Introducing Tool Support for Managing Architectural Knowledge: An Experience Report

Muhammad Ali Babar[1], Andrew Northway[2], Ian Gorton[3], Paul Heuer[2], Thong Nguyen[2]
[1]Lero, University of Limerick, Ireland, [2]Air Operations Division, Defence Science and Technology Organisation, Australia, [3]Pacific Northwest National Laboratory, USA.
[1]malibaba@lero.ie [2]{andrew.northway,paul.heuer,thong.nguyen}@dsto.defence.gov.au, ian.gorton@pnl.gov

Abstract

Management of software architecture knowledge is vital for improving an organisation's architectural capabilities. Despite the recognition of the importance of capturing and reusing software architecture knowledge, there is currently no suitable support mechanism available. To address this issue, we have developed a conceptual framework for managing architecture design knowledge. A web-based knowledge management tool, Process-based Architecture Knowledge Management Environment (PAKME), has been developed to support that framework. PAKME is being trialled to help systematise the architecture knowledge management and evaluation process of an industrial collaborator. This paper reports the objectives, logistics and initial findings of this project, Specifically we have deployed and used PAKME in an Australian Defence acquisition environment for evaluating architecture of a aircraft system.

1. Introduction

Although significant progress has been made to support the various activities of the software architecture process over the last decade, there has been little effort on developing techniques and tools for effectively capturing and managing the knowledge required or generated during the architecture process. We have developed a framework for managing architectural knowledge [1]. This framework consists of techniques for capturing design decisions and contextual information, an approach to distilling and documenting architectural information from patterns, and a data model to characterise architectural knowledge [1]. The central objective of this framework is to provide a theoretical underpinning and conceptual guidance to design and implement repository-based tool support for managing architectural knowledge. The novelty of this framework resides in its ability to incorporate all the components into an integrated approach, which has been implemented in a web-based tool called PAKME (Process-based Architecture Knowledge Management Environment) [2].

PAKME is aimed at enabling organisations to create a repository of critical design knowledge about their software applications. We have trialled PAKME with the Defence Science and Technology Organisation (DSTO), who wish to improve their architecture evaluation process. DSTO was particularly interested in tool support for managing technical knowledge, capturing contextual information surrounding critical design decisions, and managing architectural assets for reusability purposes. Moreover, they also wanted to help their evaluators to keep record of justifications for their evaluation decisions and to have a source of learning from 'good' design decisions made in previous projects. This paper describes the process and logistics of tailoring, deploying, and trialling PAKME in the context of evaluating architectures for mission systems in a defence environment. It also reports what the research group and the industrial partner have learned from this experience in terms of technical and practical recommendations for building and trialling knowledge management tool support in the context of software architecture process. We also discuss suggestions for improving PAKME and its exploitation in an industrial environment.

2. Managing architectural knowledge

Software architecture knowledge is rarely documented, which results in lack of access to knowledge underpinning the design decisions and process [1, 3]. Many researchers and practitioners [1, 4, 5] believe that architectural knowledge should be systematically captured and rigorously managed. Otherwise, it usually results in downstream consequences such as:

- The evolution of the system becomes complex and cumbersome and may result in violation of fundamental design decisions.
- Inability to identify design errors and track changes.
- Inadequate clarification of arguments and context sharing about the design and process.
- Difficult to access or discover knowledge required to make or evaluate architectural design decisions.
- loss of key personnel may mean loss of knowledge [6-8].

Apart from facing these issues, our industrial partner also felt that they needed to have a rigorous process of keeping track of their architecture evaluation decisions, rationale for those decisions and profiles of evaluators in case of inquiries for contract evaluation by a governing body.

978-0-7695-3141-0/08 $25.00 © 2008 IEEE
DOI 10.1109/ECBS.2008.27

3. The Project

We undertook a research and development project jointly carried out by National ICT, Australia (NICTA) and DSTO. The project was part of an ongoing collaboration between NICTA and DSTO aimed at exploiting software architecture evaluation technologies developed by NICTA for improving DSTO's capabilities in evaluating architectural risks during avionics system acquisition.

3.1 Organizational Context

DSTO is a research and development organisation, which provides scientific and technical advice on acquisitions by the Australian Defence Organisation (ADO). One of the key responsibilities of DSTO is to evaluate Request for Proposal (RFP) responses from suppliers to identify technical and project risks of each proposal. The Airborne Mission Systems (AMS) division of DSTO is responsible for evaluating software architectures for aircraft acquisition projects. As modern mission systems are increasingly becoming more reliant on software, evaluating proposed architectural solutions has become much more important as software intensive projects are historically considered the most risk prone in the defence domain [9].

AMS is required to understand and organise large amounts of design knowledge for a mission system's architecture to support the evaluation process. Hence, there has been growing recognition of the importance of systemising software architecture evaluation processes within DSTO and building its capabilities in software architectures and managing architectural knowledge.

3.2 Project Background and Objectives

Evaluating software architectures for avionics systems requires access to documentation on a system of interest and to large amount of technological information. The former characterises a system, while the latter forms the basis for identifying risks (e.g., rate of obsolescence of a type of component) and mitigations (e.g., processor roadmaps, candidate technologies). Extracting information and capturing knowledge and its context relevant to a system are extremely important for establishing an audit trail for the assertions made in an evaluation report.

Currently, AMS's evaluation process relies on subject matter experts' domain knowledge and experience with evaluating architectures to identify potential risks in achieving the required capability of a system to be built. Evaluation criteria are derived from documentation of previous projects and system's description provided to an evaluation team. AMS technical staff felt that an architecture evaluation without a suitable tool support places the onus on domain experts to ensure the correct breadth of analysis, makes the analysis criteria less explicit, and typically relies on a document-focused reference/bibliography system that is an intrinsic part of an analysis report. Knowledge reuse requires reading previous reports and re-establishing an analysis process for each architecture evaluation.

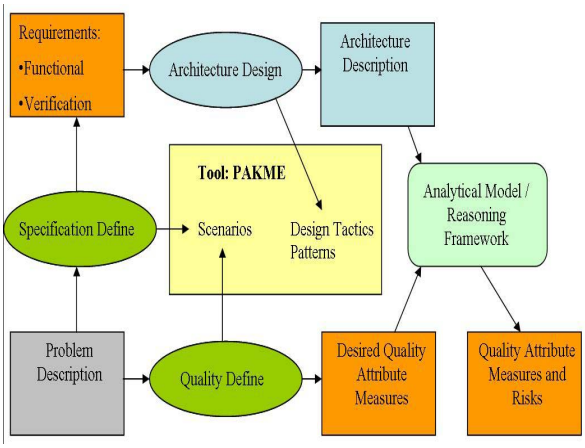

Figure 1: AMS's software architecture evaluation process supported by PAKME.

The main objective of this project was to enable AMS's technical risk assessment team to capture the required knowledge in an enduring, searchable and versionable format in order to improve AMS's ability to understand the evaluated systems as they are developed and evolved over the lifetime of an aircraft. Moreover, AMS wished to reduce reliance on domain experts for knowledge transfer and training of new staff members.

To achieve these objectives, AMS management decided to trial PAKME for capturing and managing architecture knowledge for supporting their architecture evaluation process. A simplified illustration of how PAKME was embedded in AMS's architecture evaluation process is shown in Figure 1. PAKME was expected to:

- help build quality models using scenarios (abstracts and concrete),
- help reason about the suitability of various design options proposed by contractors, capture the rationale for ranking, approving, or rejecting various design proposals,
- centralise architecture design knowledge.

Although a knowledge management initiative requires considerable time and resources, it is anticipated there will be considerable time and cost savings in the long-term [10]. The benefits for AMS from this project were expected to be as follows:

- Capture rationale for architecture decisions;
- Help build architectural capabilities;
- Improve architectural reusability;
- Provide an audit trail for findings of architecture evaluation;
- Reduce demands on subject matter experts;
- Encourage best architectural practices;
- Improve efficiency of architectural processes.

- Accelerate the training process of new employees within the organisation.

3.3 An Overview of PAKME

In order to provide tool support for managing architectural knowledge, we have developed PAKME, which incorporates concepts from knowledge management [11, 12], experience factories [13], and pattern-mining [1] paradigms. Architectural details and features of this tool have been reported in [2]. The following section describes some of PAKME's features, which were used in the reported case study.

PAKME is a web-based tool, which provides a knowledge repository, templates and various features to capture, manage, and present architectural knowledge. PAKME's knowledge repository is logically divided into two types of knowledge:

1. *Generic*: including general scenarios, patterns, quality attributes, design options.
2. *Project specific:* including concrete scenarios, contextualised patterns, quality factors, architecture decisions.

Generic architectural knowledge is accumulated by using knowledge capture techniques described in [1]. Project-specific architectural knowledge consists of the artefacts either instantiated from the generic knowledge or newly created during different activities of the software architecture process. Access to a repository of generic architectural knowledge enables designers to use accumulated "wisdom" from different projects when devising or evaluating architecture decisions for projects in the same or similar domains. The project specific part of the repository captures and consolidates other architectural knowledge artefacts and rationale that are specific to a particular project such as concrete scenarios, design history, and findings of architecture evaluation. A project specific architectural knowledge repository is also populated with knowledge drawn from an organisational repository, standard work products of the design process, logs of the deliberations and histories of documentation to build organisation's architecture design memory [14]. Here we briefly discuss the four main services of PAKME:

- The knowledge acquisition service provides various forms and editing tools to create new generic or project specific knowledge in the repository. The knowledge capture forms are based on various templates that we have designed to help maintain consistency during knowledge elicitation and structuring processes.
- The knowledge maintenance service provides different functions to modify, delete and instantiate the artifacts stored in the knowledge repository. Moreover, this service also implements the constraints on the modifications of different artifacts based on the requirements of a particular domain.
- The knowledge retrieval service helps a user to locate and retrieve desired artifacts along with the information about the artifacts associated with them. PAKME provides three types of search mechanisms. A basic search can be performed within a single artifact based on the values of its attributes or keywords. An advanced search string is built using a combination of logical operators within a single or multiple artifacts. Navigational search is supported by presenting the retrieved artifacts and their relationships with other artifacts as hyperlinks.
- The knowledge presentation service presents knowledge in a structured manner at a suitable abstraction level by using templates (such as provided in [15]) and representation mechanisms like utility and results trees described in [16].

These services not only satisfy the requirements identified to provide knowledge management support for architecture design and evaluation methods reported in [17, 18], but also support many of the use cases proposed in [19]. PAKME can support several of the ten techniques proposed in [18] for the Software Engineering Institute (SEI)'s methods for architecture evaluation. For example, PAKME provides suitable templates to capture and present architectural artefacts and contextual information for making architecture evaluation techniques consistent across evaluators.

PAKME's services also provide different features to support activities such as generating a utility tree, identifying a suitable reasoning framework, recording evaluation findings, and building a results tree to visualise risks and risk themes of an architecture evaluation method like ATAM [16]. In addition, PAKME helps the evaluation team to capture findings from architecture evaluation decisions and the justification for those finding.

3.4 Tailoring PAKME

PAKME provides a generic solution to address the architectural knowledge management issues during the software architecture process. It needs to be customised depending on organisational requirements and role in the software architecture process. For example, AMS only evaluates architectures proposed by contractors. Thus, it needed features of the tool that support software architecture evaluation tasks. Initial discussions between AMS staff and the PAKME team revealed the need for a workshop session involving all major stakeholders. The main objectives of this workshop were:

- To understand the architecture evaluation process.
- To built AMS's quality model for evaluation.

Table 1: Quality attributes and Quality factors of the AMS's quality model

Quality attributes	Quality factors
Performance	Initialization, Capacity, Throughput, Resource usage, Scalability, Schedulability
Reliability	Fault tolerance, Recoverability, Maturity, Survivability, Availability
Usability	Understandability, Learnability, Operability
Maintainability	Changeability, Testability, Growth, Analyzability
Interoperability	Communication, Data, Programming languages, Development and testing tools
Safety	Failure detection mechanism, Certification, User Interface, Modularity
Security	Security architecture, Operational data security, Encrption
Reusability	COTS components, Hardware
Portability	Adaptability, Standards conformance, Hardware replaceability, Reconfigurability

To identify the extra features and modifications required in PAKME. In April 2006, a two day workshop was organised at AMS's office. Eight AMS's staff, both experienced and new members, participated in the workshop, which was designed and ran based on a modified Quality Attributes Workshop (QAW) process [20].

During the first session, we helped the AMS team to model their architecture evaluation process and identify the activities that can be supported by PAKME. Figure 1 shows AMS's architecture evaluation process supported by PAKME. During the workshops with the AMS staff, we also identified and discussed various ways of using architectural knowledge and artefacts managed by PAKME for each activity of architecture evaluation. These include, for example, defining quality requirements using concrete scenarios based on the general scenarios from AMS's quality model captured in PAKME, or browsing through rationale to understand architectural decisions made by others.

During the second session, the AMS team defined a quality model for architecture evaluation. The quality model construction involved identifying key quality attributes to enable evaluators to assess the potential risks of architectural designs against the requirements. The quality model consists of six main quality attributes and their respective quality factors as shown in Table 1. One quality attribute of this model functionality was further decomposed into four quality attributes (i.e., Interoperability, Safety, Security, and Reusability).

The third activity in the workshop was to identify the initial set of requirements for modifying PAKME. The gathered requirements were also prioritized. Each requirement was categorized either high or low priority based on their importance to AMS's process needs and available resources. The high priority requirements were then implemented in the current version of PAKME, which has been trialled by AMS staff in the case study reported in this paper. Some of the high priority requirements implemented for tailoring PAKME were:

- Classification of project data according to the Defence classification scheme of four levels of access (top secret, secret, classified, and unclassified).
- A mechanism for recording compliance of architecture decisions with respect to requirements and standards.
- One or more relevant documents can be attached to different artefacts (such as design options, concrete scenarios, and findings) to provide reference material.
- Reports of architecture evaluation findings along with relevant contextual information.
- Each concrete scenario can have several attributes attached, which can be used to generate reports based on the desired values of attributes.
- Each concrete scenario can have one or more findings along with justification provided by an architecture's evaluator.

Some of the lower priority requirements, which are being implemented include:

- Ability to import/export data from the tool based on a classification code
- Risk management scheme for ranking decisions
- Integration with requirements management and architecture modelling tools.

4. A Case Study

In order to assess PAKME for supporting the AMS's architecture evaluation process, a case study was carried out by the AMS staff. This study involved using PAKME for capturing and managing architectural knowledge to support architecture evaluation of an aircraft system. It was a post-mortem analysis of the architecture evaluation conducted without using PAKME. The study was also aimed at investigating how the introduction of PAKME could help capture and manage architectural knowledge and whether or not the evaluation process is improved by using PAKME. Both organisations deemed it important to design and conduct such a case study before deploying PAKME in AMS's future evaluation projects.

A number of quality factors were chosen as measures for the mission system architecture evaluation. These quality factors were growth, security, adaptability, and reconfigurability as shown in Table1. The evaluation process involved DSTO analysts comparing alternative design decisions from multiple hypothetical tenders, to

simulate the type of evaluation completed during the real evaluation of an aircraft acquisition project. The evaluation was performed by measuring each scenario against the quality attributes, as well as assigning a measure of risk to the design solution.

4.1 Use of PAKME's knowledge base

To populate the PAKME's knowledge base with the AMS domain knowledge, two of the AMS's staff and one of the researchers defined each of the quality attributes and its quality factors from the quality model built during the workshops. The process of transferring AMS's quality model into PAKME was guided by the ISO 9126 quality model [21], the quality attributes and general scenarios provided by SEI [22], and the experience of avionics domain experts gathered during the workshop.

This quality model was created in PAKME's repository and made available to all the AMS's evaluators. PAKME was also populated with general scenarios for characterising each of the quality attributes included in the quality model as shown in Figure 2. Apart from scenarios characterising the AMS quality model, several hundred general scenarios were stored in PAKME to support architecture evaluation. The general scenarios stored in PAKME are used to generate concrete scenarios for different evaluation projects.

Domain specific general scenarios captured in PAKME's repository can provide assurance that the breadth of topics applicable to a system has been analysed, and deriving concrete scenarios from those general scenarios allows analysis of system behaviour under task-specific criteria. Subsets of general scenarios can be assembled into meta-projects that are applicable to a class of aircraft systems such as helicopters and fast jets. The general scenarios from the meta-projects are imported into project-specific repository to support architecture evaluation of a system. Figure 3 shows a user-defined scenario for a generic mission system architecture captured under meta-projects in PAKME.

PAKME's repository was also populated with avionics-specific general design options. These design options were captured from the architecture solutions proposed for the system reviewed for this case study, AMS domain experts, and case studies on avionics systems reported in sources such as [16]. Each design option was captured as a design decision case as shown in Figure 4. These generic design options were used as input to design decision making or evaluation processes.

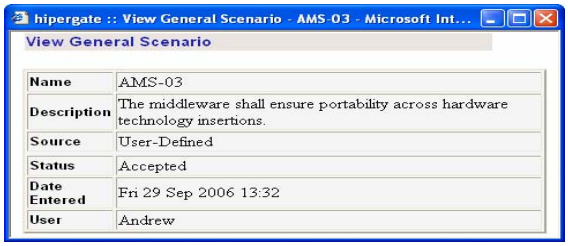

Figure 3: A user-defined general scenario

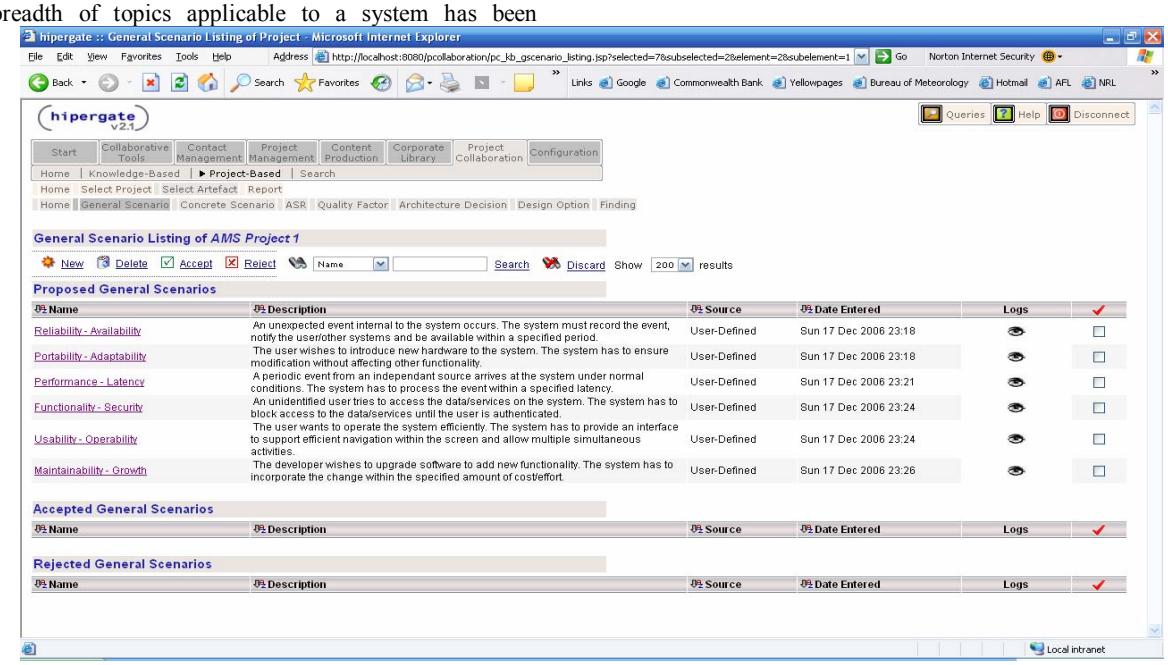

Figure 2: Some of the general scenarios characterising the AMS domain captured in PAKME's repository

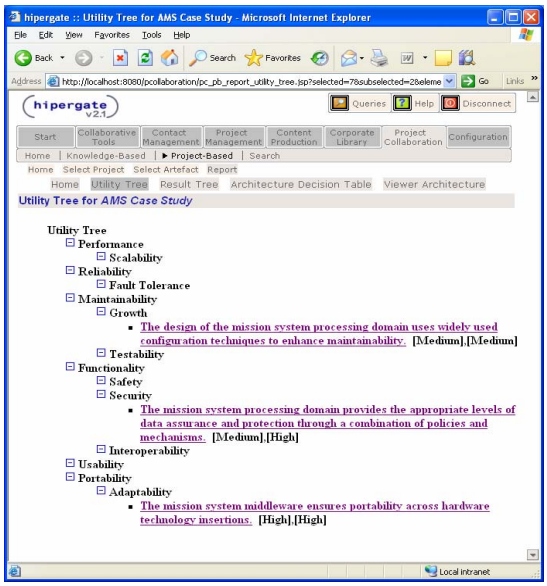

Figure 4: A utility tree for the case study

4.2 Use of PAKME's project base

For this study, the AMS team created a new project in PAKME and populated its project-base with the project specific quality model to specify quality factors with concrete scenarios based on the scenarios used to characterise the AMS's quality model built during the abovementioned workshop. Figure 4 shows a utility tree of the concrete scenarios that characterise some of the quality factors considered in the case study.

Each architecture decision proposed by different contractors for satisfying required scenarios of the project was identified and entered into PAKME. Each decision was also linked to the concrete scenarios satisfied by that architecture decision. An example of a design decision affecting architectural quality is the use of a layered architecture including an isolation layer to reduce the impact of change, and thus improving flexibility, technology refreshment and growth capability. This architecture design decision was stored in PAKME along with the rationale.

Each architecture decision of this project was also captured as a design option in the generic knowledge base of PAKME. The AMS team also captured several design options based on their domain knowledge. During architecture evaluation, each architecture design decision was assessed with respect to the design options, which are expected to satisfy the same concrete scenario. Having populated PAKME with the project specific architecture knowledge, the AMS team evaluated the architecture design decisions proposed for an aircraft system by several contractors. For this evaluation, the team used PAKME for accessing the architectural knowledge required for the evaluation and capturing the findings and rationale for evaluation. They used their existing process of evaluating architecture with one exception of introducing PAKME in the process as shown in Figure 1.

Figure 5: An architectural design decision example

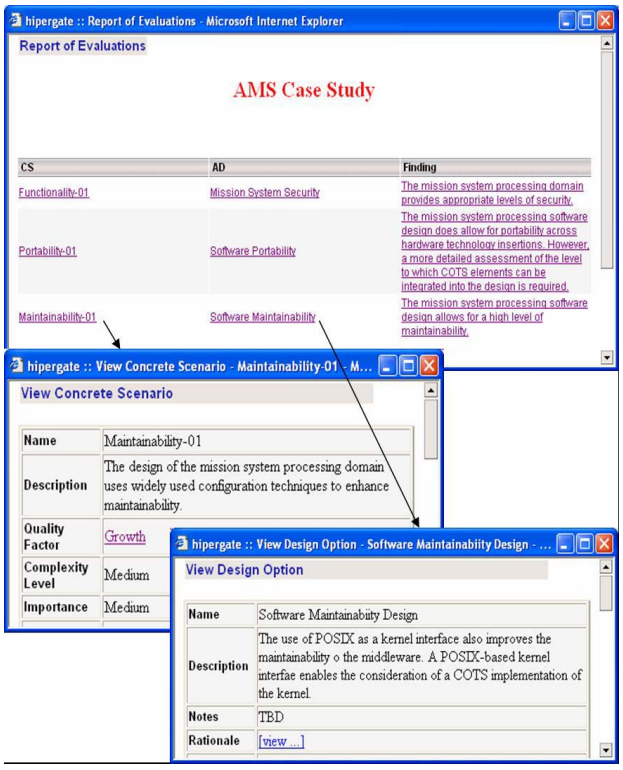

Figure 6: Reports of the evaluation case study

The architecture evaluation process involved determining whether or not a concrete scenario is satisfied by the proposed architecture decision. If there were more than one proposed architecture decision for a scenario, architecture decisions were assigned a ranking based on an evaluator's opinion about each architecture decision's capability of achieving a certain level of required quality factor. Figure 5 shows how an architectural decision is captured along with rationale using a template provided by PAKME. Evaluators recorded their findings using the provided templates. Each finding describes whether or not a certain architecture decision complied with the relevant requirement, its ranking, and rationale/justification for the findings. Based on the evaluation findings, architecture decisions were categorised as risk or non-risks. Risks were further categorised under various risk themes as suggested in [23].

At the end of evaluation, different reports were generated for project managers. Figure 6 shows one view of the web-based evaluation report generated by PAKME. At the top of Figure 6 is finding matrix (showing concrete scenario, architecture decision, and respective findings) based on the architecture evaluation carried out using PAKME for this case study. Web-based report presents main elements of findings as hyperlinks, which can be browsed for detailed information as is shown in Figure 6 for concrete scenario and architecture decisions. PAKME also generates PDF reports for evaluation teams and management based on specified criteria.

4.3 Findings

What was good about PAKME?

Based on their experiences of evaluating architectures with and without using a knowledge management tool like PAKME, AMS staff involved in this case study found that general scenarios and design options captured in the knowledge-base helped them in generating concrete scenarios and understanding proposed solutions. Having access to a codified quality model provided all evaluators with the same understanding of the quality model. Previously, they had experienced that relying on domain experts' definitions of different quality attributes tended to lead to discrepancies. This eefect was also observed by us during the quality model definition workshop when domain experts were asked to describe what each of them meant by different quality attributes such as performance, security and usability in their domain. Moreover, the evaluators also reported that PAKME's templates for capturing decision rationale were useful in identifying the kind of information required for building rational for evaluation findings.

It is also anticipated that the management of evaluation decisions and their justification using PAKME would minimise the need for contacting the evaluators of past projects for explanation. Additionally, it will also be easier to contact a relevant person as PAKME enables its user to annotate different artefacts with contact details of the creator of a particular artefact. This feature is provided to manage knowledge through a personalization strategy, which promotes personal contacts between a knowledge producer and knowledge users [24] .

It was also reported that the architecture evaluation knowledge captured in PAKME can not only be reused by subject matter experts during architecture evaluation but can also be used for providing training to new staff. However, it was also realized that one potential drawback of using a knowledge management tool as a training mechanism can be that it may provide new staff with less number of opportunities of being mentored by experienced staff. In that situation, a junior staff member may be assigned to evaluate an architecture using a knowledge management tool without being fully exposed to a large amount of tacit domain knowledge held by domain experts.

What needs to be improved?

It was found that templates need to be configurable by users based on their needs. PAKME's templates have been designed based on the kinds of rationale reported in [4, 25]. While we are convinced that certain kinds of information need to be captured as a standard part of rationale, we also agree that PAKME's templates should be configurable based on organizational needs.

The evaluation team also reported concerns about the possibility of duplication of workload as AMS and DMO's suppliers manage requirements in a commercial tool. For the reported case study, the AMS team simply 'cut and pasted' requirements in PAKME for their quality model. Hence, AMS suggested that PAKME should be integrated with their requirements management, tool if it is to be widely used within DSTO environment.

AMS's overall feedback

The AMS team have given positive feedback on the usefulness and effectiveness of various features of PAKME for supporting their architecture evaluation process. They have found that tool-supported evaluation is more effective in terms of accessing required knowledge and artefacts than contacting relevant people. Hence, the AMS management are now more convinced that an architectural knowledge management tool like PAKME will provide them with several benefits and help them institutionalising a disciplined software architecture evaluation process. In light of a new mandated role for DSTO in Defence acquisitions, it is expected that PAKME will provide AMS with a centralised infrastructure for storing and revisiting evaluation decisions quickly and to codify the software architecture evaluation process and practices.

Research team's observations

Customising and trialling PAKME for the AMS architecture evaluation process, the research team faced several organisational, technical, and logistical challenges that needed to be overcome in order to complete this project. We briefly describe a few of the

key challenges as lesson learned. We believe that some of them are general enough for any research-industry collaboration in the defence domain.

One of the key organisational challenges was that NICTA's team did not have clearance to access the documentation of the software architectures being evaluated. Nor did they have domain expertise. Hence, they had to gain a certain level of domain understanding in order to help AMS build their domain-specific quality model using scenarios, identify and capture design options and patterns from architectural descriptions of the systems being evaluated, and determine requirements for customising PAKME. The two day workshop helped NICTA's team understand AMS's domain and identify its unique requirements for evaluating architectures and managing knowledge.

The researchers also found certain requirements were unique to the DSTO domain and difficult to implement without making significant changes in existing user interface and data model. For example, DSTO data is classified based on different levels of security clearance for the data user. There are four different levels of security clearance (Top secret, secret, classified, and unclassified). Moreover, each quality attribute scenario can have more than one architecture decision proposed by different contractors. AMS required that each combination of a concrete scenario and proposed solution needed to have its own findings attached and any relevant artefacts in a matrix shown in Figure 6.

Based on our experiences during these modifications and AMS's feedback on the configuration issues with PAKME's templates, we believe that PAKME needs to be designed to be heavily customized at deployment time, and that a rigid tool is unlikely to be widely successful.

We also realized early on that the key to making PAKME useful is to incorporate it into the process already being used for evaluating a software architecture by AMS team. It was also realized that the PAKME needed not to be perceived as a replacement to the existing tools rather a complementary tool, which can eventually be integrated with the existing tools.

Overall, the researchers believe that the modified version of PAKME provides AMS with an effective and efficient mechanism to organise and understand large amounts of architectural knowledge.The current version of PAKME is suitable for capturing and managing several types of architectural knowledge and artefacts of an airborne mission system, and supporting a rigorous and systematic architecture evaluation process using well-known methods like ATAM.

5. Conclusion and Future Work

This research is aimed at improving the effectiveness of architecture-based software engineering through a knowledge management support mechanism. We have developed a tool, PAKME, to support architecture knowledge management. This paper reports on the logistics and our experiences of tailoring and trialling PAKME for evaluating architecture of an aircraft system. During this trial, PAKME has proven to be adaptable and useful to complex domains like Defence. Based on our experience of using PAKME for supporting AMS's architecture evaluation process for the reported case study, we are more convinced that PAKME has the potential to help organisations improve their software architecture processes and build architectural capabilities. Through this trial, we have identified certain limitations in PAKME as well as requirements for further enhancements, which we plan to carry out in our ongoing research. Some of the planned enhancements are discussed below.

Since majority of the Avionic systems evaluated by AMS are parts of families of systems, we have also identified the need for extending PAKME to support architecture design and evaluation in software product line engineering. This involves identifying, modelling, and managing variability and variants. Managing technical and contextual knowledge to support product derivation is considered one of the major challenges in software product line engineering [26]. We also plan to assess the utility of PAKME's pattern repository for supporting pattern-based variability modellings [27] and pattern-based description of design decisions [28] for identifying design variants to realize different variation points captured in the variability model. Following are some of the other planned enhancements to PAKME:

- Implementing metrics to measure the usage of the different artefacts of architectural knowledge. Such a feature will provide a feedback loop to improve the type of knowledge captured and features provided.
- Improving the speed and accuracy of knowledge retrieval by using the task-based retrieval techniques.
- Integrating PAKME with tools commonly used for managing requirements for large-scale systems such as DOORS. Such integration will provide an effective mechanism to maintain traceability from requirements to scenarios, to architecture design decisions along with the contextual knowledge underpinning design decisions. Moreover, it will also minimize the duplication of data entry. A similar integration with an architecture modelling/description tool has also been planed.

6. Acknowledgement

The first and third authors were working with the National ICT Australia, when the reported work was carried out. Lero is supported by Science Foundation Ireland (under grant no. 03/CE2/I303_1).

7. References

[1] M. Ali-Babar, I. Gorton, and B. Kitchenham, A Framework for Supporting Architecture Knowledge and Rationale Management, in Rationale Management in Software Engineering, A.H. Dutoit, et al., Editors. 2006, Springer. pp. 237-254.

[2] M. Ali-Babar and I. Gorton, A Tool for Managing Software Architecture Knowledge, *Proceedings of the 2nd Workshop on SHAring and Reusing architectural knowledge - Architecture, rationale, and Design Intent (SHARK/ADI 2007), Collocated with ICSE 2007*.
[3] J. Bosch, Software Architecture: The Next Step, *European Workshop on Software Architecture*, 2004.
[4] J. Tyree and A. Akerman, Architecture Decisions: Demystifying Architecture, *IEEE Software*, 2005. **22**(2): pp. 19-27.
[5] P. Avgeriou, et al., Architectural Knowledge and Rationale - Issues, Trends, Challenges, *ACM SIGSOFT Software Engineering Notes*, 2007. **32**(4): pp. 41-46.
[6] L.G. Terveen, P.G. Selfridge, and M.D. Long, Living Design Memory: Framework, Implementation, Lessons Learned, *Human-Computer Interaction*, 1995. **10**(1): pp. 1-37.
[7] T.R. Gruber and D.M. Russell, Design Knowledge and Design Rationale: A Framework for Representing, Capture, and Use, *Tech Report KSL 90-45*, Knowledge Systems Laboratory, Standford University, California, USA, 1991.
[8] A.P.J. Jarczyk, P. Loffler, and F.M.S. III, Design Rationale for Software Engineering: A Survey, *Proc. 25th Hawaii Int'l. Conf. on System Sciences*, 1992.
[9] Defence Electronic Systems Sector Stragic Plan, C. Department of Defence, Australia, Editor. Feb 2004. pp. 97.
[10] M. Barbacci, Clements, P., Lattanze, A., Northrop, L., Wood, W., Using the Architecture Tradeoff Analysis Method (ATAM) to Evaluate the Software Architecture for a Product Line of Avionics Systems: A Case Study, *Tech Report CMU/SEI-2003-TN-012*, Carnegie Mellon Software Engineering Institute, July 2003.
[11] G.J.B. Probst. Practical Knowledge Management: A Model That Works. Last accessed on 14th March, 2005, Available from: http://know.unige.ch/publications/Prismartikel.PDF
[12] I. Rus and M. Lindvall, Knowledge Management in Software Engineering, *IEEE Software*, 2002. **19**(3): pp. 26-38.
[13] V.R. Basili, G. Caldiera, and H.D. Rombach, Experience Factory, in Encyclopedia of Software Engineering, J. Marcinak, Editor. 1994, John Wiley. pp. 469-476.
[14] G. Arango, E. Schoen, and R. Pettengill, A Process for Consolidating and Reusing Design Knowledge, *Proceedings of the 15th International Conference on Software Engineering*, 1993.
[15] M. Ali-Babar, I. Gorton, and R. Jeffery, Toward a Framework for Capturing and Using Architecture Design Knowledge, *Tech Report TR-0513*, University of New South Wales, Australia, 2005.
[16] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice. 2 ed. 2003: Addison-Wesley.
[17] C. Hofmeister, et al., A General Model of Software Architecture Design Derived from Five Industrial Approaches, *5th Working IEEE/IFIP Conference on Software Architecture (WICSA 05), Pittsburgh, PA, USA*, 2005.
[18] R. Kazman, L. Bass, and M. Klein, The essential components of software architecture design and analysis, *Journal of Systems and Software*, 2006. **79**(8): pp. 1207-1216.
[19] P. Kruchten, P. Lago, and H.V. Vliet, Building up and Reasoning about Architecture Knowledge, *Proceedings of the 2nd International Conference on Quality of Software Architectures*, 2006.
[20] M.R. Barbacci, et al., Quality Attribute Workshops (QAWs), *Tech Report CMU/SEI-2003-TR-016*, SEI, Carnegie Mellon University, USA., 2003.
[21] ISO/IEC, Information technology - Software product quality: Quality model. ISO/IEC FDIS 9126-1:2000(E).
[22] M.R. Barbacci, M.H. Klein, and C.B. Weinstock, Principles for Evaluating the Quality Attributes of a Software Architecture, *Tech Report CMU/SEI-96-TR-036*, SEI, Carnegie Mellon University, 1996.
[23] P. Clements, R. Kazman, and M. Klein, Evaluating Software Architectures: Methods and Case Studies. 2002: Addison-Wesley.
[24] K. Desouza and Y. Awazu, Managing Knowledge in Global Software Development Efforts: Issues and Practices, *IEEE Software*, 2006. **23**(5): pp. 30-37.
[25] A. Tang, M. Ali-Babar, I. Gorton, and J. Han, A Survey of Architecture Design Rationale, *Journal of Systems and Software*, 2006. **79**(12): pp. 1792-1804.
[26] L. Hotz and T. Krebs, Supporting the Product Derivation Process with a Knowledge-based Approach, *Workshop on Software Variability Management*, 2003.
[27] B. Keepence and M. Mannion, Using Patterns to Model Variability in Product Families, *IEEE Software*, 1999. **16**(4): pp. 102-108.
[28] N.B. Harrison, P. Avgeriou, and U. Zdun, Using Patterns to Capture Architectural Decisions, *IEEE Software*, 2007. **24**(4): pp. 38-45.