DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2016

Comparison of Single-Page Application Frameworks

A method of how to compare Single-Page Application frameworks written in JavaScript

ERIC MOLIN

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

KTH Computer Science
and Communication

Comparison of Single-Page Application Frameworks

A method of how to compare Single-Page Application frameworks written in JavaScript

ERIC MOLIN

KTH CSC Supervisor: Dilian Gurov
Principal Supervisor: Sven Norman
Examiner: Olov Engwall
October 9, 2016

Abstract

This degree project is a scientific study where the focus is to formulate a method of how to compare Single-Page Application (SPA) frameworks written in JavaScript. To solve the problem, an abstraction of SPA frameworks is required. This abstraction is completed using a criteria-based approach. Criteria are collected from literature and interviews with experienced developers. Every criterion is defined and has a set of questions evaluating it. In addition to the abstraction concepts are extracted from other comparative methods, such as performance testing and code comparison. The method is evolved into two separate parts, a theoretical and a practical.

Finally, the method is tested on three different frameworks, AngularJS, Angular 2 and React, by implementing a prototype for each framework. From this prototype, code comparison and performance tests are conducted and evaluated. According to the method, AngularJS is suggested to be the best choice. However, the majority of the results from the theoretical part were more or less identical whereas practical part showed more differences. For future reference, this method could be evaluated to other comparative methods or be extended with more criteria and questions.

Referat

Jämförelse av ramverk för single-pageapplikationer

Detta examensarbete är en vetenskaplig studie där fokus är att formulera en metod för hur man kan jämföra ramverk för single-pageapplikationer (SPA) skrivna i JavaScript. För att lösa problemet, behövs en abstraktion av SPA-ramverk. Denna abstraktion använder sig av ett kriterium baserat tillvägagångssätt. Kriterier samlas in från litteratur och intervjuer med erfarna utvecklare. Varje kriterium definieras och har en uppsättning frågor som utvärderar det. Förutom abstraktionen extraheras koncept från andra komparativa metoder, såsom prestandatestning och kod jämförelse. Metoden utvecklades i två separata delar, en teoretisk och en praktisk.

Slutligen testas metoden på tre olika ramverk, AngularJS, Angular 2 och React genom att implementera en prototyp för varje ramverk. Från denna prototyp kan kod jämförelser och prestandatester genomföras och utvärderas. Resultatet visar att AngularJS är det bästa valet. Men de flesta av resultaten från den teoretiska delen var mer eller mindre identiska medan den praktiska delen visade fler skillnader. För framtida utveckling, kan denna metod utvärderas med andra jämförbara metoder eller utökas med flera kriterier och frågor.

Contents

Chapter 1

Introduction

The first chapter introduces the background to the thesis, a description of the problem and its delimitation. The problem is broken down into three sub questions.

1.1 Background

Despite their popularity, web applications have suffered from poor user interactivity for many years [41]. Since the beginning of 2000's the Web evolved into what is now referred to as Web 2.0. Among new techniques and concepts of Web 2.0 is AJAX (Asynchronous JavaScript and XML). This technique made it possible for web applications to asynchronously fetch new data and to update the web page, without refreshing it [43]. Later, this evolved into a new type of web application, known as SPA (Single-Page Application).

Developers often use one or more frameworks and libraries when developing large and complex web applications. They can reuse code and therefore spend more time on designing the current application by choosing an appropriate framework. On the other hand, a less suitable framework can affect the development time and reduce the quality of the application. Still, web framework comparison is not an established discipline and the closest field is software architecture comparison. Software architecture comparison is a young discipline were one of the most popular methods is SAAM (Software Architecture Analysis Method), which originated 1996 [17].

This project is performed at Decerno. It is a small IT consultancy company that has built custom systems since the 80's. Their focus is web development and as they make custom systems, they choose tools and framework as required for the particular project. Currently, all their customer's demand systems built as an SPA.

1.2 The Problem

Without any scientific methodology for comparing SPA frameworks written in JavaScript, it is troublesome to choose which one to use. The purpose of this thesis is to formu-

late a scientific way of how to compare them. This section gives a further definition of the problem.

1.2.1 Problem Description

A large number of frameworks exist and evolve constantly. There is no generally agreed definition of what a framework is. It is difficult to compare other frameworks to SPA frameworks due to its complexity. Thus, there is a lot to take into consideration when deciding what framework to use when developing SPAs. For example, there are differences between implementation of data bindings, cache performance and loading time.

Firstly, to be able to make a comparison, it is necessary to establish a definition of what a framework is and what to include. Secondly, by creating an abstraction of an SPA framework making it easier to compare since the complexity of the comparison is lowered. This abstraction can also provide a base to design a comparison method and a foundation to investigate different criteria included in a method to compare SPA frameworks. The problem that this thesis is intended to solve is: "How should a SPA framework comparison method be formulated to provide a recommendation about which framework to use?". This also includes determining the adequate abstraction level of describing an SPA framework.

1.2.2 Delimitation

The study is based on SPA frameworks written in JavaScript. Most of the SPAs have a back-end service as a database or similar. However, the focus is solely on the client side of the SPA, where the part of it using the JavaScript framework is. The proposed method focuses on SPAs of such size and complexity that benefit from using a properly chosen framework.

1.3 Approach

In order to solve the problem, a literature study of software architecture comparison and framework comparison is conducted. As a complement, interviews were made with ten experienced developers. To be able to make an abstraction of a SPA, concepts and criteria are then chosen by a quantitative and qualitative approach. Three frameworks are chosen and the method suggested in this thesis is used to compare these.

Web Related Glossary

ECMAScript: A language specification that JavaScript is implementing.

View: The visual representation of the web application.

Model: The data models or objects that are used in the web application.

Template: A template of how the view should be represented with the data from the models.

DOM (Document Object Model): Representation of objects and structure within a HTML or XML document.

Business logic: The logic of the web application, regardless of what client is used.

Component: A DOM-node that represents a visual component in a web application.

JSON (JavaScript Object Notation): JSON is a text-based format to show data. An example of a JSON object can be found in listing 1.1. It contains a person object with the key value pairs of `firstName`, `lastName` and `age`. This is supposed to represent a person named John Smith who is 25 years old.

```
{"person":
  {
   "firstName": "John",
   "lastName": "Smith",
   "age": 25
  }
}
```

Listing 1.1. Example of a JSON object

Chapter 2

Background

Chapter 2 includes a brief introduction to SPAs, to software architecture and to frameworks. These are explained and how they are related to web development. In addition, related work to software architecture comparison and framework comparisons are presented.

2.1 Framework

In literature, there are many definitions of software frameworks. Computer scientist, Ralph E. Johnson defines a framework as a reusable design represented by a set of abstract classes. In addition, he calls a framework a skeleton application that can be customized by a developer [33]. Besides being called a skeleton application, a framework may be seen as a 'semi-complete' application [51]. The purpose of the framework is to allow the developer to solve the problems that fall within the domain of the framework [48]. By reusing code the developer enhances the effectiveness and efficiency of innovation which results in higher quality products [40, p. 4].

2.2 JavaScript Languages

JavaScript contains several syntaxes and languages, such as CoffeeScript, TypeScript, JSX and Dart, which can be interpreted by common web browsers. These languages extend JavaScript with new functionality making it easier to read or to use. An example written in TypeScript of a class called HelloWord can be seen in listing 2.1, using the function `getHelloWorld`.

```
1 class HelloWord {
2     getHelloWord() {
3         return "Hello, world";
4     }
5 }
```

Listing 2.1. HelloWorld class written in TypeScript

A web browser cannot interpret TypeScript. However, by compiling TypeScript into JavaScript the browser can execute the code. Listing 2.2 shows the compilation of the HelloWord class from listing 2.1.

```
1 var HelloWord = (function () {
2         function HelloWord() {
3     }
4     HelloWord.prototype.getHelloWord = function () {
5         return "Hello, world";
6     };
7     return HelloWord;
8 }());
```

Listing 2.2. HelloWorld class compiled into JavaScript

In listing 2.2, TypeScript gives the HelloWorld class a C++ or Java-like syntax and removes the need for nested functions.

2.3 Single-Page Application

In this section, the fundamentals of SPAs are described by introducing how they communicate with a server, how SPAs are executed in the browser and how data bindings work.

2.3.1 Definition of a Single-Page Application

It is generally agreed that there is no exact definition of the concept SPA. A. Mesbah and A. van Deursen define it as: "the single-page web interface is composed of individual components which can be updated/replaced independently, so that the entire page does not need to be reloaded on each user action" [41]. To achieve a better understanding, these key attributes are defined further:

Web interface: The interaction between a user and a web server.

Individual components: The SPA is split into smaller and individual components.

Updated/replaced: A component can be updated or replaced by a new component or page.

Reloaded: A typical web page needs to be reloaded, in contrary to an SPA.

User action: A user can make input to an SPA from any I/O device, causing action to occur.

2.3.2 Communication

Figure 2.1 illustrates a typical communication of an SPA between a user and a web server.

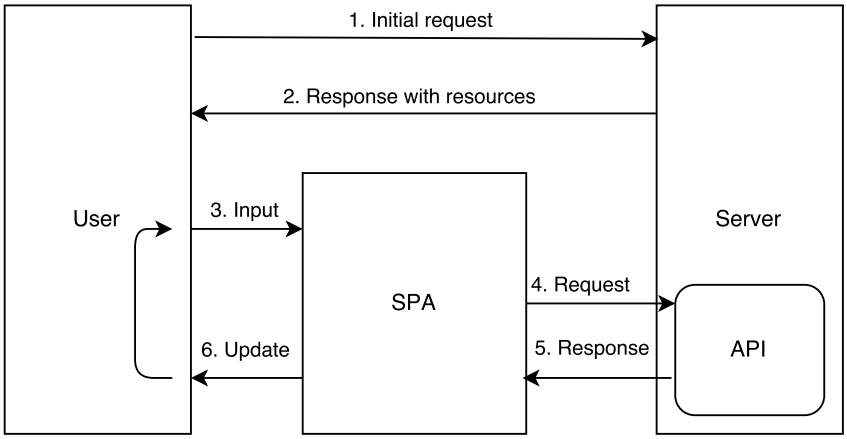

Figure 2.1. Figure of the communication between a user and a web server.

1. **Initial request:** The initial request is often made from a browser on a desktop computer or mobile device. This is done by a HTTP request from the user to a specific URL.

2. **Response with resources:** The HTTP request is handled by the web server. The web server responds by sending the JavaScript dependencies and additional libraries. When the user receives the response from the server, the SPA is executed and loaded into the web browser.

3. **Input:** The user is now able to make input to the SPA causing changes in the state of it. These changes are handled either by the application or by fetching new data via AJAX from the server API.

4. **Request:** The request from the SPA to the API is made asynchronously. The communication between the SPA and the server is usually done with JSON.

5. **Response:** The results are sent back to the SPA as soon as the API can handle the request. Mostly, this is done by sending JSON back to the SPA.

6. **Update:** With the new data received from the server, the SPA updates the components. This update is executed by a re-rendering of the DOM, performed with JavaScript. When the re-render is completed, the SPA is ready for new input from the user. This results in a loop from the third step to the sixth step until the user exits the SPA.

2.3.3 Execution

JavaScript is executed in the browser, although the way it is interpreted may vary between different browsers. This is due to the difference between implementations of the JavaScript engine between web browsers. Mozilla FireFox has an engine called

SpiderMonkey [42] and Google Chrome has V8 [29]. Apple uses JavaScriptCore in their WebKit [21] browser engine for its desktop and mobile browsers. In other words, both performance and support for newer functionality in JavaScript may differ between browsers. Consequently, it is important to test the SPA framework in as many browsers as possible.

2.3.4 Data bindings

There are two different types of data bindings, one-way data binding and two-way data binding. The one-way binding is used in many traditional server-side web applications where a template and one or many data models are merged on the server and sent to the user's view. Any changes to the models or the view are not reflected back to the user after the merge. To update the model, it is necessary for the user to send back the view to the server. These changes must be processed by the server and sent back as a new merge of the template and the models [7].

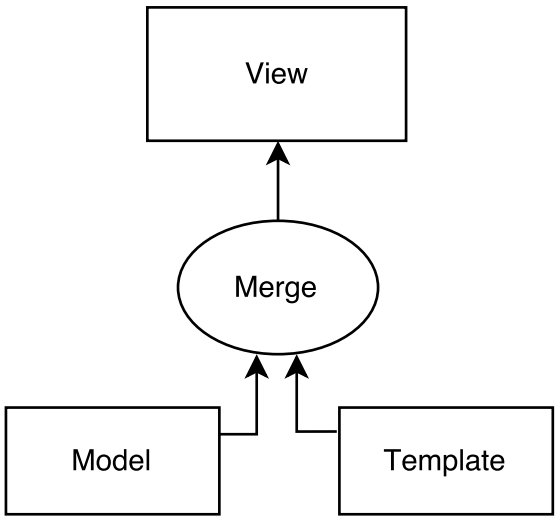

Figure 2.2. Figure of a one-way data binding

The two-way data binding is common in many SPA frameworks where the view can be seen as a "single-source-of-truth" of the data models. All changes performed by the user is instantly reflected on the model and all the changes to the model are propagated into the view [7]. However, since the two-way data binding is two ways, reasoning about how the application might behave is difficult. The developer team at Facebook Inc. state: "We found that two-way data bindings led to cascading updates, where changing one object led to another object changing, which could also trigger more updates." [22] This lead to trouble with their Messenger application in the Facebook ecosystem, and to solve the problem with the two-way data binding they developed a new type of data binding [11] called Flux. It is a one-way data binding with four components instead of three [22].

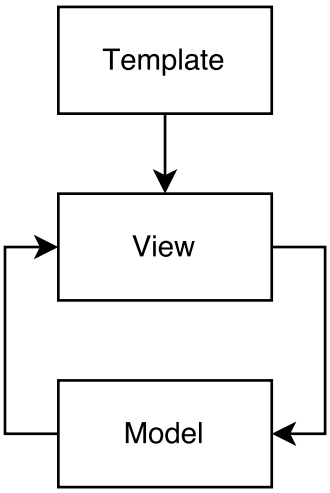

Figure 2.3. Figure of a two-way data binding

2.3.5 States

States have always existed in web applications, but to increase the user interactivity, more states are required. In server-side rendering or one-way data binding, it is difficult to implement smaller changes in the components. These small changes need to retrieve the template and model merge from the server, before changing the user's view. Typically, SPAs have more complex states than traditional server-side applications [56]:

DOM events that cause state changes: I/O into forms, the fields are validated and give feedback to the user.

Application state changes: Interaction with buttons causes a new page to appear.

Global state changes: Going offline in a real time application.

Delayed data from the API: AJAX call is delayed between the SPA and the server.

Data model changes: A data model is changed and an update is sent to the client.

2.4 Software Architecture

This section defines what software architecture is. In addition, a method of how to identify a software architecture is proposed.

2.4.1 Definition of Software Architecture

Software architecture is a process of finding a structured solution that meets all operational and technical requirements of a software project. This involves optimizing common quality attributes, such as security, performance and a wide range of other factors. Each of these factors have an impact on the quality, performance, maintainability and the overall success of the software [57].

In literature it is possible to find a large variety of definitions of what software architecture is. For example, in *Patterns of Enterprise Application Architecture*, Martin Fowler proposes: "The highest-level breakdown of a system into its parts; the decisions that are hard to change; there are multiple architectures in a system; what is architecturally significant can change over a system's lifetime; and, in the end, architecture boils down to whatever the important stuff is." [19].

While Bass, Clements, and Kazman define software architecture in *Software Architecture in Practice (2nd edition)* as: "The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. Architecture is concerned with the public side of interfaces; private details of elements – details having to do solely with internal implementation – are not architectural." [8].

2.4.2 Identify Software Architecture

There exist many methods of describing software architecture, such as 4+1, Agile modeling, IEEE 1471 and UML (Unified Model Language). The aim of all these methods is to give an idea of defining the software architecture in different views. These views can be seen as core parts of a system and all of these methods have their own idea of which views are most important and how they should be presented [58].

In this thesis, the model used to describe software architecture is proposed by Microsoft Research, *Application Architecture Guide, 2nd Edition*. It is an iterative method allowing developers to refine the design over time and even through the life cycle of the software [58]. The method contain these five steps:

1. **Identify Architecture Objectives:** Give a precise specification of the objectives of the architecture. This helps focusing on the right problems in the design.

2. **Key Scenarios:** By identifying key scenarios, focus is brought to what is most important for the architecture. These scenarios can be used to evaluate a candidate architecture.

3. **Application Overview:** The application should reflect reality and the context must be identified. For example, application type, deployment architecture, architectural styles and technologies.

4. **Key Issues:** Based on quality attributes the key issues must be identified. The most mistakes are made in this area when designing an application.

5. **Candidate Solutions:** Create an architecture prototype and evaluate it by the key scenarios, issues and deployment constraints. Then, one more iteration may be done for further refinement of the candidate architecture.

2.5 Related Work

2.5.1 Software Architecture Comparison

Among universities, scientists have developed methods for evaluating and comparing different architectures. SAAM (Software Architecture Analysis Method) is used to evaluate one candidate architecture at a time. Evaluation is performed via scenarios, quality attribute and quality metrics. The quality attributes and metrics are used to define the business value, while the scenario is used to understand the architecture. Different architectural approaches are identified and are evaluated using these scenarios, attributes and metrics [35].

ATAM (Architecture Tradeoff Analysis Method) is similar to SAAM, but this method focuses on finding tradeoffs. For every tradeoff, each architectural approach is evaluated of how this tradeoff may affect it [36].

CBAM (Cost Benefit Analysis Method) is an enhancement of the ATAM. The authors propose ATAM with the addition of finding benefit and cost for each of the proposed architectural approaches [32].

ARID (Active Reviews for Intermediate Designs) is a combination of design reviews and ATAM [12].

SACAM (The Software Architecture Comparison Analysis Method) can compare different candidate architectures. This method extracts quality criteria from the business goals and describes them as scenarios. These scenarios are then evaluated given metrics and predefined architectural views [55].

All of the above mentioned methods is originating from the same research group at Software Engineering Institute at the Carnegie Mellon University.

DoSAM (Domain-specific Software Architecture Comparison Model) is used to compare different architectures given a specific domain. By collecting quality attributes, a domain specific architecture blueprint can be created. This is used as a scheme to show how the architecture is assembled. Quality metrics are collected and the blueprint is evaluated given these metrics [9].

FAAM (Family Architecture Analysis Method) is only able to evaluate information-system family architectures. System quality requirements are taken and evaluated against the proposed architectural approaches [15].

ALMA (Architecture Level Modifiability Analysis) is used to evaluate one architecture candidate. The method includes a set of scenarios and evaluated of how well the architecture is supporting these scenarios [47].

2.5.2 Framework Comparison

This section describes current research in the field of framework comparison.

Tim Malmström proposes a method of how to compare SPA frameworks written in JavaScript. He uses seven requirements taken from interviews. After defining the requirements, he discusses how the frameworks individually fulfill the requirements. Malmström also performs a performance test on two of the frameworks [38].

Anton Gerdessen [20] proposes a method of comparing two Java back-end frameworks. He collects criteria from literature studies and removes the criteria that could not be applied to a web related framework. The criteria are sorted into two domains and Gerdessen creates a theoretical framework, used to compare the two frameworks. Then, he iterates through all the criteria and review the two frameworks side by side. A more thorough analysis is performed with two of the chosen criteria on the two frameworks [20].

Joe Lennon proposes a method of comparing JavaScript frameworks. He uses a set of features and compares how these features are implemented and used within the frameworks. Furthermore, he develops a prototype using these frameworks and performs a code comparison between the prototypes [37].

Maria del Pilar Salas-Zarate et al. present a list of best practices for web development and use these for comparison of web frameworks. Thereafter, the authors implement a prototype using the Lift framework [14].

Ignacio Fernández-Villamor et al. [17] propose a method of how to compare agile web frameworks. They define a "blueprint architecture" containing eight criteria important for a web framework. These criteria are defined using a set of questions that summarize the general features of an agile web framework. This results in a table where the authors use a percentage of how the framework fulfills the question. Then, the authors choose to use weights on each question. For each framework and the strength and weaknesses are revealed by a final discussion [17].

Tont Shan and Winnie Hua [53] propose a list of design principles that all web frameworks should follow and a taxonomy is proposed of how to group different web frameworks written in Java [53].

The method proposed in this thesis are closely related to the theoretical comparison by Ignacio Fernández-Villamor et al. [17] but it also has resemblance to Tim Malmström's [38] way of comparing SPA frameworks. By combining these theoretical and practical approaches was it possible to carry out a scientific comparison of SPA frameworks.

Chapter 3

Methodology

Chapter 3 contains an overview of the methodology used for the interviews, the literature study and the performance measuring. It also includes an explanation of the prototype and tools that are used to measure the performance of the SPA frameworks.

3.1 Overview

In this thesis, a combination of quantitative and qualitative methods is used to develop a way to compare SPA frameworks. The following sections describe how the research has been carried out.

The first step, before proposing a method of how to compare SPA frameworks, is to get an overview of the topic in order to find out whether similar studies have been carried out and find comparative methods. The analysis of the studied methods resulted in a first draft of a comparison method. This method consists of a theoretical and a practical comparison, which are extended upon further in the study.

Secondly, a more comprehensive literature study is conducted to provide a basis for an abstraction of SPA frameworks. This study formed the main source of data that are collected and sorted into different criteria. As suggested by [13, p. 88-90], key concepts and terms need to be extracted. However, the word *criteria* is used instead of *terms*. As a complement to the literature study, ten interviews are held with employees at Decerno, who encounter SPAs on a daily basis. For each criterion, a set of questions is proposed. These questions are formulated after an analysis of general features existing in some of the current SPA frameworks. In addition, questions from other framework comparisons and interviews are selected and placed into the corresponding criteria.

Finally, a testing methodology is chosen and performance tests are performed on the prototypes.

3.2 Interviews

Interviews are often used as a qualitative method as participants are able to elaborate on what they think or feel [16]. The aim of the interviews in this thesis is to gather more information concerning which criteria are important when choosing an SPA framework. Hence, these interviews are held with a more quantitative approach [52]. All participants hold different roles at Decerno, including front-end developer, system developer or project manager. These participants work with SPAs on a daily basis although not all of them are writing code. They are encouraged to speak freely and keywords from their statements are registered. The participants could speak freely without any interruption until they had nothing more to add.

An interview situation may contain personal information that might be exposed in the research. Consent and confidentiality are some of the issues explained by Michael Q. Patton in [45]. To achieve an ethical interview, the participants are introduced to the purpose of this thesis and told how their answers will be used. All the participants in the interviews take part voluntarily and their answers will remain anonymous. After the interview, the notes are presented and the permission to use the data in the thesis is acquired.

3.3 Performance Measuring Methodology

In this thesis a general method for performance testing is used. This method is created by Microsoft Research [39]. It is applied to performance tests of a web application in order to obtain a structure to the testing. The method consists of seven steps:

1. Identify test environment
2. Identify performance acceptance criteria
3. Plan and design tests
4. Configure test environment
5. Implement the test design
6. Execute the test
7. Analyze results, report and retest

In traditional web applications the testing is mostly performed on the back-end service. However, the scope of this thesis only covers the client side of SPAs. The crucial part of an SPA's performance is its data bindings, loading time and resource allocation.

3.3.1 Data Bindings

Typically, a two-way data binding is used in SPA frameworks (section 2.3.4). These bindings can be tested by loading data to the application and see how fast data bindings are created between the view and the models and also reflected to the view. Furthermore, the tests consist of updating the data bindings with new data which shows how quickly data bindings react to changes in the data model.

3.3.2 Loading Time

The SPA is packaged as one or several JavaScript files. During the initial load, the server needs to send the JavaScript files to the user and the SPA must initialize the data bindings. For a typical user, the loading time for an application should not exceed 0.1 - 1 second. Up to 10 seconds is for most users the upper limit, meaning users initial thoughts might be interrupted and they might decide to do something else [44].

3.3.3 Resource Allocation

A smartphone might not have the same amount of memory available as a desktop computer. Therefore, it is necessary to limit the memory load when an SPA is running. There are various ways to control this, such as limiting the amount of data bindings created during the first initial load.

3.4 Tools

To test the data bindings, the benchmarking framework BenchmarkJS is used [2]. The standard approach of doing performance tests is to run a certain test and present the results with a certain margin of error. These tests do not work properly on JavaScript applications since the time can vary from one browser to another. The approach of BenchmarkJS is to repeat the tests until 1% margin of error is achieved. This includes different type of clocks to get a better reliability of the time measurements [10]. Tests for resource allocation and loading time are performed by using the analytics tool provided by Google Chrome.

3.5 Prototype

The prototypes are created by TodoMVC [6] and the code is taken from their GitHub repository [59]. TodoMVC is chosen because it is a minimal example of a web application with basic functionality such as create, read, update and delete data models. Currently, TodoMVC has one or more implementations of each of the most popular SPA frameworks. These implementations are created by experienced developers who utilize the way the framework is designed to be used. The main functionality of the TodoMVC application is to be able to create, read, update and

delete tasks, which can be marked as completed. The application has three main components:

Input field: This text field is used for adding a task. At the bottom of the task list, the task is created as an uncompleted task. This component also consists of a button for marking all tasks as completed or uncompleted.

Task list: This component is a list of tasks. The individual tasks may be marked as completed or uncompleted, in addition, renamed or removed from the list.

Footer: The footer component is the row at the bottom of the list. This contains four buttons and one counter. The counter represents how many uncompleted tasks are left in the list. The buttons "all", "active" and "completed" are sorting options for the list. The "clear completed" button removes all completed tasks from the list. The footer is hidden when there are no tasks in the list.

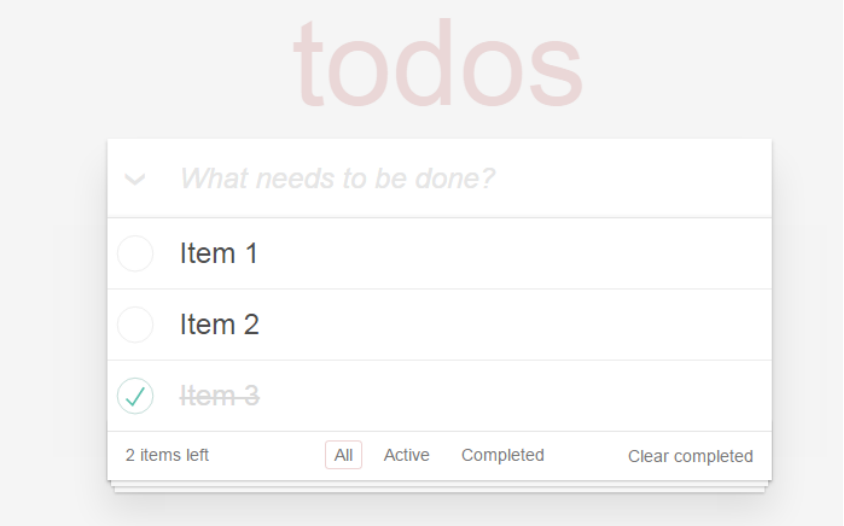

Figure 3.1. Figure of the TodoMVC application

Chapter 4

Formulating a Comparison Method

In chapter 4, the result of the interviews and the literature research are presented. Furthermore, the abstraction of SPA frameworks is presented as criteria and questions. In addition, a few concepts are chosen to further make a comparative method for SPA frameworks.

4.1 Results of the Literature Research

4.1.1 Collecting concepts

When comparing something as complex as a framework, an abstraction is needed. As proposed by [20], an abstraction can be made by creating a "conceptual framework", "blueprint architecture" or "blueprint framework". The abstraction can be built with either a scenario- or a criteria-based approach.

All methods mentioned in section 2.5 are using some of these concepts, which are summarized in table 4.1:

Concept	Reference
Criteria-based comparison	[9, 14, 17, 37, 38, 55]
Scenario-based comparison	[12, 15, 32, 35, 36]
Collect business goals	[9, 36, 55]
Scoring with yes/no	[14, 17, 37]
Scoring with percentage	[9, 17, 55]
Scoring with discussion	[12, 20, 32, 35, 36, 38, 47, 55]
Score is weighted	[9, 17, 35, 55]
Evaluation of one candidate	[12, 15, 32, 35, 36]
Comparing different candidates	[9, 14, 17, 37, 38, 55]
Involve different parties	[12, 15, 36]
Prototyping	[14, 35–38]
Performance testing	[38]
One/many metric/metrics for every criteria	[14, 17, 20, 38]

Table 4.1. Concepts found by literature research.

Conclusions

As seen in table 4.1, all the scenario-based methods can only evaluate one candidate architecture at a time. On the contrary the criteria-based methods are used to compare different candidates. In [9, 36, 55], the authors propose the use of business goals as a base for collecting scenarios. These scenarios are specific and may vary from project to project, thus cannot be used in a general case. Scenarios can fit well in an initial process of development. However, this thesis proposes a method that can be used to compare SPA frameworks, therefore scenarios are not an appropriate selection.

Another way to compare software architecture is to utilize several different professions and external experts. Some of these aspects proposed by [12, 15, 36]:

- Workshops with different development teams to give a broader perspective.
- Involve the whole team in the beginning of the development phase.
- Involve external experts in the design process.
- Presenting the findings for the other teams involved in the design process.

Involving different parties are highly recommended when working in a project. However, the proposed comparative method does not contain any specific parts requiring other participants or specific parties. It is crucial that the same person or persons perform the comparison, since the answers may vary, thus making the actual comparison biased.

The authors behind SACAM, C. Stoermer, F. Bachmann and C. Verhoef state: "Comparing software architectures implies a set of criteria. A comparison without any criteria produces no sound reasoning about a selection." [55]. As stated by

[55], the use of criteria is giving a sound reasoning of how well the architecture or framework performs when compared to another.

Naturally, it is important to identify a number of criteria that are general and essential for SPAs. This thesis is focused on identifying a set of criteria as a fundamental part of the comparison. By defining each criterion and formulating questions as proposed by [14, 17, 37, 38], an abstraction of an SPA can be made. This abstraction is the theoretical part of the comparison. The formulated questions can be regarded as a measure of how well the criteria are fulfilled [14, 17, 20, 38]. To weight the score as proposed by [9,17,35,55] means that either questions or specific criterion can be superior towards another. However, this thesis does not promote any specific way of using weights in either questions or criterion. As an example, Ignacio Fernández-Villamor et al. in [17], are using weights on each question, and calculates a final score for each criterion based on the questions and their weight.

4.1.2 Collecting Criteria

The table below consists of a summary of criteria found in the literature research:

Criteria	**Reference**
Cache performance	[38]
Documentation	[38, 53]
Integration	[53]
Maintainability	[38]
Maturity	[38]
Modifiability	[20]
Modularity	[17, 20, 38]
Performance	[20, 38, 53]
Persistence	[17, 20]
Popularity	[17, 38]
Portability	[20, 37, 38]
Presentation	[17]
Reliability	[14]
Scalability	[20, 53]
Security	[14, 17, 20]
Simplicity	[53]
Testability	[17, 20, 38]
Transparency	[20]
Usability	[14, 17]

Table 4.2. Criteria found by literature research.

4.2 Results of the Interviews

The interviews are completed as described in section 3.2. The attendances are ten employees at Decerno with varying professional roles. The criteria they find important when choosing an SPA framework are summarized in table 4.3.

Criteria	**Frequency**
Cache performance	2
Compatibility	4
Developer guidelines	3
Documentation	6
Does “in-house” experience exists?	2
Simple to use	6
Maintainability	5
Maturity	4
Modularity	3
Performance	5
Popularity	8
Portability	4
Scalability	3
Security	2
Framework’s size	1
Testability	3
Who is the developer?	4

Table 4.3. Criteria found by interviews.

4.3 Criteria Conclusions

To gather criteria, ten interviews and a literature study are conducted independently. Integration, reliability, transparency and presentation are four criteria only occurring once in literature and never in the interviews. These four criteria are only present in literature regarding web frameworks, which is related but differs from what is important for an SPA framework. Therefore, these criteria are excluded from the list. Stability and maturity are related, because software maturity impacts the stability, therefore they are put together as one criterion. To be able to use an SPA framework in as many browsers as possible, compatibility and portability are essential and thus these two criteria are grouped together. In computer science, persistence often relates to keeping the current state of the application in memory. To keep the state of an SPA in memory, good cache performance is required. This results in grouping persistence and cache performance together. Usability can be defined by the ISO 9241-11 standard, which covers the field where a user should be satisfied with the usage of the software [18]. This pairs well with

simplicity, because if something is simple to use it should have a high usability, on the contrary, something less simple to use should have a lower grade of usability. Therefore, simplicity and usability are grouped.

Documentation, size of the framework and in-house experience are chosen as questions to use within their corresponding criteria.

4.4 Criteria Definition

In this section, the criteria found in section 4.1.2 are given definitions and questions to measure to what extent the criteria are fulfilled.

4.4.1 Security

Today, many web applications store user credentials or credit card numbers in a database connected to the Internet. For example, a security breach in the web application, all the user's data can be stolen and the trust of users could be lost. OWASP (Open Web Application Security Project) [4] is a non-profit organization and an online community which main goal is to improve the security of software. One of their most well known contribution is their top 10 list of the most common security issues with web applications.

SQ1: When a security related bug is found, does the framework have a security policy [3]? A security policy is how a security issue is handled by the developers of the framework. This policy can be about responding to the issue by e-mail, who is notified and how long the framework update takes.

SQ2: Does the framework have a promotion for finding bugs [28]? These kind of promotions tend to give money to people who find security bugs in a web application. This makes people more eager to find security related bugs.

SQ3: Does the framework prevent Cross Site Request Forgery [1]? Cross Site Request Forgery is a security issue where the user is not aware of a vulnerability in a web form.

SQ4: Does the framework prevent Cross Site Scripting [5]? Cross Site Scripting is another security flaw in web applications. By utilizing this flaw, malicious JavaScript can be executed in the browser.

4.4.2 Modularity

The purpose of modularity is to give a structure for development of a particular SPA. An example of modularity could be to separate all components. If a component is moved from the application, it should work as an isolated unit and be fully functional. Separation could also mean having a separated view and data layer in the application. This approach makes it easier to change an isolated functionality in a specific component and overall make the SPA easier to maintain.

MQ1: Does the framework support a design pattern that leads to the development of separated layers or components become easier [38]? As mentioned above, separation is important and a framework can aid this by proposing a design pattern that is suitable for a component-based design or layer structure.

MQ2: Does the framework support a component-based approach of development? By having components that work as isolated units leads to better modularity where components can be moved or changed without affecting the rest of the application.

4.4.3 Popularity

Using a popular framework is an important factor for both maturity and security. Much experience is gathered on StackOverflow and similar forums where developers discuss common problems. A large developer and user base can produce more bug reports and more people are likely to be involved in fixing these issues [50].

PQ1: How are the frameworks related to each other on Google Trends [38]? A Google Trends graph can give an indication of how popular a framework is and how the number of searches has increased. If more than one framework is allowed to be displayed, it is possible to compare their popularity progression. This graph gives an overview of which framework that is most popular and how this has changed over time.

PQ2: How many GitHub watchers does the framework repository have [38]? If a framework has many followers on GitHub, a wider audience is able to test a version that still is in development.

PQ3: How many StackOverflow questions does the framework have [38]? Mostly, problems are never unique and by having many questions already answered about the framework could save time.

4.4.4 Maturity and Stability

There is no exact method of how to measure the maturity and stability of software. A CMM (Capability Maturity Model) can be used to measure the maturity of a process of software development within an organization [46]. However, it is difficult to know the processes since these are not publicly available. Instead, maturity and stability can be defined as how widely the framework is used by large corporations.

MSQ1: Is there any system developed by a larger corporation that uses the SPA framework in a production environment? If a large company uses this framework in one of their production system, this can be an indication of the maturity and stability of the framework and their belief in its future.

4.4.5 Simplicity and Usability

It is almost impossible to add new functionality to a software without doing any changes to the code base. By keeping the changes small, some defects can be avoided. In contrast, if a major change is required, the developer needs not to introduce more complexity than necessary. Given a context, the simplicity can vary between the users of the framework. It might be simpler to change the code for the original developer than a new developer. This issue can be solved by some kind of competence sharing, such as documentation or guidelines [34, p. 49-50].

SUQ1: Does in-house experience exist (Table 4.3)? The possibility for success is higher if there already exists knowledge of the framework. Furthermore, experienced staff can teach less experienced colleagues.

SUQ2: Does documentation exist [38], (Table 4.3)? When working with a new framework or expanding an existing application with new functionality, documentation is important to be able to solve problems.

SUQ3: Does any third-party tool provide code generation [17]? Instead of writing repetitive code, third-party tools can generate code and save time for the developer.

SUQ4: Does any IDE support this framework? An IDE (Integrated Development Environment) serves the developer as a support for auto completion of code or API documentation while writing.

SUQ5: Which JavaScript language does the framework promote? There are many different JavaScript versions and there exist a variety of JavaScript-based languages where the aim is to simplify development.

SUQ6: What is the cyclomatic complexity of the prototype? The cyclomatic complexity measurement can be used as an indicator of how complex the code is in an application. Having a low cyclomatic complexity can keep the software more reliable and maintainable [60].

4.4.6 Portability and Compatibility

JavaScript is executed in the browser and it is important that the framework is supported in as many browsers as possible. Equally important is the support for mobile devices, since many people are using their mobile devices for browsing the Internet.

PCQ1: Which browsers are every release tested with [38]? For portability reasons, it is important that all releases are tested in as many browsers as possible.

PCQ2: Which is the current browser versions supported [37, 38]? Most of the users tend to update their browsers more regularly except the Internet Explorer users. Most of the Internet Explorer users are spread out through versions 8, 9, 10, 11 and also Microsoft's new web browser Edge 12 and Edge 13 [54]. It shows the necessity of supporting as many versions as possible.

PCQ3: Which is the earliest browser version supported? Legacy support is another important factor because many users are still not using the latest versions of their current web browser. As seen in PCQ2, the user base of Internet Explorer is more spread out through many versions of the web browser.

PCQ4: Which version of JavaScript (ECMAScript) is the framework built with? Currently, ECMAScript 6 is not fully supported in all browsers. For this reason, all JavaScript that use the new ECMAScript 6 functionality need to be compiled to ECMAScript 5. If the framework is built with ECMAScript 6 or any future version of ECMAScript that is not currently fully supported in all browsers,all browsers, a compilation to the latest version currently supported is needed.

PCQ5: Does the framework support mobile devices? Today, many users browse the Web with their mobile device. Therefore, it is important for a framework to support mobile devices.

PCQ6: Is the framework compatible with other libraries that are required by the application? When developing a web application, it is common to use more than one library or framework. Thus, it is important that the framework is compatible with the libraries or frameworks that are used.

4.4.7 Cache Performance and Persistence

Cache performance and persistence is important in an SPA to enhance the user experience. If the initial loading time is too high, a user might exit the SPA before it is properly loaded. Another reason to have some utilities for boosting cache performance is to reduce the data load from the server to the SPA. Hence, this can save resources on the server and decrease the time the user has to wait.

CPQ1: What is the size of the JavaScript file (Table 4.3)? For a user to be able to use an SPA, the JavaScript files need to be downloaded. A larger file size increases the loading time and places more strain on the server.

CPQ2: Is there any functionality to support less data transfer between the SPA and the server [38]? The framework can save data transfers by caching resources.

4.4.8 Testability

It is important that a framework can be properly tested, even if the developers use a test-driven development method or not. This will ensure that changes do not interfere with old tests or that new functionality can be tested before a new release.

TQ1: Does the framework support unit testing [17,38]? Unit testing allows testing of a module to ensure its functionality.

TQ2: Does the framework support integration testing [17]? Integration testing or end-to-end testing allows testing of the full application.

TQ3: Does the framework support functional testing [17]? Functional testing allows testing of certain logic in the application.

TQ4: Does the framework support performance testing [17, 38]? Performance testing allows testing of parts or the whole application through benchmarking or profiling.

TQ5: Does the framework support mocking of objects [17, 38]? Mocking of objects makes some tests easier to perform when there is not sufficient data or if it is hard to define the data.

4.4.9 Maintainability

In software development, it is common for one team to start the development and another one taking over. For example, having a high level of maintainability is when a new developer can start to fix issues and bugs quickly [49].

MaQ1: Does the framework have any developer guidelines? When working in a large team it is important that every involved developer knows how to work with the framework.

MaQ2: Does experience of this framework exist among programmers [17]? If there exist many developers with knowledge of this particular framework, it reduces the risk of not being able to recruit experienced developers.

MaQ3: Who is the developer of the framework (Table 4.3)? Is the developer a single person, a developer team or a company? A company that develops a framework has professional people that work with it on a daily basis, a single person might not be able to work daily with the framework and might not update the framework as frequently as necessary.

MaQ4: Does the team behind the framework use it in their own production environment? If the developer team behind the framework uses their own framework in a production environment, this can be an indication that they believe in their product and that it is suitable for a production system.

MaQ5: How many lines of code are the prototype written in [50]? An implementation with fewer lines is considered to be more maintainable [50].

Chapter 5

Comparison of SPA Frameworks

Chapter 5 contains a comparison between AngularJS, React and Angular 2, by using the method proposed in Chapters 3 and 4.

5.1 Frameworks

In this section, the frameworks AngularJS, React and Angular 2 are introduced by a short summary.

5.1.1 AngularJS

AngularJS is developed by Google Inc. and the initial release was in 2010. The aim of AngularJS is to extend the HTML vocabulary with data bindings. This is implemented with `ng`-tags, which bind the view to one or many models. [25]. The data bindings in AngularJS are implemented with dirty checking. This means that if a value is bound to the view through a model, it is not immediately updated, instead it is updated when AngularJS does the dirty checking on the value [31]. In addition, AngularJS was created with testability in mind and therefore has built-in dependency injection. This makes it easier to test individual components [26].

Current version

Version 1.5.5. March 18, 2016.

5.1.2 React

React is a framework developed by Facebook Inc. and the initial release was in 2013. It uses a Virtual DOM, where new DOM trees can be created with JavaScript, which creates a simpler programming model. The data bindings in React are implemented with something Facebook Inc. calls *diff algorithm*. This algorithm causes a full re-render of the application every time something changes. To detect the changes in the Virtual DOM, React compares the DOM trees in every level and re-renders

when a change is found [24]. In addition, React has its own JavaScript language called JSX. This provides an extension to JavaScript by adding XML-like syntax that is similar to HTML [23].

Current version

Version 0.14.8. March 29, 2016

5.1.3 Angular 2

Angular 2 is a major rewrite of AngularJS and existing currently in beta. As AngularJS, Angular 2 also utilizes `ng`-tags, but with a different syntax. On the other hand, Angular 2 does not use dirty checking as AngularJS does. Instead Angular 2 uses component trees, where the parents are on a higher level in the tree than their children do. Every component has a change detector class, where the parent can update its children if a change is detected [27]. Angular 2 offers better performance than AngularJS, one of the reasons being that it can load code dynamically during the execution [30].

Current version

Version 2.0.0, beta 17. April 30, 2016.

5.2 Practical Comparison

In this section the practical comparison is made. This is done by presenting the results from the performance tests and a code comparison between the frameworks.

5.2.1 Performance Results

The tests shown in figure 5.1-5.4 are performed with this test setup:

- Windows 8.1 Pro
- Intel Core i7-4790k CPU @ 4.00 GHz
- 16 GB RAM
- Google Chrome 50.0.2661.102m

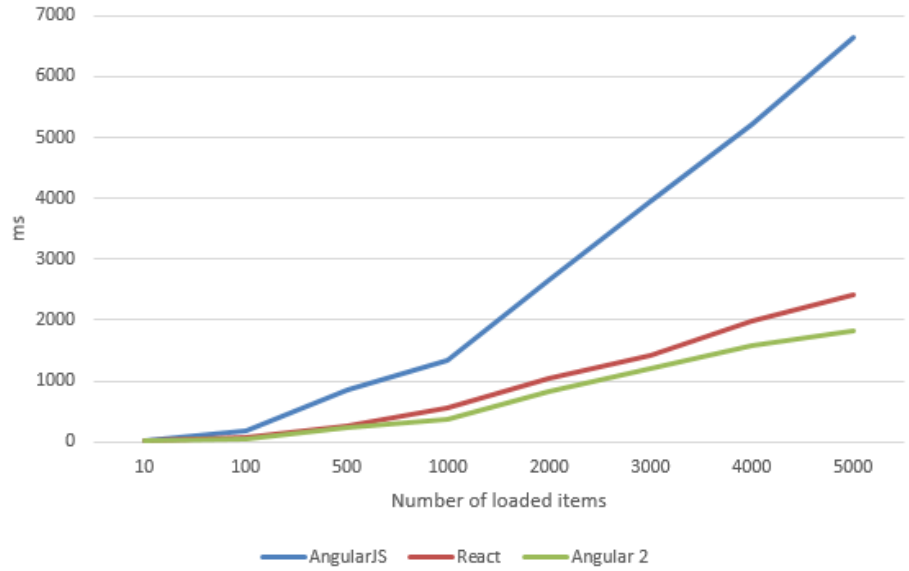

Figure 5.1. Figure of loading time for 10-5000 items in AngularJS, React and Angular 2.

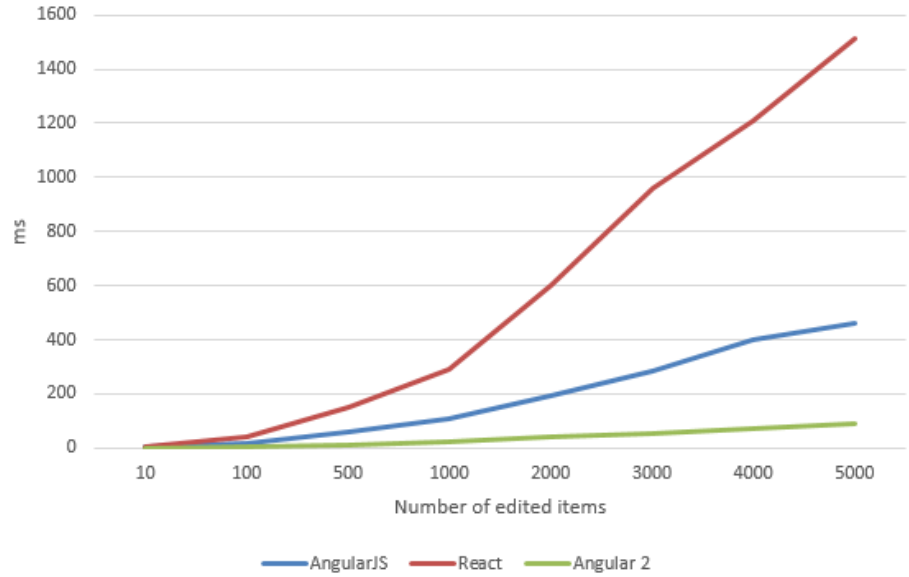

Figure 5.2. Figure of editing time for 10-5000 items in AngularJS, React and Angular 2.

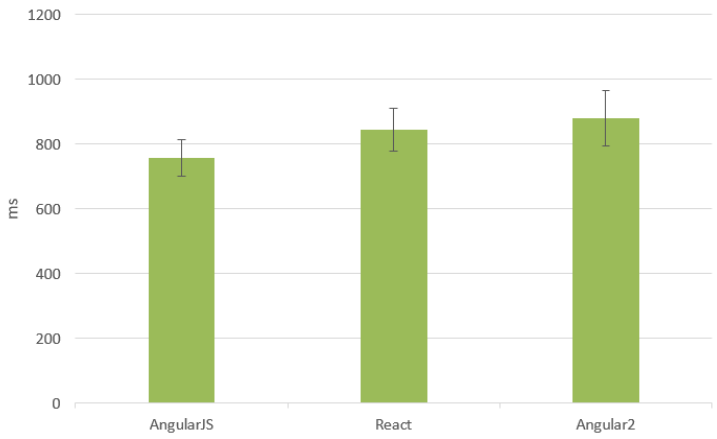

Figure 5.3. Loading time for AngularJS, React and Angular 2.

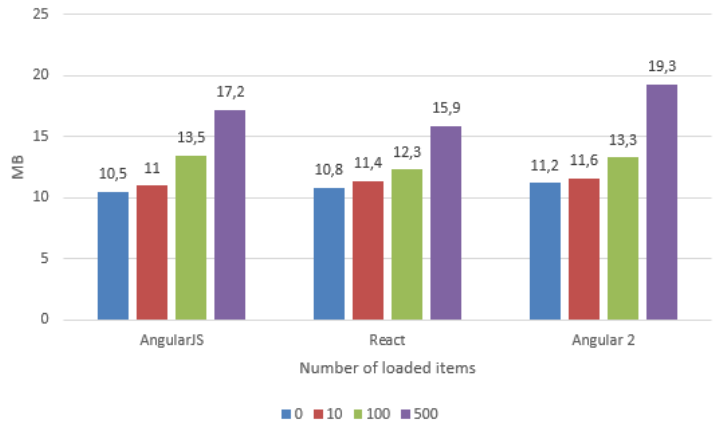

Figure 5.4. Memory allocation when loading 0, 10, 100 and 500 items in AngularJS, React and Angular 2.

5.2.2 Code comparison

In this subsection a code comparison between AngularJS, React and Angular 2 is performed. This comparison consists of comparing the prototype by some of its core features:

- How is the input saved?
- How is the footer hidden and shown?
- How are the task items created and shown in a list?

How the prototype is working can be read in section 3.5.

AngularJS

One of the main goals of AngularJS is to provide an extension to HTML. This is done with ng-tags, which are used within the HTML file to bind the view to the data models with the help of controllers. These controllers are written in JavaScript. In the following examples the controller is called `TC` and holds all the functionality for manipulating the model.

```
1 <form id="todo-form" ng-submit="TC.addTodo()">
2 <input id="new-todo" placeholder="What needs to be done?" ng-model="
    TC.newTodo.title" autofocus>
3 </form>
```

Listing 5.1. Example of how to save the input in AngularJS

As seen in listing 5.1, a data binding between the input field and the model exists. The `ng-model`-tag binds the input field to `TC.newTodo.title`. By binding the title value to `newTodo`, the controller is able to create a new task object by accessing the input field. To submit this value, the `ng-submit`-tag is used. Then, the function `addTodo` within the controller handles the creation of the task.

```
1 <footer id="footer" ng-show="TC.todos.length" ng-cloak>
2 ...
```

Listing 5.2. Example of hiding the footer in AngularJS

To show the footer, a `ng-show`-tag is used, as seen in listing 5.2. A ng-show-tag is used to show the footer when the expression inside the tag is validated to true. `TC.todos.length` is evaluated to true if the task list's length is greater than zero. If the value is zero, the footer is hidden. To avoid flickering while loading the application, the `ng-cloak`-tag is used. This causes the footer to stay hidden until the expression inside the `ng-show` can be validated.

```
1 <li ng-repeat="todo in TC.todos | filter:TC.statusFilter track by
    $index"
2 ng-class="{completed: todo.completed, editing: todo === TC.editedTodo
    }">
3 <div class="view">
4   <input class="toggle" type="checkbox" ng-model="todo.completed">
5   ...
6 </div>
7 ...
```

Listing 5.3. Example of showing a list of tasks in AngularJS

As seen in listing 5.3, the `ng-repeat`-tag is used to iterate over a list of items. A filter is applied so the tasks can be sorted by their status. The `ng-class`-tag extends the HTML class-tag, which can alter the visual style of the HTML. The `ng-class` in this example alters how the item looks like if it is completed or if it is being edited, and adds the HTML-tag accordingly. However, this depends on the state of the task.

React

The React application is separated into four components, the input field, the task list, the task items and the footer. If one component uses another, both are rendered. In this case, the main application is the input field, which uses the list and the footer, and the list using the task items.

```
<h1>todos</h1>
<input
className="new-todo"
placeholder="What needs to be done?"
value={this.state.newTodo}
onKeyDown={this.handleNewTodoKeyDown}
onChange={this.handleChange}
autoFocus={true}
/>
```

Listing 5.4. Input field component in React

```
handleNewTodoKeyDown: function (event) {
  if (event.keyCode !== ENTER_KEY) {
    return;
  }
  event.preventDefault();
  var val = this.state.newTodo.trim();
  if (val) {
    this.props.model.addTodo(val);
    this.setState({newTodo: ''});
  }
}
```

Listing 5.5. Logic for the input field component

In listings 5.4 and 5.5, the input field can be seen, and it is a separate component with its own internal logic. In listing 5.4, the visual elements of the component are written. JavaScript do not support HTML, but with the help of JSX, it is possible to combine JavaScript with a HTML-like syntax. Regarding the logic in listing 5.5, the `onKeyDown` property is triggered when a user presses a key. In this case the `handleNewTodoKeyDown` function is called. `handleNewTodoKeyDown` is a function in listing 5.5. This function checks whether the Enter key is pressed or not. If it is, the value from the input field is saved and a trim function is called on the input. The `addTodo` function handles the addition to the model and the `setState` function resets the state of the input field to an empty string.

```
if (activeTodoCount || completedCount) {
  footer =
  <TodoFooter
  count={activeTodoCount}
  completedCount={completedCount}
  nowShowing={this.state.nowShowing}
  onClearCompleted={this.clearCompleted}
  />; }
```

Listing 5.6. Example of hiding the footer in React

In listing 5.6, the TodoFooter component and its logic is presented. If `activeTodoCount` (the amount of active tasks) or `completedCount` (amount of completed tasks) is larger than zero, the footer is shown and the values from the model is bound to it via JSX. The footer component, `TodoFooter`, uses these values and re-renders the application.

```
var todoItems = shownTodos.map(function (todo) {
  return (
  <TodoItem
  key={todo.id}
  todo={todo}
  ...
  />
  );
}, this);
```

Listing 5.7. Example of a task item in React

```
<ul className="todo-list">
{todoItems}
</ul>
```

Listing 5.8. How to show all task items in React

In listing 5.7, React creates all tasks as separate components `<TodoItem ...>` via JSX. These are created and saved with their properties in a list, `todoItems`. This list is bound to the view with `{todoItems}`, as seen in listing 5.8.

Angular 2

```
<input class="new-todo" placeholder="What needs to be done?"
autofocus="" [(ngModel)]="newTodoText" (keyup.enter)="addTodo()">
```

Listing 5.9. Example of how to save the input in Angular 2

In listing 5.9, the view is able to connect to the model via the `ngModel`-tag. The `keyup.enter` is a function that runs every time the Enter key is pressed. The function `addTodo` is within the controller, controller, alters the model and saves the task. In addition, the `addTodo` function resets the input field by setting the `newTodoText` to an empty string.

```
<footer class="footer" *ngIf="todoStore.todos.length > 0">
...
```

Listing 5.10. Example of hiding the footer in Angular 2

In listing 5.10, the `ngIf`-tag is used in the same way as in AngularJS. If the expression is validated to true, the footer is shown. The `todoStore` is a data structure that makes it possible to save all task items. `todos` is a list that is saved in the `todoStore`. In AngularJS the `ng-cloak` property is necessary to avoid flickering, but this is included in Angular 2's `ngIf`-tag. Therefore, the footer is hidden until the validation can be done.

```
1 <li *ngFor="#todo of todoStore.todos" [class.completed]="todo.
    completed" [class.editing]="todo.editing">
2 <div class="view">
3 ...
4 </li>
5 ...
```

Listing 5.11. Example of showing task items in Angular 2

In listing 5.11, the `ngFor`-tag iterates over all tasks in the list `todos`. For every iteration, a `todo` task is created. The class brackets are used to alter the style of the tasks whether the task is completed, uncompleted or edited.

5.3 Theoretical Comparison

In this section the theoretical comparison is made. All questions are answered and followed by a figure of Google Trends along with conclusions.

Question	AngularJS	React	Angular 2
SQ1: When a security related bug is found, does the framework have a security policy?	Yes, an e-mail address	No	No
SQ2: Does the framework have a promotion for finding bugs?	No	No	No
SQ3: Does the framework prevent Cross Site Request Forgery?	Yes	No	No
SQ4: Does the framework prevent Cross Site Scripting	Yes	Yes	No
MQ1: Does the framework support a design pattern that leads to the development of separated layers or components become easier?	Yes	Yes	Yes
MQ2: Does the framework support a component-based approach of development?	No	Yes	Yes
PQ1: How are the frameworks related to each other on Google Trends?	See figure 5.5	See figure 5.5	See figure 5.5
PQ2: How many GitHub watchers does the framework repository have?	4256	2993	1480
PQ3: How many StackOverflow questions does the framework have?	166994	13692	961
MSQ1: Is there any system developed by a larger corporation that uses the SPA framework in a production environment?	Sony and Google	Instagram and Facebook	None

Table 5.1. Answers to the questions SQ1 to MSQ1

SUQ1: Does in-house experience exists?	N/A	N/A	N/A
SUQ2: Does documentation exist?	Yes	Yes	No
SUQ3: Does any third-party tool provide code generation?	Yes	Yes	Yes
SUQ4: Does any IDE support this framework?	Yes	Yes	No
SUQ5: Which JavaScript language does the framework promote?	ECMAScript 5	JSX	TypeScript
SUQ6: What is the cyclomatic complexity of the prototype?	6	31	6
PCQ1: Which browsers are every release tested with?	Internet Explorer, Firefox and Chrome	Internet Explorer, Firefox and Chrome	Internet Explorer, Firefox and Chrome
PCQ2: Which is the current latest browser versions supported?	Firefox 45, Safari 8, Internet Explorer 11, Chrome 50	Firefox 45, Safari 8, Internet Explorer 11, Chrome 50	Firefox 45, Safari 8, Internet Explorer 11, Chrome 50
PCQ3: Which is the earliest browser version supported?	Same as above, with Internet Explorer 9,10	Same as above, with Internet Explorer 9,10	Same as above, with Internet Explorer 9,10
PCQ4: Which version of JavaScript (ECMAScript) is the framework built with?	ECMAScript 5	ECMAScript 5	ECMAScript 5
PCQ5: Does the framework support mobile devices?	Yes	Yes	Yes
PCQ6: Is the framework compatible with other libraries that are required by the application?	N/A	N/A	N/A

Table 5.2. Answers to the questions SUQ1 to PCQ6

CPQ1: What is the size of the JavaScript file?	154 kB	142 kB	621 kB
CPQ2: Is there any functionality to support less data transfer between the SPA and the server?	Yes, $cachefactory	No	No
TQ1: Does the framework support unit testing?	Yes	Yes	Yes
TQ2: Does the framework support integration testing?	Yes	No	No
TQ3: Does the framework support functional testing?	Yes	Yes	Yes
TQ4: Does the framework support performance testing?	No	No	No
TQ5: Does the framework support mocking of objects?	Yes	Yes	No
MaQ1: Does the framework have any developer guidelines?	Yes	Yes	No
MaQ2: Does experience of this framework exist among programmers?	Yes	Yes	No
MaQ3: Who is the developer of the framework?	Google Inc.	Facebook Inc.	Google Inc.
MaQ4: Does the team behind the framework use it in their own production environment?	Yes	Yes	No
MaQ5: How many lines of code are the prototype written in?	161	493	214

Table 5.3. Answers to the questions CPQ1 to MaQ5

Figure 5.5. Figure of Google Trends between AngularJS, React and Angular 2.

5.4 Comparison Conclusions

By using the theoretical comparison of criteria and questions, as suggested in section 4.4, many similarities can be seen through the majority of the questions, and it is hard to differentiate between the SPA frameworks. However, as seen in table 5.1, the greatest differences can be found within the popularity criteria (PQ1-PQ3). AngularJS has the most watchers on GitHub (PQ2), most questions on StackOverflow (PQ3) and the highest amount of searches on Google (figure 5.5). A possible reason for this popularity may be that AngularJS is the oldest of the three frameworks and has been existing during the up going trend for SPAs in general.

Angular 2 has the largest file size and therefore has the highest initial load time and allocates most resources during runtime (figure 5.2.1). One of the reasons for this could be that Angular 2 is in beta stage and code optimizations are yet to be developed. Furthermore, Angular 2 is the fastest framework to load 10-5000 items and to edit all these items (figure 5.1, 5.2). React is faster than AngularJS when loading items, but the slowest framework to edit items. AngularJS is overall the slowest framework when it comes to loading items. When the application is loading and editing task items, the DOM is changed and therefore the implementation of the data bindings is important. As seen in figure 5.2, React with its diff algorithm is the slowest when editing and AngularJS with its dirty checking is the slowest when new data bindings are created. However, if it takes less than 1 second, the user's initial thoughts are not interrupted as described in subsection 3.3.2. With this in mind, AngularJS can load between 500-1000 items, React, 2000 items and Angular 2 roughly 3000 items, as seen in figure 5.1. A rough assumption could be that a typical SPA has 500-2000 data bindings, which makes all of the three frameworks a suitable choice if trying to stay below the 1-second limit.

AngularJS and Angular2 share a similar syntax. They both utilize ng-tags to extend the HTML with data bindings and data models through controllers. On the other hand, React has different syntax and shares the component-based approach that Angular 2 has. The difference between Angular 2 and React components is that React's components are responsible for their own state and logic and update

when needed. Angular 2 on the other hand uses watchers that check for updates on the data bindings.

By performing this comparison, it is suggested that AngularJS is the most suitable, due to its popularity and stability.

Chapter 6

Discussion and Conclusion

This chapter consists of a discussion about the method and results, proposals for future work and an ethical discussion.

6.1 Summary

To be able to make an abstraction of an SPA framework, criteria and questions are collected. These are chosen based on current frameworks, comparative methods proposed by research and interviews with developers. The abstraction is then used as a base in the theoretical part of the comparison method. However, the practical part does not use the abstraction. It consists of performance tests and code comparison. By using this method AngularJS is suggested to be the most suitable choice due to its popularity and stability even though it suffers from bad performance when creating more than 1000 data bindings.

6.2 Discussion

The first part of the proposed comparison method is the theoretical part. It contains the criteria with the responding questions to evaluate it. In the gathering of criteria, the most important criterion according to the respondents in the interviews was popularity. In contrast, in literature popularity was not considered to have the same importance. In contrast to literature where popularity did not have the same significance. It is easy to believe that there is a great demand from customers on the most popular frameworks. This explains the difference between the literature and the respondents, since literature do not depend on customer's demand. Similarly, the respondents found the criteria "simple to use" and documentation as important, whereas literature did not. A user of a framework might think something is easy to use, while in research it is difficult to find an objective answer whether something is easy to use or not.

The criteria collected in this method are not likely to change in the future, since maintainability, testability and so on have always been important within software

development. The only criterion that might have become more important during the years is security. With more web applications, security is more important than ever. With more money at stake, more hackers are going to try to get a hold of a database or credit card numbers. However, the list of criteria proposed in this thesis are not final and more can be added. Although it is highly unlikely that a new revolutionary criterion will be added. The questions, on the other hand, are likely to change. A paradigm shift may occur, where more emphasis is put on another way of developing, and not using a component or layer based approach. SPAs are prone to two major security issues, Cross Site Scripting and Cross Site Request Forgery, more might be found in the future thus making security a more important criterion.

The proposed method is tested on three frameworks. By only considering the yes and no questions a conclusion of their difference could easily be made. However, taking the other questions into consideration, the answers are more similar for all of the frameworks. It is favorable for frameworks to share complementary similar functionality. Thus, having a code comparison and performance tests is an appropriate part to differentiate between the frameworks. Nevertheless, this can also make a comparison between the frameworks more difficult, as there is a vague distinction between the frameworks. One possible reason for inexplicit difference between the answers could be the lack of criteria or questions. With more criteria and questions, there is a greater possibility for a larger variety of outcomes.

The comparison proposed in this thesis suggests that AngularJS is the best contemporary choice, however it is very likely to change in one or a couple of years. The method cannot look into the future, but as Angular2 and React gain more popularity and become more stable, they are likely to be better choices due to the performance gain. Still, there are a lot of criteria in the proposed method that are difficult to measure, thus the method can be further developed and evaluated to achieve the most accurate comparison.

6.3 Ethical discussion

One of the socio-ethical effects this thesis contributes with, is that it could lead to a more diverse Internet. Today, there are many old web applications where the back-end service is large and very computationally heavy. By building the application as an SPA with a suitable framework, a smaller company or startup with no funding, can place the logic on the client side and thus not needing a large back-end service to serve the users. By having a properly chosen SPA framework it increases the software quality and reduce the development costs. This may create an opportunity where more web applications are being developed. In addition to social aspects, an economic perspective can be seen, where companies can hopefully gain more users having an SPA instead of an old-fashioned website. Aspects concerning the environment is not found.

6.4 Future work

It would be appropriate to compare this method to other SPA comparison methods to see how it performs. However, this comparison is difficult since there is a lack of research within this field. But it is likely to grow, since SPAs are relatively new and is the new trend in web development.

In addition, to get a more diverse outcome of the comparison, more criteria and questions can be collected. This becomes possible by conducting more interviews with developers or other people with a technical role towards SPAs. Another way to improve this method can be by doing a more extensive literature study, however it might take a few years until more research is performed. Alternatively, instead of creating more criteria or questions, the already chosen ones can be further evaluated. As an example, the questions regarding browser compatibility, was this a correct approach since all of the tested frameworks have the same compatibility. There might be a norm in web development that all applications should be compatible with the most current browsers and versions.

Moreover, this method could also be tested with more frameworks than the three chosen in this thesis. If this was to be executed, more variety in the results might be found. However, AngularJS is still one of the oldest SPA frameworks and is one of the most popular. Thus, it is hard for another framework to try to compete with AngularJS.

Bibliography

[1] AngularJS: Security. https://docs.angularjs.org/guide/security, 2016. Accessed: 2016-04-05.

[2] BenchmarkJS. https://benchmarkjs.com/, 2016. Accessed: 2016-04-05.

[3] EmberJS: Security. http://emberjs.com/security/, 2016. Accessed: 2016-04-05.

[4] OWASP. https://www.owasp.org/index.php/Main_Page, 2016. Accessed: 2016-04-05.

[5] React: JSX Gotchas. https://facebook.github.io/react/docs/jsx-gotchas.html, 2016. Accessed: 2016-04-05.

[6] TodoMVC. https://www.todomvc.com/, 2016. Accessed: 2016-04-22.

[7] AngularJS and community. Data Binding. https://docs.angularjs.org/guide/databinding, 2016. Accessed: 2016-04-05.

[8] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edition, 2003.

[9] Klaus Bergner, Andreas Rausch, Marc Sihling, and Thomas Ternité. DoSAM – Domain-specific Software Architecture Comparison Model. In *Proceedings of the First International Conference on Quality of Software Architectures and Software Quality, and Proceedings of the Second International Conference on Software Quality*, QoSA'05, pages 4–20, Berlin, Heidelberg, 2005. Springer-Verlag.

[10] Mathias Bynens and John-David Dalton. Bulletproof JavaScript benchmarks. http://calendar.perfplanet.com/2010/bulletproof-javascript-benchmarks/, 2010. Accessed: 2016-04-05.

[11] Jing Chen. Hacker Way: Rethinking Web App Development at Facebook. https://www.youtube.com/watch?v=nYkdrAPrdcw, 2014.

[12] Paul C. Clements. Active reviews for intermediate designs. Technical Report CMU/SEI-98-TR-008, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2000.

[13] Jill Collis and Roger Hussey. *Business Research: A Practical Guide for Undergraduate and Postgraduate Students*. Palgrave Macmillan, 2009.

[14] María del Pilar Salas-Zárate, Giner Alor-Hernández, Rafael Valencia-García, Lisbeth Rodríguez-Mazahua, Alejandro Rodríguez-González, and José Luis López Cuadrado. Analyzing best practices on web development frameworks: The lift approach. *Science of Computer Programming*, 102:1 – 19, 2015.

[15] Thomas Dolan. *Architecture Assessment of Information-system Families: A Practical Perspective*. Technische Universiteit Eindhoven, 2001.

[16] Rolf Ejvegård. *Vetenskaplig metod*. Studentlitteratur, 2009.

[17] José Ignacio Fernández-Villamor, Laura Díaz-Casillas, and Carlos Iglesias. A Comparison Model for Agile Web Frameworks. In *Proceedings of the 2008 Euro American Conference on Telematics and Information Systems*, EATIS '08, pages 14:1–14:8, New York, NY, USA, 2008. ACM.

[18] International Organization for Standardization. Iso 9241-11:1998, 1998.

[19] Martin Fowler. *Patterns of Enterprise Application Architecture*. A Martin Fowler signature book. Addison-Wesley, 2003.

[20] Anton Gerdessen. Framework comparison method: Comparing two frameworks based on technical domains, focussing on customisability and modifiability. Master's thesis, UvA, University of Amsterdam, 2007.

[21] Apple Inc. JavaScriptCore. http://trac.webkit.org/wiki/JavaScriptCore, 2016. Accessed: 2016-04-22.

[22] Facebook Inc. Overview. https://facebook.github.io/flux/docs/overview.html #content, 2016. Accessed: 2016-04-05.

[23] Facebook Inc. React (Virtual) DOM Terminology. https://facebook.github.io/react/docs/glossary.html, 2016. Accessed: 2016-04-22.

[24] Facebook Inc. Reconciliation. https://facebook.github.io/react/docs/reconciliation.html, 2016. Accessed: 2016-05-22.

[25] Google Inc. AngularJS. https://angularjs.org/, 2016. Accessed: 2016-04-22.

[26] Google Inc. AngularJS: Unit Testing. https://docs.angularjs.org/guide/unit-testing, 2016. Accessed: 2016-04-22.

[27] Google Inc. ChangeDetectorRef. https://angular.io/docs/ts/latest/api/core/ChangeDetectorRef-class.html, 2016. Accessed: 2016-05-22.

[28] Google Inc. Chrome Reward Program Rules. https://www.google.com/about/appsecurity/chrome-rewards/, 2016. Accessed: 2016-04-22.

[29] Google Inc. Chrome V8. https://developers.google.com/v8/, 2016. Accessed: 2016-04-22.

[30] Google Inc. Features & Benefits. https://angular.io/features.html, 2016. Accessed: 2016-04-22.

[31] Google Inc. What are Scopes? https://docs.angularjs.org/guide/scope, 2016. Accessed: 2016-05-22.

[32] Software Engineering Institute. Cost Benefit Analysis Method. http://www.sei.cmu.edu/architecture/tools/evaluate/cbam.cfm. Accessed: 2016-04-05.

[33] Ralph E. Johnson. Components, frameworks, patterns. In *Proceedings of the 1997 Symposium on Software Reusability*, SSR '97, pages 10–17, New York, NY, USA, 1997. ACM.

[34] Max Kanat-Alexander. *Code Simplicity.* O'Reilly Media, 2012.

[35] Rick Kazman, Len Bass, Mike Webb, and Gregory Abowd. SAAM: A Method for Analyzing the Properties of Software Architectures. In *Proceedings of the 16th International Conference on Software Engineering*, ICSE '94, pages 81–90, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[36] Rick Kazman, Mark Klein, and Howard Lipson Jeromy Carriere Mario Barbacci, Tom Longstaff. The Architecture Tradeoff Analysis Method. Technical Report CMU/SEI-2000-TN-009, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1998.

[37] Joe Lennon. Compare JavaScript frameworks. http://www.ibm.com/developerworks/library/wa-jsframeworks/, 2010. Accessed: 2016-04-05.

[38] Tim Johan Malmström. Structuring modern web applications : A study of how to structure web clients to achieve modular, maintainable and longlived applications. Master's thesis, KTH, School of Computer Science and Communication (CSC), 2014.

[39] J.D. Meier, Carlos Farre, Prashant Bansode, Scott Barber, and Dennis Rea. *Performance Testing Guidance for Web Applications: Chapter 1 – Fundamentals of Web Application Performance Testing.* Microsoft Press, 2007.

[40] J.D. Meier, Carlos Farre, Prashant Bansode, Scott Barber, and Dennis Rea. *Reusing Open Source Code*. Microsoft Press, 2011.

[41] Ali Mesbah and Arie van Deursen. Migrating multi-page web applications to single-page ajax interfaces. In *Software Maintenance and Reengineering, 2007. CSMR '07. 11th European Conference*, pages 181–190, March 2007.

[42] Mozilla. SpiderMonkey. https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey, 2016. Accessed: 2016-04-22.

[43] San Murugesan. Understanding web 2.0. *IT Professional*, 9(4):34–41, July 2007.

[44] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[45] Michael Quinn Patton. *Qualitative Research & Evaluation Methods*. SAGE Publications, 2002.

[46] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber. Capability maturity model, version 1.1. *IEEE Softw.*, 10(4):18–27, July 1993.

[47] Jan Bosch PerOlof Bengtsson, Nico Lassing and Hans van Vliet. Architecture-level modifiability analysis (alma). *Journal of Systems and Software*, 69(1–2):129 – 147, 2004.

[48] Dirk Riehle. Framework design a role modeling approach. Diss. ETH No. 13509, 2000.

[49] A. Frederick Rosene, James E. Connolly, and Kenneth M. Bracy. Software maintainability - what it means and how to achieve it. *IEEE Transactions on Reliability*, R-30(3):240–245, Aug 1981.

[50] Ioannis Samoladas, Ioannis Stamelos, Lefteris Angelis, and Apostolos Oikonomou. Open source software development should strive for even greater code maintainability. *Commun. ACM*, 47(10):83–87, October 2004.

[51] Douglas C Schmidt and Frank Buschmann. Patterns, frameworks, and middleware: Their synergistic relationships. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, pages 694–704, Washington, DC, USA, 2003. IEEE Computer Society.

[52] Meg Sewell. The use of qualitative interviews in evaluation. http://ag.arizona.edu/sfcs/cyfernet/cyfar/Intervu5.htm, 2016. Accessed: 2016-04-05.

[53] Tony Shan, Wachovia Bank, and Winnie Hua. Taxonomy of Java Web Application Frameworks. In *e-Business Engineering, 2006. ICEBE '06. IEEE International Conference on*, pages 378–385, Oct 2006.

[54] StatCounter Global Stats. Top 12 Desktop Browser Versions Combining Chrome and Firefox (5+) from Apr 2015 to Apr 2016. http://gs.statcounter.com/#desktop-browser_version_partially _combined-ww-monthly-201504-201604, 2016. Accessed: 2016-05-02.

[55] Christoph Stoermer, Felix Bachmann, and Chris Verhoef. SACAM: The Software Architecture Comparison Analysis Method. Technical Report CMU/SEI-2003-TR-006, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2003.

[56] Mikito Takada. Single page apps in depth. http://singlepageappbook.com/goal.html, 2016. Accessed: 2016-04-05.

[57] Microsoft Patterns & Practices Team. Microsoft Application Architecture Guide, 2nd Edition. Chapter 1: What is Software Architecture? https://msdn.microsoft.com/en-us/library/ee658098.aspx, 2009. Accessed: 2016-04-05.

[58] Microsoft Patterns & Practices Team. Microsoft Application Architecture Guide, 2nd Edition. Chapter 4: A Technique for Architecture and Design. https://msdn.microsoft.com/en-us/library/ee658084.aspx, 2009. Accessed: 2016-04-05.

[59] TodoMVC. TodoMVC GitHub Repository. https://github.com/tastejs/todomvc, 2016. Accessed: 2016-05-22.

[60] Arthur H. Watson and Thomas J. McCabe. Structured testing: A testing methodology using the cyclomatic complexity metric. Technical report, Computer Systems Laboratory, National Institute of Standards and Technology. http://www.mccabe.com/pdf/mccabe-nist235r.pdf.