www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 7 July 2017, Page No. 21978-21981
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i7.13

Level of Project Management of Estimation, Planning and Tracking and uses techniques

[1]Rima Debnath, [2]Dr. Amit Asthana
[1] M TECH.-Computer Science and Engineering, [2] H.O.D of Department of Computer Science and Engineering
[1] S.I.T.E, Swami Vivekanand Subharti University, Meerut

Abstract: The process begins with estimating the size, effort and time required for the development of the software and ends with the product and other work products built in different phases of development. Model based technique is one of the best techniques used for estimation. Software engineering is the discipline which paves the roadmap for development within given schedule and effort and with the desired quality. The technique uses different parameters for estimation. The estimates should be accurate, failing to which leads to wrong estimates and consequently results in software crisis.
The tools available for automating some of the activities are great help in the whole development process. However these tools isolate the process of estimation, planning & tracking and calibration. Secondly Software Engineering is a nascent discipline and still the metrics introduced for quantifying the attributes of software are not judgments. Handling large volume of data for these processes is a tiresome task.

Keywords: **software engineering, estimation...**

1. Introduction

Software engineering requires highest degree of analyses, hard work and the management of the two. It is the discipline that aggregates the application of scientific and technological knowledge through the computer programs, to the requirements definition, functional specification, design description, program implementation, and test methods that lead up to test the code [1]. Software engineering is about engineering the software development process.
Effective estimation is essential for proper project planning and control and is one of the most critical and challenging task in the development process. Under-estimating a project leads to quality degradation, employee over exploitation and setting short schedule and hence results in missed deadlines. Allocating more resources to the project and thus increasing the cost of the project without any scope.
Proper planning of the project and tracking the project development is the second essential task for assuring the success of the project. Once the estimates are available the next task is to assign the tasks to individual's project. Estimation plays the key role in the management of the development process. The most recent data available and if standard parameters are being used in the method then those parameters should be well calibrated with the available data.
This chapter comprises the discussion on the current scenario of typical project management and drawbacks of the current scenario, proposed solution over view, benefits of proposed solution, system over view, system development phases, system development schedule and development environment.

2. The Current Scenario

In this section the current scenario for the software estimation, planning and tracking is discussed.

Figure 1.1 shows five categories of software estimation techniques in practice.

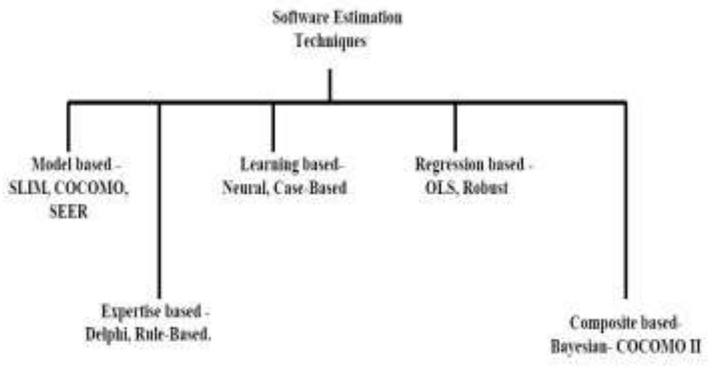
Software Estimation Techniques
Model based - SLIM, COCOMO, SEER
Learning based- Neural, Case Based
Regression based - OLS, Robust
Expertise based - Delphi, Rule-Based.
Composite based- Bayesian- COCOMO II

Figure: 1 Software Estimation Techniques

Various estimation techniques have been developed in the past which follows mathematical model for estimation. SLIM (Software life cycle model), COCOMO (Constructive Cost Model), SEER (System Evaluation and Estimation of Resources)

Case-Based technique is another kind of learning based techniques in which a database of completed projects is maintained and new project's cost is estimated by comparing the new project with similar projects in the database.

In the model based techniques the values for different standard parameters are fetched according to the project being developed and using the equations defined in the model the estimates are calculated [2, 3, 4].

2.1 Drawbacks in the Current Scenario

Study of tools [2, 3, 4] has revealed the following drawbacks in the current scenario

1. Reports at any stage of development are needed another important feature absent in available tools.
2. The tools available for planning used to send the information of task assigned to individuals through

mails and the information pertinent to the assigned task is kept in some version control system.

3. During the development, the management needs to keep track of information about the status of project; the tools available do not have such features.
4. While calibration, past projects' data need to fetched manually.
5. The method used for calibration of tools does not incorporate the expert's judgment in the resulting parameter values.
6. Tools available for the above activities are isolated to each other i.e. the tools available are either estimation tools or for planning and tracking.
7. Any supporting documents or reports should be available to the person in the organization like SRS for the project, design specification. Current tools do not have this feature.

3. System overview

The proposed system is devised by using COCOMO II method for estimation.

1. Scheduling Module
2. Tracking Module
3. Calibration Module

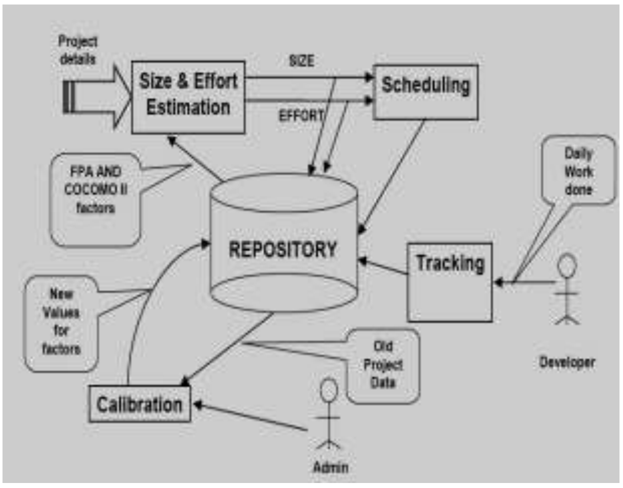

Figure: 2 systems over view

3.1 System Development Phases

The proposed System has three development phases.

1. **Phase I**

Phase I was dedicated to the database design, designing the system and for developing the part which estimates the size, effort and schedule for the project along with the programs for inserting the data into the backend and for its manipulation.

2. **Phase II**

Being estimation model at its place the next point of focus was development of planning and tracking module. Phase II was concerned about developing system for taking inputs from the developers and comparing them.

3. **Phase III**

The last but the most important phase was phase III with the implementation of calibration method using the regression analysis [7].

4. COCOMO II over COCOMO 81

COCOMO 81 was the model of 1980s. This section compares both the flavors of COCOMO [5].

1. In the era of COCOMO 81, software was developed with a limited scope and reusability was not a popular concept and hence there was no such concept in COCOMO 81 to accommodate these new features. COCOMO II incorporates the features mentioned and adjusts the estimates for reuse.
2. The estimation model needs to be consistent with the information available for the projects. In COCOMO 81 there is only three models organic, semi-detached and embedded and these models describe the nature of projects. COCOMO II has three models application composition, early design and post-architecture to be used according to the phases of development.
3. COCOMO 81 gives output in the form of an exact value, which in most of the cases is not accurate. COCOMO II gives output in the form of ranges (optimistic, pessimistic and most likely) according to the phases of development, which is a better way to plan the development process.
4. B is the constant used in both versions of COCOMO. In COCOMO 81 B is a constant value that depends on the type of project (organic, semi-detached, and embedded). Whereas in COCOMO II B is the result of equation containing five scale factors.

5. User

User is the origin of the application. The application starts with the user authentication.

5.1Utility

The package contains the classes responsible for non-business logic functions. The classes in the package are responsible for validation, connection to the database and for parsing the xml file containing the details for database access.

5.2 MailerPkg

The tool has the ability for sending mails to the developers triggered under the conditions selected by the developers. The package is responsible for sending mails.

5.3 ClientPkg

The package contains classes for the operations related to the clients like addition of a new client, searching a client, finding a client's information etc.

DOI: 10.18535/ijecs/v6i7.13

5.4 Project Pkg
The package has a collection of classes responsible for the persistence of the information pertinent to the projects. It includes projects' normal information like description, client name, project leaders name etc.

5.5 ModulePkg
The package contains the classes for keeping information about the module and for its size, effort and schedule estimation.

5.6 TaskPkg
The package contains the classes for keeping information about the task and for its size, effort and schedule estimation.

5.7 ScaleFactorsPkg
The package keeps track of the information of the scale factors in the task.

5.8 ActivityPkg
While developing the tool, a task is further subdivided into 24 activities. The classes in the package are responsible for operating on the activities and for drawing the bar chart representing the time taken by each activity in the organization.

6. COCOMO II Models

Development market in future can be divided into following categories [5]:

1. End-User programming: Increased literacy has increased the number of end users. New tools available in market allows user to develop their own software for simple uses or for information processing. Some examples are spreadsheets, query browsers, planning tools etc.
2. Application Generators: The area which generates the readymade solution which need to be customized according to user.
3. Application Composition: The problems which cannot be solved through single prepackaged solutions needs to be generated by combining different reusable components. Such development comes under the category of application composition.
4. System Integration: Large scale software requiring high degree of system engineering and cannot be generated by application composition comes under this category.
5. Infrastructure: The area concerned with the development of operating system, database management systems etc. comes under this category.

The first category (end user programming) does not need COCOMO II for estimation because its applications are easy to develop with very low complexity and can be developed within hours. For other four sectors COCOMO II has three models of estimation.

7. Related work

Software development effort estimation is the process of predicting the most realistic amount of effort (expressed in terms of person-hours or money) required to develop or maintain software based on incomplete. Effort estimates may be used as input to project plans, iteration plans, budgets, investment analyses, pricing processes [8].

- Expert estimation: The quantification step, i.e., the step where the estimate is produced based on judgmental processes [9].
- Formal estimation model: The quantification step is based on mechanical processes, e.g., the use of a formula derived from historical data.
- Combination-based estimation: The quantification step is based on a judgmental and mechanical combination of estimates from different sources[10].

This implies that different organizations benefit from different estimation approaches. Findings, summarized in[11] that may support the selection of estimation approach based on the expected accuracy of an approach include:

Expert estimation is on average at least as accurate as model-based effort estimation. In particular, situations with unstable relationships and information of high importance not included in the model may suggest use of expert estimation.

The most robust finding, in many forecasting domains, is that combination of estimates from independent sources, preferable applying different approaches, will on average improve the estimation accuracy[12][13][14]. It is important to be aware of the limitations of each traditional approach to measuring software development productivity[15].

COCOMO (Constructive Cost Model) is a model that allows software project managers to estimate project cost and duration. It was developed initially (COCOMO 81) by Barry Boehm in the early eighties[2]. The COCOMO II[1]model is a COCOMO 81 update to address software development practices in the 1990's and 2000's. The model is simple and well tested.

- Provides about 20% cost and 70% time estimate accuracy

COCOMO II estimates project cost, derived directly from person-months effort, by assuming the cost is basically dependent on total physical size of all project files, expressed in thousands single lines of code (KSLOC). The estimation formulas have the form:

Hide Copy Code
<A name=formula></A>Effort (in person-months) = a x KSLOC<SUP>b</SUP>

COCOMO is approximately cube root of effort (in person-months).

The first set is external and can be loosely matched to trade-off triangle/matrix view and its vocabulary is frequently used while negotiating costs with stakeholder, and the second set is COCOMO II internal and usually cannot be used for this purpose. In this trade-off triangle/matrix perspective, schedule is loosely corresponding to SCED (required development schedule), quality to RELY (required reliability), and functionality to a combination of CPLX (product complexity), DATA (database size), TIME (execution time), DOCU (documentation match to life-cycle needs), and occasionally

RUSE, STOR, PVOL parameters. Comparing to classical COCOMO 81, COCOMO II introduces five scale factors, at least three of them are directly related to PM activities, and, thus, raises the role of project management in reducing project costs:

- Takes into account process maturity in the organization (CMM levels)
- Takes into account the degree to which project architecture exists and is stabilized before construction phase
- Takes into account relationships perspective: team cohesion, relations with stockholder

Of course, you can use COCOMO II as it is: choose model, formulas, figure out the values for parameters, and manually calculate project costs. I believe it is a matter what tool you prefer in every spring, filling out tax forms – just simple calculator or automated software tools. I would cover in this section one of the specific tools for making COCOMO II estimates.

```
db_provider

servlet
```

1. `database_scripts`
2. `ui_web_testing`

Now, when initial state of components is set up in the Costar project, we can start making actual COCOMO II estimates. The total cost of the project in COCOMO models is largely determined by total SLOC count, adjustment and scaling parameters for a real project can vary project costs in hundreds of times. The first set, 17 cost drivers, are largely inherited from COCOMO 81 model, and the second set, 5 scale drivers, and are new in COCOMO II model.

7.1 An introduction to COCOMO II techniques and terminology based on real project

COCOMO II as it is choose model, formulas, figure out the values for parameters, and manually calculate project costs.

```
db_provider
```

1. `servlet`
2. `database_scripts`
3. `ui_web_testing`

When initial state of components is set up in the Costar project, we can start making actual COCOMO II estimates. The total cost of the project in COCOMO models is largely determined by total SLOC count. The first set, 17 cost drivers, are largely inherited from COCOMO 81 model, and the second set, 5 scale drivers, and are new in COCOMO II model.

As a project manager, you need to gather information about most important sides of your project such as required product characteristics, required schedule, required product quality, experience and capability of project team, project infrastructure readiness and maturity. Since an introduction to COCOMO II techniques and terminology based on real project, I will only briefly describe assumptions that I made for the case study project.

8. Conclusion

Estimating the project and then planning it without caring about the status of project at any instant of time is a problem worth to be considered. Everything changes with time the team, the process, and the life-cycle. So a static model cannot be used. There are much severe consequences of using a static model for estimation. The core of software crisis starts with the wrong estimation. Thus the calibration of the model being used for the estimation, with the past projects' data experienced by the organization, is an activity of utmost importance.

Instead of being one of the lately introduced branches of Engineering, Software Engineering has grown enough to invent successful software development models, estimation techniques, designs, architectures and testing methods. After so many findings still the metrics needed to measure software precisely are not complete. Under such circumstances experts' judgment cannot be ignored, but it requires time, work and money.

9. References

[1] "McGraw-Hill Dictionary of Scientific and Technical Terms", 6th edition, published by The McGraw-Hill Companies, Inc.

[2] "Construx Estimate tool", www.construx.com

[3] "CoStar tool", www.softstarsystems.com

[4] "SLIM-ESTIMATE tool", www.qsm.com

[5] "COCOMO II Model definition manual", version 1.4, University of Southern California.

[6] Swapna kishore and Rajesh Naik, "Software Requirements and Estimation", Tata McGraw-Hill, New Delhi, 2003.

[7] Bradford Clark, Sunita Devnani-Chulani and Barry Boehm, "Calibrating the COCOMO I1 Post-Architecture Model", 1998 IEEE

[8] Java, Sun MicroSystems,

[9] java.sun.com/javase/downloads/index.jsp

[10] Java Server Faces, java.sun.com/javaee/javaserverfaces/download.html

[11] MySQL, dev.mysql.com/downloads/mysql/4.1.html

[12] MySQL jdbc connector, dev.mysql.com/downloads/connector/j/3.1.html

[13] JFreeChart, www.softpedia.com/get/Multimedia/Gr aphic/Graphic-Others/JFreeChart.shtml

[14] ChartCreator, sourceforge.net/project/showfiles.php?group_id=137466

[15] MyFaces, apache.tradebit.com/pub/myfaces/binaries/

[16] Jdom, www.jdom.org