An Appraisal of Existing Evaluation Frameworks for Agile Methodologies

Masoumeh Taromirad, Raman Ramsin
Department of Computer Engineering
Sharif University of Technology
taromi@ce.sharif.edu, ramsin@sharif.edu

Abstract

The emergence of agile software development methodologies, and the sheer number of the variants introduced, has raised the need for evaluation and comparison efforts, mostly in order to facilitate the selection or engineering of an agile methodology aimed at a specific software development situation. But existing evaluation frameworks and comparison tools do not satisfy all the needs of project managers and method engineers. The principal objectives of this paper are to introduce a fundamental basis for evaluation frameworks that target existing challenges, and also to evaluate existing evaluation frameworks in order to identify their shortcomings. To this aim, we have proposed an evaluation criterion set, or meta-criteria, according to which existing evaluation frameworks have been assessed. These meta-criteria define the essential features that an appropriate evaluation framework is expected to possess. The evaluation results show that existing evaluation frameworks do not satisfy typical requirements, and fail to address existing challenges. Therefore, the need still remains for an appropriate and useful evaluation framework for appraising agile methodologies.

1. Introduction

The advent of agile software development methodologies has been a remarkable contribution to contemporary software engineering. Several agile software development methodologies have been developed in recent years, the users of which face challenges unique to the agile approach. Project managers need to select appropriate agile development methodologies for their projects, while method engineers need efficient means for engineering tailored-to-fit agile methods. These cannot be made possible unless there are suitable tools to analyze and evaluate existing agile methodologies to identify their weaknesses, capabilities, similarities, and differences. Evaluation frameworks and methods are integral parts of such tools.

In order to be useful to project managers, evaluation means should take into account the various parameters of software development projects, and thereby accentuate the similarities, differences, features and applications of available software development methodologies. Method engineers, on the other hand, require evaluation means that facilitate the construction of custom methodologies through extending existing methodologies, instantiating methodology metamodels, or assembling method-fragments selected from a method repository.

Furthermore, availability of appropriate evaluation frameworks and methods is essential if conformance to process standards and templates needs to be enforced. As a consequence, evaluation frameworks are of extensive use in governing Software Process Improvement (SPI) and process evolution efforts.

As a result of their vast application scope, evaluation frameworks are expected to satisfy certain requirements, disregard of which can seriously reduce the practicability and practicality of frameworks. Although several evaluation frameworks or methods have been introduced for comparing, analyzing, or evaluating agile methodologies, they are all lacking in addressing method engineering and project management requirements. Thus, challenges remain unaddressed, and project managers and method engineers are left without adequate support.

The aim of this paper is to evaluate existing evaluation frameworks in order to identify their shortcomings, and then provide guidance for producing an improved evaluation framework for agile methodologies. As the first step, all the high level requirements and characteristics that an appropriate evaluation framework should address have been

978-0-7695-3141-0/08 $25.00 © 2008 IEEE
DOI 10.1109/ECBS.2008.32

collected and defined. These requirements and characteristics are expressed as evaluation criteria, to be used as *meta-criteria* (criteria for evaluating criteria) for evaluating existing evaluation frameworks. As the next step, existing evaluation frameworks/methods have been analyzed based on the meta-criteria.

The rest of this paper is structured as follows: Section 2 presents a short overview of existing evaluation frameworks; in Section 3, existing meta-criteria for appraising evaluation frameworks are described, and a new set of meta-criteria is proposed; Section 4 contains the results of evaluating existing evaluation-frameworks based on the proposed meta-criteria, and Section 5 offers several observations based on the evaluation results; Section 6 summarizes the key achievements of this study and suggests ways for furthering this research.

2. Existing evaluation frameworks

In this section, existing evaluation frameworks are briefly introduced. Abrahamsson et al. have introduced a structure in which the process, roles and responsibilities, practices, status of adoption, experiences, scope of use, and current research regarding each method are identified [1]. They have used this structure as their analytical foundation to analyze the differences and similarities between different agile software development methods.

This analysis has been performed by comparing agile methodologies in two parts: 1) comparison of general features, by referring to current research and scope of use, and focusing on "key points", "special features", and "identified shortcomings"; 2) comparison of adoption taking into account the coverage of software development life-cycle, support of project management, clear process definition, and applicable and practical practices, activities, and work products, as described in each method.

In another paper, Abrahamsson et al. have proposed an analytical framework for analysis of existing agile methods [2]. Based on their previous work, software development notions, such as lifecycle coverage (including the process), project management, abstract principles vs. concrete guidance, universally predefined vs. situation appropriate, and empirical evidence have been considered as analytical criteria, called "analytical lenses".

Software development life-cycle observes which phases of the software development process the agile methods cover. Life-cycle coverage implies the need of clear and identified process definition in a method. Project management has been considered as a support function that facilitates efficient software development. Abstract principles vs. concrete guidance focuses on determining whether agile software methods provide any concrete guidance or rely on abstract rules of thumb. Universally predefined vs. situation appropriate is to show if a method claims to fit all software development situations or is adjustable depending on the situation. Empirical support is analyzed in order to see what kind of empirical evidence agile methods are based upon and focuses on studying different agile methods in real life situations.

Regarding the critical role of Software Configuration Management (SCM) in software development, especially in agile methods, Koskela has introduced a number of specialized "analytical lenses" to analyze the state of software configuration management in agile methodologies [3]. The analytical lenses defined are SCM approach, SCM planning, Configuration identification, Change management, and SCM tools.

Williams et al. have provided a benchmark for assessing the XP practices which an organization has selected to adopt and/or modify, and the outcome thereof [4]. This framework is called XP-EF and is composed of three parts: XP Context Factors (XP-cf) to record essential context information about a project, XP Adherence Metrics (XP-am) to concretely and comparatively express the practices a team utilizes, and XP Outcome Measures (XP-om) to assess and report a team's outcome through using a full or partial set of XP practices.

To record context information about a project, Williams et al. have used [5]'s key factors expressed in the six categories: software classification, sociological, project-specific, ergonomic, technological, and international, as well as a new one: developmental factors (that makes use of Boehm and Turner's five critical factors [6], [7]). Subjective and objective measures as well as qualitative analysis and traditional software development metrics are used in the second and third parts.

Germain et al., in a practical comparison between an engineering-based process, named Unified Process for Education (UPEDU) derived from the Rational Unified Process, and an agile process built around the principles of the Extreme Programming (XP) methodology, have compared and analyzed work and time spent in each of their defined activities (called "cognitive activities") [8].

One of the latest works in this context is 4-DAT, a framework-based assessment tool for analysis and comparison of agile methods, proposed in [9]. The distinguishing feature of 4-DAT is that it specifically provides a mechanism for measuring the degree of agility of any method quantitatively at a specific level in a process using specific practices.

4-DAT has four dimensions that provide evaluation criteria for the detailed assessment of agile software development methods from different perspectives: 1) Method scope characterization is a set of key scope items including: Project Size, Team Size, Development Style, Code Style, Technology Environment, Physical Environment, Business Culture, and Abstraction Mechanism; 2) Agility characterization is a set of agility features including: Flexibility, Speed, Leanness, Learning, and Responsiveness; 3) Agile Values characterization includes six values, four of which have been provided by the Agile Manifesto [10], the fifth by Koch and the sixth had been proposed by Qumer and Henderson-Sellers in a previous work (Table 1); and 4) Software Process characterization, incorporating two main components of software process, namely: product engineering process and process management. Process management in turn casts into three categories: development process, project management process and support process.

Table 2 gives an overview of the evaluation frameworks analyzed herein.

Table 1. Agile values used in [9].

Agile Values
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
5. Keeping the process agile
6. Keeping the process cost-effective

3. Meta-criteria

The first step in any evaluation is to define an appropriate evaluation criterion set which forms the evaluation. In this section we define and introduce our evaluation criteria providing the means for a structured evaluation leading to reliable results. As these criteria are used to evaluate other criteria and describe features and characteristics which an appropriate criterion set should satisfy, we call them *meta-criteria*.

In order to produce a suitable meta-criterion set, the characteristics considered desirable in the meta-criteria should be clearly defined. Meta-criteria are evaluation criteria too, and as such are expected to possess the general characteristics that are typically regarded as desirable in evaluation criteria (examples listed in Section 3.1). It should be noted, however, that some of these characteristics are not applicable to meta-criteria – since they are aimed at a different abstraction level – and some need to be redefined. In general, it can be declared that meta-criteria should be precise, simple, independent, and comprehensive. In addition to general characteristics, the meta-criteria should also satisfy certain domain-specific requirements; in the context of our research, this means that the meta-criteria should address the specific features of the agile-software-development domain.

According to our evaluation objective (analysis of agile-methodologies evaluation methods), the target meta-criteria should address three aspects, and are therefore cast into three types:

1. General evaluation meta-criteria: features that any evaluation criterion set should have.
2. Software methodology evaluation meta-criteria: features that an evaluation criterion set needs for evaluating a software development methodology.
3. Agile methodology evaluation meta-criteria: characteristics needed for evaluating agile-specific features in software development methodologies.

3.1. General evaluation meta-criteria

General meta-criteria describe requirements and properties that any evaluation-criterion set, independent of its domain, should possess.

The general meta-criteria used in this research were derived through studying evaluation-framework definition processes and identifying the measures of quality that they use for improving evaluation criteria. Frank [11], for instance, defines the main aspects of the target domain as evaluation criteria, and then removes *inconsistencies* and *overlappings* to produce an improved set of evaluation criteria. We have used the quality measures thus identified to define the following meta-criteria:

1. Precision: evaluation criteria should lead to the identification of similarities and differences in detail. In other words, results should be explicit, detailed and unambiguous [12].
2. Simplicity: easy to understand and measure [12].
3. Usability and meaningfulness: The criteria should be usable and semantically meaningful [12].
4. Consistency: criteria in a set should not be inconsistent or in conflict with each other [11].
5. Minimum overlapping: criteria should be independent and have minimum overlapping with each others [11].

Table 2. An overview of the evaluation frameworks analyzed in this research.

Evaluation Framework	Main Objectives	Special Characteristics	Main Evaluation Criteria/Focuses
Abrahamsson et al. (2002)	Identifying differences and similarities of agile methodologies	– Defines a common structure for describing methodologies, including: the process, roles and responsibilities, practices, status of adoption, experiences, scope of use, and current research.	– Coverage of the generic software development life-cycle – Support for project management – Clear process definition – Applicable and practical practices, activities, and work products
Abrahamsson et al. (2003)	Identifying new directions in the evolution of agile methodologies through comparative analysis		– Lifecycle coverage, project management support – Abstract principles vs. concrete guidance – Universally-predefined vs. situation-appropriate – Empirical evidence
Koskela (2003)	Analyzing the state of Software Configuration Management in agile methodologies		– SCM approach – SCM planning – Configuration identification – Change management – SCM tools
Williams et al. (2004)	Providing a benchmark for assessing the XP practices used and the outcome thereof	– Proposes a three-part framework (XP-EF) targeted at XP. – Defines and uses subjective and objective measures as well as qualitative analysis and traditional software development metrics.	– Software classification issues – Sociological issues – Project-specific issues – Ergonomic issues – Technological issues – International issues – Developmental issues
Germain et al. (2005)	Comparison between an engineering-based process and an agile process	– Defines a set of activities called "cognitive activities" as units of evaluation.	– Time and effort spent in each activity
Qumer et al. (2006)	Introducing a framework-based assessment tool for analysis and comparison of agile methods (4-DAT)	– Provides a mechanism for quantitative measurement of the degree of agility in any given method	– Method scope – Agility – Agile Values – Software Process

In addition to the above, there are underlying assumptions and implicit quality measures which may not be immediately recognizable, but which are commonly present in rigorous evaluation frameworks. We have identified these and added them to the list of general meta-criteria:

6. Generality: evaluation criteria should be applicable to all instances, and analyzing a methodology based on them should lead to valuable and valid results.
7. Comprehensiveness: evaluation criterion set should consider all the important aspects or features in the relevant context, considering the evaluation goals.
8. Balance and Homogeneity: adequate attention should be given to all the major features of the domain.

Naturally, these meta-criteria are rather abstract and only explain high level needs. Therefore they should be adapted to the target domain in order to be useful and meaningful.

3.2. Software development methodology evaluation meta-criteria

The oldest meta-criteria in this field are those proposed by Karam and Casselman [13]:

1. Generality: a criterion set should be *general* enough to be used for evaluating all software development methodologies.
2. Precision: a criterion set should be *precise* enough to help discern and highlight the similarities and differences among software development methodologies.
3. Comprehensiveness: a criterion set should be *comprehensive* enough to cover all significant features of software development methodologies.
4. Balance: adequate attention should be given to all three major types of features in a methodology: *technical*, *managerial* and *usage*.

These meta-criteria are not detailed enough, and evaluation based on them is difficult and error-prone. Thus, more detailed meta-criteria are required; as an example, Ramsin has introduced the following meta-criterion set for object-oriented software development methodologies based on Karam and Casselman's meta-criteria [14]:

1. Generality
2. Balance
3. Precision
4. Comprehensiveness
5. Attention to Object-Oriented aspects

3.3. New meta-criteria

All of the above are useful but not complete enough, and therefore do not satisfy our needs. The most important shortcoming of these meta-criteria is that they do not consider the main aspects of agility in software development methodologies. Moreover, they do not address the notions introduced in modern software development methodology such as handling frequent changes in requirements, development speed, and flexibility. So we introduce a new meta-criterion set for agile methodologies evaluation methods, addressing these shortcomings.

The general evaluation meta-criteria have been chosen as the top level criteria, but the lower-level meta-criteria have been changed. The proposed meta-criteria are:

- **Precision**
  1. Detail: criteria should be defined at an adequate level of detail.
  2. Unambiguity: the main intention of the criteria should be clear; this in turn leads to clearer results.
  3. Quantified results: Evaluation results should be expressed quantitatively or selected from predefined ranges of values/categories. Typically, narrative results are of little value in evaluations and analyses.
- **Simplicity**
  1. Simple definition
  2. Ease of measurement
- **Quantitative metrics**; an evaluation criterion set should contain as many quantitative metrics as possible, mainly because they produce quantified results. It should be noted, however, that quantitative metrics are but one way of producing quantified results
- **Usability and meaningfulness:** criteria should be usable and semantically meaningful in their usage context.
- **Consistency:** criteria in a set should not be inconsistent or in conflict with each other.
- **Minimum overlapping:** criteria should be independent from each other.
- **Generality**; based on Ramsin's definition [14], generality encompasses the following:
  1. Independence from methodology: evaluation criteria should not be dependent on any specific methodology. All evaluation criteria should be applicable to all the development methodologies in the target domain.
  2. Independence from software development model: criteria should not be dependent on any specific software development model (e.g. waterfall, spiral, etc.).
- **Balance**; evaluation criteria should consider all these aspects, even if just partially:
  1. Methodology definition
  2. Methodology classification
  3. Methodology usage
- **Comprehensiveness**; all the main aspects of agile methodologies should be covered by an evaluation method. A software development methodology can be considered from three perspectives: definition, classification, and usage. Definition refers to process and modeling language used in a methodology and concerns the methodology itself, not its usage or other relevant issues. Classification emphasizes those features that a methodology is expected to have based on its classification. Usage addresses the practical aspects of a methodology.

  Consequently, the main aspects of a development methodology are categorized in three types, and based on other analyses [11], [15], [16], [17], [18], are defined as:

  1. Main aspects of methodology definition

  According to the latest definition for a software development methodology, modeling language and process are the main parts of a methodology [19], [20]. Therefore, this category is further divided into two parts:

Modeling language

- Simple to learn and use
- (syntax) Accurate and unambiguous models
- (semantic) Consistent and unambiguous models
- Support for different model views
- Traceability
- Reusability
- Analyzability and Executability
- Techniques for tackling model inconsistency
- Techniques for managing model complexity

Process

- Clear, rational, and consistent definition
- Coverage of the generic development lifecycle activities (Analysis, Design, Implementation, Test, Maintenance)
- Practicability and practicality
- Seamlessness and smoothness of transition between phases, stages and activities
- Basis in the requirements (functional and non-functional)
- Testability and Tangibility of artifacts, and traceability to requirements

2. Main aspects of methodology classification

As mentioned before, methodology classification refers to the features a methodology needs to have considering its type or class. Since the principal concern of this paper is agile software development methodologies, this part focuses on agility in software development. The Agile Manifesto and the Agile Principles [10], [21], explain the underlying assumptions and main goals of agility. They have therefore been used as major resources, and lead us to the main features of agility recommended in an agile method:

- Speed
- Short iterations
- Responsiveness to changing requirements at any time
- Active customer representation and collaboration
- Emphasis on personal and managerial issues rather than a heavy process
- Reliance on self-organized teams
- Daily short sessions without documentation
- Emphasis on technical skills
- Technical excellence
- Continuous redesign
- Simplicity
- Flexibility
- Learning
- Responsiveness
- Emphasis on delivering the working product
- Avoidance of comprehensive documentation
- Leanness

3. Main aspects of methodology usage

- Techniques for integration with other methodologies (for those methodologies not covering all development lifecycle activities).
- Umbrella activities: Risk management, Configuration management, Project management, Quality assurance
- Definition of the application scope
- Manageability of complexity
- Extensibility/Configurability /Scalability
- Documentation; including training materials, case studies, empirical evidence, reports which help to understand a methodology's status in the real world.

The above are the main aspects of an agile software development methodology. Thus, it is expected that an evaluation criterion set should contain some criteria to evaluate a methodology with respect to the existence of mechanisms or guidelines dealing with these aspects.

In order to be considered valid, the meta-criteria proposed herein are expected to satisfy the yardsticks of quality that they themselves set out. However, due to the specific abstraction level targeted by the proposed meta-criteria, some of them are not applicable at the meta-criteria level. Furthermore, of those that *are* applicable, the satisfaction of some is not worthwhile and justifiable when applied to meta-criteria. Incorporation of Quantitative Metrics is a good example, as it can easily result in unwarranted complexity in meta-criteria. Consider applying this meta-criterion to the Balance meta-criterion, for instance: we can define a Balance metric based on calculating the number of criteria targeting each relevant aspect and determining the corresponding ratios, but this will just add extra complexity, with little practical value gained in return.

However, the proposed meta-criteria do satisfy the applicable general meta-criteria to an acceptable degree: the meta-criteria are categorized and clearly defined, and easily measured; they yield quantified results, have little overlapping, and provide a balanced perspective to criteria evaluation in that they cover all the main aspects of the agile-software-development domain as well as issues of a more general nature.

4. Evaluation of existing evaluation criteria

The results of evaluation based on the proposed meta-criteria are shown in Tables 3 and 4. In these tables, each evaluation criterion set has been shown with a short name:

Table 3. Evaluation results based on the proposed meta-criteria (except for Comprehensiveness).

Meta-criteria / Evaluation Criteria		Ab& 2002	Ab& 2003	Ko 2003	Wi& 2004	GR 2005	QH 2006
Precision	Detail	◐	◐	◐	●	◐	●
	Unambiguity	●	●	●	●	●	●
	Quantified results	○	○	◐	◐	●	◐
Simplicity	Simple definition	●	●	●	●	◐	●
	Ease of measurement	◐	◐	◐	●	●	●
Quantitative metrics		○	○	○	◐	●	◐
Usability and meaningfulness		●	●	●	●	◐	●
Consistency		●	●	●	●	●	●
Minimum overlapping		●	●	●	◐	●	◐
Generality	Independence from methodology	●	●	●	○	◐	●
	Independence from development model	●	●	●	○	◐	●
Balance	Methodology definition	◐	◐	○	○	○	○
	Methodology Classification	○	○	○	○	○	●
	Methodology Usage	◐	●	◐	◐	○	●

● Acceptable ○ Not Addressed ◐ Partial

1. Ab& 2002: Abrahamsson et al. in 2002.
2. Ab& 2003: Abrahamsson et al. in 2003.
3. Ko 2003: Koskela evaluation criteria for software configuration management.
4. Wi& 2004: Williams et al. for XP practices.
5. GR 2005: Germain and Robillard for comparison of engineering-based and agile methodologies.
6. QH 2006: Qumer and Henderson-Sellers for comparing XP and Scrum.

Table 3 shows the extent to which each evaluation criterion set satisfies the following meta-criteria: Precision, Simplicity, Quantitative metrics, Usability, Consistency, Minimum overlapping, Generality, and Balance. There are three possible values for each result: Acceptable, Partial, and Not Addressed.

Table 4 shows how and to what extent each evaluation criterion set addresses the Comprehensiveness meta-criteria. These especially target the main aspects of agile software development. There are five possible values for each result: Explicitly Total, Explicitly Partial, Implicitly Total, Implicitly Partial, and Not Addressed.

Most of the proposed meta-criteria can be easily applied through looking for examples/counterexamples in the appraised evaluation frameworks. Some meta-criteria, however, yield results that vary widely depending on the characteristics and capabilities of the users of the evaluation frameworks, and are therefore prone to subjective evaluation; the Precision and Simplicity meta-criteria are good examples. In such cases, we have based our judgment on the minimum capability typically expected of the users.

5. Evaluation conclusions

As shown in Table 3 and Table 4, none of the evaluation criterion sets satisfy all the required features addressed by the proposed meta-criteria, and all of them have shortcomings, especially as to comprehensiveness. The following are other immediately visible observations:

1. Criteria are acceptable as to Precision and simplicity of evaluation; however, an evaluation criterion set should be highly precise.
2. All evaluation criteria are in an appropriate state as to usability, consistency, and minimum overlapping.
3. Quantitative metrics are neglected in nearly all of the criterion sets analyzed herein; only one of the evaluation criterion sets addresses quantitative metrics.
4. Almost all criterion sets satisfy Generality at an acceptable level.
5. In all of these criterion sets, Balance suffers because of limited attention to the main aspects defined in the Balance meta-criterion.

Table 4. Evaluation results based on the Comprehensiveness meta-criteria.

Meta-criteria			Evaluation Criteria	Ab& 2002	Ab& 2003	Ko 2003	Wi& 2004	GR 2005	QH 2006
Comprehensiveness	Methodology Definition	Modeling Language	Simple to learn and use	○	○	○	○	○	○
			(syntax) Accurate and unambiguous models	○	○	○	○	○	○
			(semantic) Consistent and unambiguous models	○	○	○	○	○	○
			Support for different model views	○	○	○	○	○	○
			Traceability	○	○	○	○	○	○
			Reusability	○	○	○	○	○	○
			Analyzability and Executability	○	○	○	○	○	○
			Techniques for tackling model inconsistency	○	○	○	○	○	○
			Techniques for managing model complexity	○	○	○	○	○	○
		Process	Clear, rational and consistent definition	◍	◍	○	◍	○	◍
			Coverage of the generic development lifecycle activities	●	●	○	○	◍	○
			Practicability and practicality	●	●	○	○	○	◍
			Seamlessness and smoothness of transition between phases, stages and activities	⊖	⊖	○	○	○	○
			Basis in the requirements (functional/non-functional)	○	○	○	○	○	○
			Testability and Tangibility of artefacts, and traceability to requirements	○	○	○	○	○	○
	Methodology Classification		Speed	○	○	○	○	○	●
			Short Iterations	○	○	○	○	○	●
			Responsiveness to changing requirements at any time	○	○	○	○	⊖	●
			Active customer representation/collaboration	○	○	○	○	○	●
			Emphasis on personal/managerial characteristics rather than a heavy process	○	○	○	○	○	●
			Reliance on self-organized teams	○	○	○	○	○	⊖
			Daily short sessions without documentation	○	○	○	○	○	⊖
			Emphasis on technical skills	○	○	○	○	○	◍
			Technical excellence	○	○	○	○	○	○
			Continuous Redesign	○	○	○	○	○	○
			Simplicity	○	○	○	○	○	⊖
			Flexibility	○	○	○	○	○	●
			Learning	○	○	○	○	○	●
			Responsiveness	○	○	○	○	○	●
			Emphasis on delivering the working product	○	○	○	○	○	●
			Avoidance of comprehensive documentation	○	○	○	○	○	●
			Leanness	○	○	○	○	○	●
	Methodology Usage		Techniques for integration with other methodologies	○	○	○	○	○	○
			Umbrella activities	⊖	⊖	⊖	⊖	○	◒
			Defined application scope	○	○	○	●	○	●
			Manageability of complexity	○	○	○	○	○	○
			Extensibility/Configurability/Flexibility/Scalability	○	●	○	○	○	⊖
			Essential documentation	○	●	○	○	○	○

● Explicitly Total ◍ Implicitly Total ○ Not Addressed
◒ Explicitly Partial ⊖ Implicitly Partial

6. All the evaluation criterion sets are lacking as to comprehensiveness, which is only acceptable for Koskela's framework:
    a. Modeling language has been neglected in all of them.
    b. Process has been covered only partially.
    c. Among these criterion sets, only Qumer and Henderson-Sellers address Agility and define criteria for evaluating methodologies based on it.
    d. Methodology usage aspects have not been considered in most of these evaluation criterion sets. The best is Abrahamsson et al. [2], which is not acceptable itself.

Ultimately, none of these evaluation criterion sets is the best one or even absolutely better than the others. From agility perspective, Qumer and Henderson-Sellers evaluation framework, 4-DAT, is the most complete one. Nevertheless, their attention to the definition view of methodologies is not adequate; from this point of view, Abrahamsson et al.'s evaluation methods are more appropriate.

6. Conclusions and Feature Work

In spite of widespread usage of agile methodologies and the number of existing agile methods, answers to questions concerning the suitability of agile processes to particular projects or identifying method-chunks in an agile method are still difficult to find, if at all existent. In order to answer these challenges and requirements, software development methodology users need to have an appropriate evaluation framework satisfying all their requirements. In this paper, we have evaluated existing evaluation frameworks using our proposed evaluation meta-criteria.

Through this evaluation we have found that existing evaluation frameworks are lacking in several aspects. The most important one is that most of them have not considered agility-related issues in their evaluation. Since agility is the outstanding feature of agile methodologies, an appropriate evaluation framework is expected to contain some form of criteria to evaluate agility in a method. Despite the importance of the usage perspective – as it addresses the main concerns of the project managers – it has been neglected or partially addressed in most of the existing evaluation frameworks. Adequate coverage of quantitative metrics is the next missing feature in evaluation frameworks. Except for one evaluation framework, which is not in an acceptable condition from other views, all others have defined their criteria and explained their evaluation results qualitatively.

This study showed that agile methods' users still need an appropriate evaluation framework to satisfy their needs and answer all their questions. The next step of this study is to define and propose an evaluation framework for agile methodologies that satisfies the proposed set of meta-criteria.

7. References

[1] P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, *Agile Software Development Methods: Review and Analysis*, VTT Publication, Finland, 2002.

[2] P. Abrahamsson, J. Warsta, M. Siponen, J. Ronkainen, "New Directions on Agile Methods: A Comparative Analysis", In *Proc. of the 25th International Conference on Software Engineering (ICSE'03)*, Oregon, 2003, pp. 244-254.

[3] J. Koskela, *Software Configuration Management in Agile Methods*, VTT Publication, Finland, 2003.

[4] L. Williams, W. Kerbs, L. Layman, A. Anton, "Toward a Framework for Evaluating Extreme Programming", In *Proc. of the 8th International Conference on Empirical Assessment in Software Engineering (EASE 04)*, Edinburgh, 2004, pp. 11-20.

[5] C. Jones, *Software Assessments, Benchmarks, and Best Practices*. Boston, MA: Addison Wesley, 2000.

[6] B. Boehm, R. Turner, "Using Risk to Balance Agile and Plan-Driven Methods", *Computer*, IEEE, June 2003, pp. 57-66.

[7] B. Boehm, R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison Wesley, 2003.

[8] E. Germain, P. Robillard, "Engineering-based Processes and Agile Methodologies for Software Development: a Comparative Case Study", *The Journal of Systems and Software*, Elsevier, February 2005, pp 17-27.

[9] A. Qumer, B. Hendersson-Sellers, "Comparative Evaluation of XP and Scrum Using the 4D Analytical Tool (4-DAT)", In *Proc. of the European and Mediterranean Conference on Information Systems (EMCIS)*, Spain, 2006.

[10] K. Beck et al., "Manifesto for Agile Software Development", Available at http://www.agilemanifesto.org.

[11] U. Frank, "A Comparison of two outstanding Methodologies for Object-Oriented Design", *FIT.CSCW*, 1992.

[12] M. Rossi, S. Brinkkemper, "Complexity Metrics for Systems Development Methods and Techniques", *Information Systems*, Elsevier, April 1996, pp. 209-227.

[13] J. M. Karam, R. S. Casselman, "A Cataloging Framework for Software Development Methods", *Computer*, IEEE, February 1993, pp. 34-45.

[14] R. Ramsin, "Evaluation of Object-Oriented Software Development Methodologies", *In Proceedings of the 1st Computer Society of Iran Computer Conference (CSICC'95),* Iran, 1995, pp. 40-50.

[15] R. Ramsin, "The Engineering of an Object-Oriented Software Development Methodology", *Ph.D. Thesis*, University of York, UK, 2006.

[16] A. Sturm, O. Shehory, "A Framework for Evaluating Agent-Oriented Methodologies", In *Proc. of the 5th International Bi-Conference Workshop on Agent-Oriented Information Systems*, 2003, pp. 94-109.

[17] P. Cuesta, A. Gomez, J. Gonzalez, F. Rodrıguez, "A Framework for Evaluation of Agent Oriented Methodologies", *In Proc. of The Conference of the Spanish Association for Artificial Intelligence (CAEPIA)*, Spain, 2003.

[18] K. H. Dam, "Evaluating and Comparing Agent-Oriented Software Engineering Methodologies", *MS Thesis*, RMIT University, Australia, 2003.

[19] D. Avison, G. Fitzgerald, "Where now for Development Methodologies?", *Communications of the ACM*, January 2003, pp. 79-82.

[20] OMG, *Unified Modeling Language Specification (v2.0)*. Object Management Group (OMG), 2004.

[21] Agile Alliance, "Agile Principles", Available at http://agilealliance.org.