A Customizable Agile Software Quality Assurance Model

Gu Hongying
College of Computer Science and Technology
Zhejiang University
Hangzhou, China
guhy@cs.zju.edu.cn

Yang Cheng
College of Computer Science and Technology
Zhejiang University
Hangzhou, China
yangcheng@cs.zju.edu.cn

ABSTRACT— **Quality Assurance (QA) is an integral part of the software development process. Many models have been developed to help software development teams to build high quality software. However, most of the models are targeted to big teams with sophisticated managements and plenty of resources. It could be difficult for small and agile teams to implement these models. We are proposing a new Agile Quality Assurance Model (AQAM), which is flexible to incorporate new changes in the software industry and provides detailed guidelines and templates for real world implementation and customization. It is mainly for small and medium sized agile software development teams to analyze and improve their QA capabilities. AQAM has 20 Key Process Areas (KPAs) and each KPA includes guidelines, benefits, processes, best practices, templates, customization, and maturity levels.**

Keywords- Quality Assurance, Process Model, Key Process Area

I. Introduction

It is well recognized that quality assurance and testing are crucial to high quality software and on time delivery [1]. Different approaches and models [2-5] have been proposed to increase repeatability, predictability and manageability. We will briefly summarize several best-known models here.

Capability Maturity Model (CMM) [2] and Capability Maturity Model Integration (CMMI) [3] are initially developed for U.S. Department of Defense to select vendors. The main purpose of CMM and CMMI is to compare capability & maturity among organizations from the outside. Due to their complexity and formality, CMM and CMMI are usually more suited for medium to large sized organizations. Also, only a small part of CMM and CMMI cover software quality assurance. CMMI is mostly about what processes should be implemented, and not about how they should be implemented.

TMM (Testing Maturity Model) [4] was introduced by Ilene Burnstein from the Illinois Institute of Technology in 1996. Based on CMM, TMM is a maturity model with focus on test. It defines five levels of maturity in testing with the same level names as CMM. Being published through three articles, TMM is primarily a theoretical product and not a practical model as it is short of guides and documents. Efforts have been carried on by the TMMI Foundation since 2004.

Based on TMM and CMMI, TMMI [5] (Testing Maturity Model Integration) is a model focusing on testing process improvement and positioned as a complement to CMMI. The maturity level of TMMI is the same as TMM and the structure of KPAs of TMMI is based on CMMI. TMMI uses a staged maturity level system.

Just like CMMI, TMMI mostly deals with what should be implemented and not about how they should be implemented.

TPI (Test Process Improvement) [6] was proposed by a Netherland's software company, Sogeti, during their consulting process. TPI has 20 Key Areas (KA) and each key area has four levels. Each level contains multiple check points. TPI focuses on software testing, not other activities of software quality assurance, such as defect prevention, defect correction, or peer review which are considered very important to software quality assurance. From the usage point of view, the level system will be better if it can be incremental. Right now, a lot of KAs have a high requirement on process control capabilities at the low levels. Also the criteria of the level system can be more specific. It will be easier to implement if KA classification is based on the capability level of activities instead of the scope of the activities.

In the past decade, the software industry has changed greatly on both the research and industrial practices. Agile programming methodology has been widely adopted. Being agile means faster delivery, sometimes with partial functionality, and it also usually means higher efficiency. Agile programming strongly emphasizes team communications. It uses smaller iterations and pays more attention to testing. Among agile programming, Extreme Programming (XP), Scrum and Test Driven Design [13] (TDD) are increasingly popular.

With all these new practices mentioned above, agile software development teams need a flexible and up-to-date quality assurance model which can be customized based on different sizes and management styles. To be user friendly, the model should provide templates or implementation details so that development teams don't rely on professional consulting too much to implement the model. This paper proposes a new quality assurance process model called AQAM (Agile Quality Assurance Model) which is mainly for small and medium sized agile software development teams to analyze and improve their QA capabilities. AQAM has 20 Key Process Areas (KPAs) and each KPA includes guidelines, benefits, processes, best practices, templates, customization and maturity levels. It is customizable based on team sizes and maturity levels.

This work was funded in part by Project (No. 60803110) supported by the National Natural Science Foundation of China.

In this paper, we begin by presenting our approach for AQAM (Section 2). Then, we present a summary on KPA sample to illustrate what a typical KPA looks like in AQAM (Section 3). We then discuss the current stage of this model and outline our future work directions (Section 4). We finally present the conclusions (Section 5).

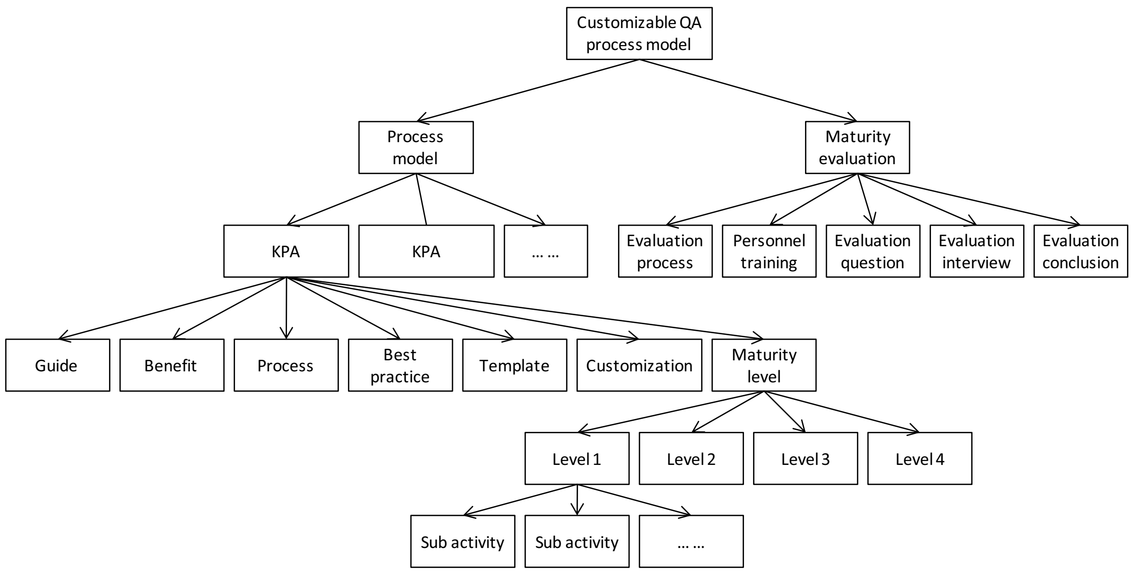

Figure 1: AQAM Architecture

II. AQAM Definition

Software process models often represent a networked sequence of activities, objects, transformations, and events that embody strategies for accomplishing software evolution [9]. AQAM has a user guide and 20 KPAs. Each KPA includes guidelines, benefits, processes, best practices, templates, customization guides, and maturity levels. The main concept of AQAM is to specify how to achieve high quality in software development activities, not just what needs to be achieved, as with CMMI.

In AQAM, KPAs are defined to examine the different maturity levels in different areas within an organization. The organization may be strong in several areas while not as strong in other areas. The purpose of identifying KPAs is to objectively evaluate the strengths and weaknesses of the important areas of the quality assurance process and then the organization can develop a plan to improve the whole process. AQAM identifies 20 KPAs, including *Requirement Management*, *Process Audit, Test Planning, Test Case Management, Peer Review, Defect Analysis, Defect Reporting, Unit Testing, Performance Testing, Configuration Management, Management Support, Training, Test Environment Management, Test Organization, Test Automation, Continuous Integration, User Experience Management, Testing Level, Defect Prevention and Static Analysis.* There are dependencies among some KPAs. Each KPA comes with:

Guidelines: Describing the KPA and providing general guidelines for the KPA;

Benefits: Describing why the KPA is needed and what are the benefits of the KPA;

Processes: Describing the processes of the KPA;

Best practices: Providing the widely accepted and proven practices in the industry;

Templates: The templates used when implementing the KPA;

Customization: Providing customization guidelines, pointing out which sections can be customized and how;

Maturity levels: Describing the activities in each level. Figure 1 illustrates the architecture of our work. Also, since not all KPAs and levels are equally important for the performance of the complete quality assurance process, not all KPAs are treated equally in AQAM. We encourage different teams to choose certain KPAs according to their specific needs and their current maturity levels. The teams can then focus on the chosen KPAs. This practice is very different from CMM or TMM. Figure 2 describes the process.

From the descriptions above, we can see the following differences between AQAM and other models:

A. Customizability

Customizability is the most creative point of AQAM. Teams are different in terms of sizes, goals, current capabilities, team models and ALM methodologies. The existing testing and QA process models usually do not give lots of details about customization, which makes the adoption of the models difficult. Our model is highly customizable and adaptive to various teams. Under each KPA, there is a Customization section, which gives guidelines about which parts are mandatory, which parts are recommended and which parts are optional and how.

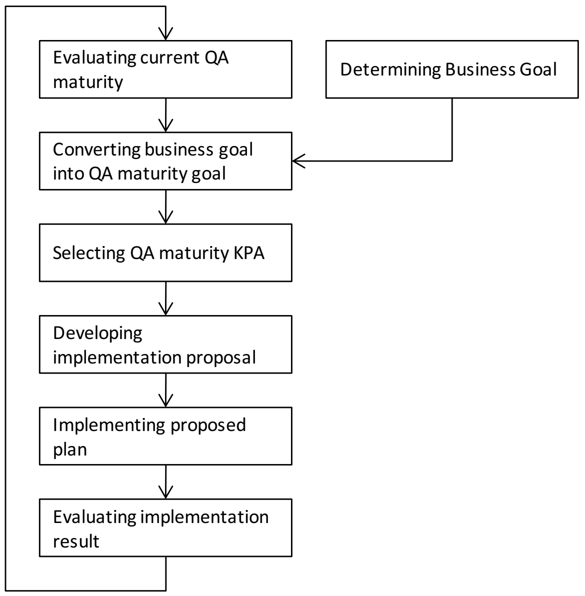
Evaluating current QA maturity
Determining Business Goal
Converting business goal into QA maturity goal
Selecting QA maturity KPA
Developing implementation proposal
Implementing proposed plan
Evaluating implementation result

Figure 2: AQAM Process

B. More details about implementation.

In order to be compatible with all the differences with different software development teams, the existing models usually avoid covering too much implementation details to make the model applicable to more teams. For example, many models mention that whether the teams collect and analyze a Defect Metric, but do not provide details about what the defect metrics are, how to calculate or analyze them [10], or what the best practices are. In many cases, teams have to depend on their own understanding (which may take a lot of time and be inaccurate) or professional consulting firms (which may incur big expenses) to implement a process model. In AQAM, we have *guidelines, benefits, processes, best practices, templates, customization, and maturity levels* under each KPA. Combined with customization guide, the implementation details make the adoption of the model much easier.

C. Up-to-date knowledge incorporation.

We mentioned some changes happened in the last decade on software development in section 1 of this paper. Other than those, continuous integration (CI) [11], testing virtualization (TV), daily build (DB) and other testing and QA methods and tools have gained exposure and have been accepted by various teams. We kept a close eye on industry trends and incorporated the new advancements into the customizable AQAM. These factors are considered and incorporated in KPA definitions and implementations within our model.

III. KPA IMPLEMENTATIONS

As we stated above, in AQAM, each KPA includes *guidelines, benefits, processes, best practices, templates, customization and maturity levels*. Due to the limited space, we are not able to describe the details of the implementation. We will use one of the KPAs to state the elements very briefly.

For example, peer review KPA has 7 guidelines, 7 best practices and 3 table templates including **Review Memo, Review Session Log, and Common Categorized Defect Table**. We also have detailed customization rules including **Required Content, Recommended Content and Customizable Content** along with the templates. Peer Review KPA has 4 different Maturity levels which are **Initial, Performed, Managed and Optimized.**

A. Guidelines

The guidelines include answers to the following questions:

1. How to choose between group review and single person review
2. How to choose between informal review and formal review
3. How to do review planning
4. Who are the participants of a review meeting
5. How to calculate review efficiency
6. How to calculate defect detection efficiency
7. How to analyze defect detection efficiency

To motivate the process practice, we list the benefits for each KPA in the model. Below are the benefits for peer review KPA.

B. Benefits

Peer review can provide team members and managers at various levels with visibility into the quality of the work products by identifying the defects and providing feedback to the team members and managers on the result of the review. This is extremely important. Without appropriate visibility, it is difficult for team members and managers to know the status of the project and thus no proper correction action can be taken.

By providing visibility into the quality and status of projects, peer review can help team members and managers to take actions to deliver projects on time and with high quality.

Peer review can better use knowledge in an organization, even if someone is not a team member. Sometimes, especially for small teams, there is no senior member in the team. During the review process, however, teams can invite senior members, as the reviewers, to give professional opinions. This way, the experience and knowledge of the senior members can be better utilized.

Peer review can discover not only the defects in the code, but also any intermediate results. Peer review can be involved in the earlier phases of the software development process than testing. As shown in Figure 3, the software development v-model, unit test can only be performed when the code is ready, after which the integrated, acceptance and release tests can be run. However, review can be performed at the very beginning of the development process.

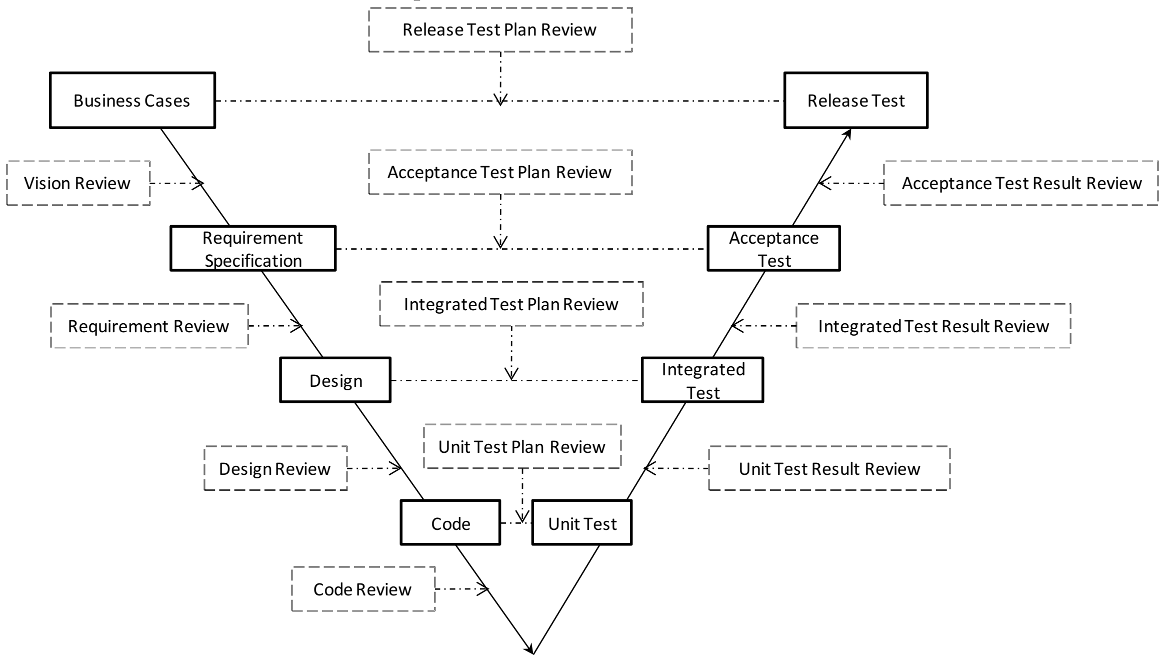

Figure 3. Software Development V-Model

Peer review can make the whole team aware of the defects so the defects can be prevented. In a review session, defects will be discussed and the causes and solutions of the defects will be determined, so that participants of the review session can learn from the defects and avoid them in the future.

Peer review is a very good training tool. When a team member's work is reviewed, specific feedback and resolutions will be given so that the team member can learn from the knowledge and experience of the review team.

Peer review, specifically code review, can find bugs that are very difficult to find through testing. Ideally the testing code coverage is 100% and all the test paths are covered. Of course, this may not be possible as test complexity could grow exponentially when the code coverage increases. Also, some scenarios are very difficult to emulate or reproduce in real tests. For example, an appropriate error message should be shown when a disk fails during a file operation. It is very hard for a tester to find a defect like this, but if code review is properly performed, it is very likely that defect can be found by experienced reviewers. Peer review can make the whole team aware of the defects so the defects can be prevented.

Figure 3 shows the possible Review activities we can carry along with the whole software development procedure. V-model is a linear development method and software was designed in 1986 by Paul E. Brook [12].

C. Processes

The roles in peer review are: moderator, reviewer, author, reviser and observer. The moderator organizes the review, collects the review materials, distributes review materials to reviewers, conducts the review session and follows up. The reviewer and moderators usually need to be trained.

One person can take multiple roles in the review process. For a single person review, the author usually is also the moderator and reviser. The reviewer is someone else and there may be no observer.

For an informal peer review, there is no particular process to follow. The reviewer and the author can choose ways they both are comfortable with.

For a formal review, before a review session, the moderator collects the review materials from the authors and determines if the peer review entry criteria have been met. If the criteria are met, the moderator distributes the materials to the reviewers. The reviewers perform the review and record the findings in a review memo. In the review session, the reviewers present their findings and issues are discussed. The review session usually records all the confirmed defects and the resolutions for the defects, with timelines decided. The moderator together with the team then decides if the review exit criteria are met or a re-review is required.

Every review must be well planned. During the planning process, the materials to be reviewed and reviewer need to be selected and the review schedule needs to be made. Before the actual review session, the reviewers must review the materials beforehand and record all the findings in the review memo. In the actual review, all of the participants should communicate effectively. The defects confirmed in the review session must be

TABLE II. REVIEW SESSION LOG

Project name:				**Date:**		**Moderator:**		
Attendees of the meeting:								
Hours:				**The amount of work reviewed:**		**Next step:**		
#	**Location**	**Description**	**Severity**	**Injection stage**	**If solved in the meeting**	**Resolution**	**Assignee**	**Estimated date of resolution**

corrected. If too many defects are found in the review process, a re-review is suggested. Figure 4 shows a typical peer review process.

Without realizing the importance of peer review, teams are tempted to skip the review process to speed up the development process. A simple method to prevent this from happening is to have the project management team demand that a percentage of team time must be put aside for peer review in the planning phase, and enforce the plan in the execution.

D. Best Practices

1. Educating the members of the organization with the benefits of peer review to motivate them;
2. Choosing experienced members as reviewers;
3. Identifying and choosing important work as review materials;
4. When new technologies or processes are adopted in the team or team members are new to the domain field, increase the review frequency;
5. The work of those team members who are not experienced should have higher priority for review;
6. Management should not use peer review as a tool of performance evaluation.
7. Peer review should be included in the overall project plan.

E. Templates

1) Review Memo

Table 1 is used to record issues found in the preparation process and to be presented in the review session.

TABLE I. REVIEW MEMO

Project name:		**Review material:**	**Reviewer:**
Hours:		**The amount of work reviewed:**	**Date:**
Defect #	**Defect location**	**Defect description**	**Severity**

Explanation:

Hours: The number of hours the review process takes. It represents how much time the reviewer spends on this particular review;

The amount of work reviewed: For source code it is how many lines of code; for requirement or other specifications it is how many pages of the specification; for test cases it is how many test cases;

Defect description: The detailed description of a defect.

2) Review Session Log

Table 2 is used to record the defects confirmed in the meeting and status of the unsolved defects.

Explanation:

Hours: The number of hours the meeting took * the number of attendees to the meeting. It represents the man hour spent on this review;

Next step: According to the number and the severity of the defects, the next step can be re-review or follow up;

Injection stage: The point in which the defect is injected. It can be requirement analysis, design, or coding;

If solved in the meeting: If a solution is agreed in the meeting;

Solution: If a solution is agreed in the meeting, enter it here;

Assignee: Who is responsible for the defect;

Estimated date of resolution: When can the defect be solved.

3) Common Categorized Defect Table

Table 3 is mainly used for common defect prevention and reviewer & author training.

Explanation:

Defect title: The brief description of a defect;

Category: The category of a defect. The possible categories are: GUI, Logic, Code Convention, Performance, Portability, Design, Test Case, etc.;

Severity: Can be high, medium, or low;

Description: The detailed description of a defect. It includes the cause of the defect and help the team avoid similar defects in the future.

The table helps to accumulate well documented experience within a team. With this, all the valuable experience and knowledge attained during this project can be transferred to the new members of this project, and other projects of the same organization. Overall, it helps a company to achieve higher performance not only in this project, but also in the upcoming projects.

F. Customization

1) Required content

- The overall project plan, requirement specification, overall design and acceptance testing plan must go through group review process.
- Peer review must be included in the project plan or testing plan.

2) Recommended content

- Review Session Log is recommended. This table documents all the defects and makes the resolution of defects more efficient.
- Common Categorized Defect Table is recommended. This table is very useful for defect prevention and team member training;

TABLE III. COMMON CATEGORIZED DEFECT TABLE

Defect title:	
Category:	**Severity:**

Description:

3) Customizable content

- Review Memo is optional, if the reviewers prefer other ways to record the preparation notes;
- The Hours field in Review Memo and the Review Session Log is optional if the team is not doing peer review cost data collecting, analysis or other management activities;
- The Amount of Work Reviewed field in Review Memo and Review Session Log is optional if the team is not doing peer review efficiency data collecting, analysis or other management activities;
- Review Session Log can be optional. But if it is excluded, the team must track the status of the unsolved issues.

G. *Maturity level*

1) Level 1: Initial

Peer review is not performed in the team.

2) Level 2: Performed

- The team recognizes the importance and benefits of peer review;
- Peer review is included in project plan;
- Peer review is conducted for important work products;
- The team understands the differences between group review and single person review. They can choose the optimal review format accordingly;
- The participants of peer review are trained;
- The result of peer review is tracked and every issue has its resolution.

3) Level 3: Managed

- Management supports peer review in terms of staffing, budget and planning;
- Entry criteria are defined;
- Exit criteria are defined;
- Re-review criteria are defined;
- Training topics are identified and training materials are prepared;
- Review processes are defined.

4) Level 4: Optimized

- Peer review work load and efficiency data are collected and analyzed;
- The scope of peer review is clearly defined and documented;
- The procedure of peer review is clearly defined and documented;
- There is training group to train the review participants;
- The review result is well documented and communicated in the team;
- The defects with high frequency are documented in Common Categorized Defects Table.

According to different situations in different organizations, this level system provides a practical method not only for a comparison between different organizations, but also for internal evaluations and performance improvements.

IV. Discussion and Future Works

Our work to date has focused on the process model definition and customization. Not all the KPAs have been developed. Usually it is enough for an organization to improve performance by implementing even just some of the KPAs. As a process, the users need to analyze the existing maturity levels of the selected KPAs in the organization, propose an action plan based on AQAM and implement the proposal.

We have done some user studies which are not included in this paper. AQAM can be optimized with more practical feedbacks. More validations are part of our future works.

Although AQAM is mainly designed for small to medium sized teams, it is possible to use the approach on a large project, in the cases the project is decomposed into smaller pieces.

V. Conclusion

Customizability and evolution of software processes and their models are acquiring a growing importance in the software process community. In this paper, we have presented AQAM as a new software quality assurance process model. It is primarily designed for small and medium sized software development teams. With AQAM, organizations can customize and manage their process model based on their own maturity, current quality assurance process and management style. We also briefly demonstrated how the model works by a peer review KPA sample.

References

[1] Fowler, P., Rifkin S., Software Engineering Institute, *Software Engineering Process Group Guide,* Technical Report, CMU/SEI-90-TR-24, 1990

[2] Caputo, K., *CMM Implementation Guide: Choreographing Software Process Improvement*, Addison-Wesley, 1998

[3] Software Engineering Institute, Capability Maturity Model Integration (CMMI®), CMMI® for Development, Version 1.3, Technical Report, Nov. 2010

[4] Burnstein, I., Suwannasart, T., and Carlson, C.R., A Model to Assess Testing Process Maturity, Crosstalk, Nov. 1998

[5] TMMi Foundation, Test Maturity Model Integration (TMMi) Version 3.1, Retrieved March 1, 2011, TMMi Foundation: http://www.tmmifoundation.org/downloads/tmmi/TMMi%20Framework.pdf, Ireland.2010

[6] Jari Andersin, TPI – a model for Test Process Improvement, http://www.cs.helsinki.fi/u/paakki/Andersin.pdf, retrieved on July 01, 2011.

[7] Deissenboeck, F., Juergens, E., Lochmann, K., Wagner, S., Software quality models: Purposes, usage scenarios and requirements, *WOSQ '09. ICSE Workshop on Software Quality,* 2009

[8] De Souza, C. R. B., Redmiles D. F., An Empirical Study of Software Developers' Management of Dependencies and Changes. In *ICSE'08, Proceedings of the 30th International Conference on Software Engineering*, May 10–18, 2008, Leipzig, Germany. 241-250

[9] Nagappan, N., Maximilien, E., Bhat, T., Williams, L., Realizing quality improvement through test driven development: results and experiences of four industrial teams, *Empirical Software Engineering,* Volume 13, Issue 3, 2008 289-302

[10] Paulk, M. C., Weber, C. V., Curtis, B., Chrissis, M. B., The Capability Maturity Model for Software: Guidelines for Improving the Software Process, Addison-Wesley, 1995

[11] Fowler, M., Continuous Integration, 2006, http://www.martinfowler.com/articles/continuousIntegration.html, retrieved on March 01, 2011.

[12] http://en.wikipedia.org/wiki/V_model , retrieved June 1, 2011