An Empirical Evaluation of Visualization Techniques for Architectural Knowledge

Cristina Roda
Computing Systems Dept.
University of Castilla-La Mancha
cristina.roda@alu.uclm.es

Elena Navarro
Computing Systems Dept.
University of Castilla-La Mancha
elena.navarro@uclm.es

Carlos E. Cuesta
Dept. LSI2
Rey Juan Carlos University at Madrid
carlos.cuesta@urjc.es

Dewayne E. Perry
Dept. of ECE
The University of Texas at Austin
perry@ece.utexas.edu

Abstract — Recent research points out the necessity for capturing and representing architectural design decisions as a key element of Architectural Knowledge. Despite the variety of tools that allow visualization of this type of knowledge, there still remains a certain lack of maturity. We classify a set of visualization techniques according to their representation form. These techniques are analyzed considering their strengths and weaknesses using an empirical evaluation. The results of this evaluation suggest some ideas for future work on visualization techniques that can improve the representation of Architectural Knowledge.

Architectural Knowledge, Visualization techniques

I. INTRODUCTION

Software development has to deal with many challenges, such as system complexity, non-functional qualities, maintenance operations, distributed production, frequent personnel changes, etc. [*1*]. Furthermore, software companies with high maintenance costs are increasingly demanding flexible, easy-to-maintain designs [*1*].

Software Architecture (SA) is a valuable asset that enables software companies to achieve a variety of goals by representing and communicating the system structure and behavior to all of a system's stakeholders [*2*]. Until recently, the primary focus of SA research has been on Architecture Description Languages (ADLs) focusing on the *description* of SA *elements* and *form* in the Perry-Wolf model of SA [*3*]. Virtually no attention was paid to the third element of their model – namely, *rationale*. In the last 6 or so years, that has changed: the importance of architectural design decisions (ADDs) and their architectural design rationale (ADR) has been recognized and become a significant research focus. ADDs and ADRs are essential aspects in architectural knowledge (AK), the modeling, managing, and sharing of which has also become a significant research focus [*2*].

In this context, it should be highlighted that whenever a design decision is explicitly recorded and documented, new activities arise during the architecting process. This AK information constitutes a new crosscutting view that overlaps that information described by other views [*1*]. Therefore, the introduction and exploitation of appropriate techniques for visualization become a necessity that should allow the different stakeholders to navigate throughout the different views of the system.

Currently, there are many visualization techniques available to represent AK. As Kruchten, Capilla and Dueñas [*1*] have pointed out, a very active research agenda is being carried out that has produced a significant number of approaches for representing and capturing ADDs. For instance, several approaches use *template lists of attributes* to describe and represent ADDs as relevant entities. One of these approaches [*4*] emphasizes how important is to classify different types of dependencies between decisions as valuable, complementary information for capturing useful traces. Another one of these approaches [*5*] advocates using more flexible approaches that employ obligatory and optional attributes to capture architectural knowledge that can be customized to specific organizations. Alternatively, other authors [*4*] have proposed *ontologies* to formalize tacit knowledge and make visible relationships between decisions and other artifacts of the software life cycle. Finally, the field of software product lines [*1*], has produced a lot of work about specification, modeling, and automation of ADDs that is used to describe and select a product line's common and variable elements.

Thus, a variety of different approaches have emerged during the last years, although very little has have been done to analyze their strengths and weaknesses so that analysts have some useful guidance whenever they have to make decisions about the best alternative for their project. The purpose of this paper is to provide insight into the strengths and weaknesses of these approaches in visualizing AK.

This paper is structured as follows. We present related work in section II and discuss our classification of the visualization techniques used in this paper in section III, along with the identification of which tools match with each of these techniques. We then describe the case study in section IV that we used in performing the subsequent empirical evaluation we describe in section V.

Finally, we present our conclusions and our future work in section VI.

II. RELATED WORKS

SHAring and Reuse of software architectural Knowledge (SHARK) has become an emerging issue of discussion and research, within software architecture development area [*6*]. As Henttonen and Matinlassi state [*6*], when software is developed in a multinational company or in an open source community, the stakeholders are often geographically distributed. Furthermore, Grinter et al. have identified SA as one of the primary mechanisms for organizing distributed software development [*7*]. This is why SHARK has become increasingly important: there is a significant need for appropriate tool support that can store, reuse and share architectural knowledge.

One of the key issues that arises whenever SHARK is introduced in a software development process is the selection of the best supporting tool. This is the question that Henttonen and Matinlassi intend to answer in [*6*]. They present an evaluation framework for SHARK tools that is used to evaluate three open source based solutions: *WebOfPatterns*[*8*][*9*], *Stylebase for Eclipse*[*10*][*11*] and *PAKME*[*12*]. These authors selected those tools because they are not limited to a particular programming language or platform. Henttonen and Matinlassi's evaluation framework for SHARK tools, proposes four points of view to perform this evaluation: *Problem the tool assists in*, *Problem solver*, *Means of problem solving* and *Maturity of the tool* (Figure 1). Each one of these points of view category is associated with a set of criteria, and each criterion is related to some evaluation questions. The evaluation of these tools revealed the strengths and weaknesses of each tool. The authors also emphasized that the target environment must be understood before selecting a tool to be deployed in that organization.

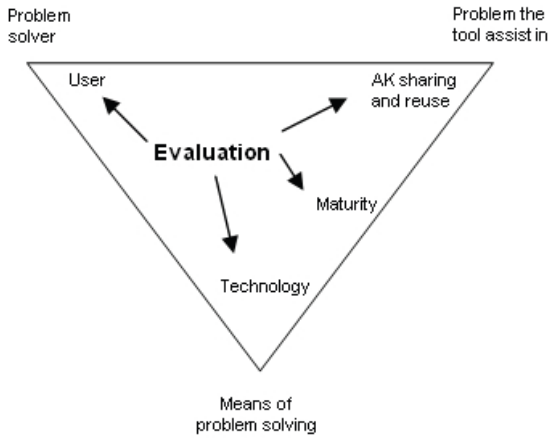

Figure 1. Elements of SHARK tool evaluation [*6*]

Note that this evaluation framework does not include any consideration for evaluating the representation of information, i.e., it does not provide any analysis of the visualization techniques that these tools provide.

Other authors, such as Tang et al. in[*2*], have also proposed a framework for comparing architecture knowledge management tools. This framework comprises several criteria, which represent the context for the comparison, having each one of them associated a research question. Each question consists of a description as well as AK activities of the architectural life-cycle that describe the usage contexts for the criterion. For instance, *Types and Representation of AK* is a criterion to determine "What are the architectural knowledge types and representations captured by a tool for general, context, reasoning and design knowledge?"[*2*]. In this way, Tang et al. propose a framework whose aim is to be a solid guide to comparing AK tools, including the majority of the features required by architecture life-cycle activities. The list of criteria proposed is as follows: *Types and representation of architecture knowledge*, *Relations between AK elements*, *Architectural analysis support*, *Architectural synthesis support*, *Architectural evaluation support*, *Architectural implementation support*, *Architectural maintenance support*, *AK customization*, *Integration with other tools* and *Collaborative environments*. As in [*6*], [*46*] does not include anything for evaluating AK visualization.

Our goal in this paper is to provide a useful framework for analyzing tools that support architectural knowledge visualization to determine their strengths and weaknesses with respect to their AK visualization techniques.

III. VISUALIZATION TECHNIQUES FOR KNOWLEDGE SYSTEMS

Several visualization techniques can be used while capturing, representing or maintaining AK. We want to emphasize that one of the assumptions of our research is that the architectural knowledge is, per se, a knowledgebase made up by ADDs and ADRs and their corresponding relationships that can be used to understand and reason about the software architecture of a system. In this sense, it resembles an ontology [*13*], as other authors have already noticed [*14*]. This has led us to use as taxonomy of visualization techniques that proposed by Katifori et al.[*15*] which distinguishes five different representation types, depending on the information presentation, interaction method, or functionality supported. In the following subsections, each one of these types is described along with the available tools that can be classified according to such types.

This paper focuses on two-dimensional tools, primarily because they are closer to those commonly used by architects. Subsections A, B and C present AK visualization tools, while subsections D and E present ontological ones. We have included the latter subsections because they present the information in a hierarchical form, very similar to AK representations in AK tools. Note that if AK tools are available for a particular visualization technique, they are preferred; if AK tools are not available, ontological ones are preferred

A. *Indented list*

According to this representation type, the AK is represented by means of plain text that looks like tree view, similar to Windows Explorer. The simplicity of this textual representation makes this method not very popular today to represent architectural knowledge. An AK system that uses this visualization technique is *PHI* (*Procedural Hierarchy of Issues*)[*16*], which extends the IBIS system and presents an argumentation approach to resolving issues, i.e. any design question, deliberated or not. Some tools that support PHI methodology are [*17*]: JANUS [*18*][*17*], HOS (*Hyper-Object Substrate*) [*17*], and PHIDIAS (*Procedural Hierarchy of Issues*/*Design Intelligence Augmentation*) [*17*]. However, these tools currently do not have any support, hence some ontological tools that offer a Windows Explorer tree view are offered for consideration.

Protégé [*19*] is an ontology-editing and knowledge-acquisition environment, where classes are represented as nodes in an indented, retractable and expandable tree; and instances are displayed in another window (see Figure 2). *KAON* [*20*] is an open-source ontology management environment for business applications, which includes a complete tool suite for easy ontology management and creation; and provides a framework for building ontological applications. *OntoRama* [*21*] is a Java client which allows users to browse a knowledge base (ontology) structure in a hyperbolic layout. Finally, *OntoEdit* [*22*] is an ontology editor which supports methodology-based ontology construction.

Protégé is the tool selected to be evaluated in section V because it is open-source available, and it was cited by other authors in [*23*][*15*] using it in AK context.

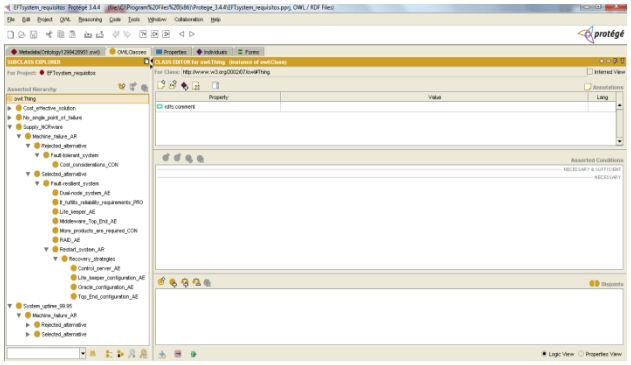

Figure 2. Protégé 3.4.4

B. *Wiki*

As Farenhorst, Lago and van Vliet state in [*24*], a wiki for capturing architectural knowledge can allow designers and architects to collaborate and communicate easily. Therefore, thanks to the capabilities of the wiki, the information can be quickly updated, and stakeholders can always know the current state of the project. Some wiki tools for architectural knowledge visualization are: *C-ReCS* (*Collaborative Requirements Capture System*) [*25*], that supports collaboration while capturing ADDs; *PAKME* (*Process-based Architecture Knowledge Management Environment*) [*26*], which is a web-based tool that supports collaboration for managing architectural knowledge; *ADDSS* (*Architecture Design Decision Support System*) [*26*] [*27*] (see Figure 3), a web-based tool, as PAKME, that manages and documents ADDs; and finally, *Knowledge Architect* [*26*] [*27*], which provides mechanisms for capturing, managing and sharing architectural knowledge, thanks to an architectural knowledge server and repository.

In this case, ADDSS is the system selected to perform the empirical evaluation presented in section V. It is preferred over other Wiki tools because is the most complete one and provides a query system that allows architects easily to find information about requirements, decisions and architectures stored in the tool.

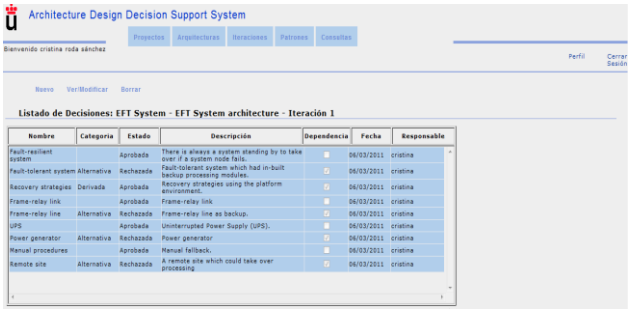

Figure 3. ADDSS 2.0

C. *Node-link and tree*

A node-link and tree approach provides an interconnected node representation, with a top-down or left-right layout. This technique allows users to expand and retract nodes and their sub-nodes, so that the information detail level can be regulated. In the following, tools corresponding to each category are presented.

QOC (*Questions, Options, and Criteria*) [*28*] is based on a semi-formal notation to analyze the design space, using three main elements: Questions (key design subjects), Options (possible answers to the questions) and Criteria (valuation on the options). *SCRAM* (*SCenario Requirements Analysis Method*) [*29*] is a requirement analysis method based-on scenarios. It has four techniques for requirements capture and validation: use of prototypes or concept demonstrators; scenarios; design rationale; and whiteboard summary. Another tool is *SEURAT* (*Software Engineering Using RATionale*) [*30*], an Eclipse development environment plug-in utility that captures and uses design rationale by linking its software code. *Sysiphus* [*30*] [*31*] is also a rationale-based set of tools that allows us to capture several system models for system development activities, and that supports rationale-based design decisions and links them with system models, using graphs. *DRIMER* (*Design Recommendation and Intent Model Extended to Reusability*) [*32*] provides explicit capture of ADR during

the software development process. *AREL* (*Architecture Rationale and Element Linkage*) [*27*] is based on UML to assist architects in creating and documenting architectural designs, focusing ADDs and ADRs. This tool captures three types of architectural knowledge: design concerns; design decisions; and design outcomes. These knowledge entities are represented as standard UML entities and linked for show their relations. *IBIS* (*Issue-Based Information System*) [*33*] is an argument-based approach for design rationale representation that consists of three simple and basic concepts: *Issues* that need to be addressed; *Positions* that answer such issues; and *Arguments* which are composed by *Pros* (arguments in favor) and *Cons* (arguments against) of a concrete *position*. *gIBIS* (*graphical IBIS*) [*34*], IBIS successor, is another AK visualization tool that uses color and a fast relational database server to facilitate construction and exploration of IBIS networks. *Compendium* [*35*][*33*] is an open source tool that is implemented based-on IBIS and supports gIBIS notations (see Figure 4).

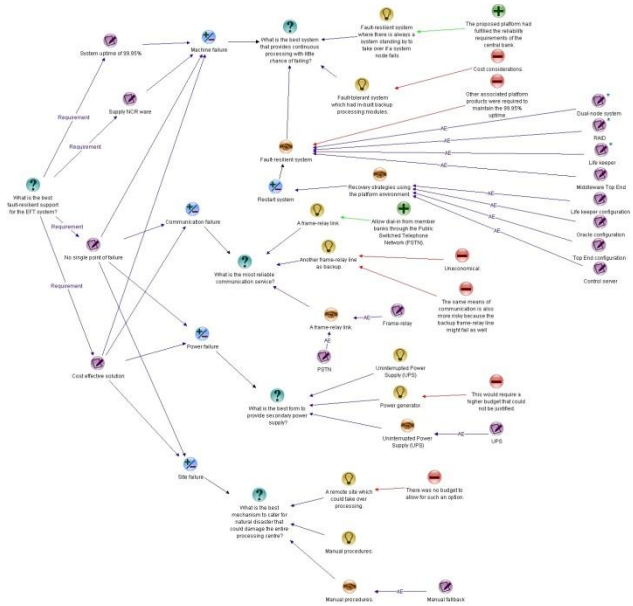

Figure 4. Compendium

DRL (*Design Representation Language*) [*36*] allows constructing decision graphs, which reflect the pros and cons in evaluating alternatives with respect to the objectives. *ARCHIUM* [*27*] is a Java extension that provides traceability through a wide range of concepts (such as requirements, decisions, architecture descriptions, and implement artifacts) which are maintained during the system life cycle. AK is presented as a component view, where a dependency graph shows ADDs and their relationships. *Kruchten's ADD Ontology tool* [*30*], as its name indicates, is based on the ontology for Kruchten's architectural design decisions. This tool facilitates both exploration and detailed analysis of decisions, thanks to four views [*30*][*33*]: decision and relationship lists; decision structure visualization view; decision chronology view; and decision impact view. *ODV* (*Ontology-Driven Visualization*) [*14*] tool combines the power of decision and relationship lists with decision structure visualization view, introduced by Kruchten's ADD Ontology tool.

The last two-dimensional tool considered is presented in [*37*]. Its authors propose the use of model transformations as an executable representation of design decisions. These transformations can be executed, and the final result is a changed architecture. So, design decisions only have to be captured as a transformation, and architecture change is a cause of transformation execution.

Among all the analyzed tools, Compendium tool was selected to perform the empirical evaluation presented in section V. It is preferred over the others because it provides explicit support for rationale visualization of ADDs, and it has the advantage of being simple and easy to use.

D. Zoomable

This visualization technique is especially interesting in the way it addresses hierarchical. It presents the nodes in the lower levels of the hierarchy nested inside their parents, with a size smaller than them. Thus, if we want to know more about nodes, we have to zoom-in to the child nodes, in order to expand them and make them the current viewing level [*15*]. In addition, some ontology visualization systems are briefly described, belonging to this category, even though they are not specifically AK visualization tools.

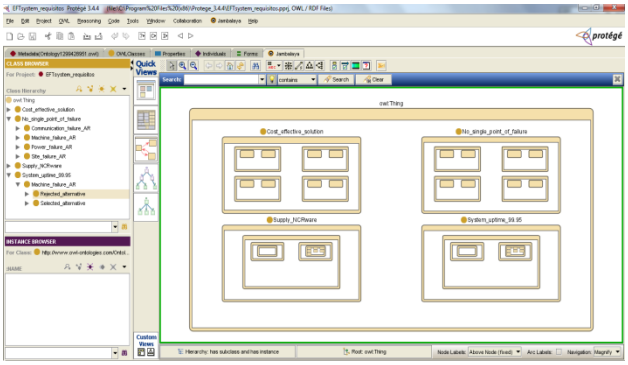

Figure 5. Jambalaya tab in Protégé 3.4.4

Grokker [*38*] is a graphical knowledge map, where information is represented graphically. It uses a clustering mechanism for presenting documents as a series of nested Venn diagrams. Another system is *Jambalaya* [*39*], that is an integration or plug-in of SHriMP (Simple Hierarchical Multi-Perspective) [*40*] with Protégé ontology tool (see Figure 5). SHriMP is a multi-perspective software visualization environment, which combines single view and multi-view techniques to support software exploration. *CropCircles* [*41*][*42*] presents the class hierarchies in ontologies as trees, so that circles represent

nodes and every child circle is nested inside its parent circle.

In this case, Jambalaya is the selected tool because it has more support than the others and it works within Protégé, which was previously selected as well.

E. Space-filling

This category presents nodes in a hierarchical form, using all the screen space, i.e. adjusts nodes to such screens. It is not considered a very interesting technique for two reasons:, its lack of clarity, and its complicated connection to Software Architecture. Note that there are not any specific AK visualization tools for this category, so some ontological ones are presented.

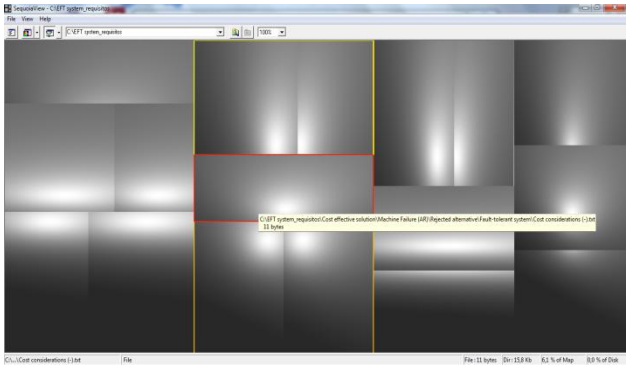

Figure 6. SequoiaView 1.3

TreeMaps [*43*] is a tool to representing hierarchies or trees that have weights or sizes on the leaf nodes, which are rectangles whose area is proportional to some attribute, such as node size. *SequoiaView* [*44*] is another 2D space-filling tool where the screen is subdivided such that rectangles approach squares as closely as possible (see Figure 6). The last tool is *Information Slices* [*45*] that represents hierarchical structures using one or more semi-circular discs, which represents multiple levels of the hierarchy.

In this case, Information Slices is discarded because it does not have enough support, i.e. there is no software availability. With respect to the other two tools, SequoiaView is selected because it has more support than TreeMaps, it is freeware, and it provides a more efficient file search and filtering mechanism.

IV. CASE STUDY

Our empirical evaluation uses architecture and AK from a case study presented in [*46*] [*47*] related to the financial control system. The People's Bank of China Guangzhou branch (PBC-GZ) is a central bank branch which is responsible for the financial control and inter-bank payments and liquidations of the financial centre Guangzhou and its surroundings. One of its systems is the *Electronic Fund Transfer* (EFT) that transfers and liquidates high value payments between all the specialized and commercial banks in the surroundings. This system has to serve over ten million people in southern China, and works as a gateway to connect all local banks to the national payment network.

The design, development and test of the EFT system took about two years, employing thirty designers and developers. Its design was highly demanding as it was necessary that this system had to be trustworthy, efficient and secure, because it is the main core of the financial system in the region. The main problem this system presented was that its design was difficult to understand for anyone outside the original development team, despite the fact that its design was widely specified. For this reason, it was decided to capture the architectural knowledge, i.e. the ADDs and ADRs, so that anyone could interpret the EFT system design.

The architectural knowledge of the EFT is used in the following section by a set of selected architects to carry out the evaluation of the different alternatives for visualization. Therefore, in section V, a set of people, previously selected, help us to analyze the strength and weaknesses of the architectural knowledge visualization tools, selected in the previous section, that illustrate the ADDs and ADRs of the EFT system.

V. EMPIRICAL EVALUATION

We present multiple exploratory case studies [*48*] in which each software architect must make changes to the EFT architecture and in which he or she uses the various different tools to aid in making those changes. After the set of changes have been successfully completed, the architect evaluates the usefulness of each of the visualization tools.

A. Study Research Goal

The goal of this study is to evaluate which visualization technique is the most effective in using architectural knowledge in the process of making a set of architectural changes (that is, in making and remaking architectural decisions) to an existing SA using the various visualization tools to represent the AK of the existing SA.

Our null hypothesis if we were doing a controlled experiment would be that there is no difference in the effectiveness of the visualization tools – i.e., they would all be equally effective. Our intuition, however, is that *Node-link and tree* technique is the most effective in representing architectural knowledge, given that its graphical view provides more information than the other techniques. While our exploratory case studies will individually support that intuition, we will nevertheless outline a global data analysis whose results clearly support this conclusion.

B. Study Constructs

The independent variable in these case studies (i.e., the input) is the AK of the EFT architecture as

represented by the various visualization tools. The tools have been extensively discussed in section III.

The dependent variable (i.e., the output as determined by the various treatments – the architecture changes) is the *effectiveness* of the various visualization tools in response to the various treatments (described in the next subsection). Effectiveness is determined by 1) usefulness of the tool, 2) ease of tool use, 3) ease of learning how to use the tool, and 4) satisfaction with tool. The questionnaire presented in [*49*] has been used to determine the effectiveness of the various visualization tools in representing the EFT's AK. For each one of the visualization techniques, a questionnaire was created using the e-learning platform Moodle that is available in the University of Castilla-La Mancha for teaching assistance. This platform allowed us to collect all the results of the questionnaires for their later analysis.

C. Study Design

Each case study consisted of 1) performing several tasks (each task is considered to be a treatment applied to the EFT architecture) that are usually done by architects while evolving a system, and 2) evaluating the effectiveness of the various techniques using the questionnaire in [*49*]. These tasks were related to the modification of the architectural knowledge or the software architecture of the EFT project described in section IV .

The study group was made up by 15 students (three women and twelve men) in the last course of the Degree in Computing at the University of Castilla-La Mancha whose ages were between 22 and 25. Each student carried out the case study design performing the tasks and effectiveness evaluation.

The exploratory case studies were carried out in the context of practical session of the Advanced User Interface Design class and so they were used to the terms as (usability) effectiveness and visualization technique. Each case study session lasted about two hours.

The information collected during the experiment was analyzed using standard statistical measures and techniques, as described in section F.

D. Study Treatments

The following is the background for the EFT software architecture and the basic issues in creating that SA and its associated AK:

- The selection of the system and software architecture platform is one of the most fundamental architecture issues. The architect must take into account that the EFT system must provide *fault-resilient support*. So, he/she has to wonder about what is the best system that provides continuous processing with little chance of failing. There are two possible choices for a reliable machine: a *fault-resilient system* which has always a system node standing by to take over if a system node fails; or a *fault-tolerant system* which has in-built backup processing modules. For the EFT system, the architects selected a fault-resilient system because it satisfied the reliability requirements of the central bank, whereas a fault-tolerant system had a high cost that made it an unattractive candidate. However, the fault-resilient system entails a disadvantage: other associated platform products are required to maintain the 99.95% uptime. So, the architect had to make another decision: what recovery strategies were necessary to deploy by using the platform environment.
- The architects also had to pay attention to *network reliability*, because banking operations must be carried out in a secure environment. For this issue two options were considered: to introduce a *frame-relay link* in the system; or to introduce another *frame-relay line* as *backup*. The architects selected the first option because it allowed dial-in from member banks through the Public Switched Telephone Network (PSTN), while the second option is uneconomical and more risky because the backup frame-relay line might fail as well.
- Another important aspect was *power failure*, given that the EFT system had to provide a continuous service. The architect has to answer the question about what is the best form to provide secondary power supply. Two alternatives were evaluated: an uninterrupted power supply (UPS); or a power generator. Given that the second alternative would require a higher budget that could not be justified the architects selected the UPS option.
- To handle natural disasters such as earthquakes or fires which could damage the entire processing centre, the architects had to select an adequate mechanism: a *remote site* which could take over processing; or *manual procedures*. Finally, the architects selected manual procedures, because there was not enough budget to allow for such first option.

Given that system design, two structures related to the EFT system architectural knowledge were created in each tool: one from the point of view of requirements, and the other one from the point of view of architectural elements. Given the system AK represented in the various visualization techniques, the following tasks (treatments) are to be applied in each case study:

- Task1 - The subjects were informed that the PBC-GZ had needed to experience no monetary problems so that EFT system had to be evolved to deploy the best alternatives, no matter their costs were. For this reason, the first change to be performed was to change the *Cost Effective Solution* requirement for *The Best Solution*.
- Task 2 - A new requirement was added: *24h Monitoring* that allows the system to be aware of any warning due to problems of power supply, machine failure, communication failure, and site failure. The

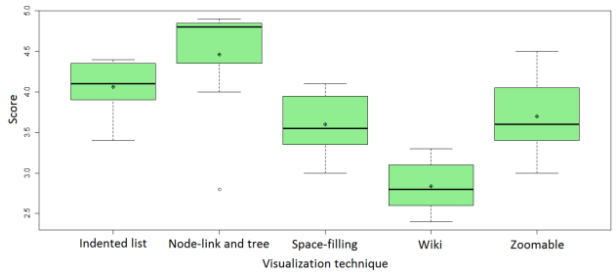

Figure 7. Boxplots for Usefulness

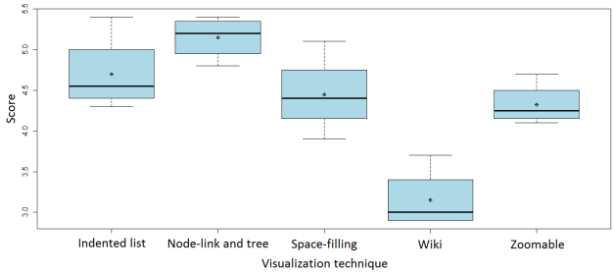

Figure 8. Boxplots for Ease of learning

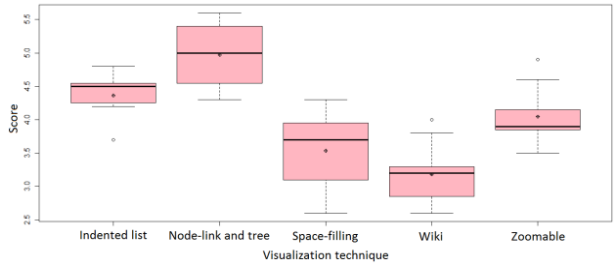

Figure 9. Boxplots for Ease of use

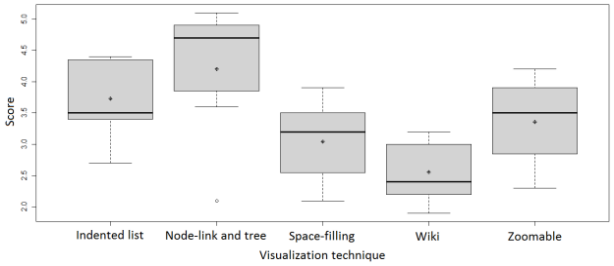

Figure 10. Boxplots for Satisfaction

introduction of this new requirement affected all the initial architecture rationales defined for the system.

– Task 3 – The last change is that *ORACLE database* is to be replaced by MySQL because PBC-GZ is going for open source software.

As can be observed, these three tasks led the study group to navigate through the structures of architectural knowledge, and modify what they considered more appropriate, taking into account that the first two changes affected the requirement-centered structure, and the last one impacted architectural element-centered structure.

E. Implementation

A virtual machine was prepared that allowed each subject to carry out his or her case study and that has a Windows XP operating system, and all the selected visualization tools. Then, this virtual machine was copied in 15 PCs Dell™ Inspiron One 19. They are all-in-one computers with built-in 19" touch screen and a CPU.

The multiple exploratory case studies were run in a laboratory using the above described equipment with the study group and it was carried out in three different phases:

– First, an introduction to the experiment was provided, describing its main aim, the case study to be used, and the tasks to be performed.
– Second, a brief introduction of each one of the selected tools was performed so that the subjects acquired the necessary abilities for their manipulation.
– Third, the study group carried out each one of the described tasks (i.e., treatments) using the various techniques and, as soon as they finished them they were required to fill in the Effectiveness questionnaire.

It is worth noting that at each phase, the subjects were advised that the main aim of the experiment was to evaluate the visualization technique and not the tool.

F. Analysis

The Effectiveness Questionnaire [*49*] chosen for our study evaluates usability effectiveness based on four usability factors: *Usefulness*, *Ease of use*, *Ease of learning* and *Satisfaction*. As already noted, our empirical evaluation was designed as a set of exploratory case studies; therefore, our actual conclusions have to be extracted as the results for each individual case study, and each one of them presents results for these factors. As also indicated, our intuition is that one visualization technique (*Node-link and tree*) is much better than the others. Our multiple case studies supports that intuition, both individually and specifically for each one of the four usability factors. The validity of this exploratory approach is discussed in more detail in the next subsection.

To provide an initial global impression about the results, we gather these individual case studies, providing a pair of graphic depictions in the *average* case. In this context, boxplots[1] are used (Figure 7, Figure 9, Figure 8, Figure 10) to show the Score-Visualization Technique relation, related to each usability factor. As we can see, the preference for *Node-link and tree* is clearly obvious.

[1] A boxplot graphically depicts numerical data through five-number summaries (bottom-up): the smallest observation, lower quartile, median, upper quartile, and largest observation. In addition, the average has been also illustrated as a point in the middle of the boxplot.

However, these average results can be misleading: as indicated, our experience is structured as multiple case studies, and therefore it is the individual preference, rather than the average results, what actually matters.

Our empirical survey was not designed as a controlled experiment, and this means that the validity and statistical significance of any "average" results has to be carefully considered. Among many other details, our sample size is comparatively small, and individuals were not randomly selected; even worse, observations cannot be considered *independent*, as the same population was used to evaluate all five techniques: i.e., the same people gave an score for every technique, and this excludes the possibility of doing a standard analysis of variance.

Of course, this does not mean that we have to discard an statistical analysis completely; more the contrary, our conclusions about average values are stronger if they are supported by a kind of analysis which takes into account these specific issues.

Hence, we outline an statistical hypothesis test which differs from the more usual approach in several aspects. First, we are not considering a single technique, but five; hence, the hypothesis test is made *in pairs*: scores in each technique are compared to those in the remaining four. Also, our observations were not independent; so instead of considering individual scores we have to take their *differences* – again, using the same pairs. Thus we have to do 20 tests (falling down to 10); and these "pair tests" have to be performed for the four usability factors.

Finally, our test cannot use the standard distribution as usual, due to the sample size but we may use Student's t-distribution instead. In summary, our hypothesis test is finally structured as a set of 40 t-tests, made in pairs.

Now our null hypothesis can be (again) that "all these techniques are equally effective". Applied to each one of these tests, this means that every time that the hypothesis is rejected, the difference between the two techniques in the pair can be considered statistically *significant*. Once we have established that, we are able to compare them using the average, as we originally intended.

The results of our Student's t-tests are included in four pairing matrices (Table I-IV); every time that the p-value is less than 0.01 the null hypothesis is rejected, and the techniques in the pair must considered non-equivalent. A quick review shows that most of the times the null hypothesis is indeed rejected, and only a few conflicts remain (these are marked by bold typeface).

Considering each one of the factors, we can see that *ease of use* provides the clearer results. Namely, all scores are considered significantly different, hence we can use the average results to state that *Node-link and tree* is perceived as the easiest-to-use technique. It is followed by *Indented list*, *Zoomable*, *Space-filling* and *Wiki*, in this order.

Satisfaction provides a similar result: all differences are significant, and the order of preference is the same. On *usefulness*, the analysis is not able to reject the null hypothesis in the pair (*Zoomable*, *Space-filling*); but still the rest of the differences make possible to choose *Node-link and tree* as the most useful technique.

Finally, *ease of learning* is the more conflicting result in our analysis, where the differences in three pairs cannot be considered significant – but this is probably due to the fact that this factor has the smallest sample size (not all of the individuals completed this part of the survey). This does not mean that *Node-link and tree* is not probably the easiest-to-learn choice (the remaining pairs still suggest

P-value	Wiki	Node-link & Tree	Indented List	Zoomable	Space-filling
Wiki		0,000001	0,000000	0,000001	0,000001
Node-link & Tree	0,000001		0,001067	0,000280	0,000214
Indented List	0,000000	0,001067		0,000737	0,000134
Zoomable	0,000001	0,000280	0,000737		**0,019256**
Space-filling	0,000001	0,000214	0,000134	**0,019256**	

TABLE I. HYPOTHESIS TEST IN PAIRS FOR USEFULNESS

P-value	Wiki	Node-link & Tree	Indented List	Zoomable	Space-filling
Wiki		0,000000	0,000000	0,000000	0,000043
Node-link & Tree	0,000000		0,000000	0,000000	0,000000
Indented List	0,000000	0,000000		0,000074	0,000000
Zoomable	0,000000	0,000000	0,000074		0,000013
Space-filling	0,000043	0,000000	0,000000	0,000013	

TABLE II. HYPOTHESIS TEST IN PAIRS FOR EASE OF USE

P-value	Wiki	Node-link & Tree	Indented List	Zoomable	Space-filling
Wiki		0,000219	0,000096	0,000211	0,000314
Node-link & Tree	0,000219		**0,020503**	0,003174	0,004385
Indented List	0,000096	**0,020503**		**0,030395**	0,008950
Zoomable	0,000211	0,003174	**0,030395**		**0,516762**
Space-filling	0,000314	0,004385	0,008950	**0,516762**	

TABLE III. HYPOTHESIS TEST IN PAIRS FOR EASE OF LEARNING

P-value	Wiki	Node-link & Tree	Indented List	Zoomable	Space-filling
Wiki		0,000006	0,000000	0,000037	0,000179
Node-link & Tree	0,000006		0,001872	0,000068	0,000021
Indented list	0,000000	0,001872		0,003598	0,000025
Zoomable	0,000037	0,000068	0,003598		0,002781
Space-filling	0,000179	0,000021	0,000025	0,002781	

TABLE IV. HYPOTHESIS TEST IN PAIRS FOR SATISFACTION

that) but the analysis cannot be conclusive.

As expected, *Node-link and tree* is considered better than the other approaches in the survey. Our intuition was correct: this technique has been the most effective for all cases. User feedback reflects that it presents architectural knowledge in a simple and clear way, so users can easily navigate and explore the EFT system decision network.

G. Study Validity Issues

Construct validity is strong. Usability effectiveness and its constituent measures are well understood by the various subjects. Further, the use of a well described architecture together with its AK represented in well understood visualization techniques has strong construct validity as well.

Internal validity suffers somewhat from the use of students for each case study rather than practicing software architect. Further studies would be needed with practicing architects to see if their effectiveness evaluations would match those of the students. The underlying reasons for the current effectiveness evaluations suggest that these studies would have results congruent with our studies.

The strength of external validity lies in the use of a realistic software architecture and its architecture knowledge and in performing multiple studies. Its weakness is analogous to that of internal validity in that the multiple case studies are performed using students but with our expectation of congruent results in further studies with practicing architects we consider external validity overall to be very good.

VI. CONCLUSIONS AND FURTHER WORK

As seen throughout this work, design decisions and their rationales have to be well documented so that that the system under development/maintenance can efficiently and easily evolve. However, sometimes this architectural knowledge is presented in an inappropriate way that does not facilitate the architects' task of system evolution.

In this context, this paper describes five 2D visualization techniques to support architectural knowledge visualization and assesses them by means of an empirical evaluation of the quality factor *usability effectiveness*. This empirical study has allowed us to observe which visualization technique is the most effective one for representing and manipulating architectural knowledge, in terms of four quality usability effectiveness sub-factors: usefulness, ease of use, ease of learning and satisfaction. Thus, *Node-link and tree* technique has proved to be the most effective one for this purpose, because its simplicity and clarity in visualizing architectural knowledge as a comprehensible graph that is easy to interpret and navigate, using simple and understandable nodes.

Our future work will be focused on 3D visualization techniques for capturing architectural knowledge and will try to determine which category is the most appropriate for this aim, as this work has been done with two-dimensional ones. In addition, we will confirm our current results with further studies with practicing architects.

ACKNOWLEDGMENTS

We would like to thank Professor Antonio Alonso-Ayuso, from the URJC DEIO, for his help with the statistical analysis; and Professor Víctor López Jaquero, for allowing us to use his Laboratory. This work has been partially supported by grants (PEII09-0054-9581) from the Junta de Comunidades de Castilla-La Mancha and also by a grant (TIN2008-06596-C02-01) from the Spanish Government.

REFERENCES

[1] P. Kruchten, R. Capilla, and J. C. Dueñas, "The decision view's role in software architecture practice," *The IEEE Computer Society*, vol. 26, no. 2, pp. 36-42, 2009.

[2] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and M. A. Babar, "A comparative study of architecture knowledge management tools," *Journal of Systems and Software*, 2009.

[3] D. E. Perry and A. L. Wolf, "Foundations for the Study of Software Architecture," *ACM Software Engineering Notes*, vol. 17, no. 4, pp. 40-52, 1992.

[4] P. Kruchten, P. Lago, and H. van Vliet, "Building Up and Reasoning about Architectural Knowledge," in *2nd Int'l Conf. Quality of Software Architectures (QoSA 06), LNCS 4214*, 2006, pp. 43-58.

[5] R. Capilla, F. Nava, and J. C. Dueñas, "Modeling and Documenting the Evolution of Architectural Design Decisions," in *2nd Workshop Sharing and Reusing Architectural Knowledge Architecture, Rationale, and Design Intent*, 2007, p. 9.

[6] K. Henttonen and M. Matinlassi, "Open Source Based Tools for Sharing and Reuse of Software Architectural Knowledge," in *The Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, Cambridge, UK, 2009.

[7] Rebecca E. Grinter, James D. Herbsleb, and Dewayne E. Perry, "The geography of coordination," in *ACM Press*, 1999, pp. 306-315.

[8] J. Dietrich and C. Elgar, "A formal description of design patterns using OWL," in *Proceedings of The Australian Software Engineering Conference (ASWEC)*, 2005.

[9] J. Dietrich and C. Elgar, "Towards a web of patterns," in *Proceedings of Workshop on Semantic Web Enabled Software Engineering (SWESE)*, 2005.

[10] K. Henttonen, "Stylebase for Eclipse. An Open Source Tool to Support the Modelling of Quality Driven Software Architecture," VTT Technical Research Centre of Finland, Espoo, VTT Research Note 2387, 2007.

[11] K. Henttonen and M. Matinlassi, "Contributing to Eclipse: A Case Study," in *Proceeding of 2007 Conference on Software Engineering (SE2007)*, 2007.

[12] M. A. Babar and I. Gorton, "A Tool for Managing Software Architecture Knowledge," in *Proceedings of the Second Workshop on Sharing and Reusing Architectural Knowledge (SHARK 2007)*, 2007.

[13] Thomas R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220,

1993.

[14] R. de Boer, P. Lago, A. Telea, and H. van Vliet, "Ontology-Driven Visualization of Architectural Design Decisions," *Proceedings of the 8th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pp. 43-52, 2009.

[15] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou, "Ontology Visualization Methods-A Survey," *ACM Computing Surveys*, vol. 39, no. 4, 2007, Article 10, 43 pages.

[16] R. J. McCall, "PHI: a conceptual foundation for design hypermedia," *Design Studies*, vol. 12, no. 1, pp. 30-41, 1991.

[17] 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, pp. 209-235, 2000.

[18] G. Fischer, R. McCall, and A. Morch, "JANUS: Integrating Hypertext with a Knowledge-based Design Environment," in *Proceedings of Hypertext '89*, Pittsburgh, Pennsylvania, 1989, pp. 105-117.

[19] N. F. Noy, R. W. Fergerson, and M. A. Musen, "The knowledge model of Protégé-2000: Combining interoperability and flexibility," in *2nd International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000)*, Juan-les-Pins, France, 2000.

[20] KAON. [Online]. http://kaon.semanticweb.org/

[21] P. W. Eklund, N. Roberts, and S. P. Andgreen, "OntoRama: Browsing and RDF ontology using a hyperbolic-style browser," in *1st International Symposium on CyberWorlds (CW2002)*, 2002, pp. 405-411, Theory and Practices, IEEE press.

[22] Y. Sure, J. Angele, and S. Staab, "OntoEdit: Guiding ontology development by methodology and inferencing," in *International Conference on Ontologies, Databases and Applications of Semantics (ODBASE'02)*, Irvine, 2002.

[23] A. Jansen, P. Avgeriou, and J. S. van der Ven, "Enriching software architecture documentation," *Journal of Systems and Software*, vol. 82, no. 8, pp. 1232-1248, 2009.

[24] R. Farenhorst, P. Lago, and H. van Vliet, "Effective Tool Support for Architectural Knowledge Sharing," in *1st European Conference on Software Architecture*, Madrid, Spain, 2007, pp. 123-138.

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

[26] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and M. A. Babar, "A comparative study of architecture knowledge management tools," *J. Syst. Software*, 2009.

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

[28] A. MacLean, R. M. Young, V. M. Bellotti, and T. P. Moran, "Questions, options, and criteria: elements of design space analysis," *Human-Computer Interaction*, vol. 6, no. 3, pp. 201-250, 1991.

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

[30] L. Lee and P. Kruchten, "A Tool to Visualize Architectural Design Decisions," *QoSA 2008, LNCS 5281*, pp. 43-54, 2008.

[31] B. Bruegge, A. H. Dutoit, and T. Wolf, "Sysiphus: Enabling Informal Collaboration in Global Software Development," in *1st International Conference on Global Software Engineering*, Costao do Santinho, Florianópolis, Brazil, 2006.

[32] F. Peña-Mora and S. Vadhavkar, "Augmenting Design Patterns with Design Rationale," *Artif. Intell. For Eng. Design, Analysis and Manuf.*, vol. 11, no. 2, pp. 93-108, 1997.

[33] 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*, 2010.

[34] J. Conklin and M. L. Begeman, "gIBIS: A Hypertext Tool for Exploratory Policy Discussion," *ACM Transactions on Office Information Systems*, vol. 6, no. 4, pp. 303-331, Octubre 1988.

[35] M. Shahin, P. Liang, and M. R. Khayyambashi, "Improving understandability of architecture design through visualization of architectural design decision," *Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge*, 2010.

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

[37] M. Biehl and M. Törngren, "An executable design decision representation using model transformations," *36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2010)*, Septiembre 2010.

[38] W. Rivadeneira and B. B. Bederson, "A Study of Search Result Clustering Interfaces: Comparing Textual and Zoomable Interfaces," *HCIL-2003-36, Tech. Rep.*, October 2003.

[39] M.-A. Storey et al., "Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé," in *Workshop on Interactive Tools for Knowledge Capture (K-CAP 2001)*, Victoria, BC, Canada, 2001.

[40] J. Wu and M.-A. Storey, "A multi-perspective software visualization environment," in *Proceedings of the 2000 Conference of the Centre for Advanced Studies on Collaborative Research*, 2000.

[41] B. Parsia, T. Wang, and J. Goldbeck, "Visualizing Web ontologies with cropCircles," in *Proceedings of the 4th International Semantic Web Conference*, 2005, pp. 6-10.

[42] T. Wang and B. Parsia, "CropCircles: topology sensitive visualization of owl class hierarchies," in *Proceedings of the International Semantic Web Conference (ISWC 06)*, 2006.

[43] B. Shneiderman, "Tree visualization with tree-maps: A 2-d space-filling approach," *ACM Trans. Graph.*, vol. 11, no. 1, pp. 92-99, September 1992.

[44] The Computer science department Technische Universiteit Eindhoven. (2002) SequoiaView. [Online]. http://www.win.tue.nl/sequoiaview/

[45] K. Andrews and H. Heidegger, "Information Slices: Visualising and exploring large hierarchies using cascading, semi-circular discs," in *Proceedings of the IEEE Information Visualization Symposium*, Carolina, 1998, pp. 9-12.

[46] A. Tang, Y. Jin, and J. Han, "A rationale-based architecture model for design traceability and reasoning," *Journal of Systems and Software*, vol. 80, no. 6, pp. 918-934, 2007.

[47] A. Tang, Ph. D. A Rationale-based Model for Architecture Design Reasoning, 2007.

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

[49] A.M. Lund. (2001) Questionnaire for User Interface Satisfaction. [Online]. http://oldwww.acm.org/perlman/question.cgi?form=USE