implica un desarrollo de software completo que considera los aspectos desde las fases iniciales del ciclo de vida y favorece la facilidad de mantenimiento, es decir flexibilidad a los cambios, i.e. permite producir sistemas evolutivos.

*Justificación: e*n particular respecto con la arquitectura, dos de los métodos estudiados incorporan la orientación de los aspectos al diseño arquitectónico. En tanto que la consideración de incumbencias transversales a nivel del MM, permite identificar aquellos componentes que serán tratados y encapsulados luego como aspectos en la fase de implementación del sistema, facilitando así la capacidad evolutiva del sistema.

f) Lenguajes y notaciones estándares para la derivación y descripción arquitectónica: conforme al nivel de abstracción donde se ubiquen los actores del proceso. *Justificación:* ninguno de los métodos estudiados utiliza lenguajes y notaciones estándares.

Sin embargo, el uso de estándares es una buena práctica de la ingeniería del software, porque facilita la comunicación y el entendimiento entre los grupos de trabajo.

Trabajos relacionados

Se consideraran los trabajos que presentan una **revisión de métodos de diseño arquitectónico**; entre estos figuran los trabajos de: - Losavio, Chirinos y Pérez (2001), proponen un conjunto de características que deben estar presentes en un método de diseño arquitectónico. Y Hofmeister (2005), presenta un enfoque general que compara artefactos y actividades utilizados en métodos de diseño arquitectónico.

Por otra parte, se estudiaron los trabajos en los que se **comparan métodos de evaluación arquitectónica**, entre los que se tienen: Kazman, Nord y Klein (2003) presentan un enfoque de evaluación dirigido a estudiar la incorporación de requisitos de calidad, así como las actividades y los artefactos que subyacen al proceso de análisis y diseño arquitectónico.

Babar, Zhu y Jeffery (2004) definen un marco conceptual de referencia para la evaluación de arquitecturas para la selección de un método apropiado. Grimán et al. (2006) presentan un marco conceptual centrado en el análisis de características de los métodos de evaluación arquitectónica.

Maurya y Hora (2010) dan una caracterización de métodos de evaluación de arquitecturas del software. Finalmente, Husnain y Ahmed (2012) ofrecen un enfoque sobre procesos de evaluación de arquitecturas.

Asimismo, en los trabajos de Hofmeister *al et* (2007) se plantea la comparación de actividades y artefactos de cinco métodos de diseño arquitectónico de la industria con el propósito de definir lineamientos para la definición de un método genérico de diseño arquitectónico.

Por último, hay enfoques que consideran también **métodos de diseño y de evaluación de arquitecturas de software** en sus comparaciones: Thiel (2005) define un marco conceptual sistemático orientado al diseño y evaluación de arquitecturas centrado en la calidad de sistemas intensivos. Este enfoque apoya el desarrollo de arquitecturas en función de los requisitos de calidad, evaluación arquitectónica y documentación de las decisiones de diseño.

Así como también, Losavio y Guillén (2010) presentan un marco conceptual para la comparación de métodos de diseño y evaluación, con la finalidad de identificar sus semejanzas y diferencias, así como explorar las heurísticas que subyacen en cada propuesta.

Conclusión

Se presenta un marco de comparación aplicado a nueve métodos de diseño arquitectónicos propuestos en la literatura, que consideran los enfoques de orientación a metas y a metas y aspectos, no encontrándose ninguno que considere estándares para la especificación de los requisitos de calidad. Como resultados, se destacan los seis métodos que admiten la orientación a metas, de los cuales tres consideran el modelado del negocio como nivel de abstracción inicial; dos métodos enfatizan la orientación en función a metas y aspectos; y solo uno combina las orientaciones a metas y a aspectos.

Con base de estos resultados, se han propuesto un conjunto de modelos que deben estar presentes en un método de diseño arquitectónico denominado DAOMAC, el cual se fundamenta en los tres enfoques objeto del estudio: orientación a metas, a aspectos y el uso de estándares de calidad.

Estos modelos son el modelo de negocio, el modelo de metas con aspectos, el modelo de calidad basado en un estándar y el modelo arquitectónico expresado en UML. Finalmente, DAOMAC está actualmente en fase de definición.

Agradecimiento

Queremos agradecer a los entes PEII-Fonacit (Proyecto DISOFT No. 2011001343), y el Consejo de Desarrollo Científico y Humanístico (CDCH) de la Universidad Central de Venezuela (Proyecto DesCCaP No. PG-03-8014-2011) por el soporte financiero.

Referencias

Babar, M. Zhu, L. y Jeffery, R. (2004). A Framework for Classifying and Comparing Software Architecture Evaluation Methods. Proc. ASWEC (4), 309.

Barbacci, M., Feiler, P., Klein, M., [et. al.] (1998). Steps in an Architecture Tradeoff Analysis Method: Quality Attribute Models and Analysis (CMU/SEI-97-TR-029).

Behnam, S., Amyot, D. y Mussbacher, G. (2010). Towards a Pattern-Based Framework for Goal-Driven Business Process Modeling. In Software Engineering Research, Management and Applications (SERA)

Brandozzi, M. y Perry, D. (2003). From Goal-Oriented Requirements to Architectural Prescriptions: The Preskriptor Process. ICSE – Proc. of 2nd STRAW

Cappelli, C., Leite, J., Batista, T. y Silva, L. (2009). An aspect-oriented approach to business process modeling. Proc. of the 15th workshop on Early aspects. ACM.

Castro, J., Lucena, M., Silva, C., [et. al.] (2011). Changing attitudes towards the generation of architectural models. *Journal of Systems and Software*, 85(3).

Chung, L., Cooper, K. y Yi, A. (2002). Developing Adaptable Software Architectures Using Design Patterns: a NFR Approach. *Journal Computer Standards & Interfaces - Special issue: Adaptable software architectures*, 25.

Chung, L., y Do Prado Leite, J. (2009). On non-functional requirements in software engineering. In Conceptual modeling: Foundations and applications, 363-379.

Dermeval, D., Pimentel, J., Silva, C., [et. al.] (2012). STREAM-ADD — Supporting the Documentation of Architectural Design Decisions in an Architecture Derivation Process. Proc. 36th International Conference on Computer Software and Applications, 16-20.

Garlan, D. y Shaw, M. (1996). Software Architecture: Perspectives on an Emerging Discipline. School of Computer Science Carnegie Mellon University, Pittsburgh.

Garlan, D., Monroe, R. y Wile, D. (2010). Acme: an architecture description interchange language. In CASCON First Decade High Impact Papers. 159-173.

Grimán, A., Pérez, M., Mendoza, L. y Losavio, F. (2006). Feature Analysis for Architectural Evaluation Methods. Journal of Systems & Software, Elsevier Science (North Holland) 79(6), 871-888.

Hofmeister, C., Kruchten, P., Nord, R., [et. al.] (2005). Generalizing a model of software architecture design from five industrial approaches. In Software Architecture.

Hofmeister, C., Kruchten, P., Nord, R., [et. al.] (2007). A general model of software architecture design derived from five industrial approaches. The Journal of Systems and Software (80), 106-126.

Husnain, W. y Ahmed, Sh. (2012). A literature review and recommendations on software architecture evaluation. International *Journal of Reviews in Computing*. Vol. 10.

ISO/IEC. (2011). ISO/IEC 25010: Software engineering-Software product Quality Requirements and Evaluation (SQuaRE), Quality model.

ITUT. (2012). User Requirements Notation (URN)-Language Definition: Recommendation ITU-T Z.151. ITU, 1-206.

Jacobson, I. (1999). The unified software development process. Pearson Education India.

Kazman, R., Bass, L., Webb, M. y Abowd, G. (1994). SAAM: A method for analyzing the properties of software architectures. In Proc. of the 16th international conference on Software engineering. 81-90, IEEE Computer Society Press.

Kazman, R., Nord, R. y Klein, M. (2003). A Life-Cycle View of Architecture Analysis and Design Methods (CMU/SEI-2003-TN-026). Software Engineering Institute, Carnegie Mellon University.

Kent, S. (2002). Model driven engineering. In Integrated formal methods. 286-298, Springer Berlin Heidelberg.

Kitchenham, B. (1996). DESMET: A Method for Evaluating Software Engineering Methods and Tools. Departament of Computer Science. University of Keele. U.K.

Kiczales, G., Lamping, J., Mendhekar, A., [et. al.] (1997). Aspect-oriented programming, Proceedings of the ECOOP, Vol. 1231.

Krechetov, I., Tekinerdogan, B., Pinto, M. y Fuentes, L. (2006). Initial Version of Aspect-Oriented Architecture Design Approach. AOSD-Europe-UT-D37, D37, 1.0, (01). Recuperado de www.aosd-urope.net/deliverables/d37.pdf

Liu, L. y Yu, E. (2004). Designing information systems in social context: a goal and scenario modelling approach. Information Systems, 29(2), 187-203.

Losavio, F., Chirinos, L., y Pérez, M. (2001). Feature Analysis for quality-based architectural design methods. XI Encuentro Chileno de Computación, Jornadas Chilenas de Computación 2001 (SCCC), Punta Arenas (Magallanes), Chile, 5-10, proceedings CD-rom. Recuperado de www.umag.cl/ec

Losavio, F. y Guillén, Ch. (2010). Comparación de métodos para la arquitectura del software: un marco de referencia para un método arquitectónico unificado. *Revista de la Facultad de Ingeniería U.C.V.*, 25(1), 71–87.

Maurya, L. y Hora, H. (2010). Comparison of software architecture evaluation methods for software quality attributes. *Journal of Global Research in Computer Science*, 1(4).

OMG. (2005). Unified Modeling Language™ (UML®), Ver. 2.0. Recuperdo de http://www.omg.org/spec/UML/.

OMG. (2008). Software Process Engineering Metamodel (SPEM), Ver. 2.0. Recuperado de http://www.omg.org/spec/SPEM/2.0/PDF/.

OMG. (2011). Business Process Modeling Notation Specification, Version 2.0. OMG Document Number: formal, 01-03.

Pa, N., Jusoh, Y. y Sani, N. (2012). Representing the business perspectives in software requirement. Proc. 3rd International Conference on Business and Economic research (3rd ICBER 2012).

Rashid, A., Royer, J. y Rummler, A. (2011). Aspect-Oriented, Model-Driven Software Product Lines: The AMPLE Way. Cambridge University Press.

Schmidt, D. (2006). Model-driven engineering. IEEE Computer Society, 39(2), 25.

Scholten, F. (2007). The concern-oriented software architecture analysis method. Computer Science, Electrical Engineering, Mathematics and Computer Science, UT.

Sutton, S. y Tarr, P. (2002). Aspect-Oriented Design Needs Concern Modelling. Aspect Oriented Design Workshop at AOSD, Enschede, The Netherlands, 22.

Tekinerdogan, B., Moreira, A., Araújo, J. y Clements, P. (2004). Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design. Workshop Proceedings, 21 March, 2004, Lancaster, UK.

Tekinerdogan, B. (2004). ASAAM: Aspectual Software Architecture Analysis Method. WICSA '04 Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture, Page 5, IEEE Computer Society Washington, DC, USA 2004.

Thiel, S. (2005). A Framework to Improve the Architecture Quality of Software-Intensive Systems. Doktor der Naturwissenschaften, zur Erlangung des akademischen Grades, der Universität Duisburg-Essen.

Vanderveken, D., Van Lamsweerde, A., Perry, D. y Ponsard, Ch. (2005). Deriving Architectural Descriptions from Goal-Oriented Requirements Models. Proc. ICSE 2006, May 20-28, Shanghai, China, Copyright 2005 ACM.

Van Lamsweerde, A. (2001). Goal-Oriented Requirements Engineering: A Guided Tour, Proc. RE"01: 5th Intl. Symp. Req. Eng., Aug.

Yu, E. y Mylopoulos, J. (1993). An Actor Dependency Model of Organization Work-With Application to Business Process Reengineering. Proc. Conf. On Organizational Computing Systems (COOCS „93), Nov. 1-4, Milpitas, California, USA, 258-268.