A Context-Driven Approach for Guiding Agile Adoption: The AMQuICk Framework

Hajer Ayed, Benoît Vanderose and Naji Habra
PReCISE Research Center
Faculty of Computer Science, UNamur
Rue Grandgagnage 21, B-5000 Namur, Belgium
emails: {hajer.ayed, benoit.vanderose, naji.habra}@unamur.be

***Abstract*—Regarding the proven benefits of agile software development, more and more practitioners are becoming interested in agile methods and have to deal with the complexity and costs of the adoption process. In this context, agile experts argue that prior to any agile method or practice adoption, its relevance to the organization and team should be evaluated to avoid unnecessary implementation efforts and resources. The goal of this research is to investigate a context-driven approach for guiding agile methods adoption: starting from the characterization of the context properties, the approach helps to identify relevant practices and to recommend process customization using adequate rules. The focus in this paper will be on the agile context characterization using relevant, reusable and measurable elements structured in a context metamodel. A purposely simple instantiation is proposed to illustrate how customization rules would be inferred from the context characterization.**

***Index Terms*—agile software development; software process customization; agile context; agile practice selection.**

I. Introduction

Even though the benefits of agile methods have been proved by successful implementations and experiences, the complexity of adopting them is high and requires lots of effort: upper management sponsorship, customer involvement, team empowerment, traditional organizational silos replacement with cross-functional teams, deals with egos and resistance to change, business model arrangement, etc.

To take advantage of the agility benefits and to overcome these common issues, experts and practitioners highlight the necessity to properly adapt practices, deliverables, activities and any other process aspect to avoid unnecessary implementation costs and efforts and to better accommodate the team's specific context and needs.

The agile literature, as explained by Dybå et al. [1], provide a broad picture of adaptation experiences and successful agile implementation but most of them are hardly reusable because they lack of structuring and are often based on experts knowledge and intuitive reasoning: the adaptation decisions are neither documented nor structured nor automated (see section II).

The goal of the *AMQuICk* framework [2][3] is to provide methods and tools to guide the adaptation of agile methods in a more objective, structured and (at least partially) automated way. To that end, it is necessary to record and formalize the intuitive knowledge of agile experts regarding the adaptation of methods to specific contexts so that decision-making may be systematized.

A key to success in this endeavor lies in the exploitation of a formalized and measured representation of the context of an ongoing development process. Given an objective model of the context (including measured attributes), the identification of relevant process elements to recommend to the team maybe more easily exploited by formalized recommendation rules (paving the way towards a complete expert system).

This paper introduces an approach to context-modeling designed to be exploited in such a way and demonstrates how an agile context can be instantiated using relevant measurable elements in order to infer customization rules. The main questions underlying the approach are therefore: (1) how can we model and compose agile processes using reusable components?, (2) what defines an agile software development context and how to model it?, (3) how to retrieve relevant components regarding the context at hand?.

The remainder of this paper focus on context modeling challenges and is structured as follows: Section II presents the existing approaches and context models to guide the customization process. Section III-A presents an overview of the framework that we propose. Sections III-B and III-C provide details on the process specification and context modeling. Section III-D refers to the formalization of the process engineering interpretation rules. An example is presented in Section IV to illustrate the context metamodel instantiation and how customization rules could be inferred from the context characterization. Finally, Section V presents closing comments and future work.

II. Related work

A. Agile Customization

Even though the literature abounds with valuable agile methods tailoring experiences reports [1], most of them are difficult to exploit, because too narrowly linked to a specific situation and often based on experts' knowledge and intuitive reasoning: neither documented nor structured nor tool-supported.

There exist structured approaches that provide practical road-maps to facilitate and guide through the implementation and tailoring process [4][5] but they definitely lack automation: most of them are just documents with guidelines and repeatable steps to follow for effective agile methods implementation. Moreover, the problem with these approaches is that each of them proposes a solution based on only few and prefixed factors influencing the implementation. For example, Cockburn [6] proposes to choose the agile methods among the Crystal family methods according to the *number of people involved* and *criticality* criteria .

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-438-1

Other approaches, such as Mnkandla [7], propose a toolbox for only practices selection and not other process aspects. Moreover, the linkage with project context is only allowed through a predefined methodology selection matrix and project taxonomy matrix. This kind of matrix synthesizes some experts' knowledge therefore preventing any possibility of extension.

Finally, more formalized and tool-assisted approaches, such as Mikulėnas et al. [8], aim to support agile methods adaptation by providing users with rich practices composition mechanisms (e.g., merging, coupling, etc.). However, the choice of suitable practices is only based on the user appreciation: the adaptation decisions are not assisted or derived from context attributes.

B. Agile Context Defined

The software development context refers to all the influential circumstances and variables that affect the work environment of all stakeholders involved in the project life-cycle, e.g.,: *market uncertainty, budget constraints, application domain, project criticality, project duration, team size, familiarity with the involved technology, etc.*

Although the term context has an intuitive meaning for agile practitioners, it's hard to formalize the relevant context variables to support software process adjustments.

Several contextual models to guide the adoption and adaptation of agile software development practices can be found in the literature.

Cockbrun et al. in the crystal family of processes [6] define different processes based on *Product Size*, *Criticality*, and *Skills*.

Boehm et al. [9] define a home ground of agile vs. plan-driven as associated to five critical factors namely, *Product Size*, *Criticality*, *Dynamism* (i.e., requirements change rate), *Personnel* (i.e., level of method understanding [10]) and *Culture* (of the team: thriving on chaos or on order).

Kuchten [11] defines 2 sets of factors that make up the context: factors that apply at the level of the whole organization, and factors that apply at the level of the project. The organization-level factors do influence heavily the project-level factors which should drive the process to adopt. The organization level factors are defined as: *Business domain, Number of instances, Maturity of the Organization, Level of Innovation* and *Culture*. Project-level context factors are: *Size*, *Stable Architecture*, *Business Model* (contracting, money flow, etc.), *Team Distribution*, *Rate of Change*, *Age of System*, *Criticality* and *Governance* (management style).

S. W. Ambler [12], in the Agile Scaling Models (ASM) framework, defines a range of 8 scaling factors for effective adoption and tailoring of agile strategies: *Team size, Geographical distribution, Regulatory compliance, Domain complexity, Organizational distribution, Technical complexity, Organizational complexity and Enterprise discipline*.

Even though the context models reported above have been defined for different purposes (i.e., Crystal family of methods configuration, defining agile vs. plan-driven home grounds, practices adoption guidance and scaling agility to larger scopes), they seem to be more or less similar with only minor variations. They are all composed of context *"factors"* or *"dimensions"* at the higher levels refined in a set of *"properties"* or *"attributes"* at the lower levels.

Based on this observation, our target was to find a way to abstract context modeling in a common paradigm, so that agile process engineers or facilitators (or any other equivalent role) can design their own profile to contextualize process components depending on their own perception. Indeed, the set of relevant context elements to support the software process adjustments is potentially different from an organization to another.

The approach investigated in this paper is an attempt to address the issues mentioned above. The following sections provide an overview of the essential set of components required for context-driven adaptation.

III. AMQuICk Framework

A. Overview

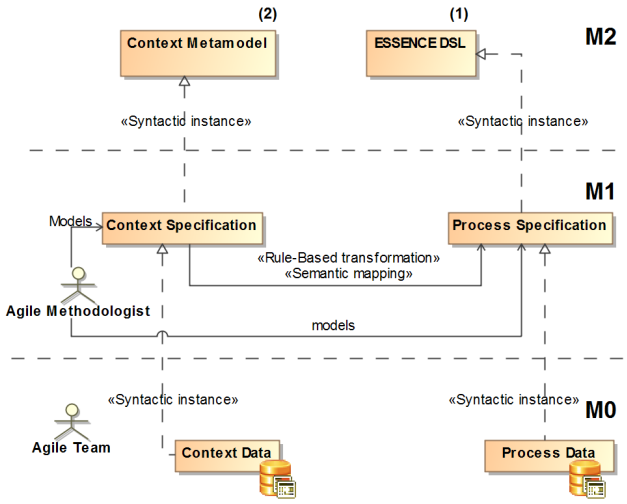

Figure 1: AMQuICk Basic Elements

In order to support the long-term vision of assisted adaptation of agile development processes, it is crucial to complement existing agile tailoring approaches with more objective and systematic guidelines. As explained in Section II, the context appears as a missing link in the formalization process regarding the tailoring of agile methods. The approach we propose is therefore aiming at better formalization of the context, so that it can be exploited further in the process tailoring part of the approach. Practically, the approach relies on a formal (and rule-based) mapping between process models and the related context models.

As illustrated in Figure 1, our approach is model-driven and complies to the Meta-Object Facility (MOF) architecture [13]. At the M2-level lie the two required metamodels: the first dedicated to the context specification, the second to process

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-438-1

modeling. The former is specifically designed for our purpose while the latter takes advantage of the preexisting Essence DSL to define the abstract syntax of agile processes (see Sections III-B and III-C for further details).

While the M0-level is concerned with actual collected data regarding the context and the process, the M1-level is the cornerstone of the approach. At this level, two syntactic instances of the metamodels may be compared and mapped against each other from a semantic point of view. In other words, at this level, a specific piece of context may call for a specific piece of process. This relationship between context and process must be guaranteed by rules derived from the body of knowledge of agile experts referred to in Section III-D.

Implementing this approach is key regarding the elicitation of objective decision-making elements that are needed to guide the evolution and decide which process adjustments to include at the right time. The components illustrated in Figure 1 form the basis of a rule-based system so that the experts can define the crucial context features that influence process adaptation and the practitioners simply enter some information about the project context (by instantiation of the latter features) and get an indication of the most appropriate adaptations for that project.

B. Process Modeling

As explained in Section I, the need for a better flexibility of software engineering motivates the construction of tailored processes to the situation at hand. This discipline is known as Situational Method Engineering (SME).

The kernel of SME consists in composing contextual methods by reusing structured "components" of existing methods. Various techniques can be used including Metamodeling, Domain Specific Languages (DSL), Ontologies, etc.

For the needs of the previously described approach we investigated and compared some of them among which, ISO/IEC 24744 metamodel [14], Software Process Engineering Metamodel (SPEM 2.0) [15] and the recently published DSL Essence 1.0 [16].

The DSL Essence 1.0 (see Figure1) is a Kernel And Language For Software Engineering Methods adopted by the OMG. It was chosen because of its intuitiveness (graphical notation, different level of abstraction: static and operational view), usability at the team level, extensibility and finally because the DSL has been developed among an active initiative which aims to develop a software engineering kernel for both agile and waterfall ways [17].

However, the Essence DSL has to be extended in order to allow for a more structured definition of context-related elements into the process modeling. As explained in [18], quality-related information may be taken into account in order to provide more objective (or a least more structured) elements of context.

C. Context Modeling

In this section, we investigate the second problem introduced in Section I, i.e., what attributes the agile software development context encompass and how it can be captured?

Figure 2 depicts the designed metamodel for context specification. It defines all the concepts (and relationships between them) that may be used in the definition of a context model. The core elements for characterizing an agile context are:

- *"Context Dimension"*: includes the high-level key-concepts of software engineering characterizing the context. Possible instantiations are project, organization, team, endeavor, customer, solution, etc. These instantiations highly depends on the tacit knowledge of agile experts.
- *"Context Property"*: the set of variables underlying a context dimension. For example, the "requirements dimension" may be characterized by the "requirements change" property.

The metamodel also includes concepts designed to describe in details the measurable entities of the context and the nature of the measures themselves (see Figure 2). This subset of the metamodel results from the conceptual alignment of various related works dedicated to define a unified terminology of software measurement metamodels specifically ISO/IEC SQuaRE [19] and the Model-Centric Quality Assessment (MoCQA) [18].

As a result, the metamodel includes all the concepts required to define a well-formed and coherent measurement method: it relies on the notion of measurable entity for which a given measurable attribute has to be mapped to a value (i.e., the measure itself). In the context metamodel, those concepts are replaced with the *"ContextDimension"* (measurable entity) and the *"ContextProperty"* (measurable attribute).

In order to be conceptually correct and allow for the right operation and comparison, the measure has to be identified by a series of variables (i.e., type of value, unit, and scale) that indicates how the sheer value must be understood, compared to other measurement values and interpreted in fine.

The metamodel also emphasizes the difference between sheer measurement values and so-called indicators. Indicators are the key to the approach since they allow bridging the gap between objectified (through measurement) context elements and derived process elements. As explained in [18], indicators are only as useful as their interpretation rules.

In a more traditional quality assessment context, the interpretation associated to a given indicator determines the action to be undertaken in the next steps of the development. Building upon this notion, the *AMQuICk* approach proposes to link the interpretation to process engineering rules (thanks to *"CustomizationRule"*) so that the interpretation of the indicator impacts directly the way the process is to be refined.

D. Rule-Based Process Engineering

Every time an organization is going to develop a project in an agile way, a context profile has to be instantiated using the context metamodel described in Section III-C.

However, a standalone context model has no meaning if not linked with adequate customization abilities. Indeed, the instantiated model only defines the structural properties of

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-438-1

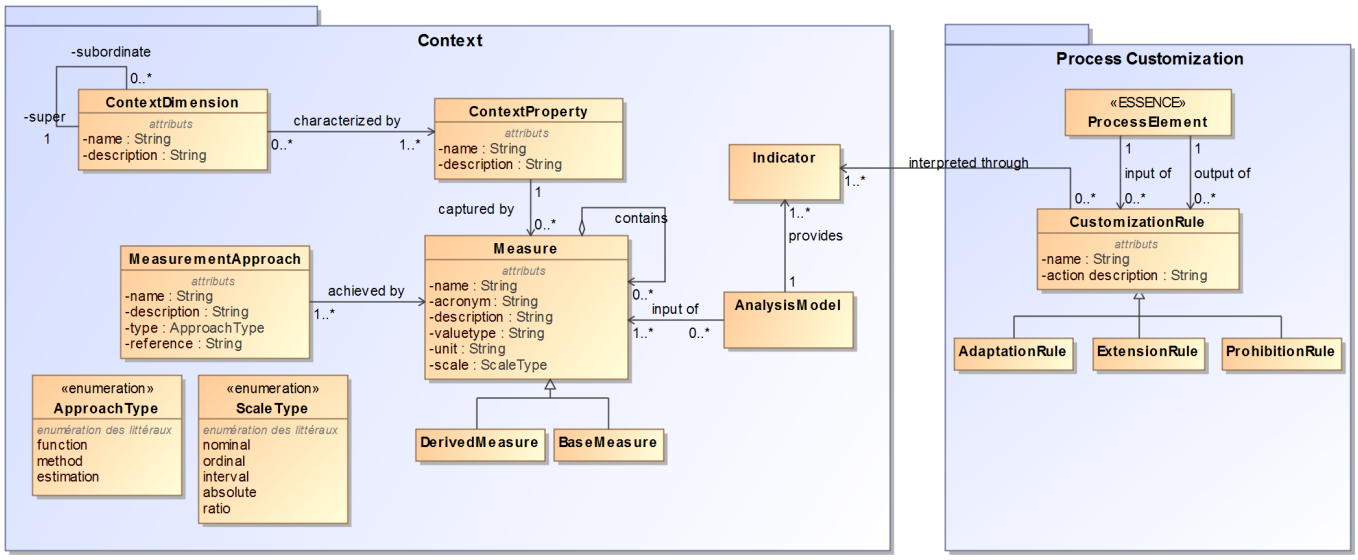

Figure 2: Context Metamodel

the context. The behavioral mechanisms regarding the process engineering have to be elicited.

To do so, the body of knowledge required in the engineering process has to be gathered, i.e., the information about typical project contexts involved in the previous developments of the organization, the information sources of past agile experiences, the tacit knowledge of experts (or experienced people), the previous process configurations and adaptations, the tailoring guidelines (if they exist), etc.

Then, this information has to be structured properly in order to support process engineering decision-making. The relationship between the context and process can be guaranteed by transformation rules. The package *"Process Customization"* of Figure 2 describes the abstract syntax of the interpretation rules engine. *"CustomizationRule"* is associated to:

- a context *"Indicator"* which determines the action or event to be undertaken,
- an input process element (*"LanguageElement"*),
- an output process element (*"LanguageElement"*),

The *"CustomizationRule"* may be of 3 types: an adaptation rule (e.g., iteration length adaptation, start integration earlier, write acceptance criteria before implementation, etc.), an extension rule (e.g., lean value stream map integration, start iterations with model storming sessions, etc.) or a prohibition rule (e.g., collective code ownership and pair-programming are inapplicable in some contexts).

This part of the approach is to be further developed in the future. At this stage of the research, the process customization subset acts as a placeholder that will be refined in the future so that the customization rule generation will be automated, which would not be feasible with the actual formalism. Language elements from the Essence 1.0 DSL in particular "Extension Element" and "MergeResolution" can be reused to compose the output process model [16][17].

IV. Illustration

In order to illustrate an application of the *AMQuICk* contextualizing approach, we propose a small example in Figure 3. The example models the context features used to detect the lack of customer involvement and an adequate adaptation rule.

The customer is here defined as a context dimension characterized by the *customer involvement* context property. This property may be measured in different ways:

- *commitment time*: base measure which refers to the effective time of collaboration between the customer and the development team. This measure is of type *ratio* and is expressed in *datetime*,
- *physical proximity*: base measure which refers to the geographical distance and is expressed in *km*,
- *communication channel*: base measure which refers to the more frequent channel used in the communication between the customer and the team. The measure is of type *nominal* with a range of possible values, i.e., face-to-face, video, phone, mail or documents.

The association between these measures in a relevant analysis model (*involvement analysis* model) provides two indicators of whether the customer involvement is satisfactory or not. In the case of lack of customer involvement detection, a possible process engineering rule to be undertaken is to extend the process with the *[Customer Proxy]* practice [20] (*Enhance Customer Involvement* rule).

The provided illustration in Figure 2 represents a purposely simple context model. Although the ultimate goal of the context model is not to be a visual representation, it is meant to illustrate the kind of objective and measurable information that could be captured by such a model. Similarly, the customization rule provided herein is not meant to be used in such a simplistic way. At this stage of the research, the customization

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-438-1

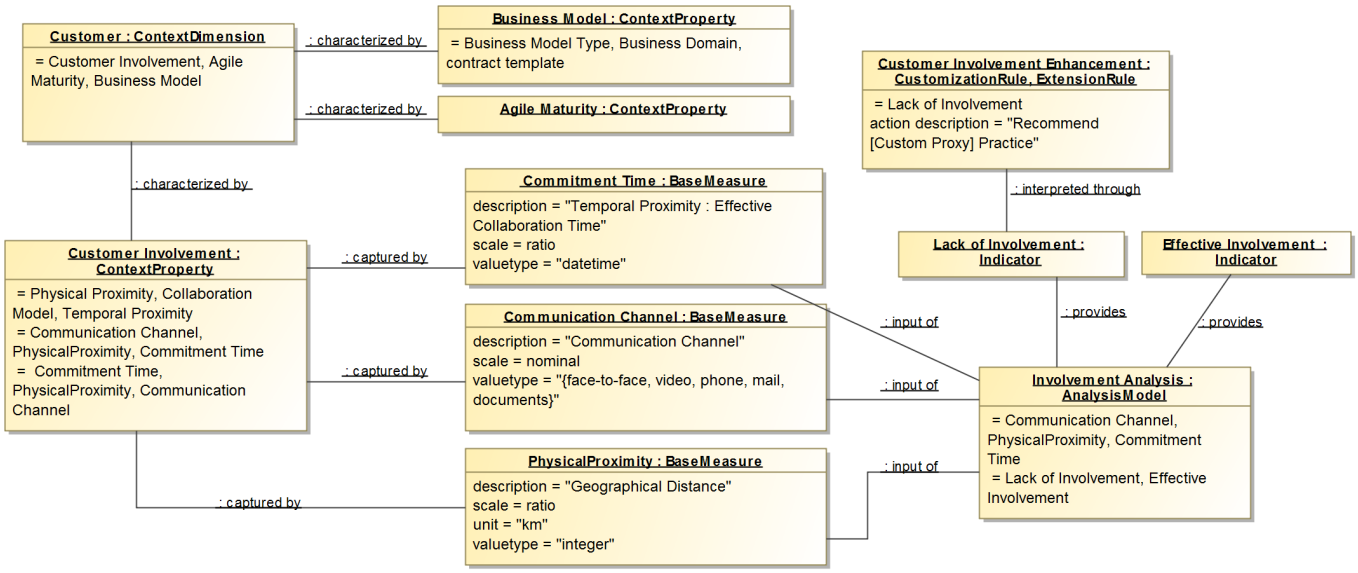

Figure 3: Illustration of a Context Model

rule acts as a placeholder that will be refined in the future so that it can be automated (which would not be feasible with a simple textual representation).

V. Conclusion and Future Work

The approach we propose in this paper aims at supporting decision making regarding agile process (or even any process) evolution in a contextualized way. It relies on an explicit modeling of relevant context-related aspects as well as the use of measurement-based elements to provide objective hints regarding the required process adaptation to undertake. At this stage, the approach focus on the conceptual steps, that is, defining relevant metamodels and conceptual elements.

These conceptual tools are required in order to make the modeling of processes, context elements and process engineering rules possible and coherent. Efforts regarding this conceptual level is still required. For instance, the opportunity to further align the proposed context metamodel with the process metamodel in order to limit the conceptual complexity should be investigated (e.g., the notion of *"Alpha"* from Essence 1.0 and the notion of *"Context Dimension"* proposed in our approach are similar and may be associated).

However, the main added value of this conceptual level lies in the fact that it lays the foundation of a tool-supported methodology. Indeed, the ultimate goal of the approach is to provide an assisted methodology that relies on an expert system. The conceptual approach described in this paper is expected to enable the design of such a system. Indeed, the approach assumes that by exploiting the available agile experiences feedback, we would be able to extract significant and useful knowledge for enhancing the decision-making ability of agile professionals when composing a suitable process.

By exploiting these available agile experiences feedback and linking them to context models, a knowledge database could be populated and enhance the decision-making abilities of the development team. Rules would not only be created by agile experts, they would also be generated through an inference engine.

In turn, this knowledge base could provide the basis of a community-based approach, where continuous feedback, cross-referenced with basic inferences rules, provides an ever improving support for process related decision-making.

References

[1] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and software technology*, vol. 50, no. 9, pp. 833–859, 2008.

[2] H. Ayed, N. Habra, and B. Vanderose, "AM-QuICk: a measurement-based framework for agile methods customisation," in *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on*. IEEE, 2013, pp. 71–80.

[3] H. Ayed, B. Vanderose, and N. Habra, "Supported approach for agile methods adaptation: an adoption study," in *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*. ACM, 2014, pp. 36–41.

[4] K. Conboy and B. Fitzgerald, "Method and developer characteristics for effective agile method tailoring: A study of xp expert opinion," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 20, no. 1, p. 2, 2010.

[5] I. Attarzadeh and S. H. Ow, "New direction in project management success: Base on smart methodology selection," in *International Symposium on Information Technology (ITSIM)*, vol. 1. IEEE, 2008, pp. 1–9.

[6] A. Cockburn, *Crystal clear: a human-powered methodology for small teams*. Pearson Education, 2004.

[7] E. Mnkandla, "A selection framework for agile methodology practices: A family of methodologies approach," Ph.D. dissertation, Faculty of Engineering and the Built Environment, University of The Witwatersrand, 2008.

[8] G. Mikulėnas, R. Butleris, and L. Nemuraitė, "An appraoch for the metamodel of the framework for a partial agile method adaptation," *Information Technology And Control*, vol. 40, no. 1, pp. 71–82, 2011.

[9] B. Boehm and R. Turner, *Balancing agility and discipline: A guide for the perplexed*. Addison-Wesley Professional, 2003.

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-438-1

[10] A. Cockburn, "Selecting a project's methodology," *IEEE Software*, vol. 17, no. 4, pp. 64–71, 2000.
[11] P. Kruchten, "Contextualizing agile software development," *Journal of Software: Evolution and Process*, vol. 25, no. 4, pp. 351–361, 2013.
[12] S. W. Ambler, "The agile scaling model (asm) : Adapting agile methods for complex environments," IBM, Tech. Rep., December 2009.
[13] *ISO/IEC 19502:2005 Information technology - Meta Object Facility (MOF)*, International Organization for Standardization and International Electrotechnical Commission Std., 2005.
[14] ISO, *ISO/IEC 24744: Metamodel for Development Methodologies*, International Organization for Standardisation (ISO) Std., 2007.
[15] *SPEM (version 2.0): Software & Systems Process Engineering Metamodel Specification*, Object Management Group (OMG) Std., 2008.
[16] *Essence (version 1.0): Kernel and Language for Software Engineering Methods*, Online at : http://www.omg.org/spec/Essence/1.0, Object Management Group (OMG) Std., November 2014.
[17] I. Jacobson, P.-W. Ng, P. McMahon, I. Spence, and S. Lidman, "The essence of software engineering: the semat kernel," *Queue*, vol. 10, no. 10, p. 40, 2012.
[18] B. Vanderose, "Supporting a model-driven and iterative quality assessment methodology: The MoCQA framework," Ph.D. dissertation, Ph. D. dissertation, University of Namur, 2012.
[19] W. Suryn, A. Abran, and A. April, "ISO/IEC SQuaRE: The second generation of standards for software product quality," in *7th IASTED International Conference on Software Engineering and Applications*, 2003.
[20] A. Alliance, "Agile alliance guide to agile practices," *Online at: http://www.guide.agilealliance.org/. Last accessed 02/10/2015*.

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-438-1