Facilitating Collaborative Decision Making With the Software Architecture Video Wall

Jan Martijn E. M. van der Werf, Rico de Feijter, Floris Bex, and Sjaak Brinkkemper
Department of Information and Computing Sciences
Utrecht University
{j.m.e.m.vanderwerf, r.defeijter, f.j.bex, s.brinkkemper}@uu.nl

***Abstract*—Although capturing and documenting the design making process in software architecture is an important task, few tools exist to support the architect in this task. Often, such decisions are made during discussions with other stakeholders, but typically these remain implicit.**

We envision the Software Architecture Video Wall as a collaborative decision making tool for the software architect. The video wall serves as a whiteboard for discussions that automatically records the screenplay. Combined with the audio recordings of the discussion participants, a rich data set is obtained that may serve as input to track design decisions, and argumentation mining, to generate architecture rationale documentation. The video wall serves different uses as explained in several possible scenarios. However, as shown in the paper, much research is still required to realize this vision.

I. Introduction

One of the main tasks of the software architect is to design the system and to develop a project strategy [1]. By making architectural design decisions, the architect constructs a software architecture. Typically, organisations do not have a single architect who makes all principle design decisions [1]. Instead, the architect closely collaborates with other stakeholders, such as software product managers, requirements engineers and software developers [2].

A recent development in software architecture is the advent of systems of systems: more systems are connected to jointly deliver their functionality. For example, dozens of small organizations deliver together the software of a single car [3]. In the business administration domain, similar trends can be observed, such as the Dutch health care system between health providers, municipalities and the central administration. Not communicating all principle decisions to all collaborating organizations, may seriously impact the system's functionality and characteristics. Consequently, both the documentation and communication of design decisions throughout the software development process becomes critical.

The twin peaks model [4], and its extension to software construction [5] (Figure 1) visualizes the collaboration between the different stakeholders during software development. The collaboration between architect, requirement engineers and software product managers result in more detailed requirements and architecture. Similarly, the actual construction of the software system results in changes in the architecture, and consecutively may influence the requirements. In the whole process, architects mainly use discussion models [2] to communicate with the different stakeholders. Such models are typically informal drawings on a whiteboard with some lines and boxes, and collaboratively altered during the discussion with the stakeholders around the whiteboard. The end result may stay for a while on the whiteboard, being photographed and filed somewhere, or not being documented at all [2].

In the past decades, many approaches have been suggested to support design reasoning in design. For example, issue-driven design methods such as gIBIS and QOC have been proposed to guide designers to focus on design issues and criteria to judge design decisions [6]. Much of this research focuses on providing software tools that can be used to make the issues, options and arguments uttered during the design phase explicit. One of the problems of these approaches is the cognitive overload that results from having to learn and use such tools at the same time as having to discuss and think about complex software designs.

During design sessions, many different design decisions are taken. However, most of these decisions are left implicit [7]. In research, different approaches have been developed to make such decisions expicit. For example, [8] adds an observer to the architecting team who in regular intervals intervenes the discussion by prompting rationale questions, to trigger reflection. Another approach has been to use a card game during the design discussion [9]. Although these approaches result in better design reasoning [10], documenting these in Architecture Documentation remains an open problem.

As observed in earlier work [2], many discussions are centred around a whiteboard. In this paper, we envision an approach that intertwines design rationale with the actual

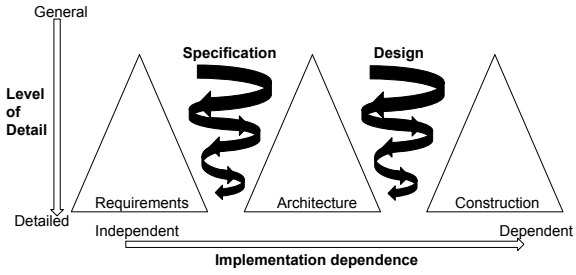

Fig. 1. Three peaks model: collaboration between the different roles in software development [5].

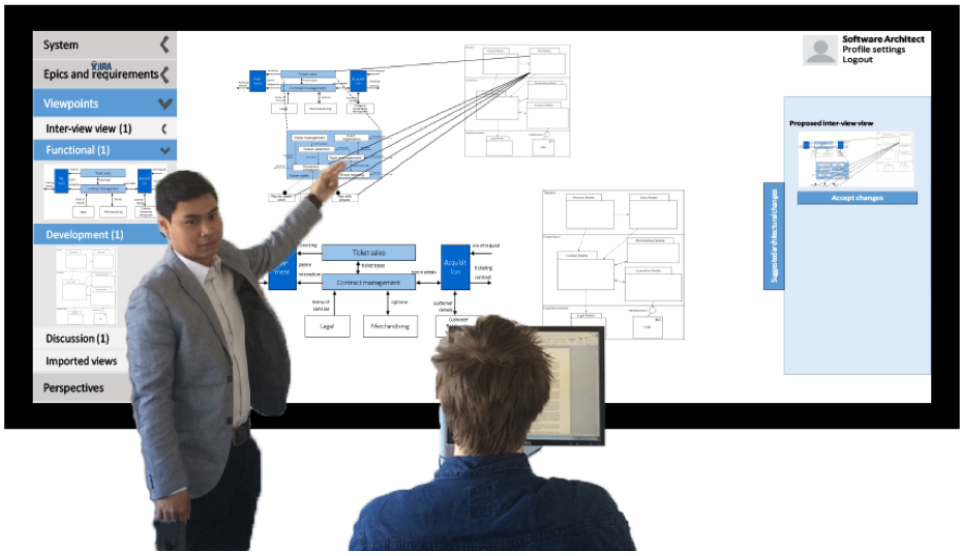

Fig. 2. The video wall: a large multi-touch screen that serves as a whiteboard for discussions, capturing the screen play during the discussion sessions.

design by capturing these discussions at the whiteboard and analyzing these automatically based on e.g. text and argumentation mining. In Section II, we elaborate on our vision. Next, Section III present several scenarios on how practitioners could use the approach. Section IV concludes with an outlook to further research required to materialize our vision.

II. Architecture Decision Support

Models are frequently used to discuss different options among stakeholders. A typical discussion session goes as follows. One of the participants stands in front of the whiteboard and sketches the model as he or she thinks the option should look like. The other participants in the discussion watch how the sketch is created, and start analyzing and discussing the option. During the discussion, others might stand up, and change the sketch, based on the ongoing discussion. After several rounds, the participants agree on the sketch models, and the session is closed. Sometimes, one of the participants, typically the architect, makes a photo and files it, and that is all that is documented. In a next meeting, the process starts again from the beginning, typically not using the photograph of the earlier created models, thus one of the participants redraws the sketch, and the discussion starts all over. As not all participants might be aware of the how and why of the current model, the discussion starts all over, not touching the actual topics that should be discussed.

Such scenarios are not uncommon in software producing organizations [2]. This is why we envision the Software Architecture Video Wall that supports capturing and documenting design sessions.

A. The Software Architecture Video Wall

The Software Architecture Video Wall, (see Figure 2) is a large multi-touch screen that essentially serves as a large whiteboard. The wall is split into two panes: a side bar with the discussion models as small thumbnails, and a main canvas. During the discussion, an empty canvas can be used to create a new discussion model, which can be stored and versioned by the participants. Additionally, a thumbnail of a discussion model on the side bar can be swiped into the current canvas, and serve as a basis for discussion. In this way, the canvas serves as an ordinary whiteboard during the discussion: elements can be drawn, coloured, highlighted, or text can be written on it.

Typically, a discussion requires several models, e.g. discussing whether to create a desktop application or to use a client-server pattern involves not only a functional viewpoint [5], but also the deployment viewpoint needs to be considered, to decide which functionality goes where.

Such inter-view relations and discussions are essential to capture principle design decisions. In this way, the video wall enables the discussion on consequences in different views by visualizing different discussion models at the same time, and allows the participants to make traceability explicit, which again improves the reasoning and documentation of the software architecture.

B. Documenting Design Sessions

A simple way to record a discourse session is by capturing the discussion itself, e.g. by equipping each participant with an audio recording device. In the transcripts of the card game research [9], one typically finds remarks like “I do not agree with this element”, while the participant is pointing at the whiteboard. Unfortunately, audio recordings do not capture what the participant is pointing at. By capturing the video wall interaction during these discussions as well, one can analyze not only what has been said, but also what was visible on the wall, how it was altered during that discussion, and by whom.

Ideally, the combined video and audio recordings are analyzed automatically using text and argumentation mining

tools [11], to automatically extract architecture documentation containing the essential drawings from the whiteboard, the traceability relations as discussed, the main rationale, and which decisions have been made. In this way, the participants can look back on why certain models have been created, or novice designers can use the recordings to study the design discourse of the particular models.

III. Possible Uses

One of the main advantages of the Software Architecture Video Wall is that models and discussions can be captured at any moment in time, similar to the use of the whiteboard in current practices. In this section, we sketch several possible uses of the Video Wall. First we sketch a scenario of a classical group meeting between a software architect and a software product manager. As a second scenario, which sketch an informal discussion at the coffee corner. Last, we show how the video wall can support the architect in capturing rationale with more advanced analyses, e.g. when running simulations.

A. Capturing a Group Meeting: Documenting decisions

A first scenario is the classical group discussion in a meeting. Consider a software product manager (SPM) and a software architect (SA) that together want to discuss the features to be added for the next release. The SPM has a set of new user stories that need to be integrated into the new release, and shows the list of user stories on the wall. The SA swipes in a functional view, and starts dividing the stories over the different model elements (see Figure 2). In this way, the SA starts to create a traceability relation between the stories and the functional view. There is a choice for story U: it could be mapped both on functional element A as well as B. The SA maps it to B and explains why ($R1$). The SPM disagrees and proposes to map story U to functional element A, and gives another argument ($R2$). As a next step, the architect swipes a technical view that realizes A and B. Based on the traceability relations between the functional and technical view ($R3$), the SPM and SA agree to map story U to functional element B.

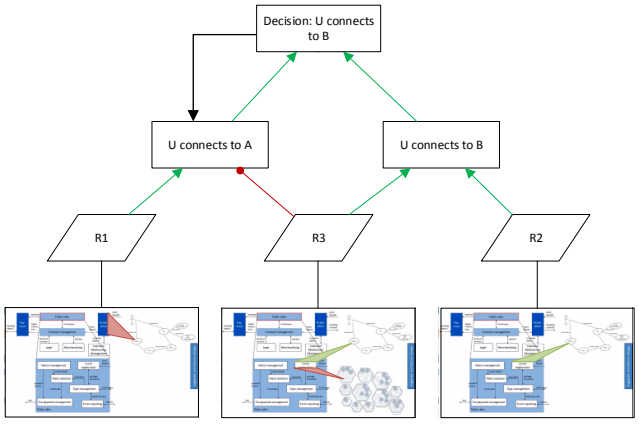

Fig. 3. Decision Model including reasons for and against design options, including the screen capture of the different arguments.

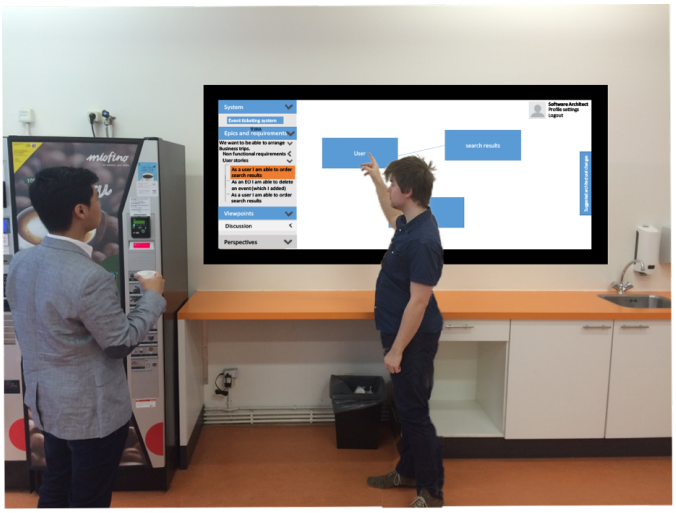

Fig. 4. A developer explains an informal model to the software architect, who enjoys a cup of coffee.

In the end, the video wall analyzes the audio and video recordings. Based on the analysis, it documents the final decision: from the choice of the two options to link U to either A or B, there was argument $R1$ by the SA that supported the former option, whereas arguments $R2$ of the SPM and $R3$ of the SA supported the latter option, resulting in the decision tree as shown in Figure 3.

B. At the Coffee Corner: Collecting Design Options

Many discussions take place in an informal setting, such as at the coffee corner. In many organisations, the coffee corner is a perfect place where many employees meet unexpectedly and informally discuss work related matters [12]. While drinking a cup of coffee with colleagues, ideas might pop up, or technical issues one is facing at the moment are discussed. Thus, the coffee corner is an ideal place to collect different design options. For example, a developer could illustrate his idea to the software architect, by creating a new view (Figure 4), or by adapting an existing view. The wall stores the adapted view, and tags it as a design option, so that the architect can study the option once he is back at his desk.

In addition, a part of capturing collaborative decision making could also encompass an audio recording mechanism, which allows the discussion to be recorded for later reference. For instance, the data recorded during one of the informal conversations that took place at a coffee corner could be requested later on to support decision making in a formal group meeting.

C. In Depth Analysis

The video wall should not only assist in building software architecture documentation. Assessing an architecture to validate software quality attribute satisfaction is an equally important task. For example, based on the workload of the actual system (measured from software operation data [13]),

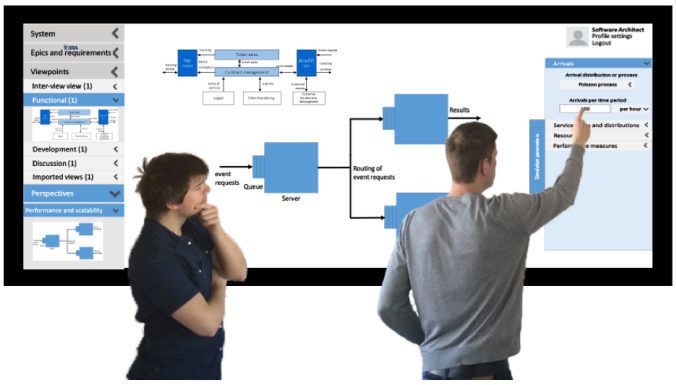

Fig. 5. Adapting the performance characteristics of a queuing model connected to a functional architecture model.

bottlenecks can be highlighted in the architecture, or a new deployment can be suggested based on analyzing the architecture and the system workload [13]. This results in new views, that are added to the wall. The architect can then visually inspect the suggestions, to accept or reject the changes.

Another example is depicted in Figure 5. In this example, a software architect and a developer jointly model a queueing network based on a Functional Architecture Model (FAM) [14]. Together they set the parameters and check the traceability. The outcomes of the queuing network analysis might affect the other, connected, views. It is thus important to also capture the actions carried out in the simulation (e.g. setting the parameters etc.), apart from the discussion between the two software architects that follows from the output of the simulation, in order to know what actions from the simulation led to the adoption of the contents of the FAM.

IV. Outlook

Although capturing and documenting the design making process in software architecture is an important task, few tools exist to support the architect in this task. In this paper, we presented our vision to capture collaborative decision making through the Software Architecture Video Wall. The video wall captures the models created during discussion sessions, and records the process of how these models were established. Capturing the discussion directly supports the architect in many ways. For example, if the architecture needs to be altered, the recordings can be replayed to recall the rationale of the current model, or new architects can study the replay to understand the current situation.

Future research to realize this vision requires many steps. For example, text and argumentation mining on the data collected by the video wall, i.e., analyzing both the screenplay and the audio recordings of the different participants, is not yet capable of extracting a decision model as depicted in Figure 3. Although capturing the screenplay helps in annotating and understanding the audio recordings, further research is required to be able to analyze the screenplay of a design session.

One way to guide argumentation mining would be to create an interactive version of the card game [9]. By giving each architect a device with the card game installed, the playing of a card can be connected to the current screenplay. Both the screenplay and the audio files are automatically annotated with the card play, which serves as extra input for argumentation mining techniques.

Another direction would be to add artificial intelligence – or more specifically, modules for computational argumentation [15] – to the video wall, that automatically plays the role of the critical observer (cf. [8]): if the system cannot create a decision tree, or misses options or arguments, the system can automatically pose reflective questions in the discussion to ensure that all decisions have a proper rationale.

We strongly believe that advances in this direction will be a corner stone for continuous architecture: the architect creates models, while the video wall keeps the architecture documentation, including its rationale, up-to-date.

References

[1] R. Taylor, N. Medvidovic, and E. Dashofy, *Software Architecture: Foundations, Theory, and Practice*. John Wiley & Sons, 2010.

[2] G. Lucassen, J. M. E. M. van der Werf, and S. Brinkkemper, "Alignment of software product management and software architecture with discussion models," in *8th IEEE International Workshop on Software Product Management, IWSPM 2014, Karlskrona, Sweden, August 26, 2014*. IEEE Computer Society, 2014, pp. 21–30.

[3] B. Boss, C. Tischer, S. Krishnan, A. Nutakki, and V. Gopinath, "Setting up architectural sw health builds in a new product line generation," in *10th European Conference on Software Architecture Workshops*, ser. ECSAW '16. ACM, 2016, pp. 16:1–16:7.

[4] J. Cleland-Huang, R. S. Hanmer, S. Supakkul, and M. Mirakhorli, "The twin peaks of requirements and architecture," *IEEE Software*, vol. 30, no. 2, pp. 24–29, 2013.

[5] N. Rozanski and E. Woods, *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, 2011.

[6] S. J. B. Shum, A. M. Selvin, M. Sierhuis, J. Conklin, C. B. Haley, and B. Nuseibeh, "Hypermedia support for argumentation-based rationale: 15 years on from gibis and qoc," in *Rationale Management in Software Engineering*. Springer, 2006, pp. 111–132.

[7] P. Lago and H. Vliet, "Explicit assumptions enrich architectural models," in *27th International Conference on Software Engineering, 2005. ICSE 05*. ACM, 2005, pp. 206–214.

[8] M. Razavian, A. Tang, R. Capilla, and P. Lago, "In two minds: How reflections influence software design thinking," *J. Softw. Evol. Process*, vol. 6, pp. 394–426, 2016.

[9] C. Schriek, J. M. E. M. van der Werf, A. Tang, and F. Bex, "Software architecture design reasoning: A card game to help novice designers," in *10th European Conference on Software Architecture, ECSA 2016*, ser. LNCS, vol. 9839. Springer, 2016, pp. 22–38.

[10] A. Tang, M. Tran, J. Han, and H. Vliet, "Design reasoning improves software design quality," in *Quality of Software Architectures*, ser. LNCS, vol. 5581. Springer, 2008, pp. 28–42.

[11] M. Lippi and P. Torroni, "Argument mining: A machine learning perspective," in *International Workshop on Theorie and Applications of Formal Argumentation*. Springer, 2015, pp. 163–176.

[12] Y. Taminiau, W. Smit, and A. de Lange, "Innovation in management consulting firms through informal knowledge sharing," *Journal of Knowledge Management*, vol. 13, no. 1, pp. 42–55, 2009.

[13] S. Klock, J. M. E. M. van der Werf, J. P. Guelen, and S. Jansen, "Workload-based clustering of coherent feature sets in microservice architectures," in *International Conference on Software Architecture*. IEEE, 2017, accepted.

[14] S. Brinkkemper and S. Pachidi, "Functional architecture modeling for the software product industry," in *ECSA 2010*, ser. LNCS, vol. 6285. Springer, 2010, pp. 198 – 213.

[15] M. Van Zee, F. Bex, and S. Ghanavati, "Rationalization of goal models in grl using formal argumentation," in *23rd International Requirements Engineering Conference (RE)*. IEEE, 2015, pp. 220–225.