Utrecht University

Master Thesis

Department of Information and Computing Sciences

Capturing Software Architecture Design Rationale Through Decision Stories

Supervisors:
dr. ir. J.M.E.M van der Werf
Prof. dr. F.J. Bex

Author:
Roel de Gouw
4166418
r.j.m.degouw@students.uu.nl

External Supervisors:
dr. Eltjo Poort
André Versteeg

December, 2019

Abstract

Within Software Architecture, there is no universal template for capturing design rationale in (architectural) decisions, many templates exist within the literature and industry. Research has shown there is a growing need for documenting design rationale. However, architects do not always document their rationale due to time and budget constraints. This research proposes a decision template that stimulates architect to capture design rationale. Building on a decision template from existing work, it answers the research question: how can design rationale of decisions be captured effectively? Through a literature review and expert interviews, concrete requirements for decision documentation are derived. The Decision Story, a lightweight template to document decisions is designed. The template is validated through case studies, expert interviews, and a focus group.

Results show the Decision Story captures the rationale aspects architects and the literature deem most important, is easy in its use due to being compact and intuitive, can be used in different visualisations, is easily modified to specific projects, fulfils the needs of stakeholders, is able to expose gaps in the decision process of architect, and leads to a higher quality decision. Further research is needed to test the effectiveness of the Decision Story in practice.

Acknowledgements

I couldn't have written this thesis without the help of many people. I would like to acknowledge some of them here.

Thank you Jan Martijn van der Werf for your constant and involved supervision. Thank you for your patience and guidance throughout this thesis and enthusiasm for the topic.

I want to thank Floris Bex for his supervision and input during the thesis project.

Special thanks to Eltjo Poort, André Versteeg and Mathijs van Riel for giving me the opportunity to conduct my research at CGI. Thank you for your guidance, discussions and involvement during the writing of this thesis.

Many thanks to all the architects at CGI who participated and contributed to this research. Thank you for your time, input and insights during our interviews.

Finally I would like to thank my family and girlfriend for their love and support.

Table of Contents

Chapter 1

Introduction

In Software Architecture and Software Design, individuals do not reason much before making decisions (Tang & van Vliet, 2015). The authors state that software designers are content with a solution that is good enough. This behaviour is called satisficing and is a problem in *design reasoning*, as it typically leads to lower quality design decisions. In addition to satisficing, Tang (2011) identifies cognitive bias, illogical reasoning and low quality premises as other problems in design reasoning. This is because the decision making of designers is often a naturalistic process, where decisions are being made on intuition and gut feeling (Zannier, Chiasson, & Maurer, 2007). However, in other domains, intuition can be used in the design process to come to more original ideas as well as faster and better decision making (Pretorius, Razavian, Eling, & Langerak, 2018). Design reasoning is important within Software Architecture because Software and Solution architects make architectural decisions to tackle stakeholders concerns based on incomplete information. Their reasoning for these decisions can be very important later on. Especially early on in a software project, these decisions have far reaching consequences across the entire software lifespan.

Several methods for improving design reasoning exist. Tang, Lago, et al. (2010) described several design reasoning techniques: the articulation of (1) context, (2) problems and (3) solutions, carrying out trade-offs (4) and the identification of (5) assumptions and (6) risks. Using these techniques, Schriek, van der Werf, Tang, and Bex (2016) created a card game to enable design reasoning through card play labelled with design reasoning techniques. Results from a student experiment show the group using these cards produced significantly more design rationale than the control group. De Jong (2017) developed the Rationale Capture Cycle (RCC). The RCC was tested on practising software architects and has shown to stimulate and improve design reasoning.

Design rationale captures the reasoning and justification for design decisions.

Example justifications include (i) considered alternatives (ii) evaluated trade-offs and (iii) argumentation for the decision (Gruber & Russell, 1991; Lee, 1997; Tang, Babar, Gorton, & Han, 2006). There is a growing need for documenting design rationale and architects do not always document their design reasoning (Bosch, 2004; Capilla, Jansen, Tang, Avgeriou, & Babar, 2016). This can be due to time and budget constraints (Conklin & Yakemovic, 1991; Tang et al., 2006; Falessi, Briand, Cantone, Capilla, & Kruchten, 2013), lack of standards and established process (Tang et al., 2006; Capilla et al., 2016), cost and benefit of documenting design rationale (Tang et al., 2006; Capilla et al., 2016; Falessi et al., 2013) and disruption of design flow and lack of understanding by stakeholders (Capilla et al., 2016). Furthermore, if rationale is not documented properly or not present at all, (i) architects tend to forget the rationale for their own design decisions and, (ii) architects that need to modify designs by other architects do not understand the design (Tang et al., 2006).

Not only software architects and designers are impacted by (architectural) decisions, but other stakeholders as well. Capilla et al. (2016) identify end-users, business managers and software maintainers as having different needs in regards to architectural knowledge. Decisions can be written down with the help of decision templates. A number of decision templates to capture decisions exist. Zimmermann, Wegmann, Koziolek, and Goldschmidt (2015) compared several industry templates on content, element count and size. An analysis on how these templates are performing in practice is currently not present as far as we know.

Different development approaches lead to different decision making. For example, Agile and Lean development approaches can lead to faster decision making. In these approaches, decisions are often made in groups during sprints. This leads to obstacles such as lack of decision ownership or unwillingness to commit to decisions (Capilla et al., 2016). A decision template needs to be suitable to these differences in development approaches.

A template for capturing design rationale which has been used successfully by the industry is the *Y-statement*, first published by (Zdun, Capilla, Tran, & Zimmermann, 2013). When a decision has been made, its outcome can be described in one sentence by filling in the different fields of the Y-statement. The template can be viewed in Figure 1.1. The Y-statement evolved from Fairbank's (2010) notation for rational architectural decisions.

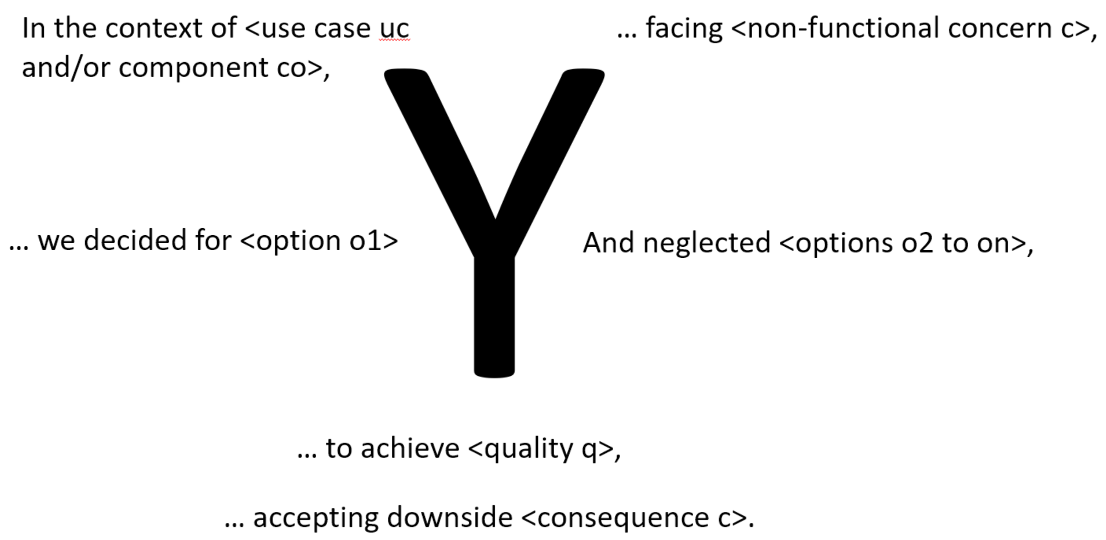

In the context of <use case uc and/or component co>,
… facing <non-functional concern c>,
… we decided for <option o1>
And neglected <options o2 to on>,
… to achieve <quality q>,
… accepting downside <consequence c>.

Figure 1.1: Y-statement template (from Zimmermann (2014))

Research Objective

The Y-statement template in Figure 1.1 is used as the starting point for this research as it has been used successfully by the industry. The objective of this research is to study how decisions are written down and how to effectively capture design rationale of architects in a decision template.

Chapter 2

Research Approach

This research will follow the design science methodology by Wieringa (2014). Design science iterates over: "designing an artifact that improves something for stakeholders and empirically investigating the performance of that artifact in a context". The artifact in this research is a decision template. Wieringa (2014) notes that since the artifact is designed for use in its context, it should be studied in this context. Thus we aim to apply a decision template within the domain of Software Architecture to study its effectiveness and improve upon it where applicable. To address the goal of this research, the following main research question is formulated:

MRQ: How can design rationale of decisions be captured effectively?

In this context effective means efficient (as a byproduct, little overhead, non intrusive), concisely (not more elaborate than needed) and valuable (fulfils the needs of its stakeholders, leads to better decisions). This main research question is further divided into sub research questions:

SRQ 1: What decision documentation templates exist?

SRQ 2: What are requirements to document decisions effectively?

SRQ 3: What are templates for documenting decisions that stimulate architects to capture rationale?

SRQ 4: How effective is this designed template?

The research questions are answered through a literature review, experimentation, case studies and interviews. These research methods are now discussed in more detail.

Literature Review

A literature review is conducted with the aim of gathering relevant information about the following topics:

1. Software Architecture
2. Design Decisions
3. Design Rationale
4. Design Reasoning

Through analysis of the literature, requirements for decision documentation templates are distilled. Furthermore, the current reasoning and rationale capturing methods from the literature are identified and compared against each other. The goal is to see the current state of decision documentation and how it can be improved.

The starting point for literature review will be the theses of previous Master students (Schriek, 2016; De Jong, 2017; Methorst, 2019) at Utrecht University who have done research in the same domain as well as the paper where the Y-statement is introduced by Zdun et al. (2013). To gain a more general domain understanding the works by Clements et al. (2002); Bass, Clements, and Kazman (2003); Fairbanks (2010); Rozanski and Woods (2011) and Poort (2014) will be used. From here, forward and backward snowballing will be used as a first search strategy. Database searches in Google Scholar will be used to look for additional literature.

Interviews

Several interviews and case studies are performed at an IT services company. These case studies take place at CGI[1], a Canadian based company with several locations across the Netherlands. The stakeholders that will be interviewed are Solution Architects, Software Architects and Software Engineers.

Several interviews are done at the start of the research project. The goal of these interviews is to gather information on:

- Gather requirements for decision documentation;
- How architects are currently affected by decisions;
- How they document decisions;
- Their needs regarding these decisions;

[1]www.cgi.com

- Problems in the current way of working.

Another round of interviews is done toward the end of the research project. The goal of these interviews is:

- Evaluate different versions of the designed decision template;
- Discuss experience using the designed decision template;
- Validate definitions of the designed decision template.

The interview protocols are set up based on the literature review and discussions with expert architects. This protocol was tested on a person familiar with architectural decisions to identify any gaps or shortcomings in the protocol and improved. The interviews will be semi-structured to define the topics of the conversation beforehand. The focus of the interviews will be on the personal experience of the stakeholders as well as their expert opinion on the decision making process in general. Participants were identified through a call for participants email that was issued through the CGI architecture practice mailing list, as well as through referrals by CGI employees. Convenience sampling is used to identify interviewees, meaning all interviewees were employees of CGI or were referred to by CGI employees.

Case Studies

Several case studies at projects within CGI were performed. The goal of these case studies is to:

- Test designed decision templates in practice;
- Identify shortcomings and potential improvements to designed decision templates;
- Gather feedback on what stakeholders think of the designed decision template.

At first this approach is action research oriented. Action research is where the researcher influences a project actively. Easterbrook, Singer, Storey, and Damian (2008) argue that a lot of software engineering research is actually action research in disguise, as in many cases new ideas are developed and iterated upon by trying them out on real projects. This will be beneficial to the research as many versions of the designed decision templates can be tested to fine tune the template quicker through constant feedback. Once the template settles down to one that is satisfactory, the research is more case study oriented to gather data in a more objective manner over a longer period of time.

Experimentation

The designed decision template is tested through a student experiment at Utrecht University during the course Software Architecture. In groups, students are tasked with the Irvine Traffic Simulator experiment (Petre & Van Der Hoek, 2013) and asked to design an architecture for a traffic simulator system. Participants are asked to document their decisions using the designed template. Afterwards, the participants receive a questionnaire asking about their experience using the template.

Focus Group

A focus group was held towards the end of the research project. The aim of the focus group was to brainstorm about which business goals the designed decision template could support, what obstacles prevented adoption and implementation of the template and solutions to these obstacles. A focus group is a research technique that collects data through group interaction on a predefined topic set by the researcher. This group interaction can produce data and insights that are not commonly found in one-on-one interviews (Morgan, 1996). In this study, the focus group is designed to add richness and depth to the interviews and case studies performed earlier in the research project and is thus used at the end of the project. Participants were drafted from the architects interviewed earlier in the research project, this ensured that all participants were familiar with the designed decision template.

Chapter 3

Design Rationale in Software Architecture Literature

This section contains a review of the relevant literature to gain domain knowledge understanding. The structure (including a small summary) is as follows:

1. **Software Architecture** is the high level structure of software. Software architects make decisions to address architectural concerns. These are represented in views and viewpoints to communicate the architecture to stakeholders. Architectural knowledge captures the architecture design and why the architecture is the way it is. This architectural knowledge can be tacit, documented or formalised. Architectural decisions can be prioritised using risk and costs, this leads to decisions with the highest impact being made first.

2. **Design Decisions** describe why a choice was made from several alternatives to address a concern or realise requirements. Stakeholders of decisions are: end-users, business managers, software maintainers & testers and software architects & designers. They all have different needs regarding decisions. Decisions vary in abstraction (architectural, design, implementation) and the decision scope changes alongside realisation of the system. Decisions can be documented from comments in code to external repositories. Value based documentation is only capturing the architectural knowledge needed for future activities. A comparison of different decision templates show most templates are very elaborate and compromise multiple pages.

3. **Design Rationale** captures the design reasoning and justification for design decisions. Rationale can be used among other things to check if a solution satisfies stakeholder concerns, communicate and teach a design to others, and assist architects to make sure their decisions are explicit. There is a growing need

for documenting rationale. Unfortunately, architects do not always document their rationale due to time & budget constraints. This is a problem as architects forget their own design rationale over time and architects do not understand designs of other architects without rationale present. Better rationale capture can be realised by convincing stakeholders of the importance of documentation, a systematic process to capturing architectural knowledge, only documenting that which is relevant and embedding rationale capture in current ways of working.

4. **Design Reasoning** is the activity where decision makers reason about a design problem and possible ideas & solutions are examined. This process has some potential problems: decision makers do not always make rational choices as individuals are subject to cognitive biases and often rely on intuition. Furthermore, individuals are subjected to bounded rationality due to limited time, information and cognitive capacity. This leads to satisficing behaviour, where decision makers are content with solutions which are "good enough" but probably not optimal. Research has shown that this can lead to lower design quality. The majority (80-85%) of decisions are made in groups. Group decision making faces several issues: individuals not understanding goals, long time to reach agreement, power difference and Groupthink (group aims to find consensus over individual opinions). Design reasoning can be improved by reflective thinking and design reasoning techniques.

3.1 Software Architecture

Software Architecture is a way to address the growing complexity of large software systems. Several definitions exist which describe the scope of Software Architecture. Kruchten (1995) describes it as "Software architecture deals with the design and implementation of the high-level structure of the software. It is the result of assembling a certain number of architectural elements in some well-chosen forms to satisfy the major functionality and performance requirements of the system, as well as some other, non-functional requirements such as reliability, scalability, portability, and availability." More recently, the view has shifted to look at software architecture as the composition or culmination of a set of architectural design decisions (Jansen & Bosch, 2005). This means that Software Architecture is not just the models or high-level structure of the software, but also the decisions that lead to this high-level structure and the reasoning and rationale behind these decisions. Because of this shift, part of the research focus in Software Architecture has shifted to Software Architecture Knowledge Management (Babar, Dingsøyr, Lago, & Van Vliet, 2009).

The system that an architect designs has to address the concerns of multiple stakeholders such as users, business owners, operations personnel, etc (Bass et al., 2003). To address these concerns, architects create *views*. A view is "a representation of a set of system elements and relations associated with them" (Clements et al., 2002). As the system is often complex and difficult to understand, using views makes it possible to look at a smaller set of the architecture at a time to address specific stakeholder concerns. Some of these views are standardised in *viewpoints*, giving architects a common toolbox of patterns, templates and conventions with which to create views (Rozanski & Woods, 2011). This means every stakeholder can use this common language when communicating about the architecture.

By creating views and viewpoints the architecture can be easier understood, but it does not show why the architecture is the way it is. The "why" is captured in *architectural knowledge* (AK). AK consists of the design of the architecture together with the design decisions that lead to that design, including the context and assumptions where these decisions are based on (Kruchten, Lago, & Van Vliet, 2006). We distinguish three types of knowledge (Nonaka & Takeuchi, 1995): (i) tacit knowledge, residing in the heads of people, (ii) documented knowledge, meaning the knowledge is available somewhere physically or digitally, and (iii) formalised knowledge, where a process is in place on how to document the knowledge. Managing this knowledge is vital for software companies. Consider when an architect leaves a project and takes their AK with them, future architects will have difficulty understanding the "why" behind the system. By documenting and formalising useful tacit architectural knowledge for other people, the company becomes less reliant on the tacit knowledge of individual people.

Risk and Cost Driven Architecture (RCDA)

Poort (2014) argues architecture is a risk and cost management discipline. Viewing architecture with risk and costs helps architects prioritise their work and timing of their decisions. Design decisions are the main deliverable of an architect using RCDA. This lends itself well in the Agile methodology, where an architect produces a "continuous stream of architectural decisions" instead of a big up-front design (Poort, 2014).

RCDA is a collection of architecture practices. Rather than describing a process that is set in stone, these practices allow architects to apply architecture specifically for their project context. These practices are divided into (i) core practices, which are roughly applicable to most complex projects, and (ii) supporting practices, which are more applicable in specific contexts.

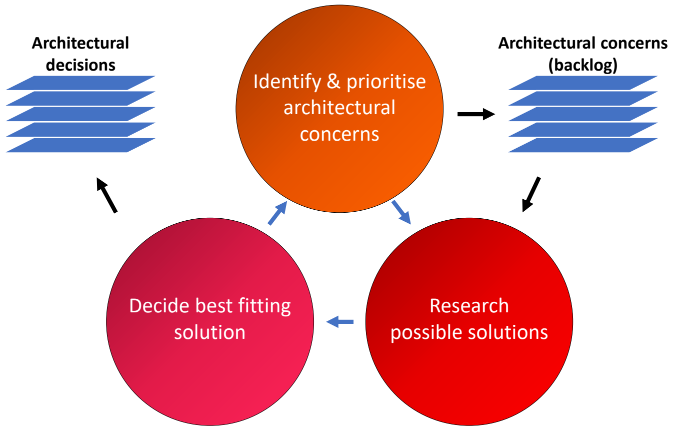

Architectural decisions
Identify & prioritise architectural concerns
Architectural concerns (backlog)
Decide best fitting solution
Research possible solutions

Figure 3.1: Architecting Microcycle (from Poort (2014))

The work flow of an architect applying RCDA can be described using the Architecting Microcycle in Figure 3.1. Poort (2014) argues that "Architects should focus their attention on decisions that have the highest impact on cost and risks of the solution". This is done by addressing architectural concerns. An architectural concern is an issue with a large impact on the solution that will be hard to change later on. These concerns are identified and prioritised by risk and cost (step 1), the concern with the highest impact is addressed by researching possible solutions (step 2) and then the best fitting solution is chosen (step 3). This process leads to a collection of architectural decisions and is repeated until all concerns are addressed. One thing to note is that every decision made has an impact on the remaining architectural concerns (e.g. some options may no longer be viable because of earlier decisions), and thus the cycle begins again at step 1.

Figure 3.2 shows how aspects of architectural decisions change over time. As the architect addresses the concerns with the highest impact first, decisions with the highest impact are made first but are also made with the least amount of available knowledge. As the first few high impact decisions are taken, the impact of subsequent decisions decreases while frequency of decisions increases. As the solution is being built, the available knowledge increases and the frequency of decisions decreases as the architecture is largely defined already.

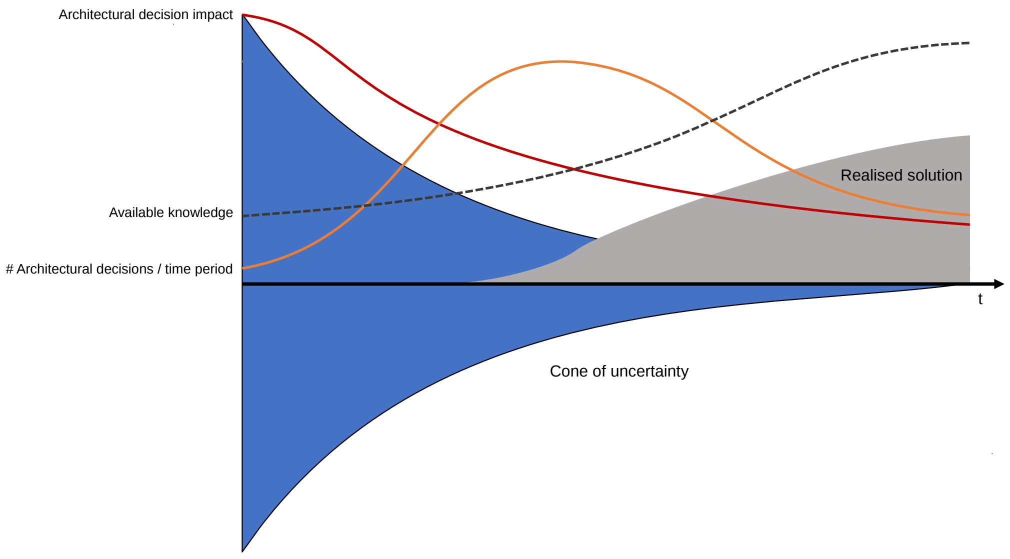


Figure 3.2: Architectural decision impact over time (from Poort (2014))

3.2 Design Decisions

A design decision describes why a particular choice was made from considered alternatives to address some concern or realise one or more requirements (Jansen, 2008). Not only software architects and designers are impacted by (architectural) decisions, but other stakeholders as well. Capilla et al. (2016) identify end-users, business managers and software maintainers as having different needs in regards to architectural knowledge. Meeting the needs of these stakeholders regarding their decision will likely improve the effectiveness of these decisions.

Decisions can be made at different abstractions and vary in scope. This can be seen in Figure 3.3. The problem space contains the problems that the system can address. The solution space contains all possible ways this system can address those problems. Within these spaces there is a level of abstraction, with at the top architectural significant requirements and architectural decisions which make up the software architecture. One level lower are requirements which map to detailed design and implementation decisions. As higher level decisions are made, the scope of the system becomes smaller as the architecture becomes defined, the system is designed, and the system is eventually implemented.

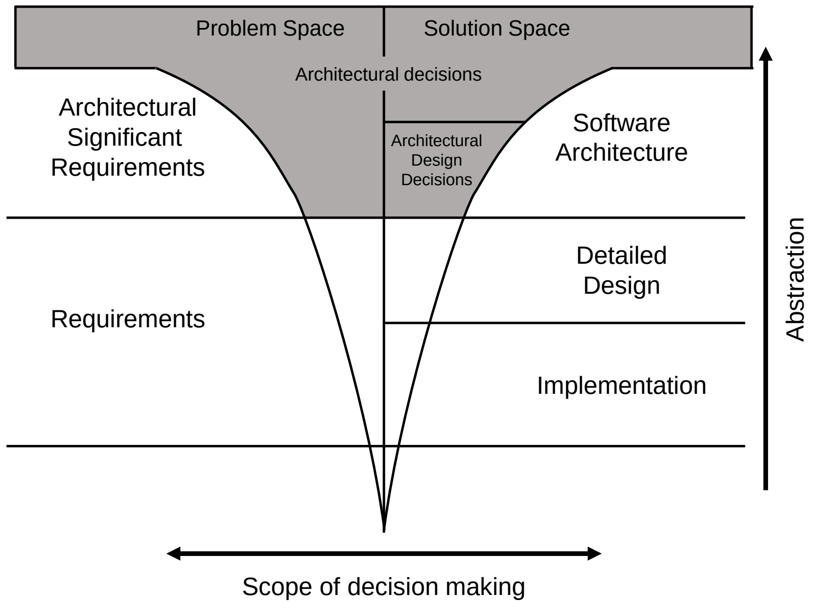


Figure 3.3: Scope and abstraction of decision making (from Jansen (2008))

Decisions can be documented in many places ranging from comments within code, files alongside code, external repositories such as Word documents, wikis, and project planning tools (Kopp, Armbruster, & Zimmermann, 2018). Documenting alongside code has the benefit of the documentation persisting after initial development. When the system is delivered, project documentation is archived and not likely to be used that much anymore, but code still needs to be maintained while the system is in operation.

Nygard (2011) argues for keeping a collection of architecturally significant decisions in version control alongside the code. A drawback is that some of the relevant stakeholders do not have easy access to these decisions as they are not close to the code as the development team is. Kopp et al. (2018) converted Y-statements into MADR records in markdown to be used in Git. This allows decisions to be maintained alongside the code while staying minimalistic. A problem that arises with this approach is that some (architectural) decisions concern larger parts of the system and should preferably be maintained somewhere more accessible.

Several tools exist to manage this architectural knowledge. Tang, Avgeriou, Jansen, Capilla, and Babar (2010) compared 5 such tools: Archium, ADDSS, AREL, Knowledge Architect and PAKME. Results show that these tools can support architectural design activities, re-usability of knowledge and traceability with requirements. Knowledge sharing is unfortunately lacking in these tools. A method which produces rationale of architectural knowledge as a byproduct is desired (Lee, 1997;

Capilla et al., 2016), the tooling that is used to manage this produced architectural knowledge should thus be incorporated into this method. As there is a lot of information that plays a role in a typical software project not all of it should or needs to be documented. Only the information that is believed to be needed in future activities should be documented, this practice is called value based documentation (Falessi et al., 2013). Farenhorst, Lago, and Van Vliet (2007) identify several desired properties of architecture knowledge sharing tools, a selection focusing on decision content is listed below:

1. *Stakeholder specific content*: by distinguishing between different types of content, stakeholders can filter only the relevant knowledge for them.
2. *Easy manipulation of content*: as architecting is an iterative process, decisions should be able to be changed easily to prevent the decision process from slowing down.
3. *Descriptive in nature*: strict prescribing on how knowledge should be managed needs to be avoided, as to not limit creativity in architects.

Falessi et al. (2013) experimented with only capturing design rationale elements which were deemed necessary for future activities to address the problem of overhead. Results show that the value of a rationale element is different per rationale category and that documenting only those items of value for future activities reduce documentation efforts (Tyree & Akerman, 2005).

Results from a survey by Tofan, Galster, and Avgeriou (2013) show that dependencies to other decisions make decision making more difficult. Decision making also becomes more difficult if there are no similar previously made decisions available. Those decisions which had more considered alternatives are regarded as better decisions than those who do not consider (or consider fewer) alternatives.

Decision templates

Many templates to capture architectural design decisions exist. Zimmermann et al. (2015) compared seven decision capturing templates that are currently used in the industry. This comparison was extended with the decision template from RCDA, as shown in Table 3.1. Many templates are very extensive and cover multiple pages, the Y-statement sticks out by its low element count and size of just one sentence.

Table 3.1: Design Decision template comparison (extended from Zimmermann et al. (2015))

AD aspect	**IEEE 42010 (v2.2)**	**IBM UMF AD Table**	**Tyree/ Akerman**	**Bredemeyer Key Decisions**	**Nygard ADRs**	**arc42 Hruschka/Starke**	**Y-Statements**	**RCDA Record of Decision**
ID	Unique Identifier	ID	(in header)	/	(part of name)	(section #)	(id)	(in header)
Outcome	Statement of the decision	Decision	Decision	Approach	Decision	Decision	we decided for	Decision
Requirements trace	Correspondence or linkage to concerns	(derived requirements)	Related requirements	Business drivers, technical drivers	/	/	/	Architectural concerns
Accountability	Owner of the decision	/	/	/	/	/	/	Participants
SA viewpoint trace	Correspondence or linkage to element	/	Related artifacts	/	/	/	in the context of	Related Information
Why-answers	Rationale	Justification	Argument	Conclusion	/	(question under decision)	(optional "because" half sentence)	Rationale
Decision drivers	Forces, constraints	/	(Constraints)	Benefits, drawback	Context	Constraints	facing	Constraints
Assumptions	Assumptions	Assumptions	Assumptions	/	/	Assumptions	/	/
Options	Considered alternatives	Alternatices	Positions	/	/	Considered alternatives	and not	Alternatives
Problem	/	Issues or problem	Issue	/	/	Problem	/	Overview
Decision dependencies	(not in template, but in standard)	Related decisions	Related decisions	/	/	/	/	Pre-requisites
Categorisation, classification	/	Subject area, topic	Group(ing)	/	/	(Decision topic)	/	/
Name	/	Name	(in header)	<<key decision>>	Title	(section heading)	/	(in header)
State of AD	(not in template, but in standard)	/	Status	/	Status	/	/	Status
Impact	(not in template, but in standard)	Implications	Implications	Issues/ Considerations	Consequences	/	to achieve, accepting drawbacks	Expected outcome
Other entries	Timestamps, Citations	Motivation	Notes, Related principles	Notes, Drivers realised	/	/	/	Strategy, Process, Participants Actions & follow-up
Element count	9	13	14	9 (plus 1-2 in header)	5	5 (with 14 questions)	6	14
Size	/	not published	(example is half a page)	/	1-2 pages	to be ordered by importance	1 (long) sentence	1-2 pages
Publication year	2011	1998	2005	2005	2011	2012	2012	2014

3.3 Design Rationale

A design rationale captures the design reasoning, how and why an artefact is designed the way it is, and the justification (considered alternatives, evaluated trade-offs, argumentation) for the design decisions (Tang et al., 2006; Lee, 1997; Gruber & Russell, 1991). However, design rationale is not just writing down design decisions and justifications. Design rationale has many uses (Burge & Brown, 1998):

1. Verifying whether the design satisfies stakeholder goals;
2. Evaluating if a design decision is correct;
3. Keeping track of earlier decisions;
4. Reusing design elements in another project;
5. Communicating and teaching the design to others;
6. Assisting architects in making sure their decisions are explicit and include justifications;
7. Providing an overview of design decisions of the system across its lifetime.

There is a growing need for documenting design rationale (Bosch, 2004; Capilla et al., 2016). Additionally, there is a lack of a standard approach for documenting design rationale (Tang et al., 2006). However, architects do not always document the design reasoning for decisions they make. This can be due to deadlines and budget constraints (Conklin & Yakemovic, 1991; Tang et al., 2006; Falessi et al., 2013), lack of standards and processes to documenting design rationale, and the cost and benefit of documenting design rationale (Tang et al., 2006; Capilla et al., 2016). Lack of rationale (or rationale that is not properly documented) can lead to architects forgetting the rationale for their own design decisions (Tang et al., 2006). Furthermore, when architects need to modify designs made by other architects, they often do not understand the design if there is no design rationale provided by the original designer (Tang et al., 2006). In conclusion, companies have much to gain from documenting design rationale.

In a survey among practitioners, Tang et al. (2006) specify 9 types of design rationale and ranked their level of importance, frequency of use and frequency to document (see Table 3.2). The results show that although architects consider rationale very important, it is not always used and even less likely to be documented. This shows there are improvements to be made in this regard.

Table 3.2: Usage of design rationale, ranked by level of importance (from Tang et al. (2006))

Type of design rationale	**Level of importance (%)**	**Frequency of use (%)**	**Frequency to document (%)**
Design benefit	90.1	85.3	69.1
Design constraints	87.6	87.6	82.7
Certainty of design	85.2	85.2	46.9
Cost of design	77.7	69.1	45.7
Certainty of implementation	76.5	76.5	39.5
Design assumptions	74.0	64.1	79.0
Design complexity	71.6	70.3	50.6
Trade-offs between alternatives	64.2	64.2	49.4
Design weakness	61.7	55.6	35.8

Capilla et al. (2016) propose several methods to promote rationale capture:

1. *Convince relevant stakeholders*: not just a software architect is impacted by design decisions. Stakeholders involved in a system can benefit from storing design decisions: how the system was built, considered alternatives, how a system evolved etc. Convincing these stakeholders they can benefit from captured design rationale, promotes further awareness in the whole company.

2. *Systematically capture architectural knowledge*: when an organisation is convinced of the benefit to capturing architectural knowledge, relevant stakeholders need to be supported to capture, share and use key decisions in all development phases through a systematic process.

3. *Use lean approaches for capturing architectural knowledge*: every decision is different and only relevant knowledge for that decision should be captured. This reduces capturing effort and makes it more cost-effective.

4. *Embed rationale capture with current approaches and tools*: having an integrated tool for all aspects of the design process prevents duplicate effort by using multiple tools. Lee (1997) note that a process-based approach where the rationale is captured as a methodological byproduct is appealing as the method is less intrusive and has a low cost.

3.4 Design Reasoning

When solving a design problem, decision makers use design reasoning. Goldschmidt and Weil (1998) describe design reasoning as an activity where "designers are engaged in a search in which many ideas and possibilities are typically raised and examined, often to be discarded later. This is done by reasoning about bits of information and knowledge, in an attempt to construct a coherent rationale for a design idea". The process of design reasoning uses both logical and rational thinking to support arguments and make decisions (Schriek et al., 2016). Unfortunately a perfect rational design reasoning process is implausible as humans will always be biased to some degree, but putting more effort into a more rational design reasoning process will improve overall design quality (Parnas & Clements, 1986). This section outlines some of the factors involved in design reasoning, as well as techniques to improve this process.

Intuition & Biases

Decision making of designers is often a naturalistic process (Zannier et al., 2007), where decisions are made on intuition, experience and gut feeling (Zsambok & Klein, 2014). This unstructured approach influences design quality, where the more experience a designer has, the more likely they are to make better decisions and vice versa (Tang & Van Vliet, 2009).

During design reasoning and decision making in software design, people are subject to cognitive biases (Razavian, Tang, Capilla, & Lago, 2016; Schriek et al., 2016; Tang, 2011; Dietrich, 2010). Cognitive bias occurs when "individuals draw inferences or adopt beliefs where the evidence for doing so in a logically sound manner is either insufficient or absent" (Buss, 2005).

Architectural decisions, especially those made early in a project, are often based on incomplete information (Poort, 2014). As new information becomes available, decision makers are disinclined to change their earlier decisions when new information becomes available. This bias is called *anchoring*, where people have the urge to rely heavily on initial information. Research has shown that both novice and experienced designers are susceptible to anchoring (Razavian et al., 2016).

People also have the tendency to look for, interpret and favour information that confirms one's existing beliefs or hypotheses, called *confirmation bias* (Oswald & Grosjean, 2004). This bias is also present among professionals in software engineering (Calikli, Bener, & Arslan, 2010).

The way a problem is presented also leads to a cognitive bias, namely *framing bias*. Framing bias occurs when people respond differently based on the way a problem is presented. People tend to avoid risk when a problem is presented positively but actively seek risk when it is presented negatively (Tversky & Kahneman, 1981). This bias is also present in software engineering, in a recent study by Mohanani, Ralph, and Shreeve (2014), an assignment to design a software system was framed in one group to include a requirements specification versus another group where the same requirements were phrased more as design ideas. Results show the first group fixated on satisfying the requirements while the other group came up with more original ideas.

Many more cognitive biases exist in software development, see Arnott (2006) for a more complete list.

Rational decision making is when all options are considered carefully and the best option is chosen every time. This is of course the preferred way of making decisions but is not so applicable to real life. In his book, Kahneman (2011) gives many examples where people make irrational decisions, often purely based on intuition. The idea that rationality is limited when people make decisions is called *bounded rationality*, where people are limited by the cognitive capacity of their mind, available information and time available to make the decision (Gigerenzer & Selten, 2002). Coming to an optimal solution is thus improbable because of lack of ability and resources of decision makers. Van Vliet and Tang (2016) theorise that decision makers apply their rationality after having simplified the problem and available choices. This leads to decision makers arriving at a solution that they think is good enough, but probably not optimal. This behaviour is called satisficing and has been shown to lead to lower quality design decisions among software designers who do not show this behaviour (Tang & van Vliet, 2015).

Intuition has been researched in many different domains. Most domains conceptualise intuition in *nonconscious* processing of information, allowing individuals to base decisions on their gut feeling and *conscious* processing, where a decision is made in a more deliberate manner (Dane & Pratt, 2009). Eling, Griffin, and Langerak (2014) identified several benefits and of unconscious processing when compared to conscious processing, namely:

1. higher capacity to processing information
2. access to higher capacity to processing information
3. access to implicit and tacit knowledge

Several shortcomings are also identified:

1. not applicable in situations where strict rules are present regarding decision criteria
2. since the process happens implicitly, the wrong intuition may be applied
3. the reasoning of an intuition is not always present

Their research shows that intuition can be very beneficial to making decisions, as long as one is aware of the drawbacks that come with it.

80-85% of decisions are made by groups (Rekhav & Muccini, 2014; Tofan et al., 2013). From a survey among practitioners, Rekhav and Muccini (2014) identify the following issues with decision making in groups: individuals not understanding the goals, long time to reach agreement, presence of power difference in groups and *Groupthink*. Groupthink happens when a group is highly cohesive and the aim is to find consensus over individuals giving their own opinions (Janis, 2008).

Improving design reasoning

A two mind model has been theorised by Razavian et al. (2016). They distinguish between reasoning (mind 1), where one systematically reasons within a design context, and reflective thinking (mind 2), where one challenges and revises their own decision. Reflective questions can be used across the process of software design to use mind 2 to challenge the reasoning of mind 1. The authors argue that combining mind 1 & 2 is desirable as it makes the decision maker aware of the potential issues in their reasoning. The reflective questions are classified as two types: external reflections (made by a third party) and internal reflections (raised by the designers themselves). Results from multiple case studies show that the group using reflective thinking had a higher quality design discourse compared to the group in which reflective thinking was not used.

Pretorius et al. (2018) suggest using rationality in combination with intuition in decision making in Software Architecture. The authors argue that intuitive reasoning does not always lead to errors, as research in other fields has shown that intuition can be used during certain stages in the design process to come to more original ideas as well as faster and better decision making. The authors identified three factors affecting decision making that warrant further research: *expertise* (more experience allows the architect to better use their intuition), *time pressure* (decisions taken based on intuition can lead to faster decisions) and *individual vs group* decisions (unclear how intuition influences group decision making).

Tang (2011) proposes several techniques to improve rational design reasoning:

1. *Reasoning and inferences*: As reasoning is a thought process, the only way to show this process is to articulate this reasoning and document the rationale. A decision maker needs to make sure that all their assumptions, arguments, premises, tradeoffs are elaborated upon and include reasoning.

2. *Problem structuring*: Determining which design issues to tackle first influences the way design activities are carried out. As such, decision makers should first consider what to tackle first before going in-depth exploring possible solutions.

3. *Assumption analysis*: Where any (conscious or unconscious) assumptions that have been made are questioned.

4. *Constraint analysis*: A system that needs to be designed is constrained by the requirements as well as project and system environments. Not all of these constraints are documented and remain tacit. Constraints should be noted down to deter or detect conflicts early on.

5. *Option analysis*: A decision maker should consider multiple options to solve a problem. This reduces the anchoring bias where the option first considered is likely to stick. Through this exploration, potential better options can be identified.

6. *Trade-off analysis*: When multiple viable options are considered as solutions, a trade-off analysis needs to be performed to determine which one is the best solution. this helps the decision maker evaluate their priorities regarding their desired solution.

7. *Risk analysis*: Decision makers need to be aware of the risks that can occur during the project life cycle. These risks can influence the eventual architecture or design and should therefore by articulated and estimated.

Previous research shows these techniques improve design reasoning (Tang, Aleti, Burge, & van Vliet, 2010; Schriek et al., 2016; Tang, Tran, Han, & Van Vliet, 2008).

3.5 Requirements from the Literature

From the literature study the following requirements with regards to Decision Documentation (DD) are derived (see Table 3.3).

Table 3.3: Derived requirements from the literature review

ID	Requirement	Source
L1	DD should capture tacit and documented knowledge in a formalised process	Nonaka and Takeuchi (1995), Capilla et al. (2016)
L2	DD should combine rational and intuitive decision making	Schriek et al. (2016), Pretorius et al. (2018)
L3	DD should support design reasoning techniques	Tang (2011)
L4	DD should be able to show the decision satisfies stakeholder goals	Burge and Brown (1998), Capilla et al. (2016)
L5	DD should be evaluated to show a decision is correct	Burge and Brown (1998)
L6	DD should include design rationale	Tang et al. (2006)
L7	DD should capture rationale as a methodological byproduct	Lee (1997)
L8	DD should capture only relevant knowledge	Falessi et al. (2013), Capilla et al. (2016)
L9	DD should capture decisions at different abstraction levels effectively	Jansen (2008)
L10	DD should be flexible in the places it can be documented in	Kopp et al. (2018)
L11	DD should distinguish between stakeholder specific content	Farenhorst et al. (2007), Capilla et al. (2016)
L12	DD content should be easily changed	Farenhorst et al. (2007)
L13	DD should try to remain descriptive in nature	Farenhorst et al. (2007)
L14	DD should take into account dependencies to other decisions	Tofan et al. (2013)

Chapter 4

Requirements Elicitation

4.1 Approach

Exploratory interviews with Software and Solution architects were conducted at CGI in the beginning of the research project. The aim of these interviews is to gather knowledge on how architects are currently making and documenting decisions and what problems they are experiencing in the current way of working.

The results are used to gather requirements for Decision Documentation (DD). In total 10 interviews were held of which eight were architects, one developer and one delivery manager. 9 of 10 interviews were conducted in a one-on-one setting and recorded using a mobile phone and 1 interview was conducted with 3 architects. The interviews follow a semi-structured approach to allow the interviewees to elaborate on their answers with examples and add more depth to the interview. This sometimes lead to some questions being skipped due to time constraints, already being (partly) answered in previous questions or the question not being applicable to the interviewee. The interview protocol (in Dutch) is shown in Appendix A.1.

4.2 Results

The interviews led to the list of requirements in Table 4.1. The table shows the requirements, the link of the requirement to the requirements from the literature of Table 3.3, the percentage of Architects which mentioned the requirement during the interviews and the degree of support there is for the requirement from the architects on a scale of * (little support) to **** (very strong support).

This section further describes the answers of the interviewees to the questions of the interview protocol.

Table 4.1: Derived requirements from the interviews combined with the requirements from the literature

ID	Requirement	Link with req's from literature	% of Architects	Rating
R1	DD should include a rationale	L2, L6, L7	83%	****
R2	DD should address a (architectural) concern	L3, L6	50%	****
R3	DD should describe the most relevant alternatives	L3, L6	25%	***
R4	DD should include consequences	L3, L6	58%	****
R5	DD should show the grounds that lead to the decision	L1, L3, L6	16%	**
R6	DD should include the context of the decision		50%	****
R7	DD should be compact and easily digestible	L11, L12	58%	****
R8	DD should show the decision process	L5, L7	41%	***
R9	DD should avoid volatile attributes	L13	16%	**
R10	DD should be a standard with ground rules with room for creative freedom	L1	16%	**
R11	DD should be formulated for its target audience	L11	41%	***
R12	DD should be at one level of detail with a hierarchical structure for sub-decisions	L9	8%	*
R13	DD should be easy to review and maintain	L12	25%	**
R14	DD should have version history		8%	*
R15	DD should be easily retrievable	L10	16%	**
R16	DD should have a link to a requirement		16%	**
R17	DD should not be open for people's own interpretation		48%	***
R18	DD should measure the degree of acceptance by its stakeholders	L4	41%	***
R19	DD should consider one subject per decision		25%	**

What is documented, how is this documented and by whom? Do you use standards or templates to do this?

Almost all architects note that they document the rationale how they arrived at their decision. Other aspects they document are: consequences, alternatives, elaboration and (architectural) concern. Furthermore, some decisions are taken implicitly, based on intuition, but are not documented (3 interviews).

One interviewee remarked that "People prefer to use their own method or way of working over that of someone else. I think the best way to play into this is to use a general way of working within the organisation using certain standards and principles with creative freedom for people's own way of working. This is probably a delicate balance."

Seven Architects use the JSTD-016 standard for documentation. This standard does not include a decision template and most document their decisions in free format. This makes the person who sets up the document for the first time the one responsible for creating the decision template which is often adhered to by the rest of the team (1 interview).

All architects were familiar with the RCDA approach by Poort (2014) and 4 of them use the RCDA Excel template to document their concerns and decisions. However, Excel is not collaboration friendly when compared to the current way of working in Git and Markdown (2 interviews). One Architect uses their own decision template based on the RCDA template but not that specific template as they think it is not usable in its current format.

How are decisions communicated?

Architects mentioned that decisions are communicated in the following ways (see Table 4.2).

Table 4.2: Ways of communicating decisions

Description	**Interviews**
Memo	3
Documents	3
Workshop with the customer	3
Meetings	2
Email	1
Verbally	1

Stakeholders of decisions can be seen in Table 4.3. When communicating with business stakeholders, two architects note they do not use complicated or big templates as these stakeholders have a hard time understanding those templates. Instead they use a PowerPoint or draw it out on a whiteboard.

Table 4.3: Stakeholders of decisions

Description	**Interviews**
(Future) Architects or the future self	3
The team who builds the system	2
The team who maintains the system after it is built	2
Business stakeholders	2

Do you experience problems in the current way of working?

The problems the architects experience in their current way of working are summarised in Table 4.4.

Table 4.4: Problems in the current way of working

Description	**Interviews**
RCDA Excel sheet is not easy to work with	3
Lack of a standard way of working for documentation/common problems	2
Making decisions based on intuition and not documenting these decisions	2
Documentation being too large leading to findability and maintainability problems	2
Conflicting interpretations of decisions	2
No re-usability in projects	1
Not using the most mature tooling	1

One architect elaborates the primary reason for not using the RCDA template is: “The RCDA templates currently available are very hard to use as they are very large and a lot is not applicable to my situation. These templates force you in a specific format which is very hard to communicate with less technical stakeholders.” Another architect adds to that with “A lot of attributes in the current template focus on status attributes, this is not relevant anymore in 5 years and there are far better tools suited to track status and such.”

Most of the projects of the architects use an Agile way of working. One architect noted that: “By working agile, there is more discussion. This should lead to more documentation, but this is not the case.”

One architect, who often works in bid processes where multiple parties compete for a project, says they often encounter friction between the architects and sales. The architect elaborates: “There is always friction between solution and sales. Sales wants to keep as many options open as possible, whereas a solution manager wants to make decisions and make more decisions based on those earlier decisions. This is a challenge for both parties.”

One developer is working on a project with a lifespan of over 10 years which using very old documentation which has become very hard to maintain. The developer noted that: “Most of the new documentation is done in comments alongside the code itself. Very rarely we change the actual design in the documentation itself. The fact that the design documentation is put on the side shows how people think about this documentation.”

What is going well in the current way of working?

Three Architects note that the RCDA approach helps them focus on the right things and lets them discover the important stuff early on in their projects. Two of them elaborate by saying their approach is based on RCDA but changed to fit their own project needs. One architect says: “For small projects, a more lightweight approach of RCDA could be beneficial”.

Two of the architects involve their customers through workshops where they address concerns with their customer. This puts the customer at ease that the project is going well, involves the customer in the project and challenges their own work through constructive feedback. One of the architects notes this often leads to easy acceptance, because the customer understands how the product came to be and they are the ones in the lead.

Do you reflect on decisions?

Reflection is mostly project and process related and often not on decisions themselves (2 interviews). Reflection with other architects happens mostly informally, during coffee or when you are free from the job, this is hard to enforce formally (1 interview).

What makes a decision effective in its use?

A summary of what makes a decision effective according to the interviewees is summarised in Table 4.5.

Table 4.5: Decision effectiveness

Description	Interviews
Compact and easily digestible	5
Includes a rationale	5
Includes context	4
Includes the impact/consequences of the decision	3
Includes alternatives	3
Traceability from requirement to decision	2
Stakeholders need to not only understand the actual decision but also the context and the why of that decision	1
Needs to be understood by the business	1

One architect emphasises decisions need to be made about how non functional requirements of the project are guaranteed. If a decision puts several of these in conflict with each other, architects need to show how they balanced the decision by documenting the rationale and impact.

Decisions need to be communicated to and accepted by the relevant stakeholders. To achieve this it needs to be well written, understandable and adjusted to the target audience, not open for people's own interpretation (1 interview). Furthermore, a decision needs to be about one decision and one subject, not a collection. Three architects further state: "A trade-off with templates is: how much do you want to make specific for your teams and how much freedom do you want to give them? In theory you could prescribe everything all the way to the bottom on how people should work."

How important is documenting the decision process?

Documenting the decision process is important to show that the correct decision was taken for the right reasons and the right assumptions (1 interview). If the stakeholders of your customer change it is also important, as that changes the context of your decision and previous decisions might be different if you had to make them today (1 interview). Another architect adds that the process is also important if there are potential legal consequences, for example if a deviation within a decision is chosen explicitly by request of the customer.

Chapter 5

Capturing Design Rationale as Decision Stories

This chapter describes how the Decision Story evolved from previous research and how it was further developed over the lifespan of the research project. The following sections describe its origin, similar approaches from the literature and how the experiment, discussions, and interviews further shaped the template in iterative versions.

5.1 Evolution

Origin

The origin of the Decision Story can be found in the book Just Enough Software Architecture by Fairbanks (2010). To make rational architectural decisions, Fairbanks argues decisions should follow the following pattern:

<x> is a priority, so we choose design <y>, and accept downside <z>

This pattern can be used to resolve disagreements between different design options by exposing relevant quality attributes in a decision, framing the problem from an engineering perspective instead of different judgements from designers.

Based on this initial format, Zdun et al. (2013) proposed another approach aiming for lean and minimalistic documentation. Their approach, called a Y-statement, produces the following statement:

In the context of <use case/user story>, *facing* <concern c> *we decided for* <option o> *to achieve* <quality q>, *accepting* <downside d>.

The authors note that lean documentation is the preferred way of working for architects instead of using large decision templates. However, when an architect is new to a project, large decision templates are preferred to get the full picture.

In the Risk and Cost Driven Architecture (RCDA) training by Poort (2014), the format was adopted and slightly altered to include the "*and not* <alternatives a>" after the primary decision to show the most relevant alternative to the decision. These six elements formed the starting point for the Decision Story template in this research project.

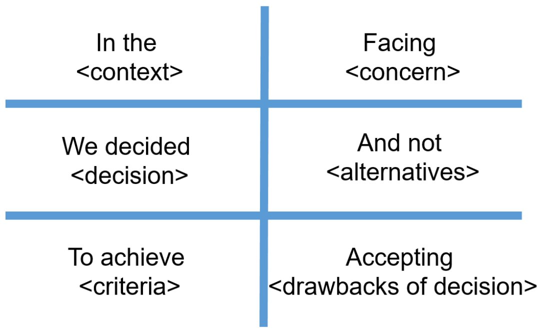


Figure 5.1: The Decision Story at the start of the research project

The Decision Story in literature

To promote critical thinking in students Snyder and Snyder (2008) use the "*IDEALS*" approach, consisting of six steps:

1. **Identify** the problem;
2. **Define** the Context;
3. **Enumerate** choices;
4. **Analyze** options;
5. **List reasons**;
6. **Self-Correct**.

These steps can roughly be mapped to the Decision Story panels as well. However, this approach describes a process, the Decision Story describes an outcome. Thus, when making a decision using the *IDEALS* approach, it could be documented using a Decision Story, mapping the outcome of each step to the Decision Story.

Incorporating the decision process

When looking at Figure 5.1, it describes an outcome. It can only be used to summarise a decision once it has been made, it does not guide the decision maker in making a decision. Thus to use the Decision Story to make decisions, a different order is needed. The process-based Decision Story is shown in Figure 5.2. The activities are based on the Architecting Microcycle by Poort (2014), shown in Figure 3.1 and named to match the corresponding fields of the Decision Story. Note that the arrows between the activities point two ways. This indicates that it is possible to revisit earlier activities. For example, it can become clear that none of your considered options are viable for your criteria.

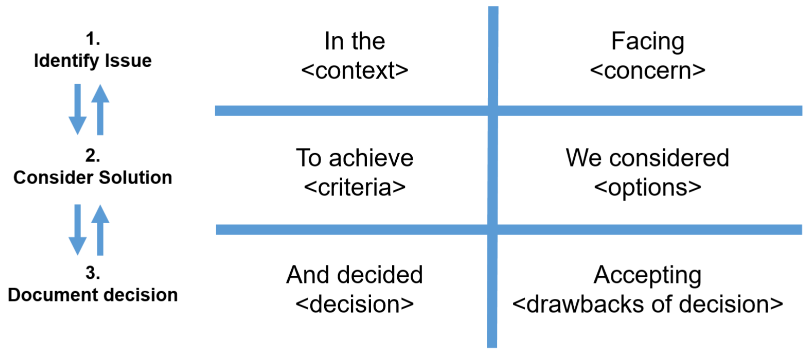


Figure 5.2: The Decision story Process. *left: the activities to perform when making a decision. right: the corresponding fields of the Decision Story that are output of the activities (identify Issue corresponds with <context> and <concern>)*

Data model and definitions

The data model (modeled as an UML Class Diagram) of the Decision Story can be seen in Figure 5.3. This model represents the final version of the Decision Story. Compared to the version in the previous section, “accepting <drawbacks>” has been changed to “Accepting <consequences>” (see Section 7.2) and “and decided <decision>” was changed to ‘and decided <decision>, because <rationale>” (see Section 6.2).

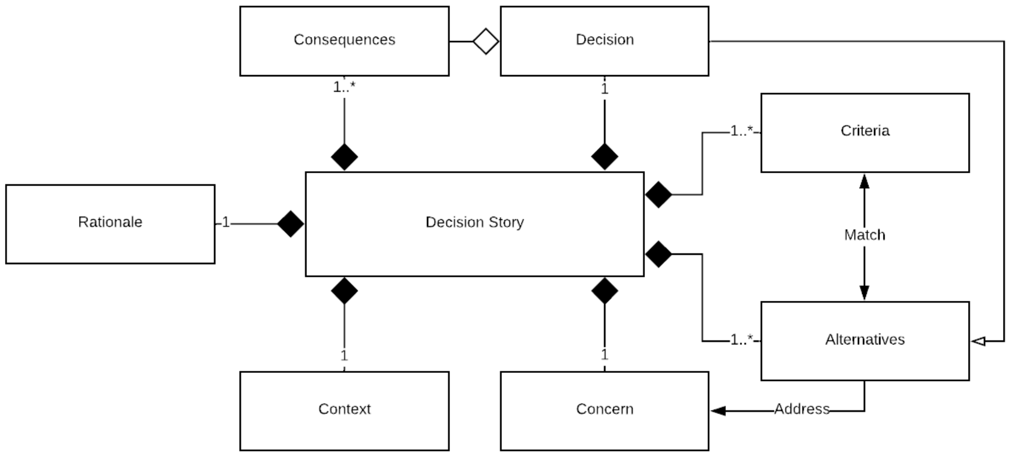


Figure 5.3: Decision story data model

As can be seen in the model, a Decision Story consists of 6 elements. A Decision Story has one concern and one context. To address this concern, one to many (though usually at least 2 and no more than 4) reasonable alternatives are considered. These alternatives all score on the set acceptation criteria (at least one, usually several). The result of this matching process leads to the best alternative (at the time of making the decision) to become the eventual decision. This decision has positive and negative consequences and potential risks associated with the decision.

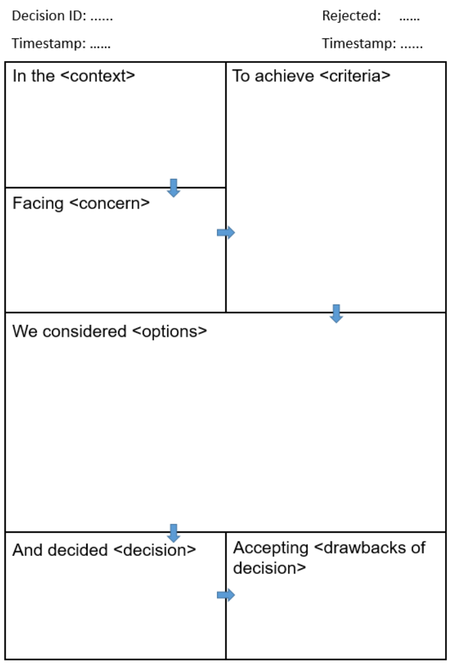
Definitions for all the decision story elements were defined through discussions with the 1st supervisor and an industry expert (See Table 5.1).

Table 5.1: Decision Story definitions

Element	**Definition**
Context	Framework of the problem you are going to talk about. A description of where the problem is situated.
Concern	A compact description of the problem you need to make a decision about, without talking about the solution.
Criteria	What your solution must adhere to, acceptation criteria.
Options	The relevant reasonable alternatives.
Decision	The decision taken.
Rationale	Justification why the chosen option is the best fit on the criteria.
Consequences	Direct effects of the taken decision plus the risks of the decision.

5.2 Decision Story Candidates

Several Decision Story templates and visualisations were created and evaluated, the proposed candidates can be seen in Figure 5.4. Candidates A1 and B1 were evaluated in the student experiment. Candidates A2, B2 and C were evaluated in the case studies, interviews and focus group.



Decision ID: ......

Rejected: ......

Timestamp: ......

Timestamp: ......

In the <context>

Facing <concern>

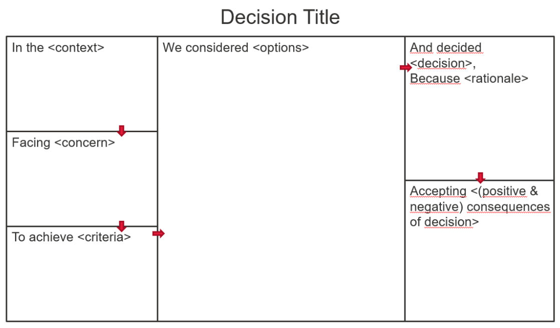
To achieve <criteria>

We considered <options>

And decided <decision>

Accepting <drawbacks of decision>

(a) Candidate A1



Decision Title

In the <context>

Facing <concern>

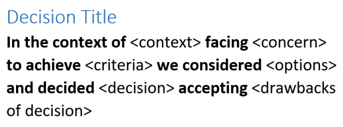
To achieve <criteria>

We considered <options>

And decided <decision>, Because <rationale>

Accepting <(positive & negative) consequences of decision>

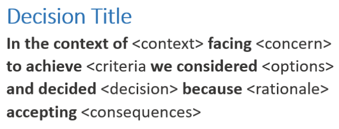
(b) Candidate A2



Decision Title

In the context of <context> **facing** <concern> **to achieve** <criteria> **we considered** <options> **and decided** <decision> **accepting** <drawbacks of decision>

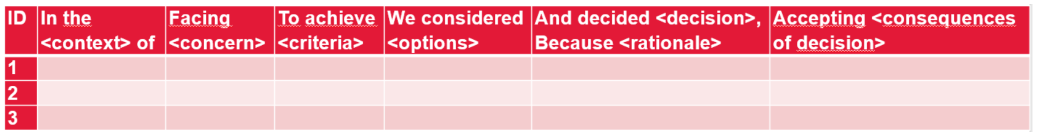
(c) Candidate B1



Decision Title

In the context of <context> **facing** <concern> **to achieve** <criteria **we considered** <options> **and decided** <decision> **because** <rationale> **accepting** <consequences>

(d) Candidate B2



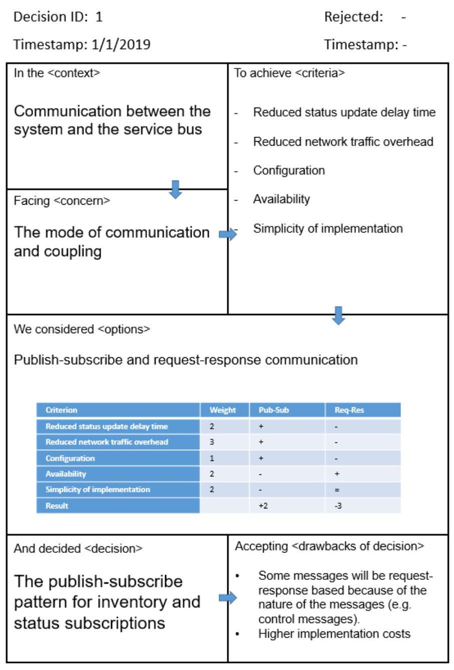
ID	In the <context> of	Facing <concern>	To achieve <criteria>	We considered <options>	And decided <decision>, Because <rationale>	Accepting <consequences of decision>
1						
2						
3						

(e) Candidate C

Figure 5.4: Decision story candidates

An example

An example of a decision in all the visualisations is shown in Figure 5.5. In examples A1 and A2, a trade-off matrix is used to weigh the different alternatives on. This is one method of weighing your alternatives, but is by no means mandatory. These examples are just to get an idea how the different visualisations look, and can be changed in whatever way the decision maker deems best.



Decision ID: 1 Rejected: -

Timestamp: 1/1/2019 Timestamp: -

In the <context>

Communication between the system and the service bus

To achieve <criteria>

- Reduced status update delay time
- Reduced network traffic overhead
- Configuration
- Availability
- Simplicity of implementation

Facing <concern>

The mode of communication and coupling

We considered <options>

Publish-subscribe and request-response communication

Criterion	Weight	Pub-Sub	Req-Res
Reduced status update delay time	2	+	-
Reduced network traffic overhead	3	+	-
Configuration	1	+	-
Availability	2	-	+
Simplicity of implementation	2	-	=
Result		+2	-3

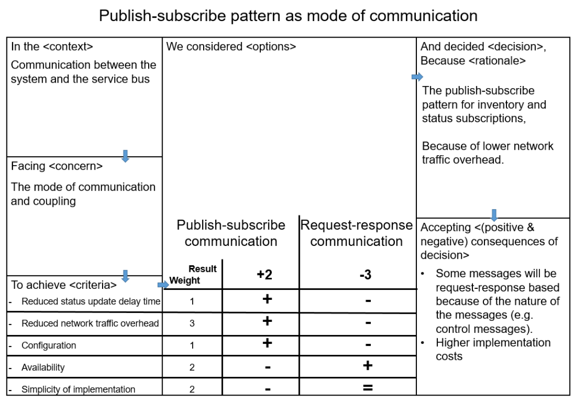
And decided <decision>

The publish-subscribe pattern for inventory and status subscriptions

Accepting <drawbacks of decision>

- Some messages will be request-response based because of the nature of the messages (e.g. control messages).
- Higher implementation costs

(a) Candidate A1



Publish-subscribe pattern as mode of communication

In the <context>
Communication between the system and the service bus

Facing <concern>
The mode of communication and coupling

We considered <options>

And decided <decision>, Because <rationale>

The publish-subscribe pattern for inventory and status subscriptions,

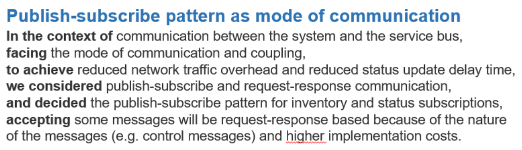
Because of lower network traffic overhead.

To achieve <criteria>	Result / Weight	Publish-subscribe communication	Request-response communication
		+2	-3
- Reduced status update delay time	1	+	-
- Reduced network traffic overhead	3	+	-
- Configuration	1	+	-
- Availability	2	-	+
- Simplicity of implementation	2	-	=

Accepting <(positive & negative) consequences of decision>

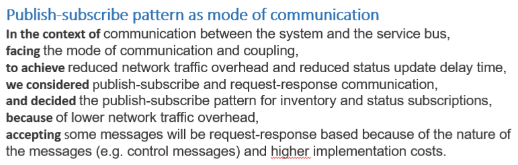
- Some messages will be request-response based because of the nature of the messages (e.g. control messages).
- Higher implementation costs

(b) Candidate A2



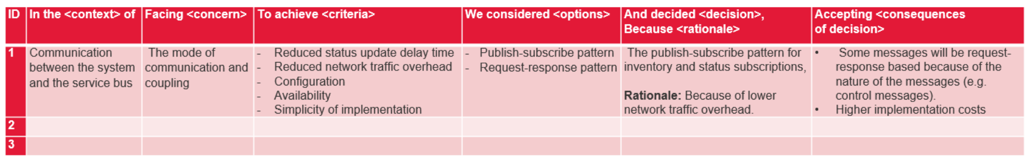
Publish-subscribe pattern as mode of communication
In the context of communication between the system and the service bus,
facing the mode of communication and coupling,
to achieve reduced network traffic overhead and reduced status update delay time,
we considered publish-subscribe and request-response communication,
and decided the publish-subscribe pattern for inventory and status subscriptions,
accepting some messages will be request-response based because of the nature of the messages (e.g. control messages) and higher implementation costs.

(c) Candidate B1



Publish-subscribe pattern as mode of communication
In the context of communication between the system and the service bus,
facing the mode of communication and coupling,
to achieve reduced network traffic overhead and reduced status update delay time,
we considered publish-subscribe and request-response communication,
and decided the publish-subscribe pattern for inventory and status subscriptions,
because of lower network traffic overhead,
accepting some messages will be request-response based because of the nature of the messages (e.g. control messages) and higher implementation costs.

(d) Candidate B2



ID	In the <context> of	Facing <concern>	To achieve <criteria>	We considered <options>	And decided <decision>, Because <rationale>	Accepting <consequences of decision>
1	Communication between the system and the service bus	The mode of communication and coupling	- Reduced status update delay time - Reduced network traffic overhead - Configuration - Availability - Simplicity of implementation	- Publish-subscribe pattern - Request-response pattern	The publish-subscribe pattern for inventory and status subscriptions, **Rationale:** Because of lower network traffic overhead.	• Some messages will be request-response based because of the nature of the messages (e.g. control messages). • Higher implementation costs
2						
3						

(e) Candidate C

Figure 5.5: Decision story examples

5.3 Evaluating the Decision Story

In this section the Decision Story is analysed against the requirements derived from the literature and interviews. Furthermore the template is compared against the other industry templates identified in Section 3.5.

Evaluating the Decision Story against the requirements

Here, the Decision Story is analysed against the requirements in Table 4.1. For the evaluation, candidate A2 from Figure 5.4 will be used, as it is the latest iteration of the Decision Story.

Requirements satisfied by the Decision Story

The Decision Story template can satisfy the following list of requirements from Table 4.1.

Req1 *DD should include a rationale*
Reason: The Decision Story has the field *because <rationale>*.

Req2 *DD should address a (architectural) concern*
Reason: The Decision Story has the field *facing <concern>*.

Req3 *DD should describe the most relevant alternatives*
Reason: The Decision Story has the field *we considered <options>*.

Req4 DD should include consequences
Reason: The Decision Story has the field *accepting <consequences>*.

Req5 DD should show the consideration that lead to the decision
Reason: A completed Decision Story tells a complete story of your consideration to make the decision.

Req6 DD should include the context of the decision
Reason: The Decision Story has the field *in the <context>*.

Req8 DD should show the decision process
Reason: The template is structured to show the process of how the decision was made.

Req9 DD should avoid volatile attributes (such as status/sprint)
Reason: The template does not include volatile attributes.

Requirements facilitated by the Decision Story

The following requirements can not be satisfied by the template directly but are facilitated by the template.

Req7 DD should be compact and easily digestible
Reason: When the template is filled in in a concrete manner, the fields will not become too big and lead to the decision remaining compact and easily digestible. This is dependent on the decision maker, and with proper training the template can facilitate this requirement.

Req10 DD should be a standard with ground rules with room for creative freedom
Reason: The template is flexible in how it can be documented and can easily be expanded if the user deems it necessary.

Req11 DD should be formulated for its target audience
Reason: This is dependent on how the decision is filled in, and this requirement can be satisfied by proper training of the decision maker. As the template shows the decision process, it should be easier to address different audiences over more technical templates.

Req13 DD should be easy to review and maintain
Reason: As the template follows the process of how the decision maker arrived at the decision it is easier for the reviewer to understand the decision. As the template is small, it also becomes easier to review and maintain.

Req17 DD should not be open for people's own interpretation
Reason: This is dependent on how the decision is filled in, and this requirement can be satisfied by proper training of the decision maker.

Req19 DD should consider one subject per decision
Reason: This is dependent on how the decision is filled in, and this requirement can be satisfied by proper training of the decision maker.

Requirements the Decision Story does not satisfy

Some of the requirements can not be satisfied by the Decision Story template.

Req12 DD should be at one level of detail with a hierarchical structure for sub-decisions
Reason: Documenting at one level of detail can be done by training the decision maker. A hierarchical structure for decisions and sub-decisions the template does not address and should be done by the decision makers.

Req14 DD should have version history
Reason: The template does not have a history of decisions or how they evolved over time, it shows the process and outcome.

Req15 DD should be easily retrievable
Reason: The template does not prescribe (on purpose) how it should be stored or retrievable, this is up to the users.

Req16 DD should have a link to a requirement
Reason: A link to a requirement is not enforced by the template.

Req18 DD should measure the degree of acceptance by its stakeholders
Reason: This is hard to measure at all and the Decision Story template does not measure this.

The Decision Story compared to industry templates

In this section, template comparison from Table 3.1 is extended with the Decision Story. This extended table can be seen in Table 5.2.

Table 5.2: The Decision Story and Industry templates

AD aspect	IEEE 42010 (v2.2)	IBM UMF AD Table	Tyree/ Akerman	Bredemeyer Key Decisions	Nygard ADRs	arc42 Hruschka/Starke	Y-Statements	RCDA Record of Decision	Decision Story
ID	Unique Identifier	ID	(in header)	/	(part of name)	(section #)	(id)	(in header)	(optional in header)
Outcome	Statement of the decision	Decision	Decision	Approach	Decision	Decision	we decided for	Decision	and decided <decision>
Requirements trace	Correspondence or linkage to concerns	(derived requirements)	Related requirements	Business drivers, technical drivers	/	/	/	Architectural concerns	/
Accountability	Owner of the decision	/	/	/	/	/	/	Participants	/
SA viewpoint trace	Correspondence or linkage to element	/	Related artifacts	/	/	/	in the context of	Related Information	in the <context>
Why-answers	Rationale	Justification	Argument	Conclusion	/	(question under decision)	(optional "because" half sentence)	Rationale	because <rationale>
Decision drivers	Forces, constraints	/	(Constraints)	Benefits, drawback	Context	Constraints	facing	Constraints	facing <concern>
Assumptions	Assumptions	Assumptions	Assumptions	/	/	Assumptions	/	/	/
Options	Considered alternatives	Alternatices	Positions	/	/	Considered alternatives	and not	Alternatives	we considered <options>
Problem	/	Issues or problem	Issue	/	/	Problem	/	Overview	facing <concern>
Decision dependencies	(not in template, but in standard)	Related decisions	Related decisions	/	/	/	/	Pre-requisites	/
Categorisation, classification	/	Subject area, topic	Group(ing)	/	/	(Decision topic)	/	/	/
Name	/	Name	(in header)	<<key decision>>	Title	(section heading)	/	(in header)	(in header)
State of AD	(not in template, but in standard)	/	Status	/	Status	/	/	Status	/
Impact	(not in template, but in standard)	Implications	Implications	Issues/ Considerations	Consequences	/	to achieve, accepting drawbacks	Expected outcome	to achieve <criteria>, accepting <consequences>
Other entries	Timestamps, Citations	Motivation	Notes, Related principles	Notes, Drivers realised	/	/	/	Strategy, Process, Participants Actions & follow-up	/
Element count	9	13	14	9 (plus 1-2 in header)	5	5 (with 14 questions)	6	14	6
Size	/	not published	(example is half a page)	/	1-2 pages	to be ordered by importance	1 (long) sentence	1-2 pages	1 (long) sentence to 1 page
Publication year	2011	1998	2005	2005	2011	2012	2012	2014	2019

Chapter 6

Student Experiment

During the second month of this research project, a student experiment was conducted. In this experiment, participants used the Decision Story to document their decisions while designing an architecture for a traffic simulation system.

6.1 Setup

The subjects for the experiment are students from Utrecht University following the course Software Architecture. In total there are 10 test groups, most teams consist of 3 students, a few teams consist of 2 students and a few teams of 4 students. The groups received the Irvine experiment assignment from Appendix B.1. For their documentation deliverable, the groups used the template found in Appendix B.2. This template uses Context, Functional and Information viewpoints as main deliverables as the students were familiar with them through the course Software Architecture. The group received instructions in how to use the decision stories from Appendix B.3 and the questionnaire from Appendix B.4. Groups received a general instruction from the researchers on how to proceed with the experiment and were seated in group work spaces across the University building.

6.2 Results

The goal of the experiment was to gain experience using the template and improving the artefact from these results. Thus this section will only analyse the decision stories and questionnaires filled in by the participating students and not their resulting architecture viewpoints.

Decision stories

This section shows the general findings analysing the decision stories produced by the teams:

- The “And decided <decision>” field is just listing one of the options evaluated before and does not add much to the template. In most of these decisions, the rationale is not made explicit. Because of this observation, the template was changed to include the “because <rationale>” element after “and decided <decision>” to remind decision makers to make their rationale explicit.
- Several groups left the concern and criteria fields open in their decision stories. In these decisions it is very hard to understand how they arrived at their decision.
- Some groups used the decision stories to decide how to visualise their viewpoints. These are not really architectural decisions based on the assignment and add little to the deliverable.
- Several groups used trade-off tables to evaluate their options. These decisions are often easier to understand than other decisions.

Questionnaire

This section shows the results of the questionnaire from Appendix B.4.

Table 6.1: Was something unclear in the template or Decision Story? If so, what was unclear and why?

Statement	**# of students**
The template was clear	13
When and how to use the template unclear	4
Decision depth was unclear	2
The template was not clear	2
Template helps the thought process	1
Assignment was unclear	1
Context was confusing at first	1
Facing concern made sentence construction difficult	1
Template did not match thought process	1
Difference between concern and criteria unclear	1
Consider options was sometimes unclear	1
Concern/criteria not always useful	1
Did not use the template	1

Table 6.2: Did you feel like something was missing from the Decision Story? What information do you think should be included into the Decision Story?

Statement	**# of students**
Did not feel something was missing	17
Reasoning why the decision was made	2
Advantages of a decision	2
Assignment requirements unclear	2
Inheritance between decisions	1
Better explanation and what should be included in each stage	1
Did not use the template	1

Table 6.3: Did you fill in the template intuitively or did it lead to confusion?
Please elaborate your answer.

Statement	**# of students**
Intuitively	11
Sometimes confusing, due to lack of time for explanation	5
Sometimes confusing	2
Felt as a natural workflow	1
Confusing	1
We don't think in terms of concerns	1
Disrupted the flow of the assignment	1
When to fill in template unclear	1
Unclear answer	1
Did not use the template	1

Table 6.4: Did the template hinder your discussion and progress or did it guide discussion and progress? Please elaborate your answer.

Statement	**# of students**
Guided discussion	7
Filling in the template took a lot of time	5
Template was used after discussion	3
Unclear answer	3
Neutral, felt that it did not help or hurt	2
Template structured the discussion	2
Made me think of new problems/opportunities	1
Made us consider alternatives and drawbacks	1
Decisions used as a reminder later in the assignment	1
Template did not add much	1
Has to stop discussion to fill in template	1
Filling in the template disrupted flow	1

Table 6.5: Did you fill in the template in the shown order or did you use another order? Do you think the order of the template should be changed?

Statement	**# of students**
Shown order	18
Started with decision, then traced back to how we got there	3
Felt this was the correct order	1
Different order sometimes	1
Sometimes filled in criteria before concern	1
Started with criteria	1
Did not use the template	1
Unclear answer	1

Table 6.6: Any other remarks?

Statement	**# of students**
The assignment is too large for the allocated time	4
Filling in decisions felt distracting	1
Decision stories are a good thing, when used at the right time for decisions that need thoughtful consideration	1

Table 6.7

Chapter 7

Expert Evaluation

The different versions of the Decision Story template were evaluated through expert interviews, case studies and a focus group.

7.1 Case studies

Documented decisions of two projects that were already finished were used for case studies. In these case studies the existing decisions were converted into decision stories by the author of the research project in the visualisations of Figure 5.4b, Figure 5.4d and Figure 5.4e. The resulting decision stories were analysed with the authors of the original decisions and other architects during the second round of interviews. The resulting decision stories (in one visualisation) can be found in Appendix C. Confidential information was removed from the decisions and replaced by grey boxes, what is important to see is the empty fields were the decision was missing information.

7.2 Interviews

In total 2 rounds of interviews were conducted to validate the Decision Story template:

- The first round consisted of 10 interviews. The aim of these interviews was to (i) gather opinions on the template, (ii) identify strengths and weaknesses of the template and (iii) identify potential additions of the template.
- The second round consisted of 9 interviews. All of these interviewees were also interviewed in the first round and were already familiar with the Decision Story. The aim of these interviews was to (i) gather definitions on the different elements

of the Decision Story template, (ii) gather opinions on the different visualisations of the Decision Story and (iii) how the template could be implemented into an organisation.

Results Round 1

The interview protocol for these interviews can be found in Appendix A.1.

What is your opinion of the Decision Story template?

When shown the Decision Story template, Three architects noted that the template contains the aspects they want to see in a decision. Additionally, it can help with the thought process (1 interview).

All architects agree the template is a good way of writing down a decision. Some interviewees had some remarks about the strengths and weaknesses of the template.

Strengths:

- The template allows you to absorb the decision in the same way as it was made. (1 interview)
- The strength of such a template is if everyone uses the same template, incorporate it in the current way of working. (1 interview)
- It is a very structured way of writing down a decision. I think it is good practice to write down these aspects, it forces you to consider your decision and not write down the first thing that comes to mind, as that happens around here very often. It triggers you to think clearly about your decision. (1 interview)
- This template is easily reviewable by someone else. An easy medium for discussion and brainstorming. (1 interview)

Weaknesses:

- If you have a lot of text in the template it might become unreadable, which is one of our current disadvantages of working in excel. (1 interview)
- What needs to be addressed is traceability, otherwise you end up with a lot of decisions where structure is missing. It could also be included in a wiki of some sort. (2 interviews)
- Date/sprint is volatile and part of project documentation, once the project is done, all that is left is product documentation. Date is better suited than sprint, as date is chronological and the sprint has no value after the project is done. (1 interview)

- The template needs to remain practical. It can not stop discussion. It needs to include a sort of filter to only fill in relevant information. (1 interview)

What is very important is that the template forces you to focus: "have I addressed every aspect?" (1 interview). The architect further states: "If the template covers more than my current way of working then I will follow this standard. I look at RCDA in a similar manner: the approach gives tools on how to go through the architecture process but I don't have to strictly follow its documentation templates, but it walks you through the process, which I like." Adding to that statement, the architect notes that the manner someone documents in in not that important to me, as long as the stakeholders and users of that document can use it effectively. "Function is above format in my opinion".

One architect stated: "I think that at a higher level, especially for decision that take a lot of time research, you retain more information/context for your decisions than those at lower levels."

Another architect stated: "I think documenting happens after you made the decision. After you made the decision you can use it as a validation if you have thought of every aspect.I think it is more important that the process of this template is in the heads of the people than using it during decision making itself. For documentation I think it's suitable."

What would you like to add to the template? Is something missing from the template?

Some potential additions to the template are:

- Make the acceptance process explicit. Add the degree of acceptance of your options/decisions and if you need to take action based on this fact. (1 interview)
- A timestamp: when was it accepted, in what manner and who was responsible? (1 interview)
- A link to the requirements for the system. A decision can have a hard relation to one requirement that lead to the decision and have multiple soft relations to other requirements it influences, either positively or negatively. (1 interview)

One architect notes that they are missing some sort of work log. However, the architect thinks the template should not handle this, as it would add too much fluff and is not easy to communicate. Thus this should be addressed in another way.

Furthermore, the interviewees commented on the template:

- There is a point where you should stop documenting decisions and it becomes counterproductive. Even if it is a lightweight template, at some point it becomes too much work. (1 interview)

- Criteria is a hard word to digest. It's almost never used in the industry. (1 interview)
- Drawbacks might be better suited as consequences. Because consequences can be positive as well, drawbacks are not. (1 interviews)
- You want rationale for the option you have chosen but also rationale for the alternatives you did not choose. You can also write down why your alternative is a valid alternative. (1 interview)

Where could you apply this in the current way of working?

For bigger decisions the template could be used as a management summary in an architecture document (1 interview).

Could the template fix the problems in the current way of working?

The template could fix the problem of not knowing the alternatives for a decision. Our alternatives are often multiple pages, writing it down in a concise manner like the template can be included in our current documentation. Including a reference to the extensive documentation for example (1 interview). Additionally, It can be used as a memory support. Instead of reading lots of text again, it could serve as a nice summary to help you remember. (2 interviews).

Results Round 2

The interview protocol for these interviews can be found in Appendix A.2. Of the 9 interviewees, 8 had completed the RCDA training.

Experience using the template

At the beginning of the research project, the interviewees were shown the Decision Story template and asked to try it if they had to make any decisions. Some architects did not use the template due to them not making any decisions during the time between the 2 interviews (5 interviews). Three architects did use the template. One of them remarked they liked it because it structured their thoughts. It helped with stakeholder buy-in during a meeting with their client. Furthermore, using the template did not cause much overhead. Another interviewee noted the template made them more aware of the justification and other aspects of the decision. It also helped during communication with another architect, who was missing something in their decision documentation.

Definitions of Decision Story elements

During the interview, the architects were asked how they would define the different elements of the Decision Story. These were then compared to the definitions of Table 5.1. The result of this analysis is shown in Table 7.1. Consensus was scored as a percentage of agreement to the definitions from Chapter 5. If the architect gave the same definition, there was 100% agreement. If the architect deviated a little from the definition, it was counted as 50% agreement. If the definitions did not match, it was counted as 0% agreement. The consensus column shows the average of all the architects.

Table 7.1: Architect consensus on the Decision Story definitions

Element	**Definition**	**Consensus**
Context	Framework of the problem you are going to talk about. A description of where the problem is situated.	62%
Concern	A compact description of the problem you need to make a decision about, without talking about the solution.	78%
Criteria	What your solution must adhere to, acceptation criteria.	50%
Options	The relevant reasonable alternatives.	100%
Decision	The decision taken.	100%
Rationale	Justification why the chosen option is the best fit on the criteria.	71%
Consequences	Direct effects of the taken decision plus the risks of the decision.	75%

Feedback on the different visualisations

The architects were shown the decisions from the case studies in the visualisations of candidate A2, B2 and C from Figure 5.4. From now on, we will refer to these visualisation as the Presentation format (A2), Sentence format (B2) and Table (C).

Most architects preferred the Presentation format, followed by the Table and then the Sentence. In the Presentation format, its easier to keep structure and is more centred around the alternatives (2 interviews). The Table is useful when working on multiple decisions (1 interview). One architect remarked using a table in an architecture document is not a good idea, but this depends on what the customer expects. Another architect said the Table makes it easy to filter and organise, creating some traceability between decisions. The Sentence format is seen as not clear as structure is missing (2 interviews). One architect said this is the format you will likely see the most in the industry.

In general, the architects agreed that the formats all serve different purposes (4 interviews). The Presentation for a meeting, the Table as work document, and the Sentence as reference. Three architects like the structured approach the template provides. The template also structured their thought process (5 interviews). One architect noted it is important for architects to be complete. It is very useful to remind architects to be complete, and these templates remind them.

Decision management and documentation in the long term

Categorising decisions helps to create an overview for your decisions (3 interviews). One architect noted decisions can be documented in an architecture document with the same decomposition as the system. A search function is very useful as more decisions are documented (2 interviews), as with a lot of decisions there comes the chance of conflicting decisions. It is the job of the architect to detect these conflicts, but it would be helped if the architect is assisted in this as well (1 interview).

One architect said that they often receive systems that are already operational. Most of the time, these systems do not have documentation. In these scenarios, documentation is very valuable. Another architect noted that documentation can give you an idea how you can change the system. For example a decision which says: 'this was the only possible technical solution' vs 'we chose the cheapest solution, but there is money now'. Maintaining documentation is also important, and should be part of the daily work process (1 architect). However, the architect also said maintaining documentation is often one of the first things that gets put aside under time pressure.

Implementation

When asked how they thought the template could be implemented within the organisation, almost all architects agreed it should become a standard within the company (6 interviews). Additionally, three architects noted that clear, open examples make adoption easier. Raising awareness is also important, one architect noted that the fact research is being done on the topic, encouraged them to use the template.

Another avenue that could be used is to incorporate the template into the RCDA approach (3 architects). Two architects noted that this training, which most architects only take once, should be promoted and discussed during workshops more regularly to stay up to date. A yearly update training could also be possibility, although expensive (1 architect).

7.3 Focus Group

In the final month of the research project, a focus group was conducted. During the focus group, architects brainstormed about which business goals the Decision Story could support and what obstacles prevented its adoption and implementation.

Setup

The participants of the focus group are software and solution architects working at CGI The Netherlands. In total 4 architects participated in the focus group. One of them acted as the moderator, directing the group to brainstorm about various topics. The other participants were asked to share their thoughts on the topic. The author and 1st supervisor of this thesis were also present and took notes and participated in the discussion by asking or providing clarifications.

Results

This section first shows the results of the brainstorm concerning the business goals the Decision Story could support. Followed by the identified obstacles and possible solutions to these obstacles. Ending with the discussion about the template and tool support.

The initial hypotheses were that the Decision Story, compared to not using a template to write down decisions, improves the following business goals:

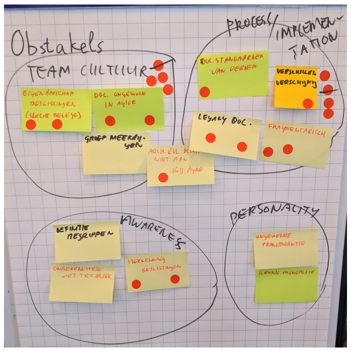
- **Quality of decisions**, by making alternatives explicit
- **Quality of rationale**, by making the justification explicit
- **Non-intrusiveness**, using the template does not cost additional energy or effort
- **Stakeholder buy-in**, by having better justified decisions, stakeholders trust you made the right decision

One architect agreed that the template helps a lot with stakeholder buy-in. They used the Decision Story during a meeting with their client. The architect notes: "The template provides you with a clear story due to its completeness, it led to the meeting being a formality while previously this was never the case with this client, they always gave us homework."

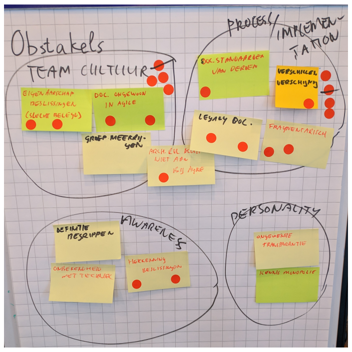
Furthermore, the architects added the following business goals:

- **Speed of decision making**, faster feedback loop due to improved stakeholder buy-in
- **Accessibility of decisions**, standardisation leads to recognisability and more stakeholders know what to expect
- **Transferability of architecture**, when other architects are familiar with the method it is easier to transfer the architecture to other architects.

Obstacles & Possible Solutions



(a) Obstacles



(b) Solutions

Figure 7.1: Obstacles and Solutions from the Focus Group

The architects brainstormed about what obstacles harm adoption of the Decision Story. Related obstacles were grouped resulting in 4 topics of obstacles. Another brainstorm looked at how these obstacles could be mitigated or solved. Afterwards the architects were asked to rank the obstacles, solutions and topics they thought were most important. They received 5 stickers for obstacles, 5 for solutions and 2 for topics. Every sticker is indicated with a '+' next to the topic, obstacle or solution.

The first topic of discussion is ***team culture (+++)*** within the Agile way of working. The architects said that **documenting is unusual in Agile** (++), information is hard to retain in Agile discussions or sessions and there is a lack of documenting. This makes it hard to **get the group on the same page**, as people are still not writing down their rationale. Furthermore, the architects noted that there is a problem with independent Agile teams taking jurisdiction to make decisions instead of the architect. This leads to unclear **ownership of decisions** (++). The architect is responsible for the high level decisions, but the layer underneath is done by these Agile teams who do not or document poorly. As the architect is not involved during this architecture work, the traditional **architect role does not fit within Agile way of working** (+).

A solution to this problem is to **redefine the architect role** (++) within these teams. By having such a template as the Decision Story it becomes easier for other people within the team to come with new alternatives. For example, developers could come with options and bring this to the architect who makes the decision. Other solutions to tackle the team culture obstacle are having someone take up a **pioneering role** (++), someone who enforces the new way of working and helps people adopting it. Further **promoting within the organisation** (+) lets people see the benefit of standardisation and get on the same page.

The next topic of discussion was ***process and implementation (+++)***.The following obstacles are related to the process of using the template or the implementation of the template within an organisation. The architects note that **documenting decisions one-by-one is too fragmented** (++). This is due to having lots of small decisions without a structure. Instead they need some structure to keep an overview. One architect shared their experience working with a client who had a lot of **legacy documentation** (++). It is hard to introduce a new template or architecture documentation if the old one is outdated and lacking information. Similarly, **documentation standards of third parties** (+) force architects to work with other documentation standards. This causes friction with their own way of working. Furthermore, there is a need for **different visualisations** (+): for a workshop a version in PowerPoint, to track progress a version in Excel, and a compact version for in an architecture document. But you do not want to document it 3 times in different locations.

A solution to this is **special tooling** (+), a database or decision register from which you can export in different visualisations. Documentation standards of third parties could also be added as export options. To tackle fragmentation, there needs to be **traceability between models & decisions and decisions & decisions** (++) to show where decisions have an impact on a certain model. To extract decisions from legacy documentation a tool could **automatically detect decisions in text documents through machine learning**. However, a problem with legacy documentation is that these are often incomplete and only include context, criteria and the decision itself, thus this largely depends on the quality of the documentation. Another solution is **manually extracting architectural decisions**, which is a very expensive solution and takes a lot of time, but proven to be effective.

The third topic of discussion is ***awareness***. The architects explained there is a lack of awareness about proper decision documentation across the organisation. One obstacle is **recognising all decisions made** (++). One architect said: "there are large decisions and small decisions, at some point there is a limit where you stop using a template". Sometimes architects make decisions unconsciously and these decisions are not documented. Furthermore, the architects noted that people generally have different ideas or interpretations with **definitions of the template**. This harms template adoption and causes confusion. Architects know how to make and document decisions, but other team members are more likely to be **Unfamiliar with technique**. This causes a gap between architects and other team members who have to work with decisions, but are unfamiliar with how to tackle documentation.

Solutions to a lack of awareness are **training** (+) **and coaching** (++). Training about how to tackle decision documentation is essential, but not enough: coaching people is crucial. Pair this together with good **open examples** (++) of architecture documents to inspire people. This can also help team culture by making people see the benefit of doing it this way. To achieve this, the architects said you need someone who is very good at this and has authority within the organisation, then this will work.

The final topic is related to the ***personality*** of architects. The architects said that there is the idea of the classic architect: these architects do not need to justify themselves and sharing information or how they do their work exposes them or their job. This **knowledge monopoly of architects** can lead to **unwanted transparency** if they are exposed. By using a template like the Decision Story they have to make their information explicit. If they are incomplete or based on incomplete information this will expose them.

Tackling personality can be done by **allowing mistakes (+)**, allowing architects to expose themselves if they do not know the answer. Furthermore stakeholders need to **challenge architects (+)** on what they deliver. If the architect is complete they do not have the be challenged. In addition to coaching, **intervision** can also be a solution, letting people on the same level discuss their problems together.

Template and Tool Support

The architects like that the template forces architects to expose themselves through making explicit what alternatives were not chosen and what can go wrong with the decision. They name (i) standardisation, (ii) recognisability and (iii) the template guiding their process well as advantages of the template. They also name some restriction of the template: (i) scaling, (ii) how to keep the template clear on one page for larger decisions, (iii) the template does not allow for a lot of text and (iv) it is hard to realise in different visualisation. You want to spend as little time as possible adjusting the lay-out of the template. Next the architects discussed possible tool support. They note Enterprise Architect could support the template pretty easily, but you need to be able to export it to some platform so other stakeholders can access it. There is a need for some register which can export to different visualisations, Confluence or JIRA should be able to support this. Tool support could furthermore address findability and searchability problems. The template is very specific to one subject, and there are many decisions like this who have lots of hierarchical relations to each other. Having tool support handle these relations helps combat the obstacle of fragmented decisions.

The architects conclude that the template is not intended for very large decisions that take weeks and for which you build a prototype. Other (larger) templates are better suited for this. This template feels appropriate for projects for smaller decisions which occur often. It is also suited for a management summary for larger decisions and as a means of communication and brainstorming.

Chapter 8

Discussion

The following chapter discusses the most important findings, threats to validity and lessons learned from this research.

8.1 Findings

Both the literature and experts agree documenting rationale is important. The expert evaluation has shown this improves stakeholder buy-in and leads to a higher quality decision. However, there is currently no universal template for capturing rationale (See Section 3.4). Many templates exist in the literature and industry which leads to many architects having their own way of working instead of a standardised template.

There is a need for a standardised template (See Section 4.2). The focus group has shown this improves the accessibility of decisions, which in turn improves the transferability of the architecture. Architects prefer to use their own decision method, their own way of working. This makes it hard to be complete, as everyone regards completeness differently. Thus, architects want their own way of working, but need to write down their decisions uniformly. In conclusion, **this standardised template needs to have minimal ground rules for documentation, but remain flexible in its implementation** (See Section 4.2). Because every project is different in scope, size and team, agreements on how to implement the Decision Story are not set in stone. In every project, agreements needs to be made, for example: What definitions do we adhere to for the different aspects? What level decisions do we document with the Decision Story?

Architects make well-considered decisions but do not document their complete rationale (See Section 7.1). This is shown in the case studies, where existing decision documentation is converted into Decision Stories. These Decision Stories expose which part of the decision process is not explicitly documented by the architect through showing empty elements in the template. Often these architects know these empty elements at the time they made the decision, but did not write it down in their documentation. This can lead to problems later on when decisions need to be revisited and architects only partly remember, or are working somewhere else.

In the beginning of this research we hypothesised the Decision Story would be non-intrusive: using it does not cost extra effort. The interviews show this is not the case. Writing down your decision, in any template, is intrusive when compared to not writing down your decision. However, the decision process of the Decision Story is experienced as non-intrusive because it is perceived as intuitive (See section 7.3). Thus, **the Decision Story is intuitive-intrusive, guiding the architects thought process to be complete**. Additionally, this makes it easier to document this thought process (See Section 7.2).

The template has shown to improve communication, both within teams as with external stakeholders (See section 7.3). This is due to the Decision Story presenting a clear story, forcing architects to be complete in their rationale while remaining easy to digest. This improves stakeholder buy-in and in turn leads to a shorter feedback loop.

Because the Decision Story is written in the context when the decision was made, **a feedback loop is created which allows for two moments of learning**. For new team members as a form of training and for architects to look back at previous decisions. If the context of the decision changes in the future, you can reflect on the decision. An alternative option might become the new decision, the decision might have been the only technically viable option, forcing architects to look elsewhere for solutions.

8.2 Threats to Validity

This section discusses some of the threats to validity present in this research. Internal validity defines how confident you can be that your treatment caused your findings and not something else. External validity are concerns about if the research is generalise to the whole population.

Internal Validity

For both the rounds of interviews the same interview protocol was used and adhered to as best as possible. However, due to the interviews being semi-structured, some interviewees were not able to answer all questions due to time constraints or the question being already partly answered in a previous question. This might have caused some information to be left out.

It is also possible that participants were exposed to selection bias. As the participants were motivated and willing to aid in the research, they might have been more likely to react positively about the Decision Story.

External Validity

The sample of architects used for this research consisted of 10 architects, all from one company and most in the transport and logistics sector. It is not a generalised representation of the whole industry. The interviewed architects have the advantage the company has a dedicated architecture practice with a globally recognised architecture process (RCDA) in place.

Furthermore, the Decision Story was not used during a complete project from beginning to end due to time constraints. Thus it is hard to look at the objective effectiveness of applying the template compared to the current way of working. What this research measured was the perceived usefulness of the Decision Story, rather than its actual use in practice.

8.3 The Decision Story and Business Goals

Figure 8.1 shows the different business goals the Decision Story supports identified in the focus group. The arrows between the business goals indicate their relations to each other and show that one goal influences another (e.g. when quality of rationale improves, stakeholder buy-in improves as well). This section details these relationships, some of which were discussed in the findings section as well.

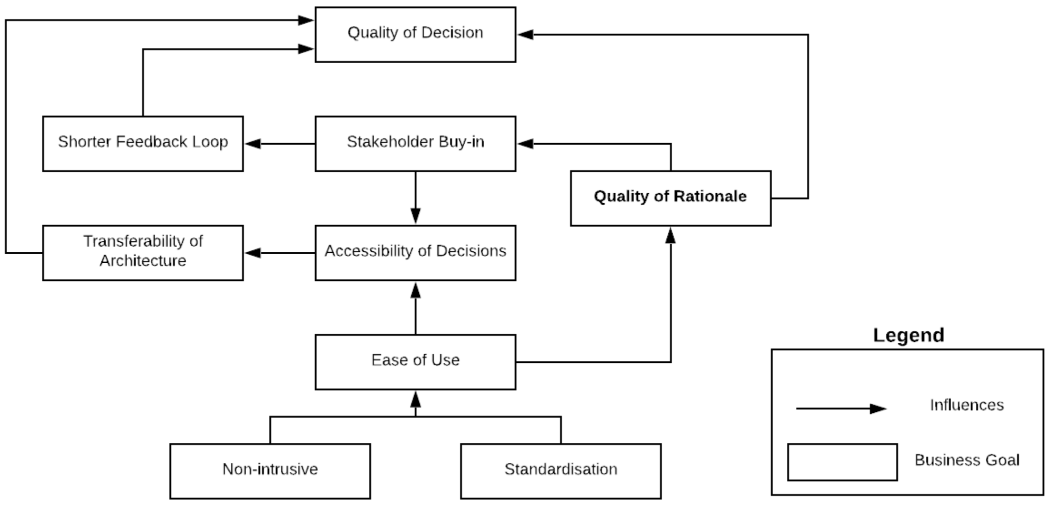


Figure 8.1: Business Goals and their relations

The Decision Story leads to improved ease of use through Standardisation and Non-intrusiveness. The former provides a uniform template with well-defined elements, the latter guides the architects thought process and forces them to be complete. This improved ease of use leads to greater accessibility of decisions, as more stakeholders are familiar with the method, and improved quality of rationale, as the template forces them to be explicit. Improved accessibility makes it easier to transfer the architecture between other architects.

A higher quality rationale has shown to improve the quality of the decision and improve stakeholder buy-in, due to the template being easy to digest and providing a clear story to stakeholders. The improved stakeholder buy-in leads to a shorter feedback loop, stakeholders do not need to challenge architects as they provide a clear story. This in turn improves the overall quality of the decision: it has a short feedback loop, is easily transferable among architects and has a higher quality of rationale.

8.4 Do's and Don'ts: Lessons Learned in Decision Documentation

This section aims to give architects and decision makers concrete tips on how to make better decisions.

Do's

- Be concise in writing down decisions. (see Section 4.2)
- Include alternatives in your rationale. (see Section 4.2)
- Let stakeholders think along with your decisions. (see Section 7.3)
- Use previous decisions as moments for learning. (see Section 8.1)
- Mistakes are moments of learning, not of penalising. (see Section 7.3)
- Make consequences and risks of decisions explicit. (see Section 4.2 and Section 7.3)

Decisions should be compact and easily digestible, both the literature and architects agree very strongly on this. Alternatives to a decision should become first class citizens when talking about decisions. Including alternatives shows why you went in one direction and not the other, and makes your decision more trustworthy to stakeholders. To add to this, let stakeholders of your decisions think along with your decision. Actively involve them and show how you arrived at your decision. As shown in the focus group, this lets meeting become a formality instead of being challenged on your decision. Writing down the context in which your decision was made allows you to reflect on the decision later and learn from it. This also lets you look back at your decision when a mistake is made. However, through being complete and exposing themselves, architects should not be penalised in case a mistake is made but have a moment of learning. Finally, making the consequences and risks of your decisions explicit further improves stakeholder buy-in.

Don'ts

- Be too extensive in documenting alternatives. (see Section 4.2)
- Include volatile attributes in your decision. (see Section 4.2)

An alternative should be used to show why your decision was the right option over the alternative. This does not mean it should be multiple pages long and bigger than your actual decision. If extensive research on another alternative is done, it is best documented as a supplementary document. Volatile attributes (such as status, sprint, log) are best left separated from documented decisions. This is all part of project documentation, not product documentation. They are unlikely to be relevant once a system is operational.

Chapter 9

Conclusion & Future Work

This research has shown that the Decision Story is a valid template for decision documentation in Software Architecture. The Decision Story is able to show where shortcomings occur in the decision process by providing insight which part of the decision process the decision maker has not explicitly documented.

9.1 Answer to Research Questions

With the results from the literature review, experiment, case studies, interviews and focus group, the research questions can now be answered. As mentioned before, within this research, *effective* means efficient (as a byproduct, little overhead, non intrusive), concise (not more elaborate than needed) and valuable (fulfils the needs of its stakeholders, leads to better decisions).

Sub Research Questions

SRQ 1: What decision documentation templates exist?
This research question was answered through expert interviews and the literature review in Chapter 3. Table 3.1 shows several decision documentation templates used in the industry. The comparison shows the templates range from small (one sentence) to large (multiple pages) and vary greatly in their contained aspects. The aspects which are present in most templates are: Outcome, Rationale, Decision drivers and Impact. The expert interviews show that even though the organisation has a standard in place for tackling decision documentation, architects still prefer to use their own method for documenting decisions. From these results we can conclude there is still no consensus on a universal decision template, both in the research and the industry.

SRQ 2: What are requirements to document decisions effectively?
Requirements were gathered from Software Architecture literature as well as through interviews with experts. This research question can be answered through Table 4.1, in which 19 requirements are detailed. The requirements from the expert interviews were combined with the requirements from the literature review and given a rating to indicate the degree of support from the architects.

SRQ 3: What are templates for documenting decisions that stimulate architects to capture rationale?
This research shows incorporating the decision process in decision documentation templates stimulates architects to be complete in their rationale capture. Several templates that incorporate this decision process have been created in Chapter 5, which can be used in different circumstances depending on the need of the decision maker. This research shows that there is no universal template which can be used in every situation. There is a need for different visualisations: one for meetings and workshops, one for tracking progress and one for an architecture document. Furthermore, the content of these templates need to be able to be modified easily depending on the need of the architect and/or project. The Decision Story is a suitable template for this purpose. It has ground rules for documentation that stimulate rationale capture, while remaining flexible to be moulded depending on the needs of its environment.

SRQ 4: How effective is this designed template?
This research question can be answered through the evaluation by experts of the designed Decision Story template. The Decision Story has little overhead in its use due to being intuitive-intrusive: similar to the decision process of architects. The resulting Decision Story emerges as a byproduct of the decision process. Furthermore, the Decision Story is compact and easy to digest. It contains all the elements architects expect in a decision and not more. Finally, the Decision Story has shown to improve stakeholder buy-in and improve the quality of decisions by exposing gaps in architect's rationale documentation. In conclusion, the Decision Story fulfils our requirements of being an effective template.

Main Research Question

With the answers to our sub research questions, we can now answer the main research question.

MRQ: How can design rationale of decisions be captured effectively?
The answer to the main research question is presented in the form of the Decision Story. A documentation template which evolved from the need for a flexible, standardised template and improved upon through thorough evaluation with experts in the industry. The Decision Story captures the rationale aspects architects deem most important, is easy in its use due to being compact and intuitive, can be used in different visualisations, is easy to be modified to specific projects, fulfils the needs of stakeholders and leads to a higher quality decision. These characteristics make the Decision Story a viable method to effectively capture design rationale of decisions.

9.2 Limitations

The focus of this research was on the outcome of the decisions: How do you effectively capture and document decisions? What are pitfalls in decision documentation? How do you take the step from implicit to explicit on paper and what is important to document explicitly?

This research did not focus on how decision making can be improved. Decision making by architects requires a lot of knowledge and experience encountering different situations and was out of scope for this thesis. However, an unintended by-effect of the Decision Story is that the presentation format has shown to improve this by guiding the decision making process.

Furthermore, decision management was not in scope in this research. After making a decision and writing it down it needs to be stored somewhere, ideally with other decisions. Management of decisions can be tackled in many ways and largely depends on the nature of the project. This is something that could be researched in future work.

9.3 Future work

This section outlines some directions for future work.

Full Case Study

Due to time constraints the Decision Story was unable to be used during a complete project as the sole method of documenting decisions. Using it during entire projects gives more insight into the effectiveness of the Decision Story and the resulting documentation.

Decision Management

How should decisions be stored in a project? How do you link them to each other? These questions were out of scope during this research project. Further research could look how best to tackle these issues.

Furthermore, future work could look at the lifespan of decisions. Some projects or systems are in operation a very long time. When maintaining these systems, decisions are changed due to no longer being relevant, new technology, cheaper alternatives etc. Showing the lifespan of these decisions and the resulting documentation can give insight in what information is needed at those moments and in turn taken into account to improve decision making early on in those projects.

The Decision Story as Architecture Summary

The Decision Story is a suitable template for most decisions an architects makes, but for larger decisions it is too small. For these larger decisions it could be used as a management summary summarising the decision. This can be applied to whole architecture documents to create an architecture summary. For example to serve as a memory aid for architects or get stakeholders up to speed.

Decision Story Tool Support

During the focus group several tooling options were discussed which could make it easier for the Decision Story to be incorporated in the daily way of working. One of the topics that kept popping up was traceability: how do you link these decisions to each other and to models? Future research could look into how this could be added to tools currently popular in the industry or via special tooling.

References

Arnott, D. (2006). Cognitive biases and decision support systems development: a design science approach. *Information Systems Journal*, *16*(1), 55–78.

Babar, M. A., Dingsøyr, T., Lago, P., & Van Vliet, H. (2009). *Software architecture knowledge management.* Springer.

Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice.* Addison-Wesley Professional.

Bosch, J. (2004). Software architecture: The next step. In *European workshop on software architecture* (pp. 194–199).

Burge, J., & Brown, D. C. (1998). Design rationale types and tools.

Buss, D. M. (2005). *The handbook of evolutionary psychology.* Wiley Online Library.

Calikli, G., Bener, A., & Arslan, B. (2010). An analysis of the effects of company culture, education and experience on confirmation bias levels of software developers and testers. In *2010 acm/ieee 32nd international conference on software engineering* (Vol. 2, pp. 187–190).

Capilla, R., Jansen, A., Tang, A., Avgeriou, P., & Babar, M. A. (2016). 10 years of software architecture knowledge management: Practice and future. *Journal of Systems and Software*, *116*, 191–205.

Clements, P., Garlan, D., Bass, L., Stafford, J., Nord, R., Ivers, J., & Little, R. (2002). *Documenting software architectures: views and beyond.* Pearson Education.

Conklin, E. J., & Yakemovic, K. B. (1991). A process-oriented approach to design rationale. *Human–Computer Interaction*, *6*(3-4), 357–391.

Dane, E., & Pratt, M. G. (2009). Conceptualizing and measuring intuition: A review of recent trends. *International review of industrial and organizational psychology*, *24*, 1–40.

De Jong, P. (2017). Master research thesis.

Dietrich, C. (2010). Decision making: factors that influence decision making, heuristics used, and decision outcomes. *Inquiries Journal*, *2*(02).

Easterbrook, S., Singer, J., Storey, M.-A., & Damian, D. (2008). Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering* (pp. 285–311). Springer.

Eling, K., Griffin, A., & Langerak, F. (2014). Using intuition in fuzzy front-end decision-making: A conceptual framework. *Journal of Product Innovation Management*, *31*(5), 956–972.

Fairbanks, G. (2010). *Just enough software architecture: a risk-driven approach.* Marshall & Brainerd.

Falessi, D., Briand, L. C., Cantone, G., Capilla, R., & Kruchten, P. (2013). The value of design rationale information. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *22*(3), 21.

Farenhorst, R., Lago, P., & Van Vliet, H. (2007). Effective tool support for architectural knowledge sharing. In *European conference on software architecture* (pp. 123–138).

Gigerenzer, G., & Selten, R. (2002). *Bounded rationality: The adaptive toolbox.* MIT press.

Goldschmidt, G., & Weil, M. (1998). Contents and structure in design reasoning. *Design issues*, *14*(3), 85–100.

Gruber, T. R., & Russell, D. M. (1991). *Design knowledge and design rationale: A framework for representation, capture, and use.* Knowledge Systems Laboratory, Computer Science Department, Stanford University.

Janis, I. L. (2008). Groupthink. *IEEE Engineering Management Review*, *36*(1), 36.

Jansen, A. (2008). *Architectural design decisions.* University Library of Groningen.

Jansen, A., & Bosch, J. (2005). Software architecture as a set of architectural design decisions. In *5th working ieee/ifip conference on software architecture (wicsa'05)* (pp. 109–120).

Kahneman, D. (2011). *Thinking, fast and slow* (Vol. 1). Farrar, Straus and Giroux New York.

Kopp, O., Armbruster, A., & Zimmermann, O. (2018). Markdown architectural decision records: Format and tool support. In *Zeus* (pp. 55–62).

Kruchten, P. (1995). The 4+ 1 view model of architecture. *IEEE software*, *12*(6), 42–50.

Kruchten, P., Lago, P., & Van Vliet, H. (2006). Building up and reasoning about architectural knowledge. In *International conference on the quality of software architectures* (pp. 43–58).

Lee, J. (1997). Design rationale systems: understanding the issues. *IEEE expert*, *12*(3), 78–85.

Methorst, M. (2019). Master research thesis.

Mohanani, R., Ralph, P., & Shreeve, B. (2014). Requirements fixation. In *Proceedings of the 36th international conference on software engineering* (pp. 895–906).

Morgan, D. L. (1996). *Focus groups as qualitative research* (Vol. 16). Sage publications.

Nonaka, I., & Takeuchi, H. (1995). *The knowledge-creating company: How japanese companies create the dynamics of innovation.* Oxford university press.

Nygard, M. (2011). *Documenting architecture decisions.* Retrieved from `http://thinkrelevance.com/blog/2011/11/15/documenting-architecture-decisions`

Oswald, M. E., & Grosjean, S. (2004). Confirmation bias. *Cognitive illusions: A handbook on fallacies and biases in thinking, judgement and memory*, *79*.

Parnas, D. L., & Clements, P. C. (1986). A rational design process: How and why to fake it. *IEEE transactions on software engineering*(2), 251–257.

Petre, M., & Van Der Hoek, A. (2013). *Software designers in action: A human-centric look at design work.* Chapman and Hall/CRC.

Poort, E. (2014). *Rcda: Risk-and cost-driven architecture: A solution architect's handbook* (Tech. Rep.). Technical report, CGI.

Pretorius, C., Razavian, M., Eling, K., & Langerak, F. (2018). Towards a dual processing perspective of software architecture decision making. In *2018 ieee international conference on software architecture companion (icsa-c)* (pp. 48–51).

Razavian, M., Tang, A., Capilla, R., & Lago, P. (2016). In two minds: how reflections influence software design thinking. *Journal of Software: Evolution and Process*, *28*(6), 394–426.

Rekhav, V. S., & Muccini, H. (2014). A study on group decision-making in software architecture. In *2014 ieee/ifip conference on software architecture* (pp. 185–194).

Rozanski, N., & Woods, E. (2011). *Software systems architecture: working with stakeholders using viewpoints and perspectives.* Addison-Wesley.

Schriek, C. (2016). Master research thesis.

Schriek, C., van der Werf, J. M. E., Tang, A., & Bex, F. (2016). Software architecture design reasoning: A card game to help novice designers. In *European conference on software architecture* (pp. 22–38).

Snyder, L. G., & Snyder, M. J. (2008). Teaching critical thinking and problem solving skills. *The Journal of Research in Business Education*, *50*(2), 90.

Tang, A. (2011). Software designers, are you biased? In *Proceedings of the 6th international workshop on sharing and reusing architectural knowledge* (pp. 1–8).

Tang, A., Aleti, A., Burge, J., & van Vliet, H. (2010). What makes software design effective? *Design Studies*, *31*(6), 614–640.

Tang, A., Avgeriou, P., Jansen, A., Capilla, R., & Babar, M. A. (2010). A comparative study of architecture knowledge management tools. *Journal of Systems and Software*, *83*(3), 352–370.

Tang, A., Babar, M. A., Gorton, I., & Han, J. (2006). A survey of architecture design rationale. *Journal of systems and software*, *79*(12), 1792–1804.

Tang, A., Lago, P., et al. (2010). Notes on design reasoning techniques.

Tang, A., Tran, M. H., Han, J., & Van Vliet, H. (2008). Design reasoning improves software design quality. In *International conference on the quality of software architectures* (pp. 28–42).

Tang, A., & Van Vliet, H. (2009). Software architecture design reasoning. In *Software architecture knowledge management* (pp. 155–174). Springer.

Tang, A., & van Vliet, H. (2015). Software designers satisfice. In *European conference on software architecture* (pp. 105–120).

Tofan, D., Galster, M., & Avgeriou, P. (2013). Difficulty of architectural decisions–a survey with professional architects. In *European conference on software architecture* (pp. 192–199).

Tversky, A., & Kahneman, D. (1981). The framing of decisions and the psychology of choice. *Science*, *211*(4481), 453–458.

Tyree, J., & Akerman, A. (2005). Architecture decisions: Demystifying architecture. *IEEE software*, *22*(2), 19–27.

Van Vliet, H., & Tang, A. (2016). Decision making in software architecture. *Journal of Systems and Software*, *117*, 638–644.

Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering.* Springer.

Zannier, C., Chiasson, M., & Maurer, F. (2007). A model of design decision making based on empirical results of interviews with software designers. *Information and Software Technology*, *49*(6), 637–653.

Zdun, U., Capilla, R., Tran, H., & Zimmermann, O. (2013). Sustainable architectural design decisions. *IEEE software*, *30*(6), 46–53.

Zimmermann, O. (2014). *Software architecture in workflow-design.* Retrieved from `https://tinyurl.com/tvppnop`

Zimmermann, O., Wegmann, L., Koziolek, H., & Goldschmidt, T. (2015). Architectural decision guidance across projects-problem space modeling, decision backlog management and cloud computing knowledge. In *2015 12th working ieee/ifip conference on software architecture* (pp. 85–94).

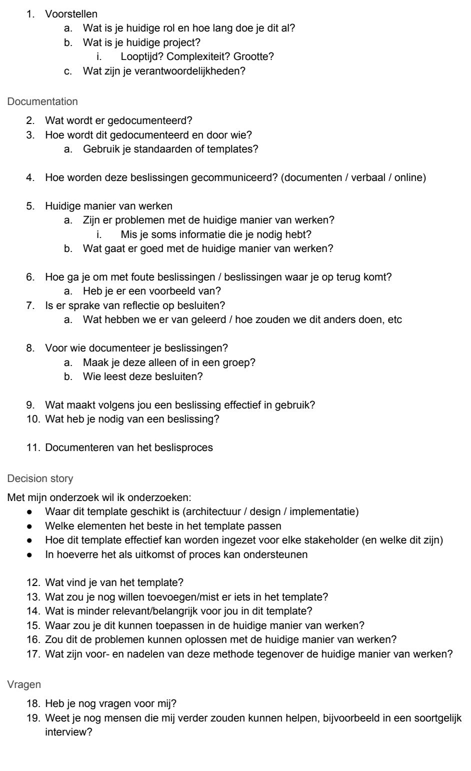
Zsambok, C. E., & Klein, G. (2014). *Naturalistic decision making.* Psychology Press.

Appendices

Appendix A

Interview Protocols

A.1 Elicitation Interview Protocol



1. Voorstellen
    a. Wat is je huidige rol en hoe lang doe je dit al?
    b. Wat is je huidige project?
        i. Looptijd? Complexiteit? Grootte?
    c. Wat zijn je verantwoordelijkheden?

Documentation

2. Wat wordt er gedocumenteerd?
3. Hoe wordt dit gedocumenteerd en door wie?
    a. Gebruik je standaarden of templates?

4. Hoe worden deze beslissingen gecommuniceerd? (documenten / verbaal / online)

5. Huidige manier van werken
    a. Zijn er problemen met de huidige manier van werken?
        i. Mis je soms informatie die je nodig hebt?
    b. Wat gaat er goed met de huidige manier van werken?

6. Hoe ga je om met foute beslissingen / beslissingen waar je op terug komt?
    a. Heb je er een voorbeeld van?
7. Is er sprake van reflectie op besluiten?
    a. Wat hebben we er van geleerd / hoe zouden we dit anders doen, etc

8. Voor wie documenteer je beslissingen?
    a. Maak je deze alleen of in een groep?
    b. Wie leest deze besluiten?

9. Wat maakt volgens jou een beslissing effectief in gebruik?
10. Wat heb je nodig van een beslissing?

11. Documenteren van het beslisproces

Decision story

Met mijn onderzoek wil ik onderzoeken:
- Waar dit template geschikt is (architectuur / design / implementatie)
- Welke elementen het beste in het template passen
- Hoe dit template effectief kan worden ingezet voor elke stakeholder (en welke dit zijn)
- In hoeverre het als uitkomst of proces kan ondersteunen

12. Wat vind je van het template?
13. Wat zou je nog willen toevoegen/mist er iets in het template?
14. Wat is minder relevant/belangrijk voor jou in dit template?
15. Waar zou je dit kunnen toepassen in de huidige manier van werken?
16. Zou dit de problemen kunnen oplossen met de huidige manier van werken?
17. Wat zijn voor- en nadelen van deze methode tegenover de huidige manier van werken?

Vragen

18. Heb je nog vragen voor mij?
19. Weet je nog mensen die mij verder zouden kunnen helpen, bijvoorbeeld in een soortgelijk interview?

A.2 Evaluation Interview Protocol

Wat is je voorkennis van RCDA? Waar zet je het in tijdens je werk?

Gebruik van het template
Heb je het template nog gebruikt?

- Wat heeft je ervan weerhouden het niet te gebruiken?
- Hoe was het in gebruik?

Algemene definities van de verschillende vlakken.
Titel:
Context:
Concern:
Criteria:
Options:
Decision:
Rationale:
Consequences:

Beoordeel de verschillende vormen op een schaal van 1-10:

1. Overzichtelijkheid
2. Intentie om het te gebruiken ("the degree to which a person believes that using a particular system would enhance his or her job performance")
3. Bruikbaarheid ("the degree to which a person believes that using a particular system would be free from effort")

Algemene feedback verschillende versies van het template

1. Feedback over het template in zinsvorm
2. Feedback over het template in tabelvorm
3. Feedback over het template in presentatievorm

Decision management
Het template kan worden ingezet voor verschillende doelen: als documentatie van een beslissing, als proces om een beslissing te maken of als een tool om decision management aan te pakken.

1. Wat is je mening over het template als puur naslagwerk?
2. Wat is je mening over het template als beslisproces (ook al is het tacit)
3. Wat is je mening over het template als tool voor decision management?

Hoe kun je omgaan met de organisatie van alle beslissingen in een project? Dit is minder belangrijk als er maar een paar beslissingen zijn, maar wat als er bijvoorbeeld 50 beslissingen zijn? Hoe pak je gelaagdheid van verschillende beslissingen aan?

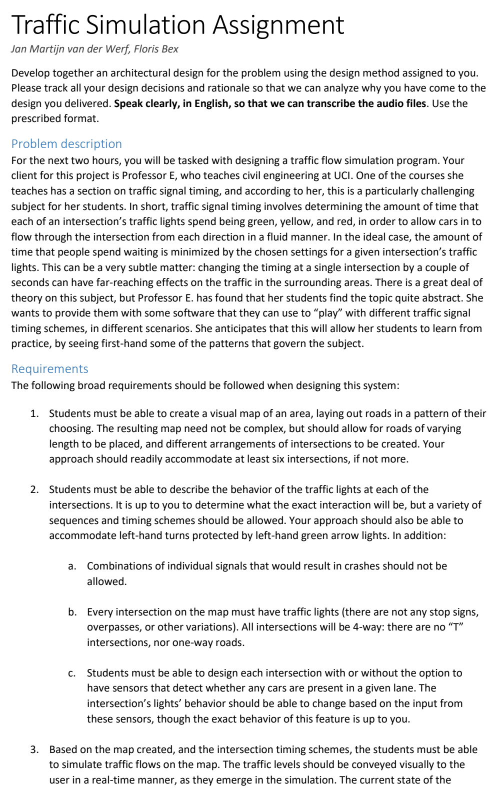
Wat is de impact van documentatie op de lange termijn, in mate van onderhoudbaarheid en waarde zodra het systeem operationeel is. Hoe moet dit worden aangepakt?

Hoe zou je adviseren om het template te implementeren binnen een project, hoe zou je dit kunnen doen binnen de organisatie?

Appendix B

Irvine Experiment

B.1 Assignment



Traffic Simulation Assignment

Jan Martijn van der Werf, Floris Bex

Develop together an architectural design for the problem using the design method assigned to you. Please track all your design decisions and rationale so that we can analyze why you have come to the design you delivered. **Speak clearly, in English, so that we can transcribe the audio files**. Use the prescribed format.

Problem description

For the next two hours, you will be tasked with designing a traffic flow simulation program. Your client for this project is Professor E, who teaches civil engineering at UCI. One of the courses she teaches has a section on traffic signal timing, and according to her, this is a particularly challenging subject for her students. In short, traffic signal timing involves determining the amount of time that each of an intersection's traffic lights spend being green, yellow, and red, in order to allow cars in to flow through the intersection from each direction in a fluid manner. In the ideal case, the amount of time that people spend waiting is minimized by the chosen settings for a given intersection's traffic lights. This can be a very subtle matter: changing the timing at a single intersection by a couple of seconds can have far-reaching effects on the traffic in the surrounding areas. There is a great deal of theory on this subject, but Professor E. has found that her students find the topic quite abstract. She wants to provide them with some software that they can use to "play" with different traffic signal timing schemes, in different scenarios. She anticipates that this will allow her students to learn from practice, by seeing first-hand some of the patterns that govern the subject.

Requirements

The following broad requirements should be followed when designing this system:

1. Students must be able to create a visual map of an area, laying out roads in a pattern of their choosing. The resulting map need not be complex, but should allow for roads of varying length to be placed, and different arrangements of intersections to be created. Your approach should readily accommodate at least six intersections, if not more.

2. Students must be able to describe the behavior of the traffic lights at each of the intersections. It is up to you to determine what the exact interaction will be, but a variety of sequences and timing schemes should be allowed. Your approach should also be able to accommodate left-hand turns protected by left-hand green arrow lights. In addition:

    a. Combinations of individual signals that would result in crashes should not be allowed.

    b. Every intersection on the map must have traffic lights (there are not any stop signs, overpasses, or other variations). All intersections will be 4-way: there are no "T" intersections, nor one-way roads.

    c. Students must be able to design each intersection with or without the option to have sensors that detect whether any cars are present in a given lane. The intersection's lights' behavior should be able to change based on the input from these sensors, though the exact behavior of this feature is up to you.

3. Based on the map created, and the intersection timing schemes, the students must be able to simulate traffic flows on the map. The traffic levels should be conveyed visually to the user in a real-time manner, as they emerge in the simulation. The current state of the

intersections' traffic lights should also be depicted visually, and updated when they change. It is up to you how to present this information to the students using your program. For example, you may choose to depict individual cars, or to use a more abstract representation.

4. Students should be able to change the traffic density that enters the map on a given road. For example, it should be possible to create a busy road, or a seldom used one, and any variation in between. How exactly this is declared by the user and depicted by the system is up to you. Broadly, the tool should be easy to use, and should encourage students to explore multiple alternative approaches. Students should be able to observe any problems with their map's timing scheme, alter it, and see the results of their changes on the traffic patterns. This program is not meant to be an exact, scientific simulation, but aims to simply illustrate the basic effect that traffic signal timing has on traffic. If you wish, you may assume that you will be able to reuse an existing software package that provides relevant mathematical functionality such as statistical distributions, random number generators, and queuing theory.

You may add additional features and details to the simulation, if you think that they would support these goals.

Your design will primarily be evaluated based on its elegance and clarity – both in its overall solution and envisioned implementation structure.

Desired Outcomes

Your work on this design should focus on two main issues:

1. You must design the interaction that the students will have with the system. You should design the basic appearance of the program, as well as the means by which the user creates a map, sets traffic timing schemes, and views traffic simulations.
2. You must design the basic structure of the code that will be used to implement this system. You should focus on the important design decisions that form the foundation of the implementation, and work those out to the depth you believe is needed.

Deliver an architecture document containing at least the context, functional and information viewpoints, according to the provided architecture document template. Remember that a viewpoint may consist of multiple views.

The result of this session should be: *the ability to present your design to a team of software developers who will be tasked with actually implementing it.* The level of competency you can expect is that of students who just completed a basic computer science or software engineering undergraduate degree. You do not need to create a complete, final diagram to be handed off to an implementation team. But you should have an understanding that is sufficient to explain how to implement the system to competent developers, without requiring them to make many high-level design decisions on their own.

To simulate this hand-off, you will be asked to make a documentation of the model you have developed, including a glossary, description and rationale.

Timeline

• 2 hours:	design session (5 minutes break is allowed.)
• 45 minutes:	Make documentation. Do not change your architecture anymore!
• 5 minutes:	fill in the provided questionnaire.

B.2 Documentation template

SOFTWARE ARCHITECTURE OF A TRAFFIC SIMULATION SYSTEM

Group	XX (for anonymity, please only provide your group number)

PRODUCT INTRODUCTION

Describe in a few words (about a parapgraph) on what the product to be designed is about.

Please extend this document with all your views. Remember that a viewpoint may contain many different views!

Good luck with the assignments!

CONTEXT VIEWPOINT

Use the following template for each of the views in the context viewpoint:

VIEW: <NAME>

MODEL

Place here the model representation of the view

DESCRIPTION

Short description of the view

GLOSSARY OF ELEMENTS

Give a description for each of the elements in the view

Id	Name	Description

RATIONALE

Describe why the view is as it is. Notice that we want an argumentation about the view, and not an argumentation of why the view is included in the architecture description.

FUNCTIONAL VIEWPOINT

Use the following template for each of the views in the functional viewpoint:

VIEW: <NAME>

MODEL

Place here the model representation of the view

DESCRIPTION

Short description of the view

GLOSSARY OF ELEMENTS

Give a description for each of the elements in the view

Id	Name	Description

RATIONALE

Describe why the view is as it is. Notice that we want an argumentation about the view, and not an argumentation of why the view is included in the architecture description.

INFORMATION VIEWPOINT

Use the following template for each of the views in the functional viewpoint:

VIEW: <NAME>

MODEL

Place here the model representation of the view

DESCRIPTION

Short description of the view

GLOSSARY OF ELEMENTS

Give a description for each of the elements in the view

Id	Name	Description

RATIONALE

Describe why the view is as it is. Notice that we want an argumentation about the view, and not an argumentation of why the view is included in the architecture description.

B.3 Decision stories Handout

Assignment: Design an architecture

Together, develop an architectural design for the problem using the design method assigned to you. Please track all your design decisions and rationale so that we can analyze why you have come to the design you delivered. Use the prescribed format.

The elements of the template
For designing an architecture you are asked to use a decision story template to help with your discussion and documentation. This template is meant to prompt discussion and support collaboration by providing elements to consider. The process of creating a decision story can be seen in Figure 1. When read from left to right, the design decision will be summarized in one sentence, like a user story.

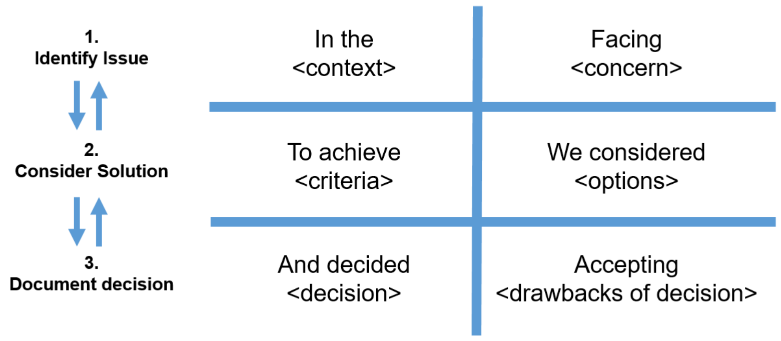


Figure 1: Process of making a decision story

The template is split up into three design activities to help make design decisions. After every activity, **all designers should agree** on the result of the activity.

1. Identify Issue
   - ➢ In what context are we working and what concern needs to be solved?
2. Consider Solution
   - ➢ What are you trying to achieve with your solution and what options can you think of that meet this criteria?
3. Document Decision
   - ➢ What option did you choose and what are the drawbacks of choosing this option?

Sometimes you will need to revisit earlier activities. For example, it can become clear that what you are trying to achieve is not addressing the correct design issue or that your solution is not meeting all your defined criteria. In these cases, return to the earlier activity and clarify the design issue/criteria.

Documentation
It is important to log the design decisions being made and the rationale behind them as these are also a part of the documentation. The decisions made during your design discussion can be used to support your reasoning.

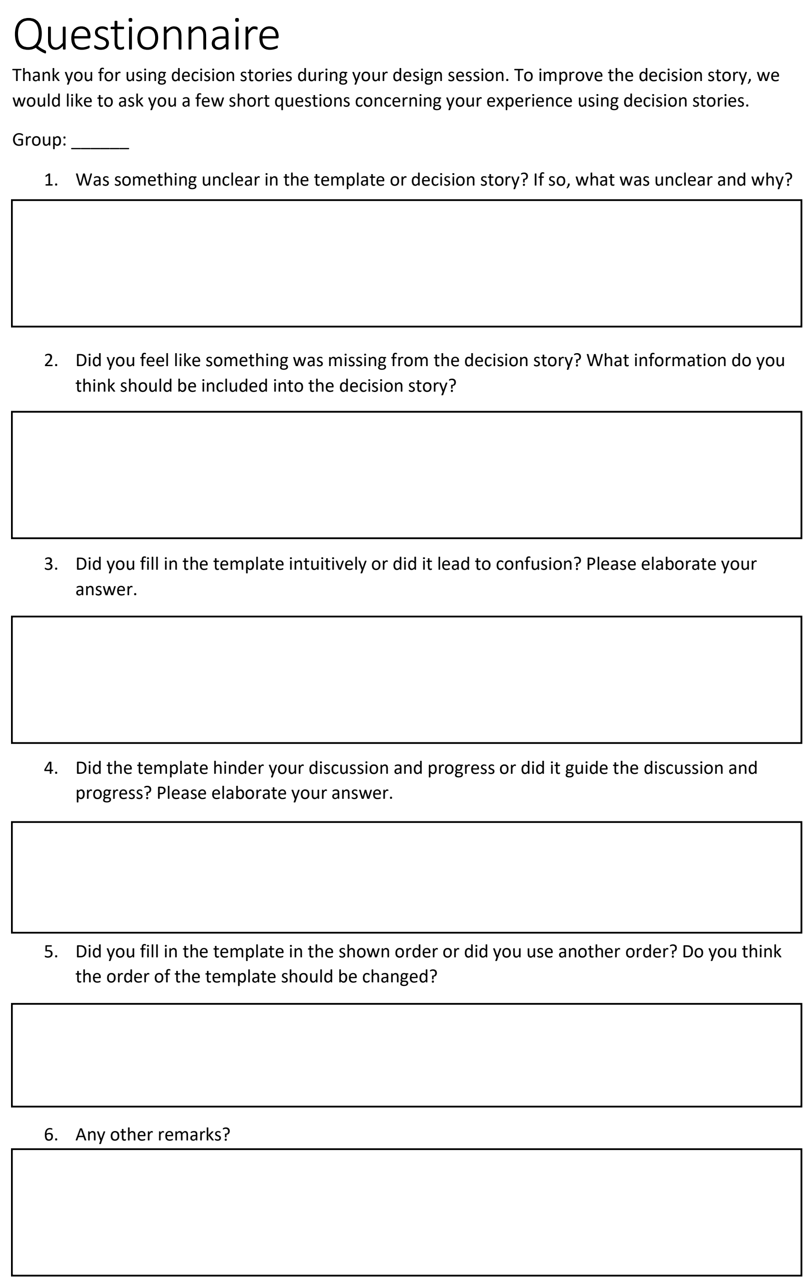
For this purpose we ask that you **record** and **document** your session. This means that you need to **speak clearly** during the discussion and think aloud and state what you are doing.

As you will be referring to the models you're designing you should also document your design progression. You can make pictures or save the designs of your developing model and refer to them during designing as model context, functional, information etc., to show the order of your design progression and make it clear on the recording what you are discussing.

Record the minutes and the seconds from the time that you start designing, you can use a stopwatch or your recording time for this purpose. Please use the **format** on the next page to document your decisions. Give every decision an ID when you start working on it and timestamp it when the template is complete. If you need to reject a previous decision, please indicate so in the top right corner and timestamp when it was rejected.

When documenting your architecture and rationale, your documented decisions can be used to support your rationale. Remember that you can use the decisions recorded during the design session.

B.4 Questionnaire



Questionnaire

Thank you for using decision stories during your design session. To improve the decision story, we would like to ask you a few short questions concerning your experience using decision stories.

Group: ______

1. Was something unclear in the template or decision story? If so, what was unclear and why?

2. Did you feel like something was missing from the decision story? What information do you think should be included into the decision story?

3. Did you fill in the template intuitively or did it lead to confusion? Please elaborate your answer.

4. Did the template hinder your discussion and progress or did it guide the discussion and progress? Please elaborate your answer.

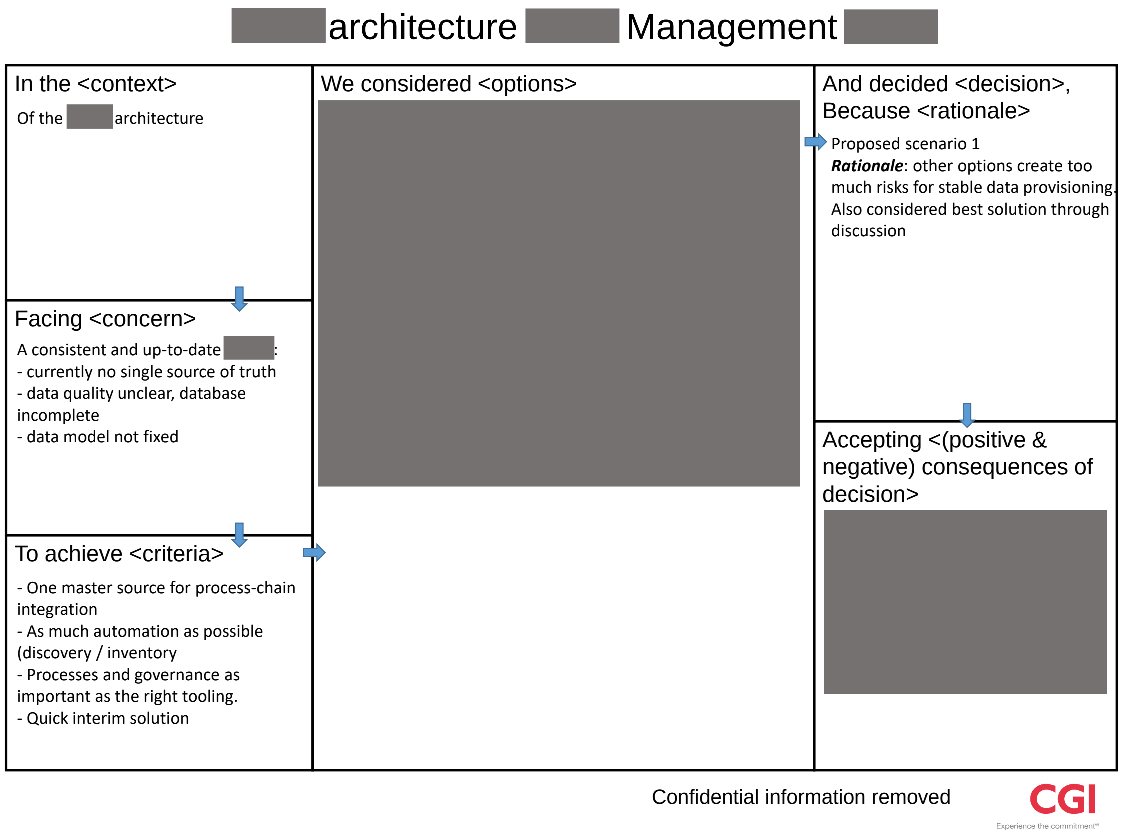
5. Did you fill in the template in the shown order or did you use another order? Do you think the order of the template should be changed?

6. Any other remarks?

Appendix C

Case Studies

C.1 Case Study 1



[redacted] architecture [redacted] Management [redacted]

In the <context>

Of the [redacted] architecture

Facing <concern>

A consistent and up-to-date [redacted]:
- currently no single source of truth
- data quality unclear, database incomplete
- data model not fixed

To achieve <criteria>

- One master source for process-chain integration
- As much automation as possible (discovery / inventory
- Processes and governance as important as the right tooling.
- Quick interim solution

We considered <options>

[redacted]

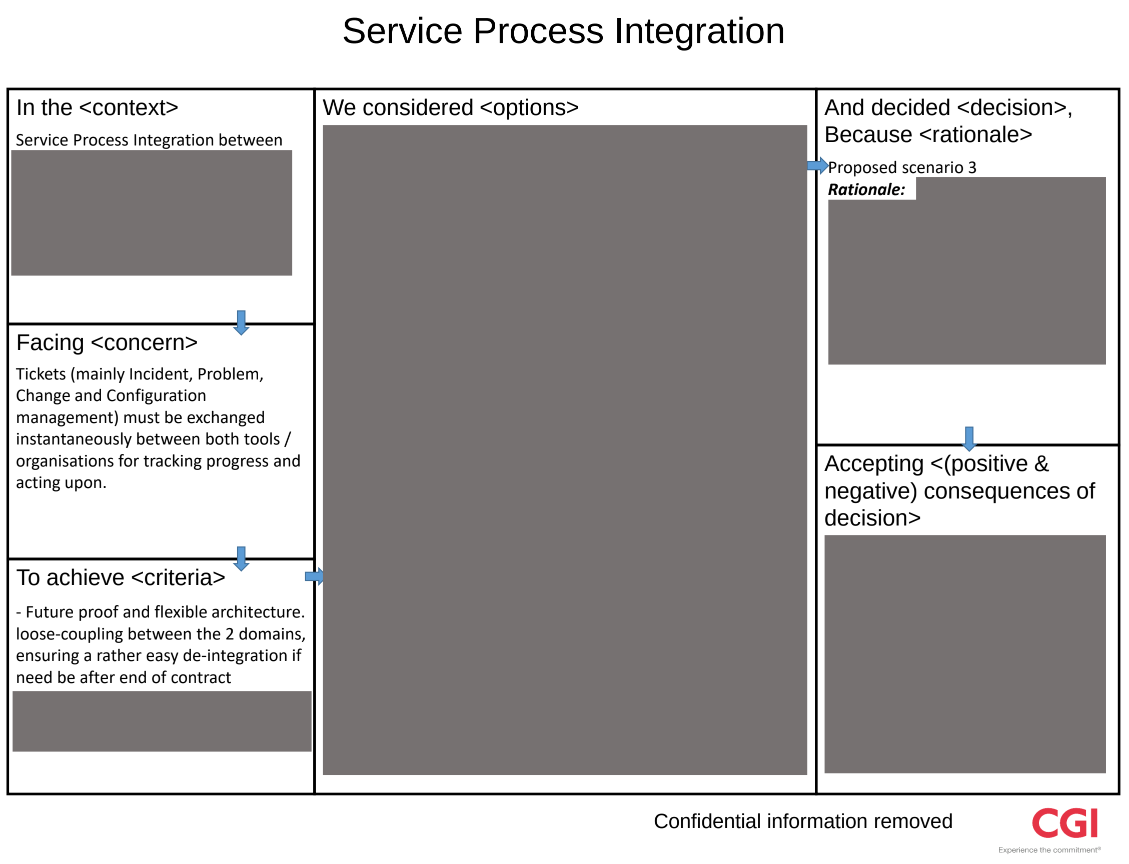
And decided <decision>, Because <rationale>

Proposed scenario 1
Rationale: other options create too much risks for stable data provisioning. Also considered best solution through discussion

Accepting <(positive & negative) consequences of decision>

[redacted]

Confidential information removed



Service Process Integration

In the <context>

Service Process Integration between

[redacted]

Facing <concern>

Tickets (mainly Incident, Problem, Change and Configuration management) must be exchanged instantaneously between both tools / organisations for tracking progress and acting upon.

To achieve <criteria>

- Future proof and flexible architecture. loose-coupling between the 2 domains, ensuring a rather easy de-integration if need be after end of contract

[redacted]

We considered <options>

[redacted]

And decided <decision>, Because <rationale>

Proposed scenario 3

Rationale:

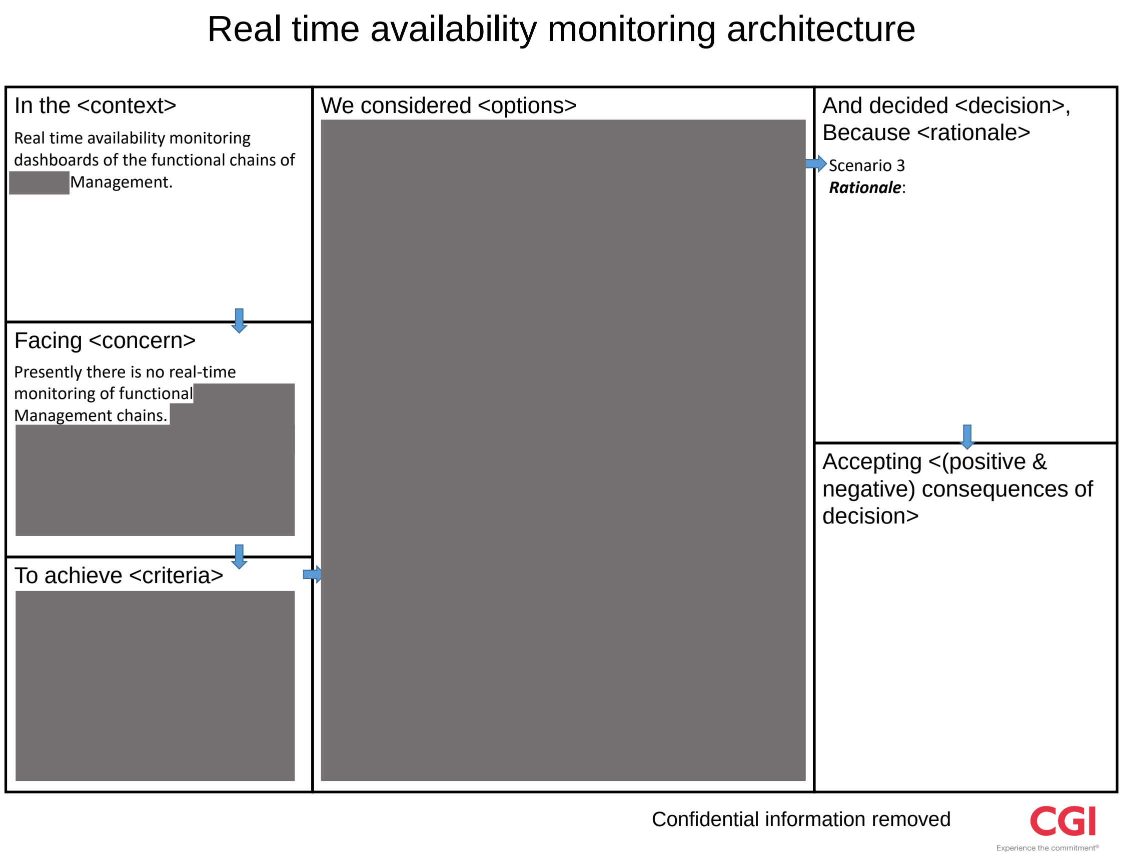
[redacted]

Accepting <(positive & negative) consequences of decision>

[redacted]

Confidential information removed

CGI Experience the commitment®



Real time availability monitoring architecture

In the <context>

Real time availability monitoring dashboards of the functional chains of [redacted] Management.

Facing <concern>

Presently there is no real-time monitoring of functional [redacted] Management chains.

To achieve <criteria>

[redacted]

We considered <options>

[redacted]

And decided <decision>, Because <rationale>

Scenario 3

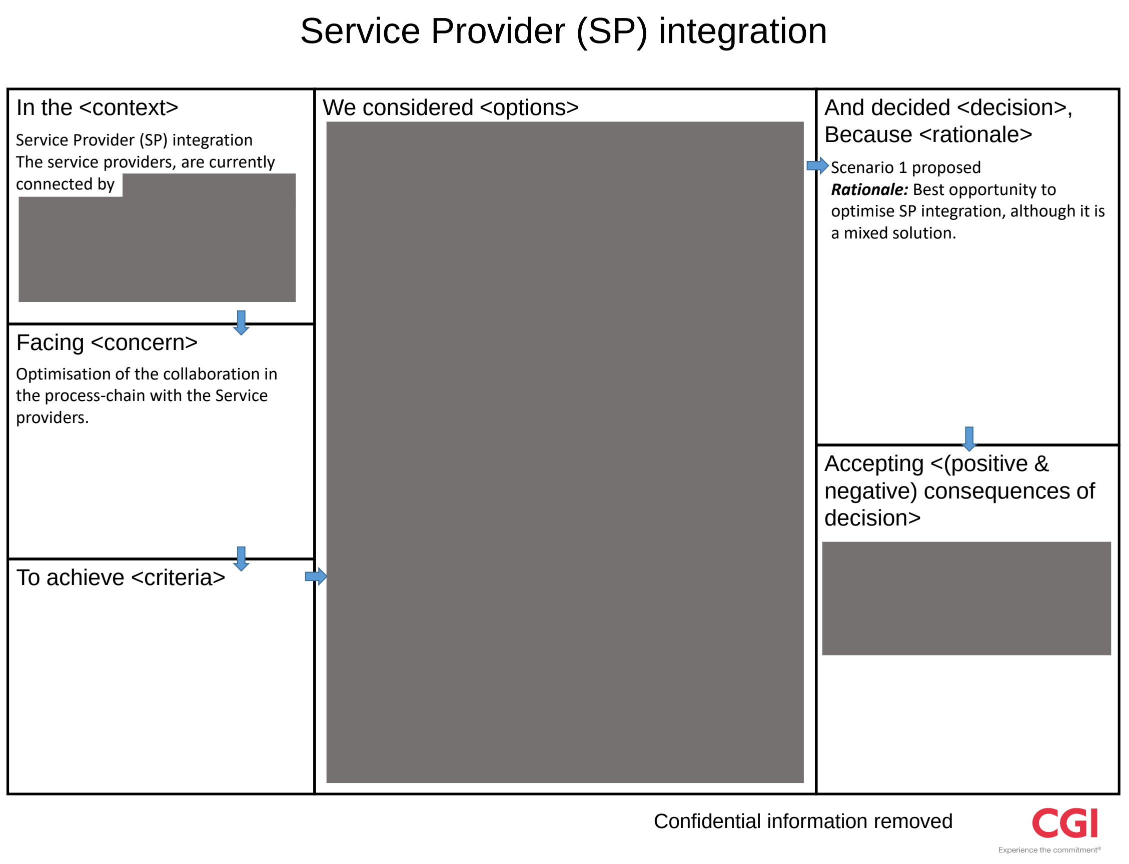
Rationale:

Accepting <(positive & negative) consequences of decision>

Confidential information removed

CGI

Experience the commitment®



Service Provider (SP) integration

In the <context>

Service Provider (SP) integration
The service providers, are currently connected by [redacted]

Facing <concern>

Optimisation of the collaboration in the process-chain with the Service providers.

To achieve <criteria>

We considered <options>

[redacted]

And decided <decision>, Because <rationale>

Scenario 1 proposed
Rationale: Best opportunity to optimise SP integration, although it is a mixed solution.

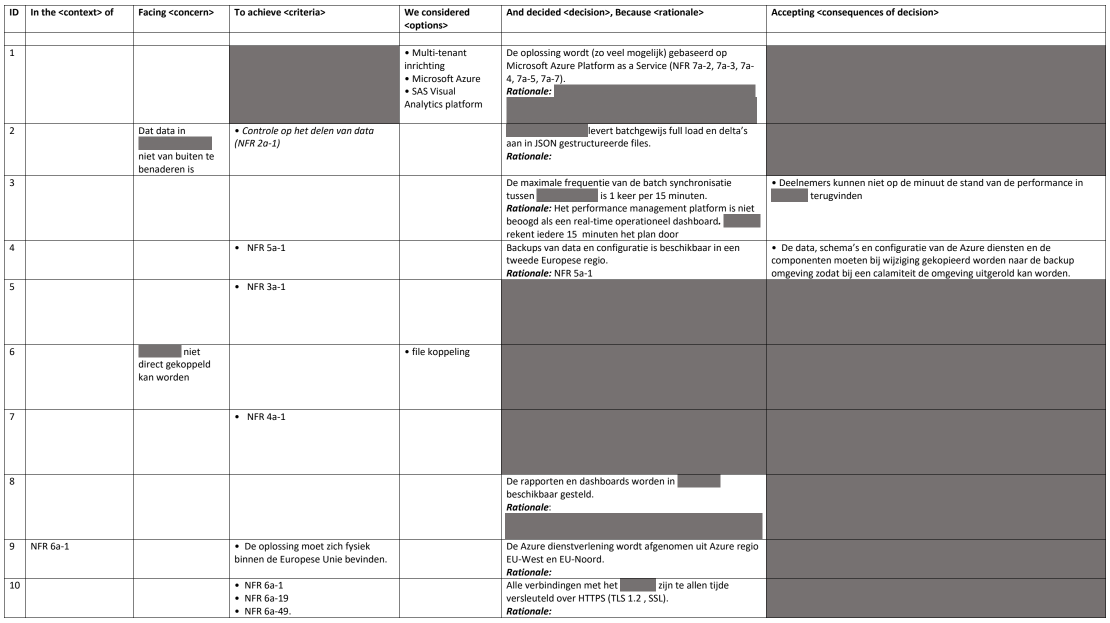
Accepting <(positive & negative) consequences of decision>

[redacted]

Confidential information removed

CGI
Experience the commitment®

C.2 Case Study 2



ID	In the <context> of	Facing <concern>	To achieve <criteria>	We considered <options>	And decided <decision>, Because <rationale>	Accepting <consequences of decision>
1			[redacted]	• Multi-tenant inrichting • Microsoft Azure • SAS Visual Analytics platform	De oplossing wordt (zo veel mogelijk) gebaseerd op Microsoft Azure Platform as a Service (NFR 7a-2, 7a-3, 7a-4, 7a-5, 7a-7). ***Rationale:*** [redacted]	[redacted]
2		Dat data in [redacted] niet van buiten te benaderen is	• *Controle op het delen van data (NFR 2a-1)*		[redacted] levert batchgewijs full load en delta's aan in JSON gestructureerde files. ***Rationale:***	[redacted]
3					De maximale frequentie van de batch synchronisatie tussen [redacted] is 1 keer per 15 minuten. ***Rationale:*** Het performance management platform is niet beoogd als een real-time operationeel dashboard. [redacted] rekent iedere 15 minuten het plan door	• Deelnemers kunnen niet op de minuut de stand van de performance in [redacted] terugvinden
4			• NFR 5a-1		Backups van data en configuratie is beschikbaar in een tweede Europese regio. ***Rationale:*** NFR 5a-1	• De data, schema's en configuratie van de Azure diensten en de componenten moeten bij wijziging gekopieerd worden naar de backup omgeving zodat bij een calamiteit de omgeving uitgerold kan worden.
5			• NFR 3a-1		[redacted]	[redacted]
6		[redacted] niet direct gekoppeld kan worden		• file koppeling	[redacted]	[redacted]
7			• NFR 4a-1		[redacted]	[redacted]
8					De rapporten en dashboards worden in [redacted] beschikbaar gesteld. ***Rationale***: [redacted]	[redacted]
9	NFR 6a-1		• De oplossing moet zich fysiek binnen de Europese Unie bevinden.		De Azure dienstverlening wordt afgenomen uit Azure regio EU-West en EU-Noord. ***Rationale:***	[redacted]
10			• NFR 6a-1 • NFR 6a-19 • NFR 6a-49.		Alle verbindingen met het [redacted] zijn te allen tijde versleuteld over HTTPS (TLS 1.2 , SSL). ***Rationale:***	[redacted]