Técnicas de Visualización para Conocimiento Arquitectónico: una Evaluación Empírica

Cristina Roda[1], Elena Navarro[1], Carlos E. Cuesta[2] y Dewayne E. Perry[3]

[1]Departamento de Sistemas Informáticos, Universidad de Castilla-La Mancha,
cristinarodasanchez@gmail.com, elena.navarro@uclm.es
[2]Departamento LSI2, Universidad Rey Juan Carlos de Madrid,
carlos.cuesta@urjc.es
[3]Department of ECE, The University of Texas at Austin,
perry@ece.utexas.edu

Resumen. Investigaciones recientes destacan la necesidad de capturar y representar las decisiones de diseño arquitectónico como elemento clave del conocimiento arquitectónico. A pesar de la variedad de herramientas que permiten la visualización de este tipo de conocimiento, todavía existe cierta falta de madurez. En este artículo, presentamos un conjunto de técnicas de visualización, de acuerdo a su forma de representación, aplicables al conocimiento arquitectónico. Estas técnicas son analizadas, considerando sus fortalezas y debilidades, mediante una evaluación empírica. Los resultados de esta evaluación sugieren algunas ideas para el trabajo futuro en técnicas de visualización que pueden mejorar la representación del conocimiento arquitectónico.

Palabras clave: Conocimiento Arquitectónico, Técnicas de Visualización, Evaluación Empírica, Casos de Estudio Exploratorios

1 Introducción

El desarrollo de software tiene que hacer frente a múltiples retos, como la complejidad del sistema, las cualidades no funcionales, las operaciones de mantenimiento, la producción distribuida, los cambios frecuentes de personal, etc. [1] Además, las compañías software con altos costes de mantenimiento demandan, cada vez más, diseños flexibles y fáciles de mantener [1].

La *Arquitectura Software* (AS) es un artefacto clave en el ciclo de vida que permite a las compañías de software representar y comunicar la estructura y comportamiento del sistema a los stakeholders del mismo [2]. Hasta hace poco, las investigaciones en AS han centrado sus esfuerzos, principalmente, en el desarrollo de *Lenguajes de Descripción de la Arquitectura* (LDA, [3]) que permiten, tanto la descripción de *elementos arquitectónicos*, como de la *estructura* de la AS. Sin embargo, tal y como describieron Perry y Wolf [4], hay un tercer tipo de elemento importante en la descripción que es la *rationale*, es decir, la motivación para seleccionar el estilo, los

elementos arquitectónicos, etc. Prácticamente, no se ha prestado atención a éste tercer elemento, hasta hace aproximadamente siete años [5] cuando el área de *Conocimiento Arquitectónico* (CA) ha surgido destacando la importancia de las *Decisiones de Diseño Arquitectónico* (DDAs) y las *Rationales de Diseño Arquitectónico* (RDAs). Tanto las DDAs como las RDAs son aspectos esenciales del CA, cuyo modelado, gestión y uso compartido se ha convertido en un foco importante de investigación.

En este contexto, cabe destacar que, siempre que se registre y documente explícitamente una decisión de diseño, surgen nuevas actividades durante el proceso arquitectónico. Esta información sobre el CA constituye una nueva vista transversal que se superpone a la información descrita por otras vistas [1]. Por tanto, la introducción y explotación de técnicas de visualización apropiadas se convierte en un elemento esencial, permitiendo a los diferentes stakeholders navegar a través de las diferentes vistas del sistema.

Actualmente, hay múltiples técnicas de visualización disponibles para representar el CA. Tal y como han señalado Kruchten et al. en [1], se está llevando a cabo una investigación muy activa, que produce una cantidad importante de enfoques para representar y capturar DDAs. Por ejemplo, varias aproximaciones utilizan *listas de atributos plantilla* como elemento esencial para describir y representar DDAs. Una de estas aproximaciones [6] enfatiza lo importante que es clasificar diferentes tipos de dependencias entre decisiones, ya que supone información valiosa y complementaria para capturar trazas útiles. Otro de estos enfoques [7] aboga por utilizar aproximaciones más flexibles que empleen atributos obligatorios y opcionales para capturar el CA, de forma que pueda ser personalizado según las necesidades de la organización. Alternativamente, otros autores [6] han propuesto ontologías para formalizar dicho conocimiento tácito y hacer visible las relaciones entre DDAs y otros artefactos del ciclo de vida del software. Finalmente, el campo de las líneas de producto software [1], también ha generado cuantiosos trabajos sobre especificación, modelado y automatización de DDAs, utilizados para describir y seleccionar elementos comunes y variables de una línea de producto.

Por tanto, durante los últimos años, han aparecido una gran variedad de enfoques distintos, aunque pocos permiten analizar sus fortalezas y debilidades para que los analistas tengan alguna guía útil a la hora de tomar decisiones sobre cuál es la mejor alternativa para sus proyectos. El propósito de este artículo es proporcionar una visión sobre las fortalezas y debilidades de estas aproximaciones en cuanto a las diferentes técnicas de visualización de CA.

Este artículo se estructura como sigue. En la sección 2, se describe el experimento llevado a cabo, es decir, la evaluación empírica realizada. En la sección 3, se presenta la clasificación de las técnicas de visualización utilizada en este artículo, junto con la identificación de qué herramientas encajan en cada una de estas técnicas. En la sección 4, se presenta el material experimental utilizado para llevar a cabo la evaluación empírica, cuyo análisis se presenta en la sección 5. Finalmente, en el apartado 6, se exponen las distintas conclusiones y el trabajo futuro.

2 Descripción del Estudio Empírico

La evaluación empírica realizada se ha planteado como casos de estudio exploratorios [8] en donde cada arquitecto software ha realizado cambios sobre la AS de un sistema real (véase sección 4), utilizando las distintas herramientas seleccionadas (véase sección 3) que ofrecen soporte a las diferentes técnicas de visualización analizadas. Después de completar el conjunto de cambios, el arquitecto debía evaluar la utilidad de cada una de las técnicas de visualización. A continuación, se presenta la estructura del estudio realizado.

2.1 Objetivo del estudio

El objetivo de este estudio es evaluar qué técnica de visualización es la más efectiva en el uso del CA al llevar a cabo un conjunto de cambios - esto es, en la toma de DDAs - sobre una AS existente.

Nuestra hipótesis nula, si estuviéramos haciendo un experimento controlado, sería que no hay diferencia en cuanto a la efectividad de las distintas herramientas de visualización, es decir, todas serían igualmente efectivas. Nuestra intuición, sin embargo, es que la técnica *Node-link and tree* (véase sección 3.3) es la más efectiva a la hora de representar el CA, dado que su vista gráfica proporciona más información que el resto de técnicas. Aunque los diferentes casos de estudio exploratorios soportan individualmente esta intuición, expondremos, no obstante, un análisis global de los datos, cuyo resultado soporta claramente esta conclusión.

2.2 Construcciones del estudio

La variable independiente (o entrada) en estos casos de estudio es el *CA de la arquitectura del sistema EFT*, representado por las distintas herramientas de visualización (véase sección 3).

La variable dependiente (o salida, según lo determinado por los diversos tratamientos o cambios en la arquitectura) es la *efectividad* de las distintas técnicas de visualización como respuesta a los diferentes tratamientos (véase subsección 2.4). La efectividad viene determinada por: (i) la *utilidad* de la herramienta; (ii) la *facilidad de uso* de la misma; (iii) la *facilidad de aprendizaje*; y (iv) la *satisfacción* con dicha herramienta. El cuestionario estándar USE [9] ha sido utilizado para determinar la efectividad de las distintas técnicas de visualización para representar el CA del sistema elegido. Para cada técnica de visualización se creó un cuestionario asociado, utilizando la plataforma electrónica de aprendizaje *Moodle*, la cual está disponible en la Universidad de Castilla-La Mancha para apoyo a la enseñanza. Esta plataforma nos permitió recolectar todos los resultados de los cuestionarios para su posterior análisis.

2.3 Diseño del estudio

El grupo de estudio estaba formado por 15 estudiantes pertenecientes al último curso del Grado en Ingeniería Informática de la Universidad de Castilla-La Mancha, cuyas edades estaban comprendidas entre 22 y 25 años. Cada estudiante llevó a cabo el caso de estudio en dos fases. Durante la primera fase, cada uno realizó diferentes tareas que son, por lo general, llevadas a cabo por los arquitectos cuando evolucionan un sistema. Estas tareas estaban asociadas con la modificación del CA o AS del sistema objetivo descrito en la sección 4. Durante la segunda fase, cada estudiante evaluó la efectividad de las distintas técnicas, utilizando el cuestionario disponible USE.

Los casos de estudio exploratorios se llevaron a cabo en el contexto de una sesión práctica (dos horas) de la asignatura *Interacción Persona Ordenador II*, por lo que los estudiantes ya estaban familiarizados con términos como (usabilidad) efectividad y técnica de visualización. La información recogida durante el experimento se analiza en la sección 5.

2.4 Tratamientos del estudio

A continuación, se presentan los antecedentes de la AS del sistema EFT (sistema de control bancario utilizado en esta evaluación) y los aspectos básicos para crear dicha AS y su CA asociado:

- La selección del sistema y la plataforma de la AS es una de las cuestiones arquitectónicas más relevantes. El arquitecto debe tener en cuenta que el sistema EFT ofrece *soporte a la resistencia a fallos* (*fault-resilient support*), por lo que ha de incorporar un sistema para procesamiento continuo con pocas posibilidades de fallo. Hay dos opciones posibles: un *sistema resistente a fallos* que tenga siempre un nodo listo para asumir el control si otro nodo falla; o un *sistema tolerante a fallos* que tenga módulos de procesamiento de copia de seguridad incorporados. Para el sistema EFT, los arquitectos seleccionaron un sistema resistente a fallos porque cumplía con los requisitos de confiabilidad del cliente, evitando el alto coste de un sistema tolerante a fallos, que lo convertía en un candidato poco atractivo. Sin embargo, el sistema resistente a fallos conlleva una desventaja: se requieren otros productos asociados a la plataforma para mantener el tiempo de actividad un 99.95% de las veces. De esta forma, el arquitecto tenía que tomar otra decisión: qué estrategias de recuperación era necesario implementar.
- Los arquitectos también tuvieron que prestar atención a la *confiabilidad de la red*, ya que las operaciones bancarias deben llevarse a cabo en un entorno seguro. Para ello, consideraron dos opciones: introducir un *enlace frame-relay* en el sistema; o introducir otra *línea frame-relay* como copia de seguridad. Los arquitectos seleccionaron la primera opción porque permitía la marcación de los bancos miembros a través de la *Red Telefónica Conmutada Pública*, mientras que la segunda opción era poco económica y más arriesgada, dado que la línea frame-relay de copia de seguridad también podía fallar.
- Otro aspecto importante era el *fallo de potencia*, dado que el sistema EFT tenía que proporcionar un servicio continuo, por lo que se hacía necesario disponer de

un suministro de potencia secundario. Se evaluaron dos alternativas: un *Suministro de Potencia Ininterrumpido* (UPS); o un generador de potencia. Dado que la segunda alternativa requería un presupuesto más alto, los arquitectos seleccionaron la opción UPS.

– Para responder ante desastres naturales, como terremotos o incendios, que podrían dañar el centro de procesamiento, los arquitectos tuvieron que seleccionar un mecanismo adecuado: un *sitio remoto*, que pudiera asumir el control de procesamiento; o *procedimientos manuales*. Finalmente, los arquitectos optaron por procedimientos manuales, porque no había suficiente presupuesto para la primera opción.

Dado este diseño del sistema, se crearon dos estructuras en cada herramienta, asociadas al CA del sistema EFT: una desde el punto de vista de los requisitos, y otra desde el punto de vista de los elementos arquitectónicos. Utilizando el CA del sistema EFT, representado con las distintas técnicas de visualización, los sujetos tuvieron que realizar las siguientes tareas:

– Tarea 1: Se notificó a los sujetos que el cliente ya no presentaba problemas monetarios, por lo que el sistema EFT tenía que ser evolucionado para implementar las mejores alternativas. Por esta razón, la primera modificación era cambiar el requisito *Solución de Coste Efectivo* por *La Mejor Solución*.
– Tarea 2: Se añadió un nuevo requisito, *Monitorización 24h*, que permitía al sistema estar al tanto de cualquier problema en el suministro de potencia o fallos de computación o comunicación. La incorporación de este nuevo requisito afectó a todas las RDAs iniciales de la arquitectura definidas para el sistema.
– Tarea 3: El último cambio consistía en que la *base de datos ORACLE* tenía que ser reemplazada por *MySQL*, dado que el cliente estaba apostando por el software de código abierto.

Como se puede observar, estas tres tareas permitieron que el grupo de estudio navegara a través de las estructuras del CA, y modificaran lo que consideraran más adecuado, teniendo en cuenta que los dos primeros cambios afectaban a la estructura de los requisitos, y el último cambio a la estructura de los elementos arquitectónicos.

2.5 Implementación

Se preparó una máquina virtual que permitió a cada sujeto llevar a cabo su caso de estudio. Estaba equipada con el sistema operativo Windows XP, y todas las herramientas de visualización seleccionadas. A continuación, se copió dicha máquina virtual en 15 ordenadores Dell™ Inspiron One 19. Los casos de estudio exploratorios fueron ejecutados por el grupo de estudio en un laboratorio, utilizando el equipo descrito, y fueron llevados a cabo en tres fases diferentes:

– Primeramente, se proporcionó una introducción al experimento, describiendo su objetivo principal, el caso de estudio a utilizar, y las tareas a realizar.
– En segundo lugar, se realizó una breve introducción a cada una de las herramientas seleccionadas para que los sujetos adquirieran las habilidades necesarias para su manipulación.

– En tercer lugar, el grupo de estudio realizó cada una de las tareas, utilizando las distintas técnicas y, seguidamente, rellenaron el cuestionario sobre la efectividad.

Cabe destacar que, en cada fase, se notificó a los sujetos que el principal objetivo del experimento era evaluar la técnica de visualización y no la herramienta en sí.

2.6 Amenazas a la validez

La *validez de construcción* es fuerte. Los distintos sujetos comprendían bien la efectividad de la usabilidad y sus medidas constituyentes. Además, el uso de una arquitectura bien descrita junto a su CA representado con técnicas de visualización bien comprendidas, también tiene una fuerte validez de construcción.

La *validez interna* no es tan fuerte debido al empleo de estudiantes para cada caso de estudio, en lugar de arquitectos software experimentados. Serán necesarios más estudios con arquitectos experimentados para ver si sus evaluaciones de la efectividad coinciden con las de los estudiantes. Sin embargo, las razones subyacentes para las evaluaciones actuales de efectividad sugieren que estos estudios tendrían resultados congruentes con nuestros estudios.

La fortaleza de la *validez externa* reside en el uso de una AS realista y su CA, y en realizar múltiples estudios. Su debilidad es análoga a la de la validez interna en el sentido de que los casos de estudio son realizados por estudiantes, pero con expectativas de resultados congruentes en otros estudios con arquitectos experimentados. Por tanto, consideramos que la validez externa, en general, es muy buena.

3 Técnicas de Visualización para Conocimiento Arquitectónico

Se pueden utilizar diversas técnicas de visualización para capturar, representar o mantener el CA. Queremos resaltar que una de las bases de nuestra investigación es que el CA es, per se, una sistema de conocimiento compuesto por DDAs y RDAs y sus correspondientes relaciones, las cuales pueden ser utilizadas para comprender y reflexionar sobre la AS de un sistema. En este sentido, se asemeja a una ontología [10], como otros autores ya han notado [11]. Esto nos ha llevado a utilizar como taxonomía de técnicas de visualización aquella propuesta por Katifori et al. [12], la cual distingue cinco tipos de representación diferentes, dependiendo de la presentación de la información, el método de interacción, o la funcionalidad soportada. En las siguientes subsecciones, se describe cada uno de estos tipos, junto a sus herramientas disponibles.

Cabe destacar que este artículo se centra en herramientas bidimensionales, principalmente porque resultan familiares, es decir, similares a las usadas comúnmente por los arquitectos. Las subsecciones 3.1, 3.2 y 3.3 presentan herramientas de visualización de CA, mientras que los apartados 3.4 y 3.5 presentan herramientas ontológicas. Hemos incluido las últimas dos técnicas porque presentan la información de una forma jerárquica, muy similar a las representaciones utilizadas por herramientas específicas de CA. Nótese que se prefieren las herramientas de CA,

siempre que estén disponibles para una técnica de visualización particular; si no existen herramientas de CA disponibles, se prefieren las ontológicas frente a cualquier otro tipo.

3.1 Indented list

De acuerdo a este tipo de representación, el CA es representado mediante texto plano en una vista de árbol, similar al explorador de Windows, es decir, una *lista indentada*. La simplicidad de esta representación textual hace que este método no sea muy popular hoy día para representar CA. *PHI* (*Procedural Hierarchy of Issues*) [13] es un sistema CA que usa esta técnica de visualización, el cual extiende al sistema IBIS y presenta un enfoque de argumentación para resolver problemas. Algunas herramientas que soportan la metodología PHI son [14]: *JANUS* [14,15], *HOS* (*Hyper-Object Substrate*) [14], y *PHIDIAS* (*Procedural Hierarchy of Issues/Design Intelligence Augmentation*) [14]. Sin embargo, actualmente, estas herramientas no tienen ningún soporte, por lo que se van a considerar herramientas ontológicas que ofrezcan una vista de árbol similar al explorador de Windows.

Protégé [16] es un entorno de edición de ontologías y adquisición de conocimiento (véase Fig. 1). *KAON* [17] es un entorno de gestión de ontologías de código abierto para aplicaciones de negocio. *OntoRama* [18] permite a los usuarios visualizar una estructura (ontológica) de conocimiento en un diseño hiperbólico. *OntoEdit* [19] es un entorno de ingeniería de ontologías que combina el desarrollo de ontologías basadas en metodologías con capacidad para colaboración e inferencia. *Protégé* es la herramienta seleccionada para ser evaluada, dado que es de código abierto y ha sido citada por otros autores [12,20] en el contexto de CA.

3.2 Wiki

Como indican Farenhorst et al. [21], una *wiki* permite a los diseñadores y arquitectos colaborar y comunicarse entre sí fácilmente. Por tanto, la información puede actualizarse rápidamente, y los stakeholders pueden saber en cualquier momento el estado actual del proyecto. Algunas herramientas de este tipo son: *C-ReCS*

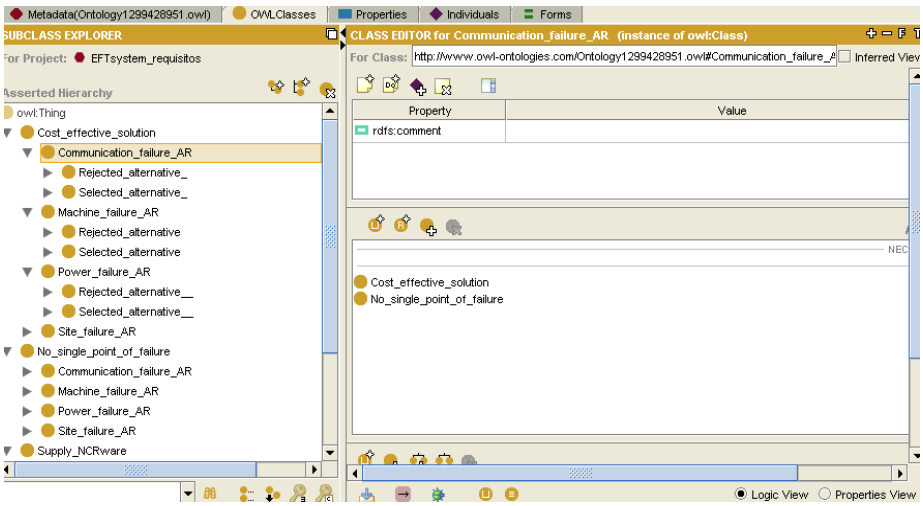

Fig. 1. Protégé 3.4.4

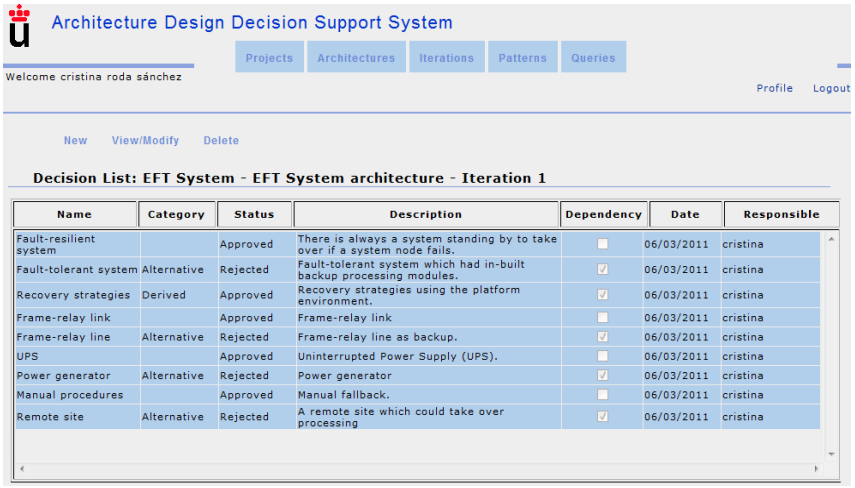

Fig. 2. ADDSS 2.0

(*Collaborative Requirements Capture System*) [22]; *PAKME* (*Process-based Architecture Knowledge Management Environment*) [2]; *ADDSS* (*Architecture Design Decision Support System*) [2,23] (véase Fig. 2); y *Knowledge Architect* [2,23].

En este caso, *ADDSS* ha sido la herramienta seleccionada para llevar a cabo la evaluación presentada en la sección 2. Su elección frente a otras herramientas viene motivada porque es la más completa y proporciona un sistema de consultas que permite a los arquitectos encontrar fácilmente información.

3.3 Node-link and tree

Este enfoque proporciona una representación de nodos interconectados, que permite a los usuarios expandir/contraer nodos, regulando el nivel de detalle de la información. Algunas herramientas correspondientes a esta categoría son: *QOC* (*Questions,*

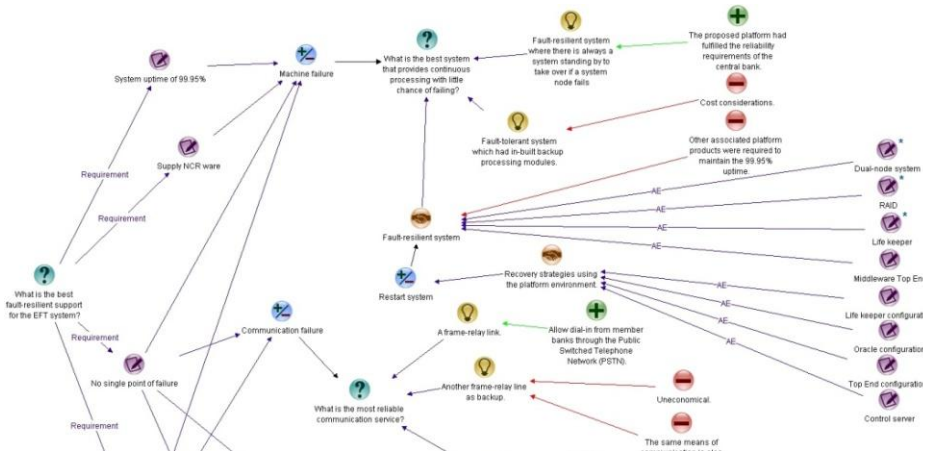

Fig. 3. Compendium 1.5.2

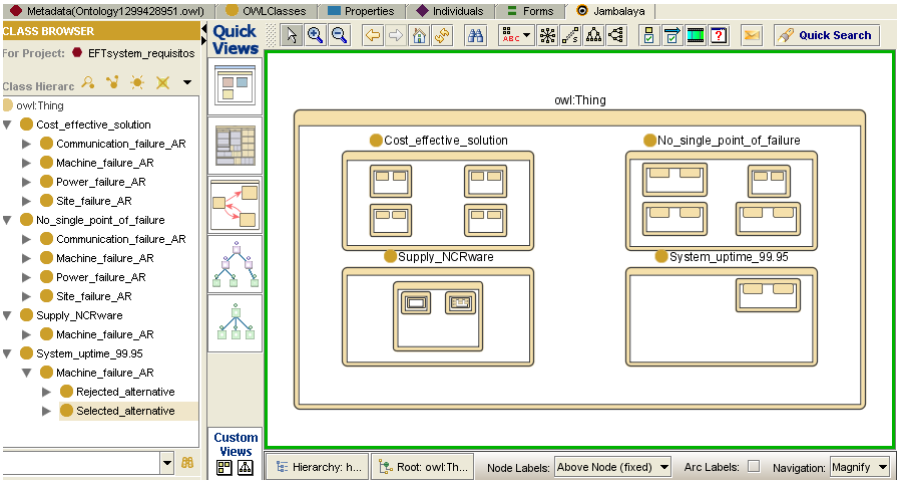

Fig. 4. Pestaña de Jambalaya en Protégé 3.4.4

Options and Criteria) [24]; *SCRAM* (*SCenario Requirements Analysis Method*) [25]; *SEURAT* (*Software Engineering Using RATionale*) [26]; *Sysiphus* [26,27]; *DRIMER* (*Design Recommendation and Intent Model Extended to Reusability*) [28]; *AREL* (*Architecture Rationale and Element Linkage*) [23]; *IBIS* (*Issue-Based Information System*) [29]; *gIBIS* (*graphical IBIS*) [30]; *Compendium*, herramienta de código abierto que está implementada basada en IBIS y soporta notaciones *gIBIS* (véase Fig. 3); *DRL* (*Design Representation Language*) [31]; *ARCHIUM* [23]; *Kruchten's ADD Ontology tool* [26,29]; y *ODV* (*Ontology-Driven Visualization*) [11], que propone el uso de transformaciones de modelo [32] como una representación ejecutable de decisiones de diseño.

Entre todas las herramientas analizadas, *Compendium* fue la elegida para llevar a cabo la evaluación presentada en la sección 2 porque proporciona soporte explícito a la visualización de la rationale de las DDAs, y tiene la ventaja de ser simple y fácil de utilizar.

3.4 Zoomable

Esta técnica de visualización es especialmente interesante por la representación jerárquica que utiliza. Los nodos de los niveles más bajos de la jerarquía están anidados dentro de sus padres, con un tamaño menor que éstos. De esta forma, tendremos que hacer zoom en los nodos hijos para expandirlos y hacer que sean el nivel de visualización actual [12]. En este caso no existen herramientas específicas de CA en esta categoría por lo que se tuvieron en cuenta la siguientes herramientas de visualización de ontologías: *Grokker* [33], un mapa gráfico de conocimiento, donde la información se representa gráficamente; *Jambalaya* [34], un plug-in de *SHriMP* (*Simple Hierarchical Multi-Perspective*) [35] integrado en la herramienta *Protégé* (véase Fig. 4); y *CropCircles* [36], que presenta las jerarquías de clases en ontologías como árboles.

En este caso, se ha seleccionado la herramienta *Jambalaya*, dado que tiene mayor

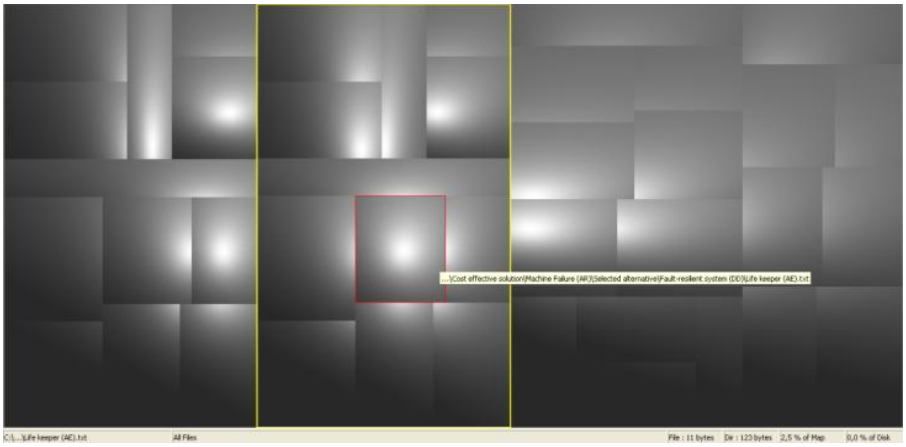

Fig. 5. SequoiaView 1.3

soporte que el resto de herramientas y además está integrada en *Protégé*, herramienta seleccionada previamente.

3.5 Space-filling

Esta categoría presenta los nodos de forma jerárquica, ajustándolos a la pantalla. No es considerada una técnica muy interesante por su falta de claridad y su complicada conexión con la AS. Nótese que no hay herramientas de visualización de CA específicas para esta categoría, por lo que se citan algunas de tipo ontológico: *TreeMaps* [37]; *SequoiaView* [38] (véase Fig. 5); *Information Slices* [39].

En este caso, *Information Slices* se descarta porque no tiene suficiente soporte. Con respecto a las otras dos herramientas, se selecciona *SequoiaView*, dado que tiene más soporte software que *TreeMaps*, es de libre distribución y proporciona una búsqueda de ficheros y un mecanismo de filtrado más eficientes.

4 Material Experimental

Nuestra evaluación empírica utiliza la AS y el CA de un sistema de control financiero denominado *EFT* (*Electronic Fund Transfer*), presentado en [40,41]. La sucursal PBC-GZ es una rama del banco central responsable del control financiero y de pagos y liquidaciones interbancarias del centro financiero de Guangzhou y sus alrededores. Uno de sus sistemas es el *EFT*, que transfiere y liquida pagos de alto valor entre todos los bancos especializados y comerciales de sus alrededores. Este sistema tiene que servir a más de diez millones de personas en el sur de China, y trabaja como una pasarela que conecta a todos los bancos locales con la red de pagos nacional.

El diseño, desarrollo y prueba del sistema EFT llevó cerca de dos años, empleando treinta diseñadores y desarrolladores. Su diseño fue muy exigente dado que era necesario que fuera un sistema fiable, eficiente y seguro, debido a que es el núcleo principal del sistema financiero de la región. El principal problema que presentaba este sistema era que su diseño era difícil de entender para alguien externo al equipo de

desarrollo original, a pesar del hecho de que su diseño fue ampliamente especificado. Por esta razón, se decidió capturar el CA (DDAs y RDAs) para que cualquier persona pudiera interpretar el diseño del sistema EFT.

El CA del sistema EFT es utilizado según se indica en la sección 2 por un conjunto de arquitectos seleccionados para llevar a cabo la evaluación de las diferentes alternativas para la visualización. Por tanto, con ayuda de estas personas, podremos analizar fortalezas y debilidades de las técnicas de visualización de CA, seleccionadas en la sección anterior, que ilustran las DDAs y RDAs del sistema EFT.

5 Análisis

El cuestionario USE (*Effective questionnaire*) [9] permite evaluar la efectividad de la usabilidad basada en cuatro factores de usabilidad: *Utilidad*, *Facilidad de uso*, *Facilidad de aprendizaje* y *Satisfacción*. En la Fig. 6, se muestra un resumen sobre el nivel de usabilidad según los sujetos del grupo de estudio, para cada uno de los cuatro factores de usabilidad, y asociados a cada técnica de visualización.

Primeramente, hay que resaltar que las evaluaciones de efectividad son notablemente similares en los cuatro factores de usabilidad, con una ligera desviación en la facilidad de aprendizaje. En segundo lugar, cabe destacar la distancia significativa entre la mejor y la peor técnica efectiva. Las tres técnicas del medio presentan valores cercanos de efectividad, pero con consistencia a través de tres de las cuatro métricas de efectividad de la usabilidad.

Compendium es la herramienta favorita de los usuarios. Por tanto, nuestra intuición fue la correcta: la técnica *Node-link and tree* ha sido la más efectiva en todos los casos. La retroalimentación de los usuarios refleja que esta técnica presenta el CA de una manera simple y clara, por lo que los usuarios pueden navegar y explorar fácilmente la red de decisiones del sistema EFT.

Por otro lado, la técnica *Wiki* recibió las peores puntuaciones en todos los casos. Los usuarios indicaron que era muy difícil entender cómo hacer cosas concretas y básicas, como crear una decisión de diseño. Además, la idea de una herramienta colaborativa presenta serios problemas si no ha sido diseñada para soportar control de versiones.

Con respecto a las restantes técnicas, *Indented list* es la segunda opción más efectiva, según los usuarios, dado que presenta la información de la forma más

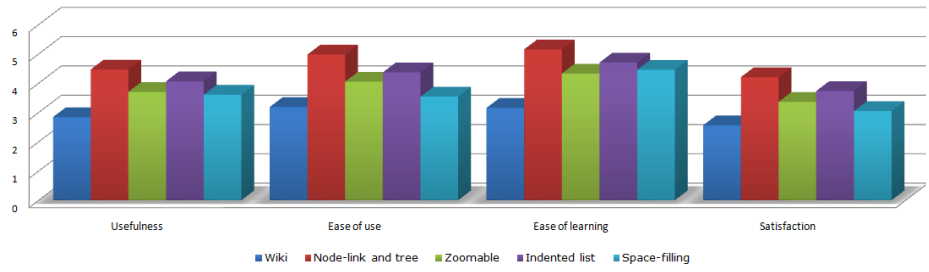

Fig. 6. Evaluación de la usabilidad

sencilla. La técnica *Zoomable* es la tercera opción más efectiva, porque su vista resulta algo más difícil de explorar que la que muestra la categoría anterior. La opción *Space-filling* se sitúa en cuarto lugar, ya que es demasiado simple para proporcionar alguna información relevante acerca del CA.

6 Conclusiones

Como se ha visto a lo largo de este trabajo, las decisiones de diseño y sus rationales han de estar bien documentadas para que el sistema bajo desarrollo/mantenimiento pueda evolucionar fácil y eficientemente. Sin embargo, algunas veces, este CA se presenta de manera inapropiada, lo que dificulta a los arquitectos la tarea de evolución/mantenimiento del sistema.

En este contexto, este artículo describe cinco técnicas de visualización 2D para soportar la visualización del CA y las evalúa mediante una evaluación empírica del factor de calidad *efectividad de la usabilidad*. Este estudio empírico nos ha permitido ver qué técnica de visualización es la más efectiva con respecto a la representación y manipulación del CA, según los cuatro factores de calidad de la efectividad de la usabilidad: utilidad, facilidad de uso, facilidad de aprendizaje y satisfacción. Por tanto, la técnica *Node-link and tree* ha demostrado ser la más efectiva para este propósito, debido a su simplicidad y claridad a la hora de visualizar CA, utilizando un grafo comprensible, fácil de interpretar y navegar, y utilizando nodos simples y entendibles.

Nuestro trabajo futuro estará centrado en las técnicas de visualización 3D para capturar CA e intentaremos determinar qué categoría es la más apropiada para este fin, al igual que se ha hecho en este trabajo con las técnicas bidimensionales. Además, confirmaremos nuestros resultados actuales con estudios adicionales y arquitectos experimentados.

Agradecimientos. Este trabajo ha sido parcialmente financiado por proyectos de la Junta de Comunidades de Castilla-La Mancha (PEII09-0054-9581) y del Ministerio de Ciencia y Tecnología (TIN2008-06596-C02-01 y CONSOLIDER CSD2007-022). El profesor Perry es parcialmente financiado por los proyectos NSF CISE IIS-0438967 y CCF-0820251.

Referencias

[1] P. Kruchten, R. Capilla, and J.C. Dueñas, "The Decision View□ s Role in Software Architecture Practice," *IEEE Software*, vol. 26, 2009, pp. 36-42.

[2] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and A.B. Muhammad, "A comparative study of architecture knowledge management tools," *Journal of Systems and Software*, vol. 83, Mar. 2010, pp. 352-370.

[3] N. Medvidovic and R.N. Taylor, "A Classification and Comparison Framework for Software Architecture Description Languages," *IEEE Trans. Software Eng.*, vol. 26, 2000, pp. 70-93.

[4] D.E. Perry and A.L. Wolf, “Foundations for the Study of Software Architecture,” *ACM Software Engineering Notes*, vol. 17, 1992, pp. 40-52.

[5] J. Bosch, “Software Architecture: The Next Step,” *1st European Workshop in Software Architecture (EWSA’04)*, Heidelberg: Springer, 2004, pp. 194-199.

[6] P. Kruchten, P. Lago, and H. van Vliet, “Building Up and Reasoning about Architectural Knowledge,” *2nd International Conference on Quality of Software Architectures (QoSA 06)*, C. Hofmeister, I. Crnkovic, and R. Reussner, eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 43-58.

[7] R. Capilla, F. Nava, and J.C. Dueñas, “Modeling and Documenting the Evolution of Architectural Design Decisions,” *2nd Workshop on Sharing and Reusing architectural Knowledge Architecture, Rationale, and Design Intent.*, IEEE Computer Society Press, 2007, p. 9.

[8] R.K. Yin, *Case Study Research: Design and Methods, Third Edition, Applied Social Research Methods Series, Vol 5*, Sage Publications, Inc, .

[9] A.M. Lund, “Questionnaire for User Interface Satisfaction,” *http://oldwww.acm.org/perlman/question.cgi?form=USE*, 2001.

[10] T.R. Gruber, “Technical Report KSL 92-71 Revised April 1993 A Translation Approach to Portable Ontology Specifications by A Translation Approach to Portable Ontology Specifications,” *Knowledge Acquisition*, 1993.

[11] R.C. de Boer, P. Lago, A. Telea, and H. van Vliet, *Ontology-driven visualization of architectural design decisions*, IEEE, 2009.

[12] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou, “Ontology visualization methods: a survey,” *ACM Computing Surveys*, vol. 39, Nov. 2007, p. 10-es.

[13] R. Mccall, “PHI: a conceptual foundation for design hypermedia,” *Design Studies*, vol. 12, Jan. 1991, pp. 30-41.

[14] W.C. Regli, X. Hu, M. Atwood, and W. Sun, “A Survey of Design Rationale Systems: Approaches, Representation, Capture and Retrieval,” *Engineering With Computers*, vol. 16, Dec. 2000, pp. 209-235.

[15] G. Fischer, R. Mccall, A. Morch, and M. Drillings, “JANUS : Integrating Hypertext With a Knowledge-Based Design Environment,” *Source*, 1990.

[16] N.F. Noy, R.W. Fergerson, M.A. Musen, and S.M. Informatics, “The knowledge model of Protégé-2000 : combining interoperability and flexibility,” 2000, pp. 1-20.

[17] FZI, “KAON,” *http://kaon.semanticweb.org/*, 2011.

[18] P. Eklund, N. Roberts, and S. Green, “OntoRama: Browsing RDF ontologies using a hyperbolic-style browser,” *First International Symposium on Cyber Worlds, 2002. Proceedings.*, IEEE Comput. Soc, 2002, pp. 405-411.

[19] Y. Sure, J. Angele, and S. Staab, “OntoEdit : Guiding Ontology Development by Methodology and Inferencing,” *Management*, 2002, pp. 1205-1222.

[20] A. Jansen, P. Avgeriou, and J.S. van der Ven, “Enriching software architecture documentation,” *Journal of Systems and Software*, vol. 82, Aug. 2009, pp. 1232-1248.

[21] R. Farenhorst, P. Lago, and H.V. Vliet, “Effective Tool Support for Architectural Knowledge Sharing,” *1st European Conference on Software Architecture*, Madrid: 2007, pp. 123-138.

[22] M. Klein, “An Exception Handling Approach to Enhancing Consistency, Completeness, and Correctness in Collaborative Requirements Capture,” *Journal of Concurrent Engineering Research and Applications*, vol. 5, Mar. 1997, pp. 73-80.

[23] B. Dutoit, A. H.; McCall, R.; Mistrik, I.; Paech, *Rationale Management in Software Engineering*, Springer, 2006.

[24] A. Maclean, R.M. Young, V.M.E. Bellotti, and T.P. Moran, “Questions , Options , and Criteria : Elements of Design Space Analysis,” *Human-Computer Interaction*, vol. 6, 1991, pp. 201-250.

[25] M. Sutcliffe, A. G.; Ryan, "Experience with SCRAM , a SCenario Requirements Analysis Method," *3rd International Conference on Requirements Engineering, CA: IEEE Computer Society Press*, 1998, pp. 164-171.

[26] L. Lee and P. Kruchten, "A Tool to Visualize Architectural Design Decisions," *QoSA 2008*, 2008, pp. 43-54.

[27] B. Bruegge, A.H. Dutoit, T. Wolf, T. Universit, and D.- Garching, "Sysiphus : Enabling informal collaboration in global software development," *1st International Conference on Global Software Engineering*, Costao do Santinho, Florianópolis, Brazil: 2006.

[28] S. Peña-Mora, F.; Vadhavkar, "Augmenting Design Patterns with Design Rationale," *Artif. Intell. For Eng. Design, Analysis and Manuf.*, vol. 11, 1997, pp. 93-108.

[29] M. Shahin, P. Liang, and M.R. Khayyambashi, "Rationale visualization of software architectural design decision using compendium," *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10*, New York, New York, USA: ACM Press, 2010, p. 2.

[30] J. Conklin and M.L. Begeman, "glBIS : A Hypertext Tool for Exploratory Policy Discussion," *ACM Transactions on Office Information Systems*, vol. 6, 1988, pp. 303-331.

[31] J. Lee, "Sibyl: A Qualitative Decision Management System," *Cambridge, Mass. : Center for Coordination Science, Massachusetts Institute of Technology, Sloan School of Management*, 1990, pp. 106-133.

[32] M. Biehl and M. Torngren, "An Executable Design Decision Representation Using Model Transformations," *36th EUROMICRO Conference on Software Engineering and Advanced Applications*, IEEE, 2010, pp. 131-134.

[33] W. Rivadeneira and B.B. Bederson, *A Study of Search Result Clustering Interfaces : Comparing Textual and Zoomable User Interfaces*, 2003.

[34] M.-anne Storey, C. Best, N. Ernst, M. Musen, J. Silva, R. Fergerson, and N. Noy, "Jambalaya : Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé," *Workshop on Interactive Tools for Knowledge Capture (K-CAP 2001)*, Victoria, BC, Canada: 2001.

[35] J. Wu, V. Bc, C. Vw, and M.-anne D. Storey, "A Multi-Perspective Software Visualization Environment," *Proceedings of the 2000 Conference of the Centre for Advanced Studies on Collaborative Research*, 2000.

[36] B. Parsia, T. Wang, and J. Golbeck, "Visualizing Web Ontologies with CropCircles," *Proceedings of the 4th International Semantic Web Conference*, 2005, pp. 6-10.

[37] B. Shneiderman, "Tree visualization with Tree-maps : A 2-d space-filling approach," *ACM Trans. Graph.*, vol. 11, 1992, pp. 92-99.

[38] T.C. science department T.U. Eindhoven, "SequoiaView," 2002.

[39] K. Andrews and H. Heidegger, "Information Slices : Visualising and Exploring Large Hierarchies using Cascading , Semi-Circular Discs," *Proceedings of the IEEE Information Visualization Symposium*, Carolina: 1998, pp. 9-12.

[40] A. Tang, "A Rationale-based Model for Architecture Design Reasoning," Swinburne University of Technology, 2007.

[41] A. Tang, Y. Jin, and J. Han, "A rationale-based architecture model for design traceability and reasoning," *Journal of Systems and Software*, vol. 80, Jun. 2007, pp. 918-934.