Determining the Suitability of Agile Methods for a Software Project

R.W. van Dijk
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
r.w.vandijk@student.utwente.nl

ABSTRACT
Nowadays a lot of different software development methodologies (SDMs) are available, but little academic supports exists which aids practitioners in deciding which methodology suits their project best. In this paper a framework is proposed with a different view on how to approach situational methodology suitability for the domain of agile methods. This is done by carefully constructing a definition for suitability in this context by conducting a literature study from the viewpoint of contingency theory and by analyzing agile methods in general to find relevant contingency factors regarding agile methods. The conclusion is that suitability is a scale where complete suitability only exist in an ideal situation. Two contingency factors are distilled to determine the suitability of a project for an agile method: the influence of limitations of agile methods and the organizational capability to handle agility, which is determined by the culture values of the organization and the individual capabilities of the team. This is taken together in a framework which enables a practitioner to assess the project situation and which provides guidance for a good line of reasoning when deciding whether an agile method is suitable or not for a project.

Keywords
Project characteristics, agile software development, suitability framework

1. INTRODUCTION
In the beginning years of software development, in the 1950’s, software was built as engineers would build hardware. Engineers would only start producing code when the design was carefully thought-out and calculated through. When in the 1960’s and 1970’s software became more complex, the need for formality and structure gave rise to the well-known waterfall-model by Royce [3]. Requirements-driven engineering was already an established approach in the late 1950’s, but the Royce’s waterfall model introduced the concepts of “confining iterations to successive phases”. Before moving on to the next stage in development, all artifacts in the previous stage required careful verification and validation. Royce’s model became one of the most cited examples of a traditional software development methodology [3]. When time progressed, so did technology, the market and the field of software development methodologies. After the waterfall model, other methodologies emerged dealing with the corresponding demands and developments over time; the demands of practitioners changed, which caused the methodologies to evolve [20]. In the 2000’s, the term agile software development came around, introducing a more lean and light-weighted development methodology compared to the traditional approaches [14].

The reason agile development became popular was because many researchers and practitioners called for a method which would replace the heavy-weighted, increasingly unsuccessful, and bureaucratic traditional methodologies. Agile stood for a more flexible way of developing software, allowing practitioners to adapt better to changes in the environment. In 2001 the Agile Alliance was formed, which published the Agile Manifesto, providing a list of 12 principles to establish a vision for the modern software industry could follow. Examples of these principles are “Deliver working software frequently”, “Simplicity is essential” and “Build projects around motivated individuals”. These 12 principles were derived from different software development methodologies (Scrum, Crystal, XP) which considered themselves ‘agile’. These methods are now well established and accepted methods in software development. [6].

While the agile methods became popular and widely adapted, supported by many examples of success stories, little academic proof exists validating their claims. In 2008 Dyba and Dingsoyr conducted a very extensive literature research to see what the current state was of the agile research field and what gaps still were not answered adequately [7]. They evaluated 1986 studies on agile software development from which only 36 were seen as having an acceptable level of credibility and relevance. They showed that empirical research about the actual problem domain of agile compared to other methodologies, validating the claims made by agile methodologies, was still lacking [7].

As Glass (2004) describes, good scientific research about when to apply a certain methodology in which situation, is not the main research topic of today’s scientific literature [9]. Current research is more focusing on comparing the different methodologies, pointing out differences with for instance the more traditional approaches or developing new methodologies, but without proper validation.

Conboy and Fitzgerald (2004) conclude that it is pointless to argue whether agile methods are superior to traditional methods. An agile method is only better when there is a need to be agile and the organization is capable of being agile. What practitioners should do instead of just going agile, is making an assessment of the organizational capabilities and the agile needs before selecting a methodology [6]. The problem here is
however that there is not enough academic support to do this properly.

1.1 Problem statement
The main goal of this study is to design a framework based on the current scientific knowledge available, which helps software practitioners determine whether a software project is suitable for an agile method. This framework should be able to, given a certain project in a certain context, map this project to the characteristics of agile development and give a judgment about the suitability of agile methods. The scope of the research is to look at software projects and agile methods in general, and not at different types of software or at the different types of agile methods available in detail. This framework is to be validated by researchers as future work; the scope of the study here is only to design such a framework.

1.2 Research approach
Originally the idea was to design this framework from the perspective of contingency theory and situational method engineering (SME). Contingency theory says that there does not exist one method for every situation. SME is an approach which supports that perspective and requires the best suitable methodology for an organization or its projects to be designed or selected from pieces of already available methods [11] [6]. In other words, a good and useful framework to help practitioners decide whether or not to use an agile method for their project should not be a framework which is bended to apply to all situations, but flexible in a way that it can include different viewpoints from already available methods when the situation requires that.

Then, to determine if a method is suitable for a project, the framework should include (a) project characteristics, (b) agile characteristics and (c) the relationship between (a) and (b) and a way of judging them to make a decision.

To achieve this, the following main question is formulated:
**Main question:** How can be determined if a certain software project is suitable for an agile software development methodology?

While conducting the initial literature search it became clear that the concept behind suitability of agile methods had to be more than a mapping between context/project parameters and characteristics of agile methods. The conclusion up until then was that this viewpoint was not a good perspective for the intended framework of this study, because there were little material available stating clear definitions of suitability and relevant project parameters.

Therefore, the viewpoint of this study will start from the question around how suitability can be expressed, comparing multiple options with the definitions of agile methods and construct the relationship between ‘suitability’, ‘software project’ and ‘agile methods’ carefully. To achieve this and to answer the main question, the following sub-questions are formulated:

1) How is suitability of a software development methodology for a project defined?
2) What is the definition of an agile method?
3) What is the relation between the definition of an agile method and the suitability?

The study is done according to a structured approach derived from the method described by Webster and Watson [18], using the WebOfScience and Scopus databases. Later on in the study, when the basic literature is collected, additional literature is found using forward and backward referencing.

The paper is structured as follows: in section 2 “suitability” will be defined, answering the first sub-question. Then, agile methods will be analyzed to answer the second sub-question in section 3, and at the end of section 3 the relation with suitability and agile methods will be explained. Finally in chapter 4 the results will be combined in a framework to give an answer on the main question.

2. SUITABILITY

2.1 Contingency theory
Classical contingency theory says that design decisions depend on the conditions of the environment. In order words, there does not exist a one-size-fits-all, different circumstances require a different approach. In every situation different ‘contingency factors’ play a roll which have dynamic relationships with each other and therefore should be taken into account altogether when making design decisions. [16]

Scott and Davis define suitability of an organizational form as follows: “suitability is determined by the goodness of fit between organizational form and the diverse environments to which they relate” (Scott & Davis, 2007:108).

Weill also refers to suitability in terms of “fit” and states that the better the “fit” among contingency factors, the better the performance of the influenced organizational sub-unit. In his study he states that there is a difference between factors influencing the performance of an organizational sub-unit and the variables by which the performance is measured or expressed.

A typical early representation of contingency theory is shown in Figure 1 (derived from [19]).

![Figure 1. Typical contingency theory representation](image)

This model is widely criticized and considered naïf because of its linear approach. Weill states that when in the unlikely situation where ‘fit’ and performance can be clearly defined and determined, their relationship is to be judged subjectively since deterministic approaches yet only resulted in inconsistent models and unproven assumptions [19].

2.1.1 Contingency theory interpreted
Translating the above statements to project methodology selection, suitability or fit of a project methodology is not defined by the anticipated degree of success of a project (the performance), but by contingency factors related to the environment of the project. The question now is what these contingency factors are in the context of selecting a software development methodology (SDM) and how they are related.

2.2 Contingency factors
Determining which contingency factors play a role when choosing a project methodology is not only relevant for information system development (ISD), but for project management (PM) in general.

In a recent extensive literature study by Howell, five themes of contingency factors affecting projects in general were identified from prior research: uncertainty, complexity, team empowerment, criticality and urgency. All themes encompass
different factors, which are shown in Table 1. Important to note is that the contingency factors stated in Table 1 are combined factors from several disciplines.

Table 1. Contingency factors by theme [12]

<table>
<thead>
<tr>
<th>Theme</th>
<th>Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncertainty</td>
<td>Goals, methods, market, technology, change, external influences</td>
</tr>
<tr>
<td>Complexity</td>
<td>Organizational complexity, environment complexity, scope, sophistication, diversity, project/team size</td>
</tr>
<tr>
<td>Team empowerment</td>
<td>Corporate and national culture, team composition, communication capability, team size and dispersal, team skill and experience</td>
</tr>
<tr>
<td>Criticality</td>
<td>“the effect upon organization or individuals of project failures”</td>
</tr>
<tr>
<td>Urgency</td>
<td>Absolute speed, time pressure</td>
</tr>
</tbody>
</table>

But how is known which factors are relevant for determining suitability? From this list one could derive relevant factors for software development, but the fact remains it is then only known which factors could play a role, not how and to what extent. Some deeper analyzes is therefore required.

2.2.1 Contingency factors and suitability

From the perspective of contingency theory, a project methodology is suitable for a certain situation if the contingency factors ‘fit’ the situation. But it remains unclear how to determine or measure this ‘fit’ or whether we should measure this at all. Some factors like ‘team skill’ or ‘culture’ can be observed and labeled with observable values [4], but when reviewing the literature there is no empirical research which links observable values of contingency factors to methodology selection.

An example of an alternative approach is a study by Kettunen and Laanti (2004). Their research question was “How can the project manager avoid typical project problems by selecting an appropriate software process model, based on the project situational factors?” Their focus was on describing project situations in abstract and general terms and map them to methodologies. A practitioner then could find the most applicable entry in the table for his situation and read about the pros and cons of different SDMs for that specific situation [15]. The problem with this approach however is that because of the abstract expression of situations, multiple entries of the table could apply to some unique situation, e.g. the table is not explicit. Also, this approach is based on the assumption that the decision regarding the best methodology to use is based on evaluating risk factors or possible problem factors which could affect the success of the project in a negative way. The table of Kettunen and Laanti maps a large set of these possible factors and problems to a set of SDMs and describes how good or bad the methodologies deal with them. This is however in conflict with the contingency theory, where performance and ‘fit’ have their own contingency factors and their relationship should not be approached deterministic.

On the other hand, there is something to say to include risk factors and possible project problems, since selecting a project methodology is something a manager or project owner does before a project is initiated. An assessment of the best viable methodology option with respect of risk factors therefore sounds the most logical thing to do. The assumption here however is that the success of a project is affected by the suitability of the applied methodology. From project success research we know that project success is a far more complex systems model [2] and the right software development methodology as a success factor has more to do with the

![Figure 2. Comparing the life-cycle, project management and concrete guidance support [1]](image-url)
amount of experience with a certain methodology than the methodology itself [8].

The reason for this is that during the project’s life cycle the used methodology can be adapted and changed to fit the situation. Rather than the suitability itself, the capability of the manager and the involved personnel in executing this tailoring affects the success of a project, not the methodology itself. A methodology merely offers day-to-day practices and guidelines, suitability therefore should be viewed from the perspective of ‘what a methodology can do or explicitly cannot’.

### 2.3 Conclusion

Deciding whether a methodology is good in a certain situation therefore is more complicated than answering a yes or no question. Suitability is not a term which is expressed in observable, quantifiable factors, but rather is a scale where complete suitability only exist in an ideal situation. In all other situations, suitability of a methodology in the context of a project is determined by the capabilities of the methodology and the relation with the requirements of the context in which the project exists.

We now know how to express suitability; in the next sections we will apply this. In section 2.2, the question “what factors are relevant?” has been rose, but this question has not been answered. Next will be looked at agile methods to find out how suitability in terms of contingency factors relevant for agile methods is expressed, without forcing on a deterministic viewpoint.

### 3. Agile Methods

“Agile methods” in the context of software development methodologies (SDMs) is a container term for all methods on which the Agile Manifesto as defined by the Agile Alliance is based. There is no certificate or standard to determine whether a methodology is agile or not, they are considered agile methods as a general consensus. While there are a numerous of methods which are considered agile, they still differ from each other in terms of for example project life-cycle span and day-to-day practical detail.

When comparing them on the day-to-day practices and project life-cycle coverage, one would get a model proposed by Abrahamsson et al (Figure 2).

He analyzed the different agile methods and concludes that: [1]

1. All agile methods cover different phases of the life-cycle
2. True support for project management is missing
3. Focus is on abstract principles instead of concrete guidance
4. Universally solutions dominate situation appropriate solutions
5. Empirical evidence is limited, most research is on a conceptual level

From this is can be concluded that analyzing them on implementation level to determine a common ground for expressing suitability is not a good option, since they differ too much. Rather the focus must be on what “being agile” implicates and their underlying principles and from there look for contingency factors.

### 3.1 Agility

Agile methods are considered to be “agile”, or have some degree of “agility”. First of all, agility is not an a priori characterization of agile software development methods, but rather an emergent property [13]. Conboy carefully reconstructed the definition of agility based on multidisciplinary sources of literature and came to the following definition:

Agility is: *The continual readiness of an ISD (Information System Development) method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment.* [5]

A SDM is then agile if its characteristics match the above definition. Conboy refines that as follows [5]:

1. To be agile, an ISD method component must contribute one or more of the following:
   a) Creation of change
   b) Proaction in advance of change
   c) Reaction to change
   d) Learning from change
2. To be agile, an ISD method component must contribute one or more of the following and must not detract from any:
   a) Perceived economy
   b) Perceived quality
   c) Perceived simplicity
3. To be agile, an ISD method component must be continually ready i.e. minimal time and cost to prepare the component for use.

Conboy states that in order to claim that an organization uses an agile method as their development methodology or to claim as “being agile”, the organization must have adopted an agile method properly in the sense that the methodology used must comply with the above definition. An organization does not have to implement one of the agile methods completely, but can use different aspects of different methods and combine them to achieve this.

Next will be looked if a common ground can be found by looking further at the underlying principles of the agile methods.

### 3.2 Agile principles and assumptions

In a study by Hansson et al, the agile principles from the manifesto are broken down to four values: [10]

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

Turk et al analyzed these further in their study about the underlying assumptions of agile methods and came up with the model in Figure 3.
They conclude that the agile methods in their current form have assumptions which are based on the basic principles as defined in the manifesto and the values derived from them. However, these assumptions do not hold in all organizational environments and therefore lead to limitations.

They define the following list of limitations [17]:

1. Limited support for distributed development environments.
   Agile methods assume that face-to-face communication with developers and customers is the most productive way to interact. When working in a distributed development, this obviously becomes a problem. The alternative is to communicate with video-conferences, schedule meetings in advance due to time-zone differences and such, but these adjustments undermine the agility. Also, co-location requires documentation standards to keep parties involved and on the same track, which requires formal processes to be introduced.

2. Limited support for subcontracting
   Subcontracted tasks are often explicit defined in contracts and accepted by bid. Before coming up with a bid, a subcontractor presents a plan containing the process steps in detail, cost estimations and milestones. Agile methods assume that requirements can develop along the way, which conflicts with formal subcontracting.

3. Limited support for development involving large teams
   Agile methods lean on frequent face-to-face informal meetings with little management hustle which only works efficiently in smaller teams where little lines of communication have to be maintained. A project which requires a large team can be managed agile, but only local. The large overall team then for example has to be managed more formally with strict guidelines for documentation.

4. Limited support for building reusable artifacts
   Agile methods tend to focus on solving a specific problem. Therefore, little documentation (no more than the code itself) is produced and artifacts produced in the process can be redesigned continuously. Producing reusable artifacts require a certain degree of design, documentation and continuous quality management, which require careful adjustments to the agile methods.

5. Limited support for developing safety-critical software
   For some software, formal specification, rigorous test coverage and other formal analysis and evaluation is required to assure robustness and safety of a system. This may conflict with the informal, lean and light agile methods. In these situations, formal techniques can be used in combination with agile techniques.

6. Limited support for developing large complex software
   In large complex software (thousands, millions lines of code), quality assurance, building by iteration and refactoring have to be managed much more formal than they are in agile methods. It is possible, but it requires a more formal and strict spanning management.

So, when one wants to work with an agile method regardless of the limitations, one has to adapt or modify the methodology to deal with the situation. While this paper briefly touches how to address these limitations, a more detailed read is found in [17].

The viewpoint of the limitations adds up with the discussion in section 2.2.1 about relevant contingency factors to determine suitability. Limitations show what a methodology is capable of by defining the boundary of what it cannot do.

As stated in section 1 and 2.2.1, an organization can only use an agile method effectively if the organization is capable enough. Looking in the contingency factors from Table 1, we see a number of contingency factors aligned with the theme “team empowerment”. They can be summarized in two topics which will be addressed next; “cultural values” and “team capabilities”.

### 3.3 Cultural values

Livary et al tried to model the relationship of cultural values of an organizational and the effectiveness of agile methods. They made two distinctions in their model: change vs. stability and internal focus vs. external focus. When putting these categories against each other, four groups of organizational cultures can be distinguished: the group culture (change and internal focus), the development culture (change and external factors, the rational culture (stability and external focus) and the hierarchical culture (stability and internal focus).

Each culture orientation, except the hierarchical one, favors agile methods. This however does not say that agile methods are better. Enterprise agility often is associated with adaptively and flexibility, therefore it would seem that the development culture sounds most suitable for an agile method. But from research it seems that it depends in the way the organization adopts the agile method; it seems that the more formalized the method becomes, the less the members of the development are tolerant of the agile method. Being more formal in an environment which encourages ‘being agile’ negatively influences the effectiveness of an organization.

Of course these four types are ideal types and an organization in reality will consist of combinations of characteristics of the four types, but when an organization mostly is considered having a
hierarchical culture, adapting an agile method is not favored [13].

3.4 Team capabilities
As said in section 2.2, organizational capabilities are an important factor to consider when determining if an agile method is suitable for a project. Boehm and Turner state that this can be determined by evaluating the experience and capabilities of the team members. While it may sound obvious that skilled personnel is a must for every SDM, it is argued that especially for agile methods a certain mix of team members is required. Where in traditional approaches a large team of lesser skilled employees can work on different pieces of code by following strict procedures, agile methods require at least a certain degree of flexibility which not everyone can handle adequately. Boehm and Turner state a minimum requirement of at least 30% full-time Cockburn Level 2 and 3 experts and no level 1B or -1 personnel in a team [4]. For reference, an overview table of these levels can be found in Appendix A.

3.5 Conclusion
From the definition of agile methods and the context of this study, two contingency factors are distilled to determine the suitability of a project for an agile method. The first one is the influence of limitations of agile methods. The second factor is the organizational capability to handle agility, which is determined by the culture values of the organization and the individual capabilities of the team.

4. THE FRAMEWORK
From the results of section 2.1 it seems that determining methodology suitability is very complex and is not something deterministic. From contingency theory it is defined as a scale where complete suitability only exists in an ideal situation. In all other situations, suitability of a methodology in the context of a project is determined by the capabilities of the methodology, the capabilities of the organization and the relation with the requirements of the context in which the project exists.

When looking at the agile methods available today (section 3), capabilities of agile methods can be expressed in limitations and organizational capabilities can be expressed in culture and personnel capabilities. The above construct is illustrated in Figure 4.

To use this construct for a decision, one has to do an assessment of the project environment and determine the following:

1. Which limitations as defined by [17] apply to the situation?
2. Is the organization’s culture mainly hierarchical? (See [13])
3. Does the organization currently use agile method components in software developing or had experience with them in the past? A component is considered agile if it complies with the definition given in section 3.1.

To get the suitability of agile methods, one has to consider the organizational culture and personnel capabilities. If the organizational culture and personnel capabilities do not meet the required level, there is a risk that the project is not suitable for an agile method. The required level is determined by the culture values of the organization and the individual capabilities of the team.

Figure 4. Suitability construct
viewpoint to other categories of SDM, like rapid prototyping for instance. It would be interesting to see if their also exist limitations for other SDM as there are for agile methods and if they can be related to suitability the same way as proposed in this framework. This would be a very useful contribution to the field of knowledge of SDM selection, since currently very little academic support is available.

6. CONCLUSION

The main question of this study is “How can be determined if a certain software project is suitable for an agile software development methodology?”. From contingency theory it is concluded that deciding whether a methodology is good in a certain situation is more complicated than answering a yes or no question. Suitability is a scale where complete suitability only exist in an ideal situation. In all other situations, suitability of a methodology in the context of a project is determined by the capabilities of the methodology and the relation with the requirements of the context in which the project exists. From the definition of agile methods, two contingency factors are distilled to determine the suitability of a project for an agile method: the influence of limitations of agile methods and the organizational capability to handle agility, which is determined by the culture values of the organization and the individual capabilities of the team. This is taken together in a framework which enables a practitioner to assess the project situation and which provides guidance for a good line of reasoning when deciding whether an agile method is suitable or not.

7. ACKNOWLEDGMENTS

My thanks to Diederik Rothengatter who provided me with plenty of additional readings and who gave very useful feedback during the process of this study. Also, my thanks to my fellow students, who reviewed my paper and research proposal, for their comments.

8. REFERENCES

## APPENDIX

### A. LEVEL OF SOFTWARE UNDERSTANDING AND USE

Table derived from [4]

<table>
<thead>
<tr>
<th>Level</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Able to revise a method (break its rules to fit an unprecedented new situation)</td>
</tr>
<tr>
<td>2</td>
<td>Able to tailor a method to fit a preceded new situation</td>
</tr>
<tr>
<td>1A</td>
<td>With training, able to perform discretionary method steps (e.g., sizing stories to fit increments, composing patterns, compound refactoring, complex COTS integration). With experience, can become Level 2.</td>
</tr>
<tr>
<td>1B</td>
<td>With training, able to perform procedural method steps (e.g., coding a simple method, simple refactoring, following coding standards and CM procedures, running tests). With experience, can master some Level 1A skills.</td>
</tr>
<tr>
<td>-1</td>
<td>May have technical skills, but unable or unwilling to collaborate or follow shared methods.</td>
</tr>
</tbody>
</table>