Electronic Communications of the EASST
Volume 65 (2014)

Proceedings of the International Workshop on Software Quality and Maintainability (SQM 2014)

Implementing a model-driven and iterative quality assessment life-cycle: a case study

Benoît Vanderose, Hajer Ayed and Naji Habra

15 pages

Guest Editors: Lodewijk Bergmans, Tom Mens, Steven Raemaekers
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Implementing a model-driven and iterative quality assessment life-cycle: a case study

Benoît Vanderose[1], Hajer Ayed[2] and Naji Habra[3]

[1] benoit.vanderose@unamur.be
[2] hajer.ayed@unamur.be
[3] naji.habra@unamur.be
PReCISE Research Center
University of Namur, Belgium

Abstract: Assessing software quality through quantitative and reliable information is a major concern of software engineering. However, software is a complex product involving interrelated models with different abstraction levels targeting different stakeholders and requiring specific quality assurance methods. As a result, although Software Quality has gained maturity from a theoretical point of view, the practical quality assessment of software still does not fulfil enough involved actors' expectations.

In order to improve quality assurance in practice, a more integrated approach to assessment is required. This paper describes a case study in which a quality assessment framework (MoCQA) relying on a model-driven and iterative methodology has been used to this end. For a year and a half, the framework has been used by the quality assurance team of a small IT department to maintain and monitor a portfolio of projects in both production and development.

The study shows the feasibility and the relevance of a model-driven and iterative quality assessment methodology in a professional environment. Besides, although its results still require more generalisation, the study provides interesting insights on how such an approach may help ensure a continuous and explicit communication between stakeholders, leading to a more efficient quality assessment.

Keywords: quality assurance, model-driven, iterative approaches, case study

1 Introduction

In the context of a constantly evolving field like software engineering and with the steadily increasing level of complexity of software, software quality has become increasingly delicate to define and assess. As new fields and paradigms of software engineering have been appearing, quality concerns have been dispatched into several different and more or less independent subdomains managed by different stakeholders. Quality assessment has therefore become a concern in every field of software engineering (from requirements engineering to design and coding). As a result, several quality assessment approaches (i.e., quality models, software measurement methods, etc.) have been proposed for the past three decades.

Despite the proficiency of research works addressing quality, the main observation remains the overall misguided or inefficient use of measures in industry, leading to costly [FN99], misused or useless measurement plans. Some surveys (notably [Kas06]) also show that software measurement tends to appeal more to the management than it does to the development team, therefore showing its inability to switch from a control-oriented paradigm to a guidance-oriented paradigm. This inability to satisfy the expectations of all stakeholders may be attributable to the lack of evolution in the way quality assessment is performed.

In this paper, we argue that a key to improve quality assurance is to perform quality assessment through a model-driven and iterative methodology. By envisioning quality assessment as an iterative life-cycle performed in parallel to the software development and allowing for continuous refinements, we expect to circumvent most of the reluctance linked to quality assurance. This would in turn lead to a situation in which each type of stakeholders may contribute at different times of the life-cycle and therefore be satisfied by the quality assurance process. In order to illustrate the feasibility of such an approach, we report a case study in which the model-driven and iterative MoCQA framework [Van12] has been deployed and used in a professional environment.

The study took place in the IT Department (D443) of the "Direction Générale opérationnelle de l'Agriculture, des Ressources Naturelles et de l'Environnement (DGARNE)", one of the department of the public administration of the Walloon Region. This IT Department counts 70 agents (including development teams, maintenance teams and a quality assurance team) and manage a pool of about 100 software applications used by a total of 2400 users in the DGARNE. Except for a few isolated cases, no metric or quantitative assessment of any sort was being used to monitor these products prior to the study. Internal and specific quality standards were used to guarantee the global quality of projects.

The remainder of this paper is organised as follows. Section 2 discusses the benefits of a model-driven and iterative approach to quality assessment. Section 3 describes the framework that has been used during the study. Section 4 and 5 present the study that was carried out as well as its results. Section 6 discusses the limitation of the study and, finally, Section 7 provides conclusions and future work to be addressed.

2 Model-driven and iterative quality assessment

2.1 Quality assessment models

The concept of model-driven assessment is not entirely new in software quality. In many regards, goal-driven measurement methods based on the GQM approach [BCR94] define an implicit model of the measurement to be performed. That measurement model guides the quality assurance and this approach may be considered model-driven. Research efforts regarding quality metamodels [KLN10, GSC+07] or software measurement modelling [MPRG08] also contribute to quality modelling. Those approaches allow the modelling of customised quality models and related quality indicators to fit a specific context. However the function of goal-driven measurement models may be pushed further by using *quality assessment* models designed to provide useful information to the different members involved in the development team as a reference regarding quality goals and related efforts.

As for goal-driven measurement models, the main objective of *quality assessment models* is to assist the quality assurance team in the planning and execution of the quality assessment process for a specific development context [Van12]. The models are therefore designed to record information on the quality goals and the evaluation methods that must be used. However, quality assessment models are also designed to record more specific information on the resources the development team is acting on and to relate these resources to the high-level quality requirements identified in the context. In order to help communicate with the managers or end users, quality assessment models also record information on the way high-level quality indicators should be interpreted and which actions should be taken in the software development process according to these interpretations.

Recording this heterogeneous information in a central model and using this model as a basis for the measurement process is the core of model-driven quality assessment. It pursues the goal of ensuring that the quality assessment performed is meaningful regarding the needs of all the relevant stakeholders. Through quality assessment models, the quality assessment becomes a full-fledged model-driven process in the same way model-driven engineering relies on design-related models used through the implementation process to provide a software product all stakeholders can agree upon.

2.2 Iterative and incremental methodologies

In order to support a continuous model-driven quality assessment from the early stage of development and towards the maintenance process, it is crucial to address the possible emergence of new quality goals, and the refinement of existing assessment methods. Considering software development methodologies, we may see that incremental and iterative approaches (such as the Agile paradigm [BBB+01]) are now widely recognised as beneficial in order to deal with such situations.

One of the main advantages offered by iterative/incremental approaches, is to provide a way to capture and react to the evolution of the context while keeping stakeholders involved [Rea05]. Another advantage of such approaches it to allow mistakes during the course of a process and their correction in a short frame of time [Coc06]. Any activity relying heavily on a human processing is prone to mistakes but addressing them and correcting them as they occur help people learn from the mistakes and is a beneficial process overall [Boo04].

Despite an increasing level of automation witnessed in software metrics, quality assessment still remains a process that relies heavily on human processing (e.g., prioritization of the quality goals, definition of the corrective actions to undertake, etc.) [BMB02]. As such, an iterative management of software quality may be beneficial. As a matter of fact, [Dro96] already shows that quality models should be refined gradually to fit the goals and the context they are used in. The main hindrance to an early quality assessment is the fact that measurement plans often require a certain level of maturity in order to be applied. Relying on an iterative quality assessment process makes the integration of less sophisticated measures possible during the early phases of the development. Then, the methods are refined as the evaluated product gains in maturity. Although the first iterations could integrate very rough and imprecise evaluation methods, they would at least provide indicators regarding the global direction in which the software quality is

heading. On the other hand, addressing quality assessment through an incremental process let the quality assurance team avoid dealing with goals that are not yet clearly stated or measurable entities that are just not mature enough to undergo any relevant evaluation.

3 MoCQA framework

The MoCQA framework has been designed to help plan and support quality assessment during software development (from the early stages of development to the maintenance and evolution processes) in a model-driven and iterative way [Van12].

The framework defines a quality assessment metamodel that provides an abstract syntax for the systematic and consistent design of *operational customised quality assessment models* (or MoCQA models) specifically designed for a defined software project and its particular environment.

3.1 MoCQA models

The main goal of MoCQA models is to centralise the relevant information to support the quality assessment process. Once defined, a MoCQA model takes the role of a map that guides the execution of the quality assessment process and the subsequent exploitation of its results. It also serves as a central mechanism to the inclusion of any relevant stakeholder's expectation in the quality assessment process. Concretely, MoCQA models aim at providing the required support thanks to the combination of:

1) A hierarchy of quality goals specifically designed for a given development environment (i.e., taking into account the specific environmental factors of the software project and the quality requirements of its stakeholders);

2) a set of customised measurement/estimation methods designed to monitor the level of satisfaction of the various quality goals;

3) a structured and detailed definition of the resources targeted by the measurement/estimation methods, taking into account their relations to each other and the multidimensional nature of the software project (i.e., multiple levels of abstraction/maturity for the resources).

Contrary to traditional quality models defining quality for a specific product, a MoCQA model extends this limited scope by documenting all the relevant assessment-related aspects for a given project (i.e., what/how/why/for whom we measure and inspect different parts of the project).

The underlying quality assessment metamodel has been designed to allow the alignment, tailoring and integration of quality models and measurement/estimation methods coming from different sources. It therefore grants that MoCQA models are *customised* quality assessment models. The quality assessment metamodel also supports the detailed characterisation (i.e., relation between quality goals and stakeholders, status of a given measurement/estimation method regarding its validation, etc.) of the information contained in MoCQA models so that they may be regarded as *operational* quality assessment models. Finally, as an abstract syntax, the quality assessment metamodel eases the design and revision of the models. Therefore, MoCQA models

are not set in stone and are bound to evolve during the software development life-cycle through their own quality assessment life-cycle.

3.2 MoCQA methodology

The introduction of MoCQA models brings the quality assessment process closer to the model-driven engineering of the product itself (i.e., design of a model based on elicited requirements, "implementation" of the quality assessment process through the measurement plan, "testing" of the quality profile with regard to the needs of the stakeholders). As a result, it is necessary to bind this conceptual model (mainly designed to communicate among stakeholders) to the actual (and possibly tool-assisted) measurement process. Designing the model itself and acquiring the necessary knowledge from the stakeholders is another challenge raised by the model-driven nature of the approach. Finally, the process involves a systematic reflection on the quality assessment process.

In consequence, the framework defines a dedicated assessment methodology designed to support a quality assessment life-cycle built upon the design, exploitation and evolution of MoCQA models. In order to implement an iterative and incremental approach, the MoCQA methodology breaks the overall quality assessment life-cycle into successive quality assessment cycles, defined as *"sequences of quality-related activities beginning with the planning of the assessment and leading to the actual assessment of a software project"* [Van12]. Each quality assessment cycle therefore results in a set of decisions made by the development team about the forthcoming activities regarding the development life-cycle and the next quality assessment cycle. The fact that the MoCQA methodology breaks down the process into iterative quality assessment cycles allows for a systematic revision of the quality goals and assessment methods. At the end of each cycle, the quality assurance team needs to reflect on the assessment performed so far and, together with the stakeholders, decide if the indicators and the way they are defined are relevant.

Additionally, the quality assessment methodology defined by the framework decomposes each quality assessment cycle into five successive steps (Figure 1) described below.

1. Acquiring contextual knowledge: This step focuses on the elicitation of relevant contextual information on the software development environment and on the specific quality requirements, as well as the classification of involved stakeholders.

2. Designing the MoCQA model: This step focuses on the creation and structural validation of a MoCQA model by instantiation of the quality assessment metamodel.

3. Tailoring of the measurement plan. This step addresses the definition of practical guidelines for the measurement and quality assessment, based on the conceptual definitions provided in the MoCQA model.

4. Assessing the software project: This is the step where the actual measurement-related and quality-related data (i.e., measurement results and indicators) are collected in order to produce a quality profile of the software project.

5. Exploiting the quality profile: In this step the quality indicators are interpreted and used as input of the decision-making process related to the remainder of the development and/or the evolution processes and to the next quality assessment cycles.

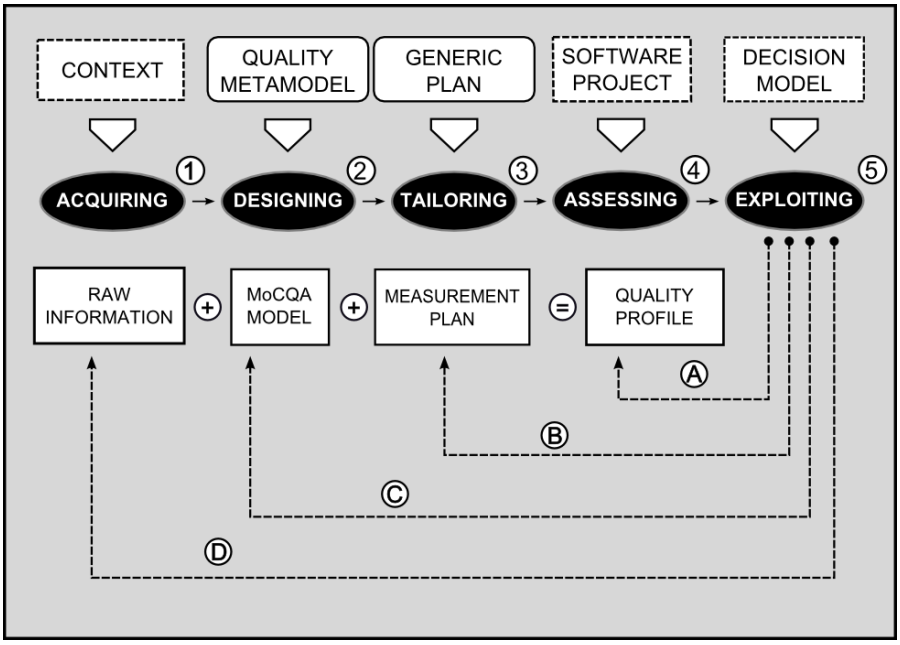

Figure 1: The MoCQA methodology

Through this assessment methodology, the framework provides the support needed to produce coherent and structurally valid MoCQA models. The approach supports an effective use of measurement (i.e., a measurement that is tailored according to the goals of the stakeholders and focus on the satisfaction of their quality-related information needs).

4 Description of the study

4.1 Objectives

The main objective of this study was to investigate whether the implementation of a model-driven and iterative approach such as MoCQA would fulfil the need of continuous quality assessment and improvement of the IT department and be manageable by its quality assurance team. Showing the applicability of such an approach requires to ensure that each step of the proposed methodology could be carried out. Another goal of the study was to determine whether the iterative and model-driven methodology would be well received by all stakeholders involved in the development and maintenance process (i.e., developers, designers, etc.).

4.2 Preliminary phase

A preliminary learning phase was required to help the quality assurance team adopt the concepts of the framework. This learning phase was performed through several meetings with the quality assurance team leader, on the basis of the existing MoCQA documentation. In turn, the team leader was in charge of informing his team (constituted of 4 additional members). This learning

phase ultimately gave birth to an industrial MoCQA deployment guide [Han12]. Following the learning phase, the quality assurance team of the D443 department started applying the MoCQA framework on a daily basis.

4.3 First quality assessment cycle

4.3.1 Acquisition

Amongst the many possible stakeholders to include in the process, 5 stakeholders were selected by the quality assurance team. This selection was based on the availability and role of the actors. The stakeholders were classified as either applicative stakeholders (i.e., any stakeholder that has to act on the software applications, regardless of his specific role in the process) or management stakeholders. Out of the 5 stakeholders, 3 were coming from management and 2 were applicative stakeholders (i.e., development team leaders). The acquisition step was performed by the quality assurance team leader, through a round of individual interviews with each stakeholder. This process was formalised as a series of internal reports.

This initial round of interviews lead to the elicitation of 26 quality goals/requirements. They were classified, organised and prioritised with the help of the head management of the D443 department, who may therefore be considered as an additional stakeholder. The priority was given to the "reliability" requirement for the first quality assessment cycle.

4.3.2 MoCQA model Design

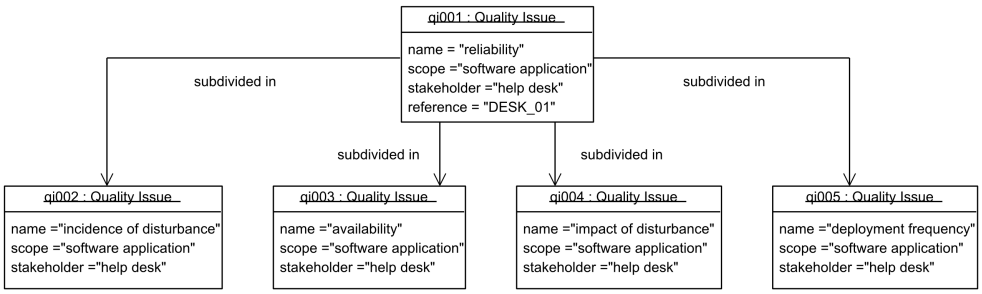

Figure 2: Example of quality issues expressed during the case study

Based on the structured list of quality requirements, the hierarchy of **quality issues** (i.e., *"quality goals characterised by the quality factor they embody, the part of the software project they are relevant for, the stakeholders they are defined for, the indicators used to assess how they are satisfied and the way these indicators should be interpreted"* [Van12]) was designed (Figure 2). As we may see, although the "reliability" quality factor may appear to originate from the ISO/IEC 9126 quality model [ISO01], it is fact inherited from the internal standard of the organisation. Therefore it is decomposed a the following series of specific sub-issues:

- Incidence of disturbance

- Availability (of the software application)
- Impact of the disturbance
- Deployment frequency

These quality issues encompass all the relevant reliability aspects of a software application in production in the environment of the case study. The first quality issue is concerned by the frequency of unexpected behaviours from the software application. The second quality issue complements the first and is concerned by the overall availability of the application over time. The third issue intends to measure the criticality of the disturbances. Finally, the deployment frequency quality issue intends to provide a sense of the number of times the system has to be modified and re-deployed, following a major disturbance. Note that the names, although non conventional are inherited from the internal standards but could be aligned with other standards (e.g., the availability in this context may be aligned with the fault-tolerance characteristic of the ISO/IEC quality model).

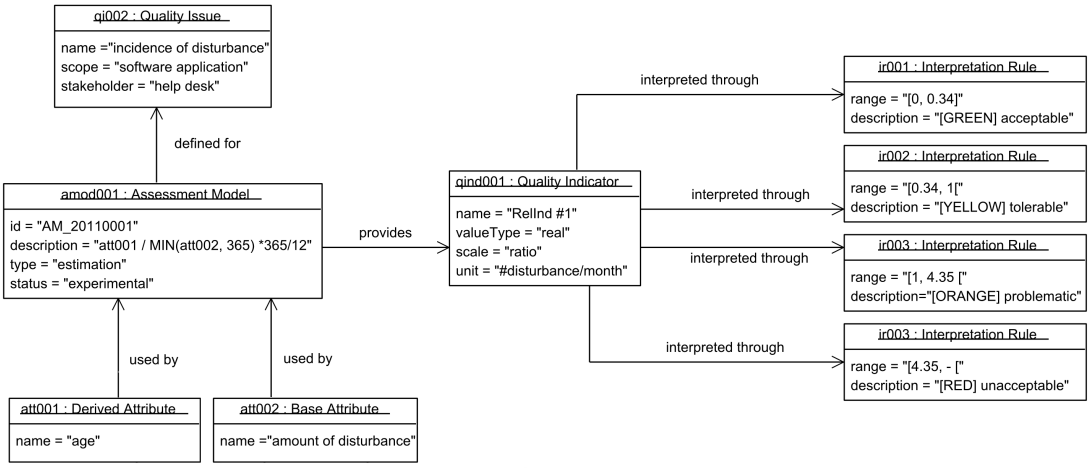

Figure 3: Example of quality indicator expressed during the case study

During the first assessment cycle, only the “incidence of disturbance” quality issue was addressed. The MoCQA model was completed with the description of the measured entities, measurement methods, functions and assessment models required to define relevant quality indicators for this issue. Figure 3 provides an excerpt of the MoCQA model showing the design of the quality indicator for the issue. As we may see, this quality indicator was based on two attributes of the assessed application (i.e., age and amount of disturbance). For this assessment model, the measurement methods were based on the behavioural observation of the assessed application (in order to evaluate the amount of disturbance) as well as a function deriving the age of the application based on its deployment date and the current date of the assessment (as shown in the complete model [Van12]). At this stage, interpretation rules were provided based on “educated guesses” of the quality assurance team members.

4.3.3 Measurement plan tailoring

Providing actual procedures to acquire measurement data was straightforward for the first indicator. Two "repository-mining" procedures were defined. Therefore, the relevant repositories to interrogate (one for the acquisition of the original deployment date and one for the report of disturbance) were identified by the quality assurance team. Specific SQL queries were also designed for each of the measurement methods. All other computations were performed manually, although the functions and assessment models were formalised in C#, in prevision of a future automation of the process. The data collection was planned using spreadsheets.

4.3.4 Assessment and Exploitation

Based on the MoCQA model, the members of the quality assurance team were able to apply the model to the 56 software projects selected during the acquisition step. The first exploitation step was performed with the management stakeholders. They were explained the quality assessment process on the basis of the MoCQA model. The quality assessment process was agreed upon and the assessment results analysed. The decisions taken on the basis of this first quality assessment cycle were mainly related to the continuation of the quality assessment life-cycle. Modifications in the interpretation rules, including the addition of specific recommendations regarding required corrective actions, were proposed. However, the assessment results were perceived positively by the stakeholders as confirming intuitions on several software applications of the pool.

4.4 Continuation of the quality assessment life-cycle

The next quality assessment cycles focused on the refinement of the MoCQA model in order to support corrective actions. During the second quality assessment cycle, exploitation occurred with the contribution of applicative stakeholders. New quality issues were added with each new quality assessment cycle. At the end of the study, 14 quality issues were monitored with the support of the MoCQA framework.

5 Results and discussion

During the one year and a half lifespan of the study, each step of the MoCQA methodology (Section 3.2) has been applied several times. The quality assurance team leader reported his progress and results to the management of the D443 department on a regular basis, in order to define if the course of the project was considered satisfying and should be continued. Subsequently, the quality assurance team leader provided us with reports on the events. Details on these reports are available in [Han12]. The relevance of the model-driven and iterative approach has therefore been assessed through the feedback of the stakeholders and the quality assurance team leader. No quantitative data has been collected to evaluate the *level* of satisfaction of the stakeholders but the application of the framework was not discarded by the management at any point. Based on the reports provided by the quality assurance team leader, several points may be noted:

1) The iterative and incremental aspects of the methodology have been accepted and applied. Although no case of quality indicator or measurement/estimation method deprecation was observed

during the course of the study (and therefore no observation on how the framework handles such cases), the apparition of new quality requirements leading to new quality indicators occurred several times and was supported by the methodology.

2) The assessment performed on the basis of the MoCQA methodology was considered relevant according to both the head management and the quality assurance team. The quality indicators defined during the quality assessment life-cycle of the study were accurate in their support for refinement. The quality assessment performed based on the MoCQA models led to the inspection and maintenance of several software systems and to the refactoring of the help desk supporting repository.

3) The deployment of the MoCQA framework in the context of the IT department D443 also allowed to determine how manageable the assessment methodology is in terms of efforts. Due to organisational requirements, efforts where recorded and communicated to us for the first quality assessment cycle of the two first quality issues investigated by the quality assurance team. For those two quality assessment cycles, the efforts were estimated to an average of 10 man-days[1] per cycle. These efforts were reviewed by the quality assurance team and the management and considered acceptable (i.e., not inducing an unacceptable overhead). The main overhead was identified as the initial learning phase, evaluated to 14 man-days.

4) The framework has since been integrated in the D443 department as a full-fledged quality assessment support for the quality assurance team.

Regarding our initial goals (Section 4.1), the study proves positive on both aspects. The model-driven nature of the approach helped define the specific quality requirements in the studied environment. Besides, the assessment results were well received and led to actual actions carried out in the studied environment. The framework was also reported to be used without any hindrance by the quality assurance team, past the learning phase. The effort estimation, although not providing general results, tends to show that the overhead induced by the methodology is not a stumbling block, in comparison to the benefits it provides. Additionally, the reports provided us with several observations about the use of the MoCQA framework and Software Quality in general.

5.1 Impact of the use of quality indicators

During the course of the study, we had the opportunity to observe the impact of the introduction of formalised quality indicators in the context. At the end of the first quality assessment cycle, the assessment results provided stakeholders with unsurprising conclusions. The problems reported by the quality assessment model were mostly known or sensed to some level by the management stakeholders. However, the introduction of quality indicators and the rationale behind these values helped reinforce the motivation to take actions in order to solve the problems. Although the indicators introduced in the first quality assessment cycle were not very specific or refined, their impact was already important. Moreover, the notion of iterative quality assessment life-cycle guarantees that problems reported at the beginning of the process will be reported again recurrently. This iterative mechanism acts as a reminder of known problems.

[1] In the context of the D443 department, a man-day is assimilated to 7.6 hours of work for 1 employee

The quality assessment process itself may also lead to interesting conclusions that impact the environment. Since the exploitation step analyses both the assessment results and assessment process, it is possible to report valuable information while trying to improve the quality assessment process. For instance, the end of the first quality assessment cycle showed that the collection of data was hampered by the lack of a centralised repository to find the necessary data (i.e., mainly the reported disturbances).Therefore, although no actual measurement was performed during this cycle, the exploitation step still led to a corrective action (i.e., centralise the information on the various software applications).

Regarding the interpretation of the quality indicators, the iterative methodology was also well received. The caveat with indicators in general is to avoid interpreting them without a critical view on what reality they encompass. The fact that the framework allows for a critical revision of quality indicators (e.g., modify the threshold of over-demanding quality indicators) and provides the formalised rationale behind the quality indicator was beneficial for the fine-tuning of quality assessment over time.

5.2 Human aspects

The course of the study also helped confront the approach to the perception of the various stakeholders. Some reluctance or scepticism towards the introduction of a formalised quality assessment framework appeared during the first and second quality assessment cycles. This circumspection took different forms depending on the type of stakeholders.

The management mainly worried about the return on investment of the application of the MoCQA methodology. The concern was thus the amount of time and effort the deployment of the framework would require. The conclusion of the first quality assessment cycle provided reassuring answers to this concern.

The applicative stakeholders (i.e., development team leaders) were more concerned by the quality indicators themselves, raising the issue that the quality indicators may not reflect the truth of the applications they were responsible for. This reaction is not surprising since individuals tend to dislike the notion of quality control [WR05]. During the second exploitation phase, explaining to them the fact that taking into account their feedback on the results and interpretation was part of the process helped solve the issue.

A transversal issue regarding the deployment of the framework was also raised during the first quality assessment cycle. This issue was related to the perceived "subjectivity" of the quality assessment process. The choice of reliability as a first quality issue was questioned by other stakeholders. The same occurred with the way quality issues were assessed. This concern was integrated into the decision-making regarding the quality assessment process. Therefore, the input of stakeholders that were not concerned by the reliability was used to decide which quality issue should be investigated next. The assessment process for reliability was maintained after exchanges between the quality assurance team and the aforementioned stakeholders

As expected, another important aspect of the deployment of a quality assessment plan was to communicate on the target of the assessment. The key to a successful assessment is to prevent individuals from feeling assessed themselves. The availability of the MoCQA model provided a transparent way to clearly define the goals of the assessment. Through the consultation of

the model, each member of the department (even if they are not listed as stakeholders) may understand the process. MoCQA models provide many constructs but clearly none of them is designed to assess individuals. Therefore, the quality assessment process was well received in the context of the study.

5.3 Stakeholder classification

As explained previously, a simple classification was proposed in the context of IT department D443. This dual categorisation turned out to be sufficient during the course of the case study. The two categories of stakeholders clearly elicited different goals and, as seen in the previous section, different worries regarding the quality assessment process.

The dual classification management/applicative stakeholders also led to an interesting observation. Indeed, the way measurement and assessment results are introduced to the type of stakeholder varies slightly. Basically, we distinguished two tendencies:

1) Managerial stakeholders are more prone to react positively to dashboards. Although the presence of the MoCQA model itself is reassuring, the outcome management stakeholders are expecting is a set of indicators.

2) Applicative stakeholders are more prone to react negatively to dashboards. Providing a set of values to the individuals that actually act on the software applications raises concerns on the origin of the values and how they were computed. In that case, the support of the MoCQA model helps provide a good understanding of the rationale behind the indicators in a format that is familiar to the applicative stakeholders (i.e., models).

5.4 Support from the management

The case study also showed that quality assessment must be management-driven in order to be productive. Although the framework provides many elements to counter the reluctance or the scepticism from the development team (i.e., participative and iterative methodology), the framework must be applied with the full support of the management. During the course of the study, the support from the management helped the quality assurance team motivate and decide the development team to take part in the quality assessment and improvement processes. This observation reinforces the considerations provided by [WR05]. The fact that each quality indicator is defined with a given purpose (originating from the management) helps reinforce the perception that quality assessment is a useful process. Additionally, the management has to clearly support the guidance-oriented perspective of quality assessment. The fact that the management supported the deployment of a framework that relies on this “guiding over control” philosophy helped greatly in reassuring the applicative stakeholders in the studied environment.

6 Threat to validity

Although the results of this case study are positive, they only show that the approach is applicable in this specific context. The effort estimation cannot be generalised at this point since this aspect is highly sensitive to the context of use and the complexity of the designed MoCQA model. The

support provided by the management was crucial to the success of this deployment. Besides, the views of the quality assurance team (i.e., guiding over control) were already close to the underlying concepts of the framework. In other words, the environment of the case study was suited to introduce a model-driven and iterative framework such as MoCQA. Results therefore call for experimentation in other contexts in order to generalise these results.

Additionally, the case study does not provide quantitative data regarding the criteria that have been assessed. Future industrial case studies should focus on obtaining more quantitative results regarding the validation process. Obtaining data regarding the productivity and effort in an industrial context is a complicated task. Each project is unique and therefore, the comparison of quality assessment efforts across projects is not relevant. Future studies could however address the problem of quantitative validation thanks to more structured approaches such as satisfaction surveys to determine the level of satisfaction of the stakeholders.

7 Conclusion and future work

The case study carried out in the D443 department reinforces the hypothesis that model-driven and iterative approaches allow for more focused and relevant quality assessment. As we have seen, the iterative nature of the framework deployed in the D443 department provided several advantages regarding the success of its adoption. The methodology helped circumvent some reluctance from the stakeholders and provided a way to react to their input in a satisfactory way. The availability of a quality assessment model upon which the assessment would be performed also insured a continuous and explicit communication between all the actors.

Among the risks inherent to such approaches, we have seen that the learning curve was the most notable overhead during the study. Based on this observation, a way to improve the approach may be to integrate it more tightly with already known patterns such as the Agile paradigm. Explaining the concepts of the iterative quality assessment methodology in terms of Agile concepts would help reduce the learning curve. As it is, the methodology proposed by the MoCQA framework already bears several similarities with Agile concepts. For instance, the exploitation step may be assimilated to a retrospective while a cycle is basically a quality-oriented Agile iteration. Furthermore, like the MoCQA framework, Agile quality management promotes the guidance-oriented perspective and the ”inspect-and-adapt” cycles. Efforts should be carried out to make these similarities explicit.

These similarities could also contribute to the field of Agile methods customisation. Indeed, the problem of customising Agile methods is to provide objective ways to select the adequate methodological elements [AVH12]. A model-driven Agile Methods quality-Integrated Customisation frameworK (AM-QuICK) integrating concepts from the MoCQA framework has already been proposed in [AVH12]. AM-QuICK comprises an agile metamodel adapted from well-known process metamodels (i.e, SPEM, OPF and SMSDM [HG05]), designed to support agile methodologists during the construction of context-specific methods and to provide guidance throughout their assessment and refinement. The agile metamodel integrates a subset of the MoCQA metamodel in order to map the measurement process with the software process elements. For instance, in the case of an agile method construction based on measurement values, the selection of the appropriate process element may be regarded as a hierarchy of *quality is-*

sues. The stakeholders for these quality issues are the "agile methodologists" whereas their scope would be the different part of the project where specific process elements are used. This integration could be pushed forward regarding the integration of the MoCQA framework with AM-QuICK. Indeed, the MoCQA methodology, due to its iterative nature, could be merged with the agile methods customisation framework life-cycle which is based on the Quality Improvement Paradigm (QIP).

Finally, at this stage, we cannot guarantee that every context will allow a suitable integration of a model-driven and iterative quality assessment. Only through repeated empirical studies in various contexts will the approach collect enough evidence of its advantages, or reveal other shortcomings that the approach needs to overcome. Additionally, the case study described in this paper mainly focused on the feasibility of quality assessment processes relying on the MoCQA framework. Future case studies should therefore investigate the *efficiency* of the approach (i.e., whether or not the approach actually increases the productivity and the cost-effectiveness of quality assessment).

Acknowledgements: This research has been co-funded by the European Regional Development Fund (ERDF) and Wallonia, Belgium. The research is also partially supported by the e-Government Chair of the University of Namur.

Bibliography

[AVH12] H. Ayed, B. Vanderose, N. HABRA. A metamodel-based approach for customizing and assessing agile methods. In *Proceedings of the 8th International Conference on the Quality of Information and Communications Technology (QUATIC 2012)*. 2012.

[BBB+01] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas. Manifesto for Agile Software Development. 2001.

[BCR94] V. Basili, G. Caldiera, D. H. Rombach. The goal question metric approach. 1994.

[BMB02] L. C. Briand, S. Morasca, V. R. Basili. An Operational Process for Goal-Driven Definition of Measures. *IEEE Trans. Softw. Eng.* 28(12):1106–1125, 2002.

[Boo04] G. Booch. *Object-Oriented Analysis and Design with Applications (3rd Edition)*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[Coc06] A. Cockburn. *Agile Software Development: The Cooperative Game (2nd Edition) (Agile Software Development Series)*. Addison-Wesley Professional, 2006.

[Dro96] R. G. Dromey. Cornering the Chimera. *IEEE Softw.* 13(1):33–43, 1996.

[FN99] N. E. Fenton, M. Neil. Software metrics: success, failures and new directions. *J. Syst. Softw.* 47(2-3):149–157, 1999.

[GSC+07] F. García, M. Serrano, J. Cruz-Lemus, F. Ruíz, M. Piattini. Managing software process measurement: A metamodel-based approach. *Information Sciences* 177(12):2570–2586, June 2007.

[Han12] S. Hanoteau. Déploiement de l´approche MoCQA en environnement professionnel. Master's thesis, University of Namur, 2012.

[HG05] B. Henderson-Sellers, C. Gonzalez-Perez. A comparison of four process metamodels and the creation of a new generic standard. *Information and software technology* 47(1):49–65, 2005.

[ISO01] ISO/IEC. 9126-1, Software engineering - product quality - Part 1: Quality Model. 2001.

[Kas06] M. Kasunic. The State of Software Measurement Practice: Results of 2006 Survey. Technical report, Software Engineering Institute, 2006.

[KLN10] M. Klaes, C. Lampasona, S. Nunnenmacher. How to Evaluate Meta-Models for Software Quality? In *Proceedings of the 20th International Workshop on Software Measurement (IWSM2010)*. 2010.

[MPRG08] B. Mora, M. Piattini, F. Ruiz, F. Garcia. SMML: Software Measurement Modeling Language. In *Proceedings of the 8th Workshop on Domain-Specific Modeling (DSM'2008)*. 2008.

[OMG08] OMG. Software & Systems Process Engineering Metamodel Specification (SPEM) version 2. 2008.

[OPF09] OPFRO. Open Process Framework. 2009.

[Rea05] D. Read. Iterative development: Key technique for managing software developments. In *Proceedings of ICT WA '05*. 2005.

[Van12] B. Vanderose. *Supporting a model-driven and iterative quality assessment methodology: The MoCQA framework*. PhD thesis, University of Namur, 2012.

[WR05] L. Westfall, C. Road. 12 Steps to Useful Software Metrics. *Proceedings of the Seventeenth Annual Pacific Northwest Software Quality Conference* 57 Suppl 1(May 2006):S40–3, 2005.