

# TrustSECO: An Interview Survey into Software Trust

Universiteit Utrecht



**Universiteit Utrecht**

Floris Jansen - 6002919

Dr. R.L. Jansen

Dr. G.C. van de Weerd

Department of Information and Computing Sciences

January 2021

# Abstract

The software ecosystem is a trust-rich part of the world. Collaboratively, software engineers trust major hubs in the ecosystem, such as package managers, repository services, and programming language ecosystems. This trust, however, is often broken by vulnerabilities, ransomware, and abuse from malignant actors.

But what is trust? In this paper we explore, through twelve in-depth interviews with software engineers, how they perceive trust in their daily work. From the interviews we conclude three things. First, software engineers make a distinction between an adoption factor and a trust factor when selecting a package. Secondly, while in literature mostly technical factors are considered as the main trust factors, the software engineers in this study conclude that organizational factors are more important. Finally, we find that different kinds of software engineers require different views on trust, and that it is impossible to create one unified perception of trust.

Keywords: software ecosystem trust, empirical software engineering, Trust-SECO, external software adoption, cross-sectional exploratory interview analysis, trust perception.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Framework</b>	<b>5</b>
2.1	Trust . . . . .	5
2.2	Bias . . . . .	6
<b>3</b>	<b>Research Methods</b>	<b>7</b>
3.1	Literature study . . . . .	8
3.2	Interviews . . . . .	9
<b>4</b>	<b>Results</b>	<b>11</b>
4.1	Literature study . . . . .	11
4.1.1	Technical adoption factors . . . . .	11
4.1.2	Human/organizational adoption factors . . . . .	12
4.1.3	Economic adoption factors . . . . .	13
4.1.4	Trust factors . . . . .	14
4.2	Interviews . . . . .	16
4.2.1	Section 1, adoption factors and selection procedure . . . . .	16
4.2.2	Section 2, Trust factors and metrics . . . . .	21
4.2.3	Section 3, Personal bias and experience . . . . .	23
<b>5</b>	<b>Discussion</b>	<b>24</b>
<b>6</b>	<b>Conclusion</b>	<b>25</b>
<b>7</b>	<b>Appendix</b>	<b>26</b>
7.1	Literature review . . . . .	26
7.1.1	Article list SLR . . . . .	26
7.2	Interviews . . . . .	30
7.2.1	Informed consent . . . . .	30
7.2.2	Interview layout . . . . .	32
7.2.3	Interview protocol . . . . .	33
7.3	Adoption factors . . . . .	34
7.3.1	Technical adoption factors . . . . .	34
7.3.2	Organizational adoption factors . . . . .	35

7.3.3	Economical adoption factors . . . . .	36
7.4	Trust factors . . . . .	37
7.4.1	Technical trust factors . . . . .	37
7.4.2	Organizational trust factors . . . . .	37
7.5	Trust metrics . . . . .	38
7.5.1	Technical trust metrics . . . . .	38
7.5.2	Organizational trust metrics . . . . .	39
7.6	Odyssey Momentum . . . . .	40

# Chapter 1

## Introduction

Software engineers use software packages for creating new solutions from different package managers. [Mojica et al., 2014] [Nguyen et al., 2020]. There is a significant amount of implicit trust in these packages. While the packages could easily be compromised, software engineers assume that as the package comes from a reliable source, it must be trustworthy. This is not always the case [Duan et al., 2020]. Moreover, there are several attack vectors that can compromise a package like registry exploitation of typo squatting [Hou F, 2020].

Before the factors that constitute trust in software packages and package repositories are looked upon, one must look at how software engineers choose software packages and how trust is gained. Moreover, what the impact factors are that influence this. The TrustSECO project aims to uncover all the factors that influence the trust that software engineers have in software packages. In order to uncover these impact factors, a survey will be developed that has as its main aim to uncover how software engineers perceive trust. [Vargas et al., 2020] has done similar research and found 26 factors that influence the selection process of software packages. Rather than looking at the whole selection process, this research will just focus on the trust aspect of that selection process. This will be done by analyzing existing literature and the results of cross-sectional interviews with experts. This information will then be the basis for a large scale survey. This research thus sets out to find what trust factors influence the decision to choose software packages.

## Chapter 2

# Framework

### 2.1 Trust

One of the most important aspects of collaboration is trust [Bunduchi, 2013]. Moreover, trust creates the basis in decision making for the usage of long term product use. [Cho et al., 2015]. Using external software is just that, collaboration. In order to find out what factors induce trust in packages. One must take a look at the term trust. The term is widely used in computer science and has many different definitions across the spectrum. [Artz and Gil, 2007].

There has been a lot of research on this subject that lead to the following three most general and common definitions of trust [Artz and Gil, 2007].

- “Trust is a subjective expectation an agent has about another’s future behavior based on the history of their encounters.” [Mui et al., 2002]
- “Trust is the firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context.” [Grandison and Sloman, 2000]
- “Trust of a party A to a party B for a service X is the measurable belief of A in that B behaves dependably for a specified period within a specified context (in relation to service X).” [Olmedilla et al., 2006]

The first definition is a reputation based one because it concerns the producer of the software rather than the software itself. While the producer is highly relevant in gaining trust in a software package. The producer of the package will not be more important than the product or service itself. This is because this research looks at what factors induce trust in software packages, not at the factors that induce trust in the entities that develop the software.

The second definition is a definition that suits this research better since it concerns the belief in the competence of the software. Therefore the characteristics for the software have to ensure that it acts dependably, securely and reliably. These characteristics will induce a list of factors that ensure that the system acts this way.

The third and final definition of trust concerns a service from a party to another party. This trust also comes from the belief in the product or service, rather than the party that produces this. Therefore this definition of trust will also suit this research.

The definition of trust that will be leading for this research is:

*“Trust of a party A to a party B for a service X is the measurable belief of A in that B behaves dependably for a specified period within a specified context (in relation to service X).”*

For this definition specifies the collaboration of two parties regarding a specific service or product. Thus the trust influence of the developer is not discarded while the focus is on the product, as is the scope for this research.

## 2.2 Bias

Bias is a form of error that can affect research. [Sica, 2006]. For this research, this may occur in the form of selection bias. Selection bias may occur on the participants in the interview if they do not represent the general ideas and thoughts of the average participant. [Hernán et al., 2004] Therefore it is important that the participants have different perspectives and roles on the same matter. This thus leads to participants in different organizations across different fields across different jobs and functions.

Another more intuitive form of bias present will be when several trust factors will be discussed in the interviews, personal bias. After all each participant has their own personal experiences with packages. Minimizing this bias will be next to impossible, since the participant might not even be aware of this bias. However this personal bias will be decreased by first letting the interviewee answer some more technical impersonal questions before diving into the personal part.

## Chapter 3

# Research Methods

To find what factors influence trust in software packages, one must first find all the adoption factors that are considered when making such decision. After this, a selection can be made of the factors that actually contribute to trusting a package. In order to explore the current state of impact factors already present in literature, a literature study will be conducted. These results will be compared with the results from the cross-sectional interviews with software engineers to complement this current literature. This process is shown in figure 3.1.

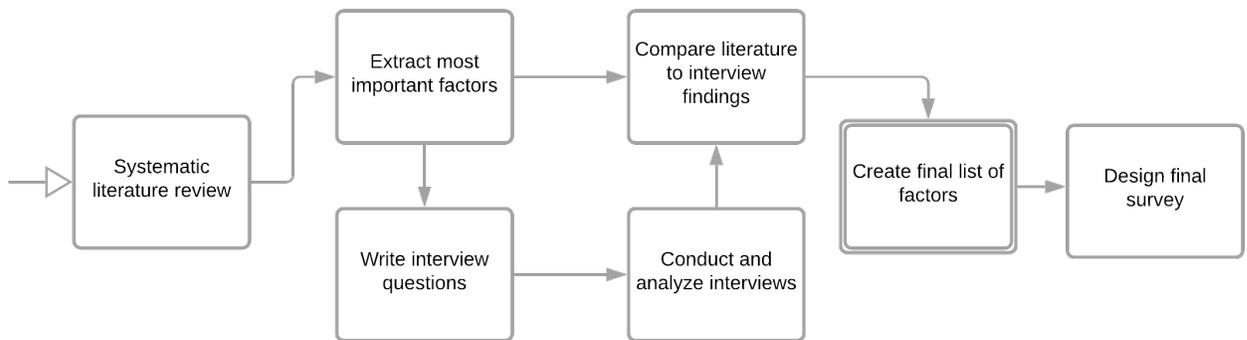


Figure 3.1: Research process

### 3.1 Literature study

The literature study will be based on the SLR done by the TrustSECO team. This SLR will try to uncover the already existing factors in literature. Various search queries will be entered in the following search engines:

- Google scholar
- IEEE Xplore
- ScienceDirect
- Jstor

Figure 3.2 illustrates all the used search queries. By systematically using all the combinations as search queries, all the relevant literature is found. This relevant literature will be stored and analyzed for trust factors.

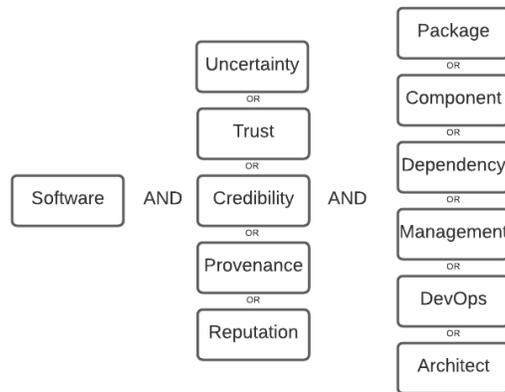


Figure 3.2: Search terms

The combination of these search terms will result in a list of articles. This list will be narrowed down through exclusion and inclusion rules. These are as follows:

- Literature should be about open source software
- Literature should list at least one impact factors on adopting open source software or one factor for gaining trust in a package
- Literature should be public and accessible

The last round of elimination will be done through abstract analysis of the articles. This will result in a final list of articles to be analyzed. These articles can be found in appendix 7.1.1. These articles will be scoured for adoption and trust factors and will be categorised as follows:

- Technical factors
- Organizational factors
- Economic factors

The categorization is based on [Vargas et al., 2020] which holds an explanation as to why a certain factor is categorized as such. Technical factors are factors related to the release process, code quality attributes and the functionality. Organizational factors concern the individual perception, community around the project and other aspects of the organization where the package is developed. The economical aspects cover the financial aspect of package selection like licences, total cost of ownership and risks.

## 3.2 Interviews

The literature review will provide a solid foundation of knowledge to start creating interview questions. Interviews can be used to get detailed personal experiences and thought processes in the selection process [Hiller and DiLuzio, 2004]. These semi-structured interviews shall be conducted with software engineers, DevOps, architects and other experts (see table 3.1) The choice for semi-structured is made since this gives the freedom for the interviewee to really elaborate on personal experiences. In addition, this allows the interviewer to create a new line of questioning based on those experiences [Hove and Anda, 2005]. These interviews are held in English or Dutch. Since the literature provided English factors, a list of translated terms will be provided to ensure that the interviews in Dutch will not yield different results because of difference in understanding in certain terminology across these languages. Prior to the interview, an interview protocol is created based on [Jacob and Furgerson, 2012].

These semi-structured interviews are a form of exploratory research. The subject of exploratory research has been named by among others [Glaser and Strauss, 2017] with the discovery of grounded theory. The goal of exploratory research has been to form hypotheses rather than the testing of hypotheses [Kothari, 2004]. This is the case for this research since the objective is to define a hypothesis describing which factors lead to trusting a software package.

Each interview discusses roughly 9 questions regarding software trust. In order to ensure the participants have the required knowledge to answer the interview questions, certain standards have to be met, namely:

- The participant needs to speak English or Dutch fluently
- The participant has to have been involved with the selection process of software packages
- The participant needs at least three years of relevant working experience

Nr	Organisation	Sector	Size	Function/role	Experience
P1	Triodos Bank	Banking	1000+	Software Engineer	9
P2	Bol.com	E-commerce	1500+	Product Owner	6
P3	Universiteit Twente	Education	3000+	Tech Product owner	20
P4	Keylane	Insurance	400+	Software Engineer	7
P5	Xebia	IT Consultancy	300+	DevSecOps	26
P6	BOC Group	IT Consultancy	200+	Web developer	5
P7	Channable	Marketing	100+	DevOps	5
P8	Ministry of Defence	Military	3000+	Software Engineer	12
P9	NOS	News	600+	Software Engineer	7
P10	Gemboxx	Software dev	10+	Software Engineer	5
P11	Sogeti	Software dev	2500+	Software Architect	23
P12	Grasple	Ed-sec	10+	Software Engineer	16

Table 3.1: Participant overview

The responses give insight as to why certain packages are chosen and the factors that contributed to this decision. Moreover to answer:

### How do software engineers develop trust in a package?

In order to answer this question, the interviews must first provide answer to the following questions:

- What factors are important when selecting external software packages, what is the protocol?
- Which of the selecting factors contribute to gaining trust in a package?
- How do personal aspects influence trust in packages?

The first and second sub question may seem alike. However the literature review has shown that the factors that induce trust are a subset of the factors that influence the choice of a package. These sub questions will each be answered through a section in the interview. Each section contains the actual interview questions that the interviewee will be answering. Section one will contain Q1...Q4, section two will contain Q5 and Q6 and section three will contain Q7...Q9. The full interview structure can be seen in appendix 7.2.2.

These interviews are recorded and transcribed. After transcript approval from the participant a proper analysis will conclude the most important trust factors. This will create a list of factors that have quotes to back them up.

# Chapter 4

## Results

### 4.1 Literature study

The literature study will first take a look at what factors influence the adoption of certain software packages, these will be categorized based on the research of [Vargas et al., 2020]. What follows is an analysis of which of those factors actually lead to more trust in a software package.

[Sánchez et al., 2018] was found during the SLR. This research describes a massive systematic literature review that was conducted to find the most important adoption factors when choosing open source software packages. This research will be setting the stage for the first part of the literature review. The factors will be discussed briefly to provide context and eventually be analyzed to see if they also induce trust.

#### 4.1.1 Technical adoption factors

The technical factors are encountered in literature very frequently. These factors are concerning the technical aspects of software packages and are found in 49 of 54 pieces of literature. Impact factors such as compatibility with software, reliability usability and customization are the most common and thus have the highest importance. A brief description of the impact factors found according to [Wheeler, 2011] to get a good understanding how important certain factors can be:

- **Compatibility:** This impact factor refers to the degree to which a piece of software integrates with existing software. Also whether or not additional programs are required to adopt this piece of software.
- **Reliability:** This factor measures if the software gives the wanted answers. This could be compared to availability. It is not a quantitative property and thus hard to measure.

- **Usability:** This describes how intuitive the program is for the user. This impacts the difficulty of the software to learn. When software is very usable it will be adopted faster.
- **Customization:** Customization is the degree to which a component can be changed to do something it could not do before. Configure changes to its initial configuration.
- **Documentation:** This refers to the available qualitative and quantitative documentation on a software package, this impact factor also falls under the 'support' impact factor, however it is heavily discussed in literature since it describes the technical capabilities of a package.
- **Re-usability:** Re-usability concerns the quantity of actual code that can be reused in different fashions than the particular library uses it for. The more general a piece of software is, the more goals it can serve.
- **Triability:** This factor describes the ease of implementation in a system. When this is the case, several other factors can be tested quite easily and thus benefit the package.
- **Portability:** This is the least mentioned factor and thus carries the least importance of the technical factors. It concerns the ease to deploy the package in multiple different systems.

Technical adoption factors	Mentioned number of times
<i>Compatibility</i>	34
<i>Reliability</i>	23
<i>Usability</i>	17
<i>Customization</i>	17
<i>Documentation</i>	12
<i>Maintainability</i>	12
<i>Re-usability</i>	8
<i>Triability</i>	9
<i>Portability</i>	6

Table 4.1: The 9 technical factors found in literature according to [Sánchez et al., 2018]

#### 4.1.2 Human/organizational adoption factors

This second category of factors concerns the organizational aspects of the entity. This category of factors did also have a dominant presence in literature. Overall this was a bit less than the technical factors however, this category does contain the most important impact factor: "support". This was named 45 times out of the 54 pieces.

- **Support:** This impact factor also contains 'technicalities' like documentation and release frequency. It is a very broad impact factor since it contains a variety of factors that express the need to have a backup plan when the organization does not have the technical skills to solve certain problems. This ranges from documentation to the community.
- **Training:** A presence of sufficient training material ensures that technical staff can learn to fix problems themselves. It also ensures the users know the package is easy to learn and does not have to figure out the technical details themselves.
- **Top management support:** Some organizations' strategy can also impact the decision in which packages to adopt. The users select packages based on policies that the company has, e.g. privacy policy or information protection. [Vargas et al., 2020]
- **Attitude towards change:** This factor describes how employees are looking towards the adoption of new packages.
- **Case studies of FLOSS adoption:** This factor describes the success of implementing new software packages. When a package is widely adopted it gains in reputation. Hence it can influence the decision process.
- **Time adoption:** This factor concerns the total time it takes to implement a package. The longer it takes to fully adapt to new software the less appealing it is.
- **Centrality IT:** This factor describes the dependency of the organization to the new software. Since the proper implementation of more important systems is way more urgent less important systems.
- **Business process engineering:** This describes "when an organization is changing its internal business processes due to any particular circumstance(e.g. quality improvement, organizational restructure)." This factor is only mentioned once throughout all literature and makes it the least named and thus least important.

### 4.1.3 Economic adoption factors

The last category of factors concerns the evaluation of economic factors. These factors are have the least dominance in literature however are still important for software practitioners to look upon.

- **Total cost of ownership:** Total cost of ownership contains all the costs related to the software package. This includes licensing, however also includes operational and support costs.

Organizational adoption factors	Mentioned number of times
<i>Support</i>	45
<i>Training</i>	25
<i>Vendor locks-in</i>	13
<i>Top management support</i>	10
<i>Attitude</i>	6
<i>Centrality IT</i>	3
<i>Time of adoption</i>	2
<i>Case studies</i>	2
<i>Business process re-engineering</i>	1

Table 4.2: The 9 Organizational factors found in literature according to [Sánchez et al., 2018]

- **Licensing cost:** Is a part of the total cost of ownership. This is the main cost that practitioners think about when discussing software costs, and concerns the cost of obtaining a particular license.
- **Operational cost:** This consists of three aspects: the cost to change from systems, the maintenance of the solution and the costs to implement the solution.
- **Support cost:** These are the costs related to external support as well as keeping the system updated. This is only referenced twice in literature and thus not a very influencing factor.

Economical adoption factors	Mentioned number of times
<i>Total cost of ownership</i>	10
<i>License cost</i>	16
<i>Operational cost</i>	4
<i>Support cost</i>	2

Table 4.3: The 4 economical factors found in literature according to [Sánchez et al., 2018]

#### 4.1.4 Trust factors

The factors that influence the adoption of open source software may differ from the factors that induce trust. Factors like functionality and reliability are constantly ranked as high for building trust, next to some other technical and organizational factors. [Del Bianco et al., 2011]. However it seems that the trustworthiness of a system can supersede other adoption factors. [Bernstein, 2005].

The research done by [Del Bianco et al., 2011] provides a list of factors that are believed to affect trustworthiness the most according to it's interviewees.

The two most important factors are "Reliability" and "Degree to which an OSS product satisfies/covers functional requirements the most". These factors are the two biggest technical impact factors in this research.

<b>Technical trust factor</b>	<b>Rank</b>
<i>Reliability</i>	8
<i>Alignment with software</i>	8
<i>Interoperability</i>	7
<i>Maintainability</i>	6
<i>Standard compliance</i>	6
<i>Performance</i>	5
<i>Usability</i>	5
<i>Security</i>	5
<i>Portability</i>	4
<i>Reusability</i>	4
<i>Modularity</i>	4
<i>Standard architecture</i>	4
<i>Human interface</i>	3
<i>Complexity</i>	2
<i>Patterns</i>	2
<i>Self-containedness</i>	2
<i>Size</i>	1

Table 4.4: The technical attributes ranked on inducing the most trust according to [Del Bianco et al., 2011]

The most important adoption factor for this research was support. However on the list of trustworthiness "Short-term support" is number fourteen on the list. This illustrates that even though for adoption it is the most important factor, for trusting certain software the technical factors are more important.

Another remarkable difference between adoption and trustworthiness factors is that the economic factors are mentioned 12(licence) and 13(total cost of ownership) times out of the 31 articles. Yet in the list of trustworthiness factors there is only one economical factor present out of the 37 named trust factors. That is also why there is no table for economical trust factors.

On the contrary, a factor that is in the second highest scoring group for trustworthiness is customer "Satisfaction". While it is trivial that this factor is important for most businesses, it is not explicitly named in literature about adopting open source software. This could be since it is often used to create a solution for customers rather than being the solution itself.

<b>Organizational trust factor</b>	<b>Rank</b>
<i>Documentation</i>	7
<i>Mid-/long existent community</i>	6
<i>Community experience</i>	5
<i>Short-term support</i>	5
<i>Availability of support tools</i>	4
<i>Environmental issues</i>	4
<i>Availability of best practices</i>	3
<i>Programming language uniformity</i>	3
<i>Training</i>	2
<i>Benchmarks/ test suites</i>	2
<i>Organization</i>	2
<i>Reputation</i>	2
<i>Distribution channel</i>	1

Table 4.5: The organizational attributes ranked on inducing the most trust according to [Del Bianco et al., 2011]

## 4.2 Interviews

The interview protocol is divided into 3 sections. These sections will be discussed separately and will answer the sub question set for that section.

### 4.2.1 Section 1, adoption factors and selection procedure

Section 1 of the interview consists of two separate parts containing four questions. The answers to these questions aim to answer the following sub question: What factors are important when selecting external software packages, what is the protocol?

#### Selection procedure

In order to get all the relevant information from the interviewees. The focus of the first section of the interview was to find out the procedure when selecting other packages, as well as finding out all the relevant factors. This procedure is highly relevant because it describes what role a developer has when selecting packages and the processes they have to go through. This sets the stage on their perspective on this matter. When asked about the procedure, all participants somehow mentioned that this was dependant of the situation. Important and sensitive project require delicate measures and a more sophisticated procedure. Where less important projects would not require these. After that, the participants were asked to describe the typical procedure when selecting packages in their company. In some occasions there was a formal protocol when selecting packages (P1, P8, P12). These procedures stated the responsibilities of certain actors within the company and delegated roles. These procedures were set be-

cause of the impact a vulnerability could lead to. P8: *"So if we are want to use a package it first arrives in the 'demilitarized-zone', then we conduct our analysis. Both by hand and with automated systems, and they tell us whether or not it meets all the set requirements."*

A more common practice amongst the interviewees is an informal procedure. (P2, P3, P5..P7, P9..P11). This was not a clear set of rules and responsibilities. However there is a common practice when it comes to selecting packages. The most important aspect across all informal procedures is that a decision to adopt a package is always discussed within the development team. P3: *"We do this by reviewing each others work, and review of the library's code. Each code change is monitored by the team."* In two occasions this sometimes has to be explicitly discussed with an architect or DevOps (P7, P11). This social security ensures that all code is reviewed and no code is accepted without being looked by at least two members of the team.

In addition to discussing these matters within the team, there is one other aspect that is heavily relied on when selecting packages with an informal selection process. This is the common sense and self-awareness of the developer. P:6 *"... rules are not written in stone to be honest, and we just do what we think is right."* While this is trivial, it is named very specifically by P3, P5, P6, P10. The participants that had a formal process did not mention this, since the risk a development team could have is safeguarded by the formal procedure. So for the informal procedure this implies that there is a lot of implicit trust amongst team members, this can bring a significant risk to the project.

When looking at the participants who either had a formal or informal protocol, one can see that P4 has none. This is because this participant has not operated in a development team. However when continuing the line of questioning, after a while he revealed he had some sort of self-made protocol in his mind for selecting packages. Through time a mind model was created on how to approach these matters.

### **Adoption factors**

All the adoption factors that were found during the interviews are divided into the following categories:

- Technical adoption factors
- Organizational adoption factors
- Economic adoption factors

One may notice that the table holds more concrete examples of factors rather than the factors itself. Another noticeable aspect is that some of the named aspects may overlap in their meaning. E.g. the 'Number of users' and 'Number of contributors' are two separate aspects, where 'Community' also holds the number of users and contributors. This is because the majority of the participants described the actual actions and thought processes of why they would look at

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
<b>Technical</b>												
<i>Compatibility</i>		•				•	•		•	•		•
<i>Documentation</i>		•	•	•	•	•	•	•		•		•
<i>Complexity</i>	•		•		•			•				
<i>Security</i>							•	•		•		
<i>Source code</i>		•							•			
<i>Code quality</i>		•							•			•
<i>Customization</i>			•									
<i>Usability</i>						•						
<b>Organizational</b>												
<i>Active maintenance</i>	•		•	•	•	•			•		•	•
<i>Number of contributors</i>	•								•		•	
<i>Contributor process</i>					•			•				
<i>Git Issues</i>	•			•	•	•			•			•
<i>Number of users</i>	•	•	•			•				•		
<i>Backing company</i>		•		•							•	
<i>Ease of integration</i>	•	•	•			•				•		•
<i>Community</i>					•	•		•				
<i>Known vulnerabilities</i>							•	•				
<i>Reputation</i>		•					•	•				
<i>Support</i>				•	•						•	
<i>Git stars</i>				•								
<i>Knowledge of team</i>		•										
<b>Economical</b>												
<i>Licence</i>								•			•	•
<i>Total cost of ownership</i>	•									•		

Table 4.6: Adoption factors overview

a certain aspect. Therefore it would not be correct to categorize those specific aspects in a category, even though they do fall under that category.

A brief description of the specific aspects will now follow, along with a list of quotes to back them up.

### Technical factors

Appendix 4.1 holds the found technical aspects that influence the decision of which package to choose. After every aspect there is at least one quote from experts that support this. The following technical aspects were found in the interviews that influenced the interviewees as follows:

- *Compatibility* - The package in question must align with the current framework or build. This is an aspect that has to hold before any other characteristics are considered.
- *Documentation* - Dependent on the size and the complexity, documentation can play a big role in the decision of a package. This can also illustrate a picture on how serious the project is.
- *Complexity* - A package must not be too complex for the problem that is trying to solve, and should make coding easier.
- *Security* - A package should be secure, however there is no clear ways to determine this were mentioned.

- *Source code openness* - If the source code is available, a global scan can give an indication of the quality of the code and what it does.
- *Code quality* - The code should do what it claims it does and nothing more. Tests can help to validate this.
- *Customization* - If a package does not fully cover your needed functionalities, it is very convenient if small adjustments can be made to get full coverage.
- *Usability* - If a package is easy to use, it will contribute to the selection of that package.

### Organizational factors

Appendix 4.2 holds the found Organizational aspects that influence the decision of which package to choose. After every aspect there is at least one quote from experts that support this. The following organizational aspects were found in the interviews that influenced the interviewees as follows:

- *Active maintenance* - If a package is actively maintained, and has been for a period of time. It illustrates a stability, which is an important factor in the selection process.
- *Number of contributors* - When a lot of contributors work on a project it increases the stability of the package as well because if one of the contributors quits, there are enough others to take over the workload.
- *Contributor process* - This describes the ease of becoming a contributor for a project. There is no general way of finding this out, however one can get a feel of how hard it is to make contributions to a project. If this is very easy, it also means that people with a bad agenda can do this and potentially create a vulnerability in an upcoming release.
- *Git Issues* - Dependent on the type of Git issues, this can influence the choice of a package a lot. Issues can be used for feature requests but also for bugs or possible conflicts. The fashion in which these are taken care of shows a good picture on how serious the project is.
- *Number of users* - The number of users influences the choice of packages in two ways. Firstly, if a lot of people use it, it is implicitly tested. So if there is a bug or vulnerability, it will eventually be found. The second reason is that if a vulnerability was to be found, a lot of people would experience this so it will be patched very fast.
- *Backing company* - A backing company can ensure the project has enough resources to continue and therefore contributes to the stability of the project. However if such backing company gains too much influence, it can alter certain aspects of the project that would endanger its integrity.

- *Ease of integration* - When a package is easy to integrate, it can be verified quickly and this is important when selecting packages.
- *Community* - A large and active community contributes to the selection of a package since the power of open source is because of the diversity of contributors, users and documentation writers from different backgrounds to contribute to one project.
- *Known vulnerabilities* - The Common Vulnerabilities and Exposures (CVE) holds a database that has all known public security vulnerabilities. This database can be accessed to see if a project has had vulnerabilities. The number of vulnerabilities and the resolve time can provide a good indication of how serious the project is.
- *Reputation* - The reputation can either be from the project itself, if it has made a name for itself. Or from the contributors or developing entity. One tends to believe that an entity with a good reputation creates good software, and that one with a bad reputation can create bad software. Yet an important aspect is that no reputation has no negative influence.
- *Support* - Support is not a necessary aspect, however does positively influence the choice of a package if present.
- *Git stars* - The Git stars can give somewhat of an indication on the popularity of the project.
- *Knowledge of team* - Rather than an aspect about the project, this is an aspect of the current team a developer is operating in. To look at the strengths and weaknesses of the team, and start selecting packages with that in mind rather than choosing a package solely based on characteristics of that package.

### **Economical factors**

Appendix 7.3.3 holds the found economical aspects that influence the decision of which package to choose. After every aspect there is at least one quote from experts that support this. The following economical aspects were found in the interviews that influenced the interviewees as follows:

- *License* - Licenses may restrict the usage of projects for certain purposes and may influence one's own project.
- *Total cost of ownership* - The total costs are considered when selecting a package. When using open source software this is not very relevant most of the times. However this can sometimes result in some tricky business.

## 4.2.2 Section 2, Trust factors and metrics

The second section of the interview consists of two questions regarding trust in software. Trust factors as well as trust metrics were discussed in order to answer the following subquestion: Which factors contribute to gaining trust in a package?

### Trust factors

For the first three categories, the interviewees will think of all the factors that influence the decision of what packages will be selected. After this, they will create a subset of all those factors that actually cause them to trust a certain package. While these two aspects are very related, they are definitely not the same. P1: *"For me the economical factors play a role in deciding what package, however not in trusting the package."*

As with the adoption factors, some participants named the actual factors where other named concrete examples of those factors. They are displayed separately to prevent information loss. The found trust influencing aspects can also be categorized as technical and organizational aspects. Appendix 7.4.1 holds the found technical trust influencing aspects. Each of these aspects is supported by at least one quote and appendix 7.4.2 holds the found organizational trust influencing aspects. Table 4.7 displays all aspects that were named during the interviews:

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
<b>Technical</b>												
<i>Documentation</i>	•	•								•		
<i>Source code openness</i>	•	•										•
<i>Code quality</i>		•					•		•			
<b>Organizational</b>												
<i>Number of users</i>	•	•				•			•	•		•
<i>Active maintenance</i>				•		•						
<i>Community</i>			•			•	•				•	•
<i>Contributors</i>					•		•	•	•			
<i>Git Issues</i>				•	•						•	
<i>Git stars</i>				•					•			
<i>Backing company</i>		•						•	•		•	
<i>Ease of integration</i>						•						
<i>Stack overflow activity</i>			•									

Table 4.7: Trust factors overview

There are some trust factors that are not mentioned at the adoption factors. These factors are:

- *Source code openness* - When the source code is available it provides the possibility to go through the actual code and to let one decide for itself whether or not this piece of code is trustworthy. However as P1 pointed out: "There are so many sneaky ways to get a backdoor into software..." This does not guarantee that the code is safe however does contribute to the possible ways of finding out.

- *Stackoverflow activity* - This is a factor that one participant named specifically, and it is related to the community and the issues. Since stackoverflow is a question-and-answer platform that displays the activeness of the community and users.

### Trust metrics

Trust factors and trust metrics seem very alike, however there certainly is a distinction between them, moreover in the context that they were asked. One could argue that the trust metrics contain the concrete examples of trust factors, Which is true. However, the distinction for this research comes from the context in which the two were asked.

The question regarding the trust factors aims to uncover what aspects are relevant for an expert to gain trust in a software package. This question regards their personal perspective on what is important. Sometimes this would lead to concrete examples of trust factors: trust metrics. However, the question regarding trust metrics was asked after a detailed explanation about the TrustSECO project and what it aims to accomplish. A situation was described to the participants in which TrustSECO provided a trust score and they were asked what metrics they would value most in a trust score-breakdown. This contextual inconsistency makes all the difference here.

The remarkable aspect about this contextual difference is that some participants came up with metrics that represent a certain category. That were not named when asked about what factors are relevant when trusting a package, one question earlier.

As with the adoption and trust factors, some of the participants would name a category of metrics where others would name concrete examples. For now these will be named separately to prevent information loss. These can also be categorized in technical and organizational metrics.

Table 4.8 shows what categories of metrics, or concrete examples of metrics were found:

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
<b>Technical</b>												
<i>Code quality</i>		•						•	•			•
<i>Complexity</i>				•				•				
<i>Tests</i>		•		•		•						
<b>Organizational</b>												
<i>Active maintenance</i>			•			•	•		•	•		
<i>Number of contributors</i>						•		•	•			
<i>Users</i>		•	•	•		•		•	•	•		
<i>Stability metrics</i>					•							•
<i>Supporting platforms</i>							•					
<i>Reputation</i>		•									•	
<i>Git issues</i>					•		•		•			•
<i>Known vulnerabilities</i>			•		•		•					•
<i>Stack overflow</i>							•					
<i>Developing entity</i>		•								•		

Table 4.8: Trust metrics overview

One may notice that P1 does not have any trust metrics, this is because the question was added after the first interview. Appendix 7.5.1 and 7.5.2 hold a list of all found categories of metrics and metrics themselves with quotes to back them up.

### 4.2.3 Section 3, Personal bias and experience

This section of the interview consists of three questions aimed to answer the question: How do personal aspects influence trust in a package?

This section was created since the literature mainly concerned characteristics of the packages and projects, rather than the actual people who have to make decisions based on these characteristics. The difficult part in this is that there is no one correct answer as to how personal aspects influence this trust. This is because each of the participants has another perspective on this due to different past experiences. However there is one main conclusion to be drawn from this. That is that each of the participants have changed the way they look at selecting software over the years. This is due to the experiences they, but also experiences that the people around them had. Some have experienced more severe situations than others, and this results in them being more careful and paying more attention to this problem.

Another important aspect that could be concluded from this last section of the interview, is that some participants came up with additional factors or metrics that were not mentioned before. This illustrates that this is not a subject that is consciously thought about a lot, since it takes some thought process to come up with all the relevant aspects related to this matter. Hence proving the importance of this research. This then also implicitly means that there could be more relevant factors that are not uncovered during the interviews and would be if there would be more or different questions. Some participants, who had taken prior interest in this matter did not experience this growth in the interview since they had given this subject some more in-depth thought at some point in their career.

The interviews lead to the thought that there is no one-size-fits-all list of factors why a certain individual trusts a certain package since it there was a lot of variance between participants. However that by looking at enough individuals, a well defined list of factors can be defined that determines if a package is trustworthy.

## Chapter 5

# Discussion

One of the most prominent differences between the results from the literature and interviews, is the presence of technical factors for both the adoption and trust aspects. Technical factors are a lot more represented in literature than in the interviews. This is because only the factors or category of factors was noted if the participant named it before they were asked about it explicitly. Which revealed the following: a lot of the technical factors that were named in literature have to hold for interviewees to even consider looking at other factors. So this is definitely relevant when either adopting or trusting a package, and has to match with the current project in order for the project to be looked further into.

Another very important difference between some results are the results between the trust factors and trust metrics. First the participants were what factors could influence the trust on a package. After this they were asked what metrics they would like to see in an overview for determining trust. The question was initially not for this research and was to get a better idea for the general TrustSECO project, however turned out to be quite the contrary. These two questions would sometimes give vastly different answers.

This is due to a complication that this research has seen previously. Namely the difference in naming categories of factors vs. naming concrete examples of factors and the fact that there is a form of growth in the matter as the interview progresses. When discussing the trust metrics, the participants would sometimes name completely different aspects since the aspects had to be somewhat measurable and had not thought of certain aspects before.

This then also illustrates that there could be more trust factors than the participants have named collectively, since a lot of the steps and thought processes they go through are implicit.

### **TrustSECO**

This research set out to create a list of adoption and trust factors for software packages. The purpose of this list is to serve as guidance for creating a survey

that will sort the factors on importance. This will then be a major guidance for the TrustSECO project to weigh certain metrics connected to the prioritized factors, in order to create the final project. So from now on a survey needs to be created that ranks all the found factors, as well as all the metrics. The metrics are eventually going to be the most relevant for the TrustSECO project. A prototype of the project was recently created during Odyssey Momentum. This is a mass-online collaboration event in which the whole TrustSECO team invested a whole weekend to start creating the first version of the software. Appendix 7.6 holds more detail regarding my personal experience for this event.

## Chapter 6

# Conclusion

This research has shown that there are a lot of different factors that play a role when selecting or trusting software packages. It has also shown that these factors have a different impact on stakeholders depending on their role and the and situation.

For this research, there is a distinction between adoption factors and trust factors, both in literature and in the interviews. The adoption factors in literature share a great similarity with the ones found in the interviews. There are some differences. For example the interviews hold concrete examples of the better formulated categories in literature and there are few aspects that really differ from each other. Where the trust factors found in literature differ vastly from the majority of the found trust factors in the interviews. In literature the technical factors are the highest scoring factors where in the interviews the organizational seemed to be the most important.

Then there is the comparison between all found adoption and trust factors. Where the adoption factors display all the factors that are looked upon when selecting a package, the trust factors aim to illustrate what factors make people gain trust in a package. For the latter, this research has shown the difficulty to create a one size fits all trust factor list for individuals to gain trust in a package. This is because each individual has his or hers own past experiences with this matter, and thus has a different perspective on this. However, a good estimation of factors that make a package trustworthy can be generated by learning from enough individuals' experiences.

# Chapter 7

## Appendix

### 7.1 Literature review

#### 7.1.1 Article list SLR

Article title	Author	Year
The infeasibility of experimental quantification of life-critical software reliability	RW Butler, GB Finelli	1991
The infeasibility of quantifying the reliability of life-critical real-time software	RW Butler, GB Finelli	1993
Software reliability and system reliability	JC Laprie, K Kanoun	1996
Method and system for determining software reliability	MR Siegel, JI Ferrell	1996
Predicting software reliability from testing taking into account other knowledge about a program	A Bertolino, L Strigini	1996
Understanding the sources of variation in software inspections	A Porter, H Siy, A Mockus, L Votta	1998
Software metrics: successes, failures and new directions	NE Fenton, M Neil	1999
The paradoxes of free software	SM McJohn	2000
Open source software projects as virtual organisations: competency rallying for software development	K Crowston, B Scozzi	2002
Government preferences for promoting open-source software: A solution in search of a problem	B Reddy, DS Evans	2002
Why hackers do what they do: Understanding motivation and effort in free/open source software projects	KR Lakhani, RG Wolf	2003
Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel	G Hertel, S Niedner, S Herrmann	2003
Why open source software can succeed	A Bonaccorsi, C Rossi	2003
Open-source software development as gift culture: Work and identity formation in an internet community	M Bergquest	2003
Open source software for the public administration	GL Kovács, S Drozdik, P Zuliani. . .	2004
Open source software and open data standards in public administration	GL Kovács, S Drozdik, P Zuliani. . .	2004
The Collaborative Integrity of Open-Source Software	GR Vetter	2004
Resistance as motivation for innovation: Open source software	JF Kavanagh	2004
Agents of responsibility in software vulnerability processes	A Takanen, P Vuorijärvi, M Laakso, J Rönning	2004

Article title	Author	Year
Relationships between open source software companies and communities: Observations from Nordic firms	L Dahlander, MG Magnusson	2005
Participant satisfaction with open source software	BL Chawner	2005
Motivation, governance, and the viability of hybrid forms in open source software development	SK Shah	2006
Assessing the Impact of Project Founder Reputation and Project Structure on Motivation to Participate in Open Source Software Projects	K Ghosh, J Ziegelmayer, A Ammeter	2006
Location, location, location: How network embeddedness affects project success in open source systems	R Grewal, GL Lilien, G Mallapragada	2006
Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects	KJ Stewart, AP Ammeter...	2006
Software estimation: demystifying the black art	S McConnell	2006
Bounty programs in free/libre/open source software	S Krishnamurthy, AK Tripathi	2006
A software component quality model: A preliminary evaluation	A Alvaro, ES De Almeida...	2006
OSS opportunities in open source software—CRM and OSS standards	G Bruce, P Robson, R Spaven	2006
New Perspectives on Public Goods Production: Policy Implications of Open Source Software	JA Lee	2006
Developing an open source software development process model using grounded theory	Y Tian	2006
A Reputation-Based Mechanism for Software Vulnerability Disclosure	X Zhao	2007
The governance of free/open source software projects: monolithic, multidimensional, or configurational?	ML Markus	2007
Intrinsic motivation in open source software development	J Bitzer, W Schrettl, PJH Schröder	2007
An empirical analysis of the impact of software vulnerability announcements on firm stock price	R Telang, S Wattal	2007
Reputation in Open Source Software Virtual Communities	IV Casalo, J Cisneros, C Flavián...	2008
Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties	J Hahn, JY Moon, C Zhang	2008
Temporal metrics for software vulnerabilities	JA Wang, F Zhang, M Xia	2008
Method and apparatus for detecting vulnerabilities and bugs in software applications	VC Sreedhar, GF Cretu, JT Dolby	2008
User and developer mediation in an Open Source Software community: Boundary spanning through cross participation in online discussions	F Barcellini, F Détienne, JM Burkhardt	2008
An approach for selecting software-as-a-service (SaaS) product	M Godse, S Mulik	2009
Impact of license choice on open source software development activity	J Colazo, Y Fang	2009
Research on testing-based software credibility measurement and assessment	Q Hongbing, Z Xiaojie...	2009
Designers wanted: participation and the user experience in open source software development	PM Bach, R DeLine, JM Carroll	2009
3.5 Open Source Software Research and Blockchain	J Lindman	2009
“Constructing the users” in open source software development	N Iivari	2009
System and method for maximizing software package license utilization	S Varadarajan, G Sridhar, KK Rao	2010
Software metrics and software metrology	A Abran	2010
Creating and evolving developer documentation: understanding the decisions of open source contributors	B Dagenais, MP Robillard	2010
Code forking in open-source software: a requirements perspective	NA Ernst, S Easterbrook, J Mylopoulos	2010
Trust and reputation for successful software self-organisation	JM Seigneur, P Dondio	2011

Article title	Author	Year
SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments	L Wu, SK Garg, R Buyya	2011
A systematic literature review on fault prediction performance in software engineering	T Hall, S Beecham, D Bowes, D Gray...	2011
Software quality: theory and management	A Gillies	2011
A risk assessment framework for evaluating Software-as-a-Service (SaaS) cloud services before adoption	L Bernard	2011
Understanding broadcast based peer review on open source software projects	PC Rigby, MA Storey	2011
A theory-grounded framework of Open Source Software adoption in SMEs	RD Macredie, K Mijinyawa	2011
Understanding open source software peer review: Review processes, parameters and statistical models, and underlying behaviours and mechanisms	PC Rigby	2011
Design and evaluation of a process for identifying architecture patterns in open source software	KJ Stol, P Avgeriou, MA Babar	2011
Analyzing and Identifying SaaS for Development of a Project by calculating its Reputation	BR Rao	2012
Carrots and rainbows: Motivation and social practice in open source software development	G Von Krogh, S Haefliger, S Spaeth, MW Wallin	2012
A model of open source developer foundations	D Riehle, S Berschneider	2012
Research of trustworthy software system in the network	Y Liu, L Zhang, P Luo, Y Yao	2012
Study on credibility level of trustworthy software development process based on grey nonlinear cluster	S Liu, J Forrest, Y Yangjie, K Zhang, C Mi, N Xie...	2012
A non-functional requirements tradeoff model in trustworthy software	MX Zhu, XX Luo, XH Chen, DD Wu	2012
How peripheral developers contribute to open-source software development	P Setia, B Rajagopalan...	2012
Research note—Lock-in strategy in software competition: Open-source software vs. proprietary software	KX Zhu, ZZ Zhou	2012
Why do commercial companies contribute to open source software?	M Andersen-Gott, G Ghinea, B Bygstad	2012
Do the allocation and quality of intellectual assets affect the reputation of open source software projects?	R Méndez-Durón	2013
Towards reputation-as-a-service	C Hillebrand, M Coetzee	2013
Software fault prediction metrics: A systematic literature review	D Radjenović, M Heričko, R Torkar...	2013
Automatic polymorphic exploit generation for software vulnerabilities	M Wang, P Su, Q Li, L Ying, Y Yang, D Feng	2013
'Computing'Requirements in Open Source Software Projects	X Xiao, A Lindberg, S Hansen, K Lyytinen	2013
From closed to open: Job role changes, individual predispositions, and the adoption of commercial open source software development	O Alexy, J Henkel, MW Wallin	2013
Learning and best practices for learning in open-source software communities	V Singh, L Holt	2013
All complaints are not created equal: text analysis of open source software defect reports	U Raja	2013
How social QnA sites are changing knowledge sharing in open source software communities	B Vasilescu, A Serebrenik, P Devanbu...	2014
Secured trust and reputation system: analysis of malicious behaviors and optimization	A Bradai	2014
Measuring the health of open source software ecosystems: Beyond the scope of project health	S Jansen	2014
Software Reliability: State of the Art Report 14: 2	A Bendell, P Mellor	2014
Auditing and maintaining provenance in software packages	Q Pham, T Malik, I Foster	2014
Estimating development effort in free/open source software projects by mining software repositories: a case study of openstack	G Robles, JM González-Barahona, C Cervigón...	2014

Article title	Author	Year
Transactive memory system, communication quality, and knowledge sharing in distributed teams: An empirical examination in open source software project teams	X Chen	2014
The Spack package manager: bringing order to HPC software chaos	T Gamblin, M LeGendre, MR Collette. . .	2015
Analysis and assessment of software library projects	JW Nicol, BL Roberts, JO Pillgram-Larsen. . .	2015
Software applications have on average 24 vulnerabilities inherited from buggy components	L Constantin	2015
Raising the general public's awareness and adoption of open source software through social QnA interactions	N Choi, K Yi	2015
Group Reputation in an Open Source Software Community: Antecedents and Outcomes	Y Cai, D Zhu	2016
Maintenance effort estimation for open source software: A systematic literature review	H Wu, L Shi, C Chen, Q Wang. . .	2016
Modeling library dependencies and updates in large software repository universes	RG Kula, C De Roover, DM German, T Ishio. . .	2017
Secure dependency enforcement in package management systems	L Catuogno, C Galdi, G Persiano	2017
Large-scale Modeling, Analysis, and Preservation of Free and Open Source Software	S Zacchiroli	2017
Open Source Software Hosting Platforms: A Collaborative Perspective's Review.	G Alamer, S Alyahya	2017
Software processes analysis with provenance	GCB Costa, HLO Dalpra, EN Teixeira. . .	2018
Software Provenance: Track the Reality Not the Virtual Machine	D Wilkinson, L Oliveira, D Mossé. . .	2018
Hackers vs. testers: A comparison of software vulnerability discovery processes	D Votipka, R Stevens, E Redmiles, J Hu. . .	2018
A business model for commercial open source software: A systematic literature review	S Shahrivar, S Elahi, A Hassanzadeh. . .	2018
Collaborative SLA and reputation-based trust management in cloud federations	K Papadakis-Vlachopapadopoulos. . .	2019
A systematic examination of knowledge loss in open source software projects	M Rashid, PM Clarke, RV O'Connor	2019
THE TAKEOFF OF OPEN SOURCE SOFTWARE: A SIGNALING PERSPECTIVE BASED ON COMMUNITY ACTIVITIES.	P Setia, BL Bayus, B Rajagopalan	2020

## 7.2 Interviews

### 7.2.1 Informed consent

#### Informed Consent



##### Interview information

- The interview will be recorded and a transcript will be produced
- A transcript will be sent to you and you will be given the opportunity to correct any factual errors
- The transcript of the interview will be analysed by Floris Jansen as research investigator.
- Access to the interview transcript will be limited to Floris Jansen and academic colleagues and researchers with whom he might collaborate as part of the research process
- Any summary interview content, or direct quotations from the interview, that are made available through academic publication or other academic outlets will be anonymized so that you cannot be identified, and care will be taken to ensure that other information in the interview that could identify yourself is not revealed.
- The actual recording of the interview will be destroyed upon completion of the research.
- Any variation of the conditions above will only occur with your further explicit approval.

##### Notes information

- In case you do not agree with the interview being recorded, research notes will be taken during the interview.
- The research notes will be processed and you will be sent the results and you will be given the opportunity to correct any factual errors.
- Afterwards, the notes will be treated like the interview transcripts described above.

## Informed Consent



I, the undersigned, confirm that (please tick box as appropriate):

1.	I have been provided with information on the purpose of this project.	<input type="checkbox"/>
2.	I have been given the opportunity to ask questions about the project and my participation.	<input type="checkbox"/>
3.	I voluntarily agree to participate in the project.	<input type="checkbox"/>
4.	I understand I can withdraw at any time without giving reasons and that I will not be penalised for withdrawing.	<input type="checkbox"/>
5.	The procedures regarding confidentiality have been clearly explained (e.g. use of names, pseudonyms, anonymisation of data, etc.) to me.	<input type="checkbox"/>
6.	The use of the data in research, publications, sharing and archiving has been explained to me.	<input type="checkbox"/>
7.	Select only <b>one</b> of the following:	
	<ul style="list-style-type: none"><li>• I agree to recording the interview conducted for this project</li> <li>• I want to participate in the interview without it being recorded. The researcher will make notes instead.</li></ul>	<input type="checkbox"/> <input type="checkbox"/>
8.	I have read and understand the information provided on the use of interview recordings/research notes.	<input type="checkbox"/>
9.	I agree with the use of interview recordings/research notes as stated in this information.	<input type="checkbox"/>
10.	I, along with the Researcher, agree to sign and date this informed consent form.	<input type="checkbox"/>

### Participant:

\_\_\_\_\_  
Name of Participant

\_\_\_\_\_  
Signature

\_\_\_\_\_  
Date

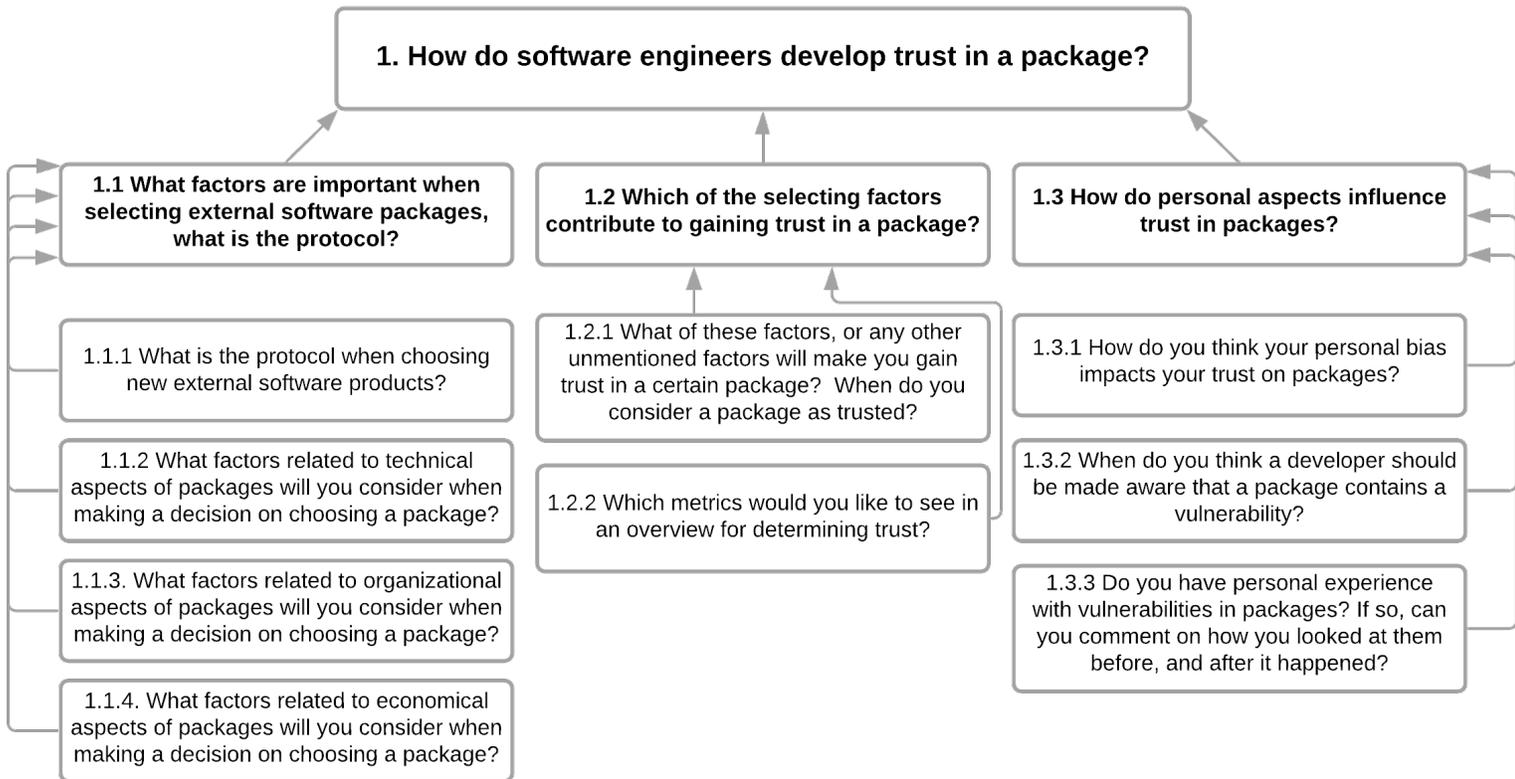
### Researcher:

\_\_\_\_\_  
Name of Researcher

\_\_\_\_\_  
Signature

\_\_\_\_\_  
Date

## 7.2.2 Interview layout



### 7.2.3 Interview protocol

<p>Utrecht University is researching the factors that influence software packages selection. Such an impact factor can be defined as a characteristic of a software package that results in trust in that package. The results of these interviews, together with a literature study will lead to a survey that, if done on a large scale, will reveal what these impact factors are. This interview will be about 45 minutes. This information will be available through the informed-consent, that needs to be signed before the interview starts.</p>
<p>State: name, age, nationality, education, function(profession), years of experience, organization, involvement in choosing packages, selection process used in the organization, type of organization, industrial sector of organization, usage of external products within organization</p>
<p>1. What is the protocol when choosing new external software products?</p>
<p>2. What factors related to technical aspects of the packages will you consider when making a decision on choosing a package?</p>
<p>3. What factors related to organizational aspects of the packages will you consider when making a decision on choosing a package?</p>
<p>4. What factors related to economical aspects of the packages will you consider when making a decision on choosing a package?</p>
<p>5. What of these factors, or any other unmentioned factors will make you gain trust in a package, when do you consider a package trusted?</p>
<p>6. What metrics would you like to see in an overview for determining trust?</p>
<p>7. How do you think your personal bias impacts your trust on packages?</p>
<p>8. When do you think a developer should be made aware that a package contains a vulnerability?</p>
<p>9. Do you have personal experience with vulnerabilities in packages? If so, can you comment on how you looked at them before, and after it happened?</p>

## 7.3 Adoption factors

### 7.3.1 Technical adoption factors

<b>Compatibility</b>
<ul style="list-style-type: none"><li>• "Well first of all it has to fit within the current application"</li><li>• "So the most important one is that it is compliant with your current framework."</li><li>• "First we take a look how secure the project is, and how compliant it is with our current build"</li><li>• "I globally scan the code to see if I understand what it does, also to see if it fits my use case"</li><li>• "We take a look at the functionality, and does this fit within the current framework"</li><li>• "I look at the documentation to see what datastructures are used, and if they are compatible with my project"</li></ul>
<b>Documentation</b>
<ul style="list-style-type: none"><li>• "Good comments and a good readme are crucial"</li><li>• "With Python I mainly look at the documentation, what does it look like ..."</li><li>• "dependant on the size and complexity, the documentation can be really important or irrelevant..."</li><li>• "If every new function requires 2 days of reverse engineering because the documentation is bad, drop this package!"</li><li>• "if there is no documentation I have to figure it out myself. It makes it hard for us to work with so it is not a matter of trust but then it comes back down to the pillar's ease of use."</li><li>• "... and after that you look at things like documentation and community"</li><li>• "Documentation can be a good indication on how mature a project is, and that multiple people are seriously working on this project."</li><li>• "By looking at the documentation we can see if it is a good or bad package"</li><li>• "I check that by looking at the documentation or the API"</li></ul>
<b>Complexity</b>
<ul style="list-style-type: none"><li>• "If I want to solve a problem, I want to find a dependency that only solves this problem and nothing else"</li><li>• "It should be compact, especially with Python people do this elegantly, is should serve a single purpose and be clear"</li><li>• "A solution should not be too complex for what it is trying to solve, it should make the coding easier"</li><li>• "Oh I forgot to mention, I also check some of the source code of the project. To get a feel on how the comments are, how complex are the functions etc. So if i want to make little adjustments I can easily understand how and where this should be done."</li><li>• "... if that is the case we might use it, one of the aspects we then look at is the size of the package"</li><li>• "One other aspect to look at is: how difficult is it to replace that component?"</li></ul>
<b>Security</b>
<ul style="list-style-type: none"><li>• "First we take a look how secure the project is, and how compliant it is with our current build"</li><li>• "We take a look at how easy it is to integrate in our system, and whether or not it is secure"</li><li>• "We think security is very important, so we use several static code analysis tools created by the software improvement group"</li></ul>
<b>Source code openness</b>
<ul style="list-style-type: none"><li>• "I globally scan the code to see if I understand what it does, also to see if it fits my use case"</li><li>• "If you can read what the code does exactly, that is very important for me"</li></ul>
<b>Usability</b>
<ul style="list-style-type: none"><li>• "And releases are very important, and also the usability aspect should not be overlooked"</li></ul>
<b>Customization</b>
<ul style="list-style-type: none"><li>• "I want a package that is not too complex for what it should do, so you want it to be customizable enough to ensure it can be used for your purpose."</li></ul>
<b>Code Quality</b>
<ul style="list-style-type: none"><li>• "If there are tests available, so you can see that some functions really do the things they claim they do"</li><li>• "There are also several code quality tools that grade a package, they are also good indicators to watch"</li><li>• "It should do what you need it do you, and what it says that it does. If that is not the case then I will not even consider it"</li></ul>

### 7.3.2 Organizational adoption factors

<p><b>Active maintenance</b></p> <ul style="list-style-type: none"> <li>• "For me one of the first things I look at it how regularly it is updated"</li> <li>"When was it updated for the last time, first time in 3 years or recently?"</li> <li>• "If the last update was in 2016, we will not even consider this package"</li> <li>• "You could then use an external library, yet you want to know how active it is and if it is maintained well "</li> <li>• "The first step for me is to go to GitHub and check how often there are new commits"</li> <li>• "the second step is how regular those commits are..."</li> <li>• "An important factor is to get the feeling that the project is actively maintained, so check the list of releases"</li> <li>• "They are constantly updating and releasing. And releases are very important,"</li> <li>• "The thing is that I want to see that this is updated regularly."</li> <li>• "The first thing I do is go to GitHub and check the amount of contributors and how active it is maintained"</li> <li>• "I don't like it either when nothing has changed over the past 2 years, or only 1 file at the time"</li> <li>• "How many commits there are, when the last commit was and how active it still is. also take a look at the git issues and how they are handled"</li> </ul>
<p><b>Amount of contributors</b></p> <ul style="list-style-type: none"> <li>• "Also the amount of contributors is important"</li> <li>• "The first thing I do is go to GitHub and check the amount of contributors and how active it is maintained"</li> <li>• "If there are many users, but only 2 contributors I do not like it. Since that comes with big risks"</li> <li>• "If there are many contributors, that are also bonus points"</li> </ul>
<p><b>Contributor process</b></p> <ul style="list-style-type: none"> <li>• "Another aspect is to know what kind of people are involved in the project, have they done project like this in the past or is it their first time, and how easy is it to become involved"</li> <li>• "... another important thing is to try to get an idea of how easy someone can become a maintainer, and have there been many over the years or is there a steady team"</li> </ul>
<p><b>Git Issues</b></p> <ul style="list-style-type: none"> <li>• "If it is on GitHub, it is certainly important how many issues are open and how often they are responded to"</li> <li>• "The 3rd step is to look at the git issues, how many are open and what types of issues are there"</li> <li>• "A good way to get a feeling about how serious a project is, is to look at the issue list in GitHub. There you can see what the responses are and how they are being handled"</li> <li>• "And the community as well. What are the known issues, and how fast are they resolved?"</li> <li>• "If there are 10k issues, which do not get a response then this is a no go for me"</li> <li>• "How many commits there are, when the last commit was and how active it still is. also take a look at the git issues and how they are handled"</li> </ul>
<p><b>Amount of Users</b></p> <ul style="list-style-type: none"> <li>• "The amount of users is also a good indication, because if many people use this, it is implicitly tested. Even if this was not the case before it launched. It certainly tested now in practice.</li> <li>• "If many others use it as well, that helps a lot"</li> <li>• "An important matter is how many other people use this"</li> <li>• "Amount of users! If I see 2 frameworks, 1 with 20 and 1 with 80 people. The choice will go to which is most used.""</li> <li>• "The amount of users for example, that is important"</li> </ul>
<p><b>Backing company</b></p> <ul style="list-style-type: none"> <li>• "Big frameworks like angular, react or vue, angular is backed by google and react by Facebook. So they will go down if the company goes down and that is not going to happen anytime soon"</li> <li>• "If there is a backing company that delivers support it really gives bonus points"</li> <li>• "For me it is really important to see the maturity of the organisation"</li> <li>• "If a company has a certain way of developing , that is important"</li> </ul>
<p><b>Ease of integration</b></p> <ul style="list-style-type: none"> <li>• "It is very dependant on how easy it is to implement."</li> <li>• "It should be easy to implement..."</li> <li>• "If it is a small library, I look at the implementation to see if it is easy to integrate"</li> <li>• "How easy is it to implement, what is the available documentation for that?"</li> <li>• "a trustworthy package is one where I can install it and see if it is working within a couple of minutes. "</li> <li>• "We take a look at how easy it is to integrate in our system, and whether or not it is secure"</li> </ul>

<b>Community</b>
<ul style="list-style-type: none"> <li>• "Another thing for me is the community."</li> <li>• "And the community as well. What are the known issues, and how fast are they resolved?"</li> <li>• "... and after that you look at things like documentation and community"</li> <li>• "After that I start to take a look at the community, to see if I know some people. I might have met them at a conference maybe."</li> <li>• "Next to reputation and community there are not a lot of things I look at, they are the 2 most important ones"</li> </ul>
<b>Known vulnerabilities</b>
<ul style="list-style-type: none"> <li>• "I do not even consider the package if there are a lot of cve's known for it"</li> <li>• "You can look at the history of the security vulnerabilities to get a feeling of how serious the project is"</li> </ul>
<b>Reputation</b>
<ul style="list-style-type: none"> <li>• "Are other people talking about this, if it is popular that helps"</li> <li>• "Either the project or the developers can have a reputation that influences the process"</li> <li>• "Next to reputation and community there are not a lot of things I look at, they are the 2 most important ones"</li> </ul>
<b>Support</b>
<ul style="list-style-type: none"> <li>• "as a user of open source software you need to know how serious you are treated, and what kind of support is available to you"</li> <li>• "If there is a backing company that delivers support it really gives bonus points"</li> <li>• "The frameworks I use, I always want them to have long term support"</li> </ul>
<b>Git stars</b>
<ul style="list-style-type: none"> <li>• "Oh and I forgot to mention GitHub stars, they are essential!"</li> </ul>
<b>Knowledge of team</b>
<ul style="list-style-type: none"> <li>• "If there already are people with knowledge of the package in my team, that helps a to decide whether or not it will fit in our project"</li> </ul>

### 7.3.3 Economical adoption factors

<b>License</b>
<ul style="list-style-type: none"> <li>• "A license can actually be very dangerous for us, there are licenses that forbid military usage, but also others that require that our code needs to be open source too"</li> <li>• "We had to take a look at licenses, since if we used certain software with limited licenses, we could not sell the end product"</li> <li>• "License is definitely a big factor, you do need to have a clear image on costs or limitations there"</li> </ul>
<b>Total costs</b>
<ul style="list-style-type: none"> <li>• "For me the economical factors play a role in deciding what package, however not in trusting the package"</li> <li>• "We take a look at the pricing, open source is better since its free"</li> </ul>

## 7.4 Trust factors

### 7.4.1 Technical trust factors

<b>Documentation</b>
<ul style="list-style-type: none"><li>• "Good documentation definitely contributes to trusting a package, it shows me I can use the software for what I want to use it for"</li><li>• "A good read me is crucial, has someone put real effort in that. In other words how is the documentation"</li><li>• "By looking at the documentation we can see if it is a good or bad package"</li></ul>
<b>Code quality</b>
<ul style="list-style-type: none"><li>• "especially to validate that it does what it should do tests are very important"</li><li>• "Not just the code quality, but also if there is a general test suite and how much it covers the whole package"</li></ul>
<b>Source code</b>
<ul style="list-style-type: none"><li>• "The good thing about open source is that the source code is often available, so I can often check the code itself. This really helps with trusting the package"</li><li>• "Another aspect is the quality of the code itself, sometimes you cannot read this but you can get a general feeling on the code quality"</li><li>• "There are also several code quality tools that grade a package, they are also good indicators to watch"</li></ul>

### 7.4.2 Organizational trust factors

<b>Amount of users</b>
<ul style="list-style-type: none"><li>• "A package that is used by 300.000 others, I trust more than a package that is released last week and has been used by 5 people, my trust in that is way lower even though it might be better that the bigger one"</li><li>• "If many others use it as well, that helps a lot"</li><li>• "Amount of users! If I see 2 frameworks, 1 with 20 and 1 with 80 people. The choice will go to which is most used."</li><li>• "The amount of users for example, that is important"</li><li>• "A good indication on how active the project is used is the weekly downloads for example"</li><li>• "After that I look at the size of the community and other users... If there are enough other users, it means there are enough people who can also experience problems that need fixing"</li></ul>
<b>Community</b>
<ul style="list-style-type: none"><li>• "I think community is very important, it should be actively used"</li><li>• "I would like to add something to the previous question, namely an active mailing list around a library actually does gives me more trust in the library or package"</li><li>• "Another thing for me is the community."</li><li>• "And the community as well. What are the known issues, and how fast are they resolved?"</li><li>• "Things that really give me a feeling of trust are, are there smart developers involved, how are the discussions online and to the developers give reasoning why they do certain things"</li><li>• "How large the community is says a lot"</li><li>• "After that I look at the size of the community and other users..."</li></ul>
<b>Contributors</b>
<ul style="list-style-type: none"><li>• "... and if a project is maintained by 1 or two people, the chance that the project stops due to issues in their personal life are quite real. So for me, community really gives trust if it is a large and stable community."</li><li>• "If a project with 3000 contributors, however 99,9 of the code is written by 1 person than the amount of contributors still doesn't tell me anything, so I will have less trust for that project"</li><li>• "Things that really give me a feeling of trust are, are there smart developers involved, how are the discussions online and to the developers give reasoning why they do certain things"</li><li>• "Try to get an idea of how hard it is to become a contributor, over the last years several back doors have appeared in libraries because it was to easy to become a contributor for example"</li></ul>

<b>Active maintenance</b>
<ul style="list-style-type: none"> <li>• "My trust in a package is mainly from GitHub, so the stars, commit history and the open and closed issues"</li> <li>• "They are constantly updating and releasing. And releases are very important,"</li> <li>• "The thing is that I want to see that this is updated regularly."</li> </ul>
<b>Reputation</b>
<ul style="list-style-type: none"> <li>• "I work a lot with PHP, and there are some well known companies and people in the community. If you see a project is guided or written by one of them you know you are in the clear. That ensures more trust instantly"</li> </ul>
<ul style="list-style-type: none"> <li>• "If you can read what the code doe exactly that is important"</li> <li>• "I think that being able to see the source code really creates trust"</li> <li>• "Like I previously said, it first has to do what I need it to do before I look any further"</li> </ul>
<b>Ease of integration</b>
<ul style="list-style-type: none"> <li>• "a trustworthy package is one where I can install it and see if it is working within a couple of minutes. "</li> </ul>
<b>Stack overflow activity</b>
<ul style="list-style-type: none"> <li>• "Stack overflow activity also adds trust"</li> </ul>
<b>Git Issues</b>
<ul style="list-style-type: none"> <li>• "My trust in a package is mainly from GitHub, so the stars, commit history and the open and closed issues"</li> <li>• "I have seen several teams that used a project and then stopped using it after a while because the issues were not resolved, so I think that is really important for trusting a project"</li> <li>• "I look at everything from Git Issues, til meetups and how the community communicates"</li> </ul>
<b>Git Stars</b>
<ul style="list-style-type: none"> <li>• "My trust in a package is mainly from GitHub, so the stars, commit history and the open and closed issues"</li> <li>• "One thing I then look at is the stars on GitHub"</li> </ul>
<b>Backing company</b>
<ul style="list-style-type: none"> <li>• "I also look if there is a commercial interest, with perhaps a company that supports this project. Red-hat is a nice example of that"</li> <li>• "If a project is backed by a good company, that also creates trust"</li> <li>• "For me it is really important to see the maturity of the organisation"</li> <li>• "If a company has a certain way of developing , that is important"</li> </ul>

## 7.5 Trust metrics

### 7.5.1 Technical trust metrics

<b>Complexity</b>
<ul style="list-style-type: none"> <li>• "A metric to determine this is the cyclomatic complexity, or how many lines of code in general. I think that would be a good start."</li> <li>• "Another aspect would be to get an indication how complex it is, and how many people use the full complexity"</li> </ul>
<b>Tests</b>
<ul style="list-style-type: none"> <li>• "Would a metric like test coverage be a solid one?"</li> <li>• "Is there something that can prove that it is functional, and how is the test coverage in that?"</li> <li>• "What is important is basically, you can see with many package managers I think these build scripts that are run to test this. And to validate this.</li> <li>• "Yes you can see how many times has it passed, and how many times has it failed. And I think these are important metrics to calculate trust.</li> </ul>

<b>Code quality</b>
<ul style="list-style-type: none"> <li>• "I think it is hard because it is about trust, however I think code quality is the most important in this one"</li> <li>• "If we then have the possibility I would like to see some code quality metrics, dependant on the ecosystem."</li> <li>• "This does fall under code quality however does deserves to be named separately is the documentation."</li> <li>• "We would then be discussing activity and security metrics... For security and quality I would grasp to some models I already know from the Software Improvement group."</li> </ul>

## 7.5.2 Organizational trust metrics

<b>Active maintenance</b>
<ul style="list-style-type: none"> <li>• "It is not about the bugs, but how fast they are fixed. This way we can see that there are active patches coming out." So how often is it updated? This is a tricky one since if a certain package is released and it is a very basic function but it works, then why would you update it?</li> <li>• "So how would you then define the number of releases. It is a hard metric."</li> <li>• "Yeah so some measurable aspects then, this could be the last release and amount of commits per year I think."</li> <li>• "In addition, the last release date and perhaps some data on how often there are new releases"</li> <li>• "In addition I would also like to see it actively maintained, unless it is such a fundamental package that is does not need maintenance."</li> </ul>
<b>Amount of contributors</b>
<ul style="list-style-type: none"> <li>• "I think users, contributors, releases. "</li> <li>• "I think community size, with that I mean 2 aspects: the amount of contributors and the amount of users that use that component"</li> <li>• "I would like to see the amount of stars, the amount of contributors and amount of weekly installs."</li> </ul>
<b>Users</b>
<ul style="list-style-type: none"> <li>• "I think community size, with that I mean 2 aspects: the amount of contributors and the amount of users that use that component"</li> <li>• "Who made it and how many people are using it?"</li> <li>• "A really cool thing would be the ratio between people who try to use it and 3 months later still do"</li> <li>• "I would like to see how many people use the component and for how long they have used it"</li> <li>• "I think users, contributors, releases. "</li> <li>• "I would like to see the amount of stars, the amount of contributors and amount of weekly installs."</li> <li>• "I would like to see that it is active, more specific actively used"</li> </ul>
<b>Stability metrics</b>
<ul style="list-style-type: none"> <li>• "So what would be nice to see is something to cover the stability, so how many incidents have happened with a certain version."</li> <li>• "If we look at open source projects, I would like to see something on the stability of the team"</li> <li>• "Another aspect is how diverse their revenue streams are, if they are relying on 1 source this could fall apart more easily"</li> <li>• "You should mention the supported frameworks, platforms and operating systems"</li> </ul>
<b>Git issues</b>
<ul style="list-style-type: none"> <li>• "Most of the times there is no response to the issues, so if those are responded to that already tells a great deal"</li> <li>• "If we could get a score on how fast issues on GitHub are closed, that would score points"</li> <li>• "I think that the amount of GitHub stars as well as the amount of issues that are open"</li> <li>• "I think the ratio between the amount of closed and open issues would be a nice addition"</li> <li>• "I would like to see the amount of stars, the amount of contributors and amount of weekly installs."</li> <li>• "I think they also did some measurements for an issue resolution time, that would be a great addition"</li> </ul>

<b>Known vulnerabilities</b>
<ul style="list-style-type: none"> <li>• "At first we want to know that there are few known vulnerabilities"</li> <li>• "Another thing to note is the amount of CVE's that are known"</li> <li>• "Some statistics and history on prior security bugs"</li> <li>• "Look if there are known vulnerabilities I would like to know, even if it's that simple, it's still something already"</li> </ul>
<b>Stackoverflow presence</b>
<ul style="list-style-type: none"> <li>• "If we may fantasize about it a little I think that the amount of questions on stackoverflow would be a great addition, as well as blog posts or presence on reddit."</li> </ul>
<b>Reputation</b>
<ul style="list-style-type: none"> <li>• "What important is for me, is that e.g. someone like Linus Torvalds expresses faith in the project"</li> <li>• "Reputation of the developer is a good metric as well"</li> </ul>
<b>Developing entity</b>
<ul style="list-style-type: none"> <li>• "I would like to see who is behind it, the person or company. Even though it then is difficult to find out if you can trust it but you can at least try."</li> <li>• "It can very well be that you do not get any information out of it but let's take react for example, that is created by Facebook. Even though I do not like Facebook as a company, I do trust that the software they produce is good."</li> <li>• "Reputation of the developer is a good metric as well"</li> </ul>

## 7.6 Odyssey Momentum

The massive online collaboration event or hackathon as it is normally called was a three day event focused on creating projects and sharing knowledge. There were 13 different tracks that each covered different social or technical problems. A team could then reach out to the organisation with a possible solution, and this weekend then provided the opportunity to start creating. This weekend had a very large focus on the collaboration aspect. This is a yearly physical event and due to the corona virus could not be held this year. Instead the creators created this online collaboration platform in which one could roam freely and socialize with other participants of the event. This way if a team was stuck on a certain problem, they could fly around in the Odyssey-world and find some other participant who does have the knowledge to solve this issue. This is a beautifully designed concept that allowed teams to collaborate remotely whilst working on their own projects. Since my coding skills are not nearly as good as my team members'. I was part of the 'marketing team' for this weekend. This meant keeping the social media platforms up-to-date with the team's progress. It also meant communicating with other teams to find the appropriate other participants that could help us solve issues that we had. There also were several presentations that were done by the marketing team. Overall this was an amazing addition to the research project. Even though I did not code too much of the actual prototype, this weekend was a great learning experience in terms of communicating within a team and being the middle man for communication towards our team and the organization. The cherry on top is that we won the track in which we competed! This was an amazing finalization of the weekend!

# Bibliography

- [Artz and Gil, 2007] Artz, D. and Gil, Y. (2007). A survey of trust in computer science and the semantic web. *Journal of Web Semantics*, 5(2):58–71.
- [Bernstein, 2005] Bernstein, L. (2005). Trustworthy software systems. *ACM SIGSOFT Software Engineering Notes*, 30(1):4–5.
- [Bunduchi, 2013] Bunduchi, R. (2013). Trust, partner selection and innovation outcome in collaborative new product development. *Production planning & control*, 24(2-3):145–157.
- [Cho et al., 2015] Cho, J.-H., Chan, K., and Adali, S. (2015). A survey on trust modeling. *ACM Computing Surveys*, 48:1–40.
- [Del Bianco et al., 2011] Del Bianco, V., Lavazza, L., Morasca, S., and Taibi, D. (2011). A survey on open source software trustworthiness. *IEEE software*, 28(5):67–75.
- [Duan et al., 2020] Duan, R., Alrawi, O., Kasturi, R. P., Elder, R., Saltaformaggio, B., and Lee, W. (2020). Measuring and preventing supply chain attacks on package managers. *arXiv preprint arXiv:2002.01139*.
- [Glaser and Strauss, 2017] Glaser, B. G. and Strauss, A. L. (2017). *Discovery of grounded theory: Strategies for qualitative research*. Routledge.
- [Grandison and Sloman, 2000] Grandison, T. and Sloman, M. (2000). A survey of trust in internet applications. *IEEE Communications Surveys & Tutorials*, 3(4):2–16.
- [Hernán et al., 2004] Hernán, M. A., Hernández-Díaz, S., and Robins, J. M. (2004). A structural approach to selection bias. *Epidemiology*, pages 615–625.
- [Hiller and DiLuzio, 2004] Hiller, H. H. and DiLuzio, L. (2004). The interviewee and the research interview: Analysing a neglected dimension in research. *Canadian Review of Sociology/Revue canadienne de sociologie*, 41(1):1–26.
- [Hou F, 2020] Hou F, Farshidi S, J. S. (2020). Trustseco: A distributed infrastructure for providing trust in the software ecosystem.

- [Hove and Anda, 2005] Hove, S. E. and Anda, B. (2005). Experiences from conducting semi-structured interviews in empirical software engineering research. In *11th IEEE International Software Metrics Symposium (METRICS'05)*, pages 10–pp. IEEE.
- [Jacob and Furgerson, 2012] Jacob, S. A. and Furgerson, S. P. (2012). Writing interview protocols and conducting interviews: tips for students new to the field of qualitative research. *Qualitative Report*, 17:6.
- [Kothari, 2004] Kothari, C. R. (2004). *Research methodology: Methods and techniques*. New Age International.
- [Mojica et al., 2014] Mojica, I. J., Adams, B., Nagappan, M., Dienst, S., Berger, T., and Hassan, A. E. (2014). A large-scale empirical study on software reuse in mobile apps. *IEEE Software*, 31(2):78–86.
- [Mui et al., 2002] Mui, L., Mohtashemi, M., and Halberstadt, A. (2002). A computational model of trust and reputation. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 2431–2439. IEEE.
- [Nguyen et al., 2020] Nguyen, P. T., Di Rocco, J., Di Ruscio, D., and Di Penta, M. (2020). Crossrec: Supporting software developers by recommending third-party libraries. *Journal of Systems and Software*, 161:110460.
- [Olmedilla et al., 2006] Olmedilla, D., Rana, O. F., Matthews, B., and Nejdil, W. (2006). Security and trust issues in semantic grids. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.
- [Sica, 2006] Sica, G. T. (2006). Bias in research studies. *Radiology*, 238(3):780–789.
- [Sánchez et al., 2018] Sánchez, V. R., Ayuso, P. N., Galindo, J. A., and Benavides, D. (2018). Open source adoption factors—a systematic literature review. *IEEE Access*, 8:94594–94609.
- [Vargas et al., 2020] Vargas, E. L., Aniche, M., Treude, C., Bruntink, M., and Gousios, G. (2020). Selecting third-party libraries: The practitioners’ perspective. *CoRR*.
- [Wheeler, 2011] Wheeler, D. A. (2011). How to evaluate open source software/free software (oss/fs) programs. URL [http://www.dwheeler.com/oss\\_fs\\_eval.html](http://www.dwheeler.com/oss_fs_eval.html).