Thesis Project

Understanding Software Adaptation and Evolution
Degree project in progress

Author: Seva MEYER
Supervisor: Mauro CAPORUSCIO
Examiner: Johan HAGELBÄCK
Semester: HT2015
Subject: Computer Science
Abstract

Software maintenance is a significant part of software system lifetime. Software system lifetime incorporates many processes including software adaptation and software evolution. These processes collide into one another and create confusion as the boundaries that separate them are often difficult to distinguish. Knowing what exactly these concepts indicate and how they are related can bring simplicity to future development of adaptive systems. The following document presents a performed systematic literature review, which aims to outline the similarities and the differences of adaptation and evolution and further explain how they are related. The results of the study show that adaptation and evolution have become more entwined with growth of interest to self-managing dynamic software.

Keywords: Software adaptation, Self-adaptation, Software evolution, Evolutionary software.
Acknowledgments

To Mauro Caporuscio, my supervisor, for his guidance and support.
To Ola Flygt for all the support which made it possible for me to get to this point.
To Spyridon Artelaris for proofreading, long discussions on the topic, valuable ideas and support.
To my mother, Nika Meyer, for pushing me forward whenever I felt like giving up.
# Contents

1 Introduction .............................................. 1
  1.1 Introduction / Background ................................. 1
  1.2 Previous research ........................................ 1
  1.3 Problem definition ....................................... 2
  1.4 Purpose and research question ............................ 2
  1.5 Scope / Limitation ....................................... 2
  1.6 Target group ............................................ 3
  1.7 Outline ................................................. 3

2 Background / Theory ....................................... 4

3 Method ....................................................... 5
  3.1 Planning the review ....................................... 5
    3.1.1 Research questions .................................. 5
    3.1.2 Review Protocol ..................................... 5
    3.1.3 Exclusion and inclusion criteria ...................... 5
    3.1.4 Search strategy ..................................... 6
  3.2 Conducting the review ................................... 6
    3.2.1 Data extraction ..................................... 6
  3.3 Reporting the review ..................................... 7
  3.4 Reliability ............................................. 7

4 Empirical data ............................................. 8

5 Analysis/Discussion ....................................... 15
  5.1 Evolution vs. Adaptation ................................. 15

6 Conclusion .................................................. 17
  6.1 Conclusions ............................................. 17
  6.2 Further research ........................................ 17

References .................................................. 18
1 Introduction

Software development is nowadays a necessary part of the world. Software is everywhere and while its importance is growing rapidly day by day, the concern of quality grows as well. As a result, the idea of software engineering emerges as a crucial part of the software reality.

The maintenance of software is an important part of software engineering. L. Arthur criticizes the attitude of the software community towards software maintenance, claiming that this process must not be taken as a fault detection and solution, but rather as an evolutionary process. According to Arthur: "Maintenance is an important process as today most of the development is based on reusing the existing systems" [1]. Further, he believes that a software has to evolve according to market changes and users’ needs. According to his research and conclusions, the software evolution process has a category called Adaptive. This category implies that a software changes in order to adapt itself to environment changes. For instance, if a certain part of a system fails, the other parts have to adapt to the current situation.

At this point the problem of distinguishing these two processes arises. Some scientific studies refer to them as two separate processes, while others claim that adaptation is a sub-process of evolution. This paper aims to find the boundaries and bring some order and clarification to this subject. It will find common properties and differences between these two notions and summarise them in a way that aids to future research and software development.

1.1 Introduction / Background

Changes in the system and the environment surrounding it, changes in requirements and a big variety of users’ demands can require a software to adapt both statically and dynamically during its lifetime. Therefore, "Continuing evolution is reflected by a sequence of modifications to the objects. New objects are added. Others are changed or removed" [2].

Software continues to live after its deployment by adapting and evolving throughout its lifetime. Since nowadays it is convenient to reuse software instead of designing a new one, adaptation and evolution play a big role in the software world. Adaptive software is a software that can adapt to changes in the environment and inside the system. Such software can reconfigure and heal by itself if required. Software evolution is a process of software change. It is proved by time that software changes from release to release to fix system problems and satisfy new requirements.

To maintain and adapt a software can become an expensive and time-consuming work. It demands a lot of resources and that is the reason why adaptive and evolutionary software gains more attention nowadays. With a software that can manage and evolve itself without the need to be watched and guided, the costs of system maintenance can decrease dramatically.

1.2 Previous research

In 2013 Danny Weyns and Tanvir Ahmad introduced a systematic literature review on Architecture-Based Self-Adaptation [3]. In this paper they focused on:

1. The focus of research on architecture-based self-adaptation,
2. The claimed benefits of architecture-based self-adaptation,
3. The evidence that is provided for these claims.

However, this study was more specific, outlining the type of self-adaptation that is based on system architecture. It concludes that the quality of the research done on this area can be improved. It also highlights some claims and evidences of those claims about self-adaptation.

In 2010 Breivold H.P., Chauhan M.A. and Babar M.A. performed a systematic review of Studies of Open Source Software Evolution [4]. This study concentrated on creating an overview on existing studies and defining metrics that are used for evolution in this kind of systems.

However, despite all the effort done by the various researchers, a clear definition of the notions, as well as an examination of the boundaries between software adaptation and evolution, has yet to be found. Besides this, most of the papers focus on a particular type of adaptation and evolution. This thesis project aims to compare and evaluate the metrics of the concepts, which is a key difference from all other studies.

1.3 Problem definition

The main problem arises when it comes to the definition of self-adaptive systems. Autonomic, self-adaptive and evolutionary software is a main discussion topic at a number of significant software conferences. However, these concepts are always confused, considered to be the same or to be completely separate processes. It can be confusing for a developer to know what type of system in fact is being built, which properties are significant to focus on and how the maintenance of a system has to be made.

1.4 Purpose and research question

The purpose of this research is to define self-adaptive and evolutionary systems, taking into consideration research studies that have been performed on these subjects. After studying a great amount of articles it became clear that essentially distinguishing evolutionary software is a rather complicated and tedious process. It is also not possible to say with 100% certainty what a self-adaptive system is. Now it comes to the point where it is important to summarize all available definitions, find out all common properties (characteristics) and put an end to this endless discussion. Concluding all earlier mentioned arguments, significant questions of this report are:

1. What is a software adaptation?
2. What is a software evolution?
3. What are significant differences and commonalities between adaptation and evolution of software?

1.5 Scope / Limitation

This paper is not intended to investigate the problem deeply. It does not concentrate on 'how' of adaptive and evolutionary software. This study will try to highlight the important characteristics of adaptive and evolutionary software and provide a theoretical overview, reasoned by a set of former researches.
1.6 Target group

There are two main audience groups that this research is performed for:

- Self-adaptive and evolutionary software developers
- Self-adaptive and evolutionary software researchers

Software engineers that are expected to design and implement a system that is able to manage itself during run-time and can be responsive to any changes in the environment and in the system itself, can face a problem in understanding what such software essentially is. On the other hand, scientists performing their studies find it easier to carry out their research when the topic is more specific. Thus knowing exactly what one or the other term means can bring clarity to a given task. Moreover, this research is a concrete foundation for later researches on the given theme.

1.7 Outline

Section 2 provides a background of the problem. Here a basic theory on researched area can be found. Later, in section 3, the method used to conduct the research is described. Section 4 lists obtained results in a form of a table and several charts. Section 5 analyses and discusses the results provided in section 4. Finally, the sixth section gives a conclusion derived from the discussion.
2 Background / Theory

This chapter introduces the background of the problem researched in this thesis and emphasizes the theory behind the concepts.

In 2001 IBM stated in its manifesto that today the main concern in software engineering is a complexity of already existing software systems. It is fairly obvious considering the fact that once a software is implemented, it is being maintained, re-configured, adapted to new requirements throughout its entire life cycle. The more it is modified, more lines of codes are added, more enormous the models become, and more complex becomes the design of the system. To handle all these changes and to be able to continue the maintenance, companies have to spend countless numbers of resources, where money and human resources are among them. Millions of lines of code and thousands of interconnected complex systems have already become a big problem for programmers and designers. A human being cannot easily manipulate such systems any longer [5]. Autonomic systems can be seen as a good solution for complex systems. This idea was presented by a vice senior of IBM in 2001 and has become a widely discussed matter in IT society [5].

Changes in a system and the environment surrounding it, changes in requirements and a big variety of users’ demands can require a software system to adapt both statically and dynamically during its lifetime. As a result, the system has to adapt to all changes that occur and this adaptation can take place at any phase of software life time and at any level of the system. It can be triggered by various events and can lead to numerous results. It is worth mentioning that most of the changes require to stop the software run-time process. Therefore there is a big demand on software that not only can manage itself, but can also maintain itself during run-time and self-adapt to unpredictable conditions. A system is considered to be self-adaptive if it owns the ability to adapt itself to outside changes in the environment, requirements, etc., to modify its behavior if required, to re-configure itself when changes in hardware take place and to heal itself in case of emergency [6].

Self-adaptation is a software that can ‘think’ in terms of functionality and adapt to any kind of situation. This process does not come by itself, it is expected to lead to evolution of the software. The Online Oxford Dictionary proposes two main definitions of evolution:

• “The gradual development of plants, animals, etc. over many years as they adapt to changes in their environment” [7].

• “The gradual development of something” [7]

Besides, adaptation is mentioned as one of many synonyms of evolution. Software evolution is a process of software change throughout the maintenance period, when it evolves to a better state, and improving and fixing the performance during its lifetime as stated by L. Arthur [1].

These facts lead to a question:

• Are software adaptation and evolution two different processes tightly interconnected or are they completely identical?

If it earlier was crucial for software developers to be capable of managing such changes manually, understand the architecture, design and the code of the existing system, then today it is more important for engineers to succeed in creating a self-adaptive, self-managed, autonomic and evolving software. However, the task is not conspicuous as the concepts are quite easy to be confused and a decent clarification is required.
3 Method

This chapter describes the method used to perform the research. Initially it was decided to execute a systematic literature review (SLR) in order to gather the required data and perform the expected analysis on the topic. This methodology is convenient for collecting, comparing and analyzing data from primary studies as it helps to control the process of the study, without deviating from its chosen course and wasting time on wrong information. Moreover, the method can become a strong basis for further, more sophisticated research. It can also bring clarity to the studied concepts and identify missing or existing factors necessary for the further development of research. Kitchenham refers to systematic literature study as a secondary study. Following her guidelines SLR has three main stages: planning, conducting and reporting the review [8].

3.1 Planning the review

Two crucial points of this phase are research questions definition and a review protocol specification.

3.1.1 Research questions

Identifying the research questions is a crucial part of the review planning phase. Well defined, reasonable questions reduce the risk of publication bias, broadly discussed by Kitchenham [8]. This leads to simplicity in data extraction and analysis. The research questions are stated in section 1.4.

The two first questions are expected to bring clearness to these concepts and emphasize the main features that define their purpose. The third question is devoted to the main reason for this study. Answering this question can bring the desired clarity to the divergence of these concepts.

3.1.2 Review Protocol

The protocol of the review is created after the main research questions are specified. This document provides a better focus on the task and defines all inclusion and exclusion criteria used during the research. The research questions are also included in this document. The structure of the protocol is borrowed from Kitchenham [8]. Consequently it contains the information about the research strategy, like exclusion and inclusion criteria, research questions, research background and data extraction strategy. The protocol was written by the researcher and approved by the research supervisor. Some corrections were done during the later phases, as the strategy of the research was changed.

3.1.3 Exclusion and inclusion criteria

The inclusion criteria that used are:

- Articles that discuss software evolution and adaptation in detail.
- Papers that provide a theory of evolution and adaptation.
- Studies that are not older than twenty years e.g. 1995-2015.
The latter criteria was not followed along the whole study process as several articles of older publication date were discovered and taken into account due to the important contents.

Following, the selected exclusion criteria were:

- Articles conducted on Autonomic and self-managing systems.
- Papers that do not cover the software engineering area.

The first criteria was specified after several articles were studied. The concept of autonomic software is broad and therefore provides a general idea of evolution and adaptation, nevertheless the articles read were included into the resulting paper, as they provided useful information on both research aspects.

3.1.4 Search strategy

There were two main digital libraries that were exclusively used for article search:

- IEEExplore
- ACM Digital library

The keywords used for the searching were: “adaptive software”, “self-adaptive software”, “evolutionary software”, “software evolution”, “software maintenance” and "autonomic systems". The study that played a significant role in pointing out relevant articles was [9]. The studies were selected according to the criteria described in 3.1.3 section. Initially, studies having one or several key words in the title were chosen. As the study progressed, papers were chosen and approved on account of their abstract. If the keywords and abstract of an article were seen to be relevant, the papers were chosen as candidates for the next phase, otherwise they were discarded. However, more articles were not included even after passing through the search phase, as their content did not prove the relevance of the paper or the information given by the article was not useful for the conducted research.

3.2 Conducting the review

One of the goals of this phase is a selection of primary studies to be used in the review. The criteria used to select the studies are described in section 3.1.3.

3.2.1 Data extraction

During the second phase of the study certain parameters which provide the results for this study, were revealed. These parameters give a brief and clear definition of adaptation and evolution properties. The properties were extracted by using the 5W method [10]. Thus, the information inherent to evolution and adaptation is extracted by answering the following questions:

- Why – Specifies the goal of the process. Adaptation and evolution take place in order to preserve either functional or quality requirements.
- When – Both processes take place in a particular period of time or at a specific moment. Hence, an adaptation and an evolution can happen during run-time, implementation, deployment, compile time or along the lifetime of a system.
• What – Adaptation and evolution influence a software in two major ways: they either change the behavior or the structure of the system. Most of the times, behavior modification is caused by changes in the source code, while the structure of the system is affected by the restructuring of the architecture.

• Where – Here the layer of the system is mentioned. Both source code and the architecture of software can be changed by any of these two discussed processes.

• Who – A software can be maintained by a software engineer. However, many software systems are intended to be self-managing which makes it emergent and does not require a coordinated work from a system developer.

3.3 Reporting the review

A report is an integral part of the study, and as it is in this phase, all the obtained data from the conducted review is collected and put together. By thorough analysis, all the parameters are calculated and presented in a more acceptable form for future researchers/readers. The report is divided into two major parts:

• Empirical Data – all results obtained are presented in this chapter

• Analysis/Discussion – this paragraph is intended to provide analysis of the results presented in the previous chapter and a main discussion of the analysis that leads to a final conclusion

3.4 Reliability

The data collected and presented in this report is a result of a detailed study on various reliable sources. There are several Software Engineering books that were used, such as Software Evolition: The Software Maintenance Challenge [1] and Successful Evolution Of Software Systems [11]. Along with the books numerous papers from conferences like IEEE International Workshop on Program Comprehension [12], 4th International Workshop on Principles of Software Evolution [13], ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS) [14], and International Conference on Information Technology, Computer Engineering and Management Sciences (ICM) [15] were included.
4 Empirical data

This chapter depicts all results derived from the literature study and calculations of the results. A reader can clearly see the outcomes of the data extraction and follow the process of calculations.

In order to carry out a detailed study, 28 papers were selected and examined. 15 papers mention the adaptation of software, its settings and distinctive properties. 17 articles include discussion on the evolution and growth of software, its needs and its specific characteristics.

Table 4.1 contains all the information discovered in the process of the study. It outlines why, when and where adaptation and evolution of software take place. Moreover, a reader can find out what aspect of software is affected by any of two processes and by whom adaptation or evolution is performed. The characteristics are distributed over the columns thus they are clearly separated. The last column contains a weight of each paper read. This number is calculated by the formula:

\[ \text{Weight} = \frac{\text{Citations}}{\text{PresentYear} - \text{PublicationYear}} \]

The weight of the paper defines its value in today’s software engineering community and the reliability of the study.

As a result of calculations, the numbers of papers supporting a particular claim are viewed as a percentage. This data can be seen on charts 4.1, 4.9, 4.7, 4.2, 4.5, 4.3, 4.4, 4.6, 4.8, 4.10.

![Figure 4.1: Adaptation. Why?](image)

Figure 4.1 shows that in 80% of the papers the authors claim that adaptation takes place in order to maintain quality properties, while only 20% state that adaptation takes care of both functional and non-functional requirements. None of the papers mention functional requirements being preserved alone.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2 [16]</td>
<td>Adaptation</td>
<td>Extra-functional</td>
<td>Run-time</td>
<td>Behavior</td>
<td>Architecture</td>
<td>Emergent</td>
<td>74.9</td>
</tr>
<tr>
<td>4 [18]</td>
<td>Adaptation</td>
<td>Extra-functional</td>
<td>Run-time</td>
<td>Attributes/Behavior</td>
<td>Multilevel</td>
<td>Emergent</td>
<td>111.2</td>
</tr>
<tr>
<td>10 [21]</td>
<td>Adaptation</td>
<td>Extra-functional</td>
<td>Run-time</td>
<td>Behavior</td>
<td>Source code</td>
<td>Emergent</td>
<td>4</td>
</tr>
<tr>
<td>11 [22]</td>
<td>Adaptation</td>
<td>Extra-functional</td>
<td>Run-time</td>
<td>Structure</td>
<td>Architecture</td>
<td>Emergent</td>
<td>0</td>
</tr>
<tr>
<td>17 [27]</td>
<td>Evolution</td>
<td>Functional</td>
<td>Development</td>
<td>Structure</td>
<td>Source Code</td>
<td>Coordinated</td>
<td>1.8</td>
</tr>
<tr>
<td>19 [28]</td>
<td>Evolution</td>
<td>Extra-functional</td>
<td>Development</td>
<td>Structure</td>
<td>Architecture</td>
<td>Coordinated</td>
<td>2.8</td>
</tr>
<tr>
<td>21 [15]</td>
<td>Evolution</td>
<td>Extra-functional</td>
<td>Run-time</td>
<td>Behavior</td>
<td>Multilevel</td>
<td>Coordinated</td>
<td>0.3</td>
</tr>
<tr>
<td>22 [29]</td>
<td>Evolution</td>
<td>Functional/Extra-functional</td>
<td>Lifetime</td>
<td>Structure and behavior</td>
<td>Multilevel</td>
<td>Emergent</td>
<td>32.9</td>
</tr>
<tr>
<td>24 [30]</td>
<td>Evolution</td>
<td>Extra-functional</td>
<td>Lifetime</td>
<td>Structure</td>
<td>Architecture</td>
<td>-</td>
<td>0.8</td>
</tr>
<tr>
<td>29 [33]</td>
<td>Evolution</td>
<td>Functional/Extra-functional</td>
<td>Development</td>
<td>Behavior</td>
<td>Architecture</td>
<td>Coordinated</td>
<td>40.2</td>
</tr>
</tbody>
</table>

Table 4.1: 5W of Software Adaptation and Evolution (the weights were calculated on the 12th of May, 2015)
Figure 4.2: Adaptation. When?

Figure 4.2 depicts that staggering 70% of the studies highlight the fact that adaptation happens during the software run-time, whereas 15% claim that it is a development phase, 10% that it is a deployment phase and only 5% state that adaptation might take place during compilation. None of the papers sees adaptation as a process following a software throughout its lifetime.

Figure 4.3: Evolution. Why?

The reason of software evolution (figure 4.3): 75% of the articles see it as a way to preserve quality requirements, while only 6% claim that functional requirements are the source of the process. 19% of the papers see it as a combination of both.
According to 30% of the authors, evolution takes place during run-time of software, while other 40% see it as a process happening during the development phase. 20% of papers prefer the term lifetime to specifying a certain phase and only 10% see evolution as a part of the deployment process (figure 4.4).

In figure 4.5 62% of the studies suppose that adaptation changes or adapts behavior of software, only 15% state that adaptation affects both behavior and structure. 23% of the studies see it affecting structure.
In 47% of papers evolution affects the structure of software as is presented in figure 4.6. Almost equivalent number of papers (41%) propose a software behavior as a modification target. Only 12% propose that both structure and behavior are affected by evolution.

According to figure 4.7, adaptation is a multilevel process according to 50% of the studies. 36% claim that it happens on architectural level, 14% see changes going on in source code.
Software evolves on architectural level which is claimed by 50% of studies. 33% see evolution as a multilevel process and the remaining 17% propose that evolution influences the source code of a system. These results are displayed on the figure 4.8.

Figure 4.8: Evolution. Where?

Figure 4.9: Adaptation. Who?

Figure 4.9 makes obvious that adaptation is seen as an emergent process by 67% of the researches, 27% argue that adaptation can be both emergent and coordinated and 6% say that it is coordinated.
However, evolution is declared coordinated by 67% of studies, while 27% of papers claim it is emergent. The remaining 6% state that it is both emergent and coordinated according to figure 4.10.
5 Analysis/Discussion

In this chapter general analysis of the results obtained during the research is presented. A comparison between software adaptation and evolution is illustrated and discussed.

5.1 Evolution vs. Adaptation

Figure 5.11 depicts certain properties that seem to be winning the quantitative competition between the different studies according to table 4.1. Besides, it demonstrates overlapping characteristics of adaptation and evolution. However, the weight of the papers can also lead to completely different results.

Research question 1. The weights of paper number 8 and 4 may serve as a sign of trust of the information provided by the authors. Their weights are the highest among other studies related to adaptation, and both agree that adaptation is a run-time, multilevel, self-driven process that has an influence on software behavior and preserves the qualitative properties of the system. This serves as an answer to the question of software adaptation definition, since it matches to the overall results.

Research question 2. The heaviest papers among the evolution studies (16 and 18) agree that evolution occurs during run-time on architectural level and it is affecting the structure of a system. However, they both propose the possibility that evolution can be emergent, which negates the results shown in figure 5.11. Concluding this and neglecting the numerical priority of papers, it is reasonable to assume that evolution can be performed without human intervention. Therefore, there is a possibility that evolution happens not only during development, for instance, through numbers of releases carried out by developers, but also during run-time as a result of software adaptation.

Research question 3. The results reviewed earlier clearly show that adaptation and evolution primarily happen in an effort to preserve the quality properties of software. Software systems can evolve and adapt during run-time, however evolution can happen throughout a number of releases, which implies that it is a coordinated process. Software adaptation is primarily seen as emergent. Many studies map evolution to adaptation, but only when it comes to evolution that is performed during run-time. In some cases both concepts are discussed closely to one another as if they are two parts of one. As an example, the work of M. Salehie states: "Static and dynamic adaptation, related to the temporal dimension of this view, are mapped to compile-time evolution and load-time/run-time evolution, respectively. For this reason, dynamic adaptation is sometimes called dynamic evolution" [18].

Also according to one of Lehman’s laws, software should be continuously adapted in order to satisfy the changing requirements e.g. evolve [29].

![Figure 5.11: Adaptation & Evolution](image-url)
A large number of papers states that adaptation is a run-time process. During run-time a software changes its behaviour, which will, consequently, affect the whole system. Some researchers refer to multilevel adaptation, some other do not focus on this aspect, which probably is an indicator of their general representation of this concept. In most of these studies it is complicated to put markers and highlight the boundaries on which level of software system adaptation starts on. Despite the fact that the authors do not clarify it; when the behaviour is being affected one can assume that the changes happen at source code level. In the meantime evolution also affects and preserves quality requirements or properties, it can also happen at run-time as well as through a number of releases. It is said to affect structure and even behaviour. Evolution takes place on architectural level and is mostly seen as a coordinated process.

However, many papers have already begun discussing self-evolutionary software. It is not surprising, because if adaptation happens during run-time and changes the behaviour, it consequently changes the source code. This results to the evolution of the software, since it is already changing during run-time to adapt itself. Studies like 2,6,17,19, 21, 20 (table 4.1) claim that evolution or adaptation is caused by changes in the surrounding environment or in the system itself. This leads to a proposal that evolution can be caused by adaptation since adaptation changes the system. Changes in source code can lead to changes in the architecture as well. As said by Keith H. Bennett "Strategically, progress in software architectures is crucial, so that we can extend and adapt functional and nonfunctional behaviour without destroying the integrity of the architecture in order to respond to unexpected new user requirements" [33]. If so, it means that adaptation leads to evolution. At the same time, if software adapts often then at some point there will be a need to change architecture, meaning to restructure the software. This implies another release process, which is assumed an evolution in major number of studies.

Architecture, structure and behaviour do change, and all of these changes can be the result of adaptation. Therefore, in many cases, adaptation can lead to evolution and this makes adaptation a small part of evolution. Evolution can happen separately from adaptation, but the fact that they are very similar cannot be disregarded. Some non-adaptive software systems can still go through the evolution process as they can change during run-time even if the system is not adaptive. However, in this case it is most probable that the change will be carried out by an administrator. All the mentioned factors propose that adaptation is a sub-part of evolution. If a self-adapting software is to be implemented, it is going to evolve on architecture level too, both with or without intervention of developer.
6 Conclusion

In this chapter reader finds all the conclusion based on the research conducted and results gained in the end. Suggestions on further research can be found here.

6.1 Conclusions

Software evolution and software adaptation possess vast areas for scientific research. These two aspects are crucial parts of software lifetime along with infinite number of other processes that constantly go on. Until today many studies on both concepts have been conducted and various outcomes have been published. According to the systematic literature review described in this report, software adaptation is often a process that maintains quality requirements during run-time of an application. It is capable of changing the behavior of software depending on changes in the surrounding environment or inside the system itself. Nowadays, most of the scientists focus on the concept of self-adapting systems that do not require human intervention, as it practically reduces the costs and is time-saving.

Software evolution, on the other hand, represents a process that can occur not only during run-time, but also during development phase. According to J. Kramer: "However, dynamic change, which occurs while the system is operational, is far more demanding and requires that the system evolves dynamically, and that the adaptation occurs at run-time" [16]. This leads to major changes in a system’s structure and sometimes behavior. The independence of this process from software developers is not broadly discussed yet, but it becomes more popular nowadays.

The study has shown that a self-adaptive system can become self-evolving, since changes in behavior may sooner or later require changes in structure. Until recently evolution and adaptation were not tightly dependent on each other, nevertheless dynamic evolution will need dynamic adaptation. This leads to a close interrelation between them in the near future.

6.2 Further research

Knowing basic properties of an adaptive and evolutionary software and answers to the 5W questions it is time to look deeper into the problem and start asking a question of how? Further research might look closer at a possible implementation of an adaptive and evolutionary software. Moreover, the present study can serve as a solid basis for future investigations. Such a project could prove the validity or refute the results presented in this paper. There is an ethical question that arises after the conducted research:

- Does humanity need an artificial organism that can adapt and evolve without human help and that can easily overcome humans’ abilities if desired?
References


