Applying the IEEE 1471-2000 Recommended Practice to a Software Integration Project

Rikard Land

Department of Computer Science and Engineering
Mälardalen University
PO Box 883, SE-721 23 Västerås, Sweden
+46 21 10 70 35

rikard.land@mdh.se
http://www.idt.mdh.se/~rld

Abstract

This paper describes an application of the IEEE Standard 1471-2000, "Recommended practice for architectural description of software-intensive system" in a software integration project. The recommended practice was introduced in a project without affecting its schedule and adding very little extra costs, but still providing benefits. Due to this "lightweight" introduction it is dubious whether it will be continually used within the organization.

Keywords
Architectural Description, IEEE 1471-2000, Recommended Practice, Software Architecture, Software Integration.

1. Introduction

The software field is developing rapidly. New areas of practice and research are emerging with an ever-increasing speed. Each one claims to be crucial to the success of software: web technologies, security, software processes, or, as in our case, software architecture. There is clearly a difficult tradeoff to solve for companies between making profit in the relative short term and investing time in the study of new techniques and practices. To spread awareness of new concepts and techniques, it is not enough for the research community to publish results, researchers must also more actively meet practitioners in their current situation; if Mohammed cannot come to the mountain, the mountain has to come to Mohammed. We believe that standards and recommended practices are an important means of bridging this gap between research and practice.

There are standards a company has to be aware of concerning the products it produces (e.g. network protocols or programming languages). There is also a class of standards named "recommended practices", which describe good work practices that are believed to yield high-quality products in a cost-effective manner. Recommended practices are aimed at practitioners, but to our experience "recommended practices" are not used as much as they deserve. With this paper we would like to increase the interest for recommended practices in general and the IEEE Standard 1471-2000 [7] in particular, by describing an application of the latter. In doing this, we address the following questions:

- There is typically very little extra time available for introducing a "recommended practice"; can it be beneficially introduced at a very low cost?
- What criteria should be used to evaluate whether such an application is successful or not?

With the support of a case study, presented in section 2, we show in section 3 that a very lightweight introduction of the recommended practice can be beneficial using some evaluation criteria. In section 4 we describe related work. In section 5 we present our conclusions.

2. The Case Study

The case study concerns Westinghouse, a US-based industrial enterprise with thousands of employees operating in the nuclear business domain, which acquired the Swedish company ABB Atom (~800 employees) in late 2000. The software developed in the (formerly) two organizations overlapped to some extent, and three systems were identified that should be integrated. A project was launched with the aim of arriving at a decision on the architecture for an integrated system. In this paper, we will focus on how the use of a recommended practice was used in this process.

2.1 Background

The project was divided into three phases, each containing different stakeholders: evaluation of existing systems, design and analysis of future system alternatives, and decision of which design alternative to use. Each phase had to include people representing the existing systems as well as the two sites. There were three internal deliverables defined: a draft requirements specification, descriptions of the three existing systems, and one or more alternative descriptions of a new integrated system. See Figure 1.

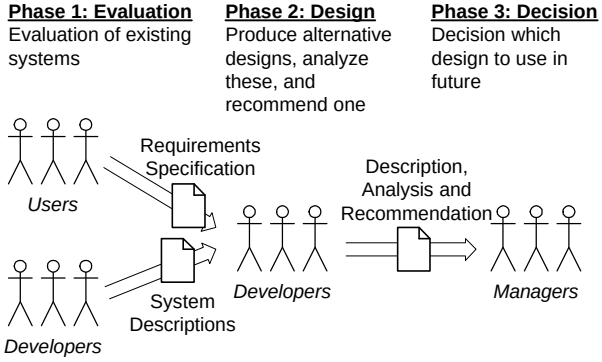

Figure 1. Project phases.

The role of the author was that of an active member of the developers group and the responsibility of documenting the outcome of the meetings as well as to prepare documentation for the different project phases. The author believed it to be beneficial for the project to introduce to the developers and architects the concepts of software architecture [1,3,4,5,6,7]. Given very limited preparation time by the other project participants, he decided to use the IEEE Standard 1471-2000, "Recommended practice for architectural description of software-intensive systems" [7].

Previously, a number of meetings had been held characterized by "brain-storming", during which no decisions were reached. Thus, there is an indication that the changes made in the project design (including the use of the recommended practice) were beneficial. We will in the following describe the project and argue how the changes were improvements, which eventually enabled a well-founded decision on which architectural alternative to use for an integrated system.

2.2 The Recommended Practice

The recommended practice contains a framework of concepts but does not mandate any particular architectural description language or set of viewpoints to use. The following key terms are defined [7]:

- **Architecture.** "The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution."
- **Architectural Description (AD).** "A collection of products to document an architecture."
- **View.** "A representation of a whole system from the perspective of a related set of concerns."
- **Viewpoint.** "A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis."
- **System stakeholder.** "An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system."
- **Concern.** "Each stakeholder typically has interests in, or concerns relative to, that system. Concerns are those interests which pertain to the system 's development, its

operation or any other aspects that are critical or otherwise important to one or more stakeholders. Concerns include system considerations such as performance, reliability, security, distribution, and evolvability."

To summarize the terminology: the *architecture* of a system should be described (as an *architectural description*, AD) in several *views*, each of which should adhere to a *viewpoint*. The documentation of the AD in each view must have a rationale; i.e. it must address the *concerns* of one (or more) *stakeholder*.

2.3 Project Preparations

In advance of the first project phase, the author condensed the most relevant parts of the recommended practice into a five-page summary, which was sent together with other project information to the participants one week in advance. The summary was focused on two parts of the recommended practice:

- **The technical concepts.** Some of the concepts of software architecture were explained, to provide a basis for descriptions, discussions, and analysis. The concepts of architecture, component, connector, view, viewpoint, stakeholder, and concern were used.
- **Focus on concerns.** According to the recommended practice, all activities and artifacts should focus on addressing stake-holders' concerns. By using the concept of "concerns" explicitly, the discussions should be less likely to drift away too far from the essentials. A preliminary list of concerns perceived as important by the author or communicated in advance was included, intended to be further refined as new concerns appeared in the discussions.

The participants were expected to prepare themselves by spending one day (eight hours) studying the project documentation. At the time of the first meeting, only one participant out of three (apart from the researcher-secretary himself, who prepared this document) had studied it in advance. The recommended practice summary was therefore briefly presented.

2.4 Phase One

In phase one, the task was to understand the three systems as detailed as time allowed and forward this information to the second phase. The existing documentation of the systems was of quite different kinds. Although all had overall system descriptions, they were of an informal and intuitive kind (for example, none of them used UML [2,14]), and none consisted of an explicit architectural description using the terminology established in the software architecture field (e.g. separated into views), which meant that the descriptions were not readily comparable. One of the purposes of the first phase was therefore to produce an architectural description of each of the systems, in as similar manner as possible, to be able to use as an input in the second phase. As time was limited, the intention was to maintain a balance between the following elements:

- **Addressing concerns.** Every important concern was dealt with to some extent. This means that sometimes the participants shifted focus to another concern, although the first one was not completely addressed – it was considered better to deal with every concern on the list at a high level than to analyze only some at a detailed level (it is better to be "somewhat" sure about maintainability and performance than being very sure about only performance).
- **Architectural refinement.** Within a view, based on a concern that needed to be clarified, the description was refined (a component "zoomed in"). But at some point, further refinement was of less practical interest compared to dealing with another concern or refinement within another view.
- **Annotations of components and connectors.** The components and connectors were annotated with relevant information (templates were provided).

As the meeting proceeded, two viewpoints were found to reveal the most about how the systems addressed the concerns of the stakeholders: a code structure view and a runtime view. UML was used, although in a somewhat informal manner during the meeting. The components and connectors were annotated with

information on e.g. programming language and size. At the end of the meeting, there were three comparable architectural descriptions.

2.5 Phase Two

In the second phase, the task was to create a design for the new, integrated system. By having created the architectural descriptions in the first phase it was possible to discuss similarities and differences in a structured way, both at a structural level and component-by-component. By having the components separated into two different views, runtime components (processes or threads) could not be confused e.g. with code components (modules such as general libraries or specific programs). Moreover, the discussion was guided by the list of stakeholder concerns, which was extended or modified from time to time as the discussions revealed additional concerns.

It was relatively easily to use the existing descriptions and "merge" them into a new system. The difficulties experienced in this process lay no longer in the actual analysis but in agreeing on the best way of solving tradeoffs, given the estimated properties. After some compromises two alternatives were left.

2.6 Phase Three

In phase three, the use of the recommended practice was less apparent. Still, the architectural descriptions of alternative solutions created in phase two, and the analyses of them, were used as a basis for the decision. The managers participating in the last phase needed some help from the developers to be able to understand the architectural descriptions, and when translated to plain English it was possible to understand it.

The actual decision on which alternative to use for the integrated system was ultimately based primarily on organizational concerns rather than technical ones – but concerns of a stakeholder nevertheless. This emphasizes the sense of using the concept of "concerns" explicitly, both in the project and in the recommended practice itself.

3. Measurable Benefits

Similar sets of meetings had been carried out before, without using the recommended practice. These meetings had a more "brainstorming" character, and the participants were not able to agree on an integration solution. There is thus some scientific support for the hypothesis that the introduction of the recommended practice was an improvement (although there were other changes in the project design as well, which we intend to publish elsewhere).

3.1 Changes

The use of the recommended practice changed the way the architectural alternatives were prepared in several ways, arguably improvements:

- **Similar Descriptions.** The existing documentation was too different from system to system to be readily compared. The systems were described in a more uniform way through the adoption of certain concepts: views, components, and connectors. When designing a new, integrated system, it was easier than during the previous (failed) sets of meetings to combine components from the three systems and be confident in the informal analyses made.
- **Relevant discussions.** By focusing on stakeholder concerns, the focus of the discussions stayed on relevant issues. Sometimes a discussion had to be interrupted either because it was digging into some irrelevant detail or in order for another concern to be addressed; but sometimes the discussion was indeed relevant and it was the list of concerns that had to be modified.
- **Less number of alternatives.** In the second phase, the developers were able to agree on two main alternatives and discard several alternative architectures that were discussed in the previous meetings.
- **Confidence in analysis.** Not only was it easier than before to merge the systems, the developers also had greater confidence in their estimates of its properties than they had had in the previous series of meetings.

The two parts of the recommended practice that the researcher had intended to focus on (the technical concepts and stakeholder concerns) thus

lifted the discussions from the previous "brainstorming" level to a more structured one.

3.2 How To Evaluate Success

How successful was the implementation of the recommended practice? The case study illustrates that the measure of success depends on the evaluation criteria used – do we mean that a single *project* was more efficient than otherwise, or that it is used throughout an *organization* in a consistent manner? The concepts were not the most prominent during the project discussions; the concepts of viewpoints and connectors were not fully understood by all participants; it is unknown if the recommended practice will be used in the *organization* in the future. It could therefore be argued that the use of the recommended practice was unsuccessful. But from the perspective of the outcome of the *project*, the concepts provided a tool that improved the discussions to some degree, which should be considered a (partial) success: the discussions were kept more focused, and the architectural descriptions produced were similar enough to enable comparison. This made the participants more confident in the results and their analysis.

4. Related Work

Our case study emphasizes the importance of documenting and evaluating the architecture of a software system. UML [2,14] and the framework provided by the recommended practice [7] were used explicitly. Elaboration on documentation issues in general can be found in [4,6]. Which views to use are discussed in e.g. [4,6,11]. The importance of architecture in the software process is discussed by e.g. [6,13]. The IEEE Architecture Group's resource page on the IEEE 1471-2000 [7] may be found at: http://www.pithecanthropus.com/~awg/public_html, but this web page currently does not list any successful applications of the recommended practice.

While there are processes and methodologies described that could have been used, none of them were completely feasible for the task. The rest of this section will briefly discuss the arguably most widely known and explain why none of those were chosen.

The *Architecture Trade-off Analysis Method* (ATAM) [5,9] builds on stakeholder-generated scenarios and has been reported useful in practice [5,10]. Several of the methods nine steps would not be possible to carry out within the case study project: in step 2 the business drivers should be presented, but these were not well defined (it was e.g. discussed throughout the project whether the system would be used only in-house or also deployed to external customers); in step 5, quality attributes are to be organized, but these were not specified in advance but found during the project. Of course, it would have been possible to reorganize the project so as to define business drivers and important quality attributes in a separate phase beforehand. In many senses, it would even have been beneficial. But, and this is our point in this paper, it would require efforts of an organizational kind that one cannot expect to be carried out.

The *Software Architecture Analysis Method* (SAAM) [1,5,8] is a predecessor of ATAM and has also been reported useful in practice [1,5,12]. Given an architectural description, it supports the analysis of virtually any system property, as defined by scenarios, but is oriented towards analyzing functionality and maintainability [5]. In the case study, it would have been too time-consuming to analyze the concerns in detail. There were several architectural alternatives, a large number of concerns to analyze (originally 13), and as said above, the exact properties or scenarios to analyze were not defined in advance. Therefore the project relied more on the analysts' experience and intuition – for good and bad.

The description of the *quality attribute-oriented software architecture design method* (QASAR) [3] includes numerous case studies where it has been used. According to this methodology, one should first design an architecture that fulfills the functional requirements (which the three existing system do) and then refine the architecture until the quality attributes are satisfactory. In the case study, this was what actually happened to some extent, but with more intuition than formality in the analyses

(as said, the actual attributes and evaluation criteria were not fixed in advance, and there was not enough time for more thorough analyses). One difference between the case study and the methodology description was that there were several alternatives in development simultaneously, on direct orders from management.

The *Active Reviews for Intermediate Designs* method (ARID) [5] builds on Active Design Reviews (ADR) and incorporates the idea of scenarios from SAAM and ATAM. It is intended for evaluating partial architectural descriptions, which is exactly what was available during the project work. However, it is intended as a type of formal review involving more stakeholders, which was not possible because the project schedule was already fixed, and too tight for an ARID exercise.

The basic reason for not using any of these methodologies is that when new practices are to be introduced "on the fly" in an industrial project, it is not possible to adjust the project. It is the practices to be introduced that have to be adjusted so as to make a minimal negative impact on the project, while having at least some positive impact.

5. Conclusion

As a participant in the project, it was possible to introduce new concepts and use them in the actual work even though there was very little time for the participants of the project to study and adopt new concepts. The most important artifact used was a recommended practice, the IEEE "Recommended practice for architectural description of software-intensive systems" [7]. The case study shows how a recommended practice can be beneficially introduced into a *project* without affecting its schedule negatively, although it is unsure whether the *organization* has adopted it and will use it in the future. To make a long-lasting impact on an organization, the implementation of these practices requires a champion within the organization to promote their use. The practices were used on the Westinghouse software integration project due to the efforts of the present author and would likely be used in the future if a motivated individual within Westinghouse is indoctrinated in the IEEE 1471-2000 methodology.

Based on the case study, we suggest that a recommended practice be introduced in the manner we have described due to its low cost. If this first, perhaps partial, application to a *project* is successful, and the first users gain insight, experience and confidence in it, it might be more widely used throughout the *organization*, thus making future projects more efficient.

A number of objections can be raised concerning how the project was performed – the participants were insufficiently prepared, no established methodology was used, the evaluation relied heavily on intuition and experience, the evaluation criteria were not clear, etc. The purpose of this paper is not to evaluate the project or the organization as such, but to describe how a recommended practice can be used to improve it without requiring changes to a project that already has a tight schedule and limited resources. In this respect, we believe we have shown that a recommended practice with little effort can be used to introduce new concepts and arguably improve the outcome of a project to some extent. Still, we must bear in mind that our conclusions are weakened by the fact that there were other changes in the project design which we also intend to publish, factors we consider to be at least equally important factors for the success of the project (as compared to the previous meetings).

Although we have argued that the application of the recommended practice was beneficial in the project presented, one important remaining question is whether the recommended practice, and the concepts embodied in it, will remain in the minds of the project participants and increase the state of practice in the organization. Other ways of introducing it may prove more successful in making a longer-lasting impact, and we are looking forward to more reports on applications of the recommended practice.

6. References

[1] Bass L., Clements P., and Kazman R., Software Architecture in Practice, ISBN 0-201-19930-0, Addison-Wesley, 1998

[2] Booch G., Rumbaugh J., and Jacobson I., The Unified Modeling Language User Guide, ISBN 0201571684, Addison-Wesley, 1999

[3] Bosch J., Design & Use of Software Architectures, ISBN 0201674947, Addison-Wesley, 2000

[4] Clements P., Bachmann F., Bass L., Garlan D., Ivers J., Little R., Nord R., Stafford J., Documenting Software Architectures: Views and Beyond, ISBN 0-201-70372-6, Addison-Wesley, 2002

[5] Clements P., Kazman R., Klein M., Evaluating Software Architectures: Methods and Case Studies, ISBN 0-201-70482-X, Addison-Wesley, 2002

[6] Hofmeister C., Nord R., and Soni D., Applied Software Architecture, ISBN 0201325713, Addison-Wesley, 2000.

[7] IEEE Architecture Working Group, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE Std 1471-2000, IEEE, 2000

[8] Kazman R., Bass L., Abowd G., and Webb M., SAAM: A Method for Analyzing the Properties of Software Architectures, In Proceedings of the 16th International Conference on Software Engineering, 1994

[9] Kazman R., Klein M., Barbacci M., Longstaff T., Lipson H., and Carriere J., The Architecture Tradeoff Analysis Method, In Proceedings of the Fourth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), IEEE, 1998

[10] Kazman R., Barbacci M., Klein M., and Carriere J., Experience with Performing Architecture Tradeoff Analysis Method, In Proceedings of the 21st International Conference on Software Engineering, New York, 1999

[11] Kruchten P., The 4+1 View Model of Architecture, IEEE Software, volume 12, issue 6, 1995.

[12] Land R., Improving Quality Attributes of a Complex System Through Architectural Analysis - A Case Study, In Proceedings of 9th IEEE Conference on Engineering of Computer-Based Systems (ECBS), IEEE, 2002

[13] Paulish D., *Architecture-Centric Software Project Management: A Practical Guide*, ISBN 0-201-73409-5, Addison-Wesley, 2001

[14] UML Home Page, URL: http://www.uml.org/