Master's thesis

Master's Programme in Computer Science

Documenting software architecture design decisions in continuous software development – a multivocal literature review

Säde Harhio

December 6, 2022

Faculty of Science

University of Helsinki

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki,Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Faculty of Science	Master's Programme in Computer Science

Tekijä — Författare — Author

Säde Harhio

Työn nimi — Arbetets titel — Title

Documenting software architecture design decisions in continuous software development – a multivocal literature review

Ohjaajat — Handledare — Supervisors

Dr. M. Raatikainen

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	December 6, 2022	53 pages

Tiivistelmä — Referat — Abstract

The importance of software architecture design decisions has been known for almost 20 years. Knowledge vaporisation is a problem in many projects, especially in the current fast-paced culture, where developers often switch from project to another. Documenting software architecture design decisions helps developers understand the software better and make informed decisions in the future. However, documenting architecture design decisions is highly undervalued. It does not create any revenue in itself, and it is often the disliked and therefore neglected part of the job. This literature review explores what methods, tools and practices are being suggested in the scientific literature, as well as, what practitioners are recommending within the grey literature. What makes these methods good or bad is also investigated. The review covers the past five years and 36 analysed papers. The evidence gathered shows that most of the scientific literature concentrates on developing tools to aid the documentation process. Twelve out of nineteen grey literature papers concentrate on Architecture Decision Records (ADR). ADRs are small template files, which as a collection describe the architecture of the entire system. The ADRs appear to be what practitioners have become used to using over the past decade, as they were first introduced in 2011. What is seen as beneficial in a method or tool is low-cost and low-effort, while producing concise, good quality content. What is seen as a drawback is high-cost, high-effort and producing too much or badly organised content. The suitability of a method or tool depends on the project itself and its requirements.

ACM Computing Classification System (CCS)

Software and its engineering → Software creation and management → Software post-development issues → Documentation

Software and its engineering → Software organization and properties → Software system structures → Software architectures

Avainsanat — Nyckelord — Keywords

documentation, software architecture

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Software study track

Contents

1 Introduction

Software architecture knowledge has been discussed for over 20 years (Perry and Wolf, 1992). Over the years, there have been two different ways of looking at software architecture (Capilla et al., 2016). Initially, software architecture was seen as a very technical task. The end result was most important and it was achieved through components, patterns, and formal architecture description languages. Afterwards, the approach became socio-technical and the focus shifted from the end result to how the end result was reached. Reasoning and decision-making became an important part of architecture. The importance of decisions as a part of architectural knowledge was acknowledged already in 2004 (Bosch, 2004).

Software architecture documentation is important to avoid knowledge vaporization and to enable knowledge transfer to people who have not worked on the project before (Kazman et al., 2016). In modern software development teams do not always have a designated architect and sharing information between team members is especially important (Capilla et al., 2016). Ensuring compliance to requirements as well as traceability is also supported by documentation. While software architecture documentation may have been traditionally overdone, it is sometimes almost completely forgotten when it comes to agile software development practices (Maric et al., 2016). The benefits and costs of software architecture documentation are not always understood, causing it to be highly undervalued (Capilla et al., 2016).

There is already a significant amount of research related to software architecture documentation despite the practice itself being undervalued. Capilla et al. (2016) studied the software architecture knowledge management tools between 2006 and 2016. They presented tools according to generation. The first generation supported representation, capturing and documentation. The second generation added sharing and personalization as well as evolution aspects and assessment features. The third generation improved assessment, advising and reasoning features. Despite all of the tools and their improvements, the practitioners still were not systematically capturing architecture knowledge in 2016.

The main goal of this thesis is to shed light on the current situation of software architecture decision documentation methods, tools and practices in research as well as non-scientific sources. To further analyse the methods their benefits and drawbacks are also gathered.

This literature review gathers tools, methods and practices presented on or after the year 2016. This thesis also analyses grey literature sources to gain an insight into what methods, tools and practices are readily available to most practitioners.

Searching for material for this thesis was challenging due to the keywords being exceptionally common. The scope of the thesis is also quite narrow – there is only one scientific database (Web of Science) used and one non-scientific search engine (Google). All papers containing information about methods, practices or tools for documenting architecture design decisions were analysed if they were also published on or after 2016 by an organization or a person with expertise in software engineering.

Many different approaches to software architecture design decision documentation were found. These approaches were grouped and categorised in the following way: *Tools, Guidelines, Templates, and Processes*. Tools refers to software or algorithms that aid the work by structuring and/or automation. Guidelines are only suggestions for what might be beneficial – for exactly following a certain structure. Templates suggest a certain set of headlines and what information to fill in to create a collection of individual documents. Processes suggest a certain set of actions that are necessary to successfully perform the action of knowledge capturing. Tools are further divided into Creation, Extraction, Gamification, and Search. Creation refers to tools that help practitioners input information into a system. Extraction means tools that take a set of existing information as an input and give documentation as an output. Gamification covers software which aims to reward the practitioner for performing documentation activities. Search tools minimize the need for good documentation organization. Analysis was done for categories in general and individual methods, practices, and tools where they differed from the majority of the group.

Most analysed scientific papers focus on tools, especially *Extraction tools*. The grey literature heavily concentrates on *ADR templates*, which are not mentioned in the scientific literature. The Extraction tools aim to solve the problem of cost and motivation to create documentation. The ADRs seem to be what practitioners are used to using for documentation. While the ADRs are simple and aim to reduce the amount of work, they still require someone to make and maintain them, and there are certain drawbacks to using them incorrectly. The benefits of the methods and tools are mainly using minimal effort to produce the right amount of good quality content, while the drawbacks do the opposite. The benefits and drawbacks are often tradeoffs or depend on the level of experience of the professional writing the documentation.

The thesis is organized in the following way. Chapter 2 discusses the research method including the research questions. Chapter 3 presents the results for each research question. Chapter 4 discusses the results as well as the limitations and validity of the review. Chapter 5 concludes the findings.

2 Research Method

The literature review method was chosen for this study because the aim was to evaluate the methods and practices of modern architecture design decision documentation in a credible and repeatable way. Multivocality was chosen because the latest practices and methods published in "grey" literature offer valuable insight into this topic to both practitioners and researchers and may still be missing from formal studies. The guidelines suggested by Garousi et al. (2019) were followed.

First, the research questions were defined (section 2.1). Based on the questions, preliminary search terms were decided and a search process was designed. The inclusion and exclusion criteria were established in the search phase. Then, paper quality assessment strategy was decided. Based on the research questions, it was decided which data would be extracted from the papers, so that the questions would be answered as methodically as possible. The data extraction form also includes quality assessment data. Finally, data analysis strategy was decided.

The validity of the protocol was checked by applying the process to a small subset of the papers and testing the data extraction form on a single paper. Issues discovered when applying the process were identified and fixed. When testing the data extraction form, special attention was paid to being able to collect data based on it, and that the resulting data helps answer the research questions.

2.1 Research questions

This work answers the following research questions:

RQ1. How are architecture design decisions documented in the literature?

RQ2. What are the benefits of the suggested documentation methods or practices?

- RQ2.1. What are the benefits to writers?
- RQ2.2. What are the benefits to readers?
- RQ2.3. What are the benefits to those who maintain the documentation?

RQ3. What are the drawbacks of the suggested documentation methods or practices?

RQ3.1. What are the drawbacks to writers?

RQ3.2. What are the drawbacks to readers?

RQ3.3. What are the drawbacks to those who maintain the documentation?

The goal of RQ1 is to bring together information on current methods and practices. Knowing what methods are available and deemed effective aids the creation of good quality documentation suited for modern fast-paced development. High-quality documentation can significantly aid the work of designers and developers in the long run and prolong the lifespan of a system. The knowledge of current method suggestions, especially those unique to the grey literature, also increases the overall understanding of the researchers. This may help highlight gaps in research.

RQ2 and RQ3 focus on finding the benefits and drawbacks of the identified methods. They are further divided into categories according to whom the benefit or drawback applies to. The goal of this is to make trade-offs between different methods more obvious. This aids the selection of a suitable method and gives direction for the development of these methods.

2.2 Search strategy

The main search method uses the search engines of electronic data sources. It is not simple to optimise the search terms, so the optimisation suggestions made by Huynh Khanh Vi et al. (2022) were followed. Because search engines for finding grey literature often find numerous results with varying quality, and the relevance of the results declines the further down they are in the list, saturation was used as a stopping criterion for the search. This means that the search was stopped when no new and relevant results were found anymore.

2.3 Scope and sources

The scope of this review is limited to the documentation of design decisions in the fast-paced web software development. The publishing year of the results is limited to a minimum of 2016 as a previous study researching methods and tools has been conducted before

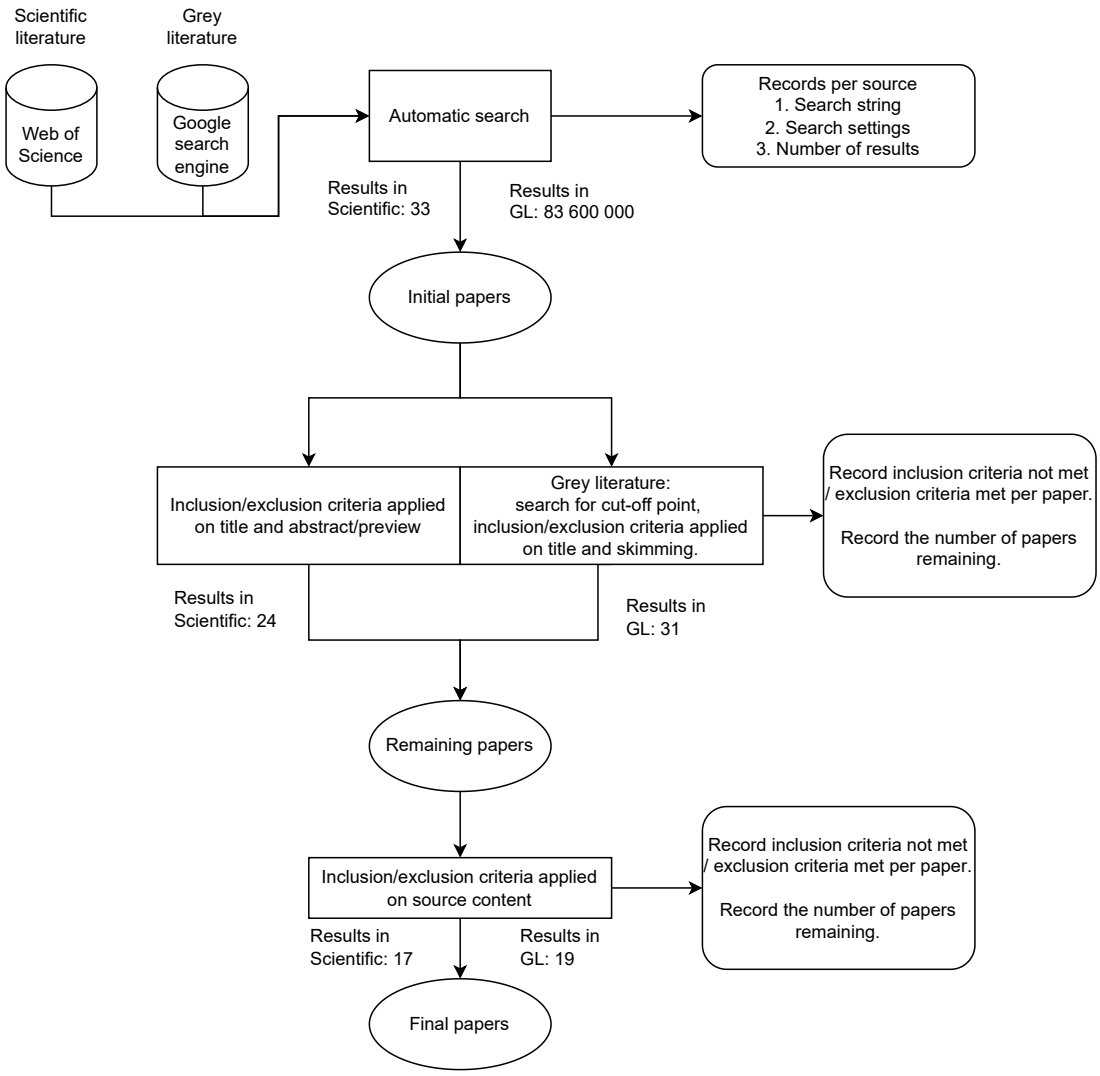

Figure 2.1: Search process resulting in 36 papers.

2016 (Capilla et al., 2016). The Web of Science database was used for finding scientific papers and Google was used for finding grey literature.

2.4 Search process

First, the automatic search was performed on the selected database and search engine listed in Section 2.3. Duplicate results were ignored. Next, the titles were read and, if the title was not off-topic, the abstract was also read. For grey literature, the content was skimmed. Some papers were excluded at this point. The reasons for the exclusions were recorded. If there was doubt about whether a paper should be included or not, it was

included at this point. Because of the number of grey literature results, only the promising results were recorded at first. The results included many broken links, outdated pages, advertisements, and results that were not related to the topic or only vaguely related, such as Wikipedia and other general information pages. The highly irrelevant results were not recorded.

The full texts for papers that fulfilled the inclusion criteria were retrieved and added to the Atlas.ti data-analysis tool for preliminary analysis. The inclusion and exclusion criteria were applied to the content of the papers. The reasons for excluding papers were documented. For a summary of the search process, see Figure 2.1.

2.5 Search terms for formal search

The research questions were used to identify the major keywords for search terms. The initial keywords identified were “documentation”, “architecture” and “design decisions”. “Records”, “recording” and “capturing” (knowledge) were identified as synonyms for “documentation” and “Design rationale” and “design reasoning” for “design decisions”.

“Software” was also recognised as a potentially important defining keyword, but it can unnecessarily exclude relevant papers when combined with the ‘AND’ operator (Huynh Khanh Vi et al., 2022). This was also observed in the preliminary searches here, so “Software” as a keyword was excluded. However, leaving the keyword out caused the need to filter out completely unrelated results in some other way. Unrelated results were limited by filtering certain other keywords out (see Table 2.1) and defining the relevant research areas. Although, defining research areas can limit the results unnecessarily as sometimes research areas are not correctly filled in (Huynh Khanh Vi et al., 2022). Here, a large number of irrelevant results were filtered out and no significant loss of relevant results was noted.

The Web of Science search engine has a convenient field tag called “Topic”, which covers title, abstract, author keywords and Keywords Plus. Unfortunately, using the field also caused an influx of irrelevant results. It appears that the field often contains keywords that only distantly relate to the topic of the paper. Thus, the main terms were searched in titles and abstracts only. From the smaller pool resulting from the more limited search, it was easier to find a set of the most relevant results, which were then observed to find more relevant keywords (see Table 2.1, section “Keywords”). The abstracts of the papers were also skimmed to find synonyms and alternative ways of communicating the same idea.

Table 2.1: Search string structure for the scientific search. Each section was connected to the others with an "AND" operator to form a single query.

Purpose	**Keywords**	**Target field(s)**
Topic	document* AND architecture	Topic
Synonyms for "documenting"	captur*	Title or Abstract
	OR document*	
	OR record*	
Synonyms for method/tool/process	method*	Title or Abstract
	OR process*	
	OR tool*	
	OR guideline*	
	OR "documentation approach"	
	OR recommend*	
	OR improv*	
Synonyms for "architecture decision"	"architecture document*"	Title or Abstract
	OR "architecture knowledge"	
	OR "architectural knowledge"	
	OR "design decision"	
	OR "design rationale"	
	OR "design reasoning"	
Keywords	"documentation"	Author Keywords
	OR "System document*"	
	OR "architectural design decision*"	
	OR "architecture document*"	
	OR "architecture documentation model"	
	OR "AKM"	
	OR "Knowledge acquisition"	
	OR "requirements communication"	
	OR "Architectur* knowledge"	
	OR "Software architecture*"	
	OR "Software evolution"	
	OR "architecture erosion"	
Research area	"Computer Science"	Research Area
	OR Engineering	
	OR "Information Science & Library Science"	
Filtering irrelevant results	NOT "Virtual reality"	Author Keywords
	NOT "VR"	
	NOT "robotics"	
	NOT "security"	
Filter for document types	NOT "Review Articles"	Document Types

Without using these synonyms and alternatives, many relevant papers did not appear in the result set because they never actually mentioned the word "documentation", for example. The opposite was also done to the most irrelevant papers by checking their keywords. This was fairly simple as the sorting by result relevancy works well enough. The keywords found were "Virtual reality", "VR", "robotics" and "security".

The set of relevant papers was also used as a standard to compare the new result sets. It was assured that at least the most relevant papers would still be present. This was done to ensure that the quality of the search results would not deteriorate due to ill-informed search terms or filtering choices.

It was quite challenging to obtain papers focusing on documentation without also getting papers that discuss decision making. Choosing to use multi-word keywords appears to make the results better, but it also makes it more likely that the term will cause some papers to be excluded because they do not use any such combination. Review papers were filtered out of the results, but papers are not always correctly labelled, so at least one review still made it to the result set.

Finally, the search term has the following structure:

topic AND *synonyms for "documenting"* AND *synonyms for "method"* AND *synonyms for "architecture decisions"* AND *"keywords"* AND *"research areas"* AND *"filters"*

See Table 2.1 for the entire list of terms and fields used in the search term. Many terms also have an asterisk to ensure that using a different form of a word will not cause a paper to be excluded from the results.

2.6 Search terms for grey literature search

First, it is very challenging to evaluate the quality of the search results gained from Google.com. Google does not always appear to give the same results in the same order when using the same phrase. Several preliminary searches were conducted only to conclude that Google appears to do the best when it is given "intuitive" search terms. "Intuitive" referring to a simplistic term written without much consideration, but covering the topic enough. Complicated queries with "AND" and "OR" operators only appeared to obtain either the same results as an "intuitive" query or actually worse results. The search term that was used was *"documenting architecture design decisions"*.

As simple as the query is, it does offer an interesting insight into a search that could be conducted by someone who was simply looking to discover more about the topic. This is of interest, as architecture design decision documentation appears to suffer from not enough of it being done. Also, Google is often just the thing that people turn to when seeking practical information.

2.7 Inclusion and exclusion criteria

The inclusion and exclusion criteria were based on the research questions and scope as well as the quality criteria for grey literature suggested by Garousi et al. (2019). Each paper was evaluated against these criteria. First, only the title and abstract were evaluated and, if the paper was not excluded, also the content of the paper. To be included, the papers needed to meet *all* of the inclusion criteria. Simultaneously, the papers were excluded if they did not meet any inclusion criteria or they met *any* exclusion criteria.

The inclusion criteria the source must meet:

I1. Paper discussing, describing, suggesting or evaluating documentation methods, practices, or tools for software architecture design decisions.

I2. The discussed methods, practices, or tools are meant for software development context.

I3. The author(s) has expertise in software engineering or the publishing organisation is reputable in the field of software engineering.

I4. Published on or after 2016.

For I3 the operator “or” is used instead of “and” to not unnecessarily exclude results where one makes up for the other. Some sources might not disclose the author because the source strictly represents an organization instead of an individual. Also, some professionals write blogs, which are generally not considered reputable, but the reputation of the author can make up for this to some extent.

The exclusion criteria the paper must not meet:

E1. Architecture design decision documentation is discussed in general, but no methods, practices, or tools are mentioned.

E2. Architecture design decision documentation methods, practices, or tools are mentioned, but not described in detail and no benefits or drawbacks are presented.

E3. The date of publishing is unclear.

E4. Not available in English.

E5. Editorial, introduction, or similar paper that only summarises or refers to a separate original work.

The purpose of E4 is to directly exclude any papers that partially duplicate an original work. In these cases, the original work is chosen instead.

2.8 Quality assessment

As the main interest is in practical methods of documentation, it is important to know how relevant the papers actually are. The more realistic the setting of the study is, the more relevant the paper is (Ivarsson and Gorschek, 2010). The practical, six level classification approach presented by Alves et al. (2010) was used for the evaluation of each paper:

1. No evidence.
2. Evidence obtained from demonstration or working out toy examples.
3. Evidence obtained from expert opinions or observations.
4. Evidence obtained from academic studies, e.g., controlled lab experiments.
5. Evidence obtained from industrial studies, e.g., causal case studies.
6. Evidence obtained from industrial practice.

The quality assessment of the grey literature was performed according to a simplified version of the quality assessment checklist proposed by Garousi et al. (2019). See Table 2.2 for the quality assessment criteria of the grey literature. Unlike in the original quality assessment checklist, the criterion for dates was excluded because it is already included in the selection criteria. Even though the criteria for author expertise and publisher reputation are also included in the selection criteria, they were left in here because the selection criteria demand one or the other and not both. Novelty was also excluded because exclusion criteria E5 already leaves out sources that only repeat another piece of work. Methodology

Table 2.2: Quality assessment checklist of grey literature based on the original table by Garousi, Felderer and Mäntylä (Garousi et al., 2019).

Criteria	Questions
Q1 Authority of the publisher	- Is the publishing organization reputable? Score 1: Independent websites/ reputable organizations (for example, government wesites / non-profit communities / university websites) Score 0.5: Company websites Score 0: Personal websites / open platforms
Q2 Authority of the author(s)	- Does the author have expertise in the area? Score 1: Author has more than 20 years of experience. Score 0.5: Author has 10 years or more, but less than 20 years of experience. Score 0: Author has less than 10 years of experience or experience years are unknown.
Q3 Methodology	- Is the information supported by references?
Q4 Objectivity	- Is there vested interest?
Q5 Impact	- Impact metrics based on the backlinking scores of the individual webpage and the domain it is under.
Q6 Outlet type	Score 1: High outlet control/ high credibility (for example: books, magazines, theses, government reports, white papers) Score 0.5: Moderate outlet control/ moderate credibility (for example: annual reports, news articles, presentations, videos, Q/A sites [such as StackOverflow], Wiki articles) Score 0: Low outlet control/ low credibility (for example: blogs)

was narrowed down to references as the sources are generally not methodological, their quality being on the lower end. Objectivity was narrowed down to only cover vested interest. However, higher value was given to those sources that did use references. The impact metrics were simplified to only contain backlinking information collected from the Free Backlinks Checker tool by SEO Review Tools (https://www.seoreviewtools.com/valuable-backlinks-checker/). The *page authority* value together with the *domain authority* value were normalised to 0-1 range.

2.9 Data collection and analysis

Data extraction was performed on each paper using a data extraction form (see Table 2.3). Items F1-F4 record the basic information of the papers, including the source for reliability. Items F5-F11 were extracted from the papers to answer the research questions. Results for F5 were also given a more detailed label to categorise the methods, practises and tools

Table 2.3: Data extraction form.

#	Field	Concern/research question
F1	Author(s)	Documentation, GL Quality assessment
F2	Year	Documentation
F3	Title	Documentation
F4	Source	Reliability of review
F5	Method/practice/tool (label, brief description)	RQ1
F6	Benefits of the method or practise from writers point of view	RQ2
F7	Benefits of the method or practise from readers point of view	RQ2
F8	Benefits of the method or practise from maintenance point of view	RQ2
F9	Drawbacks of the method or practise from writers point of view	RQ3
F10	Drawbacks of the method or practise from readers point of view	RQ3
F11	Drawbacks of the method or practise from maintenance point of view	RQ3
F12	Context of evidence (6 level classification)	Quality assessment
F13	Publishing organization	Q1
F14	Author(s)	Q2
F15	Methodology	Q3
F16	Objectivity	Q4
F17	Impact	Q5
F18	Outlet type	Q6

appropriately. Item F12 records the context of the evidence in the paper for the quality assessment of the scientific literature. Items F13-F18 record the quality attributes of the grey literature.

The Atlas.ti data analysis tool was used to record the results. Initial codes for the results were generated based on the data extraction form. More refined codes were added during and after the initial data extraction to categorise the findings further.

3 Results

Data was systematically extracted from selected papers to answer the research questions. See Figure 2.1 for the steps to determine the included papers. In this chapter, first there is an overview of the included papers. Then, the results are presented for each research question.

3.1 Overview of included studies

A total of 36 papers were included in the analysis. The search in the scientific database found 17 of the selected papers. The grey literature search found 19 papers. Four papers found with the grey literature search are conference papers, which potentially could have also been found with the scientific database search (see the validity section 4.4 for more information). The included papers were divided into categories depending on their approach to documentation and organised by publication year to analyse changes in the methods, processes, and tools over the years. The papers found in the scientific database are referred to using the style Sn, while the style for the papers found by the Google search engine is Gn.

The scientific papers have much more variety in the types of methods, processes and tools as opposed to the grey literature. The majority of the grey literature bases their approach on the idea of ADRs (Architecture Decision Records) first introduced in 2011 by Michael Nygard (2011). The analysed grey literature papers discussing ADRs offer guidelines, templates and tools for using ADRs. Two grey literature papers give more generic guidelines for documentation [G24] [G6].

Table 3.1: Scientific literature quality assessment of analysed papers.

Evidence levels	Number of papers	Paper identifiers
1. No evidence	1	[S27]
2. Demonstration or toy examples	5	[S16], [S30], [S31], [G12], [G14]
3. Expert opinions or observations	4	[S15], [S20], [S24], [G5]
4. Academic studies	7	[S3], [S5], [S8], [S21], [S2], [S25], [G28]
5. Industrial studies	3	[S17], [S12], [S26]
6. Industrial practice	1	[S6]

The quality of the scientific papers was assessed and each paper was put into a category based on the evidence level (see Table 3.1 for the quality assessment of the scientific papers). The papers are almost equally divided into the lower and upper half of the evidence level categories. One paper has no evidence [S27] and one paper has industrial practise level evidence [S6]. The biggest evidence level group is "academic studies" with seven papers.

All the analysed grey literature source types and the papers that belong to each source type group are shown in Table 3.2. Overall, the types of sources were mostly blogs (8 of 15 grey literature papers). Three sources [G2][G1][G10] were documentation pages in addition to a book [G6], an experience report [G18], a presentation [G17] and instructional archives [G13].

Table 3.2: Grey literature source types.

Grey literature type	Number of papers	Paper identifiers
Blog	8	[G7][G24][G15][G19][G23][G29][G32][G4]
Documentation	3	[G2][G1][G10]
Book	1	[G6]
Experience report	1	[G18]
Presentation	1	[G17]
Instructional archives	1	[G13]

Each grey literature paper received a quality score between 0 and 6 (see Table 3.3 for the scores). The highest score is 5.7 and the lowest score is 1.7. Most of the papers received a score greater then 3 and five papers received a score greater than 5. The low scoring papers (score less than 3) are private blogs [G15][G4][G23]. The high scoring papers are documentation papers [G2][G1], a presentation [G17], an experience report [G17] and a book [G6].

3.2 RQ1. How are architecture design decisions documented in the literature?

To analyse the results, the documentation approaches and tools were divided into categories. In this section, the categories will be explained first. After, the results for each category are presented.

The analysis resulted in identifying and differentiating four different categories of ap-

Table 3.3: Grey literature quality assessment of analysed papers.

Paper	Quality score						
	Q1	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Total**
[G7]	1	0.5	1	1	0.6	0	4.1
[G2]	1	1	1	1	0.7	0.5	5.2
[G1]	1	1	1	1	0.2	0.5	4.7
[G24]	0	1	1	1	0.2	0	3.2
[G10]	1	1	0	1	0.1	0.5	3.6
[G13]	0.5	0	0	1	0.7	0.5	2.7
[G15]	0.5	0	0	1	0.2	0	1.7
[G4]	0.5	0.5	0	1	0.4	0	2.4
[G17]	1	1	1	1	0.7	0.5	5.2
[G18]	1	1	1	1	0.4	1	5.4
[G19]	0	0.5	1	1	0.3	0	2.8
[G23]	0	0.5	1	1	0.2	0	2.7
[G29]	0	1	1	1	0.4	0	3.4
[G32]	0	1	1	1	0.3	0	3.3
[G6]	1	1	1	1	0.3	1	5.3

proaches in the included papers. The categories emerged during analysis by grouping the papers. These categories are *Tools, Guidelines, Templates and Processes*, as summarised in Table 3.4. Some papers appear in more than one category as they may, for example, argue for guidelines and also present a tool (for instance [S2]). *Tools* covers software or algorithms, which help with creating, maintaining, or using documentation. Tools for using documentation were included because this implies that documentation should be made in certain ways or that the way documentation is made is not as important because the tool will compensate for the drawbacks of the documentation itself. Papers discussing *Templates* present a certain structure and content, which documentation should follow for certain reasons. *Guidelines* refers to general information, which aims to help with creating, maintaining, or improving documentation. Guidelines can be general or refer to a specific method of creating documentation. Guidelines are simple, stand-alone pieces of information instead of a set of rules. An example of a guideline is that "decision documentation should be made at the same time as design decisions are made". The details of how this guideline is put to use is up to the practitioner. *Processes* covers a set of actions required to successfully perform documentation tasks. These are techniques or frameworks, which determine *how* documentation should be recorded. For example, a process demands that one follows a specific set of design steps and at each step records certain pieces of information.

Table 3.4: The distribution of analysed papers into categories.

Category	Number of papers	Paper identifiers
Tools	15	[S2][S27][G14][G5][S8][S30][S3][S5][S6][S31][G28] [S25][S16][G12][G23]
Creation	4	[S2][G14][G5][S8]
Extraction	8	[S3][S5][S6][S31][G28][S16][G12][S27]
Gamification	1	[S25]
Search	1	[S30]
Processes	2	[S21][S26]
Guidelines	6	[S17][S15][S2][S20][G24][G6]
Templates	15	[S12][S24][G10][G17][G18][G29][G32][G13][G15][G2] [G1][G7][G4][G19][G23]
ADR Templates	12	[G17][G18][G29][G32][G13][G15][G2][G1][G7][G4] [G19][G23]
Other Templates	3	[S12][S24][G10]

Tools were further divided into *Creation, Extraction, Gamification, and Search*. A summary of this categorisation is in Table 3.4 as subcategories of *Tools*. *Creation* tools cover anything that the architect interacts with to write documentation or which helps with deciding the content or structure of documentation. *Extraction* refers to a tool, which takes in as input a set of existing materials, such as general documentation or chat logs, and outputs the architecture decision details found in the material. *Gamification* covers tools, which add a game element to the process of documentation to make it more attractive to architects and encourage beneficial practices. *Search* tools take a backwards approach to ensuring the usefulness of documentation by making it easy to find information even if the documentation itself is not very clear.

The scientific and grey literature sources differed in their approaches significantly (see Table 3.5 for the categories of the scientific papers and Table 3.6 for the categories of the grey literature papers). The scientific papers mostly focused on presenting different kinds of tools as well as some processes, guidelines and templates. The grey literature was much more homogeneous in the approaches than the scientific literature. Twelve of fifteen grey literature papers discussed ADRs, which could technically be categorised as a template approach. However, there was another template type approach, so for clarity the papers discussing ADRs were placed into a separate category. One paper was discussing ADRs generally, but also introduced a tool, so it appears in two different categories: “ADR” and “tools”. The categories used for grey literature are “ADR”, “templates”, “tools” and “guidelines”.

Table 3.5: The distribution of analysed scientific papers into categories.

Category	Number of papers	Paper identifiers
Tools	14	[S2][S27][G14][G5][S8][S30][S3][S5][S6][S31][G28][S25] [S16][G12]
Creation	4	[S2][G14][G5][S8]
Extraction	8	[S3][S5][S6][S31][G28][S16][G12][S27]
Gamification	1	[S25]
Search	1	[S30]
Processes	2	[S21][S26]
Guidelines	4	[S17][S15][S2][S20]
Templates	2	[S12][S24]

Table 3.6: The distribution of analysed grey literature papers into categories.

Category	Number of papers	Paper identifiers
Tools	1	[G23]
Guidelines	2	[G24][G6]
Templates	13	[G17][G18][G29][G32][G13][G15][G2][G1][G7][G4][G19] [G23][G10]
ADR Templates	12	[G17][G18][G29][G32][G13][G15][G2][G1][G7][G4][G19] [G23]
Other Templates	1	[G10]

Tools were discussed in fourteen (14) scientific papers and one (1) grey literature paper. Because there are different goals for tools, they were further divided into finer categories. Each category was analysed separately and the results are presented in the following sections.

Creation tools [S2][G14][G5][S8][G23]. A summary of creation tools is in Table 3.7. ProSAD [S2] is a documentation strategy recommender. The creators of the tool find it especially important that it is considered what information the project stakeholders wish to gain from the documentation. This means that the preferences of the stakeholders are always reflected in the contents of the documentation. For example, a stakeholder could be interested in the performance of the software; so in this case, it would be important to record any performance-related decisions. They also strongly believe in an incremental approach to the documentation process. DecDoc [G14] is designed to help decision-making by supporting the relations between decision documentation and requirements, design diagrams and code. It specifically supports working collaboratively and incrementally. Another tool attempts to leverage voice commands to make the process of documentation easier [G5]. Software Architecture Knowledge Base (AKB) [S8] is a web application for

Table 3.7: Summary of the *Creation* tools organised by year.

Name	Short description	Source	Year
ProSAD	Documentation strategy recommender. Supports stakeholder preferences.	[S2]	2016
DecDoc	Captures decision-related information and supports collaboration and incremental work and highlights relationships between documentation and other materials.	[G14]	2016
Software Architecture Knowledge Base (AKB)	Web application for documenting and sharing. Based on the Architecture Profile (contains architecture knowledge items, relationships modeling).	[S8]	2019
-	Captures design decisions during design conversations. Links the captured decisions to UML artifacts.	[G5]	2020
LoqBook	Combines creation, extraction and search functions.	[G23]	2021

documenting and sharing. The approach AKB takes regarding the content and organisation of documentation is called an Architecture Profile, which contains design rationale among other architecture knowledge items. The profiles are also hierarchically structured and they explicitly model relationships between the architecture knowledge elements they contain. Loqbook [G23] reported in the grey literature combines creation, extraction and searching. The extraction feature uses Slack chat logs to find decisions and record them.

Extraction tools [S3][S5][S6][S31][G28][S16][G12][S27]. The summary of the *Extraction* tools is in Table 3.8. Notably, three [S3][S5][S6] of the seven papers were by the same group of authors expanding on the same line of work and two [S16][G12] other papers were by another author. These tools extract information from other types of existing documentation and automatically turn it into architectural knowledge. *Extraction* tools include ADeX (Amelie - Decision Explore) – an automatic natural language extraction tool that uses, for example, chat logs, meeting notes, code commit messages and issues in Issue Management Systems (IMS) as the source and produces design decisions [S3][S5][S6]. It also provides recommendations to support the decision-making process. The tool is automatically kept up to date when new information is added to the systems it uses. Machine learning is used for data classification so that design decisions can be recognised. Gilson, Annand and Steel also take a natural language processing approach to capturing design decisions directly from written design conversations [G12]. Their approach disposes of the need for intensive training that is necessary with machine learning approaches. REM is only a partially automated method, which is based on systematic code inspection [S31]. This requires some manual labour in the form of systematic code inspection using the Grounded

Table 3.8: Summary of the *Extraction* tools.

Name	Input	Output	Source	Year
–	Data from Enterprise Architect	Task specific documentation (including design decisions)	[S27]	2016
REM	Results of systematic code inspection	Recurrent clusters and interfaces (can be used to identify design decisions)	[S31]	2018
RecovAr	History artifacts such as the issue tracker and version control repository	Design decisions	[G28]	2018
ADeX (Amelie - Decision Explore)	Chat logs, meeting notes, code commit messages and issues in Issue Management Systems (IMS)	Design decisions	[S3][S5][S6]	2017/2019
–	Written design conversations	Design decisions	[S16][G12]	2019/2020
LoqBook	Slack chat logs	Design decisions	[G23]	2021

Theory research method. The automated part of it uses model-transformation technology in the last stage of the process to find recurrent clusters and interfaces. RecovAr [G28] uses history artifacts such as the issue tracker and the code repository to create documentation. An older proposal would create custom, task specific documentation [S27]. This tool is only on the concept level, but it would use information from Enterprise Architect to create documentation for a developer performing a specific development task.

Gamification tools [S25]. In this paper, gamification was studied in relation to creating architecture design documentation. This study did not find any particular benefits from the gamification, although this could be attributed to the length and design of the experiment itself.

Search tools [S30]. The purpose of this tool is to find relevant information within software architecture documentation by locating "usage-based chunks" that correlate with certain information needs. A successful tool of this type would lessen the importance of any specific documentation style as information would be easily retrievable regardless.

Processes [S21][S26]. These papers place more emphasis on how the process of documentation happens rather than the content of the documents. One of these methods is called the Rationale Capture Cycle [S21]. This aims to be non-invasive and low effort as the doc-

umentation becomes part of the design process. Another paper has a similar approach, where architecture knowledge recovery is made a part of the design process [S26].

Guidelines for documentation [S15][S17][S20]. One paper discusses which organisation style is better for architecture decision documentation – ontology or file-based – when it comes to finding information [S17]. Ontology-based documentation uses concepts and categories to organise information non-linearly according to relationships. File-based documentation relies on one or more text files to store information. The results of the study state that the needs of the project determine which style is better. File-based documentation has many issues with finding the needed information due to the complexity of documentation, problems with the structure of the documentation and cross-referencing being hard to maintain. The ontology-based approach solves many issues that the file-based approach has, but may not be cost-effective enough and quality issues may significantly reduce its usefulness. Another paper suggests that for documentation to be as concise as possible only information important to the specific project should be recorded [S15]. They also present a method for finding out which pieces of information are important to individual projects. Generally, the purpose of documentation should be to preserve architectural knowledge, facilitate communication by improving comprehension, guide the implementation and support the assessment of architecture. The process includes determining the non-functional requirements, prioritising them and developing the solutions for these requirements. These types of decisions are important to document formally. Decisions with less risk can have less formal documentation. Structures and relationships between architecture knowledge elements are important [S20]. Knowledge elements can be, for example, *decision, decision components, persons (decision makers), problem, solution, context, rationale, issue, goal, claim, alternatives, assumptions, implications, constraints, arguments* and *assessments*. Recognising these elements and recording them individually allows making small, incremental changes, which increases the efficiency of documentation. Linking these elements to requirements or design artifacts helps increase understanding.

Guidelines found in grey literature suggest that documentation is meant to support communication and aid understanding and decision-making in the future [G24][G6]. It is important to remember that documentation is supposed to create business value [G24] and is not made just for the sake of documenting. Decision documentation should be made when decisions are made as otherwise the rationale might be forgotten [G24][G6]. The main information to record is rationale or justification, as not all decisions are optimal [G6]. Minimum fields would be driver (such as a quality attribute), decision and

location (in the codebase) as well as rationale. On top of these, it might make sense to also record assumptions, evidence, who is involved, and why shortcuts or trade-offs were decided on. Critical systems need more documentation than less critical systems. Decisions that ensure that quality attributes are satisfied should be carefully documented, other decisions are not as important. Important decisions keep the implementation on the right track and make maintenance easier in the future, ensuring that the architecture does not deteriorate and quality attributes are not forgotten.

Templates [S12][S24][G10]. The papers emphasise the content and organisation of the documentation. One of these presents High-level Design Stories, which records design decisions and their context, architecture assumptions, quality-attribute analysis and system issues [S12]. These artifacts are modular and the global architecture is a combination of all of these artifacts. Another paper describes an artifact called "The Sofware Architecture Document", which is meant to contain a minimal amount of information informally presented [S24]. This document contains eight sections, but we are mostly interested in the following three of them: "Architectural Decisions on the Data Model", "Architectural Decisions on the Deployment Model", and "Architectural Decisions on the Implementation Model". Only the most relevant architectural decisions should be recorded. Minimality and informality are meant to better accommodate agile software development practices. Informality especially encourages the practice of documentation. This document contains the global architecture and it is updated when the architecture is verified and reviewed. The decision part of this artifact is specifically described as "without many details". The artifact follows the guideline that documentation should be simple, short, understandable, revised, and used for communication. The Unified Architecture Method (UAM) framework documentation, found in the grey literature, also mentions a template for documenting details of an architecture decision [G10]. According to it, it is important to record priority, domain, state and type together with author, revision and the date. Only decisions affecting the fundamental structure should be recorded. Documentation storage should be searchable and able to provide summary reports. Decision documents that are replaced with new decisions should be deleted.

ADR templates was what grey literature mostly focused on. ADR is a documentation format popularised by a blog post by Nygard (2011) consisting of context, decision, status and consequences. The contents of the ADR related papers ranged from actual documentation [G2][G1] to custom templates or tips and best practices for using ADRs.

The style in which ADRs are written is meant to be concise and to the point. It is recom-

mended that the language of ADRs is kept as concise as possible, while still using complete, understandable sentences [G32][G17]. The titles of ADRs should also be descriptive and written in "active voice", by using words like "use" (an example with placeholders: "use-<technology>-for-<purpose>.md") [G7]. The rationale of decisions should be based on evidence and requirements should be referred to where they are relevant [G32]. Evidence can be older ADRs.

The language used to make ADRs is generally agreed on. The original blog post recommends using markdown (or another lightweight language) and most papers echo this [G18][G17][G29][G13][G32][G23], although one paper [G15] also recommends a plain text markup language called AsciiDoc (*AsciiDoc* 2022). It is also said that it does not really matter what file type is used as long as the ADRs are written somehow [G32].

How much information is documented is difficult to determine. It is important to ensure that not everything ends up documented [G17][G32]. Experiences reported by Keeling and Runde also support this [G18]. People tend to overcompensate and document too much if they are not sure what counts as an "architectural decision". The decisions recorded should somehow influence quality attributes or have necessary adverse effects on the program, which need to be managed. The content which is documented should be limited. The decisions that should be documented are hard or costly to change [G32]. They could also be important or risky or affect a large amount of code [G1]. They could modify externally visible features, change public interfaces, affect dependencies, be caused by constraints, change general structures or cause changes in current development approaches [G17].

Documentation maintenance can be performed in two ways – documents are deprecated, or changes are made to them overtime. Two papers emphasise the point made in the original blog post that ADRs should not be changed after they are made [G29][G7]. This is because they are meant to represent a point in time. Changes are marked by making a new ADR.

The length of ADR documents is important. The original blog article states that ADR files are "short" – the entire document is one to two pages long (Nygard, 2011). Keeling and Keller agree with the shortness [G17][G19], but Keller also mentions that the sentences still need to be complete [G19]. So rather than filling the space with as much information as one can fit in two pages, one should aim for less information, but clearly worded. Then again, Google Cloud Architecture Center [G13] says that they need to be just as long as is necessary to convey all relevant information related to an individual decision.

What information should be recorded is mostly agreed on. A summary of the recom-

Table 3.9: Summary of ADR field recommendations. Notice that these are recommended by different sources and those sources recommend different combinations of subsets of these recommended fields. The fields mentioned in the original introduction of ADRs [43] are underlined.

Field	Source	Year
Required fields		
Stakeholders	[G19]	2017
Status	[G19]	2017
Categories	[G19]	2017
Related (documents)	[G19]	2017
Authors/Team/Driver/Decision maker	[G19][G13]	2017/2021
Context/Problem	[G19][G15][G13][G2]	2017/2019/2021/2021
Decision	[G19][G15][G13]	2017/2019/2021
Consequences/Outcome/Result (of the decision)	[G19][G15][G23][G2]	2017/2019/2021/2021
Title	[G23]	2021
Requirements	[G13][G2]	2021
Critical user journey	[G13]	2021
Options/Alternatives	[G19][G29][G2][G13][G23][G1]	2017/2020-2022
Reasons/Reasoning	[G13][G23][G1][G29][G2]	2020-2022
Date/Timestamp	[G19][G19][G13][G1]	2017/2017/2021/2022
Optional fields		
Status	[G15]	2019
Date/Timestamp	[G15][G23]	2019/2021
Names of the reviewers	[G23]	2021

mended fields is in Table 3.9. The fields are divided into required and optional. The original ADR template contains a title, context, decision, status and consequences (Nygard, 2011). The papers included in the analysis of this thesis mostly follow this format, although there are some slight differences as described as follows. Google Cloud Architecture Center suggests more information: authors or team, context or problem, requirements, critical user journey, key options, decision and reasons, and timestamp [G13]. Two more papers [G19][G1] enforce the timestamp field and it is optionally enforced by another two [G15][G23]. Keller suggests a different template, which contains summary (date, driver [person], stakeholders, status, categories and outcome), context, related, alternatives, decision, and consequences [G19]. Both Janssen and Peinelt recommend keeping the information to the minimum with only three required fields. Janssen suggests the original context, decision and consequences fields and date and status only if really wanted [G15]. Peinelt suggests title, decision result (outcome) and considerations (other options and reasoning) and possibly date of the decision and names of the reviewers if needed [G23]. Among other template and tool resources, the GitHub ADR organisation recommends the

Y-statement format originally suggested by Zdun et al. (2013) [G2]. The Y-statement format is a template with the following structure "In the context of <requirement>, facing <concern>, we decided for <option> and neglected <option> to achieve <quality>, accepting downside <consequence>."

Three papers [G32][G7][G23] mentioned that the template does not really matter as long as one is chosen and used consistently. The point of ADRs is to answer "why" questions [G32]. ADRs should be easily accessible to everyone and be as basic and simple as possible and lack excessive formality [G23]. The documentation needs to be tailored for the needs of the project [G13]. Decisions always have options and those options should be recorded as well as the reasoning for the selection [G1][G29].

The location of ADRs can be with source code, a separate version control system, or a location accessible to most shareholders. Most analysed papers agree that the correct location of the ADRs is with the code in the same version control [G13][G15][G17][G19][G7]. Although, keeping the documentation in a separate location would help non-technical shareholders to also access it [G23]. It is also suggested that ADRs should be kept with the rest of the documentation, although it should still be version controlled [G29]. However, there is also the opinion that the location does not really matter as long as it is convenient for the team [G32].

How writing ADRs is integrated into the everyday work of practitioners affects the success of documentation. The process of recording ADRs needs to be delegated to someone for it to happen [G17]. It should also become a habit for the team. Peer review is an important part of the process [G17][G7]. This ensures that the correct things are recorded and the quality and usefulness of the ADRs. It is also suggested that ADRs should only be recorded after the final decision has been made because there is no point in recording ideas, which might not happen [G18]. Although, the original description of ADRs includes "proposed" as an acceptable status of an ADR. This is probably because it allows the recording of the reason for rejecting a certain solution, which might be important later.

3.3 RQ2. What are the benefits of the suggested documentation methods, practices and tools?

Certain methods, practices and tools are suggested or studied for a reason. They may solve a problem or aim to improve a requirement. A requirement can be, for example,

"low effort" or "non-intrusive". The summaries of the benefits are in Tables 3.10, 3.11, and 3.12. The benefits are grouped by who it applies to and what type of method, practise or tool it relates to as well as what feature enables it. The following subsections focus on explaining the benefits of the different approaches. The benefits are grouped by the type of approach (see Tables 3.4, 3.5, and 3.6 for the categories).

3.3.1 RQ2.1. What are the benefits to writers?

Guidelines for documentation were found to address project start-up times, support agile software development methods, effort (required to create documentation), quality of information, and cost of writing. Reducing the size of the documents has been observed to reduce project start-up times and support the use of agile software development methods [G24]. Recording only essential and documenting incrementally lessens the documentation effort [S15][S2]. Using file-based structure in documentation instead of ontology-based reduces the cost of writing [S17]. The cost of writing is balanced by the cost of using the written information. The more often the information is used, the more cost effective it is to write ontologies.

There are two papers about processes and they both mainly aim to make documentation faster and non-intrusive. Non-intrusiveness and speed are due to the processes being tied to the workflow of the design process itself [S21][S26].

Template approaches are low effort when the individual documents are kept small [S12]. The small document size also encourages regular documentation and still manages to produce the so-called big picture of architecture via the combination of these small documents. The template approach by Maric, Matkovic, Tumbas and Pavlićević [S24] is especially suited for agile software development methods due to using informal, non-traditional, and familiar methods that the team members are comfortable with.

ADR, being a specific type of template approach, shares the benefits of templates. They also benefit from simplicity, easy starting (without prior knowledge), non-intrusiveness and speed (of documenting). However, these benefits are sometimes dependent on the ADR template in use and other practices involving the decision documentation. When ADRs are stored with the source code, the effort of accessing is lowered and writing and reviewing is easier [G15][G18]. This also means that it is easier to include decision documentation in the existing workflow, which makes their use non-intrusive. The ADR templates are easy to work with when they are simple [G15][G19]. Starting documentation is straightforward

because the ADR template is supposed to both guide the memory and it may also contain instructions [G19]. Onboarding is easier for the same reason. Because they take very little effort and they are quick to make, they are also cheap [G18].

Tools tend to make documentation low effort and cost due to the reuse of partial information and automation [S6][S3]. Although, different types of tools have different levels of automation. Different tools are also used in different situations. Some extraction tools are only beneficial with projects that are old and a lot of material already exists from which to derive the decision documentation [S31]. This is dependant on what the tool takes in as an input. The creation tool by Brandner, Mayer and Weinreich [S8] specifically benefits from reusability. The tool is used to produce architecture profiles, which are combined from parts. A benefit of extraction tools is non-intrusiveness [S16]. The design can happen with as little attention put into documentation as possible, while the tool records the necessary information.

A summary of benefits by result group for writers is in Table 3.10.

Table 3.10: Benefits of different features of methods, practices and tools for writers.

Category	**Feature**	**Benefit**
Templates, ADR	Small size of the documents	Low effort, reduce project start up times, support the use of agile software development methods, regular documentation (up-to-date information)
Templates, ADR	Recording only essential information	Low effort, low cost
ADR	Simplicity	Easy to start documenting
ADR	Stored near source code	Low effort of access, easy to write and review
ADR	Integrated instructions	Better chance of writing correct content, easier onboarding
ADR	Memory prompts	Important information not forgotten, easier onboarding
ADR, Processes	Documenting as a part of the workflow	Faster (no task switching), low cost, non-intrusive
Tools, Creation tools	Reuse of partial information	Reusability, low effort, low cost
Tools, Extraction tools	Automation	Low effort, low cost, non-intrusive
Guidelines	Documenting incrementally	Low effort
Guidelines	File-based organization	Low cost of writing

3.3.2 RQ2.2. What are the benefits to readers?

The benefits for readers focus on minimal information, quality of information, ease of access and finding information, startup times and supporting agile software development practices.

The simplicity of ADRs makes decision-making easier as they are quick to read and easy to understand [G18][G17][G32]. This also translates into efficient onboarding [G17][G19]. Location near the source code also increases efficiency for readers as source control tools usually allow the reading of documents (other than code) as long as the format is supported. Designing new features is made easier as the reasoning is located close to the existing implementation [G18].

Creation tools aim to make decision-making and sharing information easier [S8]. Tool by Brandner [S8] also offers concrete guidance for decision-making. Tools for creating custom content based on user needs also benefit from not having too much information which increases reader efficiency [S2]. This also does not rely on writer discretion when it comes to the recording – potentially making the outcome for the reader more reliable.

Guidelines state that only essential decisions should be documented [S15]. This makes reading the documentation more efficient and less time can be spent looking for the essential information. When there is less documentation people are also more likely to read it, while large amounts of documents can cause the documentation to not be read at all. Dedicated search tools can help transform existing low value documentation into usable material [S30].

A summary of benefits for readers is in Table 3.11.

3.3.3 RQ2.3. What are the benefits to those who maintain the documentation?

The majority of the analysed literature did not touch on the subject of documentation maintenance, except for some template and ADR papers [S12][G17][G18][G32][S15].

The template by Diaz-Pace supports maintenance by keeping information about current system issues [S12]. This helps with documentation maintenance as it encourages actively reviewing documentation while performing other maintenance tasks.

ADRs benefit from close proximity to the source as well as minimal content. The close

Table 3.11: Benefits of different features of methods, practices and tools for readers.

Category	Feature	Benefit
Templates, ADR	Small size of the documents	Reduce project start up times, support the use of agile software development methods, regular documentation (up-to-date information)
Templates, ADR	Recording only essential information	Less to read
ADR	Simplicity	Easy to find information
ADR	Stored near source code	Effort of access, easy to review
ADR	Integrated instructions	Quality of content, easier onboarding
ADR	Memory prompts	Quality of content, easier onboarding
ADR, Processes	Documenting as a part of the workflow	Up-to-date-information
Tools, Extraction tools	Automation	Quality of content (less human errors)
Guidelines	Documenting incrementally	Up-to-date information
Guidelines	Ontology-based organization	Low cost of use / easy to find information

proximity of ADRs to source code makes it more efficient to make changes to documentation because changes can be made at the same time as code changes [G17][G18]. This benefit does not apply in the situations where the documents are stored elsewhere, as is suggested in some cases. Maintainers also benefit from not having to do it too much. When only essential information is documented, fewer changes are also required [G32][S15].

The summary of benefits to maintenance is in Table 3.12

Table 3.12: Benefits of different features of methods, practices and tools for maintainers.

Category	Feature	Benefit
Templates	Recording current system issues	Encourages reviewing decision documentation during maintenance tasks
Templates, ADR	Recording only essential information	Less changes are required
ADR	Stored near source code	Making changes is efficient

3.4 RQ3. What are the drawbacks of the suggested documentation methods or practices?

No method, practice, or tool is perfect and the benefits often come with trade-offs and challenges. For example, recording only a minimal amount of information requires knowledge of what is important. Summaries of the drawbacks are in Tables 3.13, 3.14, and 3.15. The next subsections focus on the drawbacks of the different approaches. The drawbacks are grouped by the type of approach (see Tables 3.4, 3.5 and 3.6 for the categories)

3.4.1 RQ3.1. What are the drawbacks to writers?

Drawbacks for writers often mean more work. It can also mean mental overhead and low motivation due to complicated processes, multiple differing documentation styles, and simply the method not aligning with the preferences of the individual. Other problems are unclear instructions causing too much information being recorded and having to go through training to use new tools.

The format of other documentation (non-decision) can affect how decision documentation is received. Filling in templates is sometimes considered as excessive overhead because the architecture design itself may be represented in a more visual manner [S26]. This means that the architecture design and decision documentation may have a disconnect, which causes the decision documentation to be less useful. This can make the entire decision documentation process seem pointless for those doing it.

ADRs are small and very easy to write, but it is also easy to use them incorrectly. Putting a lot of meaningless information into an ADR template results in a large amount of useless, low-quality documentation [G18]. Training people to correctly use an ADR template can take months.

Some tools require a lot of learning, which overloads the already cognitively challenging task of designing and also interferes with the design process [S21]. Dedicated creation tools in general suffer from the fact that practitioners rather use text-files than dedicated tools [G12].

Extraction tools still require post-processing, as the extracted information needs to be validated, which can be time consuming and tedious [S16]. The tool may also not capture everything [G28], and manual labour is still necessary to find additional information.

Document organisation can also have negative consequences on the writing. The drawback of an ontological approach is that it takes a lot of effort to make the necessary semantic annotations [S17]. When the results of that work are not used in the end, the cost effectiveness of the method becomes low.

When and how documentation is performed affects the effort or feeling of effort being put into documentation activities. Sometimes the documentation process can be too isolated from other parts of the design work, making it seem like an unnecessary extra task [S26]. This can be due to the documentation process not covering all design activities when documentation is supposed to be undertaken as a part of the design process. A summary of the drawbacks for writers is in Table 3.13.

Table 3.13: Drawbacks of different features of methods, practices and tools for writers.

Category	Feature	Drawback
Templates	Differing documentation formats (for example, decision documentation vs architecture design graphs)	Overhead, low motivation
ADR	Excessive simplicity	Easy to write unnecessary information, training required
Tools	Dedicated tools	Training required, low motivation (users rather write into simple text files)
Extraction tools	Post-processing required	High effort
Extraction tools	Automation missing information	High effort
Processes	Isolated documentation process	High effort, low motivation
Guidelines	Ontology-based organization	High effort

3.4.2 RQ3.2. What are the drawbacks to readers?

Drawbacks for readers include quality issues related to the content and the length of the documents as well as not being able to understand the relations between different documentation sources.

Documentation organisation has a big impact on the readers. With standard file-based approach information tends to become very scattered making necessary information hard to find [S17]. Readers also become dependent on the table of contents to find necessary information. The importance of the structure of information may become overly pronounced from the perspective of the readers. On the other hand, an ontology-based approach fails to deliver information about the relationships between different parts of knowledge.

ADRs can suffer from quality issues, which are frustrating to readers. ADRs can be low-quality, and contain, for example, general guidelines rather than actual analysis on why decisions were made and what consequences there are [G18]. Despite their small size there may also be too many of them, causing people to read them less [G18][G32]. Template approaches in general may result in a large number of documents of varying sizes. Therefore, going through these documents can be overwhelming and unattractive [S26]. Especially overly long documents are daunting to read [S12]. Connecting template information with architecture design documentation can be hard when they are documented separately and in different formats [S26]. This also makes decision-making harder, which is often the reason for reading documentation. When updating documents is allowed it can be hard for the readers to know what the new content is [S12]. This combined with documents that are over one page long causes users to not actually read and review the changes.

A summary of drawbacks for readers is in Table 3.14.

Table 3.14: Drawbacks of different features of methods, practices and tools for readers.

Category	Feature	Drawback
Templates	Differing documentation formats (for example, decision documentation vs architecture design graphs)	Different documents disconnected
Templates	Too long documents	Hard to find necessary information
Templates, ADR	Too many small documents	Hard to find necessary information
ADR	Excessive simplicity	Low quality documentation
ADR	Existing documentation updatable	Hard to find the updated parts, less motivation to read
Guidelines	File-based organization	Information scattered and hard to find

3.4.3 RQ3.3. What are the drawbacks to those who maintain the documentation?

Maintenance was not discussed extensively in the analysed literature. Drawbacks assigned to readers often apply to maintainers as well because maintaining documentation also requires reading it. Only strictly maintenance-related drawbacks are discussed in this section. Problems concerning maintenance cover documents being too large, updating cross-referencing and overlapping information.

Documentation organisation affects documentation maintenance. File-based approach re-

quires cross-referencing related parts of the documentation to keep the information coherent. Keeping these references up to date is not a small task [S17].

Using templates causes a mass of separate documents by default. The separate nature of the documents can cause overlapping information to be recorded [S12]. It can take a lot of effort to keep the documents from having, for example, conflicting assumptions. ADR is a type of template and when the collection of ADRs becomes large the maintenance becomes hard [G32]. A summary of the drawbacks for maintainers is in Table 3.15.

Table 3.15: Drawbacks of different features of methods, practices and tools for maintainers.

Category	**Feature**	**Drawback**
Templates, ADR	Too many small documents	Overlapping information, conflicting information
Guidelines	File-based organization	Hard to maintain cross-references

4 Discussion

The results of this thesis cover the tools, methods and practices discussed over the last five years. A total of 36 papers were analysed. In this chapter, the findings are first discussed separately for each research question. The last section discusses the limitations and threats to the validity of the review.

4.1 Findings related to RQ1: how architecture design decisions are documented

Four different approaches were found in the analyzed literature. The approaches were categorized as *Tools*, *Templates*, *Processes* and *Guidelines*. *Tools* were further categorized as *Creation*, *Extraction*, *Gamification* and *Search*.

The approaches discussed in the grey literature and scientific literature differ significantly. Comparing Tables 3.5 and 3.6, scientific papers mostly concentrate on tools and the analysed grey literature mostly focus on templates, especially ADRs. All mentions of ADR templates were found within the analysed grey literature, which is not surprising as the approach was originally made popular by a blog post. Only two scientific papers mentioned templates, and one of those was similar to ADRs. In this approach a single decision is recorded per document and the big picture is formed by the collection of individual documents. It appears that grey literature sources deem ADR as the expected way of documenting architecture design decisions. One tool was mentioned within the analysed grey literature and it was in the context of ADRs.

The term 'ADR' seems somewhat vague in definition when looking at the different template content suggestions (see Table 3.9 for the recommended fields). In this review, all sources using the term ADR were counted as ADR approaches. The recommended fields for ADRs differ a lot. The suggested storage location and the length and type of file are not always the same. Because the recommendations for ADRs are extremely varied it could be argued that any 'short', template type approach could be counted as an ADR. It is surprising how often the field 'Decision' itself is not included in the recommended fields. Instead, the decision information is included in another field like 'Title' or 'Decision Result'. In some

cases, the analysed papers might not mention all the fields, possibly assuming that they are obvious, such as the field 'Title'. The quality of the papers discussing ADRs varies from very high to very low and all of them are grey literature. The highest quality sources with a score above 5 (out of 6) include documentation [G2], an experience report [G18], a presentation [G17] and a book [G6].

In the category of other template approaches, there is a high-quality scientific paper discussing a very similar approach to ADRs, called High-level Design Stories (HLD) [S12]. The evidence level of the paper is 5 (industrial studies). A company in the insurance domain was modernizing their software systems and they did not have an effective documentation strategy. The modernizing also required many architecture decisions to be made. Existing documentation strategies were investigated, but they did not fulfil the needs of the project. For example, reviews were very important for this project to avoid creating more issues along the way. This is why a new strategy was developed. The paper describes modular artifacts recording design decisions and their context, architecture assumptions, quality-attribute analysis and system issues. The HLD approach appears more refined than the ADR approach, for example, the workflow has been described. This was expected as the ADR was introduced in a blog post and the HLD was introduced in a scientific paper. The HLD differs from the ADR, especially in the way it covers parts of the system instead of individual desicions, which can make the documents large. HLDs also record system issues, which is not typical for ADRs, although considering the large number of variations in the recommended fields for ADRs, it could be done. HLDs are especially good for validating decisions, but suffer from the documents easily becoming too large as their coverage is not defined strictly.

Using tools to aid documentation is the most common approach within the analysed scientific literature. The majority of the papers discussing tools (8 out of 14) are about Extraction tools, which automate at least a part of the process. This is expected as artificial intelligence is increasingly popular and rapidly developing in itself. The development of AI is seen in these tools as well. The earlier tools are simpler and require some manual labour. The later tools produce finished documentation and one even attempts to reduce the training requirements of AI.

One of the main concerns that the analysed scientific studies focus on is the cost-effectiveness of documentation. The benefits of documentation are recognised by scientists, but the uptake of the practices has been lacking in the industry. This is at least partially due to the perceived cost and unclear benefits, which is why automation is an attractive option.

Unfortunately, some of the Extraction tools require some manual labour in addition to the automated process. However, these tools are developing rapidly and they are becoming more consumer friendly. The older approaches require more manual labour than the more recent ones. See Table 3.8 for the summary of information, including the publication years. The quality of these studies varies a lot. The only paper having an evidence level 5 (industrial practice) is the one introducing Adex [S6] and published in 2019 – an extraction tool using natural language processing to extract information from chat logs and other sources. It appears that the requirement for more efficient documentation and tooling has been taken into account in the scientific research. The most recent tool, LoqBook, was found within the analysed grey literature and is a final product available for consumers [G23]. However, the quality score for the paper introducing LoqBook is only 2.7 out of 6 due to it being mentioned only in a blog post on the website of the LoqBook company.

The next biggest group within the analysed scientific papers is the Creation tools. The quality of the papers in the Creation tools category is not of the industrial level as they all belong to the evidence level categories 2–4. The papers in the highest level evidence group include ProSAD [S2] (a documentation strategy recommender) and AKB [S8] (a web application for documenting and sharing). The Creation tools remove the need to put a lot of effort into how the documentation is organised and stored, but they suffer from having to learn another system just for documentation and possibly having to pay for the use of the tool. What really sets these tools apart from others, is the support for collaboration as well as re-using and highlighting information. However, cost-effectiveness being a strong driver in the industry this approach alone is unlikely to be the first choice. The tool presented in the most recent paper combines creation with extraction and search [G23] (see Table 3.7 for all of the creation tools). Combining the features of different types of tools is likely to make them more attractive. However, according to the analysed grey literature, it looks like using simple text files for documentation is still prevalent as this tends to be the easiest, cheapest and fastest way of getting started with documentation.

The issue with most of the discovered tools is that they tend to be pure academic studies or just demonstrations. Out of the six academic papers discussing tools two are also preceding papers to one in the industrial practice category. The paper discusses the extraction tool called Adex [S6], which is being developed together with an industrial partner. So, even though it is used in practice it is still in development. The extraction tool called Recovar [G28] and two of the creation tools are also still in development. One creation

tool is in the process of being tested in an industrial setting [S8] and another has only initial evidence of its feasibility [S2]. The academic studies also include the gamification tool which was not demonstrated to be viable due to the setup of the study [S25]. A tool that is available for use was found in the grey literature and is tied to the ADR approach – LoqBook [G23]. This further supports the ADRs as the method of choice.

The process of documentation affects all methods except those that are automated. The main idea is to promote good habits and make documenting as small and painless a task as possible. Only two papers support paying attention to the steps of documentation, but their evidence levels are relatively high – an academic study [S21] and an industrial study [S26].

The vagueness of some guidelines and suggestions may be their downfall. A statement like "record only minimal information" is good in theory, but it is very hard to gauge what really is minimal, but still enough information, unless one is already quite experienced. This may explain the popularity of the ADR approach as it tends to come with concrete rules like length of 1–2 pages. This does not, however, automatically translate into good quality content. Low-quality content can be combatted by adding instructions to the templates themselves, but it is not a cure-all as it still requires some understanding. Guidelines to help determine what is important to individual projects could be helpful here as suggested by Gerdes et al. [S15] for finding essential information. The documentation method must reliably produce a positive outcome for the project. Producing incorrect documentation ends up being frustrating to practitioners and a waste of resources. The team should be comfortable with the documentation method to successfully perform the task of documenting.

Choosing the best method, practise or tool for architecture design decision documentation requires analysing the project itself and the needs of the project. Sometimes the approaches are aimed at systems at a certain lifecycle stage or systems that are a certain size. This is simple enough, but other times it is necessary to really think about how much documentation is necessary, who needs it and for what reasons. Different approaches can also be adjustable depending on the needs of the system as well. Certain guidelines and trade-offs should be considered when selecting a suitable way of documenting. Only important information should be recorded, but what is important depends on the individual system. The more risk is involved, the better the decisions should be recorded.

4.2 Findings related to RQ2: benefits of the methods and tools

Not all the papers gave a clear reason why it was suggested – this was especially apparent with the analyzed grey literature papers. In many cases, the reasoning was more about documenting decisions in general rather than what are the benefits of the discussed method in comparison to other methods. Then again, some benefits, such as documenting being an integral part of the workflow, are present when using any method or tool that requires manual input. Some benefits associated with a specific method could theoretically be adopted by other methods or tools. For example, the benefits of storage location near source code is not necessarily unique to ADRs, although, there is no evidence of the benefit of this in connection to other methods within this review.

Sometimes the methods give a lot of freedom to choose how they are used. This is common to agile software development practices, as flexibility usually allows teams to work more efficiently. However, some benefits mentioned are only present if the methods are used in the intended ways. Especially ADRs and similar template approaches, where there is a collection of short documents, can have their benefits diminished if they are used incorrectly. Many different types of ADR templates and ways of using them exist, so it helps to be aware of the trade-offs being made when choosing.

A minimal amount of work is the goal of most methods and tools. The majority of the approaches are tools aimed at automatically extracting decision information or creating documentation. Even templates aim to minimise the amount of text produced. This is not only beneficial to writing but also to reading and maintenance.

Motivation to actually do the work helps with most manual approaches. This is the result of specific processes. When the work is divided into small, simple chunks, it is much more manageable from the cognitive perspective. Multiple other factors also support the motivation, such as the amount of content required to be written or read. Approaches demanding less content are more beneficial in this sense.

Usability is essential to readers because unused documentation is essentially a waste of resources to even produce. This is where the organisation, content, size and location become important. Creation tools remove the need to put too much thought into this, but it is balanced out by the need to select and possibly learn a new tool. ADRs and other templates tend to give suggestions related to usability, but there are also different

suggestions about what is best. The freedom can be great if the writers know what they are doing, but it also gives space for mistakes. Usability was not really mentioned in the context of automatic creation, but is probably something that should be considered. The size of the documentation is less important if the search tools are good.

Reliability is another benefit for readers. This covers the quality of the content (correct and complete information) as well as the documentation being up-to-date. These are difficult to improve in approaches relying on human input, as there is always a chance for mistakes. Especially extraction tools offer improved reliability. Although all automation solutions are not equally reliable, their reliability is still higher than that of human input due to automation being more consistent because people can forget and overlook important parts by mistake.

Low cost is often a necessity because documentation is not something that brings tangible profit quickly. Documentation tends to be neglected if the cost threatens to become too high. Extraction tools tend to be very cost-effective, especially the most modern ones, which require less human intervention. Simple, text-based approaches, like templates, are also an excellent option for cutting costs because they do not require much set up.

4.3 Findings related to RQ3: drawbacks of the methods and tools

Like the benefits, the drawbacks were also not widely discussed. However, some papers identified general issues with previous documentation practices. Sometimes the drawbacks are only present if a particular documentation method is used incorrectly. Some drawbacks have to be accepted as a tradeoff when prioritizing the requirements of a project.

Length of the documents can be problematic. If the length of the document is not limited, they become too long very quickly. This is a problem sometimes even with templates that are meant to be short. Sometimes, even a single page can seem daunting when someone is looking for specific information.

Effort is often seen as the issue with writing documentation. Choices in file-type can reduce the effort. For example, ontology-based approach takes a lot of effort despite its benefits, which is why it is sometimes opted against [S17]. More often, effort is countered with a smaller amount of optimised content, tools to reduce the need to organise and maintain relationships between pieces of information and ultimately automation.

The process of documenting design decisions can also adversely affect other design tasks. When a practitioner is forced to make a separate effort for documenting, time and effort are taken away from the actual design tasks. The process of documenting decisions should be strongly linked to the process of making decisions.

Some automation solutions rely on teaching an artificial intelligence, which still takes some effort. They also tend to rely on a large amount of existing data, which is not optimal for younger projects. More recent natural language processing solutions are more attractive and are meant to keep documentation up-to-date when a project is on going. However, automation may not always be reliable, which means that the effort only moves from writing to reviewing the output.

The drawback of Creation tools, which store and organise information that the user inputs, is the fact that practitioners often would rather just use text files instead. This is most likely due to convenience, which is typical for agile software development methods. Dedicated tools require decisions (which tool and why) and learning the use of it, as well as, committing to using the tool once the selection process has happened. The tool could also require purchasing a license, which would make the threshold even higher.

Gamification was not found to add any benefits. However, only one study related to gamification was analyzed. Additionally, the experiment set-up did not take into account the requirements of the game, which invalidated its benefits. Therefore, gamification cannot be dismissed. Gamification tends to require the use of a certain tool which supports the game. This leads to gamification suffering from the same issues as Creation tools.

4.4 Limitations and threats to validity

This section takes a look at the recognized limitations and discusses validity. The explicit description of the methodology supports the validity. However, the most noticeable shortcomings are due to only one person working on the review and the major keywords being common.

This literature review is not likely to cover all relevant studies. The fact that some papers were found only in the grey literature search when they were available on the Web of Science database indicates this. The availability of the papers was confirmed by performing an individual search in the scientific database for each scientific paper found in the Google search. The papers being available but not found with the search string

demonstrates that defining the search string was problematic. The key terms are very common and many specific keywords and filters had to be used. So, at least some on-topic results did not show up in the results of the scientific search. Also, only one scientific library and one generic search engine were used. More relevant material could have been found if more libraries and search engines were utilised. The repeatability of the study suffers from the optimization of results done by the Google algorithm. The results may not always be the same even if the same key terms and settings are used. Relevant grey literature material was likely to also have been left out of the study. There were mentions of tools related to the ADRs, but only one such tool showed up in the search results.

While crafting the search string it became clear that the terminology is fuzzy. Papers mentioned, for example, *architecture decisions*, *design decisions* and *architectural knowledge*. While these terms mean different things the line between them was not always clear. The terms might sometimes be used interchangeably. The search term was designed to cover as many ways of discussing architecture decisions as possible. Because of this fuzziness there is the possibility that some data was misinterpreted.

The quality of the studies was evaluated by extracting quality criteria from each study. The quality assessment of the scientific papers observes the context of the study, as suggested by Alves et al., 2010. Each paper was given an evidence level rating (see Table 3.1). The evidence levels of the papers were fairly evenly divided into the lower end and the higher end of the evidence level categories. The papers with a higher evidence level were paid more attention when evaluating, for example, the usefulness of the found tools. The validity of the studies was not assessed, which could affect the results somewhat. For example, one of the papers [S25] mentions that the results are not transferrable to a professional environment because the experiments were conducted on students and there are significant differences in the circumstances of these groups. So, the external validity of the study could be tied to the potential viability of the method or tool.

Each grey literature paper was given a quality score based on six quality criteria (see Table 3.3). Each criterion had the same weight, which means that even a paper with low outlet control was able to get a relatively high score, such as 4.1/6 [G7]. The impact metrics were oversimplified for this thesis. It did not take into account the different types of source materials and brought down some quality scores unfairly. The impact metric was based on the traffic on the website and its domain. This does not give a realistic impact metric on all types of results. For example, one of the search results presented a book. The traffic on that particular webpage presenting the book is not likely to accurately represent the

impact of the book itself. Additional metrics could have helped the scores to show the impact of each paper more accurately. A high quantity (8 out of 15) of the grey literature papers are blogs making their general quality relatively low.

The data extraction form was piloted to find potential problems before collecting the data, but it was not reviewed. As it is, the form did help collect the necessary information. Nevertheless, the fields could have been more refined or fine-grained. For example, the extraction form could have explicitly defined fields for collecting information about the status of the method or tool. In other words, whether it is ready to be used or years away from a practical solution. This would mostly apply to the scientific literature as the grey literature tends to discuss solutions that are ready or already in use. There could have also been a separate field for the "problem" that the tool or method attempts to solve to help analyse the focus and direction of the subject.

Only one person performed everything from planning to data extraction, which means that some of the data may contain errors. Also, the inclusion and exclusion criteria are defined and explained, but there were no sanity checks because only one person made all the decisions about inclusion and exclusion. The same applies to the validation of the entire process.

Some results are based on only a subset of the analysed papers. Not all of the papers gave a clear reason for why the method, approach or tool itself was beneficial to documenting architecture decisions. Instead, papers often focused on the benefits of documentation in general and not so much on why one documentation method should be picked over another. This was especially apparent in the analyzed grey literature papers. Some individual categories of tools and methods were quite small because of the relatively small amount of papers covering a large area of study. It is acceptable because this thesis builds an overview of what is out there. However, reliable conclusions about the details of all individual categories cannot be drawn.

5 Conclusions

The goal of this review was to examine the current situation of software architecture design documentation methods, tools, and practices. The focus was on modern continuous software development. The current research was assessed through scientific papers, and the current practitioner's point of view was assessed through grey literature. The benefits and drawbacks of the methods, practices, and tools were also studied to assess the reasons behind certain developments and potential directions in the future. Altogether, 36 out of 64 discovered papers were analysed.

The results show that research mainly concentrates on tools. Especially automated extraction tools are under ongoing research. The grey literature indicates a preference for simple, template based documentation. This is mainly because that the practitioners appear established in their use of ADRs. However, simple template based documentation does not exclude tool usage. Practitioners require the tools to be extremely cost-effective and simple, partly because documentation is not especially appreciated as a practice, despite it being proven to be beneficial. This puts a lot of demands on the research. Automation in itself is not enough if the content it produces is not high-enough quality. Effort needs to be truly reduced instead of just shifting the work from writing documentation to something else, when it is still documentation-related work.

The benefits of individual methods, practices, and tools were not always extensively discussed in the analysed literature, especially in the grey literature. However, the benefits mainly focus on reducing the amount of work and ensuring the usability and reliability of documentation while keeping the cost as low as possible. What is seen as a drawback is a high cost, high effort and producing too much or badly organised content. There are requirements and tradeoffs to be considered when choosing how to do the documentation because the demands of the project itself affect what kind of documentation is good enough. There is a delicate balance between too much and not enough documentation.

The different approaches to the issues of documentation all have their own merits. The perfect tool is likely to combine multiple aspects into a single tool. Automation would help a lot with the initial issue of neglecting documentation. The content would still need to be editable, shareable, searchable, conveniently located (connected to the source code) and of good quality. Due to the different needs of different types of projects customizability

would be quite important. Low motivation for documenting needs to be taken into account. Gamification deserves more research because it still has potential. However, the rewards need to be interesting enough as it is not likely that people are interested in points that mean nothing if the task to gain them is tedious. Supporting the process of design and documentation is also important, especially with less experienced professionals. Although, making architecture design decisions is a whole area of study in itself. Maintenance of documentation should be paid more attention to because documentation loses all of its value once it is out-of-date. It would be nice to see some automation or linking between the documentation and the code. As reading through the documentation for maintenance purposes tends to be a problem, it could be reduced by automatic indication of related code being changed.

Because ADRs are already the standard way of documenting architecture decisions an interesting next step would be to study them further and develop tools for them specifically. Further research would need to be conducted on how widely ADRs are used within the industry. Only one ADR tool appeared in this review, so further research on the already existing ADR tools would also be required. Another interesting question is what motivates professionals to document. It would be interesting to study cases where decision documentation was done well. Of course, such cases would have to be found first and the definition of “done well” would need to be established.

Bibliography

Reviewed – Web Of Science

[S1] Ahn, H., Kang, S., and Lee, S. (2018). "Reconstruction of Execution Architecture View Using Dependency Relationships and Execution Traces". In: *33rd Annual ACM Symposium on Applied Computing (ACM SAC)*. Assoc Computing Machinery, pp. 1417–1424. ISBN: 978-1-4503-5191-1. DOI: 10.1145/3167132.3167284.

[S2] Andres Diaz-Pace, J., Villavicencio, C., Schiaffino, S., Nicoletti, M., and Vazquez, H. (2016). "Producing Just Enough Documentation: An Optimization Approach Applied to the Software Architecture Domain". In: *Journal on Data Semantics* 5.1, pp. 37–53. ISSN: 1861-2032. DOI: 10.1007/s13740-015-0053-0.

[S3] Bhat, M., Shumaiev, K., Biesdorf, A., Hohenstein, U., and Matthes, F. (2017). "Automatic Extraction of Design Decisions from Issue Management Systems: A Machine Learning Based Approach". In: *11th European Conference on Software Architecture (ECSA)*. Vol. 10475. Lecture Notes in Computer Science. Springer International Publishing Ag, pp. 138–154. ISBN: 978-3-319-65831-5; 978-3-319-65830-8. DOI: 10.1007/978-3-319-65831-5_10.

[S4] Bhat, M., Shumaiev, K., Biesdorf, A., Hohenstein, U., Hassel, M., and Matthes, F. (2017). "An Ontology-based Approach for Software Architecture Recommendations". In: *23rd Americas Conference on Information Systems (AMCIS)*. ISBN: 978-0-9966831-4-2.

[S5] Bhat, M., Shumaiev, K., and Matthes, F. (2017). "Towards a framework for managing architectural design decisions". In: *11th European Conference on Software Architecture (ECSA)*, pp. 55–58. DOI: 10.1145/3129790.3129799.

[S6] Bhat, M., Tinnes, C., Shumaiev, K., Biesdorf, A., Hohenstein, U., and Matthes, F. (2019). "ADeX: A Tool for Automatic Curation of Design Decision Knowledge for Architectural Decision Recommendations". In: *IEEE International Conference on Software Architecture (ICSA-C)*, pp. 158–161. ISBN: 978-1-7281-1876-5. DOI: 10.1109/icsa-c.2019.00035.

[S7] Brandner, K. and Weinreich, R. (2019). "A Recommender System for Software Architecture Decision Making". In: *13th European Conference on Software Archi-*

tecture (ECSA). Assoc Computing Machinery, pp. 22–25. ISBN: 978-1-4503-7142-1. DOI: 10.1145/3344948.3344959.

[S8] Brandner, K., Mayer, B., and Weinreich, R. (2019). "Software Architecture Knowledge Sharing with the Architecture Knowledge Base (AKB)". In: *13th European Conference on Software Architecture (ECSA)*, pp. 30–33. ISBN: 978-1-4503-7142-1. DOI: 10.1145/3344948.3344960.

[S9] Capilla, R., Jansen, A., Tang, A., Avgeriou, P., and Babar, M. A. (2016). "10 years of software architecture knowledge management: Practice and future". In: *Journal of Systems and Software* 116, pp. 191–205. ISSN: 0164-1212. DOI: 10.1016/j.jss.2015.08.054.

[S10] Cho, C., Lee, K. S., Lee, M., and Lee, C. G. (2019). "Software Architecture Module-view Recovery Using Cluster Ensembles". In: *IEEE Access* 7, pp. 72872–72884. ISSN: 2169-3536. DOI: 10.1109/access.2019.2920427.

[S11] Dasanayake, S., Aaramaa, S., Markkula, J., and Oivo, M. (2019). "Impact of requirements volatility on software architecture: How do software teams keep up with ever-changing requirements?" In: *Journal of Software-Evolution and Process* 31.6, p. 19. ISSN: 2047-7473. DOI: 10.1002/smr.2160.

[S12] Diaz-Pace Andres, J. and Bianchi, A. J. (2019). "High-level Design Stories in Architecture-centric Agile Development". In: *IEEE International Conference on Software Architecture (ICSA-C)*, pp. 137–144. ISBN: 978-1-7281-1876-5. DOI: 10.1109/icsa-c.2019.00032.

[S13] Farwick, M., Schweda, C. M., Breu, R., and Hanschke, I. (2016). "A situational method for semi-automated Enterprise Architecture Documentation". In: *Software and Systems Modeling* 15.2, pp. 397–426. ISSN: 1619-1366. DOI: 10.1007/s10270-014-0407-3.

[S14] Galster, M., Treude, C., and Blincoe, K. (2019). "Supporting Software Architecture Maintenance by Providing Task-specific Recommendations". In: *35th IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE International Conference on Software Maintenance. IEEE, pp. 370–372. ISBN: 978-1-7281-3094-1. DOI: 10.1109/icsme.2019.00060.

[S15] Gerdes, S., Jasser, S., Riebisch, M., Schroder, S., Soliman, M., and Stehle, T. (2016). "Towards the Essentials of Architecture Documentation for Avoiding Architecture Erosion". In: *Acm Proceedings of the 10th European Conference on Software Architecture Workshops (ECSA-W)*, p. 4. DOI: 10.1145/2993412.3004844.

[S16] Gilson, F. and Weyns, D. (2019). "When Natural Language Processing Jumps into Collaborative Software Engineering". In: *IEEE International Conference on Software Architecture (ICSA-C)*. IEEE, pp. 238–241. ISBN: 978-1-7281-1876-5. DOI: 10.1109/icsa-c.2019.00049.

[S17] Graaf, K. A. de, Liang, P., Tang, A., and Vliet, H. van (2016). "How organisation of architecture documentation affects architectural knowledge retrieval". In: *Science of Computer Programming* 121, pp. 75–99. ISSN: 0167-6423. DOI: 10.1016/j.scico.2015.10.014.

[S18] Graaf, K. A. de, Tang, A., Liang, P., and Khalili, A. (2017). "Querying Software Architecture Knowledge as Linked Open Data". In: *IEEE International Conference on Software Architecture (ICSA)*, pp. 274–277. ISBN: 978-1-5090-4793-2. DOI: 10.1109/icsaw.2017.13.

[S19] Guillem, A., Bruseker, G., and Ronzino, P. (2017). "Process, concept or thing? Some initial considerations in the ontological modelling of architecture". In: *International Journal on Digital Libraries* 18.4, pp. 289–299. ISSN: 1432-5012. DOI: 10.1007/s00799-016-0188-0.

[S20] Hesse, T. M. and Paech, B. (2016). "Documenting Relations Between Requirements and Design Decisions: A Case Study on Design Session Transcripts". In: *22nd International Working Conference on Requirements Engineering - Foundation for Software Quality*. Vol. 9619. Lecture Notes in Computer Science. Springer International Publishing Ag, pp. 188–204. DOI: 10.1007/978-3-319-30282-9_13.

[S21] Jong, P. de, Werf, J. van der, Steenbergen, M. van, Bex, F., and Brinkhuis, M. (2019). "Evaluating Design Rationale in Architecture". In: *2019 IEEE International Conference on Software Architecture Companion (ICSA-C 2019)*, pp. 145–152. DOI: 10.1109/icsa-c.2019.00033.

[S22] Keim, J. and Koziolek, A. (2019). "Towards Consistency Checking between Software Architecture and Informal Documentation". In: *IEEE International Conference on Software Architecture (ICSA-C)*, pp. 250–253. ISBN: 978-1-7281-1876-5. DOI: 10.1109/icsa-c.2019.00052.

[S23] Lytra, I., Carrillo, C., Capilla, R., and Zdun, U. (2020). "Quality attributes use in architecture design decision methods: research and practice". In: *Computing* 102.2, pp. 551–572. ISSN: 0010-485X. DOI: 10.1007/s00607-019-00758-9.

[S24] Maric, M., Matkovic, P., Tumbas, P., and Pavlicevic, V. (2016). "Documenting Agile Architecture: Practices and Recommendations". In: *Information Systems:*

Development, Research, Applications, Education 264, pp. 56–71. ISSN: 1865-1348. DOI: 10.1007/978-3-319-46642-2_4.

[S25] Mayer, B. and Weinreich, R. (2019). "The Effect of Gamification on Software Architecture Knowledge Management: A Student Experiment and Focus Group Study". In: *34th ACM/SIGAPP Annual International Symposium on Applied Computing (SAC)*, pp. 1731–1740. ISBN: 978-1-4503-5933-7. DOI: 10.1145/3297280.3297449.

[S26] Roldan, M. L., Gonnet, S., and Leone, H. (2016). "Operation-based approach for documenting software architecture knowledge". In: *Expert Systems* 33.4, pp. 313–348. ISSN: 0266-4720. DOI: 10.1111/exsy.12152.

[S27] Rost, D. and Naab, M. (2016). "Task-Specific Architecture Documentation for Developers Why Separation of Concerns in Architecture Documentation is Counterproductive for Developers". In: *10th European Conference on Software Architecture Workshops (ECSA)*. Vol. 9839. Lecture Notes in Computer Science. Springer International Publishing Ag, pp. 102–110. ISBN: 978-3-319-48992-6; 978-3-319-48991-9. DOI: 10.1007/978-3-319-48992-6_7.

[S28] Schroder, S. and Riebisch, M. (2018). "An Ontology-Based Approach for Documenting and Validating Architecture Rules". In: *12th European Conference on Software Architecture (ECSA)*. Assoc Computing Machinery. ISBN: 978-1-4503-6483-6. DOI: 10.1145/3241403.3241457.

[S29] Sobernig, S. and Zdun, U. (2016). "Distilling Architectural Design Decisions and their Relationships using Frequent Item-Sets". In: *13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pp. 61–70. ISBN: 978-1-5090-2131-4. DOI: 10.1109/wicsa.2016.9.

[S30] Su, M. T., Hosking, J., Grundy, J., and Tempero, E. (2016). "Usage-based chunking of Software Architecture information to assist information finding". In: *Journal of Systems and Software* 122, pp. 215–238. ISSN: 0164-1212. DOI: 10.1016/j.jss.2016.09.009.

[S31] Tamburri, D. A. and Kazman, R. (2018). "General methods for software architecture recovery: a potential approach and its evaluation". In: *Empirical Software Engineering* 23.3, pp. 1457–1489. ISSN: 1382-3256. DOI: 10.1007/s10664-017-9543-z.

[S32] Ven, J. S. van der and Bosch, J. (2016). "Busting Software Architecture Beliefs A Survey on Success Factors in Architecture Decision Making". In: *42nd Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA)*.

IEEE Computer Soc, pp. 42–49. ISBN: 978-1-5090-2819-1. DOI: 10.1109/seaa.2016.35.

[S33] Yang, C., Liang, P., Avgeriou, P., Eliasson, U., Heldal, R., Pelliccione, P., and Bi, T. T. (2017). "An industrial case study on an architectural assumption documentation framework". In: *Journal of Systems and Software* 134, pp. 190–210. ISSN: 0164-1212. DOI: 10.1016/j.jss.2017.09.007.

Reviewed – Google

[G1] arc42 contributors (2022). *9. Architecture Decisions*. URL: https://docs.arc42.org/section-9/.

[G2] Architectural Decision Records community contributors (2021). *ADR*. URL: https://adr.github.io/.

[G3] Atmakur, N. (2020). *What is Architecture Decision Record (ADR)? Do I need it? How is it useful?* Blog. URL: https://www.linkedin.com/pulse/what-architecture-decision-record-adr-do-i-need-how-useful-atmakur.

[G4] Blake, J. (2020). *When Should I Write an Architecture Decision Record*. URL: https://engineering.atspotify.com/2020/04/when-should-i-write-an-architecture-decision-record/.

[G5] Capilla, R., Jolak, R., Chaudron, M. R. V., and Carrillo, C. (2020). "Design Decisions by Voice: The Next Step of Software Architecture Knowledge Management". In: *Human-Centered Software Engineering: 8th IFIP WG 13.2 International Working Conference, HCSE 2020, Eindhoven, The Netherlands, November 30 – December 2, 2020, Proceedings*. Eindhoven, The Netherlands: Springer-Verlag, pp. 166–177. ISBN: 978-3-030-64265-5. DOI: 10.1007/978-3-030-64266-2_10.

[G6] Cervantes, H. and Kazman, R. (2016). *Designing Software Architectures: A Practical Approach (SEI Series in Software Engineering)*. ISBN: 978-0134390789. URL: https://www.informit.com/articles/article.aspx?p=2738304&seqNum=6.

[G7] Chappen, E., Hofschneider, R., King, M., and Ballard, T. (2021). *Architecture Decision Records: Helpful now, invaluable later*. Blog. URL: https://18f.gsa.gov/2021/07/06/architecture_decision_records_helpful_now_invaluable_later/.

[G8] Deshpande, T. (2018). *A Simple but Powerful Tool to Record Your Architectural Decisions*. Blog. URL: https://betterprogramming.pub/here-is-a-simple-yet-powerful-tool-to-record-your-architectural-decisions-5fb31367a7da.

[G9] Dorado, S. H. and Hurtado, J. A. (2019). *Documenting architectural rationale using source code annotations: An exploratory study*. Ed. by F. Harris, S. Dascalu, S. Sharma, and R. Wu. DOI: 10.29007/f5md.

[G10] Enstrom, D. W. (2018). *Guideline: Architectural Decision*. URL: http://www.unified-am.com/UAM/UAM/guidances/guidelines/uam_architectural_decision_BAE7AFA2.html.

[G11] Evans, P. (2018). *Lightweight Architecture Decision Records*. Blog. URL: https://peterevans.dev/posts/lightweight-architecture-decision-records/.

[G12] Gilson, F., Annand, S., and Steel, J. (2020). "Recording Software Design Decisions on the Fly". In: Joint Proceedings of SEED NLPaSE co-located with 27th Asia Pacific Software Engineering Conference 2020.

[G13] Google Cloud (2021). *Architecture decision records overview*. URL: https://cloud.google.com/architecture/architecture-decision-records.

[G14] Hesse, T. M., Kuehlwein, A., and Roehm, T. (2016). "DecDoc: A Tool for Documenting Design Decisions Collaboratively and Incrementally". In: *Proceedings 2016 1st International Workshop on Decision Making in Software Architecture*, pp. 30–37. DOI: 10.1109/march.2016.9.

[G15] Janssen, J. W. (2019). *Documenting Your Architectural Decisions*. Blog. URL: https://www.avisi.nl/blog/documenting-your-architectural-decisions.

[G16] Kapuscik, J. (2020). *What's an Architecture Decision Record?* Blog. URL: https://betterprogramming.pub/what-is-architecture-decision-record-110c597c13d.

[G17] Keeling, M. and Runde, J. (2017). *Architecture Decision Records in Action*. Powerpoint. URL: https://resources.sei.cmu.edu/asset_files/Presentation/2017_017_001_497746.pdf.

[G18] Keeling, M. and Runde, J. (2018). *Share the Load: Distribute Design Authority with Architecture Decision Records*. Conference Proceedings. URL: https://www.agilealliance.org/resources/experience-reports/distribute-design-authority-with-architecture-decision-records/.

[G19] Keller, F. (2017). *Documenting Architecture Decisions*. Blog. URL: https://www.fabian-keller.de/blog/documenting-architecture-decisions/.

[G20] McLintock, A. (2020). *Key Design Decisions*. Blog. URL: https://www.cafeassociates.co.uk/2020/10/28/key-design-decision-documents/.

[G21] Morris, B. (2020). *Architecture without documentation is incomplete*. Blog. URL: https://www.ben-morris.com/architecture-without-documentation-is-incomplete/.

[G22] Nigh, M. (2021). *How to Document Software Architecture Decisions*. URL: https://medium.com/leading-literally/how-to-document-software-architecture-decisions-9529a3a3efb6.

[G23] Peinelt, J. (2021). *Architectural Decision Record (ADR) Templates in Markdown and Excel*. Blog. URL: https://loqbooq.app/blog/architectural-decision-record-templates.

[G24] Poort, E. (2019). *Value-driven Architecture Documentation*. Blog. URL: https://eltjopoort.nl/blog/2019/02/01/value-driven-architecture-documentation/.

[G25] Rupp, H. W. (2021). *Why you should be using architecture decision records to document your project*. Blog. URL: https://www.redhat.com/architect/architecture-decision-records.

[G26] Sami, M. (1027). *Architectural Design Decisions*. Blog. URL: https://melsatar.blog/2017/04/29/architectural-design-decisions/.

[G27] Scheufler, B. (2020). *Documenting Design Decisions using RFCs and ADRs*. Blog. URL: https://brunoscheufler.com/blog/2020-07-04-documenting-design-decisions-using-rfcs-and-adrs.

[G28] Shahbazian, A., Kyu Lee, Y., Le, D., Brun, Y., and Medvidovic, N. (2018). "Recovering Architectural Design Decisions". In: *2018 IEEE International Conference on Software Architecture (ICSA)*, pp. 95–9509. DOI: 10.1109/ICSA.2018.00019.

[G29] Smith, S. (2020). *Getting Started with Architecture Decision Records*. Blog. URL: https://ardalis.com/getting-started-with-architecture-decision-records/.

[G30] Srikrishnan, Y. (2017). *Documenting Architecture*. Blog. URL: https://dzone.com/articles/documenting-architecture-1.

[G31] Taylor, T. (2020). *4 best practices for creating architecture decision records*. URL: https://www.techtarget.com/searchapparchitecture/tip/4-best-practices-for-creating-architecture-decision-records.

[G32] Zimmermann, O. (2021). *Architectural Decisions — The Making Of*. Blog. URL: https://ozimmer.ch/practices/2020/04/27/ArchitectureDecisionMaking.html.

References

Alves, V., Niu, N., Alves, C., and Valença, G. (2010). "Requirements engineering for software product lines: A systematic literature review". In: *Information and software technology* 52.8, pp. 806–820. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2010.03.014.

AsciiDoc (2022). URL: https://asciidoc.org/ (visited on 06/19/2022).

Bosch, J. (2004). "Software Architecture: The Next Step". In: *Software Architecture*. Ed. by F. Oquendo, B. C. Warboys, and R. Morrison. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 194–199. ISBN: 978-3-540-24769-2.

Fairbanks, G. (2010). *Architecture Haiku*. Presentation. URL: https://de.slideshare.net/matthewmccullough/architecture-haiku.

Garousi, V., Felderer, M., and Mäntylä, M. V. (2019). "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering". In: *Information and software technology* 106, pp. 101–121. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2018.09.006.

Huynh Khanh Vi, T., Börstler, J., Ali, N. bin, and Unterkalmsteiner, M. (2022). "How good are my search strings? Reflections on using an existing review as a quasi-gold standard". In: *e-Informatica Software Engineering Journal* 16.1. ISSN: 2084-4840. DOI: 10.37190/e-Inf220103.

Ivarsson, M. and Gorschek, T. (2010). "A method for evaluating rigor and industrial relevance of technology evaluations". In: *Empirical software engineering: an international journal* 16.3, pp. 365–395. ISSN: 1382-3256. DOI: 10.1007/s10664-010-9146-4.

Kazman, R., Goldenson, D. R., Monarch, I., Nichols, W. R., and Valetto, G. (2016). "Evaluating the Effects of Architectural Documentation: A Case Study of a Large Scale Open Source Project". In: *IEEE Transactions on Software Engineering* 42, pp. 220–260.

Kruchten, P., Lago, P., and Vliet, H. van (2006). "Building up and Reasoning about Architectural Knowledge". In: *Proceedings of the Second International Conference on Quality of Software Architectures*. QoSA'06. Västerås, Sweden: Springer-Verlag, pp. 43–58. ISBN: 3540488197. DOI: 10.1007/11921998_8.

Manteuffel, C., Avgeriou, P., and Hamberg, R. (2018). "An exploratory case study on reusing architecture decisions in software-intensive system projects". In: *The Journal of systems and software* 144, pp. 60–83. ISSN: 0164-1212. DOI: 10.1016/j.jss.2018.05.064.

Nygard, M. (2011). *Documenting architecture decisions*. Blog. URL: https://www.cognitect.com/blog/2011/11/15/documenting-architecture-decisions.

Perry, D. E. and Wolf, A. L. (1992). "Foundations for the Study of Software Architecture". In: *SIGSOFT Softw. Eng. Notes* 17.4, pp. 40–52. ISSN: 0163-5948. DOI: 10.1145/141874.141884.

Tyree, J. and Akerman, A. (2005). "Architecture decisions: demystifying architecture". In: *IEEE Software* 22.2, pp. 19–27. DOI: 10.1109/MS.2005.27.

Zdun, U., Capilla, R., Tran, H., and Zimmermann, O. (2013). "Sustainable Architectural Design Decisions". In: *IEEE Software* 30, pp. 46–53. DOI: 10.1109/MS.2013.97.