Improving Solution Architecting Practices

Poort, E.R.

2012

9

Risk- and Cost Driven Architecture: a Pragmatic Solution Architecting Approach

This chapter describes RCDA, the solution architecting approach we developed in Logica. The approach consists of a set of practices, harvested from practitioners and enhanced by the research presented in this thesis. We present the structure of the approach and its rationale, and the result of a survey measuring RCDA's effect among architects trained in the approach. The survey shows that for the majority of trainees, RCDA has significant positive impact on their solution architecting work.

9.1 Introduction

In Chapter 6, we described how in 2006, we started out to create a generic architecting process for Logica. The result of this effort is presented in this chapter. It is an integrated set of solution architecting practices, collectively called Risk- and Cost Driven Architecture (RCDA). We first give a short summary of the principles and practices that make up RCDA. We then explain the structure that integrates the practices, including the rationale behind it, and explain how the approach was implemented among the company's architects. We then present the result of a survey measuring the effect of RCDA training on the architects.

9.2 The RCDA Approach

In this section, we first describe the practices that are the basic building blocks of RCDA. We then explain the four key principles the approach is based on, and the way the approach is implemented in the organization. We conclude with a clarification and justification of the structure of RCDA: why it uses practices, what elements exists beyond the practices, and what structures are used to organize them.

9.2.1 RCDA practices

The basic building blocks of RCDA are *practices*. A practice is a way to systematically characterize a problem and address it. The practice concept will be explained further in §9.2.4. Practices that address closely related problems are clustered into practice sets. There are practice sets for Requirements Analysis, Solution Shaping, Architecture Validation, Architecture Fulfillment, Architectural Planning and Architectural Asset Management.

The practices of RCDA, ordered by practice set, are:

Requirements Analysis practices, where the requirements originating from stakeholders are prepared for shaping a Solution:

- **Architectural Requirements Prioritization** addresses the problem of pinpointing architecturally significant requirements and concerns, according to the principles laid out in Chapter 8.
- **Dealing with Non-Functional Requirements** gives guidance on handling NFRs, which are often underexposed and can have major impact on the solution; it contains the key elements from Part I of this thesis.
- **Stakeholder Workshop** is a practice for obtaining architectural requirements from stakeholders, based on the SEI's Quality Attribute Workshop [Barbacci et al., 2003].

Solution Shaping practices to define a solution's architecture:

- **Solution Selection** addresses the problem how to identify and select the best fitting strategy to fulfill architectural requirements on a Solution in an objective manner, implementing one of the lessons learned from Chapter 5.
- **Solution Shaping Workshop** is a special case of a Stakeholder Workshop. All Logica stakeholders are gathered to kick start the solution shaping process, led by the solution architect.

- **Cost-Benefit Analysis** helps architects to consider the return on investment of any architectural decision and provides guidance on the economic tradeoffs involved, based on the SEI's CBAM practice [Kazman et al., 2002].
- **Applying Architectural Strategies** describes how to implement architectural strategies selected in previous steps, determining a solution's structure according to the principles explained in e.g. [Bass et al., 2003, Gamma et al., 1995].
- **Architecture Documentation** documents the current state of the solutions architecture in a set of views [Kruchten, 1995, ISO 42010, 2011], focussed on effectively communicating the architecture to the relevant stakeholders.
- **Documenting Architectural Decisions** addresses the problem of tracking architectural concerns and decisions throughout their lifecycle, based on e.g. [Tyree and Akerman, 2005, Jansen and Bosch, 2005].
- **Solution Costing** gives guidance on early costing of delivering a solution using a selected architecture.

Architecture Validation practices aimed at validating the architecture developed in previous steps:

- **Architecture Evaluation** to create transparency and identify risks in the architectural decisions made, and to verify that the architecture meets its requirements; roughly based on [Abowd et al., 1997]
- **Architectural Prototyping** is performed when there is uncertainty about the feasibility of (parts of) an architecture which can be resolved by "trying it out".
- **Supplier Evaluation** helps architects identify potential risks when committing to delivering third party products as components of an architected solution.

Architecture Fulfillment practices, related to the development and delivery of the solution under architecture:

- **Architecture Implementation** making sure that the architecture developed and validated in previous steps is actually implemented in the solution.
- **Architecture Maintenance** provides guidance on taking an existing solution into operation, and on maintaining a solution's architecture once it is operational.
- **Blended Architecting** gives guidance on solution shaping and fulfillment in a geographically distributed solution delivery setting.

Architectural Planning practices, giving guidance on how to plan architecting activities:

- **Architecting Lifecycles** addresses the problem of when to apply RCDA practices, showing how the various RCDA practices map to certain common scenarios.
- **Requirements Convergence Planning** addresses the problem of finding the best balance of affordability between cost and benefit of architectural requirements, as explained in Chapter 3.
- **Architecture Contingency Planning** helps mitigate the risk of having to backtrack architectural decisions when it turns out an architecture cannot fulfill the stakeholders' needs.

Architectural Asset Management practices, aimed at re-using architectural assets like knowledge, reference architectures and re-usable components across solutions:

- **Architecture Knowledge Management** addresses the problem of codifying and sharing architectural knowledge such as patterns and lessons learned across the company, using techniques like those documented in [Zimmermann et al., 2007] and [Farenhorst and van Vliet, 2008].
- **Software Product Line Management** gives guidance on how to implement and manage software product lines so that they serve as the basis for multiple solutions, based on the SEI Software Product Line materials [Clements and Northrop, 2002].
- **Technology Monitoring** helps architects keep abreast of new developments that can provide more fitting alternatives for solution selecting.

The practices, grouped in practice sets, are visualized in Fig. 9.1.

9.2.2 RCDA principles

Risk- and Cost Driven Architecture is based on the following key principles:

- Cost and Risks drive architecture.
- Architecture should be minimal.
- Architecture as both Blueprint and Design Decisions.
- Solution Architect as Design Authority.

These principles are based on our experiences in technical assurance (see §1.1), enhanced by literature. Solution architects are encouraged to always keep these principles in mind when applying RCDA practices. The principles are applied through the individual RCDA practices as explained below.

The first principle, *Cost and risks drive architecture*, is explained extensively in Chapter 8. It is applied throughout RCDA, but most explicit in the Architectural Requirements Prioritization practice.

Architecture should be minimal is based on recent insights such as expressed in [Malan and Bredemeyer, 2002] and [Fairbanks, 2010]. In order to keep overview of the whole system, the solution architect's decisions should be limited to those that have critical impact on the system and its delivery - leaving a maximum of design space for filling in details within the constraints set by that architecture. This should of course be done with due consideration for the capabilities of those designers and developers, and should not detract from the clarity with which the architecture is communicated. Kazman, Bass and Klein formulate this principle as: "A software architecture should be defined in terms of elements that are coarse enough for human intellectual control and specific enough for meaningful reasoning." [Kazman et al., 2006] It is applied through the Architectural Requirements Prioritization practice.

Architecture as both blueprint and design decisions is based on the second view on architecture described in §1.2.1 and papers such as [Jansen and Bosch, 2005, Tyree and Akerman, 2005]. The architecture of a system is more than just a blueprint of its high-level structure - the design decisions leading to that structure and the underlying rationale are equally essential. No architectural description is complete without a well-documented set of design decisions. By thinking about architecture as a set of design decisions, we abstract away from the modeling details inherent to a particular technology or view, and are able to give generic guidance on how architects make trade-offs and document decisions. It also helps to focus on the rationale behind the decisions, which is important to future architects and those implementing or reviewing the architecture. This principle is applied through the practices Architectural Requirements Prioritization, Solution Selection and Documenting Architectural Decisions.

Solution architect as design authority is based on views like those documented in [Fowler, 2003] and [Clements et al., 2007]. The complexity of today's IT solutions requires that the most critical design decisions are made by one person with an overview of the whole system. This person should have the authority and the subject matter skills and knowledge to make such decisions. This role is distinct from the project manager's role, and is called the Solution Architect in RCDA. Of course, architecture is often team work, and architects should surround themselves with experts to help them make critical decisions - but in the end, no matter how big the design team, one person is responsible for making all the trade-offs and the final decision. This principle is applied through the RCDA Solution Architect role.

9.2.3 Implementation

RCDA is documented on a company intranet site. On the RCDA site, each practice is described according to a defined structure, with the following sections: Objectives, Approach, Roles, Input, Activities, Output. Next to the practice descriptions, there are web pages explaining the principles, key concepts, roles and templates. These is also a quick reference guide and an "about" section that explains the background and future plans.

The current version of RCDA is 1.1. It was reviewed and ratified by an international panel of representatives from every major company cluster and country. It is embedded into the company's business operating model as the recommended solution architecture approach.

The RCDA approach underpins the company's internal training program for solution architects. The core of this program is the Solution Architecture Practitioner Course. In 2010 and 2011, a total of 159 architects were trained in RCDA.

9.2.4 Structure of RCDA

As described in Chapter 6, we started out to create a generic architecting process in 2006. We identified a number of business goals and usage scenarios to scope the process (§6.2.2), and documented requirements that the process we were creating had to fulfill, based on the business goals (Table 6.1)and the CMMI maturity level 3 objective (Table 6.3). Once the requirements were clear, it took us about a month to write a 60 page draft process description – and then we got stuck. It turned out that the scalability and flexibility requirements (*rq.scalable* and *rq.generic*) were too much to be accommodated by a single process description. [Kazman et al., 2006] mentions "component techniques" for architecture analysis and design that can be "combined in countless ways to create needs-specific methods in an agile way"; we had harvested a number of such techniques, but a traditional process description did not allow us the agility required. We put the issue aside until we could find a way to resolve it.

We found a solution in 2008, when we came across Ivar Jacobson's practices approach [Jacobson et al., 2007]. Jacobson identifies a number of problems with traditional process descriptions, that touch the core of our issues in designing a generic architecture process:

Problem of Completeness "By striving for completeness, the processes end up as brittle, all-or-nothing propositions." We had tried to construct a complete architecting process for all possible business scenarios, and ended up with a document that made it hard for our architects to identify the parts and techniques that would add value to their specific situations. We also had many pieces of guidance that we wanted to share with our architects, but which were waiting for the process description to be complete before they could be released.

Problem of Adopting a Complete Process "Each [...] team has its own way - of - working (explicit or tacit), changing everything is silly, changing one thing may be smart." We wanted our architects to improve their existing processes to improve their architecting practices, rather than completely replace them with a heavy new architecting process.

Problem of Acquiring Knowledge "People don't read process manuals or language specifications, they want to apply processes not read about them." We realized our architects didn't need a 60 page detailed process description: they needed easily digestible, bite-sized pieces of guidance that would help them deal with their specific problems in their specific contexts.

Jacobson introduces an alternative to the process description: the *practice*, a "way to systematically and verifiably address a particular aspect of a problem." We decided to adopt this alternative for our solution architecting approach. The key aspects of the practice approach we adopted for RCDA are:

- Practices describe a way to characterize a problem and a way to address it.
- Practices can be picked and applied individually or in combination with each other to fit a particular situation.

The practices approach helped us address a number of challenges that we had run into when writing a traditional process description:

- It allowed us to disseminate important guidance without first having to document a complete, extensive process, solving the problem of completeness.
- It allowed architects to easily find relevant guidance without being forced to read a whole process description, partly solving the problem of acquiring knowledge.
- It allowed architects to apply individual practices in digestible bites, and change only those aspects of their way of working that would add value, solving the problem of adopting a complete process.

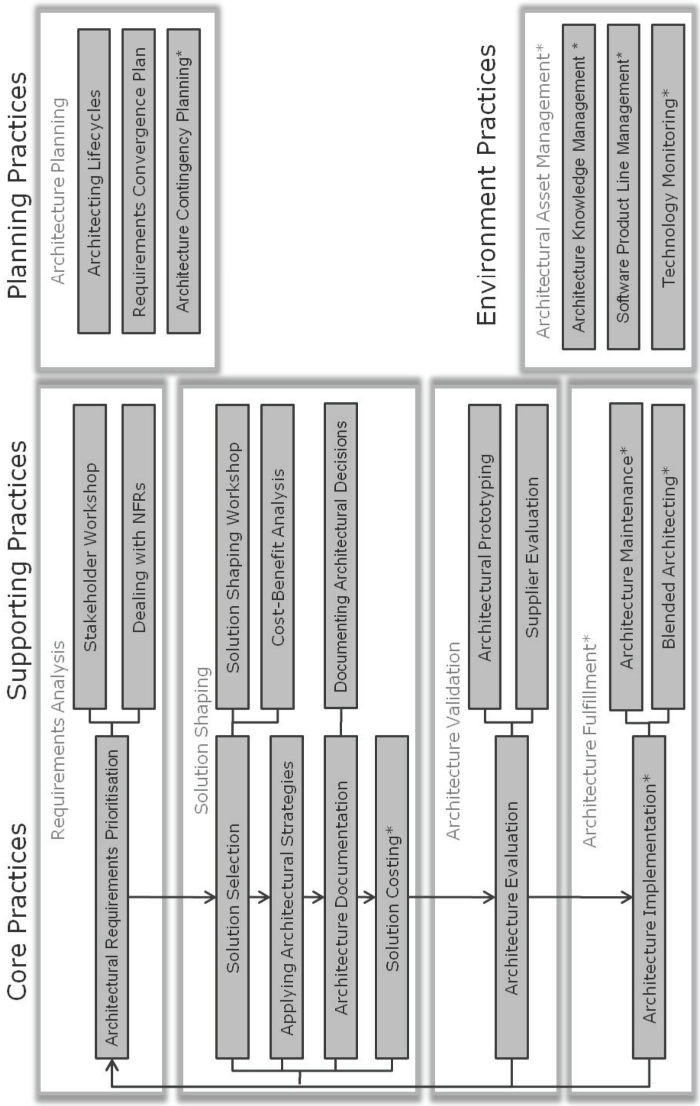


Figure 9.1: RCDA Practices organized by practice set and practice category.

Two years after adopting the idea of working with practices, we rolled out the result: the first version of our solution architecture approach, called Risk- and Cost Driven Architecture (RCDA). In 2011, we rolled out RCDA 1.1, the current version. RCDA 1.1 consists of 14 practices.

Architects are not normally expected to use all of the practices all of the time: they need to identify the best fit practices for their specific context. In order to help architects select and navigate through the practices of interest to them, we created a number of dimensions to structure the collection of practices:

Practice sets group practices into related tasks with similar objectives, as described in §9.2.1

Architecting lifecycles are typical real-life architecting scenarios, indicating which practices are used when; they are documented in the Architecting Lifecycles practice

Practice categories divide practices into four categories:

- **Core practices** The downside of moving from a traditional process description to a practice approach was that a set of practices by itself cannot fulfill the CMMI compliance requirement for a defined process (*rq.cmmi.gen* in Table 6.3). We resolved this by chaining the seven RCDA core practices together to form a core architecting process that should be followed in every reasonably complex project. The core process is one of the scenarios documented in the Architecting Lifecycles practice. Core practices may refer to supporting practices for (optional) additional guidance. The core practices embody the principles of RCDA.
- **Supporting practices** Supporting practices provide additional guidance on good architecture in a project, product or bid context. This category contains all non-core practices in the solution domain, except planning practices.
- **Planning practices** Planning practices help the architect and project / bid manager to plan architecture activities. They are all the practices in the Architecture Planning practice set.
- **Environment practices** Environment practices are architecture practices in a bid, project or product's environment that provide and consume artifacts of the solution domain, and in general impact the solution architecting, but are not solely directed at one solution. Thus, environment practices are about architecting across multiple individual solutions. At the moment the only environment practice set in RCDA is Architectural Asset Management; in

the future, additional environment practice sets may be added, related to e.g. architectural quality monitoring.

These structures are visualized in Fig. 9.1. The figure shows 22 practices: 14 that are part of RCDA 1.1, and 8 that have been identified for future versions. The 8 future practices are marked with an asterisk. In the figure, the 7 core practices are chained together by arrows to symbolize the core architecting process that they form.

9.3 Impact Survey

In October 2011, all Logica architects that were trained in RCDA were surveyed. The objective of the survey was to assess the impact of RCDA and its training on the work of the architects. In addition to the survey itself, we organized an expert workshop; a guided discussion with a select group of RCDA trained architecture experts. The workshop was held after the survey, and its purpose was to enhance the initial quantitative analysis results with qualitative knowledge from practicing architects.

9.3.1 Survey description

At the time of the survey, 159 people were registered as having received RCDA training. All of these registered trainees received an invitation by e-mail to participate in the survey. After two weeks, 32 (20%) had completed the survey, and the survey was closed.

The survey consisted of three sections:

Section A General questions about the trainees' activities after the training.
Section B Specific questions asking the respondents about the impact and frequency of use of the guidance in RCDA.
Section C Questions asking respondents whether they agreed with statements on the overall effectiveness of RCDA.

In order to measure at the level of individual pieces of guidance in RCDA, we codified the most important guidance: Table 9.1 lists the practices in RCDA 1.1. We have distilled one or more key guidance elements from each practice. Every guidance element has been given a code tag, which is used to identify the guidance element in the survey.

In Section B, respondents were asked to indicate on a Likert scale how often they had applied each guidance element in Table 9.1 both before and after receiving the training:

Table 9.1: RCDA practices and key guidance elements.

Architectural Requirements Prioritization	
ARP.rc	Identify architectural requirements by risk and cost impact.
ARP.sc	Express architectural requirements in scenarios.
ARP.wf	The architect's daily workflow is addressing architectural concerns, prioritized by risk and cost impact.
Dealing with Non-Functional Requirements	
NFR.hd	Look for hidden NFRs, since they are often crucial for acceptability even when not documented.
NFR.vf	Verify as early as possible that the architectural design will fulfill the NFRs.
NFR.cm	Don't commit to quantified NFRs until you have proof of feasibility.
NFR.dc	Document how NFRs are dealt with as proof of professional behavior.
Stakeholder Workshop	
SW.ws	Gather stakeholders in a workshop to elicit architectural requirements as early as possible.
Solution Selection	
SS.ev	Decide after evaluating multiple alternative solutions against objective criteria.
Solution Shaping Workshop	
SSW.ws	At the start of a bid or project, gather all delivery stakeholders in a solution shaping workshop to agree on a candidate solution.
Cost-Benefit Analysis	
CBA.qf	Quantify the impact of architectural strategies on a solution's quality attributes in terms of stakeholder value.
Applying Architectural Strategies	
AAS.dc	Document the impact of selected architectural strategies in terms of elements, interfaces and refined requirements.
AAS.rp	After applying strategies, re-prioritize architectural concerns.
Architecture Documentation	
AD.sa	Use a stakeholder analysis to determine to whom the documentation is communicating.
AD.vp	Use viewpoints to show stakeholders how their concerns are addressed.
Documenting Architectural Decisions	
DAD.rd	Use a formal Record of Decision to document key architectural decisions.
DAD.rg	Use an architectural concern and decision register to prioritize and order the architecture work.
DAD.pr	Communicate progress and status of architecture work in terms of architectural concerns and decisions.
Architecture Evaluation	
AE.ev	At key points in an architecture's lifecycle, perform an objective evaluation and analysis of how the architecture fulfills its stakeholders' needs.
Architectural Prototyping	
AP.pr	When necessary, build a prototype or proof-of-concept to verify that architectural strategies fulfill the requirements.
AP.oc	Prepare to deal with any outcome of the PoC, including a contingency plan in case of a negative result.
Supplier Evaluation	
SE.ev	When third parties provide critical components of our architectural solution, evaluate the supplier to identify potential commercial, technical, PR, quality and service related risks.
Requirements Convergence Planning	
RCP.pl	In case of unfeasible or unclear NFRs, agree a plan with the client that describes how to converge on acceptance criteria, representing a balance of affordability between cost and benefits.
Architecture Lifecycles	
AL.cp	RCDA Core Process.
AL.rf	Respond to RfP.
AL.ru	RUP software development.

- never
- once or twice
- regularly (whenever an applicable situation occurs)
- every day (part of my daily work routine)

Respondents were also asked to indicate the impact of the guidance by choosing between:

- n/a (never applied the guidance)
- counter-productive (I tried applying the guidance, but it made matters worse)
- neutral / mixed results
- noticeable improvement (compared to acting without this guidance)
- critical improvement (without applying this guidance, project would have failed or bid would have been lost)

9.3.2 Survey results

Some general statistics to start with:

- Elapsed time from the training to the survey was between 3 and 23 months, with an average of 9 months, meaning all respondents had time to internalize and apply the material.
- Average time spent in architect roles was 45%. 6 respondents spent less than 10% in architect roles, of which 5 indicated they had not been in any architect role. 12 respondents spent 75% or more of their time in architecting roles.
- 13 respondents (40%) were the lead architect on the majority of their assignments, meaning they were responsible for architectural decisions.

Fig. 9.2 shows the responses to the three general statements about the effectiveness of RCDA (Section C):

imp_gen *In general, my effectiveness as an architect has improved after being trained in RCDA.*
imp_com_stkh *RCDA helps me to communicate with stakeholders more effectively.*
imp_prio *RCDA helps me to better focus and prioritize my work as an architect.*

Overall, the majority of the responding architects agree that their effectiveness has increased, and about half agree that RCDA has brought them the benefits of the risk- and cost driven approach described in §8.1. Less then 15% disagree with any of the statements.

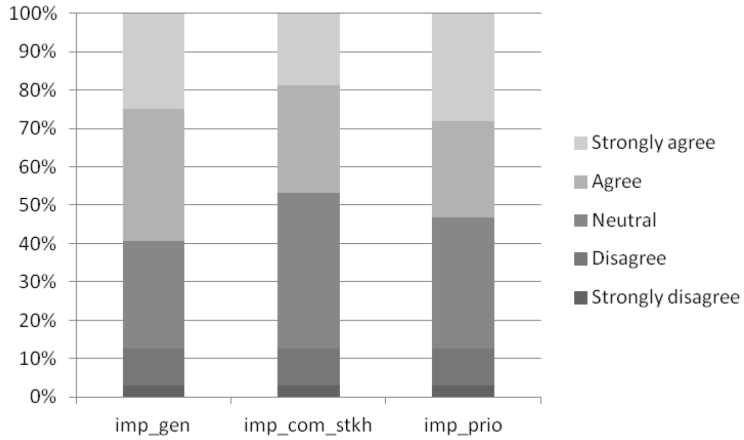


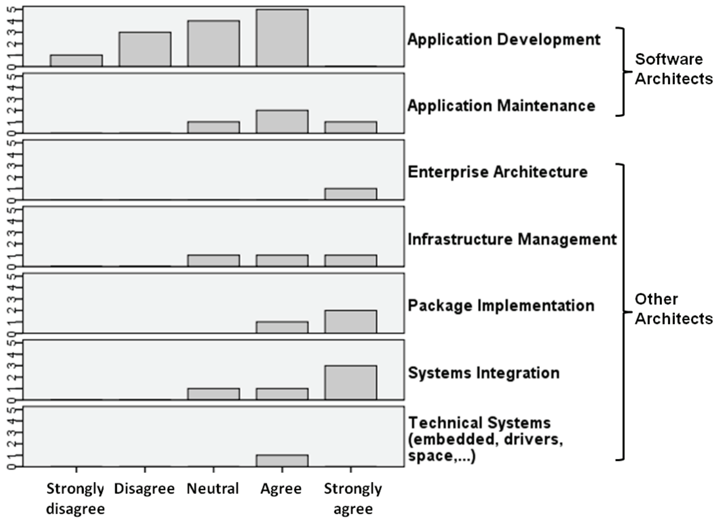
Figure 9.2: Three statements on effectiveness of RCDA

In Fig. 9.3(a), we see how the responses to imp_gen are divided over the various architecture "genres". The figure confirms that the effectiveness of the risk- and cost driven view extends beyond software architecture into the wider domain of solution architecture. In fact, the only disagreement comes from the software architects in the application development domain. The fact that only some *software* architects disagree with the effectiveness of RCDA seems remarkable, since RCDA is mostly based on ideas from the software architecture community. We discussed this paradox in the expert workshop; the most likely explanation seems to be that some software architects may have found RCDA less value-adding than other architects, because it is partly based on ideas that were already familiar to them before receiving the training.

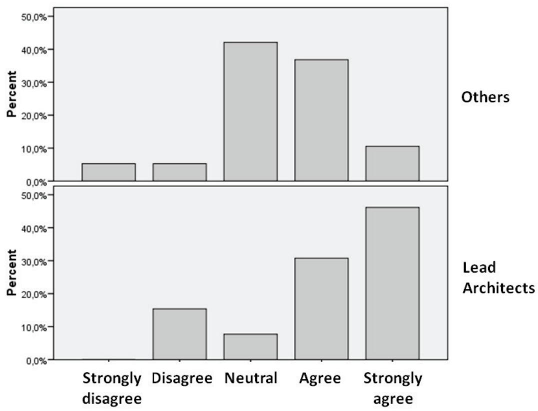
Fig. 9.3(b) shows that those who have been active in lead architect roles have stronger opinions, and in general are more positive about RCDA effectiveness than those who have not been in lead architect roles. This visual impression is confirmed by statistical analysis, which shows that the "lead architect" responses are significantly correlated to the "general effectiveness" agreement responses (Spearman's ρ correlation coefficient of 0.34, 1-tailed significance at the 0.05 level).

RCDA principles

We asked the architects about the frequency with which they applied the general RCDA principles explained in §9.2.2, and the impact. Table 9.2 shows the results. Column



(a) By architecture genre



(b) Lead architects vs others

Figure 9.3: Agreement with “In general, my effectiveness as an architect has improved after being trained in RCDA.”

Table 9.2: RCDA Principles: frequency applied and impact

Principle	Applied before	Applied after	Significant impact
Cost and risks drive architecture*	16%±6%	34% ±8%	86%±7%
Architecture should be minimal*	34%±8%	53%±8%	88%±6%
Architecture as both blueprint and design decisions	34%±8%	53%±8%	100%
Solution architect as design authority*	25%±7%	44%±8%	86%±7%

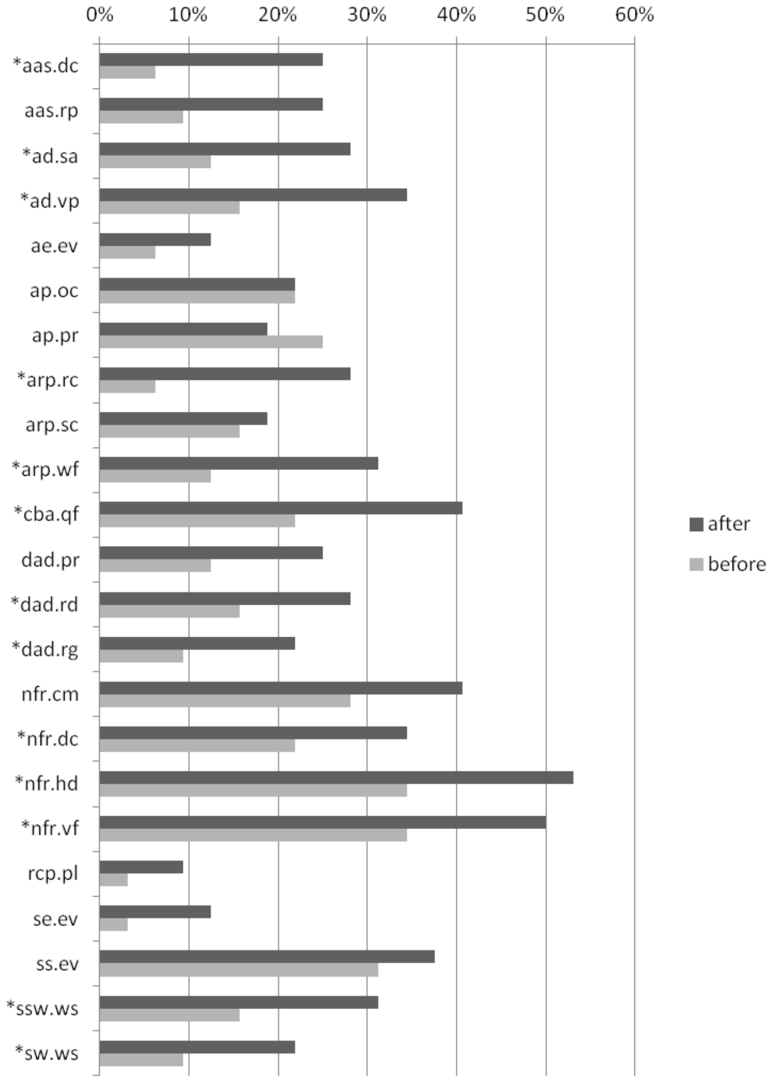
* Increase in application frequency significant at 0.05 level

“Applied before” and “Applied after” shows the percentage of respondents who applied the principle before and after receiving RCDA training; “Impact” shows the percentage of respondents who reported significant impact. Standard errors in the percentages are indicated in the table. The table shows that the number of architects applying the principles has increased considerably after the training, and that all four of the principles have significant impact when applied. A paired-sample T-test between the “frequency applied before training” and “frequency applied after training” shows that three out of the four principles have been applied significantly more after receiving the training: the increase in application of the “architecture as both blueprint and design decisions” principle is not significant at the 0.05 level, the other three are significant and are indicated with an asterisk.

RCDA practices

Fig. 9.4 shows the percentage of respondents indicating they have applied the key guidance elements of the RCDA practices listed in Table 9.1, both before and after receiving the training. All guidance elements show an increased number of respondents applying it after the training, with the exception of Architectural Prototyping. The guidance elements for which the increase is significant as calculated by a paired-sample T-test are indicated with an asterisk. The Architectural Prototyping practice was already applied before the training by 25% of respondents.

The percentage of trainees who applied the guidance after training is below 60% for all guidance. This may seem low; additional light is shed on this if we compare the percentages for lead architects versus other respondents, as visualized in Fig. 9.5. We see that more lead architects than others are applying the guidance, and almost half of the guidance elements are applied by the majority of the lead architects. Analysis shows that the “lead architect” responses are significantly correlated to the “frequency applied after training” responses for all guidance elements except ap.pr and nfr.cm



* Increase in application frequency significant at 0.05 level

Figure 9.4: RCDA practices: respondents applying guidance before and after training (abbreviations on p.157)

(positive correlation coefficient of 0.3 or more, 1-tailed significance at the 0.05 level, using Spearman's ρ). Just like in the case of "general effectiveness" above, we see a correlation between the solution architect's position of authority and responsibility (lead architect) and the frequency of applying RCDA guidance.

Fig. 9.6 shows the key guidance elements of the RCDA practices. The figure visualizes the impact of the practices and their training. The horizontal axis represents the percentage of respondents reporting an increased frequency of applying the guidance after the training. The vertical axis represents the percentage of respondents reporting that the guidance has had significant positive impact in their projects. The first observation is that none of the practices is reported to have increased application by more than 50% of respondents (which is in line with Fig. 9.4). On the other hand, all practices are reported to have significant impact by over 50%.

We have clustered the guidance elements and separated the clusters by gray lines.

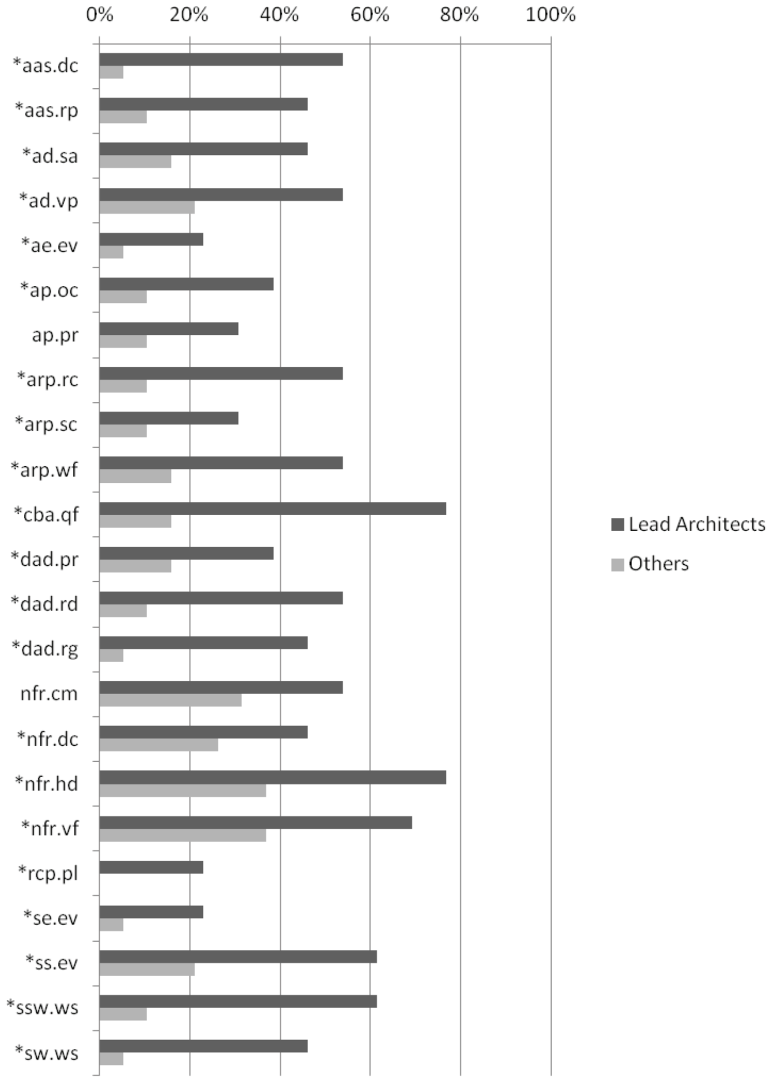
- In the center, we see 8 guidance elements that all have around average characteristics: increased application by about 25%, and significant impact reported by about 75% of respondents. This cluster includes Architecture Evaluation, Cost-Benefit Analysis, Applying Architectural Strategies, Stakeholder Workshop and guidance elements from three other practices.

- In the top left cluster, we see both the Architectural Prototyping and the Requirements Convergence Plan practices. Apparently, the use of these practices has not increased very much, even though their impact is relatively high. Part of the explanation for this could be that both of these practices require considerable resources and time to implement.

- In the top right cluster are the "stars" of the training: the guidance elements that have the highest impact in terms of both usage and effectiveness. This cluster contains the Solution Shaping Workshop, most of the guidance from Dealing with NFRs and Documenting Architectural Decisions, and the use of viewpoints in Architectural Documentation.

- The bottom left cluster has Supplier Evaluation and Solution Selection, two practices that require relatively formal evaluations to be performed. These are perceived as relatively low-impact practices by the architects.

- The bottom right cluster contains all guidance in the Architectural Requirements Prioritization practice. The training appears to be relatively successful in making architects consciously prioritize their requirements; on the other hand, "only" around 70% of the architects report that this has significant impact. A possible explanation came out of the post survey expert workshop: because requirements



* Correlation with lead architect response significant at 0.05 level

Figure 9.5: RCDA practices: lead architects vs others applying guidance after training (abbreviations on p.157)

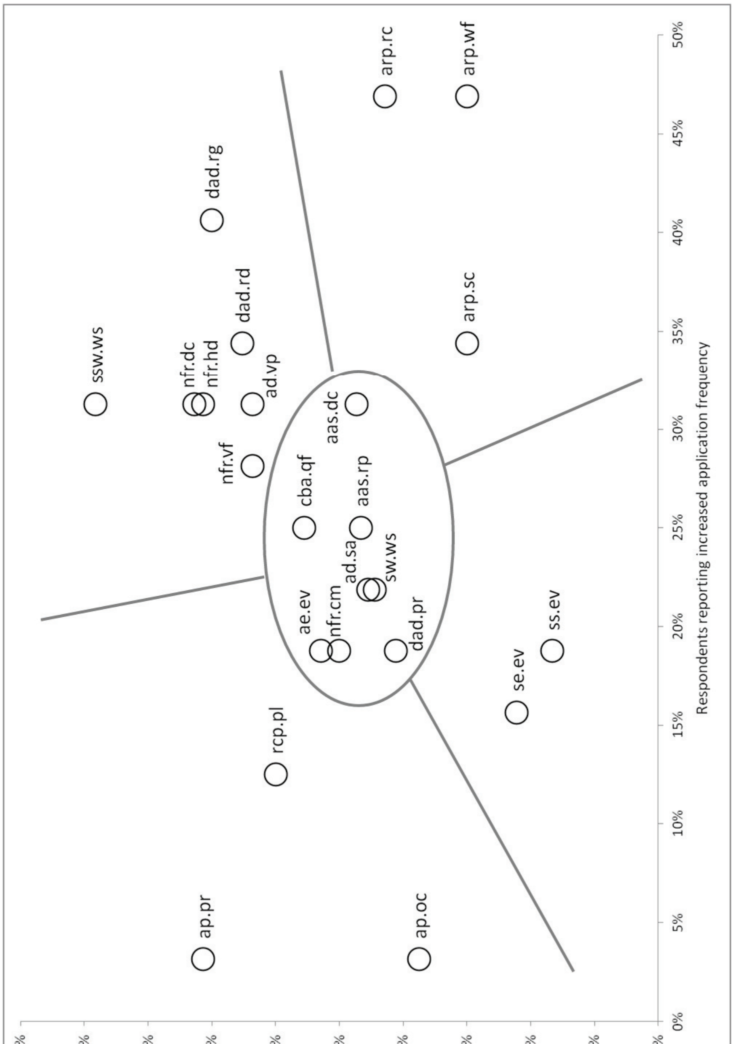


Figure 9.6: RCDA practices: impact vs. increased application after training (abbreviations on p.157)

prioritization happens relatively early in the chain of architecting activities, its impact is perceived as more indirect than that of other practices.

Due to the limited size of the population sample, the standard error in the placement of the guidance elements is in the same order of magnitude as the separation between the clusters: it ranges between 6% and 8% on the x-axis and between 5% and 11% on the y-axis. This means that the clustering should be considered tentative at this point.

Because we found significant differences between the responses of lead architects versus other architects for the frequency and effectiveness questions, we also looked for such differences in the impact responses for the individual guidance elements (the Y-axis in Fig. 9.6). We found only one: all 10 lead architects (100%) who applied architectural prototyping (ap.pr) reported significant impact, while of the 11 other architects who applied ap.pr, only 8 (73%) reported significant impact. Analysis shows that the "lead architect" response is significantly correlated to the "impact" response for ap.pr (positive correlation coefficient of 0.435, 2-tailed significance at the 0.05 level, using Spearman's ρ).

9.4 Conclusions and Discussion

The results of the analysis above indicate that for the majority of trainees, RCDA 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 RCDA training is effective in increasing the application of the principles and practices.

The survey validates RCDA as an approach that improves the solution architecting practice in Logica.

The sparse application and relatively low appreciation of formal evaluation practices like se.ev and ss.ev (Fig. 9.6) is in line with findings by e.g. [Clerc et al., 2007], which reports that "methods and techniques to validate the architecture . . . are not embedded within the mindset of architects." Another finding from the [Clerc et al., 2007] survey is that "the architects mindset lacks focus on reflections on those decisions as building blocks for software architectures"; the success of the RCDA Documenting Architectural Decision practice in terms of both frequency of use and perceived impact indicates that this lack of focus can be remedied by e.g. the RCDA training.

In the after-survey expert workshop, we discussed some of the more remarkable results of the survey with a selected group of senior architects who were familiar with RCDA and its training. The results of this workshop are discussed below.

Architectural prototyping

The prototyping guidance element "When necessary, build a prototype or proof-of-concept to verify that architectural strategies fulfill the requirements" (ap.pr) jumps out in a number of results:

O1 ap.pr is the only guidance element less frequently applied *after* training than *before* training.
O2 ap.pr is one of only two guidance elements whose application frequency after training is *not* significantly correlated with the lead architect role.
O3 ap.pr is the only guidance element whose impact response *is* significantly correlated with the lead architect role.

The workshop participants produced two possible explanations for the decrease in application after training (O1):

E1 RCDA focuses architects' attention on other activities, making prototyping a relatively lower priority.
E2 The time passed after the training is less than before the training, the architects simply didn't have enough time after the training to apply ap.pr, which require considerable resources and time to implement.

Taking all three observations together, the workshop agreed that E2 is the more likely explanation, since E2 helps explain O2, and E1 does not match with O3. E2 also is a good explanation for the fact that in Fig. 9.6, ap.pr is in the top left cluster with requirements convergence planning, another practice that requires significant planning and use of resources.

Lead architect

The trainees that were in lead architect roles after the training have given significantly more positive responses to most of the questions related to application frequency and overall effectiveness. The post survey expert workshop generated a number of explanations for this phenomenon:

L1 Those in the lead determine which practices will be followed, so they can choose to apply RCDA practices, while those not in the lead have to follow practices dictated by others.
L2 The use of a common approach like RCDA is much more important for those in the lead, since it smooths communication with stakeholders like reviewers and managers – with whom those not in leading roles have less dealings.

L3 RCDA promotes a position of authority for architects (the fourth principle in §9.2.2), so that those who apply RCDA tend to take more ownership and responsibility, which puts them in leadership positions.

The data set did not provide any means to confirm or reject any of the three explanations: they may well all be valid, and reinforce each other's impact.

9.4.1 Threats to validity

In a survey like this, there is a potential selection bias due to possible increased interest in the survey by those who have had positive experiences with the subject of the survey. In order to assess the magnitude of this bias, we picked 10 trainees at random from those who had not responded to the survey. We called these 10 trainees and asked for their reasons for not responding. The 10 gave the following answers:

- 1 indicated that he had not been able to apply the material.
- 1 indicated that he had not followed the training and was on the list of trainees by mistake.
- 8 indicated that they had been too busy to respond to such an extensive survey.

This seems to indicate that the proportion of trainees who had not been able to apply any of the material is roughly the same for those who responded to the survey as those who did not respond, implying there is no significant selection bias.

Another threat is in the survey population: all results are subject to the perception of the architects. A good example is the architects' subjective evaluation of the impact of the practices. The post survey expert workshop noticed that practices that reenforce the importance of the architects and their skills tend to get higher impact ratings. Examples of this phenomenon are:

- The highest rated impact is for the Solution Shaping Workshop, which puts the solution architect in a key position right at the beginning of the solution shaping process.

- Formal evaluation practices like se.ev and ss.ev are sometimes seen as reducing the architect's importance, since they require the architect to justify their decisions; they get a relatively low impact rating.

- Architectural Requirements Prioritization directs the architect in his priorities - and gets a much lower impact rating than the related Documenting Architectural Decisions, which positions the architect as an ("important") decision maker.

The only way to assess the seriousness of this bias is to measure the impact of the practices in ways that exclude the architect's opinion.

Like with the other surveys in this thesis, the results are influenced by cultural aspects of both the Logica company and the Netherlands location, and should be used with care when applied outside of these boundaries.