Importance of Software Architecture during Release Planning

Markus Lindgren[†] Christer Norström[‡] Anders Wall[*] Rikard Land[‡]

[†]ABB Force Measurement
Västerås, Sweden
markus.lindgren@mdh.se

[‡]Mälardalen University
Dept. of Computer Science
Västerås, Sweden

[*]ABB Corporate Research
Västerås, Sweden
anders.wall@se.abb.com

Abstract

Release planning is the process of deciding what to include in future release(s) *of a product. In this paper we look at how software architects are involved during release planning in industry today, and how architectural issues are considered during this phase.*

1. Introduction

Release planning can be seen as company-wide optimization problem involving many stakeholders where the goal is to maximize utilization of the often limited resources of a company and turn them into business benefit [9]. The release planning results in a decision of what to include in future *release(s)* of a product. In making this decision one needs to consider how to make a product profitable both in the short- and long-term. As input to release planning are a set of *needs* that, when realized into a product, provides some business/customer value. Normally the cost of implementing all of the proposed needs is larger than the budget allocated to a release, therefore a decision needs to be made of what to include in a release and what to post-pone. Thus, the *set of needs* needs to be prioritized in order to maximize business value of the needs included in a release. In addition, there are constraints that need to be considered during release planning [9]. For example, time-to-market, dependencies between different needs, required competencies, competitors' product offerings, new technology, market demand, and quality aspects, as is illustrated in Figure 1. A *need* is a proposal for a change that normally is negotiable to some extent; needs later become project requirements.

Today most product development is required to be based on existing systems for economical and time-to-market reasons [8]. This forces existing systems to be evolved by adding features or improving existing features. During such evolution the existing system often has impact on which extensions/improvements are easy/hard/possible to perform.

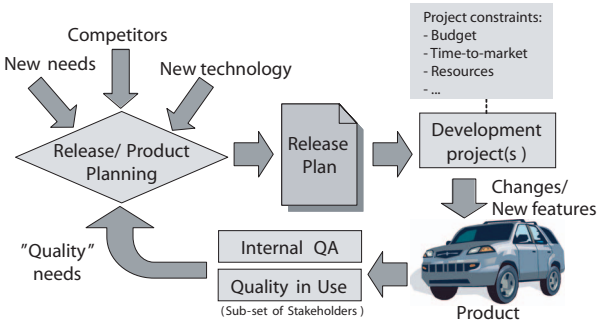

Figure 1. Overview of release planning.

Within the research community it is known that the early design decisions, manifested by the software architecture, "...are the most difficult to get correct and the hardest to change later in the development process, and they have the most far-reaching effects" [1]. These considerations can have impact on what needs to be prioritized in a release. Examples of relevant areas for a software architect to address during release planning are:

- Identify possible architectural constraints, e.g., assess whether a new need can be accommodated by a *local change*, a *non-local change*, or an *architectural change* [1]. There can be refactorings/technical debts [4] suitable to compensate for while introducing new needs.
- Develop cost and time estimations [1], identify possible technical risks associated with proposed needs.
- "The architecture inhibits or enables a system's quality attributes" [1], which can become troublesome for evolving systems with long-life, e.g., 10–20 years, since customer expectations normally change over time. Changes in expectations can impact the quality attributes a software architecture must support. These changes need to be identified and addressed.
- Identify if any of the proposed needs introduce an architectural conflict, which, for example, can be identified using methods such as the ATAM [5].

The consequence of poor release planning can be, for

0-7695-3092-3/08 $25.00 © 2008 IEEE
DOI 10.1109/WICSA.2008.10

example, increased technical debts [4] which are costly to address, increased need for replanning (resulting in project change requests), poor product quality, and lost business opportunities (when missing important market dates or when having "wrong" product features and/or quality). These issues have direct impact on the profitability of a company.

There are a number of reasons why it would be natural for a company to address architectural issues during release planning. However, in our experience this is not always the case, which is our motivation for studying this topic. In addition, there are a number of interesting research questions related to this, such as:

- How are release planning decisions made? Who participate in making the decisions?
- How do companies balance investments in quality improvements vs. feature growth to be profitable in the short and long-term?
- When and how are architectural issues addressed during release planning?

To study this, we have performed a multiple case study involving 7 industrial companies, most of them part of the Forbes Global 2000 list. Our interviews have mainly been with *product management*, since product management of a company normally is responsible for performing release planning. All companies in this study develop embedded systems with different degrees of *software*, *hardware*, and *mechanics* in them. However, our focus is on release planning of the software part of these products.

The outline of this paper is as follows: Section 2 presents a selection of related work, Section 3 presents the research method used in this study, Section 4 provides some data from the seven industrial companies involved in the study, and Section 5 presents the findings of our study. Finally, Section 6 concludes the paper.

2. Related Work

Release planning has many similarities with *product planning* and *requirements prioritization*. Research within release planning is mainly focused on formalization of the release planning problem; typically done by formulating the problem as an optimization problem where *customer value* should be maximized under a number of constraints; a comparison of such methods/tools is available in [10].

Several methods exist for software architecture evaluation as can be seen in [3][1], where the one mainly used in practice is the ATAM [5]. Klein [6] discusses how the role of a software architect changes as the software ages, Bredemeyer and Malan [2] discuss technical, business, organizational politics, consulting, and leadership skills that a software architect should have, and Bass, Clements, and Kazman [1] also discuss the role of the software architect.

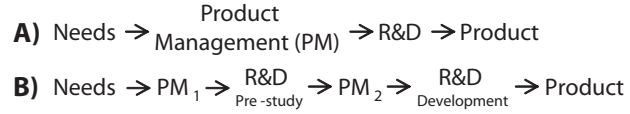

Figure 2. Release planning process *A* and *B*.

3. Research Method

We have followed the recommendations by Yin [11] for multiple case studies. Due to space restrictions, we only describe a few steps of our research method. As an example, multiple researchers have been used in most of the steps in the study thereby improving *construct validity* and *internal validity*, as well as a counter for *researcher bias*. *External validity* is improved by the relatively large number of companies and persons included in our study. To increase *reliability* all collected data, and derivations thereof, are stored in a database. We have used semi-structured interviews as the primary data collection method, in total 16 interviews.

Saliu and Ruhe [10] have identified a number of release planning *key-aspects*, which we have used as a reference model, extended with software architecture aspects, to simplify comparisons between the companies.

4. Cases

Table 1 summarizes release planning at the seven companies in the study, where *Objectives* refer to criteria used in making release planning decisions and *Involvement* refers to how the software architect/R&D is involved. More complete case descriptions are available in [7].

5. Findings

5.1. Structured Pre-Studies

Based on our data we see that there are two basic ways of doing release planning, referred to as process *A* and *B*. In process *A* the needs that should be prioritized is an input to product management (PM), as illustrated in upper part of Figure 2. Product management prioritizes the needs to a suitable set that can fit within the release project's budget, and R&D realizes the list of needs into a product. Typically R&D estimate development time, which is used by product management to determine a suitable set of needs that can fit within the release project's budget.

In process *B* the needs are gradually prioritized as more details become investigated, as illustrated in lower part of Figure 2. In the first step (PM1) product management reduces the set of needs, by prioritization, to a set suitable for investigation in pre-studies by R&D. R&D investigate, e.g., how the needs can be realized, what impact these needs have on existing functionality and quality in the product,

Case 1 *Objectives:* Each product has an attribute profile (about 20 attributes), with roots in the company profile. *Involvement:* Pre-studies (R&D) investigate consequences of proposals, results in decision material.
Case 2 *Objectives:* Needs are prioritized based on 8 core values, and 3 prestige values (higher priority). *Involvement:* R&D is involved via pre-studies.
Case 3 *Objectives: Measure of Effectiveness* (MOE in IEEE 1220) for need prioritization; MOE is a mathematical expression. *Involvement:* The architect should investigate system concepts in parallel with product management's work.
Case 4 *Objectives:* The goal is to maximize business value. *Involvement:* R&D involved via pre- and feasibility-studies. System responsibles and product management propose needs.
Case 5 *Objectives:* Starts from an economic viewpoint, i.e., how to turn a need into profit. *Involvement:* There is low/no R&D involvement.
Case 6 *Objectives:* Not explicitly defined. Normally customer needs prioritized over cost-cut projects. *Involvement:* Plan quantification by R&D. Product management & market responsibles propose/prioritize needs.
Case 7 *Objectives:* A *release profile* describes the strategy for the coming release; base for need prioritization. *Involvement:* Plan quantification by R&D.

Table 1. Brief case descriptions.

cost and time estimations for realization of the needs, and required competencies. The results of the pre-studies are documented as decision material and returned to product management. Using this decision material product management makes a second round of prioritization and decides a suitable set of needs that fit the release project's budget (PM2), which R&D realizes into a product.

Pre-studies are sometimes employed by companies using process *A*, but these are not as *structured* as the ones used in process *B*. By structured pre-studies we refer to pre-studies that have: 1) an established format for what kind of decision material the pre-study should produce, and 2) the decision material is used to aid in prioritization of needs. We classify Case 1, Case 3, and Case 4 to use process *B*, while Case 5, Case 6, and Case 7 use process *A*; we have insufficient data for making explicit claims for Case 2.

5.2. Architectural Awareness

Our impression is that the *architectural awareness among product management generally is low*, which is motivated by the interview responses together with the interviewees experience with software, which we have analyzed

Case Interviewee	1 A	1 B	2 C	2 D	3 E	4 F	4 G	5 H	5 I	6 J	6 K	6 L	6 M	7 N	7 O	7 P
SW Edu.	M	N	L	N	-	N	L	N	N	M	N	M	?	M	H	N
SW Exp.	Y	N	N	N	-	N	N	N	N	Y	N	Y	?	M	N	N
RP Role	N	Y	N	N	-	Y	N	Y	Y	Y	Y	N	N	Y	Y	Y
Awareness	M	L	L	L	-	M	L	L	L	L	L	M	-	M	L	L

Table 2. Architectural awareness of interviewees A–P, where H=High, M=Medium, L=Low, Y=Yes, and N=No.

and summarized in Table 2, where there are columns for software development education (*SW Edu.*), software development experience (*SW Exp.*), whether the interviewee takes part in release planning (*RP Role*), and finally, our qualitative conclusion concerning the architectural awareness for each interviewee. (The grey text in Table 2 mark cases that required some analysis to reach a conclusion.)

By *architectural awareness* we (informally) refer to the ability of a person to address the architectural issues from Section 1. We grade architectural awareness as follows:

High A person who is capable of making the architectural assessments (as discussed in Section 1) and taking necessary action to address these issues.

Medium A person who has an understanding for architectural considerations, and can explain the relevant considerations to other people, but doesn't have the necessary skill to address these issues himself/herself, hence, these assessments are deferred to other people.

Low A person that might have some understanding for architectural considerations, but doesn't defer these assessments to people with required skills.

Now, even though the architectural knowledge of product management is low doesn't necessarily mean that architectural issues are not considered early on. For example, in Case 1, Case 3, and Case 4 it is clear that such technical considerations are addressed via pre-studies, which reach product management in the form of decision material. In these cases this is explicit in their processes as well. However, in Case 5, Case 6, and Case 7 it is uncertain if such considerations are taken during release planning. In Case 2 we have too poor coverage in our interviews for explicit claims.

Case 4 seems to have the most thought through release planning process, simultaneously other sources indicate that Case 4 has highest CMMI level among the studied companies. Hence, it is possible that architectural awareness in release planning is related to organizational maturity.

5.3. Quality Improvements vs. Features

Another topic covered during our interviews is how the companies balance investments in quality improvements vs.

feature growth. Compared to feature growth, which relatively early becomes visible in price lists, there is a longer feedback loop for quality, which is another reason for early considering the consequences of too poor quality. However, for the companies in our study *no one seems to have a method*, or rule-of-thumb, for how to balance investments in quality and feature growth respectively.

Given that product management has low architectural awareness it would be natural to use the software architect/R&D for judgments concerning how (software) quality aspects can/need to be addressed. Without such input, combined with analysis of the consequences of performing quality improvements, it is probable that these issues do not become eligible for need prioritization.

For software architects working in organizations using process *A* it becomes relevant to look in more detail on how release planning decisions are made today, in order to find strategies for improving work practices.

5.4. Early Architect Involvement

In lack of good data, people use "gut-feeling", based on experience, to different degrees in making judgments concerning what to include in future releases. This "gut-feeling" can be based on many different things such as: what is of benefit for the company, benefit to my own department, benefit to my own country (in case of distributed development), and benefit to my own career. There can be many such reasons that affect how people argue/reason during release planning. To increase the chances of their own proposals getting through product management, and supposedly people generally, use lobbying, sell-in, and politics.

Our study indicates that the release planning decisions are in all cases, to various degrees, affected by gut-feeling, lobbying, politics, and strong individuals.

A software architect needs to be aware of how release planning decisions are made since this (partly) controls the available resources, and the goals these resources should strive for. The need for an architect not only to have technical skills is reported in several other studies [2][6].

Our data indicates that companies employing structured pre-studies are *less* dependent on lobbying, sell-in, and politics for release planning decisions. Therefore we conclude that early architect and/or R&D involvement, e.g., via pre-studies, reduces the need for "gut-feeling", lobbying, etc.

Now the question becomes, is it better to make use of gut-feeling than to base decisions on material produced in structured pre-studies? For many cases gut-feeling, based on long proven experience with the system and by people who (presumably) have their position for good reasons, can probably be as successful. But as the complexity of products increase our hypothesis is that the possibility of experience leading to wrong conclusions become higher.

6. Conclusion

We have performed a multiple case study involving seven industrial companies and investigated how software architectural quality concerns are considered during release planning, with a focus on the evolutionary phase of product development. We have identified the following findings:

- product management generally has low architectural awareness,
- there is no method for how to balance investments in quality improvements vs. feature growth, and
- the role of "gut-feeling", lobbying, and sell-in is lower in the companies that involve the software architect/R&D.

These findings have implications on the role of a software architect and the issues he/she needs to be aware of. Perhaps most important is the need for the software architect to be involved in the release planning decisions, since without such involvment it is possible for important quality issues to be left out. Existing literature within software architecture does not give sufficient attention to this issue.

In future work we plan to address the balance between feature growth and investments in (architectural) quality.

References

[1] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice* 2^{nd} *edition*. Addison-Wesley, 2003.

[2] D. Bredemeyer and R. Malan. The Role of the Architect. Web-site: http://www.bredemeyer.com, 2006.

[3] L. Dobrica and E. Niemela. A survey on software architecture analysis methods. *IEEE Transactions on Software Engineering*, 28(7), July 2002.

[4] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 1999.

[5] R. Kazman, M. Klein, and P. Clements. ATAM: Method for Architecture Evaluation. Technical Report CMU/SEI-2000-TR-004, CMU — Software Engineering Institute, 2000.

[6] J. Klein. How Does the Architect's Role Change as the Software Ages. In *Proc.* 5^{th} *IEEE/IFIP Working Conferance on Software Architecture*. IEEE Computer Society, 2005.

[7] M. Lindgren. Release Planning in Industry: Interview Data. Technical Report MDH-MRTC-219/2007-1-SE, Mälardalen Real-Time Research Centre (MRTC), 2007.

[8] G. Mustapic, A. Wall, C. Norström, I. Crnkovic, K. Sandström, J. Fröberg, and J. Andersson. Real World Influences on Software Architecture - Interviews with Industrial Experts. In *WICSA 2004*. IEEE Computer Society, 2004.

[9] G. Ruhe and M. O. Saliu. Art and Science of Software Release Planning. *IEEE Software*, 22(6):47–53, 2005.

[10] M. O. Saliu and G. Ruhe. Supporting Software Release Planning Decisions for Evolving Systems. In *29*th *Annual IEEE/NASA Software Engineering Workshop*, pages 14–26. IEEE Computer Society, 2005.

[11] R. K. Yin. *Case Study Research: Design and Methods,* 3^{rd} *Edition*. Sage Publications Inc., 2003.