Supporting Customizable Architectural Design Decision Management

Lianping Chen
Lero—The Irish Software Engineering Research Centre
University of Limerick
Limerick, Ireland
Lianping.Chen@lero.ie

Muhammad Ali Babar (Supervisor)
Software Development Group
IT University of Copenhagen
Copenhagen, Denmark
malibaba@itu.dk

***Abstract*—When engineering complex software systems, the key Architectural Design Decisions (ADD) and the reasoning underlying those decisions need to be fully understood by all stakeholders. Achieving such understanding usually requires the use of ADD management tools. Most existing ADD management tools apply prescriptive ADD models and do not provide sufficient customizability. However, forcing architects to follow an ADD model that does not fit their specific needs can cause significant problems (e.g., extra cost is needed, and architects' willingness and motivation can negatively be affected). This research project aims at solving this issue by developing a highly customizable solution, which can enable practitioners to define ADD models according to their preferences and working situations. The detailed needs for ADD model customization will be identified by multiple case studies and semi-structured interviews; the proposed solution will be evaluated using different empirical research methods.**

Architectural knowledge; architectural design decision; customizability; design rationale; software architecture

I. INTRODUCTION

Software architectures play an increasingly important role in engineering large scale software intensive systems. Designing and maintaining software architectures of large systems involves complex and knowledge intensive activities [1, 2]. Software architecture design is a process of making a set of significant design decisions [3, 4], which are often referred to as Architectural Design Decisions (ADD) [3, 5-7]. ADDs are usually difficult or very expensive to change if found faulty or inappropriate later in the development lifecycle. Hence, it is vital that the key ADDs are not only made with extreme care, but they should also be appropriately documented along with the rationale underpinning those ADDs. Capturing and managing ADDs and their rationale have been interesting topics for researchers since Perry and Wolf formally laid the foundation of the software architecture discipline in [8]. However, only recently (beginning with [3]), software architecture researchers and practitioners have started paying serious attention to the importance of ADDs and their rationale. Recent research in this area has revealed that it is important that participants in software projects understand the key ADDs and the reasoning underpinning them (i.e. rationale). Absence of such understanding can result in serious errors in design, implementation, maintenance, redesign, coordination and project management [9, 10].

Achieving such understanding requires the use of software engineering tools for capturing and sharing ADDs. These tools are commonly known as Architectural Design Decision Management (ADDM) tools. Researchers have recently developed several ADDM tools (see Section II.B). All of these tools prescribe fixed data models to be followed for capturing and sharing ADDs and their rationale, without sufficient support for customization. We call these data models ADD models, because they define and govern the structure and formalization of captured ADDs. It is also worth clarifying that by customization we mean the modification of the ADD model underpinning an ADDM tool to suit the specific needs of architects without requiring programming or complex operations.

However, an ADDM tool that prescribes a fixed ADD model falls short of meeting architects' specialized needs for ADDM. In different working situations, architects usually have specialized needs for an ADD model [11, 12]. For example, a model that is appropriate for architects working in a mature domain may not suitable to those working in emerging domains [13]. Forcing architects to follow a fixed ADD model that does not fit their needs can cause significant problems (e.g., extra cost is needed [14], and architects' willingness and motivation can negatively be affected [11, 15]).

The aim of this research project is to develop a highly customizable ADDM solution that can enable architects to define ADD models according to their specific needs raised by their respective working situations. A specialized ADDM tool support will be automatically generated from the ADD model defined by the architects.

The rest of this paper is organized as follows. Section II elaborates on the motivation and objective of this research project. Section III describes the methods we have been using to carry out this research. Section IV summarizes the progress to date. Section V presents the proposed solution. Section VI outlines the expected contributions and planned future work.

II. RESEARCH MOTIVATION AND OBJECTIVES

In this section, we first elaborate on why ADD model customization is important, then we review how existing ADDM tools support ADD model customization, discuss why supporting ADD model customization is difficult, and finally give our research objectives.

A. Importance of Customizing ADD Model

Most of the existing ADDM tools provide prescriptive and fixed ADD models to be followed. A prescribed ADD model may be too fine-grained, too coarse-grained, or its Ontology may be too abstract. If practitioners are forced to use an unsuitable ADD model, they have to adapt their way of thinking and describe their ADDs according to that ADD model. That means they have to convert ADDs from their preferred model to an imposed model. The conversion process usually requires extra effort that can decrease practitioners' willingness and motivation for externalizing and documenting ADDs. This can have dramatically negative effects on the practice of ADD documentation in organizations, according to Poort et al. [15].

Recently, many researchers have been highlighting the importance of ADD model customization. Burge et al. stated that the design decisions management systems must allow practitioners to invent new data models and to arbitrarily modify data models to accommodate information that is specific to particular software projects, software engineering methods, and the problem-solving styles of software engineers [12]. According to Burge et al. [9], existing design rationale tools (e.g., gIBIS [16]) fall short of such customizability. Lago states that technologies should be flexible to fit practitioners' needs and be adaptable to their preferences [17]. Tang et al. [18], and Henttonen and Matinlassi [19] also consider the customizability of ADD model as an important feature of ADDM tools. Our empirical study [11] has also revealed that there can be huge mismatches between the required ADD model and the one prescribed by an ADDM tool.

Empirical studies in other areas also suggest the importance of customization [20-26]. Results of those studies show that actual work is more situational and contingent than can be accommodated by fixed models [21-25] and that knowledge workers have their own modus operandi for doing their work [20]. Supporting customization is important to enable knowledge workers to develop their own strategies for working in complex and dynamic environments [20, 26]. Since software architects are also knowledge workers, we assert that the above findings apply to them as well.

B. Customization Provided by Existing Tools

In order to determine the customization provided by existing ADDM tools, we have investigated the customizability of a set of currently available ADDM tools: PAKME [27], ADDSS [28], Kruchten's Ontology Tool [13], and ADkwik [14, 15]. PAKME (Process-based Architecture Knowledge Management Environment) is a web-based ADDM tool, which has been built on top of an open source groupware platform called Hipergate [29]. ADDSS (the Architecture Design Decision Support System) is a research web-based tool for storing, managing, and documenting ADDs during the architecting process [28]. Kruchten's Ontology Tool is developed to capture design decisions in a structured form (i.e. the architecture design decision ontology that was proposed by Kruchten [6]). ADkwik (Architectural Decision Knowledge Wiki) is an application wiki for managing architectural decision knowledge collaboratively. It is available on IBM alphaWorks [30].

Though several frameworks exist for evaluating ADDM tools (such as reported in [18] and [19]), none of them provides detailed criteria for assessing the extent to which a tool supports ADD model customization. We decided to use the customization scenarios identified in [11] as criteria for this study. The customization scenarios are as follows.

Delete an attribute: It is possible or even common that some attribute (we refer a data field of an ADD model as an "attribute") of a prescribed model is not useful for a specific context in which users are working. For example, "contractor" is an attribute prescribed by PAKME [27], the users do not need it if the project does not have contractors. Thus, the users need to delete that attribute.

Add an attribute: Users may need a new attribute to represent additional information, e.g., some piece of context specific information. Thus, the users need to be able to add an attribute.

Change the name of an attribute: A user may need to change the name of some attribute of a prescribed model to fit to the terms that are most familiar to the user. We found using a proper name is important in order to avoid any misunderstandings.

Change the description of an attribute: Some attributes may need to have contextualized or adapted meanings. Thus, the attributes' description needs to be changed.

Change the property of "mandatory or optional": An attribute may be mandatory in situation A but optional in situation B, and vice versa. Thus, users need to change the property of "mandatory or optional" for some attributes of a prescribed model. Capilla et al. reported similar observation in [31].

Change value range of an attribute: An attribute may have a fixed set of values. The values in the set can vary from situation to situation. For instance, the attribute "decision status" may have a value range {Approved, Obsolete, Pending, Rejected} in one situation, and a value range {Idea, Tentative, Decided, Approved, Challenged, Rejected, Obsolesced} in another situation. Thus, users need to be able to change the value range of an attribute.

Change the position of an attribute in the attribute list: A suitable order of the attributes can vary in different working situations. Thus, users need to be able to adjust the order of the attributes.

Merge two attributes: A coarse-grained ADD model may be more suitable for users' context. Thus, users may need to merge two attributes to form a coarse grained attribute.

Split an attribute: A fine-grained ADD model may be more suitable for users' context. The users may need to split an attribute into two or more attributes.

TABLE I. THE SUPPORT OF ADD MODEL CUSTOMIZATION BY EXISTING ADDM TOOLS

Scenarios\Tools	PAKME	ADDSS	Kruchten's Ontology Tool	ADkwik
Delete an attribute	N	P	N	N
Add an attribute	N	P	N	N
Change attribute name	N	N	N	N
Change attribute description	N	N	N	N
Change attribute optionality (mandatory or optional)	N	N	N	N
Change value range of an attribute	N	N	N	N
Change attribute position	N	N	N	N
Merge attributes	N	N	N	N
Split an attribute	N	N	N	N

(N = not support; P = partially support; Y = support)

It is worth to note that some of these scenarios can be grouped into more course-grained scenarios. However, to concretely characterize the needs for ADD model customization, we have decided to organize these scenarios in a very fine-grained manner. Because these scenarios characterized the needs for ADD model customization in detail, we believe that they can be used as criteria to assess in detail to what extent a tool supports ADD model customization

The results of such an assessment are summarized in Table 1. It can be seen that none of the scenarios is fully supported by any of these tools. Except ADDSS, the other three tools do not support any of the customization scenarios. ADDSS only partially supports the scenarios of "delete an attribute" and "add an attribute", because it prescribes only a few attributes to be added to the model or deleted from the model. Users cannot add any new attributes except the ones prescribed and delete any attributes except the ones prescribed. These results show that the support for ADD model customization provided by existing tools is far from satisfactory.

C. *Why Supporting ADD Model Customization Is Difficult*

Supporting highly flexible ADD model customization is difficult for several reasons. First, ADDM tools are often data-centric. Most of the functionalities of an ADDM tool are based on the underlying ADD model. If there is any change in the underlying ADD model, a large number of the functions of that tool are likely to be affected. Limited customization is easier to implement as it only affects a few functions. However, it is quite difficult to enable a user to perform arbitrary customization of the ADD model underpinning an ADDM tool.

Second, ADD model customization adds an extra step when setting up an ADDM tool. The customization step consumes efforts and requires a certain level of skill. If the customization step is not easy to perform and requires users to modify the source code or complex configuration files, practitioners are unlikely to use such tool as they are usually reluctant to invest extra efforts without immediate benefits and value.

D. *Research Objectives*

The importance of ADD model customization together with the lack of customization support by existing ADDM tools has motivated this research. The overall objective of this research is to provide a highly customizable ADDM approach, which can enable practitioners to get a specialized tool support for ADDM by customizing or defining an ADD model according to their specific needs easily. By "easily", we mean the customization step will not involve any computer programming tasks or any complex operations. A person with basic computer skills should be able to perform the customization.

III. RESEARCH METHODS

In this section, we describe the research process being followed for this project. Fig. 1 provides the visual representation of the research design and process. Multiple case studies and semi-structured interviews are used to identify ADD model customization needs, which mainly cover two aspects: 1) the extent to which ADD model customization is needed, and 2) the concrete scenarios of ADD model customization. Existing ADDM tools are evaluated using customization scenarios to determine the level of support for ADD model customization. It has been mentioned that we found that there is a strong need for ADD model customization. In response to this need, a highly customizable solution and a supporting infrastructure called Customizable Architectural Design Decision Management System (CADDMS) will be developed. This solution will be evaluated with potential end users and industrial case studies. The tasks involved, the research methods used, and the reasoning underlying the selection of research methods are described in the following subsections.

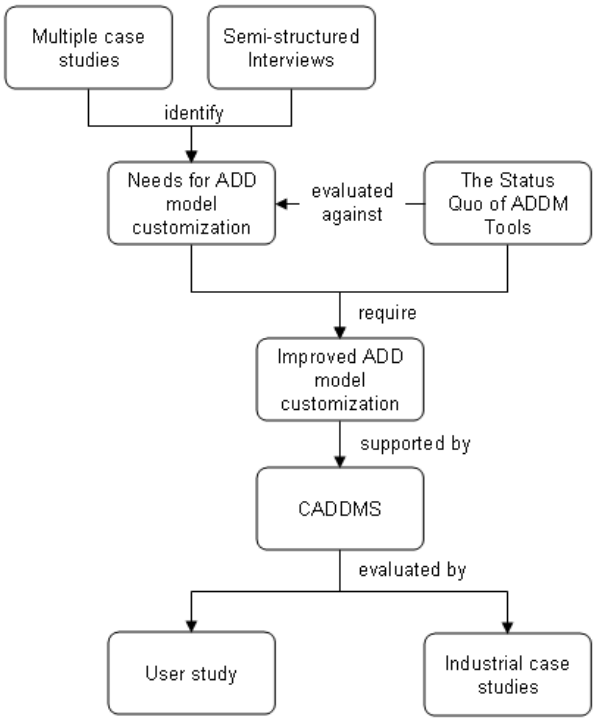

Figure 1. Overview of the doctoral research

A. Identifying ADD Model Customization Needs

Although there is a strong indication of the needs for ADD model customization, the existing research merely states, very generally, that supporting ADD model customization is important. There has been no detailed investigation of ADD model customization needs. Some important questions such as: to what extent is ADD model customization required, and: what kinds of customizations are needed, were not answered. Without answering these questions, a deep understanding about the customization needs is difficult to achieve. In particular, tool developers may have no concrete idea of what types of customization to implement, and tool evaluators may not know how to assess to what extent a tool supports ADD model customization.

To investigate the customization needs, multiple-case studies and semi-structured interviews will be conducted. Case study [32-34] research method can help gain an in-depth understanding of the factors involved in a particular context [35]. However, case studies usually involve only a single individual or just a few; thus, the studied cases may not be representative of the general population [33]. To compensate this limitation, we will validate the findings of these case studies with semi-structured interviews of practitioners.

In particular, the multiple case studies aim at answering the following research questions:

- Are there mismatches between the ADD models prescribed by the ADDM solutions and users' specific needs of the ADDM in practice?
- If there are mismatches, how and to what extent do these mismatches exist?
- If there are mismatches, what types of mismatches exist?
- What customization scenarios can be used to resolve these mismatches?

The mismatches mentioned in the research questions refer to the mismatches between practitioners' specific needs and the ADD model prescribed by the ADDM tools. For example, some practitioners need to capture decision issue (i.e., the architectural design issue the ADD is trying to address [36]) explicitly in their working situation, but a tool's prescribed ADD model does not support this. We call this a mismatch between the practitioners' needs in a particular working situation and the ADD model prescribed by that particular ADDM tool. The extent of the existence of mismatches between practitioners' required ADD model and a tool's prescribed ADD model indicates the extent of the needs for ADD model customization. The types of such mismatches have implications for the required customization scenarios.

The objective of the first research question is to confirm if there is a need for ADD model customization in real working contexts. The objective of the second research question is to investigate the extent to which ADD model customization is needed. The second research question also investigates how the mismatches happen. The objective of the third and fourth research questions is to get a list of customization scenarios that can characterize the needs for ADD model customization.

We plan to conduct multiple (two or three) case studies. For the first study, the "case" is the ADDM practices in one of our research tool development projects. Researchers were involved in the project; thus, it is a participant-observer study (which was also used by other researchers, e.g., Kitchenham et al. [37, 38]). The researchers' participation enables them to obtain detailed data and observations that are not easy, or even impossible to obtain otherwise. For the remaining studies, the "case" will be the ADDM practices in a real industrial settings.

All of the case studies follow the following procedure. First, we will elicit the requirements of an ADD model in a particular software development context. Then, we will assess different ADD models with respect to the elicited ADD model requirements. For simplicity, we call the ADD model required in a particular working context the required model, and the ADD models reported in the literature the comparison models. We will keep detailed documentation of each identified mismatch between the required model and each of the comparison models. Finally, we:

- summarize the extent to which the mismatches exist between the comparison models and the required model;
- categorize the mismatches into several types;
- derive a list of customization scenarios that can resolve those identified types of mismatches.

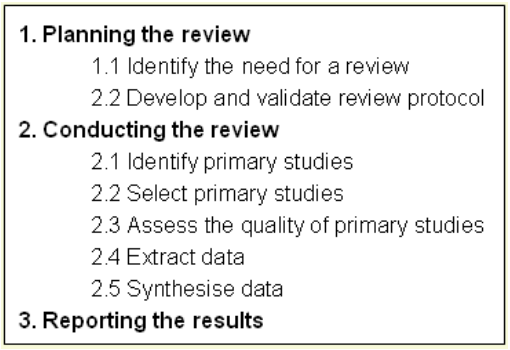

Figure 2. The SLR process

The set of comparison models will be identified using an evidence-based software engineering research method called Systematic Literature Review (SLR) [39, 40]. A SLR is conducted by following a predefined protocol, which consists of several phases and steps as shown in Fig. 2. Because a SLR is formally planned, methodically executed and grounded in facts, it has a greater level of scientific value than ordinary literature reviews [41, 42]. Another reason for using SLR in this research is that the first author already accumulated some experience in doing SLRs [43-45].

The output of the task of identifying needs for ADD model customization consists of two major elements: 1) the extent of the needs for ADD model customization, and 2) the customization scenarios that characterize the needs for ADD model customization.

B. Customizability of Existing ADDM Tools

Several ADDM tools have been developed (see Section II.B). In order to determine the extent of customization support provided by these tools, an investigative study is required.

To perform this investigative study, we need to know: (1) the criteria to be used for the investigation, and (2) the existing ADDM tools. For the evaluation criteria, as described in Section II.B, none of existing evaluation frameworks provides detailed criteria for assessing to what extent a tool supports ADD model customization. Thus, we have decided to use the ADD model customization scenarios (identified via activities described in Section III.A) as criteria in this investigation. For the existing ADDM tools, we have decided to identify them via a SLR [39, 40].

C. Developing a Customizable ADDM System

We have adopted an iterative and incremental development process [46] for developing CADDMS, the infrastructure for supporting our proposed solution (see Section V for more detail of the proposed solution). The iterative and incremental process is expected to allow us to get early feedback on our proposed solution from potential users by showing them the initial version of the system created during the initialization step. Using such a process, the development of CADDMS will also be continuously informed and guided by the feedback from potential users and our experience gained in previous iterations. For example, a major goal of our solution is to enable users to get a specialized tool support for their specific ADDM needs without requiring programming tasks or any complex operations. That means CADDMS should be easily usable by people with basic computer skills. With an iterative and incremental development process, we can first implement the ADD model editor and perform a user study to test if this goal can be achieved.

We will implement CADDMS as a web-based application. So a user only needs a web browser to access the system and the captured ADDs from anywhere. This will help to reduce the participation barrier and foster distributed collaboration. Advanced web technologies such as AJAX [47] and Rich Internet Application frameworks will be used for achieving good usability.

D. Evaluating the Proposed Solution

Customizability is the principal criterion for evaluating CADDMS, which is expected to satisfactorily fulfill all customization scenarios outlined in Section II.B. Further evaluation will involve potential users and industrial case studies.

1) End user study: The end user study is inspired by feedback from industrial practitioners. According to the feedback, a customizable approach usually increases the initial effort. The more flexible an approach is, the higher the effort and skills usually required for customization. The effort required for performing the customization, and the usability and learnability [1] [14] of CADDMS will determine if we can achieve the goal that users are able to easily create a customized ADDM tool. Thus, the aspects of needed effort for customization, learnability and usability of CADDMS, are important to evaluate. We planned a study to evaluate these aspects. In particular, the main objectives of the study are as follows:

- determine the efforts required for obtaining a personalized ADDM tool support;
- evaluate the usability of CADDMS;
- evaluate the learnability of CADDMS.

According to the study protocol, the participants are asked to build a personalized tool support for managing ADDs. Each of the participants is provided with a short (around 15 minutes) introduction to the study and CADDMS. The introduction to the study is designed to set up a concrete context for the task to be performed (e.g., the participant is a lead architect in a development team, she/he is going to ask the members of her/his team to manage ADDs following the ADD model). The introduction to CADDMS is performed by the researcher, who explains how to use the tool. After the introductory session, the participants are asked to perform the task using CADDMS and Microsoft Word. The choice of Microsoft Word has been made for two reasons: (1) none of the existing ADDM tools provides similar level of

[1] The capability of a software product to enable the user to learn how to use it

customizability, as described in previous sections; (2) Microsoft Word is the most commonly used productivity tool for drawing templates for architecture documentation in industry. Half of the participants are randomly asked to use CADDMS first and half of them to use the Microsoft Word first.

The researchers observe the participants' activities and note down the time taken by each participant for performing the assigned task. After finishing the assigned task, each participant is interviewed about his/her experience of performing the assigned task. The interview questions include:

1. Have you learnt how to use CADDMS?
2. Is CADDMS easy to use?
3. Is the automatic generated template satisfactory?
4. Do you have any suggestions to improve CADDMS?
5. Which tool would you prefer to use for managing ADD in your team and why?

The researchers take extensive notes, which are codified for analysis. The time taken for the assigned tasks is analyzed using descriptive statistics.

2) Industrial case studies: We also plan to conduct industrial case studies to assess the effectiveness of CADDMS. These case studies are also expected to help identify the requirements for further improvement of the proposed solution.

IV. PROGRESS TO DATE

We have conducted the participant-observer case study to investigate the needs for ADD model customization. The initial findings have revealed the possibility of the existence of significant mismatches between the prescriptive ADD models used in ADDM tool and practitioners' specific needs. These findings suggest that supporting ADD model customization is essential for an ADDM tool to satisfy the specific needs of practitioners in their specific working situations. These findings have also identified a list of customization scenarios (see Section II.B), which provides a detailed characterization of ADD model customization needs. These scenarios can be used as criteria for assessing an ADDM tool's customization support, or as requirements for developing a customizable ADDM tool. The result of this study has been documented in [11].

We have primarily evaluated existing ADDM tools with respect to their support for ADD model customization. The result of this study has been summarized in Section II.B.

We have been developing an infrastructure (i.e., CADDMS) to support the proposed solution. We will describe more detail of the proposed solution and the supporting infrastructure in the next section.

We have conducted an end user study to determine the required customization effort, usability and learnability of the initial prototype of CADDMS. The results from this study are quite encouraging. The effort required for building a customized tool with CADDMS is often less than building a template with Microsoft Word. On average the participants took 16:44 minutes for CADDMS and 21:10 minutes for Microsoft Word in order to perform the assigned task. The learnability and usability of CADDMS were also confirmed by the positive comments from the participants. All of the participants responded that if he/she were the team lead, he/she would prefer using the CADDMS to manage ADDs. The prototype of the tool and its initial evaluation have been reported in [48].

V. PROPOSED SOLUTION

A. An Overview of the Proposed Solution

Since an ADD model serves as the base of an ADDM tool, we propose an ADD model-centered customizable solution that can help practitioners to get a specialized tool for their specific needs by configuring the ADD model.

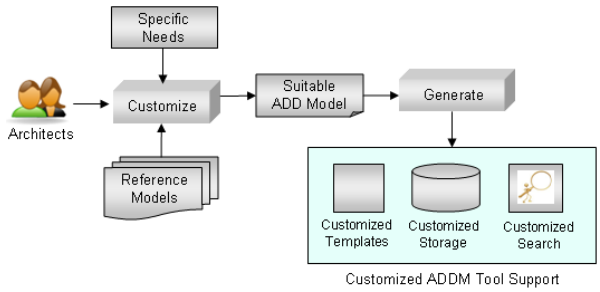

Figure 3. Overview of the proposed solution

Fig. 3 presents an overview of the solution. The solution provides a list of ADD models that practitioners can customize to get the ADD models that fit their specific needs. For simplicity, we call these ADD models the reference models. These reference models mainly come from the literature on ADDM. From this list of reference models, architects select an ADD model that is mostly close to their specific needs. The architects customize the ADD model according to their specific needs with a WYSIWYG (What You See Is What You Get) style ADD model editor. The customization will result in an ADD model that suits the architects' specific needs. It is worth mentioning that if none of the reference models is close to their needs, the solution allows practitioners to create an ADD model from scratch. Based on the suitable ADD model, a customized ADDM tool support will be automatically generated. The customized tool support contains, among other functionalities, customized ADD templates, a customized storage, and a customized search facility.

The customized templates are presented as web-based forms, which represent the customized ADD model from two aspects: a data structure prescribed by the ADD model and the constraints placed on the attributes. For example, a user can specify that for attribute A, a value should always be given and the value should be numerical within a certain range. These constraints are enforced when a user enters the design decisions. Moreover, a user can also see the semantics, which are intended by the creator of the

ADD model, of each attribute when providing values for different attributes of a design decision.

The customized storage is a structured repository for storing the information captured by the customized template. This storage for ADDs can make the search and other data manipulations (e.g., analysis, visualization, and codification of ADDs) easier than using productivity applications (e.g., Microsoft Word or Excel) or traditional wiki pages [49]. The machine processing of ADDs captured in Word documents, Excel sheets, or wiki pages is not easy [49].

The customized search enables a user to search their captured ADDs. Based on a customized ADD model definition, the system automatically generates a search facility that enables a user to specify his/her search criteria in the granularity of each attribute of the model. For example, a user can specify that "list all ADDs with value V1 for attribute A1, with value V2 for attribute A2, and with value V3 for attribute A3". The system fetches and shows only those ADDs that satisfy the criteria. We assert that compared with keywords-based search, the customized search can be more fine-grained and accurate. Since, a keywords-based search may be sufficient for some situations, the system supports keywords-based search as well.

Having been customized and used, over time, practitioners' needs for ADDM many change due to changes of working situations. Thus, practitioners may need to change their ADD models over time. We call this ADD model evolution issue. The major challenge of ADD model evolution is to achieve compatibility between the evolved ADD model and previously captured ADDs. The solution should also provide support for practitioners to adjust their ADD models to reflect their changing needs.

B. A Supporting Infrastructure

To realize the above described solution, a supporting infrastructure, called CADDMS, is being developed. There are two major design goals for CADDMS. First, the system should provide very flexible customizability. The system should allow practitioners to arbitrarily customize the reference models or invent new ADD models to suit their specific ADDM needs. Second, CADDMS should be easy to use. The system should enable users with basic computer skills to generate a specialized tool for their specific ADDM needs without requiring any programming tasks or any complex operations.

Only supporting ADD model customization cannot make a usable ADDM tool. CADDMS should also support several other features. We briefly describe some important features here.

Subscription and notification: Users can subscribe to the types of ADDs they are interested in (they can also use search strings to specify their interests in a fine-grained manner), and CADDMS notifies them whenever new ADDs are entered.

Sharing expertise and experience: CADDMS maintains each user's profile based upon which he/she could be searched for sharing knowledge, especially for the knowledge that is hard to articulate. A user can configure his/her profile not to be visible to members outside of her/his team.

ADD based discussion: Users can express their opinion on an ADD by posting comments on and rating a particular ADD.

Meta-data-enriched document management: The platform enables users to store documents with sufficient meta-data attached, which can be used for searching the documents.

Flexible classification of the content: The system enables users to make free classification of the content by using tags.

Scope-of-interest based views: the platform organizes ADDs based on the scope-of-interest, which can be a project, a department, or even the whole community. It is hierarchically organized. An inner scope-of-interest can be a member of an encompassing scope-of-interest. The members in an inner scope-of-interest can share information proprietary to them, they can also share some more general knowledge with a broader community consisting of the members of the encompassing scope-of-interest. A project can share some proprietary information within them. They can also share some more general knowledge with other teams in the same department. The knowledge in the innermost scope-of-interest will be dominant in the view that shows to the members of the scope-of-interest. For example, the system shows the knowledge pertaining to the project as the dominant part of the view that shows to the members of the project.

Role-based views: Within each scope-of-interest, different roles may have different views. For example, the views for the managers contain more summary information in the form of different types of reports.

VI. EXPECTED CONTRIBUTIONS AND FUTURE WORK

Increasing complexity in software systems requires that participants fully understand the key Architectural Design Decisions (ADD) and the reasoning underlying them. Architectural Design Decision Management (ADDM) tools can aid in this respect by facilitating the capturing, storing, managing, and sharing of ADDs. Several ADDM tools have been developed. All of these tools prescribe fixed data models for architects to follow without sufficient support for customization. This research project aims to investigate the specialized needs of practitioners for ADD models, and develop and empirically assess a customizable solution to meet the specialized needs. The expected contributions of this research are:

Identification of the ADD model customization needs, which will also identify the key customization scenarios. Such scenarios can be used as criteria for assessing the extent to which an ADDM tool supports ADD model customization, or as requirements for developing a customizable ADDM tool.

Assessment of the existing ADDM tools with respect to the customization scenarios. The findings can be useful input to practitioners when they select ADDM tools. The

findings can also identify research gaps (e.g., lack of customization support in existing ADDM tools) for ADDM researchers.

A highly customizable solution and accompanied infrastructure for practitioners to gain a specialized ADDM tool support without requiring programming or any other complex operations.

The future work includes:

- Conduct the industrial case studies and semi-structured interviews to further validate and refine the needs for ADD model customization identified using the participant-observer case study;
- Finish the development of CADDMS;
- Evaluate CADDMS with industrial case studies;
- Further improve CADDMS according to the feedbacks from the industrial case studies.

ACKNOWLEDGMENT

This work is partly supported by Science Foundation Ireland under the grant no. 03/CE2/I303_1. We would like to thank the participants of our empirical studies. Without their help, we could not conduct this research. We would also like to thank the anonymous reviewers for their comments to improve this work. The first author would like to thank Klaas-Jan Stol for the proof reading of this paper.

REFERENCES

[1] M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*: Prentice Hall, 1996.

[2] M. Shaw and P. Clements, The Golden Age of Software Architecture, *IEEE Softw.,* vol. 23, pp. 31-39, 2006.

[3] J. Bosch, "Software Architecture: The Next Step," in *Software Architecture*: Springer, 2004, pp. 194-199.

[4] R. N. Taylor, N. Medvidovic, and E. Dashofy, *Software Architecture: Foundations, Theory, and Practice*: Wiley, 2009.

[5] A. Jansen and J. Bosch, "Software Architecture as a Set of Architectural Design Decisions," in *5th Working IEEE/IFIP Conference on Software Architecture*, 2005, pp. 109-120.

[6] P. Kruchten, "An Ontology of Architectural Design Decisions in Software-Intensive Systems " in *2nd Groningen Workshop on Software Variability*, 2004.

[7] M. A. Babar and P. Lago, Design decisions and design rationale in software architecture, *Journal of Systems and Software,* vol. 82, pp. 1195-1197, 2009.

[8] D. E. Perry and A. L. Wolf, Foundations for the study of software architecture, *SIGSOFT Softw. Eng. Notes,* vol. 17, pp. 40-52, 1992.

[9] J. E. Burge, J. M. Carroll, R. McCall, and I. Mistrik, *Rationale-Based Software Engineering*: Springer, 2008.

[10] A. Tang, M. A. Babar, I. Gorton, and J. Han, A survey of architecture design rationale, *Journal of Systems and Software,* vol. 79, pp. 1792-1804, 2006.

[11] L. Chen and M. A. Babar, Architectural Design Decision Model Needs Customization, Lero at UL 2009. http://193.1.97.13/wikisac/images/Tr_add.pdf

[12] J. E. Burge, J. M. Carroll, R. McCall, and I. Mistrik, "An Architectural Framework," in *Rationale-Based Software Engineering*: Springer Berlin Heidelberg, 2008, pp. 241-254.

[13] L. Lee and P. Kruchten, "Customizing the capture of software architectural design decisions," in *Canadian Conference on Electrical and Computer Engineering*, 2008, pp. 93-98.

[14] ISO/IEC TR 9126: Software engineering -Product quality, 19-12-2000.

[15] E. Poort, A. Pramono, M. Perdeck, V. Clerc, and H. van Vliet, "Successful Architectural Knowledge Sharing: Beware of Emotions," in *Architectures for Adaptive Software Systems*, 2009, pp. 130-145.

[16] J. Conklin and M. L. Begeman, gIBIS: a hypertext tool for exploratory policy discussion, *ACM Trans. Inf. Syst.,* vol. 6, pp. 303-331, 1988.

[17] P. Lago, "Establishing and Managing Knowledge Sharing Networks," in *Software Architecture Knowledge Management*, 2009, pp. 113-131.

[18] 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,* vol. In Press, Accepted Manuscript.

[19] 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.

[20] S. L. Kogan and M. J. Muller, Ethnographic study of collaborative knowledge work, *IBM Syst. J.,* vol. 45, pp. 759-771, 2006.

[21] R. Guindon, Designing the Design Process: Exploiting Opportunistic Thoughts, *Human-Computer Interaction,* vol. 5, pp. 305 - 344, 1990.

[22] A. Agostini, G. d. Michelis, M. A. Grasso, and S. Patriarca, "Reengineering a business process with an innovative workflow management system: a case study," in *Proceedings of the conference on Organizational computing systems* Milpitas, California, United States: ACM, 1993.

[23] M.-M. Raul, W. Terry, F. Rodrigo, and F. Fernando, "The action workflow approach to workflow management technology," in *Proceedings of the 1992 ACM conference on Computer-supported cooperative work* Toronto, Ontario, Canada: ACM, 1992.

[24] T. Winograd and F. Flores, *Understanding Computers and Cognition: A New Foundation for Design*: Addison Wesley, 1987.

[25] M. Hammer and J. Champy, *Reengineering the corporation: A manifesto for business revolution*: HarperBusiness, 1994.

[26] D. Paul, "Process descriptions as organisational accounting devices: the dual use of workflow technologies," in *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work* Boulder, Colorado, USA: ACM, 2001.

[27] M. A. Babar and I. Gorton, "A Tool for Managing Software Architecture Knowledge," in *2nd Workshop on Sharing and Reusing Architectural Knowledge*, 2007, pp. 11-11.

[28] R. Capilla, F. Nava, S. Pérez, and J. C. Dueñas, A web-based tool for managing architectural design decisions, *SIGSOFT Softw. Eng. Notes,* vol. 31, p. 4, 2006.

[29] M. Host and P. Runeson, "Checklists for Software Engineering Case Study Research," in *First International Symposium on Empirical Software Engineering and Measurement, 2007. ESEM 2007.* , 2007, pp. 479-481.

[30] N. Schuster and O. Zimmermann, Architectural Decision Knowledge Wiki, http://www.alphaworks.ibm.com/tech/adkwik

[31] R. Capilla, F. Nava, and J. C. Dueñas, "Modeling and Documenting the Evolution of Architectural Design Decisions," in *SHARK/ADI: Workshop on Sharing and Reusing Architectural Knowledge / Architecture, Rationale, and Design Intent*, 2007.

[32] B. Kitchenham, L. Pickard, and S. L. Pfleeger, Case studies for method and tool evaluation, *Software, IEEE,* vol. 12, pp. 52-62, 1995.

[33] R. K. Yin, *Case Study Research: Design and Methods*, 3rd ed.: Sage Publications, Inc, 2002.

[34] P. Runeson and M. Höst, Guidelines for conducting and reporting case study research in software engineering, *Empirical Software Engineering,* vol. 14, pp. 131-164, 2009.

[35] H. Mintzberg, An Emerging Strategy of "Direct" Research, *Administrative Science Quarterly,* vol. 24, pp. 582-589, 1979.

[36] J. Tyree and A. Akerman, Architecture decisions: demystifying architecture, *Software, IEEE,* vol. 22, pp. 19-27, 2005.

[37] B. A. Kitchenham, O. P. Brereton, D. Budgen, and Z. Li, "An Evaluation of Quality Checklist Proposals - A participant-observer case study," in *13th International Conference on Evaluation and Assessment in Software Engineering*, Durham University, UK, 2009.

[38] B. Kitchenham, P. Brereton, M. Turner, M. Niazi, S. Linkman, R. Pretorius, and D. Budgen, "The Impact of Limited Search Procedures for Systematic Literature Reviews - An Observer-Participant Case Study," in *International Symposium on Empirical Software Engineering and Measurement*, Florida, USA, 2009.

[39] B. Kitchenham, T. Dyba, and M. Jorgensen, "Evidence-Based Software Engineering," in *ICSE*: IEEE, 2004.

[40] Barbara Kitchenham and S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, School of Computer Science and Mathematics Keele University and Department of Computer Science University of Durham, Keele 2007.

[41] T. Dyba, T. Dingsoyr, and G. K. Hanssen, "Applying Systematic Reviews to Diverse Study Types: An Experience Report," in *ESEM*, 2007, pp. 225-234.

[42] M. Staples and M. Niazi, Experiences using systematic review guidelines, *Journal of Systems and Software,* vol. 80, pp. 1425-1437, 2007.

[43] L. Chen and M. A. Babar, "A Survey of Scalability Aspects of Variability Modeling Approaches," in *Workshop on Scalable Modeling Techniques for Software Product Lines at SPLC'09* San Francisco, CA, USA, 2009.

[44] L. Chen, M. A. Babar, and N. Ali, "Variability Management in Software Product Lines: A Systematic Review," in *the 13th International Software Product Line Conference*, San Francisco, CA, USA, 2009.

[45] L. Chen, M. A. Babar, and C. Cawley, "A Status Report on the Evaluation of Variability Management Approaches," in *13th International Conference on Evaluation and Assessment in Software Engineering*, UK, 2009.

[46] C. Larman and V. R. Basili, Iterative and Incremental Developments: A Brief History, *Computer,* vol. 36, pp. 47-56, 2003.

[47] J. J. Garrett, Ajax: A New Approach to Web Applications, http://www.adaptivepath.com/ideas/essays/archives/000385.php

[48] L. Chen, M. A. Babar, and H. Liang, "Model-Centered Customizable Architectural Design Decisions Management," in *21st Australian Software Engineering Conference*, Auckland, New Zealand, 2010.

[49] R. Farenhorst and H. v. Vliet, "Experiences with a Wiki to Support Architectural Knowledge Sharing," in *Wiki4SE at WikiSym'08*, Porto, Portugal, 2008.