Requirements certification for offshoring using LSPCM

2010 Seventh International Conference on the Quality of Information and Communications Technology

Requirements Certification for Offshoring using LSPCM

Alexander Serebrenik*, Amrita Mishra†, Thomas Delissen*, Martijn Klabbers*
*Technische Universiteit Eindhoven, Den Dolech 2, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
a.serebrenik@tue.nl, t.a.delissen@student.tue.nl, m.d.klabbers@tue.nl
† Capgemini, Papendorpseweg 100, 3528 BJ Utrecht, The Netherlands
Amrita.Mishra@Capgemini.com

***Abstract*—Requirements hand-over is a common practice in software development offshoring. Cultural and geographical distance between the outsourcer and supplier, and the differences in development practices hinder the communication and lead to the misinterpretation of the original set of requirements.**

In this article we advocate requirements quality certification using LSPCM as a prerequisite for requirements hand-over. LSPCM stands for LaQuSo Software Product Certification Model that can be applied by non-experienced IT assessors to verify software artifacts in order to contribute to the successfulness of the project. To support our claim we have analyzed requirements of three offshoring projects using LSPCM. Application of LSPCM revealed severe flaws in one of the projects. The responsible project leader confirmed later that the development significantly exceeded time and budget. In the other project no major flaws were detected by LSPCM and it was confirmed that the implementation was delivered within time and budget.

Application of LSPCM to the projects above also allowed us to refine the model for requirements hand-over in software development offshoring.

***Index Terms*—requirements, certification, offshoring, quality gate**

I. Introduction

Offshoring software development activities is a popular approach seeking to increase the value delivery of IT investment. Increasing the value delivery, however, might turn out problematic in practice. Of particular concern are the hidden costs of globally distributed models of working, incurred by the demand of understanding and communicating the true business needs across organizational and cultural boundaries.

Business needs are commonly communicated in the form of requirements documents. Miscommunication of the original requirements is known to be the major reason for project failure [1], and for offshoring the second largest risk factor according to project management professionals [2]. Cultural differences between outsourcer and supplier, geographical distance, differences in attitude, levels of training, frame of reference; they form the root for failures and risks. At the handover moments these differences cause miscommunication. Therefore, we focus on assessing quality of offshored requirements documents as a quality gate; for reasons stated above we think that requirements quality certification at requirements hand-over is essential for offshored development.

The possible benefits of requirements certification are numerous; it helps organizations to obtain certainty about or confidence in quality of the requirements documents. A third party trusted by both the supplier and the customer, should be able to produce an objective and complete assessment, i.e., requirements certification can be seen as a form of an independent verification and validation (IV&V) [3]. It shares similar advantages in that it provides a stable basis to solve conflicts of interest and does not immediately suffer from a weakened quality assurance due to time pressure, see Section V. Provided that a well-established approach is followed, a third-party assessment can lead to a quality certificate being issued. A certificate can help to verify and certify legislative compliance. Specifically, by the offshored development certification can help offshoring partners, both the outsourcers and the suppliers, to convince the other partner that deliverables are of acceptable quality. Therefore, in this paper we advocate requirements quality certification by an independent third party institute as a prerequisite for requirements hand-over.

To support our claim we did a postmortem analysis on the requirements of three offshoring projects which were not deployed yet and for two first cases predicted their successfulness using LSPCM [4], [5]. Section II describes more details about LSPCM. Section III discusses the application of LSPCM to three offshoring projects. Based on these projects in Section IV we propose improvements to LSPCM. Section V reviews the related work and Section VI concludes with the improvements and benefits of LSPCM in the context of offshoring.

II. LSPCM

LSPCM, LaQuSo Software Product Certification Model, is a software product certification model introduced in [4] and further developed in [5]. The emphasis is on the quality of the product, not the process or people. In this section we briefly review LSPCM—the reader is referred to [4], [5] for further details of the model. The complete model is described in [6]. LSPCM can be seen as supporting the verification part of the IV&V activities: it aims at "determining whether the product at each step in the development cycle (a) fulfills all the requirements levied on it by the previous step and (b) is internally complete, consistent and correct enough to support the next phase" [3]. LSPCM is partly based on the ISO/IEC 14598 and ISO 9126 standards and can therefore be compared with these and the newer ISO/IEC 25000 (Software product Requirements and evaluation – SQuaRE) that combines and

978-0-7695-4241-6/10 $26.00 © 2010 IEEE
DOI 10.1109/QUATIC.2010.30

relates the two previous standards. In contrast to [3] and [7] application of LSPCM is more efficient and can also be used to assess more common information system development. A more detailed comparison of LSPCM and existing IV&V approaches can be found in Section V.

A. Structure of LSPCM

LSPCM is applicable to the wide variety of types of software artifacts, from requirements documents to tests, known in LSPCM as *product areas*. For each one of these product areas LSPCM assesses three *certification criteria (CC)*: availability (CC1) of all required documents/components, their uniformity (CC2) and conformance (CC3) of these documents/components with respect to the chosen certification property. Certification properties considered by LSPCM pertain, e.g., to performance and maintainability. The conformance certification criterion is the most important of all.

Furthermore, LSPCM distinguishes between different *achievement levels (AL)* that can be used to ensure the certification criteria, like uniformity. Each certification criterion has 3 levels. Availability distinguishes between required elements (CC1.AL1), optional semi-formal (CC1.AL2) and optional formal elements (CC1.AL3). For example, the availability of required elements (CC1.AL1) in the product area 'user requirements' includes 3 elements: functional requirements, non-functional requirements, and a glossary.

For the second criterion, uniformity, we consider uniformity at the following levels; within the artifact type itself (CC2.AL1), with respect to a company standard (CC2.AL2), or with respect to an industry standard (CC3.AL3). Conformance, as the third and most important criterion, can either be established manually (CC3.AL1), with tool support (CC3.AL2), or by formal verification (CC3.AL3). We say that the artifact achieves a certain level (AL1-3) if all relevant specific criteria for that AL have been verified.

Summarizing, for each product area and for each one of certification criteria and each one of the achievement levels (AL1-3), LSPCM states specific criteria.

The score of the achievement levels in the assessment is effectuated in the final *certification level*. The certification level can be between 1 and 5. Level 1 represents the "dummy" initial level, which signals the start of the certification analysis, but does not allow for a real certificate. Level 5 represents a certificate for a mature system that is purely based on formal methods. For details we refer to [5].

B. LSPCM analysis process

The analysis according to LSPCM is an iterative process involving the following seven steps:

1) The requesting party determines certification level and type (conformance properties).
2) We verify whether the delivered artifacts meet the entry criteria (CC1 and CC2). If the verification fails, the process is restarted at Step 1.
3) When the artifacts fulfill the entry criteria, they can be transformed into a representation that is easier to analyze. E.g. natural language requirements can be exported to plain text and checked automatically for keywords.
4) The transformed artifacts are assessed on the conformance properties.
5) The results of the assessment are interpreted and reported. Errors and omissions can be immediately pointed out, other aspects can give insights for further assessments (Steps 3 and 4).
6) The analysis report pinpoints the defects that either must be solved (Back to Step 3) or explained by sufficiently sound reasons.
7) If the software artifacts meet the pre-selected criteria, a LaQuSo software product certificate is handed over to the party applying for certification. This certificate is public and shows both the followed approach as well as resulting report.

C. Requirements certification in LSPCM

As indicated in the Introduction we concentrate on user requirements. Hence we focus on specific criteria for the user requirements product area, described below, discussing criteria for availability and uniformity, and dedicated to conformance. Content and numbering of the specific criteria follows the original technical document [6] as we used it for the assessments. For reasons of conciseness the description of some of the specific criteria are shortened or left out.

D. Availability (CC1)

- *AL1* Required Elements:
  - *SC1.1b* Non-functional requirements i.e., quality requirements are described.
  - *SC1.1c* Glossary defines the entities that have to be represented in the system.
- *AL2* Semi-formal Elements:
  - *SC1.2a* Data dictionary or object model that contains data elements' definitions and representations including semantics for data elements. The semantic components focus on creating precise meaning of data elements.
  - *SC1.2c* Flowcharts of processes are schematic representations of a process, including all possible paths between start- and end point, inputs, outputs, as well as the decisions between paths.
- *AL3* Formal Elements:
  - *SC1.3c* Behavioral properties specifications are expressed in a formal language. If they e.g. have a temporal aspect, temporal logic can be used.

E. Uniformity (CC2)

- *AL1* Compliance within project:
  - *SC2.1a* Elements and documents of the same type have the same style;
- *AL2* Compliance with Company Standards:

  - *SC2.2a* All elements and documents comply with company standards;
- *AL3* Compliance with Industry Standards (like ERD,UML).

F. Conformance (CC3)

For conformance we distinguish between internal consistency and external consistency. By internal consistency we understand consistency properties that can be established by considering functional requirements alone. External consistency involves additional information, e.g., non-functional requirements or additional behavioral properties.

We start by presenting the checks covering ***internal*** correctness aspects.

- *AL1* Manual Correctness Checks
  - *SC3.1a* No two requirements or use cases contradict each other.
  - *SC3.1b* No requirement is ambiguous. All stakeholders should be able to interpret the meaning of the requirement in the same way.
  - *SC3.1j* Use case diagrams correspond to use case text.
  - *SC3.1l* The use-cases or functional requirements detail the environment description in the context description (no contradictions). Each step in a business process that involves the system has been included in the requirements. Each task that the system should fulfill for its environment has been included in the requirements. All actors of the context description have been included in the requirements.
- *AL2* Automated Correctness Checks
  - *SC3.2a* Requirements are stored in a requirements management tool which uniquely identifies them.
- *AL3* Formally Verified Correctness
  - *SC3.3c* Check data model diagram for normal form.

For the ***external*** consistency, LSPCM lists the following specific criteria.

- *AL1* Manual Consistency Checks:
  - *SC3.1q* The use case or functional requirements do not conflict with the non-functional requirements.
- *AL2* Automated Consistency Checks:
  - *SC3.2b* Requirements and glossary/objects are stored in a requirement management tool showing the requirements, scenarios, actors, and objects relations.
- *AL3* Formally Verified Consistency:
  - *SC3.3d/e* Verify use case scenario models for e.g. compliance with behavioral properties and non-functional requirements.

G. Hypotheses for offshoring

Although LSPCM is not designed for the offshoring context, we expected that it would have extra benefits in the demanding offshoring context; that the offshored requirements must be explained in more detail and more accurate than non-outsourced requirements to bridge the gap caused by geographical distances and cultural differences between onshore and offshore team. The artifacts produced in offshored development are not that different from non-outsourced development. Therefore, we also assumed that the specific criteria above were sufficiently sensitive to detect all relevant defects.

III. Case studies

While LSPCM has been successfully applied in the past to assess quality of user requirements [4], it had yet to be applied to offshored development. To assess the applicability of LSPCM to offshored development we have used it in a number of comparable case studies; industrial offshored software development projects which are discussed in Sections III-A through III-C. Involved development teams were comparable, the requirements effort ratio was similar, and also the software itself had an average complexity for all 3 cases.

We also want to demonstrate that LSPCM can be applied by non-experienced IT assessors. To this end a group of 6 students, including the second and third author, were invited to enrol in the research. They were asked to do a tutorial assessment before independently analyze the industrial projects that were acquired for this research. Before applying LSPCM to the projects these students did not have any experience in quality assessment or LSPCM in particular.

For each project we briefly describe the project context, documents analyzed and results of the LSPCM application. For all 3 cases we used the criteria as stated in Section II.

A. Case study 1: Lead assignment system

The first project concerned a lead assignment system developed for a financial institute. The institute has decided to integrate the functionality of two separate software systems used for lead assignment in a new system. The assignment of advisors are assigned to leads (interested customers) should be effective and fair to the advisors in the number of leads assigned. Finally, the system should calculate the advisor's income based on the serviced customer requests.

The software development followed the rational unified process development model [8]: the inception and part of the elaboration phase took place on-site in the Netherlands, while the rest was done offshore in India. The project was almost closed at the moment of the assessment; its successfulness was not known to the assessor, in this case, the second author.

For the sake of an assessment we have obtained lists of functional and non-functional requirements, collection of business rules, description of five major use-cases as well as the software development plan. Application of LSPCM to these documents conducted by the second author and a fellow student produced an extensive defects list with defects ranging from inconsistencies resulting from mixing Dutch and English terms to ambiguities, omissions and contradictions. It took the two self educated assessors about 20 hours each to analyse the requirements of the 23.000 person hour project.

The analysis resulted in the following contradictions:

- whether zip code and address are mandatory for direct booking *[SC3.1a]*;
- whether appointments can be created solely by the advisors, or can this be done by other roles as well *[SC3.1b]*;
- whether the user interface should be composed of *three (or more) screens* or of *three interacting panes [SC3.1b]*;
- mandatory functionalities in the use case diagrams that get no attention in the detailed use case descriptions *[SC3.1j]*.

More severe were the detected omissions. Those included:

- total absence of the performance, reliability and maintainability requirements *[SC1.1b]*;
- references within requirements to non-existing business rules in the context description *[SC3.1l]*;
- lack of an appropriate glossary and a Dutch/English list of terms for user interface *[SC1.1c]*.

After the assessment, the defects found have been confirmed by the project leader. Furthermore, the project leader also stated that numerous clarification questions have been posed by the offshore developers during the implementation phase. Abundance of clarification questions can be traced back to ambiguities, omissions and contradictions as detected by applying the LSPCM. The project leader has confirmed that deadline delays and customer dissatisfaction that ultimately lead to the project being terminated could have been avoided by timely addressing the defects identified by LSPCM. After renegotiations with the customer, the project was restarted again. The same defects that were found in the LSPCM assessment were repaired and the customer was satisfied.

In this case two assessors have analyzed the complete project's documents on all relevant properties except for readability. The readability property was only checked on a small random sample of 20 pages. This reduced the amount of work considerably while preserving valid conclusions.

If LSPCM had been used as a quality gate in this project, it would have been most effective after finishing and shipping each use case. The total amount of effort for all 5 use cases, would have been taken 200 person hours; an investment of less than one percent, enabling a fairly early detection of all defects above, and hence with a much happier customer.

B. Case study 2: Unification of different ERP systems

Similarly to the previous case, our second case study involved replacement of multiple systems by a unified one. Also here, the successfulness of the project was unknown to the assessors. The customer was a large international company having organizational units all around the globe. Each one of the units has its own business processes supported by its own ERP system. The unification project aimed improvement and harmonization of the business processes of different units by providing a globally supported unified ERP.

The second author, together with a fellow student, applied LSPCM to all documents specifying functional requirements for the system. Functional requirements were presented as textual descriptions augmented with flowcharts.

Application of LSPCM required approximately the same amount of time (24 hours) per person for a much bigger project (67.000 person hour). Even with 10 assessments the total investment effort would be less than one percent. The applied assessment revealed only ambiguities:

- numerous inconsistencies between the textual and graphical representations of the same information. For instance, when a textual description required three inputs, the flowchart showed only two inputs *[SC3.1j]*;
- the lack of common notation and semantics for different flowchart elements: optional inputs were marked as "optional" in some cases and completely absent from the flowcharts in some other cases. *[SC2.1a]*.

The assessment did not reveal any omissions or large defects. The documents were compliant with company standards. The project quality analyst confirmed later that the project met its deadlines and was successful for both customer and supplier.

C. Case study 3: Replacing a budget distribution system

The third project was analyzed by the third author together with three other students. The project was a 4.900 person hour project aimed at replacing an existing governmental budget distribution system. It consisted of three mostly independent parts. The user requirements were formulated in a set of detailed use cases. They were translated from Dutch to English and sent to the service provider in India. The service provider implemented and tested the system in two separate teams, based on the translated use cases. Finally, the system was successfully acceptance tested by the customer.

Beforehand, the project was described as very successful by the both customer and supplier (Netherlands/India). In this case our goal was to see whether there were superfluous specific criteria that could be identified within LSPCM; criteria that did not help predicting the successfulness of the project, but were rather too sensitive for warning for possible failure factors. Our first goal was to detect the defects and to find out why they did not affect the project's successfulness.

The first assessment on the requirements' original first part took 20 hours for one assessor. The investment effort for an LSPCM quality gate would require three assessments of 20 hours. The total amount would be, again, around the 1 percent. It resulted in the following defects (highest severity first):

- missing documents, e.g., an extended glossary *[SC1.1c]*;
- ambiguous use case descriptions *[SC3.1b]*;
- incomplete and ill-maintained traceability matrix *[SC3.1l]*;
- missing high level documents, such as object model *[SC1.2a]* and process model *[SC1.2c]*);
- the company standard was not adhered to *[SC2.2a]*.

The problems which would normally result from such defects were the risks of low understandability (ambiguity) and low maintainability. Our hypotheses on why these defects did not result in problems were tested and confirmed by means of interviewing all stakeholders in the project.

First of all, the target system had a straightforward architecture. The necessity for high level models and documents

such as an extended glossary was, hence, low. This was also compensated by a large communication investment; offshore and onshore team met each other face-to-face several times and spoke each other almost daily through video conferencing or a chat channel. Ambiguities could be resolved instantly.

Second, the use cases included many details, which made it hard to understand for the customer. However, it did result in a very efficient document for the on- and offshore supplier.

The maintainability risk was not resolved, but created an almost certain vendor lock-in and advantage for the supplier. The consultancy firm chose a very basic maintainability standard despite the fact that the customer had requested to be able to change supplier; the firm has been contracted to do the system's maintenance as well.

The most important benefit of using LSPCM on this project was to make explicit which risks the customer and the consultancy firm were taking during the development of the system. If the missing documents such as the object model and process model had been created and if the ambiguities in the use cases were resolved, these risks would have been reduced and the system would have been easier to maintain. The customer was not aware of the benefits of these documents, and had to rely on the benevolence of the consultancy firm.

IV. Future LSPCM

Based on the findings in the case studies we have observed that LSPCM can be fine-tuned to improve:

- objectivity and consistency: assessment should be made less dependent on assessor's experience,
- speed and iterations: assessment time should be reduced also in iterative approaches,
- context sensitivity: specific criteria should be made domain and conformance property sensitive.

1) Objectivity and consistency: While LSPCM already represents an important step towards an objective assessment by standardizing which specific criteria should be met, assessment of whether these criteria have been met, still depends partly on the specific knowledge and experience of the assessor. To address this we propose to extend LSPCM, whenever possible, by specifying metrics reflecting the extent of the specific criteria being met. Furthermore, LSPCM should consider non-functional requirements as a mandatory part of the requirements document and incorporate the appropriate specific criteria. Finally, specific criteria should be consistently applied across different representations of requirements. For instance, specific criteria pertaining to consistency of use cases should also be included for process models.

2) Speed and iterations: LSPCM-based certification necessitates minutieus analysis of the requirements document and as such can be a time-consuming process. The use of metrics suggested in the preceding paragraph should provide for better automation of the assessment of the individual specific criteria. A number of existing tools could help this process already [9]. Moreover, prioritization of the certification properties considered should allow the assessors to gradually refine the assessment, which is especially the case in an incremental or agile software development process. Finally, as indicated in Section II-B a certification process is iterative in its nature: based on the defects found, requirements engineers can improve the requirements document and submit the revised version for another certification attempt. Recognition of the iterative nature of certification calls for an iterative version of the assessments: e.g., one should be able to restrict the assessment solely to the requirements that have been modified.

3) Context sensitivity: The current version of LSPCM does not distinguish between different kinds of software, while both the choice of the specific criteria and their relative importance (suggested in Section IV-1) might depend on the specifics of the application domain, results of the gap analysis or the way the system should be implemented. In relation with the domains LSPCM recognizes a number of conformance properties. However, customers interested in certifying their requirements documents may need conformance with respect to additional certification properties, e.g., imposed by international standards such as [10]. Therefore, we believe that the subsequent releases of LSPCM should provide the assessor with means of dynamically selecting the specific criteria corresponding to a specific conformance property.

V. Related work

Creating a quality system is a major scientific and engineering challenge. The increasing complexity of hardware, software, and communication does not allow for an easy definition of the quality in software systems. For this reason we think it is likely that only a small number of these quality systems has been published. The majority of these models measures the quality at the end product [11], [12], [13] and only few can measure the influence of the artifacts either in relationship with each other [14] or independently, but only on a specific property like security [15]. Our certification approach can be used to assess quality of system-related artifacts both independently and in relation with other artifacts.

Again, our certification approach focuses on software artifacts rather than on the software development process that produces these artifacts. Assessment of the software development process is intensively studied in the literature including, e.g., [8], [16], [7]. Assessment of software artifacts by third-party assessors is known as *independent verification and validation* or IV&V [3]. Moreover, as mentioned in Section II, LSPCM supports the verification activities of ISO 25000 in general and IV&V specifically. Unfortunately, existing IV&V approaches often either present high-level tasks to be carried out: e.g., "perform criticality and risk assessment of requirements" [3]; or focus on one specific verification technology, such as model checking using SPIN [17]. LSPCM fills the gap created by this dichotomy and proposes a middle path solution. LSPCM checks focus on technical aspects of software artifacts, but leaves the assessor freedom to determine the appropriate way of implementing them. Therefore, specific verification technologies can be integrated in LSPCM to support the assessment, while the results of an LSPCM-based assessment can be used to support the high-level verification.

Hence, LSPCM's structure is more balanced and offers a more effective solution for independent verification (Section VI).

In addition to general quality models several approaches specifically target user requirements. Popular approaches to requirements quality assessment include checklists and questionnaires originating from requirements engineering books [18], [19], [20], or guidelines and standards [7], [21], [22], [23], [24]. Thus, the theory of writing high quality requirements is well-established. In practice, however, the quality of requirement specifications is poor, the requirements are ambiguous, incomplete, unverifiable, and mutually inconsistent [25], [19], [26]. Therefore, more systematic approaches should be sought. An important line of research that can be seen as complementary to our work bases requirements analysis techniques on natural language analysis techniques [9], [27], [28]. These techniques, however, do not address complexity and intricate interplay between requirements expressed using different techniques, e.g., natural language, use cases and process models. Nor do they include the relationship between user requirements and high level design or tests.

VI. Conclusion

Miscommunication about the requirements is a well-known challenge in offshoring [2]. To address this challenge we advocate in this paper the use of requirements quality certification as a prerequisite for requirements hand-over. For the certification assessment we have applied the flexible and generic LaQuSo Software Product Certification model (LSPCM). Certified requirements should provide both the offshoring company and the supplier with confidence in quality of the requirements transferred.

We have shown the benefits of application of requirements certification in three cases of offshored development. The relatively low investment (around 1 % of the total project time) of LSPCM showed an efficient way to successfully identify relevant defects in one of the projects. Later the project has been confirmed as not delivering the desired functionality within the given time and budgetary constraints. Applying LSPCM to other projects showed less significant defects. The in-depth interviews revealed how project managers addressed these defects during project run. Still, even in the milder interpretations of the latest Chaos reports [29] applying LSPCM as a quality gate looks like a solid return on investment.

As LSPCM was not designed with offshoring in mind, application of LSPCM to the offshoring projects showed to be sufficiently sensitive and very suitable. It also allowed us to suggest a number improvements. These improvements will be integrated in the subsequent releases of LSPCM.

While LSPCM as presented in [4], [5] covers all types of software artifacts, so far for offshoring only requirements have been considered. As the *future work* we plan to apply LSPCM to additional software artifacts used at hand-over moments in the offshoring projects, e.g., high-level design and code.

VII. Acknowledgments

We thank the Platform Outsourcing Netherlands (PON) for the financial support. We would also like to thank the companies and people involved who entrusted us to scrutinize their personal work. Finally, our gratitude goes to the assessors: A. Sree Kumar, S. Bhat, S. L. Moorthy, and N. S. Shetty.

References

[1] B. W. Boehm and V. R. Basili, "Software defect reduction top 10 list," *IEEE Computer*, vol. 34, no. 1, pp. 135–137, 2001.

[2] C. L. Iacovou and R. Nakatsu, "A risk profile of offshore-outsourced development projects," *Commun. ACM*, vol. 51, no. 6, pp. 89–94, 2008.

[3] R. O. Lewis, *Independent verification & validation: a life cycle engineering process for quality software*. John Wiley & Sons, Inc., 1992.

[4] P. Heck and P. Parviainen, "Experiences on analysis of requirements quality," in *ICSEA*. IEEE Computer Society, 2008, pp. 367–372.

[5] P. Heck, M. Klabbers, and M. van Eekelen, "A software product certification model," *Softw. Qual. J.*, vol. 18, no. 1, pp. 37–55, 2010.

[6] P. Heck and M. van Eekelen, "The LaQuSo software product certification model: LSPCM," Technische Universiteit Eindhoven, Eindhoven, Netherlands, TUE Computer Science Reports 0803, 2008.

[7] J. Bøegh, "A new standard for quality requirements," *IEEE Software*, vol. 25, pp. 57–63, 2008.

[8] P. Kruchten, *The Rational Unified Process: an introduction*, 3rd ed. Addison-Wesley Longman Publishing Co., Inc., 2003.

[9] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "The linguistic approach to the natural language requirements quality: Benefit of the use of an automatic tool," in *26th Annual NASA Goddard Software Engineering Workshop*. IEEE Computer Society, 2001, pp. 97–105.

[10] IEC, "Functional safety of electrical/electronic/programmable electronic safety-related systems," IEC, IEC/TR 61508-0, 2005.

[11] J. Nastro, "A software product maturity model," *CrossTalk*, vol. 10, no. 8, 1997.

[12] E. Wegner, "Quality of software packages: the forthcoming international standard," *Computer standards & interfaces*, vol. 20, no. 4–5, pp. 349–354, 1999.

[13] A. Alvaro, E. S. de Almeida, and S. L. Meira, "Towards a software component certification framework," in *QSIC '07: International Conference on Quality Software*. IEEE Computer Society, 2007, pp. 298–303.

[14] D. Welzel and H.-L. Hausen, "Practical concurrent software evaluation for certification," *J. Syst. Softw.*, vol. 38, no. 1, pp. 71–83, 1997.

[15] S.-W. Lee, R. A. Gandhi, and S. Wagle, "Towards a requirements-driven workbench for supporting software certification and accreditation," in *Third International Workshop on Software Engineering for Secure Systems*. IEEE Computer Society, 2007, p. 8.

[16] R. Bamford and W. J. Deibler, "ISO 9001;2000:for software and systems providers: an engineering approach," *Reference Engineering Quality Standards*, pp. 0511–1193, 2003.

[17] S. M. Easterbrook and J. R. Callahan, "Formal methods for verification and validation of partial specifications: A case study," *Journal of Systems and Software*, vol. 40, no. 3, pp. 199–210, 1998.

[18] R. R. Young, *The Requirements Engineering Handbook*. Artech House, 2004.

[19] K. E. Wiegers, *Software Requirements*, 2nd ed. Microsoft Press, 2003.

[20] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*. New York, NY, USA: John Wiley & Sons, Inc., 1997.

[21] IEEE-SA Standards Board, "IEEE guide for developing system requirements specifications," IEEE, IEEE Standard 1233, 1998.

[22] ——, "IEEE recommended practice for software requirements specifications," IEEE, IEEE Standard 830, 1998.

[23] ——, "IEEE standard for a software quality metrics methodology," IEEE, IEEE Standard 1061, 1998.

[24] ——, "IEEE guide for information technology system definition concept of operations (ConOps) document," IEEE, IEEE Standard 1362, 1998.

[25] S. Robertson and J. Robertson, *Mastering the Requirements Process*. Addison-Wesley Longman Publishing Co., Inc., 2006.

[26] K. E. Wiegers, *More About Software Requirements: Thorny Issues and Practical Advice*. Microsoft Press, 2005.

[27] V. Gervasi and B. Nuseibeh, "Lightweight validation of natural language requirements," *Softw., Pract. Exper.*, vol. 32, no. 2, pp. 113–133, 2002.

[28] E. Kamstis, "Understanding ambiguity in requirements engineering," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. Springer-Verlag New York, Inc., 2005.

[29] R. L. Glass, "The Standish report: does it really describe a software crisis?" *Commun. ACM*, vol. 49, no. 8, pp. 15–16, 2006.