A DECISION FRAMEWORK FOR SELECTING A SUITABLE SOFTWARE DEVELOPMENT PROCESS

Itamar Sharon, Michel dos Santos Soares, Joseph Barjis, Jan van den Berg and Jos Vrancken
Delft University of Technology, Jaffalaan 5, 2628 BX Delft, The Netherlands

Keywords: Software development process, Decision making, Large scale organization, Decision support framework.

Abstract: For streamlining the activities of software development, a number of software development processes has been proposed in the past few decades. Despite the relative maturity in the field, large companies involved in developing software are still struggling with selecting suitable software processes. This article takes up the challenge of developing a framework that supports decision makers in choosing an appropriate software development process for each individual project. After introducing the problem, the software development processes included in this research are identified. For being able to align software development processes and software projects, a number of project characteristics is next determined. Based on these two analyses, a decision framework is proposed that, given the project characteristics, determines the most appropriate software development process. In a first attempt to validate the framework, it has been applied onto two case studies where the outcomes of the decision framework are compared to those found by means of a collection of experts' opinions. It was found that the framework and the experts yield similar outcomes.

1 INTRODUCTION

Despite all advances of the Software Engineering discipline in past years, software projects still present high failure rates. According to Georgiadou (2003), a review of hundreds of software development projects in organizations indicated that around 80% of the projects were considered unsuccessful (over budget, over schedule, unreliable) and that around a third of the projects were cancelled before completion. Similar results are presented in other studies as well (Johnson, 1994, Charette, 2005) in which detailed data about the high number of software projects that are concluded over budget and time are given. Choosing a software development process is often based on past experiences. For managers to choose such a process is becoming increasingly difficult for at least two reasons. First, software projects are continuously growing in complexity. Second, the lack of a framework for suitability matchmaking forces managers to rely on experience and intuition and even personal preferences. This increases the risk of failure since the combination of process characteristics, the environment, stakeholders, factors and evolvement of the organization or project needs actually to be 'optimal' (PMI, 2008, Vliet, 2008).

Choosing a software development process that is known to the organization and familiar for all employees is often seen to be a suitable decision. However, the problem is that large organizations develop all types of projects, from very small to very large. It is unlikely that the same process can be applied to all types of projects, considering the variety of hardware platforms and programming languages that are used. Although the experience-based approaches may yield certain results for small software projects, medium to large size projects require a more rigorous and profound framework for the selection of the software development process. The success of a software process for a given software project heavily depends on the characteristics of both the project and the development process chosen. It is further sometimes observed that there exists no 'ideal process' (Sommerville, 1996), and the question thus is more how to find a 'suitable process'

A possibility to cope with the fact that no process is perfect implies a need to tailor a process according to the organization and its environment. Unfortunately, using this approach, project managers need to monitor, analyze and adjust the process

Sharon I., dos Santos Soares M., Barjis J., van den Berg J. and Vrancken J. (2010).
A DECISION FRAMEWORK FOR SELECTING A SUITABLE SOFTWARE DEVELOPMENT PROCESS.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Information Systems Analysis and Specification*, pages 34-43
DOI: 10.5220/0002873900340043

tailoring strategies and the project environment continuously during the project (Xu and Ramesh, 2008). Another solution is to use different processes for different projects. Then, the most appropriate process to use depends on the characteristics of the particular project. However, for finding characteristics of projects that influence the suitability of software development processes, the current literature does not seem to suffice. In fact, formulation of projects characteristics and properties of software development processes as a framework for decision-making is widely overlooked by researchers. Most sources discuss the characteristics of software development processes (Pressman, 2010, Sommerville, 2007), but almost none make the connection between the suitability of a process with the characteristics of the particular project at hand. Moreover, in practice and in literature as well, very different opinions regarding the characteristics of software development processes have been evolved.

This article tackles this challenge by identifying typical characteristics of software projects that influence the suitability of processes and mapping both the processes and the characteristics into a framework. By providing project managers a framework for choosing the right process, valuable time and money can be saved as far as decision making is concerned.

The remainder of this paper is structured as follows. In Section 2, the software development processes investigated are shortly described. Section 3 focuses on the characteristics of software projects that influence the suitability of software development processes. The proposed decision making framework is discussed in Section 4. By using two case studies, the framework is tested and the results are presented in Section 5. Finally, in Section 6, conclusions are drawn and future research directions indicated.

2 SOFTWARE DEVELOPMENT PROCESSES

The diligent research and practice of the past few decades resulted in numerous software development processes. In this section, we briefly discuss the software development processes that are selected for consideration in this article. As a starting point, we mention the two software development processes as used in the organization in which this research was executed (Sharon, 2009). These are the Waterfall model (Royce, 1970) and the Rational Unified Process (RUP) (Kruchten, 2004). The Waterfall model has been a popular process for many organizations for its simple and linear approach to development (Pressman, 2010). The RUP is a detailed, iterative and incremental process, in which software architecture is emphasized as an important artefact. Processes similar to these two are the V-model (Vliet, 2008) and the Spiral model (Boehm, 1988). The V-model is based on the Waterfall model but has significant differences. The V-model includes a more thorough way of verification and validation. The Spiral model intensively uses risk assessment.

Agile methodologies are also very interesting to be included in this research. These methodologies have attracted much attention not only in the research community, but also in practice. Agile methodologies are based on an entire different approach compared to the Waterfall model (Dybå, 2008). They are considered to be light-weighted approaches, they accept and encourage changes during development, and they support incremental development of software. Some of the most significant processes within this type of methodology are eXtreme Programming (XP) (Beck, 2005), Scrum (Rising and Janoff, 2000) and Feature Driven Development (FDD) (Hunt, 2006). The last method included is the Dynamic Systems Development Method (DSDM) (Beynon-Davies et al., 1999).

3 MAJOR PROJECT CHARACTERISTICS

Based on the characteristics of the project at hand, the goal of the decision support framework is to provide the development process that is most suitable to it. In order to decide which process fits the project at hand, an overview of its characteristics is needed. For each characteristic it is possible to decide whether or not a certain process is suitable. Existing literature does not suffice in finding these characteristics. By interviewing experienced personnel from a large organization and combining this with information found in literature (Boehm, 1988, Beck, 2005, Dybå, 2008, Hunt, 2006, Pressman 2010, Sommerville, 2007, Vliet, 2008), a set of major characteristics has been identified (Sharon, 2009). The interviews were conducted with personnel throughout the organization. A survey was distributed among personnel to receive organization-wide information regarding their opinions

concerning these characteristics. Furthermore, suppliers of the organization were interviewed regarding their opinion about the application of various software development processes in practice.

From these interviews and literature study, it became clear that three factors are of significant importance for software projects: (a) *Project Budget*, and (b) *Project Duration*, and (c) *Requirements*. The selection and application of software development processes have direct impact on these factors. In turn, these factors also influence the suitability of a software development process. Although the influence works both ways, the influence the factors have on the software development process is crucial for the framework, especially the influence they have on the decision of which process to use.

From the three factors mentioned it is possible to derive characteristics that influence the suitability of software development processes. To find these characteristics, three methods were used: literature research, analysis of lessons learned documents, and interviews with professionals. Starting from the three significant factors mentioned, it was possible to derive a list of characteristics that are supposed to influence the suitability of software development processes (Sharon, 2009): please see Table 1.

3.1 Project Budget

Characteristics that are connected to the factor *Project Budget* are risk clearness, scope clearness, stakeholders' flexibility, and environmental stability. If the risk and scope are not clear, the project budget will not be. A significant amount of over budget and late projects are caused by uncertain boundaries (Georgiadou, 2003). Furthermore, these characteristics influence the suitability of processes, e.g., the Spiral model is considerably better at risk mitigation in comparison to the Waterfall model (Boehm, 1988). The stability of the development environment is highly important for the project budget. If the infrastructure and techniques used are unreliable, the budget becomes unreliable as well. In addition, if the stakeholder is inflexible, occurring changes are difficult to apply and budget can be overrun.

3.2 Project Duration

The characteristics that influence the factor Project Budget are often related to the factor Project Duration. In addition, another characteristic was found that only influences Project Duration. This is the familiarity the project group has with the application to be developed. Certain projects are adaptations of already existing software. However, some projects concern entire new innovative software applications. If this is the case, more time is needed for the project group to get familiar with the domain.

3.3 Requirements

The final factor mentioned above is that of *Requirements.* Five characteristics were found that have influence on or are influenced by this factor. The first two characteristics are requirements maturity and expected changes in requirements. For many software projects, the stability of requirements is of the utmost importance. Changes at the end or during the project could cause enormous problems. The third characteristic is client's commitment. For elicitation and collection of explicit and unambiguous requirements, clients need to be committed to the project (Dybå, 2008). Furthermore, for some software development processes, business partners need to be on site for the entire duration of the project (Dybå, 2008, Hunt, 2006). The fourth characteristic is the method of contracting that is applied. Agile processes are more suitable for projects in which time and material (where the final price depends on the time and material spend) is applied (Paetsch et al., 2003), and for the Waterfall model fixed pricing (in which the amount of time and material are dependent on the agreed price) is more suitable. This is because the two methods differ in requirements management. Fixed pricing causes requirements to be frozen, while time and material do not. The final characteristic is the flexibility of stakeholders since requirements management is dependent on their wishes.

3.4 Additional Characteristics

During this research a number of the characteristics found did not relate to any of the three factors mentioned before. The first of these characteristics is team size. Software projects can differ greatly in the amount of people working on the project. This does include the stakeholders, the client, and personnel from the vendors. For some processes, such as agile processes, small teams are a necessity. Besides, the Waterfall model, for example, can cope with large teams because of its intense use of documentation.

The second characteristic, which relates to the team size, is how good the team relationship is. This is important in general, but is crucial for agile methodologies, in which team communication is a common, encouraged practice.

Table 1: Final list of characteristics and their average weights.

Characteristic	*Description*	*Weight*
Requirements Maturity	*The maturity of the wishes of the client. Are they likely to change? Are they feasible?*	4,4
Development Stability	*The stability of the development environment. Is the technique familiar? Will it change during the project?*	4,3
Project Size	*The Size of the project. Can be expressed in function points, budget, and project duration.*	4,1
Risk Clearness	*Are all the possible problems and dangers clear? Will there be no surprises?*	3,8
Outsourcing	*Is there any outsourcing? Are there any vendors used for writing code?*	3,5
Scope Clearness	*The clearness of the extent of the project. Is the project clearly demarcated? Or is it likely to change?*	3,3
Client's Commitment	*The availability of the client. Is he/she ready and willing to work for the project?*	3
Team Relationship	*The relationship among all team members. Is communication going smoothly?*	3
Team Size	*The size of the entire team. This entails every architect, programmer, account manager, project manager etc.*	2,9
Method of Contracting	*Fixed Pricing: Cost agreement is made based on the requirements. Time/Material: Open contract.*	2,9
Stakeholders Flexibility	*Is the stakeholder open-minded? Does he/she only want to do things the familiar way, or is he/she open for innovations?*	2,8

The third and final characteristic is departmental influence. In many large organizations different departments are included in the development. When these different departments use different software development processes, integration is a challenge. For certain processes, this integration factor is of significant importance.

3.5 Chosen Characteristics

In order to verify and review the characteristics found in the previous section, a survey was distributed among personnel from a large organization (the resulting, preliminary list of characteristics is given in the Appendix). In the survey, the appropriateness and weights of each characteristic were asked. In a next step, the preliminary list obtained was critically analysed. In Table 1, the *final* list of *characteristics* and their *averaged weights* (Sharon, 2009) are presented. The changes that were made concerning the first list of characteristics are the deletion of application familiarity and departmental influence. Application familiarity is in fact included in the characteristic *Development Stability*. The reason that departmental influence is deleted is because when this occurs, major benefits of a certain process still exist for a single department, even if integration benefits are lost. The characteristic *Outsourcing* is included in this final list of characteristics because it currently is applied for the majority of software projects developed at the organization. The weights are given in a range from the scale of 1 to 5, 1 being the lowest and 5 being the highest weight.

4 DECISION FRAMEWORK

Based on the characteristics of software development processes and of software projects, a decision support framework has been developed. The framework tells what software development process is most suitable regarding the particular project currently at hand.

4.1 Mapping Solution

In order to solve the mapping problem of the two different sets of information, the following solution was developed. The framework consists of individual characteristics. The suitability scores of each process on each individual characteristic are summed up: this forms the total score. The suitability score of each process on each characteristic is found by doing the following. Each characteristic is divided into five scales. Figure 1 depicts an example. The project size is in this case divided into five scales, 1 to 5. Scale 1 represents a project of a very small size. Scale 5 here represents a very large project. For each scale, it is possible to analyze how suitable each software development process is. Three grey grades are used to state the

suitability of a process. Light grey represents high suitability, medium grey represents medium suitability, and a high grey grade means minimal suitability. If a process is not mentioned on a particular scale, it means that it is not at all suitable in that particular case. On the first scale, Scrum and XP are found to be suitable. DSDM and FDD are presented in medium grey grade. When dealing with a small project, agile processes are appropriate. Scrum and XP are suitable in coping with small teams. The Waterfall model for example, consists of a high amount of documentation and minimizes face-to-face time, which is therefore less suitable for small projects. On the fifth scale of the characteristic project size, the Waterfall model is presented in medium grey. In this case, documentation is highly important.

Scrum	FDD	RUP	RUP	Waterfall
XP	RUP	FDD	Waterfall	V-Model
DSDM	DSDM	…	V-Model	RUP
FDD	Scrum	…	Spiral	Spiral
1	2	3	4	5

Figure 1: Mapping development processes on a specific characteristic of a given software project.

The mapping shown in Figure 1 can be executed for each characteristic presented in Table 1. Therefore, all these characteristics are implemented into a spreadsheet-based application.

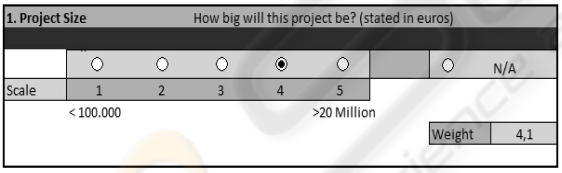

Figure 2: Example entry form.

The first element of the decision framework is the entry form. This entry form consists of the eleven characteristics presented in Table 1. The users will define their project by indicating the scale of each characteristic. The entry form is presented for a single characteristic in Figure 2. Almost all characteristics consist of five scales, plus a 'not available' option. The characteristics *method of contracting* and *outsourcing* consist of two scales. For the method of contracting these are "time & material" and "fixed pricing". For outsourcing the options are "yes" and "no". The "not available" option can be selected when no information regarding this characteristic is available or when it is not a significant factor for this particular project. If this is selected, the characteristic is removed from the calculation. Each characteristic also consists of a weight determined by employees. However, it is possible to adjust these weights, namely if clients and project managers agree that in a particular project.

4.2 Calculation

In this subsection the calculation of the suitable software development process(es) is explained. The calculation depends on the scales given in the entry form. In Figure 3, the calculation of the characteristic *project size* is depicted. In the upper table in the figure the calculation scores are presented. When a software development process scores high suitability on a particular scale, then this process receives the score 1. If the process is minimally suitable, the score 0.5 is given. In the example, *project size* is given the scale 4 (which can be found in the blue bar on the far right). This means that it concerns a large project. When a project is defined as large, more structured and documentation oriented processes are suitable (Pressman, 2010, Sommerville, 2007). On the far right of the characteristic table the suitable processes based on the scale given are presented. In this case, when the *project size* is defined as large, RUP is most suitable. The Waterfall model is also considered suitable, but in a lower degree. The Spiral model and the V-model are also depicted here meaning that they are minimally suitable. The other processes are, for this particular characteristic and its scale, not suitable.

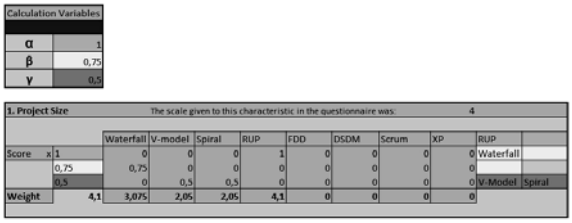

Figure 3: Example calculation.

Figure 3 shows that these processes receive the score corresponding to their suitability grade. The score is then multiplied by the weight of the characteristic, in this case 4.1. This results in higher scores for more important characteristics. For each characteristic this calculation is executed and these are finally summed up. The software development process that receives

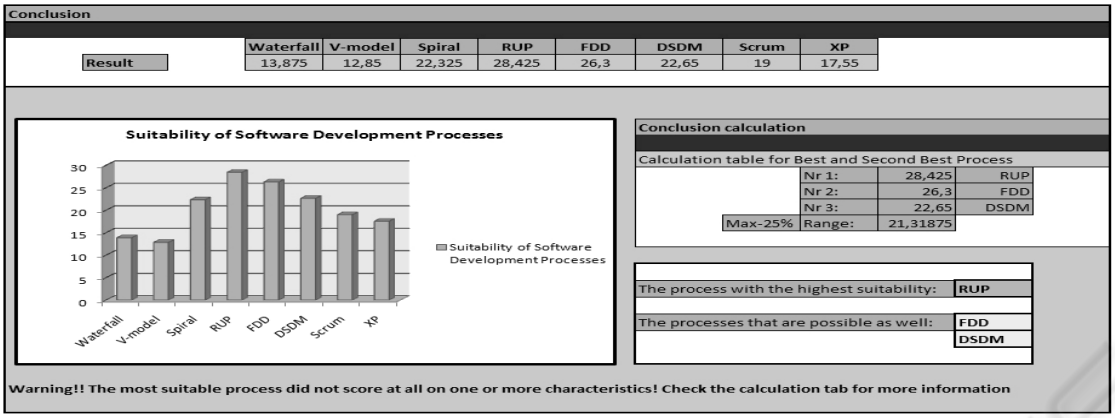

Figure 5: The conclusion frame.

the highest summed up score is most suitable. This will be explained later.

Most of the characteristics are independent. However, two characteristics are influenced by another characteristic. These dependencies are that of *team size* on *team relationship* and *requirements maturity* on *scope clearness*. For these dependencies a correction is needed. In Figure 4 this is presented for *team relationship*.

When the team size becomes big, the relationship within a team gets compromised. Geographical boundaries, lack of communication and opposing values are introduced. This is therefore corrected in the framework. On the right side of the figure the correction table is presented. This shows that when the *team size* is very big, the *team relationship* gets corrected by 60%. In the example in Figure 4, the original score of the relationship was 3. This is then corrected by 60% and results in a score of 2. This is similar for the dependency between *requirements maturity* and *scope clearness.*

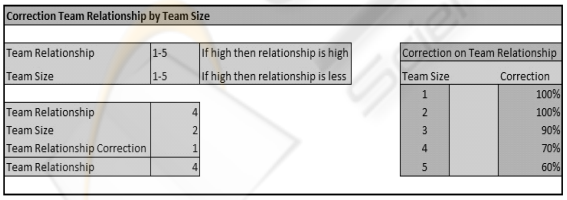

Figure 4: Correction on team relationship.

4.3 Result

The final element of the decision framework described in this article is the result. The main purpose of this framework is to indicate the most suitable software development process for a particular project. The conclusion, depicted in Figure 5, presents this information. In the conclusion five elements are presented. The first element is the table with the summed up score of the software development processes on all the characteristics. The second element is the graph that represents the same result as given in the table. The conclusion calculation is the third element. This calculates the result given in the fourth element, the final recommendation. Finally, the fifth element states a warning when the most suitable process presented in element four does not at all score on one or more characteristics. The conclusion table and the graph basically show the scoring of each software development process. In this particular example, the RUP scores highest. However, this does not necessarily mean that this process should be used.

Other processes, such as FDD and DSDM score high as well. The decision framework does recommend using the RUP for this particular project. The processes FDD and DSDM could be used. It can be concluded that the Waterfall model and the V-model should definitely not be used.

The final element of the conclusion of the decision framework is the warning statement. It is possible that the framework shows that RUP is the most suitable process while this process did not at all score (not even minimally) on one or more characteristics. If this is the case it is recommended to analyze the consequences for its suitability. For example, if the RUP scores on all characteristics except on *requirements maturity,* the user should analyze why this is the case.

If the requirements are very immature, an agile process is needed and not RUP. The user can conclude that the activities of requirements engineering (e.g., elicitation, prioritization) need to be more mature. If the requirements are more mature, the suitability of the RUP increases.

The framework also includes information tabs. The user is given a short basic guide of how to use the framework and which steps need to be taken. Furthermore, the framework also has a software development process page to present the user information regarding each process used in this framework. When a user is not familiar with a certain process this page provides a good overview.

5 CASE STUDIES

5.1 Case Study 1: A RUP Project

The first case study is considered to be a project very suitable for RUP. An expert regarding RUP tested this project on the framework to find whether or not the framework gives RUP as most suitable solution and to find which other software development processes might be possible. The framework is therefore tested on its accuracy. The project is defined as follows (Table 2).

Table 2: Project for case study 1.

Project Characteristic	Scale
Project Size	3
Team Size	3
Requirements Maturity	2
Team Relationship	4
Client's Commitment	3
Scope Clearness	2
Risk Clearness	2
Development Stability	2
Stakeholders Flexibility	2
Method of Contracting	Time/Material
Outsourcing	Yes

When this is applied to the framework the following conclusion is given (Figure 6). Only the graph of the conclusion is presented regarding the fact that this is the most clarifying and complete element.

The conclusion of the framework is similar to that of the experts' expectations. The RUP can be considered to be the most suitable process for this particular project. However, FDD is also a very viable process for this project. The scales selected are very average (no extremes). The RUP and FDD are both good software development processes to use when a project finds itself between very heavy/traditional and very small/uncertain.

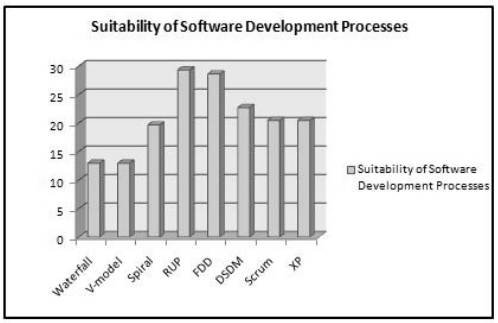

Figure 6: Conclusion of case study 1.

5.2 Case Study 2: A Large Uncertain Project

The second case study consists of a project that was considered to be very large and had many uncertainties. During the execution of this project many difficulties and problems arose. This case study is interesting for the decision framework. Many large organizations struggle with oversized projects in which the risks and scope are difficult to define. The project characteristics are defined in Table 3.

Table 3: Project for case study 2.

Project Characteristic	Scale
Project Size	5
Team Size	5
Requirements Maturity	1
Team Relationship	2
Client's Commitment	3
Scope Clearness	1
Risk Clearness	2
Development Stability	1
Stakeholders Flexibility	4
Method of Contracting	Time/Material
Outsourcing	Yes

The graph of the suitability of the software development processes is depicted in Figure 7. The result of the framework is interesting and complicated. As the graph shows, almost all software development processes are roughly equal in suitability for this particular project. This can be explained by the extreme values of scales given for this project. A project of such great size and many uncertainties will most likely fail.

According to the framework, the agile processes Scrum and XP both score highest on suitability. However, these processes are considered less suitable on very large sized projects. This conclusion should therefore be interpreted as follows: agile

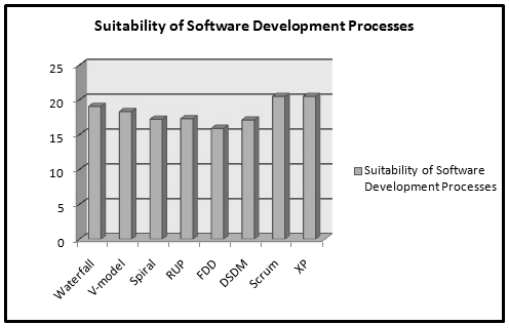

Figure 7: Conclusion of case study 2.

processes are most suitable for this project. However, taking the size of the project into account, this project in its current scale is not a feasible project to be executed. This conclusion indicates that the project in its current state is not specified enough, and further activities such as risk evaluation and requirements analysis are needed. This test shows that the framework does not only help in selecting a suitable process, it also helps in finding and discussing projects in which the specification is still an issue.

6 CONCLUSIONS

Through an analysis of literature, a lack of success was found in executing large-scale software development projects. Especially the over-budget, over-schedule, and under-functioning are most recurrent phenomena in software projects. Often, part of the cause for these issues is the lack of match between the projects at hand and the software development process in use. Therefore, organizations are struggling in selecting an appropriate process for developing software projects. In order to find a solution for this lack of success, a framework was developed to present the user with the most suitable processes. This is achieved by taking the characteristics of the software project at hand into account. An analysis of software development processes and software projects resulted in a number of characteristics that influence the suitability of each software development process. Furthermore, by performing interviews and distributing a survey, weights have been attached to these characteristics.

Based on these results, a framework was created in which the user is able to indicate how the current software project is characterized. Based on this input, a suitability score on each characteristic is given to each software development process. The summed up score of these individual suitability ratings presents the user with the most suitable process (es). The case studies have been performed at a large-scale organization involved in software development, and have yielded correct and accurate results, that can be considered as a first test and validation of the proposed framework.

The resulting framework can be considered as a suitable tool for analyzing software projects and selecting an appropriate software development process. Moreover, is also helps in finding and discussing projects that are not specified thoroughly enough. This framework triggers the user to actually consider the project at hand.

Although the framework is discussed and tested with users and experts, further validation is necessary. It is recommended to compare projects conducted while using the framework and projects conducted without using the framework. In addition, for this article, only one large-scale organization involved in developing software was included in the validation process. To analyze and test the general applicability of this decision framework, multiple organizations in different market segments should be included.

REFERENCES

Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. *IEEE Computer*, 21 (5), pp. 61-72.

Beck, K., Andres, C. (2005). *Extreme Programming Explained: Embrace Change*. (2nd Edition). Addison-Wesley, Boston.

Beynon-Davies, P., Carne, C., Mackay, Tudhope, D. (1999). Rapid Application Development (RAD): An Empirical Review. *European Journal of Information Systems* 8 (3), 212-223.

Charette, R. N., (2005). Why Software Fails. *IEEE Spectrum* 42 (9), 42-49.

Dybå, T., Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50 (9-10), 833-859.

Erickson, J., Lyytinen, K., Siau, K. (2005). Agile Modeling, Agile software development, and extreme programming: the state of research. *Journal of Database Management*, 16 (3), 226-237.

Georgiadou, E. (2003). Software Process and Product Improvement: A Historical Perspective. *Cybernetics and Systems Analyses*, 39 (1), 125-142.

Hunt, J. (2006). *Agile Software Construction.* Springer-Verlag, London.

Johnson, J. H. (1994). The CHAOS Report. The Standish Group International, Inc.

Kruchten, P. (2004). *The Rational Unified Process: An Introduction*. (3rd Edition). Addison-Wesley Longman, Inc.

Paetsch, F., Eberlein, A., Maurer, F. (2003). Requirements Engineering and Agile Software Development. *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE'03).

Pressman, R. S. (2010). *Software engineering; a practitioner's approach*. (7th edition). McGraw-Hill, New York.

PMI - Project Management Institute. (2008). *A Guide to the Project Management Body of Knowledge*. (Fourth Edition).

Rising, L., Janoff, N. S. (2000). The Scrum software development process for small teams. *IEEE Software*, 17 (4), 26-32.

Royce, W. (1970). Managing the Development of Large Software Systems. *IEEE WESCON*. 1-9.

Sharon, I. (2009), A Decision Framework for Selecting a Suitable Software Development Process, *Master's Thesis*. Delft University of technoogy, August 2009.

Sommerville, I. (1996). Software Process Models. *ACM Computing Surveys*, 28 (1), 269-271.

Sommerville, I. (2007). *Software Engineering. (*8th Edition) Pearson Education Limited, Harlow, England.

Vliet, van H. (2008). *Software Engineering; Principles and Practice*. John Wiley & Sons Ltd, USA.

Xu, P.; Ramesh, B. (2008). Using Process Tailoring to Manage Software Development Changes. *IT Pro*, July/August, 39-45.

APPENDIX

Table 4: Preliminary list of project characteristics and their weights.

Project characteristics	Weights as given by 12 respondents												Average weight
Project budget	2	5	1	2	1	5	5	1	2	1	5	5	2.5
Project duration	4	5	4	1	2	3	4	4	3	2	3	5	3.34
Team size	4	4	3	3	2	1	3	3	4	3	3	2	2.92
Requirements maturity	5	5	4	3	4	4	5	5	5	4	4	4	4.42
Expected change in requirements	4	5	5	4	4	4	3	4	5	4	4	4	4.25
Influence of other departments	3	3	0	1	2	3	2	1	3	1	2	5	2.17
Team relationship	4	4	0	3	2	5	5	2	3	1	2	5	3
Experience with application	2	3	0	2	4	3	2	4	3	1	1	4	2.42
Stakeholder flexibility	3	3	2	4	2	4	2	3	4	2	4	3	3
Client commitment	2	5	0	4	0	5	4	4	4	3	4	5	3.34
Scope clearness	3	5	5	1	3	4	5	5	3	1	5	5	3.75
Risk clearness	4	5	5	5	5	5	3	4	3	3	5	5	4.34
Environment stability	4	3	4	1	0	2	5	3	2	1	3	5	2.75
Method of contracting	2	4	0	4	4	3	2	2	4	3	1	5	2.84

In order to verify and review the characteristics identified in the first sections of chapter 3, a survey was distributed among personnel from a large organization. In this survey, the appropriateness and weights of each characteristic were asked. Table 4 shows an overview of the selected characteristics and the weights, given by twelve respondents. Each respondent has expressed its opinion by giving each characteristic a number from 1 to 5. When the respondent finds the importance of a characteristic to be very low, the number 1 is given. When the characteristic is found very important, the number 5 is given. From these weights, an average value has been calculated that is shown in the last column.

The survey gave the respondents the possibility to answer whether or not (s) he found that certain characteristics should be deleted. It turned out that four characteristics could be deleted (or changed). The characteristic “Influence of other departments” is seen as not important, and was deleted from the framework. Furthermore, the characteristic “Experience with the application” was also deleted. The characteristics “Project Budget” and “Time Schedule” were combined into the characteristic “Project Size”. This is based on the opinions given by the respondents. Another characteristic that was be added to the framework is “Outsourcing yes or no”. This characteristic was mentioned by two respondents.

In the survey, respondents were also asked to give answers to open questions. Two questions provided interesting information. The first interesting question is “*Do you think that the application of software development processes within this company is going well*?”. Nine out of the eleven respondents answered this to be not the case. Therefore, this subject is definitely an important issue and confirms the statements made in the introduction of this article. The other interesting question was “*Do you want to have multiple software development processes to be applicable within the organization for different projects, or do you rather have one process which is tailored to fit different projects*?”. This question concerns the main statement made in this research (which is that multiple processes should be available for different projects). Five out of ten respondents agreed with this statement.