QUARK: A DYNAMIC SDLC METHODOLOGY*

V. Vuppala, J. Vincent, NSCL, East Lansing, MI 48824, USA. #

Abstract

No single Software Development Life-cycle (SDLC) methodology works well for all types of software projects. The project may require a methodology that can be very predictive to very adaptive based on characteristics such as requirements volatility, requirements clarity, project criticality, complexity, and size. We describe a new iterative approach that can vary from being more adaptive to being more predictive during its iterations. The project characteristics change with iterations, and the SDLC adjusts accordingly by changing its parameters. We also discuss the results of using this methodology for projects at National Superconducting Cyclotron Laboratory (NSCL).

INTRODUCTION

Last few decades have seen an evolution of SDLC models to address the software-crisis. Some of these are Waterfall, Spiral, V-Process, RUP, and Agile among others. Each model has its advantages and drawbacks, and not all of them work for all types of software projects [1]. Some of them are predictable in terms of cost and schedule but rigid in terms of requirements, whereas others are adaptive to changes but less predictive.

In our organization there was a need to implement processes to instil engineering rigor into software development. The following were the requirements for the process model:

- Provide transparency and predictability
- Work with limited customer availability
- Not overly bureaucratic, low overhead
- Support project management
- Support critical and non-critical systems

We evaluated various models but found them to be inadequate for our needs. Many organizations, especially in the software industry, choose from a set of SDLC models based on the project characteristics. This was not an option for us, as it required the project team to be proficient in multiple software development methodologies. As a result, we developed a set of processes for software development and project management, which resulted in the Quark Model (QM). It is based on CMMI-Dev 1.2, PMBOK 4, and ISO 9000-3 standards.

Iterations

QM uses an iterative approach to software development. QM iterations are parameterized, and governed by the following parameters (QMPs):

- Duration: The duration, in terms of calendar time, of the iteration
- Change Control: Specification of Major and Minor scope changes
- Documentation: The detail and amount of documentation
- Communication: Meeting intervals and duration within project team, and with Customer
- Planning: Level of detail in planning
- Quality Controls: Frequency of Design and Code reviews, and test methodology.

By adjusting the QMPs, for each iteration, the process can be adjusted from being more adaptive to being more predictive, and anywhere in-between.

Projects

Projects are central to the QM model. A software project is a temporary endeavour undertaken to create a unique software product [2]. It is characterized by certain attributed (PCTs). Some of the PCTs that vary during the execution of a project are:

- Project Team Requirement Clarity: Project team's understanding of the requirements
- Customer Requirement Clarity: Customer's understanding of the requirements
- Size: Size of the project in terms of cost, code base, team size, etc
- Estimate Confidence Level: Accuracy of cost and schedule estimates
- Technology Expertise: Familiarity with the solution technology

Some of the PCTs remain relatively constant during the course of the project, such as criticality of the project, safety and security requirements, quality requirements, timeline constraints, customer Availability, bespoke or custom software, contract type, and team location.

QUARK MODEL

Figure 1 illustrates the Quark Process Model. The PCTs

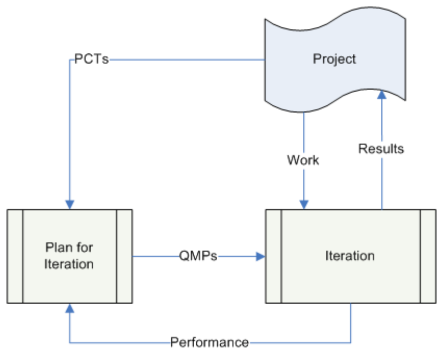

Figure 1: Quark Process Model.

# Email Addresses: {vuppala,vincent}@nscl.msu.edu

* This work is funded by National Science Foundation and Michigan State University

and performance (of previous iteration) are used to generate QMPs. The QMPs drive the next iteration, which may result in the modification of the PCTs. Iterations are useful to garner feedback but incur the overhead of test and release management. Hence the number of iterations should be optimized. The idea is to start with shorter duration iterations, and move to longer iterations as the clarity of requirements improves.

Development Process

Figure 2 depicts QM's software development process. It consists of the following major activities:

- Refine Requirements and Architecture
- Plan for iteration or release (PFI)
- Refine design and test plans
- Code, Refactor, Unit Test (CRUT)
- Release
- Deploy and Test
- Review
- Perform User Acceptance Test (UAT)

At the end of each iteration, modifications to the scope, if any, are evaluated. If the change is minor, the next iteration is initiated. However, if the change is major, a Change Request is generated, and the Perform Change Control (PCC) process is initiated. PCC is a Project Management level process, and can result in iteration through the *Plan* process (see below).

In QM, software product goes through release process even for integration tests. This helps with testing of the installation process. Not all releases are sent to the Customer for UAT, and UAT can be proceed in parallel with the execution of next iteration i.e. the next iteration need not wait for feedback from UAT. Configuration management is performed only for production releases.

Project Management

Project Management (PM) is an integral part of QM. Figure 3 shows the QM project management processes with their inputs and outputs. These processes are based on PMBOK-4 [2] but are different especially the *Initiate* process. Goals of the *Initiate* process are to define the scope, develop the solution strategy, and estimate the cost. The results of these activities are documented in Preliminary Project Plan (PPP). PPP is refined in the subsequent process, resulting in the Project Plan (PP). PP also includes the schedule, budget, and plans for quality, risk, communication, and procurement. The level of detail in PP is dictated by the QPMs. The *Execute* process consists of the following activities:

- Acquire and manage the project team
- Conduct procurements, if any
- Perform quality audits (design and code reviews)
- Develop Software using QM Development Process

The *Monitor and Control* process runs in parallel to other activities. It periodically evaluates project performance, procurement status, risks, and quality. It reports project status to stakeholders. The last step in the PM processes is to close the project. Some of the activities here are:

- Obtain Customer feedback and acceptance
- Close procurement activities, if any
- Summarize Lessons Learned, project performance, and customer feedback in Project Closure Report (PCR)
- Archive project related files, and release the team

Project Performance

QM uses Earned Value Management (EVM) [3] to report project performance. EVM is part of the Project Status Report and is measured periodically, generally every week. A Cost Performance Index (CPI) of less than 1.0 indicates that the effort was underestimated, and the project will be over budget if continued at the same pace. A Schedule Performance Index (SPI = EV/PV) below 1.0 indicates that resources were under-allocated, and the project will be delayed. Similarly a CPI of more than 1.0

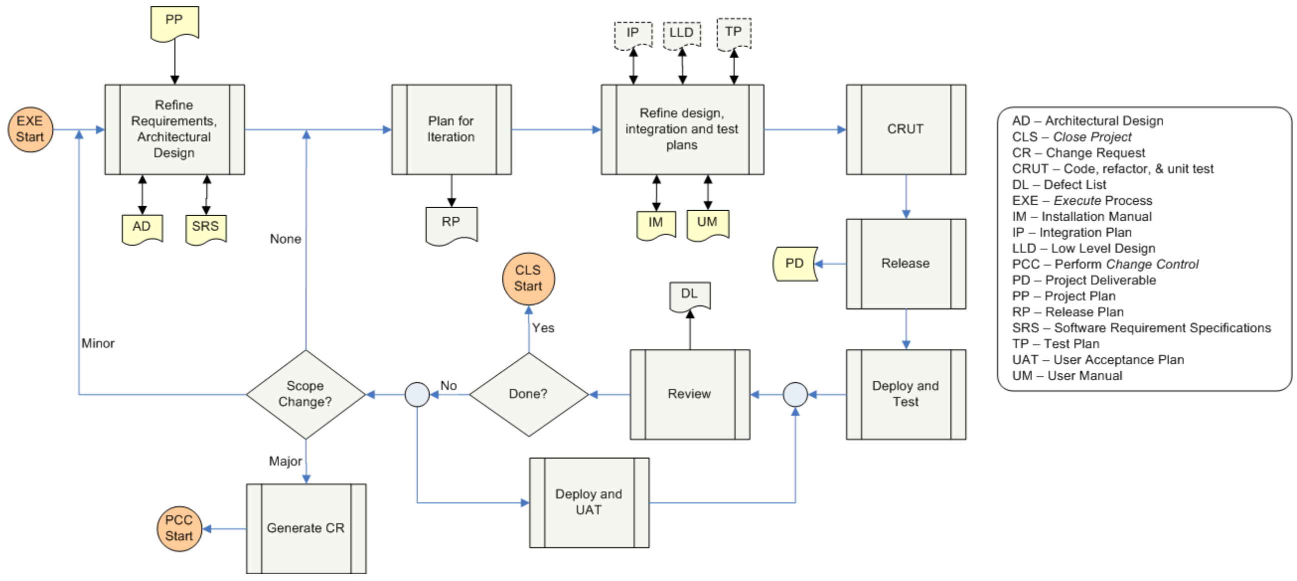

Figure 2: Quark Software Development Processes.

indicates overestimation of effort, and an SPI of more than 1.0 indicates over-allocation of resources. The CPI and SPI values are used to adjust the QPMs for the next iteration.

Documentation

The documents are refined iteratively. QPMs dictate the level of documentation detail. The requirement specifications and the design documents are modified to be in sync with the CRUT activities of the last iteration. This is essential for software maintenance. Some of the required QM documents are Project Plan, Requirement Specifications, Architecture Design, Installation Manual, User Manual, Project Status Report (includes EVM) and Project Closure Report.

IMPLEMENTATION

Based on QM, we have developed the process infrastructure --policies, procedures, guidelines, templates, tools, etc-- for the Electronics Department at NSCL. The process infrastructure is hosted on a website. The project management processes of QM have been generalized, and are being used by non-software groups within the Electronics Department. Currently there are about 5 software development and 15 hardware development projects using the QM processes. All new projects in the department must adhere to the QM processes.

We find that, for software projects, about 8-10% of effort is spent on project management, and a similar amount is spent on documentation. The Customers were very satisfied (9 out of 9) with the ability to make changes, the amount of resources they had to invest, and project management. These results are preliminary; we have not completed enough projects to give a definitive result.

SHORTCOMINGS

QM is not the silver bullet, and has the following drawbacks:

- Currently, measurement of QPMs and the evaluation of PCTs, are subjective. This leaves many decisions to project manager's judgement.
- EVM requires projects to be base-lined, and may not work well for very short iterations.
- It is a slightly heavy-weight model due to the project management processes.

CONCLUSION

Even though QM was developed for our specific needs, it is generic enough to be used by other organizations. Most of the processes, roles, and policies have been designed to be generic; only the guidelines and templates are specific to our environment.

We are currently working on formulating objective measurements of PCTs and QPMs. We are also looking into modifying EVM to suit the Quark Model.

REFERENCES

[1] I. Sommerville, "Software Engineering", 8th Edition, Addison-Wesley, 2007.
[2] ANSI/PMI, "Project Management Book of Knowledge 4th Edition", 2008; http://www.pmi.org.
[3] U.S. Department of Energy, "Earned Value Management". http://www.management.energy.gov/policy_guidance /earned_value_management.htm

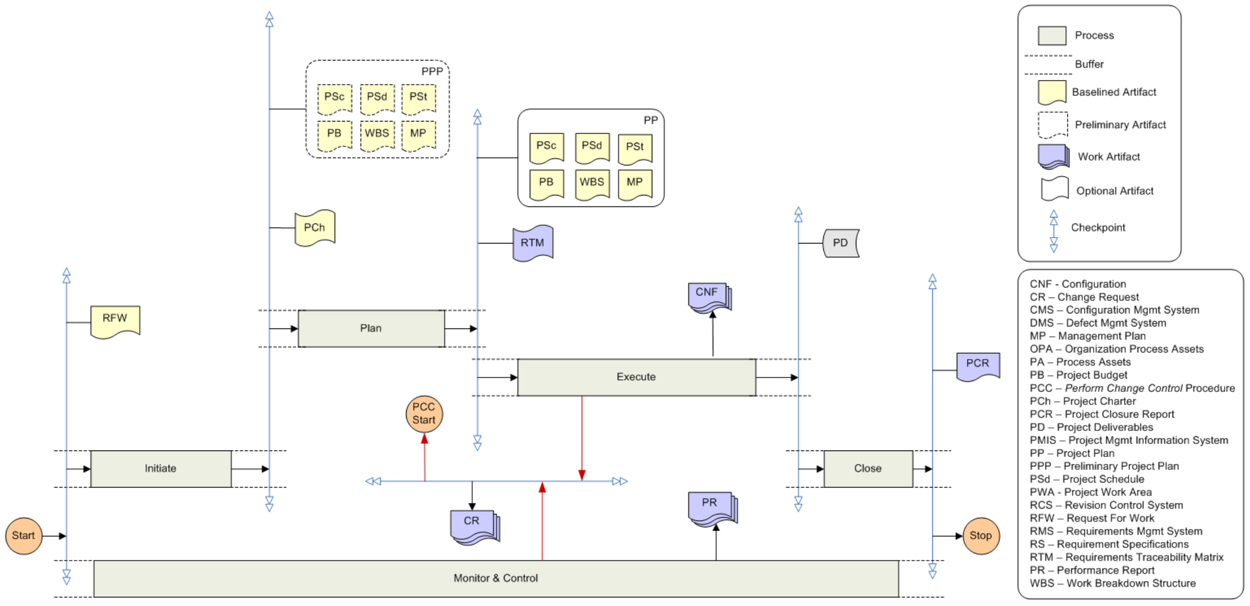

Figure 3: Quark Project Management Processes