Formulation of nominal schedule equation in the application of Software Engineering

Rajesh Kumar Sahoo
Assistant Professor, Department of Computer Sc. &Engg.
Ajay Binay Institute of Technology, Cuttack

Sambit Kumar Mishra
Associate professor, Department of Computer Sc.&Engg.
Ajay Binay Institute of Technology, Cuttack

Dayanidhi Mohapatra
Lecturer, Department of MCA
Ajay Binay Institute of Technology , Cuttack

Abstract: Software Engineering is nothing but application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. Software can be defined as computer programs and associated documentation such as requirements, design models and user manuals. It is an Engineering discipline with all aspects of software production. A systematic and organized approach is proposed in our application to formulate nominal schedule equation for actual time period and person month while developing a software project.

Keywords : Nominal Schedule , Generic process , CASE system , Heterogeneity .

1. Introduction

Software Engineering is an Engineering discipline that is concerned with all aspects of software production. Software Engineers usually adopt a systematic and organized approach to their work and use appropriate tools and techniques depending on the problem to be solved. Software products may be developed for a particular customer or may be

developed for a general market. Software products may be Generic e.g. developed to
be used by a range of different customers or Bespoke e.g. developed for a single customer. We may define Software process as a set of activities whose goal is the development or evolution of software.

1.1. Generic activities

The Generic activities in all software processes are defined as follows.

(a) Specification - what the system should do and its development constraints.

(b) Development - production of the software system

(c) Validation - checking that the software is what the customer wants

(d) Evolution - changing the software in response to changing demands.

A simplified representation of a software process is represented from a specific perspective.
Examples of process perspectives are workflow perspective e.g. sequence of activities, dataflow perspective e.g. information flow, role or action perspective e.g. who does what.
The various Generic process models are waterfall , iterative development and component-based Software Engineering.

1.2. Cost of Software Engineering

Generally 60% of costs of Software engineering are development costs, and 40% are testing costs. For custom software, evolution costs often exceed development costs. Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system

reliability. Distribution of costs depends on the development model that is used.

1.3. Methods of Software Engineering

Structured approaches to software development which include system models, notations, rules, design advice and process guidance.

(a) Model descriptions , e.g. descriptions of graphical models which should be produced,

(b) Rules e.g. constraints applied to system models,

(c) Recommendations ,e.g. advice on good design practice,

(d) Process guidance, e.g. what activities to follow.

1.4. Computer Aided Software Engineering

Software systems that are intended to provide automated support for software process activities may be termed as Computer Aided Software Engineering, or CASE.

CASE systems are often used for method support.

Upper-CASE in which tools to support the early process activities of requirements and design,

Lower-CASE in which tools to support later activities such as programming , debugging and testing.

1.5. Attributes of a good software

The software should deliver the required functionality and performance to the user and should be maintainable, dependable and acceptable.

(a) Maintainability e.g. Software must evolve to meet changing needs,

(b) Dependability e.g. Software must be trustworthy,

(c) Efficiency e.g. Software should not make wasteful use of system resources,

(d) Acceptability e.g. Software must accepted by the users for which it was designed which means it must be understandable, usable and compatible with other systems.

2. Problem Formulation

The proposed model will use a specific approach for product sizing as well as for estimating the amount of effort of person month nominal schedule (PM_{NS}) and the actual time to develop nominal schedule($TDEV_{NS}$) for a software project.

The formula of the nominal-schedule (NS) may be described as follows.

$PM_{NS} = A*x\ size^{E}\ X\ \Pi\ EMi$, where $1<=i<=n$, and $A = 2.94$
$E= B+ 0.01*x \sum SFj$, where $1<=j<=5$, and $B=0.91$

$TDEV_{NS} = C*x\ (PM_{NS})^{F}$
$F= D + 0.2*x*0.01*x \sum SFj$ where $1<=j<=5$ and $C=3.67$ and $D=0.28$

3. Key Challenges in Software Engineering

There are three key challenges in software Engineering, e.g. Heterogeneity, delivery and trust.

(i) Heterogeneity , e.g. developing techniques for building software that can cope with heterogeneous platforms and execution environments,

(ii) Delivery , e.g. developing techniques that lead to faster delivery of software,

(iii) Trust , e.g. developing techniques that demonstrate that software can be trusted by its users.

4. Review of Literature

Ambriola et.al[1] have discussed in their paper that articulation work is a kind of unanticipated task that is performed when a planned task chain is inadequate or breaks down. It is work that represents an open-ended non-deterministic sequence of actions taken to restore progress on the isarticulated task chain, or else to shift the flow of productive work onto some other task chain.
Thus, descriptive task chains are employed to characterize the observed course of events and situations that emerge when people try to follow a

planned task sequence. Articulation work in the context of software evolution includes actions people take that entail either their accommodation to the contingent or anomalous behavior of a software system, or negotiation with others who may be able to affect a system modification.

Balzer et.al[2] have discussed in their paper that Software products represent the information-intensive artifacts that are incrementally constructed and iteratively revised through a software development effort. Such efforts can be modeled using

software product life cycle models. These product development models represent an evolutionary revision to the traditional software life cycle models.

Lionel Briand et.al.[3] in their paper have argued that, in spite of the importance of measurement theory, and in the context of software Engineering, many of the prescriptions and proscriptions are either premature or, if strictly applied, and may represent a substantial hindrance to the progress of empirical research in software Engineering. This argument is based partially on studies that have been conducted by behavioral scientists and by statisticians over the last five decades. They have also presented a pragmatic approach to the application of measurement theory in software engineering. While following the specific approach that may lead to violations of the strict prescriptions and proscriptions of measurement theory, we demonstrate that in practical terms these violations would have diminished consequences, especially when compared to the advantages afforded to the practicing researcher.

N. Fenton et.al[4] have discussed in their paper that a number of powerful statistical

techniques (e.g., parametric statistics) must be proscribed when one cannot prove the measures that is used to fulfill basic scale requirements (e.g., those of the interval scale). However, this very restrictive and rigid view of the use of measurement theory is not shared by many statisticians and data analysts.

5. Reference

[1] Ambriola, V., R. Conradi and A. Fuggetta, " Assessing process-centered software engineering environments ", *ACM Trans. Softw. Eng. Methodol.* 6, 3, 283-328, 1997.

[2] Balzer, R., "Transformational Implementation: An Example ", IEEE Trans. Software Engineering, 7, 1, 3-14,1981.

[3] Lionel Briand, Khaled El Emam , Sandro Morasca , "On the Application of measurement Theory in Software Engineering " , International Software Engineering Research Network technical report #ISERN-95-04

[4] N. Fenton: "Software Measurement: A Necessary Scientific Basis." In IEEE Transactions on Software Engineering, 20(3):199-206, March 1994.