A FLEXIBLE METHOD, A RIGID ARCHITECTURE

Étienne Asselin[1] ***1773922*** and Vincent Rodier[2] ***1744784***

***Abstract*— A very important part of any software is without a doubt the architecture. Every software has an architecture, whether it was planned or not. The preferred solution is always to have a plan for the architecture that is able to fulfill every requirement of the software - reality is not always as bright. Meanwhile, agile software development techniques are gaining in popularity and discouraging people from planing up-front architecture. A project without any plan of architecture is a project set-up for failure - or is it? Is it possible to apply the Agile Manifesto philosophy all the way and not plan anything up-front? And if not, exactly how much planning is required? How to know exactly how much up-front architecture is needed?**

I. INTRODUCTION

Initially software development was only sequential following the cascade model. The steps were to list requirements, model a system architecture, implement, verify and ultimately maintain the software. This approach brought some benefits such as a complete documentation of the system, a project cost estimate from the start, the software was generally easier to maintain and the design followed a very specific standard. Despite the advantages of cascading development, it does not follow the reality of software development due to its lack of flexibility. Thus, in the early 2000s, agile development methods emerged.
Agile development follow four fundamental values:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Here, the dilemma is finding the middle ground between the initial development of a software architecture and the ability to respond to customer changes. The problem is the duality between the rigidity of the initial development of architecture and the flexibility of agile development. Indeed, it is very difficult to derogate the architecture of a program when a possible change occurs. In such a case, the risk of failure must be reduced.
This paper aims at helping people to make decisions regarding how much up-front architecture (if any) is needed to mitigate the risks failure associated with the lack of planning philosophy of agile development. In order to achieve this goal, these questions must be answered:

- Is a project without or very little plan of architecture a project set-up for failure?
- Is it possible to apply the Agile Manifesto philosophy all the way and not plan anything up-front?
- How to know exactly how much up-front architecture is needed if any?

These questions will be answered using previously performed research. In this paper, agile development and software architecture will first be explained. Then similar work and the method of research of theses will be exposed. Finally the results will be presented and detailed.

[1]É. Asselin is a student in Software Engineering at Polytechnique Montreal, Qc H3T 1J4, Canada
[2]V. Rodier is a student in student Software Engineering at Polytechnique Montreal, Qc H3T 1J4, Canada

II. BACKGROUND

This section will introduce some crucial knowledge to understand the whole paper.

A. Agile development

The term Agile Development was consolidated in 2001 following the writing of the Agile Manifesto. Agile development methods rely on an iterative and incremental development cycle. Agile development encourages several principles including prioritizing customer satisfaction, supporting change requests and often delivering operational versions. To achieve these principle Agile Manifesto encourages the reduction of the amount of up-front planning and design.

B. Software architecture

The software architecture is a model for the system and the project. First, the architecture is an artifact that allows early analysis of the system to ensure a design approach that will produce a system acceptable to the customer. Architecture is the primary support for system qualities such as scalability, performance, modifiability, security and cost reduction. So building an effective architecture can identify design risks and mitigate them early in the development process.

III. RELEATED WORK

To be able to write this paper, we search the *ACM Digital Library* and *IEEE Xplore* for work or research done in relation to this topic. We queried the library using words like ”agile development” and ”software architecture”. The follow section will review one by one and in chronological order the chosen publication that was taken into consideration to write this paper.
Waterman, Noble and Allan (2012) investigated how much up-front architecture was important and how much value agile practitioners valued it. They did a number of interviews with agile practitioners that have knowledge of software architecture to highlight practices that the industry employs in hopes of finding the right amount of up-front architecture needed in a project. After 11 interviews and one focus group

the researchers found out that using template architectures is useful because it reduces the amount of up-front architecture needed while being easier to adapt and change than a custom made architecture. They also stated that the experience of the team working on a project using agile development makes a difference. A more experienced team is more likely to make good strategic architectural decisions that are in line with the agile method and avoid needless documentation.

Waterman, Noble and Allan (2015) once again pursued their previous work to find how much up-front architecture is needed while exploiting an agile method of development. As follow up to their work in [1], they did another set of interviews with agile practitioners but this time, they interviewed 44 people in 37 semi-guided interviews. In this paper, the authors laid out 6 dimensions that were a threat to any software development project and they enumerated 5 strategies used by agile development to respond to those threat. The conclusion the team came up with in this paper is still in line with the first work they did [1], but this time focusing more on agile practitioners address problem that come with up-front architecture and agile development.

The work done by Highsmith and Cockburn (2001) in [3] depicts reasons why previous ways of working in the waterfall era don't work as they were intended in the real world. [3] describes how the agile development methods should prepare themselves for a project. [3] doesn't talk about how the architecture of a project suffers or benefits from agile methods though. It focuses more on explaining what exactly agile methods aim to do, how they do it and why they do it.

In this short yet interesting paper [4], Kruchten (2010) tries to demonstrates that there is actually very little conflict between using agile development methods like XP or SCRUM and still focus on early on an architectural design. The analysis of Kruchten in [4] is done by stating seven different dimensions that help identify why agile and up-front architectural planning seem in conflict and later answer them in a tutorial aiming at helping people find a balance between planning and doing work.

In [5], Nord and Tomayko (2006) begin by describing the view of agile practitioners stating that the majority of them see architecture planning as a very work intensive process that does not add any value. The authors also states that the architecture of any software projects reflects on the quality of that said projects and that is isn't exclusive to any development method. With an extensive analysis of the agile methods all the while identifying flaws, the authors of [5] show how agile development is not fitted to respond to all problems and how the lack of preparation can cause disastrous results sometimes. Nord and Tomayko (2006) conclude their work by showing point by point how architecture driven methods can not only help agile practitioners with their work but that it complements very well XP agile development method.

In their paper [6], Schramm and Daneva (2016) explore the implementation or SOA and agile software development. The subject the two authors set for in their research was to understand how SOA and agile development work together with three questions. Those three question help their methodology and helps guiding the work they did. Schraam and Daneva (2016) approached their subject with qualitative data in a well-known professional blogging platform publicly available. The idea behind choosing to study blogs is that by focusing on blogs written and were people who comment are practitioners they'd have a better understanding of real practices in the industry. After gathering the data, the two authors analyzed everything using the open coding and other technique. The results presented in this study are quite extensive. The authors reported three main categories that practitioners talk about a lot and then elaborated more on each one of them. In the end, it shows that SOA and agile development can work together and that utilizing certain concepts like collaboration can increase productivity and response to change.

Chen and Babar (2014) did an empirical study of the emergent architecture in a software while using agile methodology. The goal of this study was "to explore the perceptions and experiences of practitioners for identifying and building a taxonomy of the factors that can influence the emergence of a satisfactory architecture through continuous refactoring"[7]. The authors of this paper also used an interview and surveying method in order to gather information about their subject and all together, they interviewed 102 experienced software professionals from 13 countries and 6 continents. This study found 20 factors that contribute to the emergence of an architecture in the development of a software and they categorized these 20 factors into 4 category that they each explored more in detail. The results were that a satisfactory architecture can emerge from constant refactoring alone but that contextual factors play a huge role in these. Interestingly enough, they also found out that around 9% of that were interviewed have never encountered an emerging architecture that was not satisfactory.

Santos (2016) for his part focused exclusively on agile methods and what drives agile methods - why is it so popular. Santos states that simplicity is the what makes agile so great and what makes more software project successful. His problem with simplicity and agile methods though is that agile methods do not provide a good framework for applying simplicity. To explore his subject, the author divided his research in two part, beginning with a theoretical method followed by a second phase of industrial case study. The findings of this particular research is that agile methodology only encourage and emphasis simplicity but that still some concepts behind agility and simplicity are not enough understood by practitioners and researchers.

A better understanding of these concept would be helpful to improve agile software development.

In paper [9], I. Hadar, Sherman, E. Hadar and Harrison (2013) focus on the challenges associated with agile development methods and architecture documentation. The authors lead a research inside a large internationally distributed firm that utilized both agile and none-agile methods. Their objective was to create an architecture specification document format that can coincide with agile way of things. The finding of this team were made possible by a number of interview of architects and by analyzing artifact produced by different project inside the firm. The conclusion of the team was that the best results are achieved by making "an abstract architecture specification document using a supporting specification tool for creating a short and focused architecture document" [9]. This makes the architecture easy to understand and easy to modify without taking too much time.

IV. METHODOLOGY

The methodology we used decays into several distinct stages. First, we found the purpose of the literature review. Here, this paper aims at helping people to make decisions regarding how much up-front architecture (if any) is needed to mitigate the risks failure associated with the lack of planning philosophy of agile development. This first step allowed us to clearly identify the purpose of this paper so that the reader will know explicitly what is the subject. The second step is crucial since we are a team of two to write this SLR. It was essential that we were fully clear on the detailed procedure to be followed. This requires both a detailed written protocol document an agreement to how we evaluated each literature. The procedure consisted of analyzing each of the sources according to 5 criteria on which we give them a score out of 10. These 5 criteria were:

- Reliability of the source
- Reputation of the author
- Objectivity of the information
- Accuracy of information
- Actuality of the information

As a result of the evaluation each participant retains only papers that scored above 35 out of 50 and added them to a potential nomination. Then the partner must evaluate the papers retained and keep only those who have a grade greater than 35. The third step was to do literature research. We queried two databases: *ACM Digital Library* and *IEEE Xplore* using therms like "agile development" and "software architecture". Another method to find more source for our study was to do an activity that is called snowballing. We actually did both kind of snowballing (forward and backward). The main purpose of this step was to expand our sample of potential studies. The next step was to synthesize the studies. The primary goal was to combine the data extracted from studies using qualitative and quantitative techniques.

V. RESULTS

VI. DISCUSSION

VII. THREATS TO THE VALIDITY

VIII. CONCLUSIONS

REFERENCES

[1] M. Waterman, J. Noble, and G. Allan, How much architecture? Reducing the up-front effort, Proc. - Agil. India 2012, Agil. 2012, no. 2007, pp. 5659, 2012.

[2] M. Waterman, J. Noble, and G. Allan, How much up-front? A grounded theory of agile architecture, Proc. - Int. Conf. Softw. Eng., vol. 1, pp. 347357, 2015.

[3] J. Highsmith and A. Cockburn, Agile software development: The business of innovation, Computer (Long. Beach. Calif)., vol. 34, no. 9, pp. 120122, 2001.

[4] P. Kruchten, Software architecture and agile software development: a clash of two cultures?, 2010 ACM/IEEE 32nd Int. Conf. Softw. Eng., vol. 2, pp. 497498, 2010.

[5] R. L. Nord and J. E. Tomayko, Software architecture-centric methods and agile development, IEEE Softw., vol. 23, no. 2, pp. 4753, 2006.

[6] M. Schramm and M. Daneva, Implementations of service oriented architecture and agile software development: What works and what are the challenges?, Proc. - Int. Conf. Res. Challenges Inf. Sci., vol. 2016August, 2016.

[7] L. Chen and M. A. Babar, Towards an evidence-based understanding of emergence of architecture through continuous refactoring in agile software development, Proc. - Work. IEEE/IFIP Conf. Softw. Archit. 2014, WICSA 2014, pp. 195204, 2014.

[8] W. Santos, Towards a better understanding of simplicity in Agile software development projects, Proc. 20th Int. Conf. Eval. Assess. Softw. Eng. - EASE 16, pp. 14, 2016.

[9] I. Hadar, S. Sherman, E. Hadar, and J. J. Harrison, Less is more: Architecture documentation for agile development, 2013 6th Int. Work. Coop. Hum. Asp. Softw. Eng. CHASE 2013 - Proc., pp. 121124, 2013.