International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Volume 3, Issue 3 (May-June 2015), PP. 117-120

DEVELOPMENT OF A METHODOLOGY TO ESTIMATE THE EFFORT, QUALITY AND CYCLE TIME FOR IMPROVING SOFTWARE PROJECT MONITORING

Mohd Faizi, Prof. Dr. Shafeeq Ahmad
Azad Institute of Engineering Technology, Lucknow-226002
faizi406@gmail.com, ahmad_shafeeq@rediffmail.com

Abstract- The main scope of the paper is determining the effectiveness of these techniques for plummeting rate of software defects produce failures. The quality of software will be found based on the quality, cycle time, effort, product size, product complexity and schedule pressure. Developing software to meet functional needs with acceptable levels of quality, within budget, and on schedule, is a goal pursued by every software development organization. Many organizations are adopting the best practices in software development, such as those based on Capability Maturity Model

I. INTRODUCTION

CMM has been one of the most popular efforts in enhancing software quality and reducing development costs. Although development effort, software quality, and cycle time have been studied in prior research on software estimation most of the published results are based on data sets that are now considered outdated, due to various technological innovations such as the use of object-oriented languages, middleware, and newer tools and due to increased adoption of best practices in software development, that is, those based on CMM. There is a need to re-examine relationships between software project development outcomes and various factors identified from prior literature.

The Capability Maturity Model

The Capability Maturity Model for software (CMM) was developed by Software Engineering Institute to describe the principles and practices underlying software process maturity. Its aim is to help organizations improve their software process maturity through an evolutionary path, from ad hoc, chaotic to mature, and disciplined. CMM also helps assess how well defined the software development processes in an organization are. A well- defined process is one that has readiness criteria, clear inputs and outputs, probably some standards, and procedures for performing the work (or separate phases). Moreover, there are also verification mechanisms as well as completion criteria (when it is completely "done") for that process [6]. In CMM-SW model, organizations at level 3 possess defined processed.

The CMM is organized into 5 levels. Level 1 Initial is where the software process is characterized as ad hoc, or even chaotic in some cases. From level 2 to level 5, each level consists of a set of key process areas (KPA) that an organization should focus on to improve its software process. Each key process area in turn comprises a set of key practices that indicate if the implementation and institutionalization of that area is effective, repeatable, or lasting.

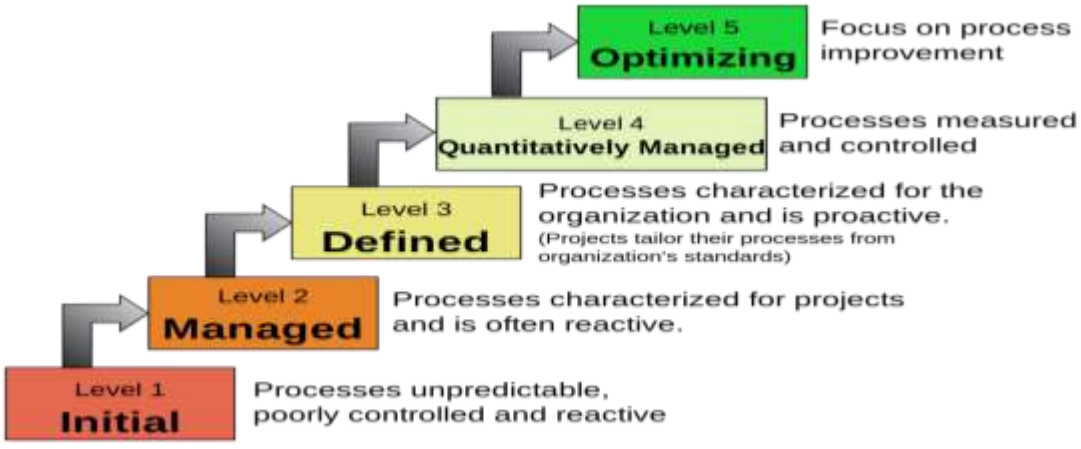

Figure 1

The Software Capability Maturity Model (SW-CMM) provides a set of requirements that organizations can use in setting up the software process used to control software product development. The SW-CMM specifies "what" should be in the software process but not "when" or "for how long." The SW-CMM has what is called a process maturity framework. There are five levels of process maturity, Level 1 (lowest) to Level 5 (highest). To be rated at a specific level an Organization has to demonstrate capabilities in a number of Key Process Areas (KPA) associated with a specific SW-CMM level, Table 3.1. The capabilities demonstrated in transitioning from lower levels

to higher levels are cumulative. In other words, a Level 3 Organization must demonstrate KPA capabilities from Level 2 and from Level 3.

The Process Maturity framework is presented in Table 3.1. All Organizations start at Level 1. This is called the Initial level. At this level few processes are defined, and success depends on individual effort. This makes the software process unpredictable because it changes as work progresses. Schedules, budgets, functionality, and product quality are generally unpredictable.

To achieve Level 2 the organization demonstrates capability in 6 KPA's. A Level 2 Organization has basic management processes established to track cost, schedule, and functionality. Problems in meeting commitments are identified when they arise. Software requirements and work products developed to satisfy requirements are baselined and their integrity is controlled. Software project standards are defined and the organization ensures they are faithfully followed. The project works with its subcontractors to establish a strong relationship. The necessary process discipline is in place to repeat earlier successes on projects with similar applications. Level 2 is called the Repeatable level.

A Level 3 Organization has demonstrated capabilities in an additional 7 KPA's. At this level the software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the whole organization. Projects tailor the standard software process to develop their own unique defined software process. A well-defined process includes readiness criteria, inputs, standards and procedures for performing the work, verification mechanisms, outputs, and completion criteria. Level 3 is called the Defined level.

A Level 4 Organization has added 2 more KPA's to its capabilities. At this level detailed measures of the software process and product quality are collected. Projects achieve control over their products and processes by narrowing the variation in their process performance to fall within acceptable quantitative boundaries. Both the process and product are quantitatively understood and controlled. Level 4 is called the Managed level.

At Level 5 an Organization has capabilities in 3 more KPA's and is in a continuous improvement state. Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies. Software project teams analyze defects to determine their causes. Processes are evaluated to prevent known types of defects from recurring, and lessons learned are disseminated to other projects. Level 5 is called the Optimizing level.

II. PROPOSED METHOD

In this research work we aimed to achieve followings:

- Effort, quality and cycle time estimation
- Identify the key project factor for CMM level 5 projects
- Study of CMM 5 projects

Effort, quality and cycle time estimation

We developed an application program to estimate the Effort, quality and cycle time in C# using .Net environment. The flow of this developed application is as follows:

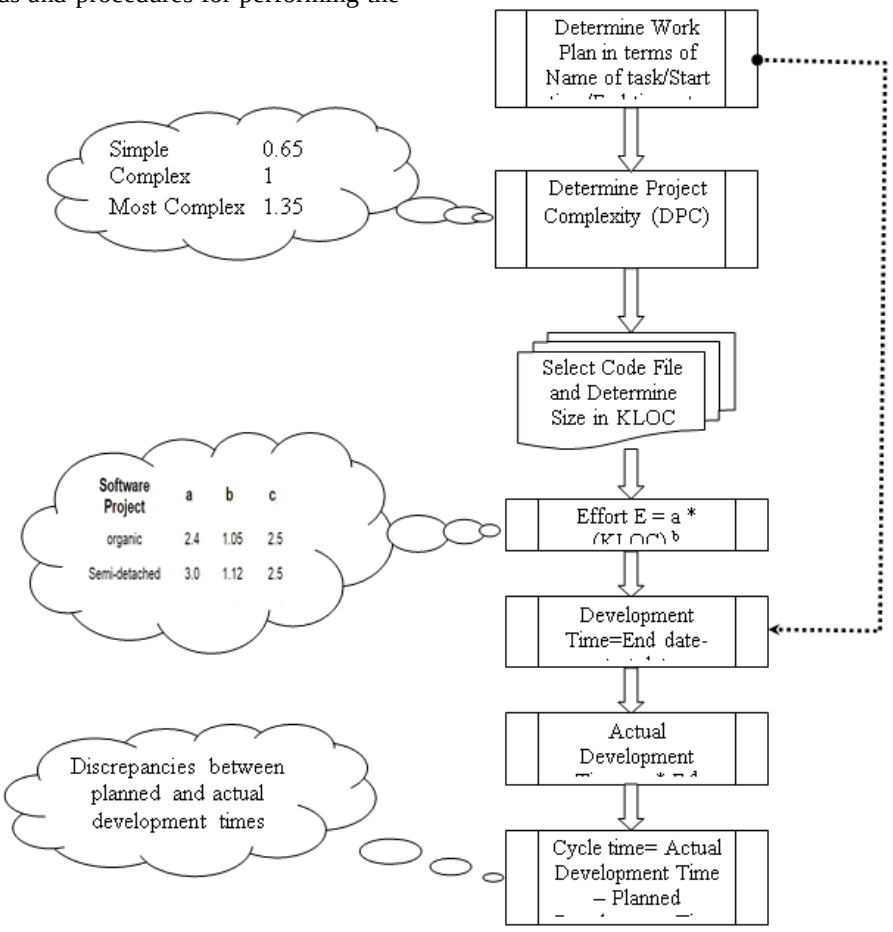

Figure 3.1 : Effort and Cycle time calculation

III. IDENTIFICATION OF KEY PROJECT FACTORS

A. SW-CMM Key Process Areas

Each KPA has a set of goals, capabilities, key practices, measurements and verification practices. The goals and key practices are the most interesting of these because they could be used to assess the impact of a KPA on a project development effort as shown in Figure 2. The goals state the scope, boundaries, and intent of a KPA. A key practice describes "what" should happen in that KPA. There are a total of 52 goals and 149 key practices.

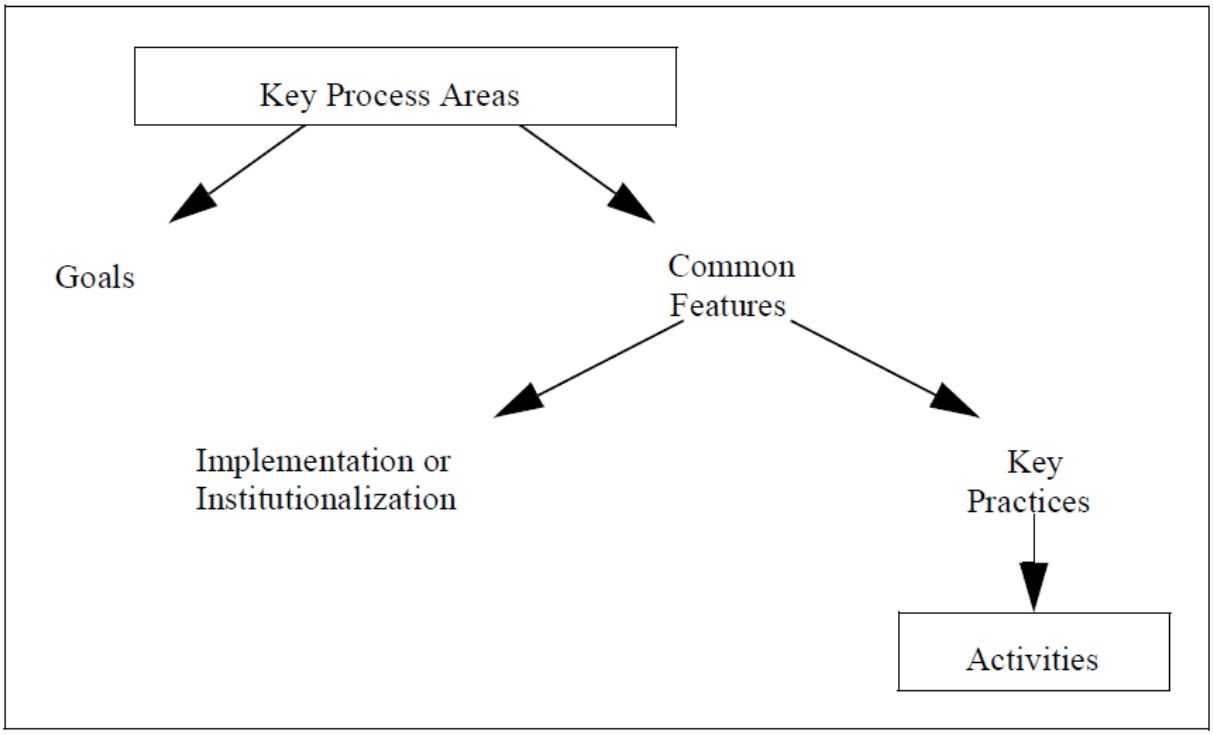

Figure 3. KPA Structure

As an illustration the goals of one KPA from Level 2, Software Project Planning, are given. The purpose of Software Project Planning is to establish reasonable plans for performing the software engineering and for managing the software project. Software Project Planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work.

B. Candidate Predictor Variables

Most analyses identify four areas that influence software development effort. Predictor variables that represent four influential areas are used as inputs into the Research Model. These predictor variables are also in the COCOMO II cost model and they are regrouped into the four areas in Figure .3.

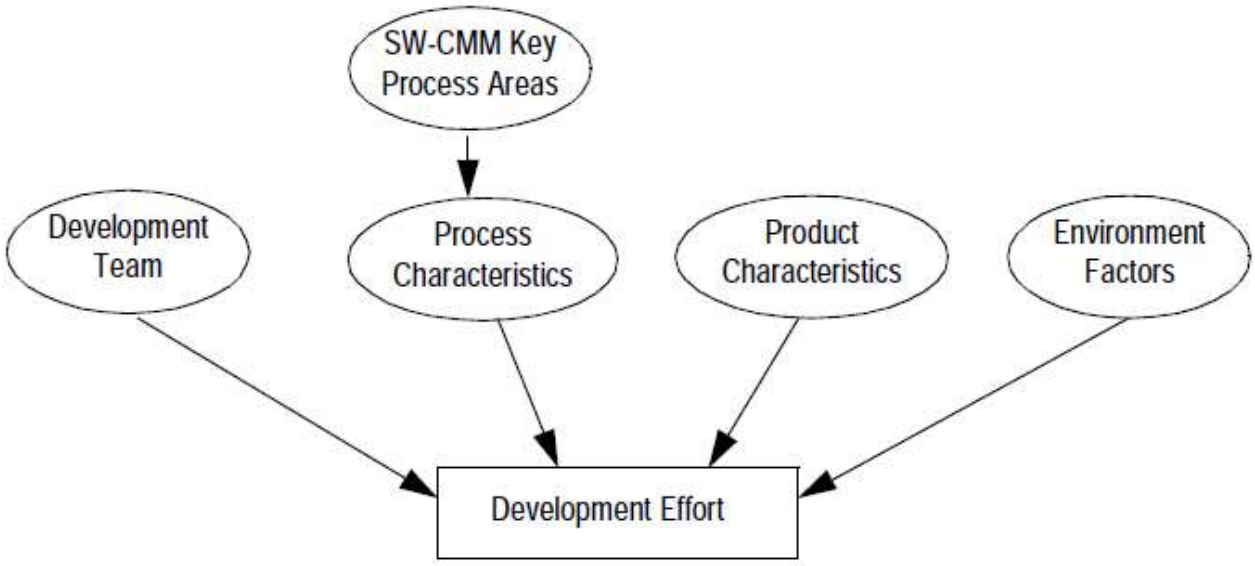

Figure 4. Effort Influencing Areas

The next four subsections are a list of COCOMO II predictor variables that support the four areas shown in Figure 3

C. Product Characteristics

The Product characteristics can have a large impact on effort. Product characteristics include size, amount of required software reuse, required reliability, complexity, storage and time constraints, and the stability of the underlying infrastructure on which the software relies.

D. Development Process

The Development Process directs the activities of the developers, quality assurance personnel, and project management. Activities include SW-CMM-oriented practices such as requirements management, product design, coding, unit testing, integration and test, configuration management, quality assurance, and peer reviews. Although the SW-CMM specifies a progression on KPAs to attain higher maturity levels, organizations may practice some of the KPAs in all of the levels.

E. Environment Factors

The Environment factors that affect effort are technology insertion (such software engineering methods and tools), facilities, and work conditions (such as multi-site development or development schedule compression).

Table 1. Environmental-related Predictor Variables

Predictor Variable	Symbol	Description
Development Flexibility	FLEX	This is the required conformity to development standards and constraints such as rigid schedules or performance requirements. It accounts for the extra effort needed to follow rigid and inflexible software development standards and constraints.
Use of software tools	TOOL	Use of Software Tools rates the use of tools in making the software development more efficient.
Multi-site development	SITE	This accounts for the extra effort needed to coordinate and integrate development activities that are not co-located and do not have access to wideband electronic communication facilities.

Table 2. Environmental-related Predictor Variables

Platform volatility	PVOL	Platform Volatility is a rating of the frequency of change in the complex of hardware and software that the product calls upon to do its work.
Required development schedule	SCED	Required Development Schedule measures the schedule constraint imposed on the project team developing the software, e.g. schedule compression.

IV. CONCLUSION

The main scope of the project is determining the effectiveness of these techniques for plummeting rate of software defects produce failures. The quality of software will be found based on the quality, cycle time, effort, product size, product complexity and schedule pressure. Developing software to meet functional needs with acceptable levels of quality, within budget, and on schedule, is a goal pursued by every software development organization. Many organizations are adopting the best practices in software development, such as those based on Capability Maturity Model.

REFERENCES

[1] Lami G., Falcini F. (2009). Is ISO/IEC 15504 applicable to agile methods? XP 2009. LNBIP 31, pp. 130-135.

[2] McMichael B., Lombardi M. (2007). ISO 9001 and Agile Development, IEEE Agile. [9] Paulk M. (2001). XP from a CMM perspective. IEEE Software, 18 (6), pp. 19-26.

[3] Pikkarainen M., Mantyniemi A. (2006). An approach for using CMMI in Agile software development assessments: experiences from three case studies. SPICE 2006 conference.

[4] Santana C., Gusmao C., Soares L., Pinheiro C., Maciel T., Vasconcelos A., Rouiller A. (2009). Agile software development and CMMI: What we do not know about dancing with elephants. XP 2009, LNBIP 31, pp. 124-129.

[5] Stalhane T., Hanssen G. K. (2008). The application of ISO 9001 to Agile Software Development. PROFES 2008, pp. 371-385.

[6] Sutherland J., Jakobsen C., Johnson K. (2007). Scrum and CMMI Level 5: The Magic Potion for Code Warriors. In: Proceedings of the Agile Development Conference, pp. 466 – 471.

[7] Agrawal M., Chari K. (2007). Software effort, quality, and cycle time: a study of CMM level 5 projects. IEEE Transactions on software engineering, 33(3), pp. 145-156.

[8] Paulk. M.C., Weber C.V., Curtis B., Chrissis M.B. (1995). The Capability Maturity Model: Guidelines for improving the software process. Addison-Wesley Publishing Company (Reading book)

[9] Racheva Z., Daneva M., Sikkel K. (2009). Value creation by agile projects: methodology or mystery? 10th International Conference on Product-Focused Software Process Improvement (PROFES 2009), pp. 141-155.

[10] Kauppinen M., Savolainen J., Lehtola L., Komssi M., Tohonen H., Davis A. (2009). From feature development to customer value creation. IEEE International requirements engineering conference 17th, pp. 275-280.

[11] Barney S., Aurum A., Wohlin C. (2008). A product management challenge: Creating software product value through requirements selection. Journal of Systems Architecture, 54, pp. 576-593.