Case Report / Reporte de Caso - Tipo 5

Difficulties and challenges in the incorporation of architectural practices

Flor de Maria Hernández Pérez, MSc(c) / fmhernandez@unicauca.edu.co

Centro de Teleinformática y Producción Industrial SENA, Popayán - Colombia

Julio Ariel Hurtado Algeria, Ph.D / ahurtado@unicauca.edu.co

Universidad Del Cauca, Popayán - Colombia

ABSTRACT The architecture software has become a key asset for software organizations because it facilitates achieving quality goals and developing of easy evolvable products. However, in small organizations, software architecture is usually a vague idea about the structure of solution. In this paper, a case of applying several methods of architecture (QAW, ADD and VaB) with small teams constituted by software developers, during a course of software development, is presented. Some difficulties to trace and correctly document the rationale associated with quality attributes, tactics architectural and selected patterns, were identified. It was established the difficulty of following an architectural process and to letconsistent evidence about that, especially when the same specification generates conflict between the established attributes and tactics and patterns that going establishing.

KEYWORDS Software architecture; QAW; ADD; quality attributes; architectural tactics; architectural patterns.

Dificultades y retos en la incorporación de prácticas de arquitectura

RESUMEN La arquitectura software se ha convertido en un activo clave en las organizaciones desarrolladoras de software, pues permite alcanzar las metas de calidad y lograr productos fácilmente evolucionables. Sin embargo, en las pequeñas organizaciones, la arquitectura de software normalmente es una idea vaga sobre la estructura de la solución. En este artículo se presenta un estudio de caso de aplicación de los métodos de arquitectura QAW, ADD y VaB, con pequeños equipos de ingenieros desarrolladores durante un curso de ingeniería de software. Se identificaron algunas dificultades para trazar y documentar correctamente el rationaleasociado a atributos de calidad, tácticas arquitecturales y patrones seleccionados. Se pudo establecer la dificultad de seguir un proceso de arquitectura y que quede evidencia consistente del mismo, más aún cuando la misma especificación genera conflicto entre los atributos establecidos y las tácticas y patrones que se van estableciendo.

PALABRAS CLAVE Arquitectura software; QAW; ADD; atributos de calidad; tácticas arquitecturales; patrones arquitectónicos.

Dificuldades e desafios na incorporação de práticas arquitectónicas

RESUMO A arquitetura de software tornou-se um elemento essencial nas organizações de desenvolvimento de software, permitindo atingir as metas de qualidade e obter produtos facilmente evoluíveis. No entanto, em pequenas organizações, arquitetura de software é geralmente uma vaga ideia sobre a estrutura da solução. Este artigo apresenta um estudo de caso de aplicação dos métodos de arquitetura QAW, ADD e VaB, com pequenas equipes de engenheiros desenvolvedores durante um curso de engenharia de software. Foram identificadas algumas dificuldades para rastrear e documentar devidamente o rationaleassociado a atributos de qualidade, táticas de arquitetura e padrões selecionados. Estabeleceu-se a dificuldade de seguir um processo de arquitetura e deixar evidência consistente do mesmo, especialmente quando a mesma especificação gera conflito entre os atributos estabelecidos e as táticas e padrões que vão se estabelecendo.

PALAVRAS-CHAVE Arquitetura de software; QAW; ADD; atributos de qualidade; táticas arquiteturais; padrões arquitetônicos.

Received / Recepción: Septiembre 11, 2016 - Accepted / Aceptación: Septiembre 26, 2016 *doi 10.18046/syt.v14i38.2290*

I. Introduction

Increasing competitiveness is a key element to the survival of the software industry (Conradi & Fuggetta, 2002). A balance between having a productive organization and quality products determines competitiveness (Cockburn, 2000). Quality in the software industry is associated with meeting the essential needs of both end users and other participants during the life cycle of software –for example, maintenance staff– (Juran, 1988). However, the urgency imposed by markets to quickly develop new systems and applications at the speed of internet, has led organizations to develop software without considering, in a disciplined way, fundamental aspects such as quality attributes (Bellomo, Nord, & Ozkaya, 2013). For example, due to the pressure of a fast delivery, organizations initially focus on implementing functionality, and then the product evolution is slow and expensive because of the limited capacity of the system to allow changes.

Some of the attributes such as usability, security and performance are directly related to the needs of the end user; however, aspects such as scalability and modifiability are related to software evolution and therefore the duration of its life cycle (Shaw & Garlan, 1996). Therefore, the speed of nowadays markets mainly prioritize the attributes visible to the end user and put aside the attributes that facilitate the evolution of software (Bellomo et al., 2013), for example to other platforms –portability–, to new requirements –extensibility– and to new infrastructure conditions –scalability–. The latter attributes are key elements to supporting the permanence of software products on the market.

In order to have a proper balance of all these quality attributes, it has emerged the concept of software architecture as a key activity within the process of software development to deliver products that have a long-lasting and adaptable life cycle and therefore, it is a key to increase the competitiveness of software organizations over time (Zachman, 1987). According to Booch et al., (2007) architecture is an activity prior to the design because their tasks are the foundation of that design. "The software architecture is the structure or set of structures of the system, that comprise software elements and the relationships between those elements externally visible, and the relationships among them" (Clements & Bass, 2010). Therefore, the software architecture defines the "skeleton" and "musculature" of any software system and is primarily responsible for achieving quality attributes of the system.

I. Introducción

Incrementar la competitividad es un factor clave para la supervivencia de la industria software (Conradi & Fuggetta, 2002). La competitividad viene dada por un equilibrio entre tener una organización productiva y productos de calidad (Cockburn, 2000). La calidad en la industria del software está asociada con la satisfacción de las necesidades esenciales, tanto de los usuarios finales, como de los otros participantes durante el ciclo de vida del software –por ejemplo, el personal de mantenimiento– (Juran, 1988). Sin embargo, la urgencia que imponen los mercados por desarrollar rápidamente nuevos sistemas y aplicaciones, a la velocidad de Internet, ha llevado a las organizaciones al desarrollar software sin considerar, en forma disciplinada, aspectos fundamentales como son los atributos de calidad (Bellomo, Nord, & Ozkaya, 2013). Por ejemplo, debido a la presión de una rápida entrega, las organizaciones se enfocan inicialmente en la implementación de la funcionalidad, y posteriormente la evolución del producto es lenta y costosa debido a la poca capacidad del sistema para dejarse modificar.

Algunos de los atributos, tales como usabilidad, seguridad y desempeño, están directamente relacionados con las necesidades del usuario final, sin embargo, aspectos como la escalabilidad y la modificabilidad están relacionados con la evolución del software y, por tanto, con la duración de su ciclo de vida (Shaw & Garlan, 1996). Por tanto, la rapidez de los mercados de hoy hace que se prioricen mayoritariamente los atributos visibles al usuario final y se dejen de lado los atributos que facilitan la evolución del software (Bellomo et al., 2013), por ejemplo hacia otras plataformas –portabilidad–, hacia nuevos requerimientos –extensibilidad– y hacia nuevas condiciones de infraestructura –escalabilidad–. Estos últimos atributos son factores clave para apoyar la permanencia de los productos software en el mercado.

Con el fin de tener un adecuado balance de todos estos atributos de calidad, ha emergido el concepto de arquitectura de software, como actividad clave dentro del proceso de desarrollo de software para entregar productos que posean un ciclo de vida duradero y adaptable y, por tanto, es un factor determinante para incrementar la competitividad de las organizaciones de software en el tiempo (Zachman, 1987). Según Booch et al., (2007) la arquitectura es una actividad previa al diseño debido a que sus tareas son el fundamento de ese diseño. "La arquitectura software es la estructura o conjunto de estructuras del sistema, que comprende elementos software y las relaciones entre esos elementos visibles externamente, así como las relaciones entre ellos" (Clements & Bass, 2010). Por tanto, la arquitectura software conforma el "esqueleto" y la "musculatura" de cualquier sistema software, y es la principal responsable de alcanzar los atributos de calidad del sistema.

En la sección 2 se presenta el estado del arte y los trabajos relacionados, donde se describen brevemente algunos de los métodos de arquitectura software propuestos por el SEI y otros que se consideran importantes dentro de la literatura en el contexto de ingeniería software; en la sección 3 se describe un estudio de caso, en el cual se aplican éstos métodos y se evalúa la calidad de las decisiones arquitectónicas, así como su documentación; finalmente, en la sección 4, se cierra el artículo con la formulación de las conclusiones y el trabajo futuro.

II. Estado de arte y trabajos relacionados

A. Métodos de arquitectura software del SEI

Durante casi tres décadas, el Software Engineering Institute [SEI] de la Carnegie Mellon University ha sido fundamental en la creación y el desarrollo del campo de la ingeniería de software conocido como arquitectura de software.

El método QAW [*Quality Attribute Workshop*] (Barbacci et al., 2003) está enfocado en el análisis y la documentación de los requerimientos que determinan la arquitectura. Estos requerimientos incluyen los atributos de calidad que son descritos a través de "escenarios" que facilitan información medible para analizar el comportamiento del sistema o de un artefacto esperado frente a un estímulo relevante para el atributo de calidad analizado.

El método ADD [*Attribute Driven Design*] (Wojcik et al., 2006), una vez que se han establecido y priorizado los escenarios, facilita realizar el diseño de la arquitectura siguiendo un enfoque recursivo de descomposición del sistema en componentes cada vez más pequeños. Durante ADD se van tomando escenarios y se van tomando decisiones de diseño, usando soluciones conceptuales llamadas "tácticas" y "patrones", que permitan satisfacer los requerimientos descritos por los escenarios a través de unas estrategias arquitectónicas.

VaB [*Views and Beyond*] (Clements et al., 2010) brinda las estrategias visuales y documentales clave para comunicar la arquitectura de software resultado de la aplicación de ADD y QAW. Por tanto, se obtienen distintas estructuras del sistema, formadas por los componentes y sus relaciones. Estas estructuras se documentan a través de "vistas" que muestran una perspectiva particular del sistema.

ATAM [*Architecture Tradeoff Analysis Method*] (Kazman, Klein, & Clements, 2000) es un método que permite revelar qué tan bien una solución arquitectónica satisface los atributos de calidad establecidos y revela, además, que riesgos, puntos sensibles y compromisos están involucrados en la arquitectura propuesta respecto de las expectativas de calidad bajo evaluación.

B. Documentación de la arquitectura

La arquitectura impulsa el desarrollo software durante todo el ciclo de vida mejorando las organizaciones y los individuos en las prácticas de ingeniería de software. John Zachman (Kazman, Gagliardi, & Wood, 2012) hace hinca-

In section 2, the state of the art and related work are presented, which briefly describes some of the methods of software architecture proposed by the SEI and others that are considered important within literature in the context of software engineering. In section 3, a case study is described in which these methods are applied and the quality of architectural decisions are evaluated as well as its documentation; finally, in section 4, the paper finishes with the formulation of conclusions and future work.

II. State of the art and related work

A. Methods of software architecture of SEI

For nearly three decades, the Software Engineering Institute [SEI] of Carnegie Mellon University has been fundamental in the creation and development of the field of software engineering known as software architecture.

The QAW [Quality Attribute Workshop] method (Barbacci et al., 2003) focuses on the analysis and documentation of requirements that determine the architecture. These requirements include quality attributes that are described through "scenarios" that provide measurable information to analyze system behavior or an expected artifact against a relevant stimulus for the analyzed quality attribute.

The ADD [Attribute Driven Design] method (Wojcik & Clements, 2006), once the scenarios have been established and prioritized, it facilitates the execution of an architecture design by following a recursive decomposition approach of the system in components increasingly smaller. During ADD, scenarios and design decisions are taken, using conceptual solutions called "tactics" and "patterns" to meet the requirements described by scenarios through architectural strategies.

VaB [Views and Beyond] (Clements et al., 2010) provides the visual strategies and key documentary to communicate the software architecture as a result of the ADD and QAV application. Therefore, different system structures are obtained formed by the components and their relationships. These structures are documented through "views" that show a particular system perspective.

ATAM [Architecture Tradeoff Analysis Method] (Kazman, Klein, & Clements, 2000) is a method that can reveal how well an architectural solution satisfies the established quality attributes and reveals which risks, sensitive points and commitments are involved in the proposed architecture in respect of the quality expectations under evaluation.

B. Documentation of architecture

The architecture drives software development throughout the life cycle improving organizations and individuals in software engineering practices. John Zachman (Kazman, Gagliardi, & Wood, 2012) emphasizes the relationship between the software structure and organization, and this interdependence suggests that there are certain levels that must be aligned. With reference to this author, reports of several organizations worldwide that have already been successful thanks to the incorporation of this architectural approach can be found (Kazman et al., 2012).

The research efforts in this area demonstrate the importance of architecture in software development. The SEI proposes a framework for documenting y communicating the architecture based on views, points of view and types of view, based on the ANSI/IEEE 1471-2000 standard, which includes recommended practices for the description of architecture of intensive software systems. The schemes that are frequently considered to represent software architecture are:

- 4+1 Views, that describes the software architecture using five concurrent views: logic view, process view, physical view, development view and the scenarios (Kruntchen 1995).
- Zachman Framework, is a framework that defines the architecture of information systems by creating a descriptive framework of disciplines highly independents of information systems, and then, by analogy, specifies the architecture of information systems based on the neutral framework and objective (Zachman 1987).
- APTIA [Analytic Principles and Tools for the Improvement of Architectures], which allows almost completely reuse analysis and improvement existing techniques, aimed to improve the architecture of a exiting system (Kazman 1996).

C. Language and tools

Normally for designing architectures are used architecture definition languages ADL [Architecture Description Languages], which provide elements for modeling the conceptual architecture of a software system, distinguishing it from its implementation (Medvidovic & Taylor, 2000). Some ADL are generic, such as UML [Unified Modeling Language] (OMG, 2011), other specific to the domain of applications. The most popular languages are ACME (Williams & Carver, 2010; Garlan, Monroe, & Wile (2000),

pié en la relación entre la estructura del software y la organización, y sugiere que en esta interdependencia hay ciertos niveles que deben estar alineados. Tomando como referencia a este autor, se pueden encontrar reportes de varias organizaciones a nivel mundial que ya ha obtenido éxito gracias a la incorporación de este enfoque arquitectónico (Kazman et al., 2012).

Los esfuerzos de investigación realizados en esta área demuestran la importancia de la arquitectura dentro del desarrollo software. El SEI propone un marco para documentar y comunicar la arquitectura con base en vistas, puntos de vista y tipos de vista, basado en el estándar ANSI/IEEE 1471-2000, el cual incluye las prácticas recomendadas para la descripción de la arquitectura de los sistemas intensivos en software. Los esquemas que se consideran con mayor frecuencia para representar la arquitectura de software son:

- 4+1 Views, que describe la arquitectura del software usando cinco vistas concurrentes: vista lógica, vista de procesos, vista física, vista de desarrollo y los escenarios (Kruntchen 1995).
- Zachman Framework, marco que permite definir la arquitectura de sistemas de información mediante la creación de un marco descriptivo de disciplinas bastante independientes de los sistemas de información, y a continuación, por analogía, especifica la arquitectura de sistemas de información basados en el marco neutral y objetivo (Zachman 1987).
- APTIA [Analytic Principles and Tools for the Improvement of Architectures), que permite reutilizar casi por completo las técnicas de análisis y mejora ya existentes, con el fin de mejorar la arquitectura de un sistema existente (Kazman 1996).

C. Lenguaje y herramientas

Normalmente para el diseño de las arquitecturas se utilizan lenguajes de definición de arquitecturas ADL [Architecture Description Languages], los cuales proveen elementos para modelar la arquitectura conceptual de un sistema software, distinguiéndola de su implementación (Medvidovic & Taylor, 2000). Algunos ADL son genéricos, como el UML [Unified Modeling Language] (OMG, 2011), otros específicos al dominio de las aplicaciones. Los más conocidos lenguajes son: ACME (Williams & Carver, 2010; Garlan, Monroe, & Wile (2000), AESOP (Garlan, 1995), C2 (Medvidović, Oreizy, & Taylor, 1997), Darwin, MetaH (Vestal, n.d; Vestal & Krueger, 2000), SAD, UniCon, Weaves y Wright (Medvidovic & Taylor, 2000).

Cada una de las etapas de desarrollo de la arquitectura es soportada por herramientas de diferente naturaleza. Para soportar la evaluación de la arquitectura se cuenta con SAAMtool (Kazman, 1996) e IDEA (Kazman et al., 2012). ArchStudio es una herramienta de código abierto basada en Eclipse para el modelado, visualización, análisis e implementación de arquitecturas de software. Softwarenaut

(Lungu, Lanza, & Nierstrasz 2014), por su parte, es una de las soluciones reportadas por la literatura para recuperar arquitecturas usando un enfoque de visualización exploratoria de software.

D. Incorporación de métodos de arquitectura al ciclo de vida

The *Open Group Architecture Framework* [TOGAF] (Josey 2011) es un *framework* de arquitectura de facto por la industria que permite incorporar la arquitectura empresarial como parte del proceso de desarrollo. Sin embargo, es un marco de gran complejidad (Gosselt, 2012), lo que dificulta su incorporación en equipos pequeños.

El SEI propone la incorporación/integración de la arquitectura (métodos QAW, ADD, ATAM, SAAM) en los ciclos de vida, tales como *Rational Unified Process* [RUP] (Sangwan, Neill, Bass, & El Houda, 2008) y *Extreme Programming* [XP]. Estos métodos muestran la importancia de incorporar las prácticas en un ciclo de vida concreto (Navarro, Fernández, & Morales, 2013; Escobar, Velandia, Ordóñez, & Cobos, 2015), para ponerlas en contexto, pero no definen los mecanismos que usaron para lograrlo y no presentan evidencia empírica de su aplicación. Muñoz y Hurtado (2012) proponen la incorporación de QAW y ADD en el contexto del método XP. Su incorporación es liviana y no permite acompañar técnicamente la apropiación completa de los métodos en los proyectos. En este trabajo se busca brindar un mecanismo integrado que incluya la descripción formal de los métodos como patrones de proceso SPEM2.0, junto con una herramienta de soporte a la aplicabilidad de los métodos en un sentido práctico.

E. El rationale

El *rationale* –la razón de fondo– se refiere a la "justificación". Reynoso (2004) lo define como la "base subyacente para la arquitectura en términos de restricciones derivadas de los requerimientos del sistema". El *rationale*, como componente de la arquitectura software, se convierte en un punto crítico para diseñarla –describir las decisiones– y evolucionarla –tomar nuevas decisiones–. Tang, Babar, Gorton, y Han (2006) exploran el valor del *rationale* desde el punto de vista del diseño, buscando documentar el conocimiento de fondo con sus decisiones de diseño y se refleja la importancia que tiene para los arquitectos documentar el *rationale* como alternativa para analizar las decisiones de diseño.

Por otra parte el *rationale* es una de las alternativas que tiene el arquitecto para justificar el diseño de la arquitectura. Este tipo de alternativa le permite al profesional de diseño razonar sobre sus decisiones; sin embargo, la mayor parte de procesos de diseño de arquitectura de un producto software no consideran explícitamente la necesidad de documentar y justificar las decisiones de diseño. De acuerdo con Bass, Clements y Kazman (2003), Bosch (2004), y Curtis, Krasner, e Iscoe (1988), son pocos quienes lo realizan, lo que lleva a preguntarse si en caso de que existieran los mecanismos apropiados para realizar la arquitectura esta situación cambiaría. Este es uno de los aspectos claves que se deben considerar en un proceso de arquitectura.

AESOP (Garlan, 1995), C2 (Medvidovic, Oreizy, & Taylor, 1997), Darwin, MetaH (Vestal, n.d; Vestal & Krueger, 2000), SAD, UniCon, Weaves and Wright (Medvidovic & Taylor, 2000).

Tools of different nature support each of the stages of development of the architecture. For supporting the evaluation of architecture, it has SAAMtool (Kazman, 1996) and IDEA (Kazman et al., 2012). ArchStudio is an open source tool based on Eclipse for modeling, visualization, analysis and implementation of software architectures. Softwarenaut (Lungu, Spear, & Nierstrasz 2014), meanwhile, is one of the solutions reported in the literature to retrieve architectures using an exploratory approach visualization of software.

D. Incorporation of architecture methods to life cycle

The Open Group Architecture Framework [TOGAF] (Josey 2011) is an architectural framework de facto for industry that can incorporate enterprise architecture as part of the development process. However, it is a framework of great complexity (Gosselt, 2012), making it difficult to incorporate it into small teams.

The SEI proposes the incorporation/integration of architecture (QAW, ADD, ATAM, SAAM methods) in the life cycles, such as Rational Unified Process [RUP] (Sangwan, Neill, Bass, & El Houda, 2008) and Extreme Programming [XP]. These methods show the importance of incorporating practices in a specific life cycle (Navarro, Fernandez, & Morales, 2013; Escobar, Velandia, Ordóñez, & Cobos, 2015), to put them in context, but do not define the mechanisms used to achieved it and do not present empirical evidence of its application. Muñoz and Hurtado (2012) propose the incorporation of QAW and ADD in the context of XP method. Their incorporation is lightweight and do not allow to accompany technically the full appropriation of the methods in projects. This paper aims to provide an integrated mechanism including the formal description of the methods as process patterns SPEM2.0, along a support tool to the applicability of the methods in a practical sense.

E. The rationale

The rationale refers to "justification". Reynoso (2004) defines it as the "underlying basis for architecture in terms of restrictions resulting from the system requirements." The rationale, as a component of the software architecture becomes a critical point for design it –describing decisions– and evolve it –taking new decisions–. Tang, Babar, Gorton, and Han (2006) explore the value of the rationale from the point of view of design, aiming to document the back-

ground knowledge with their design decisions and it reflects the importance that it has for architects to document the rationale as an alternative to analyze the design decisions.

Moreover, the rationale is one of the alternatives the architect has to justify architecture design. This type of alternative allows to the design professional reason about his or her decisions; however, most of architectural design processes of a software product do not explicitly take into account the need to document and justify design decisions. According to Bass, Clements and Kazman (2003), Bosch (2004) and Curtis, Krasner, and Iscoe (1988), only a few of them are the ones who perform it, leading to wonder if appropriate mechanisms existed to perform architecture would change this situation. This is one of the key aspects to be considered in an architecture process.

On the other hand, considering that the development of the software architecture is not an easy task, and that the outcome will depend on the experience of the architect, supporting him or her in this task by a guide mechanism, could compensate the experience that will allow an architect to address the situation methodically and applying design alternatives that he or she considers most appropriate for the system.

This paper aims to identify what knowledge should cover a supporting mechanism to the software architect to guide him or her in the architectural process and its documentation, including the rationale.

F. Adoption of architecture methods in the industry

An essential part of the architectural design are design methods and evaluation of the software architecture. The work of Losavio and Guillen (2003) provides an initial basis on which similarities and differences are identified in a certain amount of methods that are used in architectural design. This study has allowed its authors provide a framework for a unified architectural model based on quality characteristics.

The international standard IEC / ISO 12207: 2008 establishes a common framework for the processes of software life cycle, with a well-defined terminology, that can be referenced by the software industry. It applies to the acquisition of systems and software products and services for the supply, development, operation, maintenance and disposal of software products and software parts of a system, either internal or external to an organization.

III. Case study

The case study was conducted with a group of software developers engineers, most of them linked with develop-

Por otra parte, considerando que el desarrollo de la arquitectura software no es una tarea fácil, y que el resultado dependerá de la experiencia del arquitecto, apoyarlo en esta tarea, mediante algún mecanismo guía, podría compensar la experiencia que le permitirá a un arquitecto abordar la situación en forma metódica y aplicando las alternativas de diseño que el crea más convenientes para el sistema.

En este trabajo se busca identificar cuál es el conocimiento que debe cubrir un mecanismo de apoyo al arquitecto de software para guiarlo en el proceso arquitectónico y su documentación, incluyendo el *rationale*.

F. Adopción de métodos de arquitectura en la industria

Una parte esencial del diseño arquitectónico son los métodos de diseño y evaluación de la arquitectura software. El trabajo de Losavio y Guillen (2003) entrega una base inicial en la que se identifican semejanzas y diferencias en cierta cantidad de métodos que son utilizados en el diseño arquitectónico. Dicho estudio ha permitido a sus autores entregar un marco de referencia para un modelo arquitectónico unificado basado en características de calidad.

El estándar internacional IEC/ISO 12207:2008 establece un marco común para los procesos del ciclo de vida del software, con una terminología bien definida, que puede ser referenciada por la industria del software. Se aplica a la adquisición de sistemas y productos de software y servicios, para el suministro, desarrollo, operación, mantenimiento y eliminación de los productos de software y partes de software de un sistema, bien sea interna o externa a una organización.

III. Estudio de caso

El estudio de caso se llevó a cabo con un grupo de ingenieros desarrolladores de software, la mayor parte de ellos vinculados con empresas de desarrollo del suroccidente colombiano. Todos ellos son estudiantes de postgrado de la Universidad del Cauca y participaron en la investigación en el contexto de un curso avanzado de ingeniería de software focalizado en la arquitectura de software. El grupo recibió el apoyo de un experto encargado de su orientación previo al desarrollo del proyecto. Se usó como proceso contexto el proceso unificado (Rational Software, 2004), como lenguaje de modelado el UML (Rumbaugh, Jacobson, & Booch, 1999) y cómo métodos de arquitectura QAW, ADD y VaB.

A. Diseño de estudio de caso

Con la idea de evaluar la aplicabilidad de los métodos QAW y ADD en el contexto de equipos pequeños, se tomó como metodología de investigación el estudio de caso (Yin, 2009). La pregunta de investigación que orientó este trabajo fue: ¿qué tan efectivo es para los estudiantes evidenciar el *rationale* de su arquitectura utilizando QAW, ADD y VaB? La hipótesis del grupo investigador sugiere que los participantes presentan dificultades al establecer relacio-

Table 1. Case study desing / Diseño del caso de estudio

Research question / *Pregunta de investigación*	Indicador	Measurements / *Mediciones*	Instrument / *Instrumento*
Is it easy for participants to address the issue of rationale using ADD? / *¿Es fácil para los participantes abordar el tema del rationale utilizando ADD?*	GC: Level of coherence between quality scenarios, keys, chosen tactics, strategies and used patterns (rationale) / *Grado de coherencia entre escenarios de calidad, claves, tácticas escogidas, estrategias y patrones utilizados (rationale) /* *GC= (RCE *100)/RE.*	RE: Traceability relationships established / *Relaciones de trazabilidad establecidas.* REC: Traceability relationships correctly established / *Relaciones de trazabilidad correctamente establecidas.*	Architecture description document / *Documento de descripción de la arquitectura*
	GER: Level of specification of rationale / *Grado de especificación del rationale.* *GER= (DDD *100)/REC*	DDD: Traceability relationships correctly justified with the rationale / *relaciones de trazabilidad correctamente justificadas con el rationale.*	Architecture description document / *Documento de descripción de la arquitectura*

nes entre atributos de calidad, tácticas, estrategias y patrones arquitectónicos y, por tanto, dificultad para abordar el manejo del *rationale* en la descripción de la arquitectura.

De acuerdo con Yin (2009), el estudio de caso definido es de tipo exploratorio y embebido (varios equipos). El criterio de selección del caso fue ser revelatorio. Al ser el objetivo del curso obtener la arquitectura y no el sistema completo, le da un contexto rico en información arquitectónica, sin embargo, no es caso típico en la industria de software.

El caso se diseñó a través de la definición de indicadores y mediciones que permitieran responder a la pregunta de investigación y corroborar o no la hipótesis. El conjunto de indicadores y mediciones ha sido definido, tomando como referencia conceptos de arquitectura, tales como: tácticas, patrones arquitectónicos y atributo de calidad desde el punto de vista de Bass, Clements, y Kazman (2003). Estos puntos de referencia han permitido tener un punto de apoyo para poder evaluar el grado de coherencia de las decisiones de diseño que cada equipo plantee y que se consideren relevantes en el diseño arquitectónico propuesto. De acuerdo con esto, se determinaron las fuentes de información –artefactos– y se diseñaron los instrumentos para recolectar, procesar y analizar la información. Los detalles del diseño pueden verse en la **Tabla 1** –que incluye las variables que permitirán medir el *rationale* dentro del estudio de caso–, donde se calculan dos indicadores: Grado de Coherencia [GC] y Grado de Especificación del *Rationale* [GER]; el cálculo está ligado a los artefactos que, en el marco de estudio de la investigación, se consideraron clave: atributos de calidad, escenarios, tácticas, patrones arquitectónicos y *rationale*.

B. Ejecución del estudio de caso

El estudio se realizó con 18 estudiantes de la Especialización en Desarrollo de Soluciones Informáticas. Se conformaron equipos de trabajo de dos y tres personas. Para el estudio de caso se consideró involucrar prácticas de arquitectura en contextos reales. Cada equipo participante debió designar un rol para cada integrante, dentro de los cuales ment companies of the Colombian southwestern. All of them are graduate students from the University of Cauca and participated in research in the context of an advanced software engineering course focused on software architecture. An expert to their pre-project development orientation supported the group. It was used as a process context the unified process (Rational Software, 2004), as modeling language UML (Rumbaugh, Jacobson, & Booch, 1999) and as QAW, ADD and VaB architecture methods.

A. Design of case study

With the idea of evaluating the applicability of QAW and ADD methods in the context of small teams, it was taken as research methodology the case study (Yin, 2009). The research question that guided this study was how effective is for students to demonstrate the rationale of its architecture using QAW, ADD and VaB? The hypothesis of the research group suggests that participants have difficulties in establishing relationships between quality attributes, tactics, strategies and architectural patterns and therefore difficult to address management of the rationale in the description of the architecture.

According to Yin (2009), the defined case study is from exploratory and embedded type (multiple teams). The selection criteria of the case was to be revelatory. Since the aim of the course is to obtain the architecture and not the entire system, gives a rich architectural information in context, however, it is not typically the case in the software industry.

The case was designed by the definition of indicators and measurements that allow answering the question of research and corroborating the hypothesis or not. The set of

Table 2. Attributes and patterns for each group for the architecture of the information system of public notaries / Atributos y patrones por cada grupo para la arquitectura del sistema de información de notarias públicas

Team / *Equipo*	Attribute - Priority scenarios/ *Atributo - Escenarios prioritarios*	Tactics / *Táctica*	Pattern-Architectural strategy/ *Patrón-Estrategia arquitectónica*
1	Reliability / *Confiabilidad*	Information logs management / *Manejo de logs de información*	Peer to peer
		Channel encryption and management of unique encrypted key / *Encriptación del canal y manejo de llave cifrada única.*	
	Modifiability / *Modificabilida*	Separation of concerns / *Separación de intereses*	Dependency investment pattern (control inversion, programming method) / *Patrón inversión de dependencias (inversión control, método de programación).*
		Inverting of control / *Inversión del control*	
		Dependency injection / *Inyección de dependencias*	
			Facade pattern / *Patrón fachada.*
2	Reliability / *Confiabilidad*	Notifications techniques / *Técnicas de notificaciones*	Pattern / *Patrón*
		Allocation algorithms / *Algoritmos de asignación*	Publisher/*subscriber*
		Allocation control / *Control de asignación*	Publicador/*suscriptor*
			Internal client server / *Interno cliente servidor*
	Security / *Seguridad*	SQL anti-insertion techniques / *Técnicas anti-inserción SQL*	Layers / *Capas*
			Client-server / Cliente-servidor
		Restriction roles / *Restricción por roles*	
		Encryption / *Encriptamiento*	
	Usability / *Usabilidad*	Layer separation / *Separación de capas*	Model View-Controller / *Modelo vista-controlador*
		User initiative / *Iniciativa del usuario*	Presentation-abstraction- control / *Presentación-abstracción- control*
		Separation of user models / *Separación de modelos del usuario*	
		System and user / *Sistema y usuario*	
3	Availability / *Disponibilidad*	Troubleshooting tactic: Ping/Echo / *Táctica de detección de fallas: Ping/Echo*	Client-server / *Cliente-servidor*
		Recovery and preparation tactic: active redundancy / *Táctica de recuperación y preparación: redundancia activa*	
		Prevention tactic: Process Monitor / *Táctica de prevención: monitor de procesos*	
4	Security / *Seguridad*	Encryption / *Encriptación*	Layers / *Capas*
		Access control / *Control del acceso*	
5	Scalability / *Escalabilidad*	Load balancing / *Balanceo de carga*	Publisher / *subscriber*
			Publisher /*suscriptor*
			Client-server / *Cliente-servidor*
	Security / *Seguridad*	Encryption / *Encriptación*	Layers / *Capas*
6	Failover (availability) / *STolerancia a fallas (disponibilidad)*	Prevention, redundancy /*Prevención, redundancia*	Client-server / *Cliente-servidor*
			Coherence protocols / *Protocolos de coherencia.*
			Redundancy / *Redundancia*
	Modifiability / *Modificabilida*	Information concealment / *Ocultamiento de la información*	Layers, publisher, tiers / *Capas, publicador, tiers*
		Separation of concerns / *Separación de intereses*	
7	Reliability / *Confiabilidad*	a Modular independence / *Independencia Modular*	Client-server / *Cliente-servidor*
	Modifiability / *Modificabilida*	Separation of concerns / *Separación de intereses*	Layers, tiers / *Capas, tiers*

debía designarse el rol del cliente, pues se consideran importantes los aportes de este rol para el proyecto y al estudio de caso. Como se indició, el grupo de estudiantes de especialización son ingenieros que en su mayoría se desempeñan en empresas de desarrollo de la región, quienes, para la aplicabilidad del proceso de diseño arquitectónico, estuvieron apoyados por un experto que a su vez orientó un módulo de la temática previo al desarrollo del proyecto. Lo que se pretende en este primer estudio es analizar la facilidad en los estudiantes para abordar el tema del *rationale* utilizando QAW y ADD.

Se decidió abordar un solo proyecto teniendo en cuenta que los participantes solo cuentan con un máximo de cuatro fines de semana para realizar el diseño; la razón del límite de tiempo se debe a la intensidad horaria estipulada para la especialización. El proyecto planteado para el desarrollo fue un sistema de gestión de notarías públicas. Fue preciso tomar los documentos entregados incrementalmente y filtrar la información, con el objetivo de medir los indicadores en cada uno, de acuerdo con los atributos considerados como manejadores [drivers] de arquitectura. Los resultados obtenidos se pueden observar en la **Tabla 2**, que corresponde a los atributos/escenarios, tácticas y patrones determinantes para la medición de *rationale*, que fueron considerados en el estudio de Caso.

El resultado permitió la identificación de los posibles atributos de calidad que cada equipo de trabajo considera importantes, así como las tácticas y patrones que consideran adecuados para cumplir con los atributos de calidad seleccionados.

C. Resultados del estudio de caso

Tomando como base el diseño de estudio de caso, en la **Tabla 3** –que corresponde a los resultados obtenidos al medir la descripción del *rationale* propuesta por los equipos de trabajo para el sistema de notarias públicas–, se detallan los resultados obtenidos a partir de las mediciones e indicadores establecidos. En las primeras columnas se han ubicado las mediciones correspondientes RE, REC y DDD, a partir de las cuales se han obtenido los indicadores GC y GER. Los resultados observados son variados, aunque los requerimientos del problema, las indicaciones técnicas y el conocimiento fueron homogéneos entre los participantes.

IV. Discusión y conclusiones

Los resultados permiten evidenciar una variedad de situaciones al momento de determinar y documentar la arquitectura, tales como que el relacionamiento entre atributos de calidad –y sus escenarios–, tácticas y patrones arquitectónicos es una actividad compleja que se evidencia en el bajo indicador de grado de coherencia [GC]. Este mismo indicador bajo se relaciona directamente con la documentación, es decir, a menor grado de coherencia, menor grado de especificación del *rationale* [GER].

indicators and measurements have been defined, taking as reference architecture concepts, such as tactics, architectural patterns and quality attributes from the point of view of Bass, Clements, and Kazman (2003). These benchmarks have allowed having a foothold to evaluate the degree of coherence of the design decisions set out by each team that they consider relevant in the proposed architectural design. Accordingly, sources of information were determined –artifacts– and instruments to collect, process and analyze information were designed. Design details can be seen in **Table 1** –which includes the variables that will measure the rationale in the case study–, where two indicators are calculated: level of coherence [GC] and level of specification of rationale [GER]; the calculation is linked to the artifacts in the context of research study, were considered key: quality attributes, scenarios, tactics, architectural patterns and rationale.

B. Execution of case study

The study was conducted with 18 students of the Specialization in Development of IT Solutions. Teams of two and three people were formed. For the case study was considered to involve architectural practices in real contexts. Each participating team must designate a role for each member, within which was designated the role of the client, as the contributions of this role for the project and case study are considered important. As indicated above, the group of students of specialization are engineers who mostly work in development companies in the region, who, for the applicability of the architectural design process, were supported by an expert who in turn oriented a thematic module prior to the project development. The aim in this first study is to analyze the ability the students have to address the rationale using QAW and ADD.

It was decided to address a single project considering that participants only have a maximum of four weekends for the design; the reason for the time limit is due to the time stipulated for specialization. The proposed project to develop was a management system of public notaries. It was necessary to take the documents delivered incrementally and filter information with the aim of measuring the indicators in each one, according to the attributes considered as architecture drivers. The results can be seen in **Table 2**, which corresponds to the attributes / scenarios, tactics and determining patterns for measuring the rationale, which were considered in the case study.

The result allowed the identification of potential quality attributes that each team considers important and tactics and patterns considered adequate to achieve the selected quality attributes.

C. Results of case study

Based on the case study design, in **Table 3** –which corresponds to the results obtained by measuring the description of the rationale proposed by the teams for the system of public notaries–, the results obtained from established measurements and indicators are detailed. In the first columns, measurements corresponding to RE, REC and DDD have been located, from which have been obtained the GC and GER indicators. The observed results are varied, although the requirements of the problem, technical information and knowledge were homogeneous among participants.

IV. Discussion and conclusions

The results point to a variety of situations when determining and documenting the architecture, such as the relationship between quality attributes –and their scenarios–, tactics and architectural patterns is a complex activity that is evident in the low-grade indicator of the degree of coherence [GC]. This same low-grade indicator is directly related to documentation, that is, the lower degree of coherence, the lower degree of specification of the rationale [GER].

It is observed that two groups are superior to others results, which is basically because within the requirements significantly prioritized a quality attribute over others and, therefore, establish tactics and corresponding patterns as well as the rationale, it was not conflictive for them. This confirms that, despite having made a good conceptualization of the architectural aspects, establishing the architecture and a good justification for it, is not a task generally simple, so it requires a guided assistance that allow the architects to achieve a proper balance between attributes, tactics and patterns often found in the same design and generate conflict in decision making.

Through the case study could establish the difficulty of following an architecture process and that there is consistent evidence of it, especially when the same specification comes to generating conflict between the various attributes established and tactics and patterns to be established. Therefore, it is important to have an instrument to guide and support the implementation of architecture

Table 3. Results / Resultado

Team/ *Equipo*	RE	REC	DDD	GC= (RCE *100)/RE	GER= (DDD *100)/RE
1	4	2	2	50%	50%
	9	3	3	33%	33%
Total	13	5	5	38.5%	38.5%
2	3	3	3	100%	100%
	2	2	2	100%	100%
Total	5	5	5	100%	100%
3	9	3	0	33%+	0%
	9	4	6	44%	66%
	12	8	8	66%	66%
Total	21	12	14	57%	66%
4	4	4	4	100%	100%
Total	4	4	4	100%	100%
5	4	4	4	100%	100%
Total	4	4	4	100%	100%
6	4	2	2	50%	50%
	12	4	4	16%	33%
Total	16	6	6	37%	37.5%
7	2	0	2	0%	50%
	4	2	4	50%	100%
Total	6	2	6	33%	100%

Se puede ver que dos grupos tienen resultados superiores a los demás, lo que se debe, básicamente a que dentro de los requerimientos priorizaron significativamente un atributo de calidad sobre los demás y, por tanto, establecer las tácticas y los patrones correspondientes, así como el *rationale*, no les resultó conflictivo. Esto corrobora que, a pesar de haber realizado una buena conceptualización sobre los aspectos arquitectónicos, establecer la arquitectura y una buena justificación de la misma no es una tarea en general sencilla, por lo que se requiere de una asistencia guiada que les permita a los arquitectos ir logrando un adecuado balance entre atributos, tácticas y patrones que muchas veces se encuentran en un mismo diseño y generan conflicto en la toma de decisiones.

Por medio del estudio de caso se pudo establecer la dificultad de seguir un proceso de arquitectura y que quede evidencia consistente del mismo, más aún cuando la misma especificación llega a generar conflicto entre los diferentes

atributos establecidos y las tácticas y patrones que se van estableciendo. Por lo tanto, es importante contar con un instrumento que oriente y apoye la aplicación de prácticas de arquitectura dentro de una organización, sin embargo, es pertinente seguir la investigación en otros contextos que permitan reafirmar los resultados obtenidos hasta el momento. Además, la documentación y el *rationale* consistente dentro de los grupos que tomaron decisiones respecto a un solo atributo de calidad, refleja la importancia de que dicho instrumento facilite una integración con métodos de la arquitectura, que en este estudio de caso fueron: QAW, ADD y VaB.

El trabajo a futuro se centra en la realización de otros estudios de caso que aporten a definir un mecanismo para facilitar el desarrollo de la arquitectura, centrado en una guía metodológica, y permitan una base de conocimiento basada en la literatura sobre el relacionamiento concreto de los elementos clave de la arquitectura: atributos de calidad, escenarios de calidad, tácticas y patrones arquitectónicos, y el *rationale* derivable. En los estudios de caso futuros se espera involucrar el sector empresarial con el fin de avanzar en dicho instrumento de apoyo. S&T

practices within an organization; however, it is relevant to continue research in other contexts that allow reaffirm the results obtained so far. In addition, documentation and the consistent rationale within the groups taking decisions on a single attribute of quality, reflects the importance that said instrument facilitates integration with methods of architecture, which in this case study were: QAW, ADD and VaB.

Future work focuses on the realization of other case studies that contribute to define a mechanism to facilitate the development of architecture, focusing on a methodological guide and allowing a knowledge base based on the literature on the specific relationship of key elements of architecture such as quality attributes, quality scenarios, tactics and architectural patterns, and the derivable rationale. In future case studies are expected to involve the business sector in order to advance in said support instrument. S&T

References / Referencias

ANSI/IEEE 1471-2000. IEEE Recommended practice for architectural description for software-intensive. Retrieved from: https://standards.ieee.org/findstds/standard/1471-2000.html

Barbacci, M., Ellison, R., Lattanze, A., Stafford, J., Weinstock, C., & Wood, W. (2003). *Quality attributes workshops* [3rd ed. - technical report CMU/SEI-2003-TR-016 ESC-TR-2003-016]. Pittsburgh, PA: Carnegie Melon University. Available at: http://www.sei.cmu.edu/reports/03tr016.pdf

Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice*. Boston, MA: Addison-Wesley.

Bellomo, S., Nord, R., & Ozkaya, I. (2013). A study of enabling factors for rapid fielding combined practices to balance speed and stability. In: *Proceedings of the ACM-IEEE International Conference on Software Engineering* (pp. 982-991). Los Alamitos, CA: IEEE Computer Society.

Berlin-Heidelberg: Germany.

Booch, G., Maksimchuk, R., Engle, M., Young, B., Conallen, J., & Houston, K. (2007). *Object-oriented analysis and design with applications*. Upper Saddle River, NJ: Addison-Wesley

Bosch, J. (2004). *Lecture Notes in Computer Science Vol. 3047. Software architecture: The next step* (pp. 194-199).

Clements, P. & Bass, L. (2010). Using business goals to inform software architecture. In *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference, RE2010*, (pp. 69-78). IEEE.

Clements, P., Bachman, F., Bass, L., Garlan, D., Ivers, J., Little, R., ... Stafford, J. (2010). *Documenting software architectures: Views and beyond* [2nd ed.]. Boston, MA: Pearson.

Cockburn, A. 2000. Selecting a project's methodology. *IEEE Software*, *17*(4), 64-71.

Conradi, R. & Fuggetta, A. (2002). Improving software process improvement. *IEEE Software, 19*(4), 92-99.

Curtis, B., Krasner, H., & Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM, 31*(11), 1268-1287.

Escobar, A., Velandia, D. Ordoñez, H. & Cobos, C. (2015).A review of the impact on XP methodology of business model inclusion in requirements elicitation. *Sistemas & Telemática, 13*(33), 45-61. http://dx.doi.org/10.18046/syt.v13i33.2080

Garlan, D. (1995). *An introduction to the Aesop system*. Retrieved from: http://www.cs.cmu.edu/afs/cs/project/able/ftp/aesop-overview.pdf

Garlan, D., Monroe, R., Wile, D. (2000). ACME: Architectural description of component-based systems. In: G. Leavens & M. Sitaraman [Eds.], *Foundations of component-based systems* (pp. 47-68). Cambridge, UK: Cambridge University Press

Gosselt, R. W. (2012). A maturity model based roadmap for implementing. In: *17th Twenty Student Conference on IT [online].* Retrieved from: http://referaat.cs.utwente.nl/conference/17/paper/7341/a-maturity-model-based-roadmap-for-implementing-togaf.pdf

Yin, R. (2009). *Case study research*. London, UK: Sage.

ISO/IEC 12207:2008. Systems and software engineering -- Software life cycle processes. Geneva, Switzerland: ISO/IEC. Available at: http://www.iso.org/iso/catalogue_detail?csnumber=43447

Josey, A. (2011). *TOGAF Version 9.1 Enterprise Edition.* Reading, UK: The Open Group.

Juran, J. M. (1988). *Juran on planning for quality*. New York, NY: Free press.

Kazman, R. (1996). Tool support for architecture analysis and design. In: *Joint proceedings of the Second International Software Architecture Workshop (ISAW-2) and International Workshop on Multiple Perspectives in Software Development (Viewpoints '96) on SIGSOFT '96 Workshops* (pp. 94-97). New York, NY: ACM. doi: 10.1145/243327.243618

Kazman, R., Gagliardi, M., & Wood, W. (2012). Scaling up software architecture analysis. *Journal of Systems and Software, 85*(7), 1511-1519. doi:10.1016/j.jss.2011.03.050.

Kazman, R., Klein, M., & Clements, P. (2000).ATAM: *Method for architecture evaluation* (No. CMU/SEI-2000-TR-004). Pittsburgh PA: Carnegie-Mellon University.

Kruchten, P. (1995). Architectural blueprints—The "4+ 1" view model of software architecture. *Tutorial Proceedings of Tri-Ada, 95*, 540-555.

Losavio, F. & Guillén, D. (2003). Diseño arquitectónico de sistemas distribuidos. *Rapide, 15*(1). Retrieved from: http://www.dcc.uchile.cl/~mmarin/revista-sccc/sccc-web/Vol5/wis1.pdf

Lungu, M., Lanza, M., & Nierstrasz, O. (2014). Evolutionary and collaborative software architecture recovery with softwarenaut. *Science of Computer Programming, 79*, 204-223.

Medvidovic, N. & Taylor, R. (2000). A classification and comparison framework for software architecture description languages. *IEEE Software Engineering*, *26*(1), 70-93.

Medvidović, N., Oreizy, P., & Taylor, R. (1997). Reuse of off-the-shelf components in c2-style architectures. In: *Software Engineering (ICSE 1997), 19th International Conference on,* (pp. 692-700). doi:10.1109/ICSE.1997.610496.

Muñoz, L. & Hurtado, J. (2012). XA: An XP extension for supporting architecture practices. In: *2012 7th Colombian Computing Congress, CCC 2012* (p. 5). IEEE. doi:10.1109/ColombianCC.2012.6398012

Navarro, A., Fernández, J., & Morales, J. (2013). Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva, 11*(2), 30-39.

OMG. (2011). *OMG unified modeling language (OMG UML), superstructure, version 2.4.1.* Retrieved from: http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/

Rational Software (2004). *Rational unified process: Best practices for software development teams* [white paper TP026B, Rev 11/01]. Cupertino, CA: Rational Software. Available at: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Reynoso, C. (2004). *Introducción a la arquitectura de software.* Recuperado de: http://carlosreynoso.com.ar/archivos/carlos-reynoso-introduccion-a-la-arquitectura-de-software.pdf

Rumbaugh, J., Jacobson, I., & Booch, G. (1999). *El lenguaje unificado de modelado: manual de referencia.* Madrid, España: Addison-Wesley.

Sangwan, R., Neill, C., Bass, M., & El Houda, Z. (2008). Integrating a software architecture-centric method into object-oriented analysis and design. *Journal of Systems and Software, 81*(5), 727-746.

Shaw, M. & Garlan, D. (1996). *Software architecture: Perspectives on an emerging discipline*. Upper Saddle River: Prentice Hall.

Tang, A., Babar, M. A., Gorton, I., & Han, J. (2006). A survey of architecture design rationale. *Journal of systems and software, 79*(12), 1792-1804.

Vestal, S. & Krueger, J. (2000, Jan. 1). *Technical and historical overview of MetaH*. Retrieved from: http://aadl.sei.cmu.edu/aadl/documents/Technical%20and%20Historical%20Overview%20of%20MetaH.pdf

Vestal, S. (n.d). *The MetaH AADL toolset*. Retrieved from: http://www51.honeywell.com/aero/technology/common/documents/MetaH.pdf

Williams, B. & Carver, C. (2010). Characterizing software architecture changes: A systematic review. *Information and Software Technology 52*(1), 31-51. http://dx.doi.org/10.1016/j.infsof.2009.07.002.

Wojcik, R. Bachmann, F., Bass, L., Clements, P., Merson, P., Nord, R., & Wood, B. (2006, November). *Attribute driven design (ADD), versión 2.0* [technical report CMU/SEI-2006-TR-023 ESC-TR-2006-023]. Pittsburgh, PA: Carnegie Mellon University. Available at: http://www.sei.cmu.edu/reports/06tr023.pdf

Zachman, J. (1987). A framework for information systems architecture. *IBM Systems Journal, 26*(3): 454-470.

CURRICULUM VITAE

Flor de María Hernandez Perez Systems Engineer from Universidad Cooperativa de Colombia (2007); Specialist in Computer Solutions Development from Universidad del Cauca (2012); and candidate to Master of Computer Science at the University of Cauca. Member, since 2014, of IDIS research group, focused in research and development related with software engineering, at the Universidad del Cauca, and Innovatec, at the Centro de Teleinformática y Producción Industrial (SENA-Cauca), since 2015, where she also serves as a professor of higher education and as an instructor. / Ingeniera de Sistemas de la Universidad Cooperativa de Colombia (2007), Especialista en Desarrollo de Soluciones Informáticas de la Universidad del Cauca (2012) y candidata a Magister en Computación de la Universidad del Cauca, donde además es miembro del grupo de Investigación IDIS [Investigación y Desarrollo de Ingeniería de Software] desde 2014. Es integrante también del grupo de investigación Innovatec del Centro de Teleinformática y Producción Industrial del SENA, regional Cauca, desde 2015. Se desempeña como docente de Educación Superior y como instructora del SENA en el Centro de Teleinformática y Producción Industrial (regional Cauca).

Julio Ariel Hurtado Alegría SFull professor at the University of Cauca (Department of Systems) and member of IDIS (research and development in software engineering), since 2005. He received his Ph.D in Computing Science from the Universidad de Chile (2012) and his degree in Electronics and Telecommunications Engineering from Universidad del Cauca (1997). His main research topics are: software engineering, software process models, software architecture, engineering models drivers and software product lines. His work has focused on applying techniques and approaches MDE SPL for designing and analysis of software process models / Profesor titular del Departamento de Sistemas de la Universidad del Cauca. Es miembro del grupo de Investigación y Desarrollo de Ingeniería de Software [IDIS] desde 2005. Recibió su doctorado en Ciencias de la Computación de la Universidad de Chile (2012) y su grado en Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca (1997). Sus principales tópicos de investigación son: ingeniería de software, modelos de procesos software, arquitectura de software, controladores de modelos de Ingeniería y líneas de productos software. Su trabajo se ha enfocado en aplicar el uso de técnicas MDE y enfoques SPL para el diseño y análisis de modelos de procesos software.