Improving Solution Architecting Practices

Poort, E.R.

2012

10 Concluding Remarks

In this chapter, we present our key conclusions. We revisit the research questions and summarize how they have been answered in the thesis, and list the novel contributions to the field. We also discuss promising directions for further research on the topics treated.

10.1 Conclusions

The key high-level conclusions of this thesis are listed below.

On the topic of solution architecting:

1. *Solution architecting is a risk- and cost management discipline.* This is demonstrated in Chapters 8 and 9. The Risk- and Cost Driven solution architecting approach has significant positive impact on the work of most solution architects trained in it.

2. *CMMI support for architecting has improved significantly with version 1.3, but could still be further improved by adding guidance for architecture governance and architecting during the sales phase.* See Chapter 6.

3. *Dealing with emotions is a crucial factor in how architectural knowledge sharing leads to successful projects.* See Chapter 7.

On the topic of NFRs:

4. *Critical NFRs should be quantified, but we should beware of* premature *quantification.* See Chapter 3.

5. *Tendering rules and regulations have a detrimental effect on the quality of IT solutions. The key to successful IT solutions is in trust between customers and suppliers.* Also from Chapter 3.

6. *Modifiability deserves more attention than it is getting now.* Observed in Chapter 4.

One final overall conclusion:

7. *Good solution architecting is not so much a technical problem, but rather a socioeconomic one.* The most important observations above are not technical in nature. They revolve around non-technical keywords like trust, emotions, risk and cost, responsibility and authority.

In the remainder of this chapter, we will show how these conclusions are underpinned by the material presented in this thesis.

10.2 Contributions

Our journey towards improving solution architecting practices in Logica gave us the opportunity to research a number of interesting questions, presented in the introduction to this thesis (Chapter 1). By looking at how these questions were answered, we will now summarize our new contributions to the field, and relate them to the conclusions presented above.

10.2.1 How can Non-Functional Requirements be handled to improve the success of IT solutions and the projects delivering them? (RQ-1)

How can a solution be structured to best address conflicting Non-Functional Requirements? (RQ-1a)

In Chapter 2, we present a new framework that both provides a model and a repeatable method to transform conflicting requirements into a system decomposition, called Non-Functional Decomposition. NFD is a technique, based on the relationship between functional and non-functional requirements, that brings more clarity and structure in the mapping of requirements onto a solution architecture. Our new framework reveals rationale behind existing architectural patterns and tactics, and can be helpful in developing new patterns and tactics to deal with conflicting NFRs.

What is the best way to quantify Non-Functional Requirements across a contractual divide between customer and supplier? (RQ-1b)

In Chapter 3, we identify some key issues related to NFR quantification in customer/supplier relationships. We argue that economic justification of NFR quantification requires knowledge of the solution architecture.

We argue that critical NFRs should be quantified, but we should beware of *premature* quantification: as our real-life examples illustrate, prematurely quantified NFRs can cripple projects and lead to diverging points of view in customer/supplier relationships that are very hard to resolve. Optimal quantification requires sharing of information between customer and supplier, and it requires time to establish at least a reasonably proven estimate for the cost and value relationships. We suggest a possible way to create better NFR quantification circumstances for customers and suppliers: by means of a requirements convergence plan.

We conclude that trust between customers and suppliers in the IT industry is key to successful solutions. This is a matter of attitude. With the ever growing complexity of IT systems and projects, transparency and awareness between customers and suppliers about NFRs is essential to the feasibility of IT projects. So is willingness to share the risk of unquantified NFRs. Both transparency and risk sharing cannot exist without trust.

How do architects perceive and deal with non-functional requirements? (RQ-1c)

Chapter 4 presents the results of a survey about dealing with non-functional requirements (NFRs) among architects. We find that, as long as the architect is aware of the importance of NFRs, they do not adversely affect project success, with one exception: highly business critical modifiability tends to be detrimental to project success, even when the architect is aware of it. IT projects where modifiability is relatively business critical perform significantly worse on average. Our conclusion is that modifiability deserves more attention than it is getting now, especially because in general it is quantified and verified considerably less than other NFRs. Practitioners should be careful when dealing with IT projects with a strong focus on modifiability. We advise to pay particular attention to aspects like managing customer expectations, because it seems that customer satisfaction especially is significantly lower on average in this type of IT projects.

Furthermore, IT projects that applied NFR verification techniques relatively early in development were more successful on average than IT projects that did not apply verification techniques (or applied it relatively late in development). Thus, practitioners

should be aware that the long term benefits of verification outweigh the short term extra costs.

10.2.2 What is a good solution architecting approach to improve an IT service provider's success? (RQ-2)

What is the nature of solution architecting in the business context of a large IT services company? (RQ-2a)

In Chapter 8, we present our innovative view on the nature of solution architecture: as a risk- and cost management discipline. This view extends existing views of architecture as a higher level abstraction and as a set of design decisions. In comparison with these existing views, it helps architects better order their work, and it helps in better communicating about the architecture with stakeholders in business terms. We also provide guidance on implementing this view in industrial contexts.

What requirements does an architecture process need to fulfill in order to comply with CMMI maturity level 3? (RQ-2b)

Chapter 6 identifies the requirements to make a generic architecting process compliant with CMMI Maturity Level 3, and analyzes the process areas significant to architecting. Our conclusions are that architecture is not a well-defined concept in the CMMI 1.1, but it is improved in later versions of CMMI. CMMI can still be improved in the areas of architecture governance, facilitating the sales phase and learning from architectural choices.

How do architectural knowledge sharing practices relate to challenges in solution delivery projects? (RQ-2c)

In Chapter 7, we describe a survey to gain insight into the mechanisms around architectural knowledge sharing in projects. The analysis shows that architects face many challenges sharing architectural knowledge in projects, especially in large projects. Most of the common challenges appear to be generally neutralized somehow, since they show no correlation with project success. The only challenges that *are* correlated with project success are the ones related to interpersonal relationships. We conclude that *dealing with emotions* is a crucial factor in how architectural knowledge sharing leads to successful projects.

What architecting practices address an IT service provider's business requirements, and what guidance should they contain? (RQ-2d)

Logica's Risk- and Cost Driven Architecture approach is presented in Chapter 9. By combining ideas from [Jacobson et al., 2007] and [Kazman et al., 2006], we document architecting component techniques in practices, embedded in a new framework of structural dimensions to ease identification and integration of best fit practices in a particular context. The practices were harvested from Logica practitioners and enhanced by research. In a way, RCDA is the logical result and culmination of the work presented in this thesis:

- In Part I, we researched ways to handle NFRs; the resulting guidance is embedded in the RCDA practices *Dealing with Non-Functional Requirements* and *Requirements Convergence Planning*.
- In Chapter 5, we saw the importance of an environment where architects can argue their choices and priorities in an objective manner, and select practices that best fit those priorities, rather than follow fashion. RCDA stimulates such an environment by introducing practices that objectify architectural decisions and priorities, and put them in a business context.
- The RCDA core practices constitute a solution architecting process that fulfills the requirements of CMMI Maturity Level 3, as analyzed in Chapter 6.
- RCDA helps architects to deal with emotions (Chapter 7) by smoothing their communication with their solutions' stakeholders; translating architectural concerns and decisions into business terms like risk and cost (Chapter 8).

What is the effect of training architects in such architecting practices? (RQ-2e)

The results of a survey among architects trained in RCDA indicate that for the majority of trainees, the approach has significant positive impact on their solution architecting work. This is true for RCDA as a whole, for its principles, and for its individual practices. The positive effects, however, appear to be much stronger if the architects are in a position of responsibility and authority.

10.3 Discussion

The research presented in this thesis started out with the goal of finding out how to architect IT solutions that adequately serve their purpose, especially in client/supplier

situations. The research was performed within the context of technical assurance: assuring the feasibility, suitability and acceptability of solutions. Although the work treated many technical aspects, the most important conclusions are not technical in nature. They revolve around non-technical keywords like trust (Chapter 3), emotions (Chapter 7), risk and cost (Chapter 8), responsibility and authority (Chapter 9). Perhaps the main conclusion of this thesis should be that the problem of good solution architecting is not so much a technical problem, but rather a socioeconomic one; a conclusion already hinted at in e.g. [Clerc et al., 2007, Sutcliffe, 2008].

10.3.1 Future directions

Within Logica, the journey to improve solution architecting practices will continue. RCDA will be extended with new practices, some of which have already been identified in Chapter 9. More and more architects will be trained and gain experience applying the guidance. As evidence of the benefits of RCDA grows, parts of the approach will get an increasingly formal status in the company's business management system. The success of RCDA has prompted other engineering disciplines within the company to structure their guidance according to the Jacobson-like practices approach [Jacobson et al., 2007]. All of this will lead to new opportunities for research. One promising direction for such research is to relate the application of solution architecting principles and practices to actual *metrics* gathered in projects, rather than depend on surveys among architects. Another obvious extension of the research presented here is to repeat the research outside of Logica, and outside of the Netherlands.

Another interesting direction for research would be the relationship between solution architecting practices and architecting maturity. Could the identification of the solution architecting practices help in assessing an organization's solution architecting maturity, e.g. by enhancing or replacing existing Architecture Maturity Models like the IT Architecture Capability Maturity Model (ACMM) [US Department of Commerce, 2007]? Or could they be used to improve the assessment of individual architects' level of competence in frameworks like Open CA (formerly ITAC) [The Open Group]?

In this thesis, we have taken insights gathered in the software architecture community, and successfully applied them to the wider area of solution architecture. An interesting avenue of exploration would be to find out how the solution architecture principles and practices discussed here relate to other architecture genres, such as systems architecture and enterprise architecture. [Clements, 2009] provides a good basis for such work.