

A model to evaluate front-end frameworks for single page applications written in JavaScript

En modell för att utvärdera front-end ramverk för single page applications skrivna i JavaScript

Sara Abrahamsson

Supervisor : Olaf Hartig
Examiner : Olaf Hartig

External supervisor : Erik Cederlöf

Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>.

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>.

Abstract

Despite a constantly growing selection of front-end JavaScript frameworks, there is a lack of research to guide the choice of which one to use in a software project. Instead, the decision is generally based on experience and personal preferences within the team. The aim of this thesis is therefore to present a structured evaluation model to provide for more informed decisions. A preliminary study is carried out where the most important qualities of a framework are identified, both according to previous literature and to practitioners. The pre-study result is used to construct a structured model to assess framework performance for the identified qualities. Finally, a test of the model is carried out to see if it can guide the choice of framework in a specific project. The study shows that the design of the model does contribute with important insights on framework performance in prioritized quality areas and the trade-offs that this entails for other important qualities. Thus, the model provides necessary information to make well-founded decisions. Furthermore, it fills the gap in contemporary research by providing an understanding of what is important in a framework according to practitioners.

Acknowledgments

I would like to thank everyone who made this master's thesis project possible. Firstly, I want to thank Tietoevry Care for the opportunity to carry out this project. Thanks to all the employees who contributed with important and meaningful insights during the interviews. I would like to extend an extra big thank you to my external supervisor, Erik Cederlöf, for fruitful discussions and tips along the way, both technical and non-technical. Of course, I also want to thank my supervisor and examiner from Linköping University, Olaf Hartig, for all the valuable guidance during this thesis project. Finally, I would like to thank Oskar Hallström for valuable feedback and discussions during the opposition.

- Sara Abrahamsson, Linköping 2023

Contents

Abstract	iii
Acknowledgments	iv
Contents	v
List of Figures	vii
List of Tables	viii
1 Introduction	1
1.1 Motivation	1
1.2 Aim	2
1.3 Research questions	2
1.4 Delimitations	2
2 Background	3
2.1 Single-page applications	3
2.2 JavaScript frameworks	3
2.3 Tietoevry Care	4
2.4 Cambio Open Services API	4
3 Related Work	5
3.1 Evaluation of JavaScript front-end frameworks	5
3.2 Data collection in software engineering research	7
3.3 Designing an evaluation model	10
4 Method	11
4.1 Pre-study	11
4.2 Definition of evaluation model	14
4.3 Evaluation	15
5 Results	16
5.1 Pre-study	16
5.2 Definition of evaluation model	18
5.3 Evaluation	21
6 Discussion	27
6.1 Results	27
6.2 Method	29
6.3 The work in a wider context	31
7 Conclusion	32
7.1 Future work	33

Bibliography	34
A Interview Structure	36
A.1 Introduktion och presentation av studien	36
A.2 Inledande frågor	36
A.3 IntervjuseSSION	37
B Survey Structure	38
B.1 Information text	38
B.2 Survey	39
C Evaluation model	41
C.1 Scoring of quality attributes	41
C.2 Calculation of total quality scores	45
C.3 Presentation of results	45

List of Figures

2.1	Example of a single-page application's life-cycle	4
3.1	Criteria for comparative analysis of frameworks according to Gizas et al.	6
3.2	Evaluation criteria for frameworks according to Graziotin and Abrahamsson	6
3.3	Step-by-step process for thematic analysis presented by Braun and Clarke	9
5.1	Distribution of survey respondents' experience in front-end web development. . .	19
5.2	Summary of important qualities from literature search, interviews and survey. . .	20
5.3	Chosen qualities for evaluation model.	20
5.4	Start view in prototype for applications.	21
5.5	Medication view in prototype for applications.	25
5.6	The compiled scoring result presented in a spider chart.	26
C.1	Evaluation model qualities with related quality attributes and weights.	45

List of Tables

3.1	Areas and features presented by Pano et al.	7
3.2	Important factors according to Ferreira et al.	7
5.1	Important qualities in a JavaScript framework derived from the literature study . .	17
5.2	Ranking of qualities derived from the literature study according to the interviews .	17
5.3	Important qualities in a JavaScript framework derived from the interviews	18
5.4	Most important qualities in a JavaScript framework according to the survey	19
5.5	Scoring of quality attributes for Vue.js.	22
5.6	Scoring of quality attributes for React.js.	23
5.7	Scoring of quality attributes for Angular	24



1 Introduction

This chapter provides an introduction to the thesis project, starting with a motivation in Section 1.1. Thereafter, the aim is presented in Section 1.2 followed by the research questions in Section 1.3. Lastly, in Section 1.4, the delimitations are considered.

1.1 Motivation

Web applications are constantly increasing in popularity and are thus gradually replacing the old desktop applications. There are two main design patterns to choose from when developing web applications. Firstly, the traditional multipage application, running most of the logic on the server [1]. Secondly, the newer single-page application pattern, where most user interface logic instead executes in a web browser [1]. The use of the latter is continuously increasing thanks to the possibility of creating a richer user interface without interruptions and waiting time due to page reloads [10]. Instead, the content is loaded into the single web page using JavaScript.

Statistics show that 98.2% of today's websites use JavaScript to some extent [17]. This, combined with the increased popularity of single-page applications, has resulted in a large and continuously growing selection of frameworks for developing this type of application. Currently, the three most popular frameworks for single-page applications written in JavaScript are: React, Angular and Vue [15]. As a consequence of options increasing at a high rate, developers are repeatedly faced with the choice of which framework is most suited for a particular software project. The decision is generally based on experience within the team, without considering other parameters of potential importance for the outcome of the project.

Despite the number of frameworks increasing at a high rate, there is a lack of knowledge and research to guide decision makers when choosing between them [13, 6]. Therefore, it is interesting to develop a method for evaluating the fit of a framework for a specific project to be used as a basis for such decision-making.

This master's thesis is done in collaboration with the IT-consultant firm Tietoevry, at their department, Care. As of today, the selection of a frontend framework for projects is described as arbitrary: mostly based on experience or, in some cases, curiosity to test a new framework.

Overall, this suggests a need for a more thorough understanding of important parameters to consider when evaluating a framework, and a more structured approach for such an evaluation.

1.2 Aim

The aim of the thesis project is to obtain a deeper understanding of what is of importance when evaluating a framework for a software project. Thereafter, based on that, develop a structured model that can guide decision makers when approaching the question of which framework to use in a project.

1.3 Research questions

The aim of the thesis project is divided into three research questions. The first two questions intend to create a thorough understanding of important considerations when evaluating a framework. The third aims to develop a method to guide the decision of choosing a framework for a project, using the results of question one and two as a basis.

- **RQ1:** What is, according to contemporary research projects, important to consider when deciding which web front-end framework to adopt in a project?
- **RQ2:** What is, according to professional web front-end developers, important to consider when deciding which framework to adopt in a project?
- **RQ3:** How can a structured method to evaluate a front-end framework and how well it fits a project look like based on the parameters identified in RQ1 and RQ2?

1.4 Delimitations

The thesis project implementation focuses only on three frameworks for applications written in JavaScript: React, Angular and Vue. Partly due to time constraints: a very large time frame is needed to include all available frameworks in the implementation. Partly because the thesis is done in collaboration with the Care department at Tietoevry and those three frameworks are the ones previously used in their projects.

Furthermore, as the project thesis is done in collaboration with Tietoevry, the results from interviews will be limited in the sense that all interviewees work for the same company. They can, however, be considered a representative for the main users of the technologies to be evaluated, as they all work with front-end web development. In addition to the interviews, a natural survey is performed with web front-end developers from outside the company as well. The answers from the survey are then used to supplement and compare the answers obtained in interviews within the case company. Because of this, the project altogether covers the main intended users of the evaluated technologies.

Lastly, the project focuses on front-end frameworks for single-page applications. Therefore, the developed method will be specially adapted for evaluation of such frameworks.



2 Background

This chapter provides the reader with relevant background information for the thesis project. Section 2.1 describes the single page application design pattern in more detail. Thereafter, Section 2.2 explains the usage and advantages of JavaScript frameworks. Lastly, Section 2.3 and 2.4 describes the case company with which the project is carried out as well as the API that will be used to test HTTP requests in the evaluation.

2.1 Single-page applications

As previously mentioned, the single-page application, or SPA, design pattern is increasing in popularity. In consistency with that, the applications developed in this project will follow that design pattern. In traditional multipage applications, a new HTML page is loaded from the server every time the content on the web page is to change. With SPA's, new content is loaded into the single HTML page dynamically using JavaScript. As a result, the interface in single-page applications is perceived as more responsive and faster than traditional ones [10]. As users interact with the SPA, their actions can cause updates in the application and/or send an HTTP-request to an API. Figure 2.1 illustrates an example of a single-page application's life-cycle. An initial request is sent to the server and an HTML page is loaded as a response. Then, actions from the user cause the content of the page to update by re-rendering the DOM using JavaScript.

2.2 JavaScript frameworks

Use of Client-side JavaScript frameworks is becoming an increasingly large part of front-end web development. By using frameworks, developers are provided with tested and well-developed tools and libraries that bring good conditions for developing scalable and responsive web applications. Moreover, Borissova et al. [3] explains how using frameworks reduces the amount of code developers need to write. This, in turn, leads to reduced time spent debugging code [3]. The evaluation model developed in this master's thesis is tested by evaluating the three most popular frameworks: React, Angular and Vue.

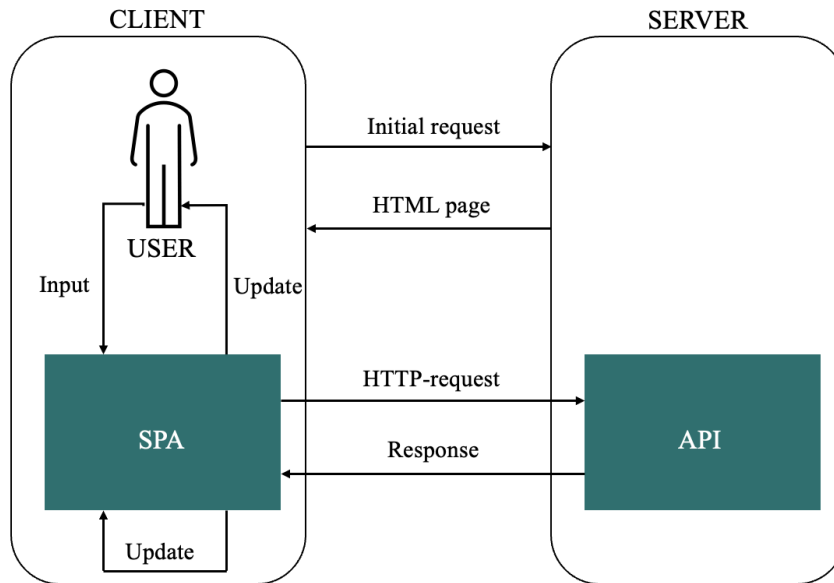


Figure 2.1: Example of a single-page application's life-cycle

2.3 Tietoevry Care

The department Tietoevry Care deliver IT-solutions for the health care sector. The rapid development within technology combined with the increasing demands on quality and coordination results in a great pressure on actors in the sector. Tietoevry contributes with knowledge and expertise in the interface between IT and healthcare.¹

In essence, the department describes themselves as .NET developers. However, in terms of front-end frameworks, there is no established procedure to decide what framework to adapt for a new project. As of now, the decision is often based on previous expertise in the team or usage of frameworks in previously developed and similar applications. Occasionally, a new framework is tested due to interest from developers in gaining experience and knowledge in it. Thus, there is a need for a deeper understanding of important considerations when deciding the framework to adopt for a project to make a correct assessment.

2.4 Cambio Open Services API

As mentioned in Section 2.1, calls to APIs are common in SPAs. It is therefore relevant to include HTTP requests in the smaller applications that are developed to apply the evaluation model to the frameworks. The Cambio Open Services APIs² intend to facilitate for innovation and integration within the healthcare sector. The open architecture of their APIs opens up for collaboration and sharing of clinical data. An absolute majority of the Swedish regions are expected to adopt the APIs. This makes it interesting for Tietoevry Care to study them and how they work, as they will implement them in future solutions. Therefore, their open APIs will be used in this master's thesis for HTTP calls to an API. This will fulfill the needs from a research perspective, by testing an important part of the frameworks. At the same time, it can hopefully lead to insights about the API that will be of interest for the case company.

¹<https://www.tietoevry.com/en/care/healthcare/ehealth-consulting/>

²<https://www.cambio.se/vi-erbjuder/open-services/>



3 Related Work

This chapter provides a summary of related work previously conducted by other authors. Section 3.1 presents related work on what to be considered when evaluating front-end JavaScript frameworks. This is followed by Section 3.2 in which the choice of data collection method for this type of research is discussed, based on previous literature. Lastly, in Section 3.3, the methodology for developing an evaluation model for frameworks is discussed.

3.1 Evaluation of JavaScript front-end frameworks

Common in contemporary research is that the extent of knowledge on how frameworks should be evaluated does not correspond to the rate at which the range of alternatives is growing [6, 13, 8].

Gizas et al. [7] highlight the importance of applying the correct metrics to make a good assessment on what framework to adapt in a project. In their article, Gizas et al. conducted a comparative study of the six most popular frameworks at that time (2012). The comparison was based on software metrics identified as important in previous research. Further, the investigation was divided into three main areas considered important for evaluating frameworks: quality, validation and performance. What was evaluated more specifically in each area is presented in Figure 3.1. Gizas et al. [7] evaluate these criteria with traditional software metrics. For example, the size is evaluated with metrics such as lines of code and ratio between comment lines and code lines.

Graziotin and Abrahamsson [8] extended the criteria presented by Gizas et al. [7]. Their belief was that the research needs that are evaluated based on the previously presented criteria must be supplemented with the needs of the real practitioners: the developers. According to Graziotin and Abrahamsson, the considerations when evaluating a framework depend on the context in which it will be used. Because of this, the authors mean that only considering traditional software metrics, representing the research needs, when comparing frameworks is insufficient.

Through discussions with four different front-end web developers, the authors present three additional areas of importance when selecting a framework: documentation, community

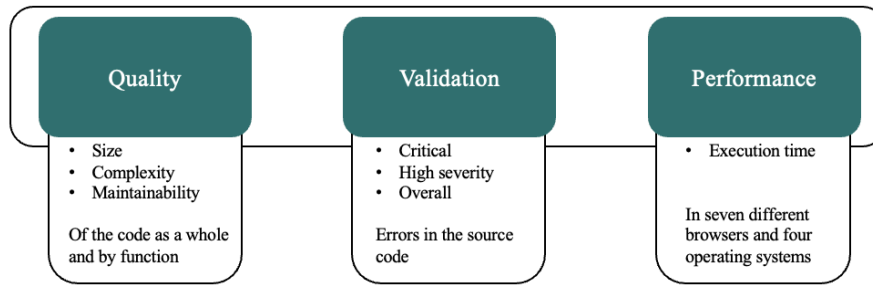


Figure 3.1: Criteria for comparative analysis of frameworks according to Gizas et al. [7]

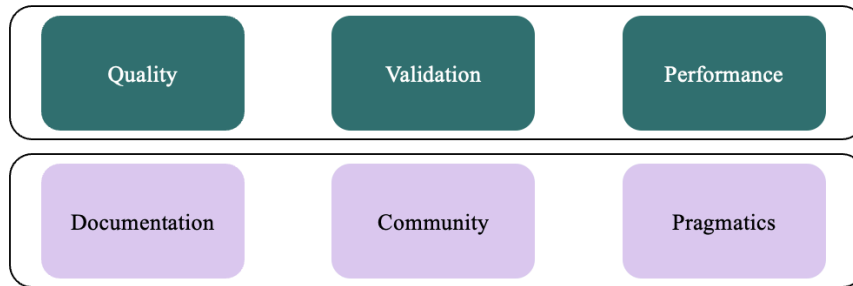


Figure 3.2: Evaluation criteria for frameworks according to Graziotin and Abrahamsson [8]

and pragmatics. The discussions mainly concerned important criteria in decisions on what framework to use, and how the ones currently used by the developers were chosen. Because of this, the authors further state that by adding these areas to the evaluation model, a new layer, covering the needs of the practitioners, is added. In the extended framework, documentation is defined as the sufficiency of the documentation [8]. Community as the extent of activity in the community, for example asked and answered questions on Stack Overflow or the frequency of updates and therewith the freshness of the framework. Lastly, the pragmatics is described as how the framework, and its architecture contributes to the “code less, do more” factor. The resulting framework, after adding the new layer presented by Graziotin and Abrahamsson [8] is illustrated in Figure 3.2, where the added areas are represented in pink.

Pano et al. [13] further develop the thesis that practitioners’ need to be included in a model for evaluating frameworks. The authors identify four types of actors involved in the selection of a framework for a project: customers, developers, the team, and the team leader. Moreover, in the article, Pano et al. present a model of important features in a framework. The model was developed using results from semi-structured interviews with 18 different decision makers. The interview questions first touched on the interviewee’s background in web development, JavaScript and with frameworks. This was followed by some questions about the framework the person was using at the time of the interview and how the choice to do so was made. Finally, a number of questions were asked about the person’s experience of that framework. In Table 3.1, the areas of the model are presented, as well as the features included in each of the areas.

A more recent work was published in 2021 by Ferreira et al. [6]. When Gizas et al. [7], Graziotin and Abrahamsson [8] and Pano et al. [13] published their articles, the frameworks that are the most popular today did not yet exist. Ferreira et al. [6] intended to examine impor-

Table 3.1: Areas and features presented by Pano et al. [13]

Area	Features
Performance expectancy	Performance Size
Effort expectancy	Automatization Learnability Complexity Understandability
Social influence	Competitor analysis Collegial advice Community size Community responsiveness
Facilitating conditions	Suitability Updates Modularity Isolation Extensibility
Price value	Cost

Table 3.2: Important factors according to Ferreira et al. [6]

Factor	Explanation	Count
Popularity	That the framework is widely used and known	19
Learnability	That the framework is easy to learn and use	17
Architecture	That the framework forces a good architecture in the client	15
Expertise	That previous experience in the framework exists	10
Community	That the framework is maintained by a large community	9
Performance	That the framework's performance is good	8
Gain experience	That increased experience with the framework is desired	6
Documentation	That the framework's documentation is adequate	5
Sponsorship	That the framework has support from respected companies	4

tant factors in the choice of framework in a more modern context to increase the relevance of research in the field. The authors present a set of factors collected from survey answers by 49 developers. The survey consisted of only two open questions. Firstly, the respondent was asked to describe why they chose to work with the framework they were using at the time of the interview. Secondly, they were asked if they had any intentions to migrate to another framework. The results show that the key factors developers consider when deciding what framework to use were popularity and learnability [6]. The result in its entirety is summarized in Table 3.2. In the table, each factor is presented together with an explanation and how many times it was mentioned in the survey.

3.2 Data collection in software engineering research

Software engineering is performed by real people, in real environments. Because of this, research in software engineering requires studies of the actual practitioners performing their work [12]. Lethbridge et al. [12] state the difficulties as well as the importance of choosing the data collection method most suitable for the research questions which the data is intended to help answer. The authors present possible data collection methods in software engineering: in what situations they are appropriate to use, advantages and disadvantages of each method.

Lethbridge et al. [12] divide the methods into three groups: first-, second-, and third order techniques. According to the authors' division, first order techniques contain direct

involvement from the studied engineers. Second order techniques instead contain indirect involvement from the engineers, for example studying them as they work without any direct interactions. Lastly, third order techniques are studies of the work artifacts of the software engineers. First order techniques are the only option for collection of data to answer research questions related to satisfaction with using a specific tool [12]. This implies that these data collection techniques are most suitable to answer the research questions of this thesis project.

Interviews and surveys are both classified as first order techniques, as they directly involve interaction with the software engineers. Both techniques are, according to Lethbridge et al. [12], suitable to obtain general information and opinions on products and processes. When using interviews and surveys as data collection methods, it is, according to the authors, critical that the representatives included in the research are appropriate to represent the intended population. This is, however, much more important if the collected data is intended to be the basis of statistical calculations and assumptions rather than identifying trends [12]. Software engineers generally enjoy answering questions about their work. However, it is important to consider possible biases in the results: the answers are often personal opinions from the respondents.

3.2.1 Interviews

Lethbridge et al. [12] explain that during interviews, the researcher is able to clarify possible misunderstandings in the questions and the interactivity is much higher than in a questionnaire. According to the authors, this generally results in a higher level of quality in obtained answers. Runesson and Höst [14] present three types of interviews with different level of structure to them: *unstructured*-, *semi-structured* and *fully structured interviews*. According to the authors, *semi-structured interviews* are common in case-studies. There is a prepared plan for the questions to be asked, which prevents the risk of missing important information [14]. However, the questions do not need to be asked in a fixed order, which allows for improvisation and a better discussion between the involved parties [14]. Altogether, this suggests that to get the greatest benefits in increased interactivity from using interviews as a data collection method, semi-structured interviews should be used. Runesson and Höst [14] further suggest three important phases that the interview setup can follow:

1. Presentation: Study- and interview objectives and how the interview data will be used.
2. Introductory questions: Easy questions on the background of the interviewee.
3. Interview session: Includes asking the actual interview questions.

Runesson and Höst [14] also recommend recording and transcription of interviews. The authors mean that when transcribing the interviews, the researcher often reaches new insights that contribute with important aspects to the analysis of the results. The number of study objects shall, according to the authors, be decided during the study with the breaking condition that no more interviews are needed if no additional insights are added from interviewing new objects. Clarke et al. [5] propose a range of 6–15 interviews for a medium size project such as a master's thesis.

3.2.2 Surveys

Surveys are a popular method for data collection in research [12]. It does not require a large amount of time and resources, and it is easier to get a geographical spread of respondents using a survey than, for example, interviews [12]. However, in order to produce valid results, it is critical that the survey is designed and performed correctly.

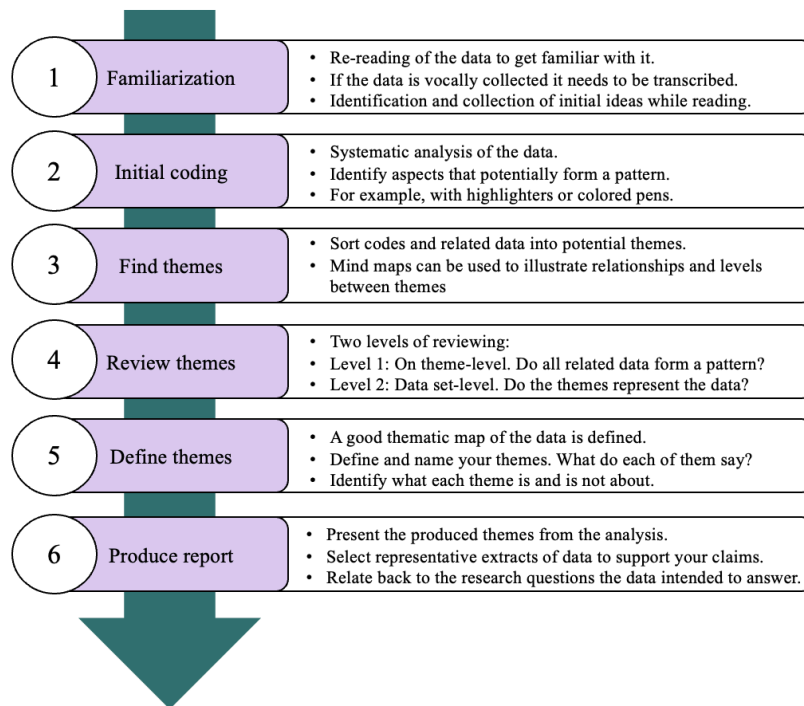


Figure 3.3: Step-by-step process for thematic analysis presented by Braun and Clarke [4]

Kitchenham and Pfleeger [11] highlight this issue and present, in a series of six articles, guidelines for how a proper survey should be designed, implemented and interpreted. The third article in the series concerns the construction of questionnaires [11]. Before constructing the survey, the authors suggest a literature search should be performed to learn from and improve how previous researchers collected their data. Furthermore, Kitchenham and Pfleeger [11] present a number of important considerations when choosing the questions for the survey. The number of questions must not be too large, and the included questions must be formulated so that they are concrete, unambiguous and purposeful [11]. The question type can either be open (free-text) or closed (categorical alternatives). Here, Kitchenham and Pfleeger [11] highlight the importance of carefully considering the appropriate type of the questions included in the survey. According to the authors, open questions imply a risk of misunderstandings, which could result in difficulties to analyze and code the results. Using closed questions with categorical alternatives often makes the results easier to interpret and analyze, according to Kitchenham and Pfleeger [11]. However, the authors highlight the importance of including neutral options for closed questions such as "No preference" or "I do not know". The authors also recommend that the questions are evaluated and improved before the survey is conducted. According to Clarke et al. [5], a range of 30–100 answers are appropriate for a survey in a medium size project such as a master's thesis.

3.2.3 Thematic analysis

Braun and Clarke [4] define a step-by-step process for thematic analysis. This is a structured approach used to identify and analyze themes in data, for example from interviews and surveys. The process presented by the authors follows six steps. In Figure 3.3, the process and its steps are visualized and explained further.

3.3 Designing an evaluation model

As contemporary work on evaluation of front-end frameworks is deficient, it may be interesting to start in a related area where research on evaluation and comparison models is more common: software architecture. Bergner et al. [2] present the model DoSAM (Domain-Specific Software Architecture Comparison Model) which allows for comparison of different architectures within the same domain. The model proposed by Bergner et al. [2] follows four steps:

1. Define and extract common views on the candidate architecture to answer the question: *What is evaluated?*
2. Identify relevant so-called quality attributes, and how these are measured, to answer the question: *How is it evaluated?*
3. Define the so-called quality computation weights for the attributes to answer the question: *What is important?*
4. Evaluating and scoring the architectures by applying the framework developed in the previous steps.

Ignacio et al. [9] presents a comparison model for agile web frameworks. The development of the model followed the phases of Bergner et al.'s [2] DoSAM model. Common views on agile web frameworks were defined, and from these a set of qualities considered important in the evaluation [9]. For each of the qualities, a set of guiding questions, so-called quality attributes, were defined and assigned a weight to represent how important the issue is to the assessment of the related quality. When evaluating the frameworks using the model, Ignacio et al. [9] rated each attribute according to how well it aligned with the evaluated framework by assigning a percentage from 0-100. These percentages were then combined with the weight of the attribute and summed up with the other quality attributes to result in a score for the entire quality [9]. The result for all qualities was then presented in a spider chart, from which it was possible to see how the frameworks scored compared to the others on all qualities [9]. Such a visualization of the result can be advantageously used for the model developed in this thesis project as well. It can guide the decision maker in various trade-offs to choose the framework that best suits the specific project needs.

Similar to the work by Ignacio et al. [9], this thesis project aims to develop a model for comparing and evaluating the usage of frameworks. The development of this evaluation model will therefore also be inspired by the DoSAM model presented by Bergner et al. [2]. However, modifications will be made to adapt the developed model to the purpose and intended application of it.



4 Method

Wohlin and Runeson [18] present three methodologies suitable for research in software engineering carried out in close collaboration with industry: Design Science Methodology (DSM), Action Research (AR) and Technology Transfer Research Model (TTRM). The authors then compare the three methodologies and present guidelines for which one to use for certain types of research projects. According to the results from Wohlin and Runeson's [18] study, DSM is a good choice for projects in which tangible artifacts are developed with the intention for it to solve an identified problem and gain knowledge from it. AR is instead suitable for projects intended to support change in the organization and TTRM for transferring research results to the industry. Thus, the method in this thesis project followed the three main phases of DSM [18]:

1. Identify and describe the challenge and current situation
2. Develop a solution by studying alternatives
3. Evaluate the solution

In this thesis project, the first phase is called *Pre-study*. This phase included two main steps: a literature study of related and contemporary work and data collection from the industry to gain further understanding in the studied challenge and the existing solutions. The second phase is called *Definition of evaluation model*. Now, the results from the previous phase were compiled into an evaluation model. Lastly, the third phase is called *Evaluation*. In this phase, the proposed model was tested on the three most popular frameworks to evaluate whether it can support decision makers in the choice of which framework to use in a project.

4.1 Pre-study

The pre-study was the first phase of the project. It consisted of two main activities: a literature study and a data collection process. The first part intended to create an understanding of what was previously identified as important qualities in a front-end framework. The result then formed a basis for the questions in the interviews and survey conducted in the second part, the data collection process.

4.1.1 Literature study

The main purpose of the study was to identify the most important qualities in a framework, according to previous literature. The intention was that the results would be used as a basis partly for the interview and survey questions and partly for the actual evaluation model. In addition, two other important aspects of the project were explored: correct methodology for interviews and surveys and possible approaches to designing an evaluation model.

To find previous research relevant to this thesis project, a few important keywords were identified and used in different combinations to search for papers and publications in Unisearch¹, the database for Linköping University Library. Examples of such keywords are "JavaScript", "front-end", "framework", "evaluation", "compare" and "model" among others. Furthermore, the search was deepened as more relevant articles could be identified in the reference lists to already used articles in several stages. This way, a number of relevant articles could be collected, which together contributed to a good basis for the continued work with the thesis. Important information and data from these articles were summarized and presented in the report. An analysis was then carried out where related data were categorized into a set of qualities that were considered important in a framework.

4.1.2 Data collection

After gaining an understanding of existing literature on the topic, the next step in the pre-study was to collect data from which new, or confirmatory to already existing, insights could be generated. As presented in Section 3.2, the most appropriate data collection approach for the research questions in this thesis is, according to Lethbridge et al. [12], first order techniques. Therefore, interviews with front-end web developers at the case company were held. Also, the interviews were complemented with a natural survey aimed at front-end web developers not working at the case company. The purpose was to broaden the response group, partly to get a fairer representation of the intended target group, and partly to increase the number of responses to enable for more substantiated conclusions.

Interviews

To create conditions for a good dialogue and opportunities for some improvisation, what Runesson and Höst [14] calls a semi-structured approach was used for the interviews. The interviews followed the three steps presented in Section 3.2.1 and defined by Runesson and Höst [14]. They began with an introduction to the thesis project and how the data from the interviews were supposed to contribute to the project's results. The interviewees then had to answer a few simple questions about themselves and their background. Lastly, this was followed by the interview session, where the actual questions related to the research questions of the study were asked. As in the studies conducted by Pano et al. [13] and Ferreira et al. [6], the questions were formulated to ask what the interviewee values in a framework rather than on actual evaluation factors. The interview session ended with the interviewee having to choose three of the qualities identified in the literature that they considered most important in a framework. The questions were developed in two iterations, in between which feedback from both the internal and external supervisor was given. The resulting interview structure and questions are presented in Appendix A.

Before the interview, all study objects were asked for approval to record the conversation. This way, the recordings could then be transcribed to facilitate the upcoming analysis of the results. In accordance with the recommendations from Runesson and Höst [14], the number of study objects were decided during the study: when the interviews no longer generated

¹<https://liu.se/en/library>

new insights, no additional ones were held. This resulted in a total of 11 interviews, which falls in the range recommended by Clarke et al. [5].

Thematic analysis

To identify themes in the data collected from the interviews, thematic analysis was applied according to the process defined by Braun and Clarke [4] and presented in Section 3.2.3. First, all interviews were transcribed to get to know the data and facilitate the upcoming analysis. Then, the transcribed data was reviewed systematically: related data extracts that potentially formed a pattern were highlighted in the same color, resulting in an initial coding. The codes were then sorted into potential themes which were subsequently reviewed, defined and named. The resulting themes were then used, together with the qualities identified in the literature study, as response options in the subsequent survey study.

Survey

As a complement to the interviews with developers at Tietoenvry, a survey was performed. It was conducted using Google Forms ². With the intention of reaching as wide an audience as possible and to get many answers, the survey was distributed through several different channels. It was posted on Tietoenvry's intranet to reach colleagues from other cities and countries. Furthermore, it was sent directly to a few additional companies with front-end web developers or to front-end web developers directly. Moreover, the survey was also posted in the open source community through a meetup group targeted at JavaScript developers ³. Finally, it was also posted on the project supervisor's Twitter account ⁴, which is focused on data content. Therefore, it was likely that people in the intended target group could be reached through it. Together, it resulted in 63 responses in total. However, 17 of the respondents had indicated more than four options on the last question and had to be removed before analysis. Thus, the number of responses that could be used in the analysis was 46, which falls within the range recommended by Clarke et al. [5].

According to Kitchenham and Pfleeger [11], it is important not to include too many questions in a survey. Since this survey was to be sent to developers who would voluntarily choose to answer it, it was kept short and with very few questions with the intention to get more responses. As described in Section 3.2.1, Lethbridge et al. [12] explain how interviews brings an opportunity to clear up possible misunderstandings. With a survey, there is no interactivity between the parties and thereby no chance to, for example, ask clarifying questions. Because of this, it was decided that the survey would be built up from closed questions. According to Kitchenham and Pfleeger [11], this implies a reduced risk of misunderstanding. In the survey, the respondents had to describe their level of experience in front-end web development and which frameworks they were experienced in. Then, they had to pick at most four qualities they considered important when evaluating a framework. The given options were derived from studying related work and from the qualities identified in the interviews held with developers at Tietoenvry. The respondents were also given a possibility to leave a neutral answer: *I do not know*. Also, it was possible to add a quality, if it was not included as an option. The questionnaire included a description of the thesis project and how the resulting data from the survey were to be used. The survey in its entirety is presented in Appendix B.

²<https://www.google.com/forms/about/>

³<https://www.meetup.com/uppsalajs/>

⁴<https://twitter.com/olafhartig>

4.2 Definition of evaluation model

When defining the evaluation model, the phases of the DoSAM model presented by Bergner et al. [2] were followed. Also, inspiration was taken from the evaluation model for agile web frameworks developed by Ignacio et al. [9]. Firstly, common views on what is important in a JavaScript front-end framework were identified. Thereafter, the most important qualities were chosen as evaluation criteria in the model. Lastly, a set of quality attributes were defined and weighted for each of the chosen evaluation qualities.

4.2.1 Identification of common views

The first phase was to identify common views on important qualities in a front-end JavaScript framework. This was done during the pre-study phase of the project: in the literature study (Section 4.1.1) and the data collection process (Section 4.1.2). During these activities, data were collected on important qualities of front-end frameworks, partly from contemporary, related research and partly from practitioners.

4.2.2 Choice of evaluation qualities

In the second phase, the qualities to be included in the model were chosen. The choice was based on how important the qualities seemed to be according to the collected data. A summation of the number of times each quality was mentioned as important in literature, interviews or the survey was made. Based on it, it could be determined which qualities were more important than others, and these were thus included in the model.

4.2.3 Definition of quality attributes

Once the most important qualities were decided, the third phase was to define the quality attributes that could guide the evaluation of each quality. The decision on the attributes that would define each quality was primarily based on explanations and definitions from the interviews, but also on the author's own knowledge in the area with guidance from the external supervisor. First, it was decided for each quality whether it should be measured quantitatively or qualitatively. The goal was that as many of the qualities as possible would be assessed based on quantitative measures. For each quality, a number of attributes were then determined, and each attribute was assigned a weight to represent how important its assessment was to the final score of the related quality. The weighing was mostly based on the number of times the attribute was mentioned in relation to the quality during the interviews. However, for some attributes such data was not available, and the weighing was instead based on the author's own knowledge. The result is a model developed according to Bergner et al.'s [2] DoSAM model and inspired by the model presented by Ignacio et al. [9] For the qualitative attributes, a definition was given for each one and the assessor had to set a score between 1-4 representing how well the statement fit the assessed framework. The following scale were defined to help the assessor score each qualitative attribute:

1. Does not apply
2. Partly apply
3. Largely apply
4. Fully apply

For the attributes of the quantitative qualities, the scoring scale could instead be based on the numerical value that the framework received for the attributes in relation to some predetermined limit values. A preliminary investigation was carried out on a number of frameworks

and how they ranked on the different quantitative attributes. The result was then used to help set reasonable limit values for the scoring scale. Below is an example of what the scale for the quantitative attributes could look like when a greater value scores higher. X, Y and Z represent the limit values, helping to guide the assessor when scoring the attribute.

1. Less than X
2. X - Y
3. Y - Z
4. More than Z

4.3 Evaluation

The evaluation, or actual application, of the model corresponds to the fourth phase of Bergner et al.'s [2] DoSAM model. Now, the developed model was tested by being applied to the three most popular JavaScript front-end frameworks at the time of the study. The model was applied to one framework at a time, and each evaluation consisted of two parts: development of a smaller application and application of the model.

4.3.1 Development of application

A smaller application was developed in all frameworks, as there was no or limited previous experience with the frameworks and their documentations. To ensure that all three frameworks were evaluated similarly and on the same visual basis, a static prototype was produced before development began. Since the prototype was created only to demonstrate visual parts of the application, it did not need to be interactive. Therefore, the simple prototype was created as two slides in a presentation in the design tool Canva⁵. The three applications were then developed with the aim of resembling the prototype as much as possible. During development, notes were taken regarding the attributes that would later be evaluated in the model. That is, how easy it was to get started and understand the framework, the quality and degree of coverage of the documentation, and the degree to which the framework contributed to the structure of the application.

4.3.2 Applying the model

After the development phase was completed, the frameworks were evaluated according to the model and all attributes were assigned a score according to the given instructions and scoring scales. Using the attributes' scores, a total quality score could then be calculated for the seven qualities. The results for the three frameworks were then presented in a spider chart, similar to Ignacio et al.'s [9] study. The chart was then studied to draw conclusions about whether the model could be helpful in a decision on which framework to use in a project.

⁵<https://www.canva.com/>



5 Results

This chapter presents the results obtained during the thesis project. Firstly, the results from the pre-study are presented. Thereafter, the evaluation model developed from the pre-study results is presented. Lastly, the results from testing the evaluation model on the three most popular JavaScript front-end frameworks are presented.

5.1 Pre-study

During the pre-study phase, data has been collected through two different activities: the literature study and the data collection process. Section 5.1.1 presents the results from the first and Section 5.1.2 the latter. These activities correspond to the first phase of the DoSAM model presented by Bergner et al. [2]: identification of common views.

5.1.1 Literature study

In the literature study, four different articles with studies of important parameters for evaluation of JavaScript frontend frameworks were presented: Gizas et al. [7], Graziotin and Abrahamsson [8], Pano et al. [13], and Ferreira et al. [6]. In Section 3.1, the results from these studies are presented. The results were analyzed and categorized into a set of qualities considered relevant for this study. In Table 5.1 the resulting qualities are presented as well as a count representing how many of the four studied articles that considered the quality important.

5.1.2 Data collection

This section presents the result from the data collection phase of the pre-study. Firstly, Section 5.1.2 presents important qualities in a JavaScript front-end framework identified in the interviews. Secondly, in Section 5.1.2 the result from the survey is presented.

Interviews

During the interviews held at the case company, two types of important data were collected. Firstly, important qualities in a framework from the questions of discursive nature. Secondly, ranking information when the interviewee had to prioritize the three most important of

Table 5.1: Important qualities in a JavaScript framework derived from the literature study

Quality	Explanation	Count
Ease of use	Easy to learn. Writing and maintaining code is easy.	4
Performance	Applications built using the framework have good performance.	4
Size of code	Contributes to a small amount of code.	4
Updated	Continuously maintained and updated.	4
Good architecture	Contributes to good architecture and to the "code-less-do-more" factor.	3
Community	Popular and with a large and active community.	3
Documentation	The quality and availability of documentation is high.	2
Previous experience	You and/or your colleagues have experience from the framework.	2
Suitability	Is suitable for the project, all necessary libraries etc. are available.	1

Table 5.2: Ranking of qualities derived from the literature study according to the interviews

Quality	Explanation	Count
Ease of use	Easy to learn. Writing and maintaining code is easy.	6
Good architecture	Contributes to good architecture and to the "code-less-do-more" factor.	6
Updated	Continuously maintained and updated.	5
Community	Popular and with a large and active community.	5
Suitability	Is suitable for the project, all necessary libraries etc. are available.	4
Documentation	The quality and availability of documentation is high.	3
Previous experience	You and/or your colleagues have experience from the framework.	3
Performance	Applications built using the framework have good performance.	1
Size of code	Contributes to a small amount of code.	0

the qualities identified in the literature study. Table 5.2 presents the results from the latter, the ranking information from when the interviewees had to pick the three most important qualities derived from the literature study. The count represents the number of interviewees (out of 11 in total) that included the quality as one of the three most important.

For the questions of discursive nature, a thematic analysis was performed to identify important themes, or qualities, mentioned in the interviews and relevant to the research questions. The resulting qualities are presented in Table 5.3. For each quality, a description is provided, as well as a count representing the number of interviewees (out of 11 in total) that mentioned the quality in their interview. Colored in green are the qualities that were not previously identified in the literature. The former quality "Good Architecture" from Table 5.1 is considered to coincide with the new "Structuring" and the former quality "Performance" is now included in the new "Lightweight". Thus, these two are not considered completely new from the interviews and thereby not colored in green.

Survey

In the survey, the respondents were asked to pick at most four of the proposed qualities that they considered most important in a JavaScript framework. Since all the qualities from the literature study also appeared during the interviews (or are considered to be part of a quality identified in them), it was the interview qualities that were presented as answer options in the questionnaire. Respondents also had the option of adding their own option, or answering neutrally and choosing the option *"I don't know"*. The survey collected a total of 63 responses. However, only 46 of these could be used in the analysis. In the remaining 17, the respondent had chosen more than four options when the (at most) four most important qualities were requested. In order to obtain a fair and accurate analysis, these responses were therefore excluded from the summarized data before the analysis.

Table 5.3: Important qualities in a JavaScript framework derived from the interviews

Quality	Explanation	Count
Ease of use	Learning and getting started is easy. Make code maintenance easier and coding more efficient.	8
Updated	Continuous maintenance, development and updates. Now and in the future.	8
Structuring	Lots of code conventions and set ways of doing things. Contributes to good and recognizable architecture.	8
Light weight	The initial output size is not too big, affecting performance and load times. Installation does not require downloading a large amount of packages.	8
Previous experience	Team members have knowledge and enjoy coding in it. Syntaxes match team member preferences.	6
Freedom	Few rules and conventions, allowing own solutions and choices. Possible to study the source code to deep dive in advanced functionality.	6
Community	Popular and widely used. Support is available from a stable, large and active community.	4
Maturity	Established and with a certain level of maturity. Stable updates, few breaking changes. Not too many bugs.	4
Size of code	Require less boilerplate code.	3
Documentation	The documentation is good, explanatory and sufficient.	3
Suitability	Adapted to the problem to be solved. Corresponds to needs, functional- and non-functional requirements.	3
Browser compatibility	Offer compatibility with all major browser vendors.	2
Security	Not too many third-party dependencies that pose an extended risk for supply chain attacks.	1

Figure 5.1 shows the distribution of answers from the 46 respondents when asked to assess their experience in front-end web development. The assessment was based on a scale from 1-5 where one represented *No experience* and five *Very experienced*. As presented, none of the respondents said to not have any experience in the area, and the majority of them assessed their experience as two or three out of five. In Table 5.4, the result from the question related to important qualities in a front-end JavaScript framework is presented. The count represents the number of survey respondents (out of 46 in total) that included the quality as one of the four most important. Colored in green are the qualities that were added by a respondent and not previously included as an answer option.

5.2 Definition of evaluation model

This section presents the results from the activities corresponding to the second- and third phase of the DoSAM model presented by Bergner et al. [2]: choice of evaluation qualities and definition of quality attributes.

5.2.1 Choice of evaluation qualities

The results from the literature study, interviews and survey were summarized to find the most important qualities in a JavaScript front-end framework. In total, a quality could have been mentioned 61 times (four scientific articles, 11 interviews and 46 survey answers). Figure 5.2 presents the total number of times a quality was mentioned and also how many of these that occurred in literature, interviews or survey answers. The qualities that were mentioned in more than 30% of all possible cases were considered most important and thus included in the evaluation model. The resulting criteria to be included and evaluated using the model is presented in Figure 5.3.

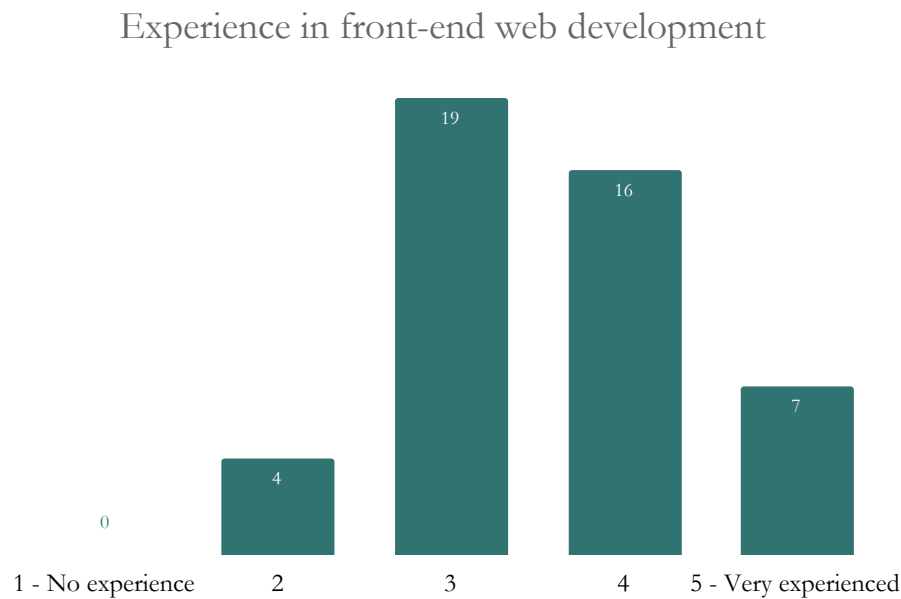


Figure 5.1: Distribution of survey respondents' experience in front-end web development.

Table 5.4: Most important qualities in a JavaScript framework according to the survey

Quality	Explanation	Count
Documentation	The documentation is good, explanatory and sufficient.	32
Ease of use	Learning and getting started is easy. Make code maintenance easier and coding more efficient.	20
Maturity	Established and with a certain level of maturity. Stable updates, few breaking changes. Not too many bugs.	18
Community	Popular and widely used. Support is available from a stable, large and active community.	17
Updated	Continuous maintenance, development and updates. Now and in the future.	15
Suitability	Adapted to the problem to be solved. Corresponds to needs, functional- and non-functional requirements.	13
Browser compatibility	Offer compatibility with all major browser vendors.	12
Structuring	Lots of code conventions and set ways of doing things. Contributes to good and recognizable architecture.	10
Previous experience	Team members have knowledge and enjoy coding in it. Syntaxes match team member preferences.	9
Light weight	The initial output size is not too big, affecting performance and load times. Installation does not require downloading a large amount of packages.	9
Size of code	Require less boilerplate code.	6
Freedom	Few rules and conventions, allowing own solutions and choices. Possible to study the source code to deep dive in advanced functionality.	5
Security	Not too many third-party dependencies that pose an extended risk for supply chain attacks.	5
Typing	Typescript	1
Not use them	The best quality of front-end JavaScript frameworks is when we don't use them.	1

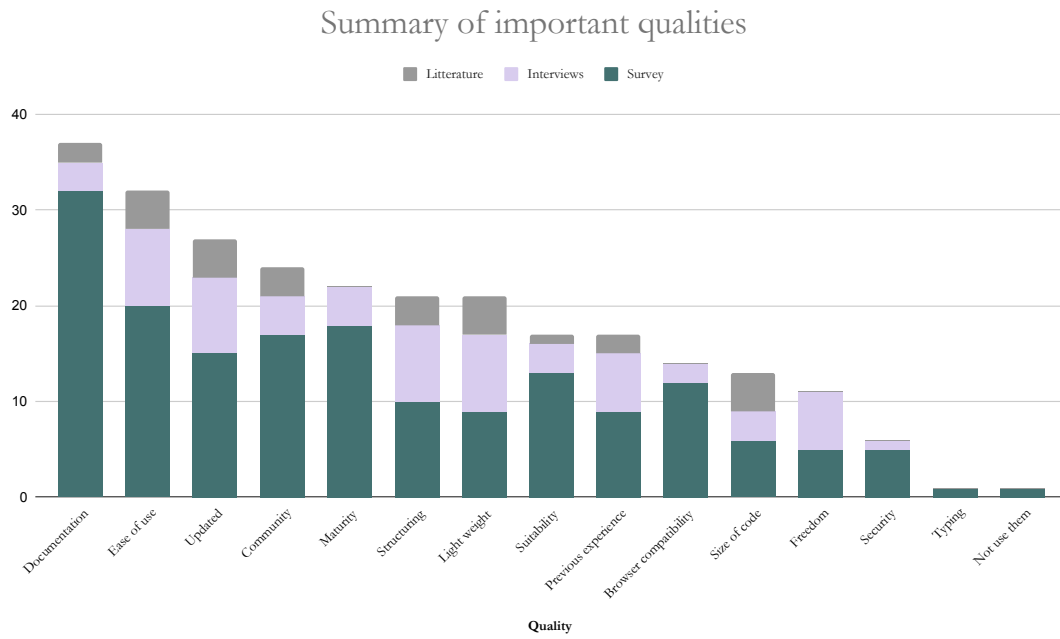


Figure 5.2: Summary of important qualities from literature search, interviews and survey.

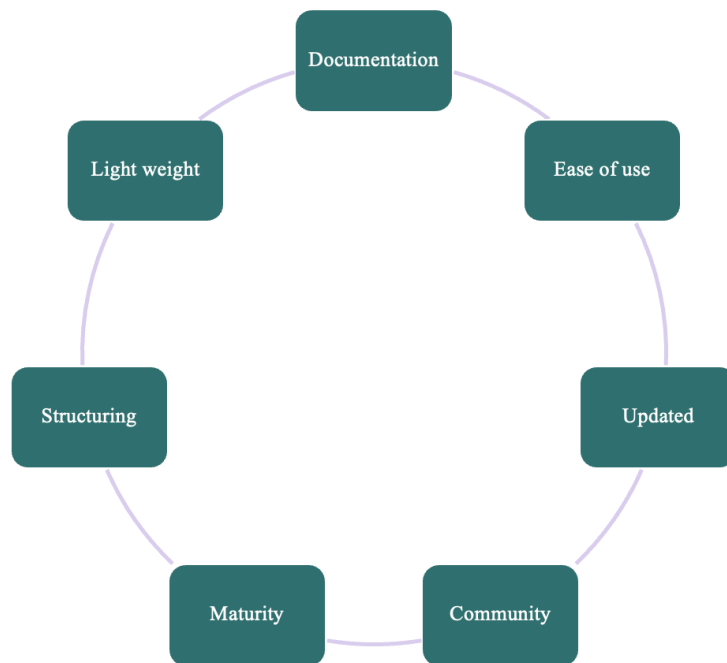


Figure 5.3: Chosen qualities for evaluation model.



Figure 5.4: Start view in prototype for applications.

5.2.2 Definition of quality attributes and resulting evaluation model

For each quality in the model, a number of attributes were defined based on information collected in literature and interviews, as well as the author's own knowledge. Each attribute was also assigned a weight to represent how important it was to the assessment of the related quality. Furthermore, a scoring scale was defined for each attribute to help the assessor assign a score to them. The resulting evaluation model: a detailed presentation of the defined attributes with associated evaluation instructions and scoring scales as well as a model on how to calculate the total quality scores are found in Appendix C.

5.3 Evaluation

Before the model could be applied to the three frameworks, a prototype was designed based on which the applications were developed. The applications consisted of two views: a start view presented in Figure 5.4 and a view where a patient's medication list could be retrieved and displayed from Cambio's APIs. The latter is presented in Figure 5.5.

5.3.1 Vue.js

Table 5.5 presents the scores for all attributes in the evaluation of the framework Vue.js. For each attribute, the scoring is justified by a short motivation.

5.3.2 React.js

Table 5.6 presents the scores for all attributes in the evaluation of the framework React.js. For each attribute, the scoring is justified by a short motivation.

5.3.3 Angular

Table 5.7 presents the scores for all attributes in the evaluation of the framework Angular. For each attribute, the scoring is justified by a short motivation.

Table 5.5: Scoring of quality attributes for Vue.js.

Quality attribute	Score	Motivation
C1	4	3 937 290 downloads
C2	3	34 contributors
C3	2	6366 questions
C4	3	43,5 %
Community	3.05	
E1	4	Vue.js is considered easy to use and get into. A contributing factor is their so-called Single-File Components. For these components all code is written in one file with three different blocks: script, template and style. Because of that, it is intuitive, fast and easy to build an interactive app.
E2	4	Functions in the framework were perceived as easy to understand and use. For example, there were easy ways to render over lists and bind variables to input fields. It also did not require a lot of code, which also contributes to higher understandability. Thus, the attribute is therefore considered to fully apply.
E3	3	Vue.js includes many features that help accelerate development, compared to using no framework at all. Time is saved by not having to write functionality for basic tasks such as model binding. However, there are factors that can possibly slow down development. For example, a Single-File Component that becomes very large and has many lines of code can be more difficult to maintain and work with. Altogether, the attribute is assessed to largely apply.
Ease of use	3.7	
D1	4	The documentation is well written and easy to follow. When possible alternative routes exist, both cases are described in detail and in an easy-to-understand manner. The attribute is considered to fully apply.
D2	3	When developing the application, the experience was that all the necessary documentation was available. However, there were some parts that were found only in the documentation for the old version, Vue2, but not yet in the Vue3 documentation. Because of this, it was sometimes less efficient to find the desired information. Thus, the attribute is assessed to largely apply.
Documentation	3.7	
U1	3	16 releases
U2	3	50.3 %
Updated	3	
M1	3	9 years
M2	3	131 reported bug-issues
Maturity	3	
S1	3	Vue.js emphasizes in its documentation that Single-File Components are recommended for most applications. Thus, a clear component structure is provided for applications written in the framework. However, there are no direct guidelines for the file structure, where the freedom is greater for the developer to control themselves. Together, the resulting assessment is therefore that the attribute largely apply.
S2	2	Although the component structure, as a result of the Single-File Components recommendation, looks the same for most Vue.js applications, the file structure of the projects can differ greatly. This is due to the freedom, since the file structure depends a lot on the personal preferences of the person who sets it up. Thus, the attribute is considered to partly apply.
Structuring	2.7	
L1	3	110 MB
L2	3	0.76 MB
Light weight	3	

Table 5.6: Scoring of quality attributes for React.js.

Quality attribute	Score	Motivation
C1	4	19 873 624 downloads
C2	3	23 contributors
C3	3	38 535 questions
C4	3	41 %
Community	3.3	
E1	4	The great structural freedom that React offers, combined with the fact that both the component's JavaScript and HTML code can be written in the same file, makes it very easy to get an interactive app up and running. The attribute is therefore considered to fully apply.
E2	3	Overall, the functionality that React provides is considered easy to understand and use. The hooks facilitate development and are relatively easy to apply to your own case using on the examples in the documentation. A disadvantage, however, can appear if something does not work as expected in your own case. Then it can be difficult to understand what is wrong because of the complicated functionality behind, for example, a hook. Thus, the attribute is assessed to largely apply.
E3	3	React includes many features that help accelerate development, compared to using no framework at all. Time is saved by not having to write code for basic functionality, such as state handling. However, lack of two-way data binding means that many local small functions need to be written to handle updates of state, which slows down the development a little bit. Moreover, the great level of freedom that comes with React imply that differences in file structure and code structure exists amongst and within projects. That could also impact the speed of the development. Altogether, the attribute is considered to largely apply.
Ease of use	3.2	
D1	4	There are two documentations from React, the reader is directed to use the new one as the old one is no longer updated. The documentation has a clear design with code examples that make it easy to follow. There is also a Quick Start guide which is very helpful for getting a basic application up and running. Altogether, the attribute is considered to fully apply.
D2	3	The documentation served as a good guide during the development of the application. Functionality and hooks in React are included, and most of what is needed for development is easy to find. However, in this case, a proxy was needed to make calls to external APIs during development. For that function, documentation from React itself was missing, but could be found elsewhere in the community. Thus, the assessment is that the attribute largely apply.
Documentation	3.7	
U1	1	3 releases
U2	3	69 %
Updated	2.6	
M1	3	10 years
M2	3	139 reported bug-issues
Maturity	3	
S1	2	The UI elements are made up of components. A large component can also consist of several smaller components. There are no clear requirements or guidelines for how these should be structured. Nor are there any regulations for the file structure in general. However, there are recommendations both in the documentation and in the community. In conclusion, a particularly clear structure is not considered to be provided, and thus attribute is considered to partly apply.
S2	1	The lack of a clearly presented structure means that applications written in React can look very different, as the developers themselves are given the freedom to shape the project. Also, it is still possible to use class components, even though the new way is functional components. As a result, the actual code structure of React can also vary, between applications, and even within the same application if the developers involved have done it in different ways.
Structuring	1.7	
L1	2	237 MB
L2	3	0.56 MB
Light weight	2.8	

Table 5.7: Scoring of quality attributes for Angular

Quality attribute	Score	Motivation
C1	4	3 349 685 downloads
C2	3	38 contributors
C3	2	10 372 questions
C4	2	38.8 %
Community	2.8	
E1	2	Although the documentation is very good, there are factors that contribute to Angular being difficult to learn and get into. For example, that each component consists of three files. This interaction and structure can be difficult to understand at first. In addition, TypeScript is default in Angular. Thus, the developer's freedom is reduced and this makes it more difficult to quickly write a code that works.
E2	2	In the application, Angular's router was used for routing. Moreover, ngModel was used to bind an input field to a variable. The experience of using these features from Angular was good. It felt easy to understand and use. However, the high level of structuring in the framework makes this attribute a little more difficult. The three-file structure of the components affects the understanding of the functionality, at least initially. The attribute is therefore judged to partly apply.
E3	3	Angular includes many features that help accelerate development compared to using no framework at all. Time is saved by not having to write code for basic functionality, such as routing. However, the high level of structuring can have a negative impact on this attribute. Some things take longer to understand due to the three-file structure of the components. Furthermore, this can cause the project to quickly grow into a very large number of files. This can also contribute to reduced understanding and efficiency. In conclusion, the attribute is deemed to largely apply.
Ease of use	2.3	
D1	3	Angular's documentation felt easy to follow. It was a reasonably detailed level, and the given example applications worked well as a supplement to see certain functionalities in the right context. However, the process of understanding were sometimes slowed down by the fact that functionality is often related to both the HTML and the TypeScript file. The documentation therefore shows code in both files. To understand what happens where and how these interact with each other can make the documentation a little complicated at times.
D2	4	During the development of the Angular application, the documentation alone was enough to guide the entire process. Along with the provided example applications, it covered all the necessary parts of the framework. Thus, the attribute is considered to fully apply.
Documentation	3.3	
U1	4	56 releases
U2	3	53 %
Updated	3.2	
M1	4	13 years
M2	4	79 reported bug-issues
Maturity	4	
S1	4	All components consist of three parts: a component-class, an HTML-template and component specific styles. The documentation suggests for these to be put in their own files. This way, each component will have these three files, making it easy to recognize and find the desired code in a project. With regard to that, Angular is deemed as very structuring. Moreover, using TypeScript instead of JavaScript is default in Angular. Typing the code and adding rules also contributes to increased structure. Thus, the attribute is considered to fully apply.
S2	3	The given three-file structure for the components contributes to large similarities in all Angular applications. However, there can still be differences in the file-structure. For example, I choose to create a folder for each component, while some of the example applications did not. How the services are structured can also differ in Angular applications since it is not as clearly stated in the docs. Altogether, this attribute is deemed to largely apply.
Structuring	3.7	
L1	2	298 MB
L2	4	0.17 MB
Light weight	3.6	

Hämta läkemedeslista

Patientens personnummer: Hämta

Patientinformation:

Namn: Johan Johansson Personnummer: 191212121212

Läkemedelslista::

Morfin Abcur Injektionsvätska, lösning 1-5 ml (1 mg-5 mg) vid behov tills vidare, högst 15 ml per dygn, minsta tid mellan doserna : 6 timmar
Morfin Abcur Injektionsvätska, lösning 1-5 ml (1 mg-5 mg) vid behov tills vidare, högst 15 ml per dygn, minsta tid mellan doserna : 6 timmar
Morfin Abcur Injektionsvätska, lösning 1-5 ml (1 mg-5 mg) vid behov tills vidare, högst 15 ml per dygn, minsta tid mellan doserna : 6 timmar
Morfin Abcur Injektionsvätska, lösning 1-5 ml (1 mg-5 mg) vid behov tills vidare, högst 15 ml per dygn, minsta tid mellan doserna : 6 timmar

Figure 5.5: Medication view in prototype for applications.

5.3.4 Spider chart

Figure 5.6 presents the compiled result from the scoring in a spider chart. This way, the evaluated frameworks can easily be compared on the different qualities.

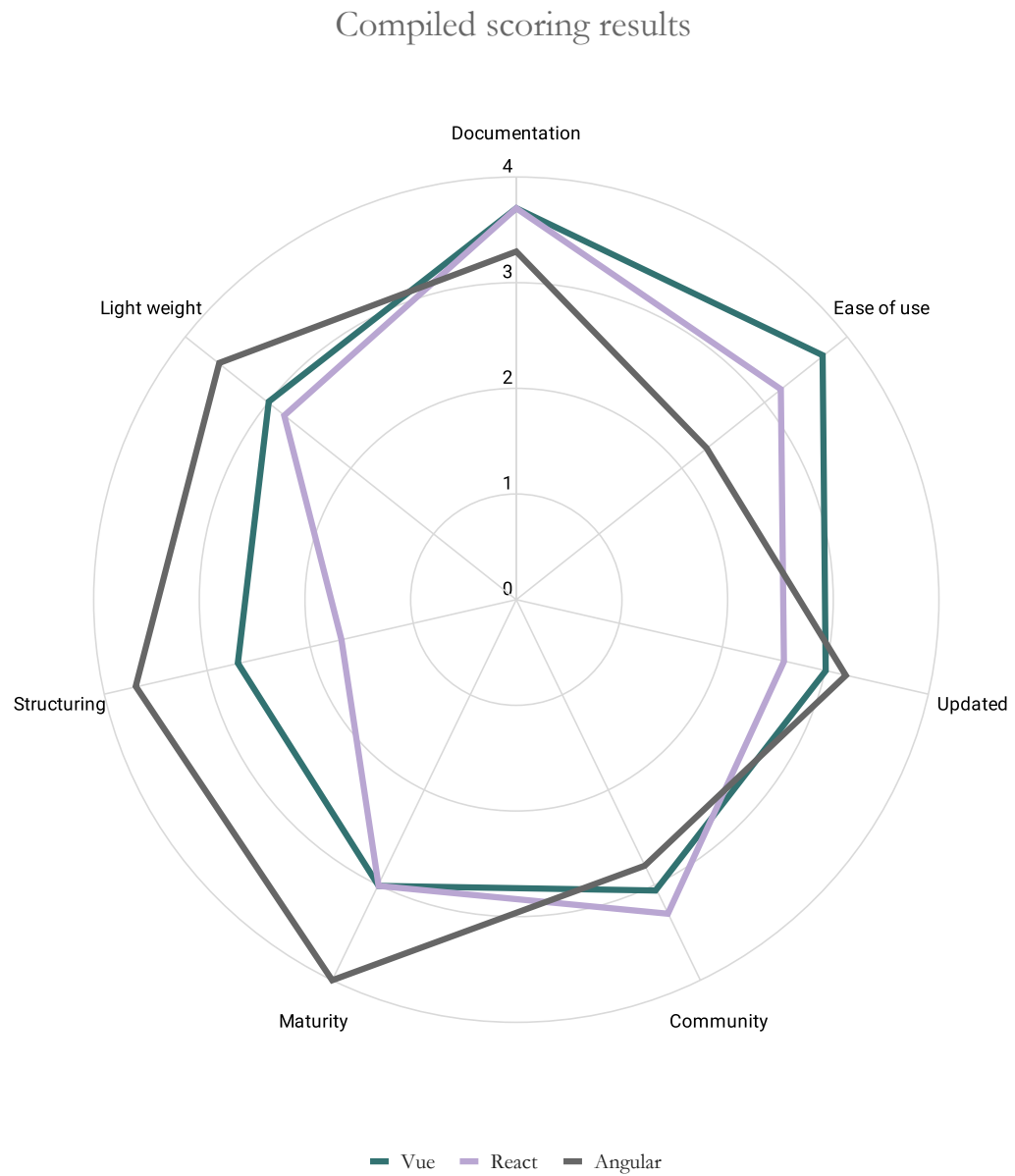


Figure 5.6: The compiled scoring result presented in a spider chart.



6 Discussion

In the following section, this thesis project is discussed from different aspects. Firstly, in Section 6.1, interesting findings from the results are discussed. Thereafter, Section 6.2 discusses and critiques the method and its limitations. Finally, the work is discussed in a wider context in Section 6.3.

6.1 Results

In the following section, the results of the project are discussed. The first part concerns the pre-study's identification of important qualities of a framework. This is followed by a discussion on the evaluation model developed in the thesis project and the testing of it.

6.1.1 Important qualities

All the qualities identified in the literature study were also mentioned at some point in the interviews, which confirms the results from the related work and indicates that it still has relevance for the field and its practitioners. However, two of the qualities from the literature were named differently in the interviews but were judged to have a similar meaning. For these qualities, it was chosen to combine them for the overall analysis. Thus, Good Architecture was included in Structuring and Performance in Light Weight.

When the interviewees were asked at the end of the interview to choose the three most important qualities from the literature, a few interesting differences could be identified. Performance and Size of Code were both highly ranked according to the literature study, but ended up at the bottom based on the interviewees' ranking. Only one person thought Performance was one of the three most important, and no one chose Size of Code. However, Performance was later merged with Light Weight, a quality by mentioned 8 out of 11 interviewees and ranked at the top of the list in the interview results. The results from the interviews thus indicate that if Performance had instead been called Light Weight in the literature, it is likely that more interviewees would have chosen the quality as one of the three most important.

The interviews identified four new qualities not previously mentioned in the studied literature. Of these, Freedom and Maturity were mentioned enough times to be considered relevant. Freedom was often mentioned in connection to Structure. It was, according to these persons, desirable that the framework contributed to Structure but did not limit Freedom too much. This indicates that the proposed presentation method, a spider chart, can be helpful when such a quality comes into play. Structure and Freedom are to some extent a trade-off. Thus, one can study how the framework places itself on that axis in a decision related to that trade-off.

In the survey, only two people chose to add a quality themselves. This indicates that literature and interviews had covered the most important ones. An interesting observation is that Documentation by far was classified as most important according to survey respondents. However, that quality did not end up at the top according to the interviews or literature. The newly identified quality Maturity was ranked highly according to the survey, which further confirms its importance. Freedom was rated lower according to the survey than in the interviews. This might be a consequence of all the interviewees working in the same company and department. Thus, it could mean that it is a quality considered important at this specific company, while others do not prioritize it as highly.

6.1.2 The evaluation model

During the evaluation phase, the developed model was tested by applying it to the three most popular frameworks at the time. The test made it clear that the chosen quality attributes combined with their weights had a decisive role in the result. Unlike the identified qualities, which were based on data from the three parts of the pre-study, the attributes and weighing were not as well substantiated. These were instead defined based on information in the interviews together with the author's own knowledge in the area, with advice from the external supervisor. As a result, another researcher, with other interviewees, could possibly get a different result in terms of quality attributes and weights. Thus, the defining of them can be said to have a negative impact on the validity and reliability of the study.

However, based on the selected attributes, the testing of the model resulted in some interesting insights. As shown in the spider chart in Figure 5.6, React.js and Vue.js are closely followed in the graph and perform similarly for many of the qualities. The two frameworks received higher scores for the attributes Documentation, Ease of use and Community compared to the third framework, Angular. Instead, Angular is the high performer for the qualities Light weight, Structuring and Maturity. The framework also ranks highest in the last quality, Updated, but the differences are marginal. According to the result from the pre-study (Figure 5.2) the four most important qualities in a front-end JavaScript framework are:

1. Documentation
2. Ease of use
3. Updated
4. Community

Thus, the result indicates that Vue.js and React.js outperforms Angular on three of the four most important qualities. However, for the qualities where Angular is the highest performer, the difference between that framework and the other two is often very large. Thus, Angular is significantly better with regard to those qualities, even if they are not considered to be the most important ones according to the results from the preliminary study.

During the interviews, Angular was mentioned several times as a heavy and "bloated" framework. The result, however, showed the opposite. When tested, it appeared that the size after a release build, which was considered to be of the most importance for the decision, was the smallest in Angular, as the compression was better than in the other two. This further confirms the need for a structured way of evaluating frameworks before adopting one in a project. Angular feels more "bloated" when downloading and starting a project, but in the most important context, release, it actually performs best out of the three for the Light weight quality.

As expected, the results show that all frameworks are good from different aspects (qualities). There is at least one quality for each framework where it is the highest performing. Furthermore, the differences are small between two or all frameworks on many of the qualities. This indicates that the model can guide a decision on the choice of framework, but that the decision maker themselves need to prioritize which of the qualities are most important for the specific project. Thereafter, the model's results, the spider chart, can be studied to identify which framework performs best for the most prioritized quality and which trade-offs that may entail for other qualities.

6.2 Method

This thesis project followed a three-phase method. In Chapter 4, a detailed description of the method is found. To achieve a high degree of replicability, the method description has been written with the intention to be descriptive and easy to follow. However, there are limitations in the method and execution of the project that may have had consequences for the outcome. In the following section, the method is discussed and criticized based on the most important limitations for each phase of the project and how these limitations may have affected the replicability of the project and the validity and reliability of its results. This is followed by a critical discussion of the sources used in the project.

6.2.1 Pre-study

The data collection during this phase took place in two parts: interviews at the case company and a survey aimed at developers outside the company. Runesson and Höst [14] highlights the importance of using more than one source of data in a study to avoid the results being biased from using one single data source. Furthermore, the authors state that the validity of a study is increased when the same conclusion can be drawn from different sources of information. Supplementing the interviews with a survey aimed at a wider target group was done with the intention of increasing the credibility of the study in the way that Runesson and Höst [14] describe. Although the survey was an improving factor, the validity of the interview results can be further discussed. It was only through the interviews that new important qualities were identified. All interviews were held with developers at the case company and the department with the project was carried out in collaboration with. In order to achieve what Runesson and Höst [14] call triangulation, and thereby increase the study's credibility, the same interview procedure would need to be conducted with developers at other companies to confirm the results. It would also increase the likelihood that another researcher would achieve the same results if the study were repeated, but with a different group of interviewees. Thus, the reliability of the study would be improved as well.

6.2.2 Definition of evaluation model

The definition of the evaluation model followed the DOSAM model presented by Bergner et al. [2] and further explained in Section 3.3. The fact that the definition follows an already established process contributes to the project's replicability. The choice of qualities to include

in the model was based on data from the three stages of the previous pre-study. However, as discussed in Section 6.1.2, the definition of the related quality attributes and weights was not as well substantiated. Finding literature to define the attributes that would measure the framework's performance on the various qualities was difficult. Instead, the choice was primarily based on information from the interviewees. In cases where such information was not available, the author's own experience in the field was used with guidance from the external supervisor. This could have had consequences both for the validity and reliability of the project. Since the metrics used to measure the performance of the frameworks are not based on data from a structured data collection, we cannot say for sure that they actually measure what we want. Furthermore, it is likely that in a repeat of the project one would not get exactly the same statements in the interviews and thus perhaps not the same attributes either. Runesson and Höst [14] suggests that the resulting analyses could be sent back to the interviewees to increase the validity of a study. In this case, it could have been allowing the interviewees to review the quality attributes that were defined to measure the performance of the qualities.

6.2.3 Evaluation

During the evaluation, the model was tested by applying it to three different JavaScript frameworks. For the qualitative qualities, the scoring is based on the assessor's own experience of the frameworks. Either through experience from the past or through a smaller application being developed in the framework to gain experience and an idea of what it is like to use the framework. For these attributes, it is possible that the assessment could be biased by the assessor's level of experience with the frameworks. Developing a small application is not the same as having worked with a framework in a real project before. This could have affected the reliability of the evaluation results. With a different person as assessor, the result could have been different. An alternative, to reduce the impact on validity, could have been to introduce an additional step in the evaluation model in which an application always is developed in the framework according to a prepared specification. This would mean that the assessor has an equivalent experience for all frameworks to base the assessment on. On the other hand, it would be an unnecessary step in the evaluation of frameworks where experience already exists. For a consulting company, as in the case company's situation, it is a waste of time and money. In addition, the smaller application developed for the evaluation can be adapted to the project for which the evaluation is carried out and include parts relevant to the specific project.

6.2.4 Source criticism

This master's thesis has been based on contemporary research published in connection with conferences within software engineering. The intention has been to use articles in the area of software engineering to the greatest extent possible. However, for certain method-related parts, articles aimed at a larger scientific area have been used.

In the literature study, four peer-reviewed articles on important parameters in front-end frameworks have been studied. All have been published in connection with good conferences and can be considered reliable sources in that respect. However, as pointed out by Graziotin and Abrahamsson [8], Pano et al. [13] and Ferreira et al. [6], there is not much peer-reviewed research in the area and because of that some relatively aged sources have had to be used. This may have affected the relevance that these articles' results have for a contemporary research project such as this. Furthermore, the sources used refer to, and build on, each other's research. The positive is that each article has aimed to improve the results of the previous ones. However, it can be argued that this also imply a less nuanced picture of the studied area, since all the sources have been influenced by each other.

6.3 The work in a wider context

The Swedish Scientific Council [16] present and discuss ethics and morality in the field of research. The council states that when people are involved in research, for example through interaction or observation, they must always be informed about the study to which their participation is intended to contribute. In this thesis project, it is relevant to carefully ensure that participating persons are informed about the purpose of the interviews and the survey. For the interviews, this was fulfilled partly through an information email that went out with the invitation to the interview and partly with a repetition of it at the beginning of the interview itself. Regarding the survey, it was clearly stated how the results would be used in the questionnaire's informational text.

During the development of the evaluation model and the identification of important qualities, the focus was solely on the needs of the practitioners. This might be counterproductive from a sustainability perspective. For example, from that perspective, the highest possible performance is perhaps desirable. However, a higher performance often also means a greater energy consumption, which is not desirable from a sustainability perspective.

An evaluation model that contributes to the correct selection of front-end frameworks can also be beneficial from an economic perspective. For a consulting firm, being able to, at an initial stage, choose the framework that best suits a specific project and its requirements can contribute to the ability to cost-effectively meet customer requirements. In the short term, it can make the project more profitable, and in the long term, increased customer satisfaction will benefit the consultant company's business.



7 Conclusion

This master's thesis aimed to gain a deeper understanding of what is important when evaluating a front-end framework for a software project. Thereafter, based on that, develop a structured model that can guide decision makers when approaching the question of which framework to use. To fulfill the aim, this thesis intended to answer the following research questions:

- **RQ1:** What is, according to contemporary research projects, important to consider when deciding which web front-end framework to adopt in a project?
- **RQ2:** What is, according to professional web front-end developers, important to consider when deciding which framework to adopt in a project?
- **RQ3:** How can a structured method to evaluate a front-end framework and how well it fits a project look like based on the parameters identified in RQ1 and RQ2?

The project's pre-study began with a literature study of four related works to identify, according to the literature, important qualities of a front-end JavaScript framework. Thereafter, interviews were carried out with front-end web developers at the case company, Tietoevry Care. The interviews fulfilled two purposes: further identification of important qualities and prioritization of those already identified. The pre-study was then concluded with a survey that was sent out via several channels to reach front-end web developers outside the case company and thus increase the validity of the results. Altogether, the three activities helped answer RQ1 and RQ2 by identifying and then prioritizing the most important qualities of a front-end JavaScript framework. The result, presented in Figure 5.2, was then used as a basis to develop an evaluation model and thus answer RQ3.

The validity of the process used to select and weight the attributes used to score the qualities in the model can be questioned. However, the testing still shows that the model and its design presents an example of what a structured method for evaluating JavaScript frameworks for a software project can look like. The result shows that all three frameworks perform best on at least one of the qualities. Thus, no framework is always the best choice. This confirms the need for a structured model, such as this one, to guide decision makers in selecting the framework that suits the needs and requirements for a specific project. The

result is presented in a spider diagram. This makes it easy for the assessor to identify which framework performs best on the most prioritized qualities, and the trade-offs it entails for the other qualities. Thereby, the model provides the decision maker with the information needed to make a well-founded decision, and thus constitutes an answer to RQ3.

The result of this thesis project is a modern investigation of important qualities of a front-end JavaScript framework. From a research perspective, the result contributes by filling the gap in a previously rather unexploited area. Practically, the result can be used by web developers and companies to choose the right framework for a project. The research has consistently focused on JavaScript frameworks. Thus, the model can mainly be applied to such. However, it is likely, but not confirmed, that the same qualities would be considered important also for other types of software frameworks. That would make the evaluation model useful in a broader context as well.

7.1 Future work

Recommendations for future work are based on the limitations discussed in Section 6.2. The first is to conduct similar interview surveys at more companies. This could confirm important qualities identified in this study through different sources of information. That way, what Runesson and Höst [14] refer to as triangulation could be achieved. Such research would increase the validity and credibility of the results obtained in this thesis project. The second recommendation is to further develop the model with more well-substantiated quality attributes. Suggestively, it can be done by having practitioners review the model presented in this project and reconstruct attributes and weights according to their feedback. Alternatively, a data collection can be carried out where interviewees are asked to describe how they believe that the performance for the identified qualities can be measured. Based on that data, the model can then be reconstructed to improve its validity.



Bibliography

- [1] Steve “ardalis” Smith. *Architecting Modern Web Applications with ASP.NET Core and Microsoft Azure*. Redmond, Washington: Microsoft Developer Division, .NET, and Visual Studio product teams, 2022.
- [2] Klaus Bergner, Andreas Rausch, Marc Sihling, and Thomas Ternité. “Dosam—domain-specific software architecture comparison model”. In: *Quality of Software Architectures and Software Quality: First International Conference on the Quality of Software Architectures, QoSA 2005, and Second International Workshop on Software Quality, SOQUA 2005, Erfurt, Germany, September 20-22, 2005. Proceedings*. Springer. 2005, pp. 4–20. DOI: 10 . 1007 / 11558569_3.
- [3] Daniela Borissova, Zornitsa Dimitrova, Vasil Dimitrov, Radoslav Yoshinov, Magdalena Garvanova, and Ivan Garvanov. “Multi-Attribute Decision-Making Model for Ranking of Web Development Frameworks”. In: *2021 25th International Conference on Circuits, Systems, Communications and Computers (CSCC)*. IEEE. 2021, pp. 3–8. DOI: 10 . 1109 / CSCC53858 . 2021 . 00009.
- [4] Virginia Braun and Victoria Clarke. “Using thematic analysis in psychology”. In: *Qualitative research in psychology* 3.2 (2006), pp. 77–101. DOI: 10 . 1191 / 1478088706qp063oa.
- [5] Victoria Clarke, Virginia Braun, and Nikki Hayfield. “Thematic analysis”. In: *Qualitative psychology: A practical guide to research methods* 222.2015 (2015), p. 248.
- [6] Fabio Ferreira, Hudson Silva Borges, and Marco Tulio Valente. “On the (un-) adoption of JavaScript front-end frameworks”. In: *Software: Practice and Experience* 52.4 (2022), pp. 947–966. DOI: 10 . 1002 / spe . 3044.
- [7] Andreas Gizas, Sotiris Christodoulou, and Theodore Papatheodorou. “Comparative evaluation of javascript frameworks”. In: *Proceedings of the 21st International Conference on World Wide Web*. 2012, pp. 513–514. DOI: 10 . 1145 / 2187980 . 2188103.
- [8] Daniel Graziotin and Pekka Abrahamsson. “Making sense out of a jungle of JavaScript frameworks”. In: *International Conference on Product Focused Software Process Improvement*. Springer. 2013, pp. 334–337. DOI: 10 . 1007 / 978-3-642-39259-7_28.

-
- [9] José Ignacio Fernández-Villamor, Laura Diaz-Casillas, and Carlos Á Iglesias. "A comparison model for agile web frameworks". In: *Proceedings of the 2008 Euro American Conference on Telematics and Information Systems*. 2008, pp. 1–8. DOI: 10.1145/1621087.1621101.
 - [10] Slavina Ivanova and Georgi Georgiev. "Using modern web frameworks when developing an education application: a practical approach". In: *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE. 2019, pp. 1485–1491. DOI: 10.23919/MIPRO.2019.8756914.
 - [11] Barbara A Kitchenham and Shari Lawrence Pfleeger. "Principles of survey research: part 3: constructing a survey instrument". In: *ACM SIGSOFT Software Engineering Notes* 27.2 (2002), pp. 20–24. DOI: 10.1145/511152.511155.
 - [12] Timothy C Lethbridge, Susan Elliott Sim, and Janice Singer. "Studying software engineers: Data collection techniques for software field studies". In: *Empirical software engineering* 10.3 (2005), pp. 311–341. DOI: 10.1007/s10664-005-1290-x.
 - [13] Amantia Pano, Daniel Graziotin, and Pekka Abrahamsson. "Factors and actors leading to the adoption of a JavaScript framework". In: *Empirical Software Engineering* 23.6 (2018), pp. 3503–3534. DOI: 10.1007/s10664-018-9613-x.
 - [14] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical software engineering* 14.2 (2009), pp. 131–164. DOI: 10.1007/s10664-008-9102-8.
 - [15] npm trends. *@angular/core vs ember-source vs next vs nuxt vs preact vs react vs svelte vs vue*. 2022. URL: <https://npmtrends.com/@angular/core-vs-ember-source-vs-next-vs-nuxt-vs-preact-vs-react-vs-svelte-vs-vue> (visited on 11/14/2022).
 - [16] Vetenskapsrådet. *God forskningsed*. 2017. URL: <https://www.vr.se/analys/rapporter/vara-rapporter/2017-08-29-god-forskningssed.html>.
 - [17] W3Techs. *Usage statistics of client-side programming languages for websites*. 2022. URL: https://w3techs.com/technologies/overview/client_side_language (visited on 01/30/2023).
 - [18] Claes Wohlin and Per Runeson. "Guiding the selection of research methodology in industry-academia collaboration in software engineering". In: *Information and Software Technology* 140 (2021), p. 106678. DOI: 10.1016/j.infsof.2021.106678.

A

Interview Structure

A.1 Introduktion och presentation av studien

- Hälsa personen välkommen och tacka för deltagande i studien.
- Presentation av studien och hur intervjun ska gå till.
 - Jag genomför mitt examensarbete här hos er under våren. Målet med min studie är att ta fram en utvärderingsmodell för att utvärdera och jämföra olika front-end ramverk för applikationer skriva i JavaScript.
 - Med denna intervju vill jag förstå vilka egenskaper och kvalitéer du som utvecklare tycker är viktiga hos ett ramverk.
 - Intervjun kommer inledas med några enkla bakgrundsfrågor om dig. Därefter kommer vi gå in på frågor om ramverk och viktiga egenskaper hos dessa. Frågorna kommer vara öppna och av diskuterande karaktär med intentionen att skapa ett samtal.
- Presentation av hur data från intervjuer kommer att användas.
 - Det är bara jag som kommer att ta del av intervjun. Svaren kommer att analyseras och slås ihop med övriga svar för att kunna kategorisera vad front-end webbutvecklare hos er på Tietoevry prioriterar i valet av ramverk.
 - Godkänner du att delar dina svar citeras anonymiserat i rapporten för att ge beskrivningar av parametrar som nämnts som viktiga? (*Ja/Nej*)

A.2 Inledande frågor

- Berätta lite mer om din erfarenhet av utveckling. Vilken typ av projekt sitter du i nu och vilka tekniker används i det projektet?
- Vilken erfarenhet har du av att jobba med front-end webbutveckling och JavaScript?
- Vilken erfarenhet har du av att använda ramverk för att utveckla i JavaScript?
 - Om tidigare erfarenhet: Vilka ramverk har du utvecklat i?

A.3 Intervjusektion

- Det finns många typer av ramverk, inte bara för att utveckla i JavaScript. Till exempel är .NET ett ramverk som ni använder er av väldigt mycket. I generella drag, vilka kvalitéer prioriterar och förväntar du dig när du använder ett ramverk?
- Finns det några egenskaper hos de ramverk du har erfarenhet av som du anser vara irriterande eller dåliga?
- Är samma kvalitéer viktiga när vi pratar om JavaScript ramverk specifikt? Är det några som tillkommer, faller bort eller blir mer eller mindre viktiga?
- Slutligen har ett antal kvalitéer identifierats som viktiga enligt litteraturen. Jag kommer att visa dessa för dig och vill då att du väljer ut tre stycken som du anser vara viktigast hos ett ramverk:

- ☐ Previous experience - You and/or your colleagues have experience from the framework
- ☐ Performance - Applications built using the framework have good performance
- ☐ Good architecture - Contributes to good architecture and to the "code-less-do-more" factor
- ☐ Community - Popular and with a large and active community
- ☐ Size of code - Contributes to a small amount of code
- ☐ Documentation - The quality and availability of documentation is high
- ☐ Updated - Continuously maintained and updated
- ☐ Ease of use - Easy to learn. Writing and maintaining code is easy
- ☐ Suitability - Is suitable for the project, all necessary libraries etc. are available
- ☐ I don't know
- ☐ Other: _____



B Survey Structure

This appendix presents the survey that was sent to front-end web developers during the data collection phase of the project. Section B.1 includes the text that was sent out to developers, together with the link to the survey. The actual survey is then presented in Section B.2.

B.1 Information text

In English

My name is Sara, and I am currently writing my master's thesis at Linköping University within Computer Science and Engineering. As part of my thesis, I am interested in what qualities professional web front-end developers value in a JavaScript front-end framework. By completing this survey, you will help me a lot in answering that question.

Answering it takes only **two minutes**, and I will be forever grateful!

Thank you in advance, and have a nice day!
Kind regards,
Sara

In Swedish

Mitt namn är Sara, och jag skriver just nu mitt examensarbete vid Linköpings Tekniska Högskola inom datateknik. Som en del i mitt arbete är jag intresserad av vilka kvalitéer professionella front-end webbutvecklare värderar hos ett JavaScript front-end ramverk. Genom att svara på den här enkäten hjälper du mig jättemycket i att besvara den frågan.

Att svara tar endast **två minuter**, och jag kommer att vara evigt tacksam!

Tack på förhand, och ha en fin dag!
Vänliga hälsningar,
Sara

B.2 Survey

What do you value in a JavaScript front-end framework?

My experience in front-end web development

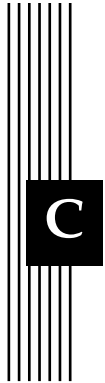
	1	2	3	4	5	
No experience	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very experienced

I have experience with the following front-end JavaScript frameworks

- ☐ Angular
- ☐ React
- ☐ Vue
- ☐ Ember
- ☐ Svelte
- ☐ Other: _____

I value these qualities the most in a JavaScript framework (pick at most 4 options)

- ☐ Previous experience - Team members have knowledge and enjoy coding in it. Syntaxes match team member preferences.
- ☐ Suitability - Adapted to the problem to be solved. Corresponds to needs, functional- and non-functional requirements.
- ☐ Light weight - The initial output size is not too big, affecting performance and load times. Installation does not require downloading a large amount of packages.
- ☐ Browser compatibility - Offer compatibility with all major browser vendors
- ☐ Structuring - Lots of code conventions and set ways of doing things. Contributes to good and recognizable architecture.
- ☐ Community - Popular and widely used. Support is available from a stable, large and active community.
- ☐ Size of code - Require less boilerplate code.
- ☐ Documentation - The documentation is good, explanatory and sufficient.
- ☐ Security - Not too many third-party dependencies that pose an extended risk for supply chain attacks.
- ☐ Updated - Continuous maintenance, development and updates. Now and in the future.
- ☐ Ease of use - Learning and getting started is easy. Make code maintenance easier and coding more efficient.
- ☐ Freedom - Few rules and conventions, allowing own solutions and choices. Possible to study the source code to deep dive in advanced functionality
- ☐ Maturity - Established and with a certain level of maturity. Stable updates, few breaking changes. Not too many bugs.
- ☐ I don't know
- ☐ Other: _____



C Evaluation model

The following model is used to evaluate JavaScript front-end frameworks based on the parameters identified as most important in such a framework. To assign a total score to the seven important qualities for a framework, start by following the instructions in Section C.1 and assign a score to each quality attribute. Then calculate the total score for each quality according to the model in Section C.2 and Figure C.1. The result is then presented in a spider diagram. Partly to facilitate comparisons between different frameworks, and partly to visualize trade-offs between different qualities. In this way, decision makers can easily find the framework that best matches a project's needs.

Note: If no previous experience with the framework exists, the evaluator needs to familiarize themselves with the framework first in order to then be able to set a score, especially for the three qualitative qualities: *Ease of use*, *Documentation* and *Structuring*. This is done by developing a smaller application in the framework using the documentation that the framework provides. For an example of what such an application might look like, see the prototype images for the applications developed in this study presented in Section 5.3. The gained experience is then the basis for the assessment on the four-element scale.

C.1 Scoring of quality attributes

Follow the table's instructions to give the framework a score for each quality attribute. Use the given scoring guidelines for help and guidance in the assessment. For the qualitative attributes, you need to make an assessment of how well the attribute matches the characteristics of the framework based on your own opinion.

Attribute	Instruction	Scoring scale	Weight
C1: Downloads during the past year	Find downloads statistics for the framework, for example on: https://npm trends.com/@angular/core-vs-ember-source-vs-next-vs-nuxt-vs-preact-vs-react-vs-svelte-vs-vue	1 p Less than 500 000 2 p 500 000 - 1 000 000 3 p 1 000 001 - 2 000 000 4 p More than 2 000 000	30%
C2: Contributors in GitHub repository during the past year	Go to the framework's GitHub repository → Insights → Contributors and mark one year back in the graph.	1 p Less than 10 2 p 10 - 20 3 p 21 - 50 4 p More than 50	20%
C3: Questions with accepted answers on Stack Overflow during the past year	Go to Stack Overflow and search for questions tagged with the framework. Use filtering to get the number of questions without any accepted answers. Find the number of questions in total during the past year, and subtract with the number without accepted answers from the last year.	1 p Less than 5 000 2 p 10 000 - 20 000 3 p 20 001 - 50 000 4 p More than 5 000	25%
C4: Ratio of questions with accepted answers on Stack Overflow during the past year	Divide the number from C3 by the total number of questions from the past year.	1 p Less than 20% 2 p 20% - 40% 3 p 41% - 70% 4 p More than 70%	25%
E1: Understanding how, and getting an interactive application up and running, is quick and easy	Select the level of attribute fulfillment according to the scoring scale.	1 p Does not apply 2 p Partly apply 3 p Largely apply 4 p Fully apply	20%

E2: Understanding and using framework functions is easy	Select the level of attribute fulfillment according to the scoring scale.	1 p Does not apply 2 p Partly apply 3 p Largely apply 4 p Fully apply	50%
E3: Using the framework accelerates development	Select the level of attribute fulfillment according to the scoring scale.	1 p Does not apply 2 p Partly apply 3 p Largely apply 4 p Fully apply	30%
D1: The documentation is easy to follow and understand	Select the level of attribute fulfillment according to the scoring scale.	1 p Does not apply 2 p Partly apply 3 p Largely apply 4 p Fully apply	70%
D2: The documentation covers all functionality in the framework	Select the level of attribute fulfillment according to the scoring scale.	1 p Does not apply 2 p Partly apply 3 p Largely apply 4 p Fully apply	30%
U1: Releases during the past year	Go to the framework's GitHub repository → Releases and count the number of releases in the past year (not including pre-releases)	1 p Less than 6 2 p 6 - 12 3 p 13 - 24 4 p More than 24	20%
U2: Ratio of closed bug issues on GitHub during the past year.	Go to the framework's GitHub repository → Issues. Use the filtering input field to filter on creation date, starting a year back, and bug label(s). Count open issues and closed issues. Divide the number of closed issues by the total number of issues (open and closed) to get the ratio.	1 p Less than 20% 2 p 20% - 50% 3 p 51% - 80% 4 p More than 80%	80%

M1: Years the framework has existed	Search on Google for the first release of the framework.	1 p Less than 2 2 p 2 - 5 3 p 6 - 10 4 p More than 10	30%
M2: Reported bug issues on GitHub during the past year	Go to the framework's GitHub repository → Issues. Use the filtering input field to filter on dates starting a year back and bug label(s). Count both open and closed issues.	1 p More than 400 2 p 201 - 400 3 p 100 - 200 4 p Less than 100	70%
S1: The framework presents a clear way to structure applications	Select the level of attribute fulfillment according to the scoring scale.	1 p Does not apply 2 p Partly apply 3 p Largely apply 4 p Fully apply	70%
S2: All applications built in the framework look the same	Select the level of attribute fulfillment according to the scoring scale.	1 p Does not apply 2 p Partly apply 3 p Largely apply 4 p Fully apply	30%
L1: Initial output size before build [MB]	Create an initial project in the framework. Run a command to install all packages and dependencies. Get the resulting size of the project folder.	1 p More than 300 2 p 201 - 300 3 p 100 - 200 4 p Less than 100	20%
L2: Initial output size after build [MB]	Create an initial project in the framework. Run a command to install all packages and dependencies. Then run a command to build a release version of the project. Get the resulting size of the project build folder.	1 p More than 3 2 p 1-3 3 p 0.5-1 4 p Less than 0.5	80%

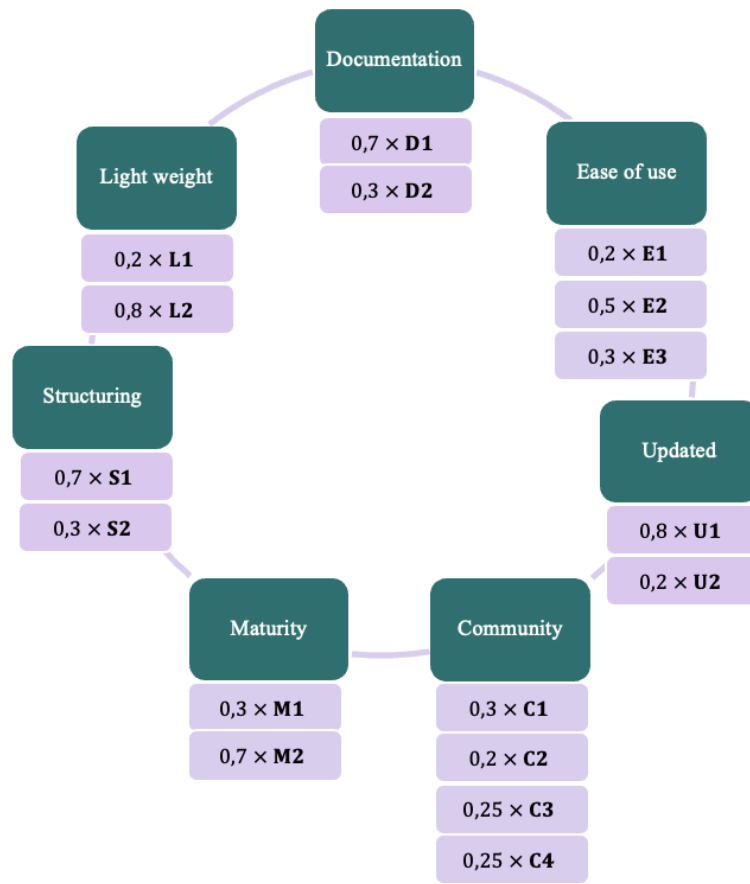


Figure C.1: Evaluation model qualities with related quality attributes and weights.

C.2 Calculation of total quality scores

Once all attributes have been assigned a score, the next step is to calculate the total score for all qualities. This is done by summing the product of each quality-related attribute and its weight, as illustrated in Figure C.1.

C.3 Presentation of results

The suggested way to present the results is in a spider diagram. There are several advantages to that presentation method. First, it is an easy way to visualize and identify trade-offs among the qualities: it is clearly visible which qualities suffer because a high score on another quality is prioritized. Furthermore, if the results of several frameworks are included in the same diagram, it is easy to compare them against each other on different qualities.