National Conference on Recent Research in Engineering and Technology (NCRRET-2015)
International Journal of Advance Engineering and Research Development (IJAERD)
e-ISSN: 2348 - 4470 , print-ISSN:2348-6406

COPEMO: A Continuous Project Evaluation Process MOdel

Arpit Mehta[1], Hitesh Vaghela[2]

[1]Asst.Professor in Computer Science Department at DJMIT Mogar
[2] Asst.Professor in Computer Science Department at DJMIT Mogar

***Abstract* -Team synchronization and stabilization are essential especially for large software projects. However, often little is done to assess and reduce the uncertainties and knowledge gaps that exist within the project. As the project progresses through its life cycle, the team can gain more information about the project and team's capabilities. These necessary data can be obtained through performing assessments on the team and project. As these assessments procedures are often complex, discouraging, and difficult to analyze, an effective framework and tool support can greatly enhance the process. Hence, with improved assessment method software project teams can quickly gather the necessary data, determine the actions to improve performance, and result in an improved project outcome in the end. The A COntinious Project Evaluation Process MOdel (COPEPMO) is a framework developed to effectively improve team synchronization and stabilization as well as project effort estimation and scoping by enabling software development teams to quickly track project progress, continuously assess team performance, and make adjustments to the project estimates as necessary.**

***Keywords*-process model; cost estimation; continuous evaluation; project planning; team synchronization and stabilization**

I. INTRODUCTION

As defined in [1], the well-known software "cone of uncertainty" problem in Fig. 1 shows that until the product is delivered, there exists a range of product that the project can result in. Essentially, the wider the "cone of uncertainty" is for the projects, the more difficult it is for projects to ensure accuracies of products and timely deliveries.

For highly precedented projects and experienced teams, one can often use "yesterday's weather" estimates of comparable size and historical productivity data to produce fairly accurate estimates of project effort. More generally, though, the range of uncertainty in effort estimation decreases with accumulated problem and solution knowledge within a "cone of uncertainty". For less experienced teams and unprecendented projects, however, these data are not readily available. To date, there have been no tools or data that effectively monitor the evolution of a project's progression within the cone of uncertainty or to aid software development teams in narrowing the cone of uncertainty for their projects.

To address these problems, we have developed a routine, semi-automated framework and tool support called COntinuous Project Evaluation Process MOdel (COPEPMO).

The framework helps track software project progress and reduces uncertainties as the project progresses through its life cycle by integrating the COCOMO II estimation models in [2], the Unified Code Count (UCC) in [3], and continuous assessment concepts. Referring to the "Cone of Uncertainty" in Fig. 1, the focus of the assessment framework will be from the product design period onwards. Prior to this period, there are many factors that contribute to the uncertainties such as conceptual understandings, requirements volatility, technologies, and available resources. During the product design, we can assume that the requirements are stable to some extent; thus, the development teams contribute the majority of impacts to the uncertainties from this phase onwards.

II. TERMS AND DEFINITIONS

For the scope of our research and this paper, we define the following terms as follows:

Development project refers to the type of projects where the product must be developed from scratch. The development team must write the majority of the source code to implement the end user functionalities.

NDI-intensive project refers to the type of projects that aim at integrating and/or tailoring either one or a set of no developmental items (NDI) or commercial off-the-shelf (COTS) products. As defined in [4], this is when 30-90% of the end user features and capabilities are provided by the NDI or COTS products.

Team synchronization refers to the level of consistencies among the team members with respect to their awareness of each other's understandings, knowledge, experience, and capabilities. The focus is on how well the team members work and coordinate with each other in unison.

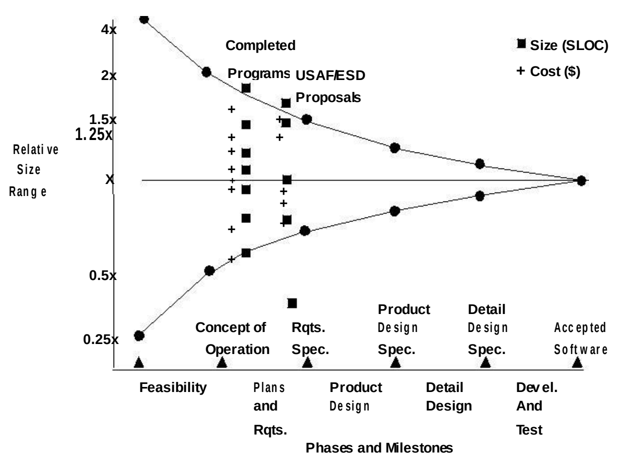


Figure 1. The Cone of Uncertainty

Team stabilization refers to the level of uncertainties that exists within the team and project. The focus is on the number of unknowns that could potentially prevent the team from performing effectively.

III. PROBLEMS AND MOTIVATION

When software development teams lack the proper data and experience, they cannot accurately assess project sizes and team capabilities. These unknowns and uncertainties can typically be reduced with proper assessments as the project progresses. Unfortunately, team assessments are often overlooked even though personnel uncertainties often have significant influences on the cone of uncertainty. Table I shows the productivity range of the COCOMO II parameters representing their magnitudes of impact on estimations and schedules. It is clear that human factors have the most significant impact; therefore, synchronization and stabilization within the development team is essential.

A. Imprecise Project Scoping

Without proper data and experience, software development teams usually generate inaccurate estimates of the effort required for the product to be developed. As a result, the teams are required to renegotiate with the clients to ensure that the product to be developed is within the scope achievable by the development team. This problem is apparent especially in the schedule-as-independent-variable (SAIV) development paradigm in [5] where project deadlines are fixed.
When projects begin with the initial overestimation of resources or effort required, the teams must negotiate with On the other hand, when projects underestimate the re-sources, the teams tend to over promise the goals that the project can achieve.
On the other hand, when projects underestimate the re-sources, the teams tend to over promise the goals that the project can achieve.

Table I. COCOMO II PRODUCTIVITY RANGE

Category	Parameter	Prod. Range	Total
Product	RELY	1.54	10.36
	DATA	1.42	
	CPLX	2.38	
	RUSE	1.31	
	DOCU	1.52	
Platform	TIME	1.63	3.55
	STOR	1.46	
	PVOL	1.49	
Personnel	ACAP	2.00	16.07
	PCAP	1.76	
	PCON	1.51	
	APEX	1.51	
	PLEX	1.40	
	LTEX	1.43	
Project	TOOL	1.50	3.28
	SITE	1.53	
	SCED	1.43	

As the project progresses to the end of its life cycle, the team may start to realize that the remainder of the project is more than they can manage to complete. When this happens, one scenario is they try to satisfy the client by attempting to complete the project as quickly as possible, while the quality of the project may suffer greatly from this attempt and result in higher long-term maintenance costs. Another scenario is they end up delivering a project that is not complete; thus, leaving the clients with unusable or unsustainable products.

B. Project Estimations Not Revisited

During the initial estimation for software projects, the teams, especially for inexperienced teams and unprecedented projects, typically do not have sufficient data to carefully analyze and perform the necessary predictions. These missing pieces of information include aspects and characteristics that are specified in the COCOMO II cost drivers and scale factors. In most cases, the project estimation turns into a constant value at the time that the project enters the development phase.

Usually, the only estimation activities that are done for the project are based on early assessments with insufficient information. As the projects proceed through the development phase, the status and progress of the projects are not assessed and re-assessed by the team in order to analyze the accuracy of the initial estimates. Although the project status maybe reviewed by the stakeholders during the major milestones, the team usually does not perform minor assessments throughout the project life cycle as discussed in [6]. There are significant levels of uncertainties at the beginning of the project as there are instabilities in concepts, requirements, choices of COTS products and cloud services, and directions that the project can proceed on.

C. Manual Team Assessments are Tedious

The tasks of manually assessing the project progress are tedious and discouraging to the team due to the amount of effort required and complexity. In order to collect enough information to have useful assessment data, the teams often need to perform various surveys and reviews to determine how well the team had performed in the previous iterations [6]. In processes with high maturity level ratings such as CMMI levels 4 and 5, development teams already must constantly go through various quantitative and qualitative assessment tasks to ensure high level of quality and performance. These procedures can take up significant amount of time and effort to perform effectively.

Furthermore, to accurately report the progress of software development projects in traditional processes, the teams are required to carefully count the number of source lines of code (SLOC) they have developed at major milestones, analyze the logical lines of code, and compare them to the estimates that they had created initially. These tasks require significant amount of effort to collect the necessary information to evaluate the initial estimations performed for the project and to identify how well the team is actually performing. These tasks require significant amount of effort to collect the necessary information to evaluate the initial estimations performed for the project and to identify how well the team is actually performing.

Traditional assessment methods can discourage the team from constantly performing assessments of the project status due to tedious and complex work. This usually leads to inability to effectively detect issues and inconsistencies within the team and project. As a result, teams may waste significant amount of effort working in inconsistent states.

D. Limitations in SDLC Cost Estimation

Regardless of what software cost estimation technique is used, there is little that the technique can compensate for the lack of information and understanding of the software to be developed. As clearly shown in [2], until the software is delivered, there exists a wide range of software products and costs that can affect the final outcome of the software project. Without proper understanding of the parameters in the software cost estimation models, software development teams would end up providing values they "think" are correct or simply done by guessing.

In addition to the fact that the initial estimations lack the necessary information to achieve accurate estimates, the software design and specifications are prone to changes throughout the project life cycle as well, especially with an overenthusiastic client or in a more agile software engineer-ing environment. Software cost estimation models cannot automatically adapt, or compensate, to these unknowns and changing environments.

E. Overstating Team's Capabilities

When teams are inexperienced in the use of software estimation models, they fail to understand all the parameters that exist in those models even after coaching, mentoring, and tutorials. In COCOMO II, the 17 cost drivers and 5 scale factors require thorough understanding in order to correctly specify the values for them. Project planners often end up with unrealistic values for the parameters or may end up guessing the values instead. These result in unrealistic estimations of the software projects.

Furthermore, from the business point of view, people tend to be over optimistic about their estimations. Team's capabilities are misrepresented in project proposals causing wide gaps between what business customers want versus what the team can deliver. This again introduces problems discussed in section III-A. not when development takes longer than expected. Planning poker in [8] is also another common method for planning each iteration; however, in order to plan effectively, it requires expert opinions and analogies. The Program Evaluation and Review Technique (PERT) sizing method in [9] focuses on sizing the individual components The estimation technique requires t he developers to provide the optimistic, most likely, and pessimistic sizes of the software.

The PERT method reduces the bias towards overestimation and underestimation, although people tend to choose the "most likely" estimates towards the lower limit, but the actual product sizes cluster towards the upper limit. Based on [1], this underestimation bias is due to the following reasons:

- People are optimistic and have the desire to please.
- People do not have complete recalls of past experiences.
- People are not familiar with the entire software job.

The Wideband Delphi Technique in [1] is an alternative method to the Delphi Technique in [10] to broaden communication bandwidth among team members to address any uncertainties. However, the process requires experts' knowledge extensively. The estimations are presented to the experts to discuss on places where estimations vary.

The COCOMO-U covered in [11] extends the COCOMO II model to allow estimating with uncertainty by using the Bayesian Belief Network. It enables estimations to be done with COCOMO II even when there are unknown parameters. However, the method also relies heavily on expertise of its users in specifying the uncertainties of the cost drivers and scale factors appropriately.

A. Project Tracking and Assessment

To date, there are many project tracking and assessment methods. Presented in [12], PERT is well-known for han-dling large and complex projects as it places emphasis on the time involved to complete tasks instead of specific start and end dates. The PERT network chart allows project teams to manage the uncertainties within the project because critical paths can be identified and updated making the progress of the project visible to the stakeholders. However, considering the number of tasks and potential dependencies within the project, the network charts can grow large and unusable fairly quickly. Once the charts grow too large, they are often disregarded for project management.

Another popular approach for progress tracking and measurement is the Goal-Question-Metric (GQM) in [13]. The GQM captures the progress from the conceptual, operational, and quantitative levels allowing the method to align with the organization environment as well as project context. However, the GQM is only useful when used correctly by specifying the appropriate goals, questions, and measurements to be monitored. Otherwise, the measurements can be meaningless and impractical.

Furthermore, the Earned-Value Management (EVM), burn up, and burn down charts in [14] are good for capturing the project progress based on team's velocity and completed features. However, these approaches are not effective at responding to major changes during each iteration.

V. THE COPEPMO FRAMEWORK

The COPEPMO framework consists of 3 sub-frameworks that together aim to improve project tracking, project estimation, team synchronization, and team stabilization through-out the project life cycle.

A. Project Progress Tracking

The first part of the framework is to help unprecedented projects and teams track their progression through the project life cycle.

This helps the project teams reduce the uncertainties of estimations and achieve eventual convergence of the estimated and actual effort spent on the project. For development projects, the framework integrates the UCC tool and the COCOMO II model to allow quick progress tracking and estimation based on the amount of source code developed. Details of the model can be found in [15]. This framework enables the team to track the progress of the project based on the actual work done and computes new estimations based on those data.

Furthermore, for NDI-intensive projects, the framework utilizes the Application Point model of COCOMO II in [2] for effort estimation and progress tracking. Instead of tracking the number of lines of code written, the model would track the number application points developed, which include the number of screens, reports, and third-generation language (3GL) components.

B. Continuous Team Assessment

Team synchronization and stabilization are essential to successful project outcomes because knowledge gaps and inconsistencies among the developers are common problems in team projects. The second framework is an assessment technique to aid in reducing those gaps in order to help stabilize the team and project understandings. The methodology is based heavily on the IBM Self-Check concept in [16], a survey-based approach effective for detecting and narrowing the knowledge gaps among the team members.

We have developed a method to assess team's performance in the following areas:

- Requirement
- Business case analysis
- Architecture and design development
- Planning and control
- Feasibility evidence Personnel
- Capabilities Collaboration

The survey questions do not contain right or wrong answers; however, they ask each developer for their opinions and their perspectives on the team's capabilities and performance. Similar to the IBM Self-Check approach, each team member would answer the survey questions individually. The deviation and mismatches in the answers are used to determine weak and inconsistent areas and the team must identify actions to resolve those issues.

The survey questions do not contain right or wrong answers; however, they ask each developer for their opinions and their perspectives on the team's capabilities and performance. Similar to the IBM Self-Check approach, each team member would answer the survey questions individually. The deviation and mismatches in the answers are used to determine weak and inconsistent areas and the team must identify actions to resolve those issues. The COPEPMO assessment framework currently consists of 45 questions. We utilized 2 approaches to develop the survey questions.

The first approach was to analyze the Software Engineering of various software companies doing projects by conducting team assessments for individual team members to evaluate their team's strengths, Weaknesses, and issues focusing on The various areas mentioned earlier. The assessment was done for various projects during the 2011 and 2013 academic years. These assessment data were then analyzed and the critical ones were picked out and categorized into their respective categories. The questions were developed to address the knowledge gaps and potential issues that commonly occurred within these assessment data. Since the questions were derived from teams' strengths, weaknesses, and issues, they are focused mainly to help resolve team issues and stabilize team performance.

In the second approach, we adopted questions from [17]. The research consists of 2 frameworks - SE Performance Risk and SE Competency Risk - which show that the effectiveness of software engineering practices can be assessed both by the performance of systems engineering (SE) functions and the competency of the personnel performing those practices. Both frameworks are currently used in the industry for assessing and analyzing the capabilities of the project personnel and for identifying potential risks and weaknesses that should be addressed. The questions adopted from these frameworks focus on evaluating the capability of the team as a whole as well as the individual members. This is to establish that the major concerns are identified and addressed by ensuring that the project team has sufficient capabilities and experience.

Moreover, since the SE Performance Risk and SE Competency Risk frameworks cover various aspects of systems and software engineering practices, we use them to verify and validate the questions derived in the first approach. Because the first approach focused specifically on team synchronization and stabilization, we were not able to adopt the questions directly from the SE risk frameworks as they were geared towards analyzing performance and competency of personnel. However, we compared the various aspects of the performance and competency risks with our set of custom questions to make sure that the risks and concerns in software engineering practices are properly addressed. This is to ensure that the all of the assessment questions that we use are consistent with the industry practices.

C. COCOMO II Estimation Adjustment

As mentioned in section III-B, using the COCOMO II estimation model often does not reflect the actual project situations because planners do not have the necessary understandings of the estimation model. The questions developed for the survey-based assessment method contain correlations to the COCOMO II cost drivers and scale factors. Each question impacts either one or multiple COCOMO II parameters where some may impact certain parameters more than others.

As team members answer the survey questions, the framework analyzes the answers and provides suggestions on changes to be made to the team's COCOMO II estimates to reflect the way they answered the survey. Table II shows a selected set of sample assessment questions and the corresponding COCOMO II cost drivers that are impacted.

Detailed discussions of all the assessment questions are beyond the scope of this paper.

To determine the relationship and impact factor between the survey questions and the COCOMO II parameters, we use expert's advice by surveying a group of COCOMO II experts and experienced users. For each survey question, the expert would identify the COCOMO II parameter that is impacted. Weights were applied to the level of expertise and experience of the person surveyed in order to justify valuable inputs and reduce bias. These data were then combined and averaged resulting in the weight, or impact factor, that each survey question has on the COCOMO II parameters. Based on these impact factors, the COPEPMO framework analyzes the assessment data and computes the adjustments that should be made to the COCOMO II ratings.

VI. USING THE COPEPMO FRAMEWORK

In the COPEPMO framework, the project assessment is expected to be done consistently throughout the project life cycle to help development teams monitor their progress, while reducing any uncertainties and knowledge gaps within the team. The duration between each assessment, or iteration, can be specified based on the team's preferences or as necessary. The monitoring mechanism can help teams ensure the feasibility of the project timeline and encourages the team to re-negotiate or re-scope the requirements, features, or budget if needed. Figure 2 shows the workflow of the COPEPMO framework. The development projects and NDI-intensive projects utilizes different methods for computing the estimated effort as discussed in section V-A. The different approaches are discussed later in this section. However, both types of project follow the same assessment method and team synchronization process of the framework for performance and estimation improvements.

A. Framework Support for Development Projects

For development projects where the majority of the source code needs to be written by the development team, the COPEPMO framework relies on the UCC tool to help track project progress and compute new project estimates. As discussed in section V-A, the integration of the UCC tool allows for development teams to quickly view their progress based on the source code developed, and the framework uses those data to compute updated, more accurate estimates for the project as shown in the workflow process in Fig. 3. Based on the information known at the beginning of the project, the development team specifies the modules to be developed and all of the corresponding COCOMO II driver ratings for each module. As the project progresses, the development team can make adjustments to the estimated source lines of code (SLOC) and COCOMO II parameters as necessary.
Based on the information known at the beginning of the project, the development team specifies the modules to be developed and all of the corresponding COCOMO II driver ratings for each module. As the project progresses, the development team can make adjustments to the estimated source lines of code (SLOC) and COCOMO II parameters as necessary. However, once the source code development has begun, the team can utilize the benefit of the UCC tool, which automatically counts the total number of logical lines of code for each source code file.

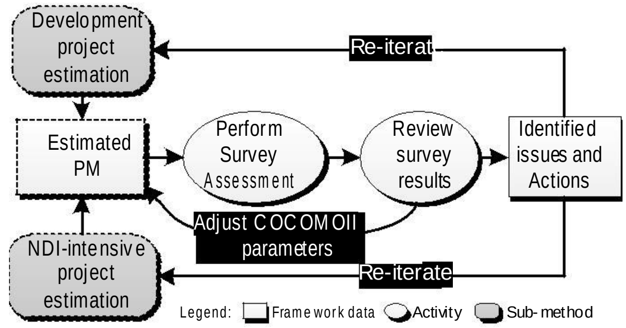


Figure 2. Workflow of the COPEPMO framework

The framework uses the accumulated SLOC and converts it into equivalent effort using the COCOMO II model. The developers then provide the percentage developed, tested, and integrated for each module. All of these data are used to compute the new estimated effort required to complete the project.

For each iteration, the development team is required to assess the team's performance and status by performing survey-based assessments. Each team member fills out and submits the surveys individually without knowing each other's answers. Based on the information known at the beginning of the project, the development team specifies the modules to be developed and all of the corresponding COCOMO II driver ratings for each module. As the project progresses, the development team can make adjustments to the estimated source lines of code (SLOC) and COCOMO II parameters as necessary. However, once the source code development has begun, the team can utilize the benefit of the UCC tool,

Table II. SAMPLE ASSESSMENT QUESTIONS WITH CORRESPONDING CO-COMO II COST DRIVERS

Category	Question	Impacted Cost Drivers
Personnel Capability	Does the team have the required do main knowledge and experience to de velop a stable architecture for the sys-tem?	ACAP PCAP APEX PLEX LTEX
Collaboration	Do you have the proper mechanisms to ensure high level of collaboration and keeping all stakeholders in the loop (i.e. use of Google Groups, MSN meetings, WebEx, etc.)?	TOOL SITE
Requirements Gathering	Project details, requirements, bound aries and scopes are thoroughly re-searched and understood by the team?	CPLX DOCU ACAP PCAP APEX

which automatically counts the total number of logical lines of code for each source code file. The framework uses the accumulated SLOC and converts it into equivalent effort using the COCOMO II model. The developers then provide the percentage developed, tested, and integrated for each module. All of these data are used to compute the new estimated effort required to complete the project.

For each iteration, the development team is required to assess the team's performance and status by performing survey-based assessments. Each team member fills out and submits the surveys individually without knowing each other's answers. This is to prevent any bias in the answers. The standard deviation is computed to detect any inconsistencies between the answers for each question. Since each survey question has been designed to focus on the individual's view on the team's performance and project status, a high deviation in answers means that there exist differences in opinions or understandings within the team. Flags are triggered for questions with high answer deviation raising issues for the team to discuss. The team then develops actions to take in order to resolve those issues in the next iterations as prolonged inconsistencies and differences in understandings among the developers can result in critical problems later in the project.

Finally, based on the survey results, the COPEPMO framework provides the team with suggestions on the adjustments that should be made to the COCOMO II parameters. The suggestions are reflective of the consolidated answers given by each team member, and since each survey question has different levels of impact on each of the COCOMO II parameters, these suggestions are computed based on the relationship discussed in section V-C. The adjusted parameters create more realistic estimations for the project.

B. Framework Support for NDI-Intensive Project

With the integration with the UCC tool to help track actual project progress, the COPEPMO framework is highly beneficial to development projects. However, the framework also has strong support for NDI-intensive projects as well. Conceptually, the use of the COPEPMO framework is the same as with the development project. Instead of providing the detailed information for modules, source code, and COCOMO II parameters, the development team provides the number of application points – screens, reports, and third- generation language (3GL) components – as well as their experience and tool support levels. Using the COCOMO II Application Point model, the effort required to complete the project is computed based on these information. For each iteration, the development team provides the number of application points developed up to that point as well as the corresponding percentages developed and tested respectively. The survey-based assessment is also required to be completed individually by the team members, which is exactly the same as that for the development projects. However, based the results of the survey, the COPEPMO framework provides adjustment suggestions for the developer's capability and experience and the integrated computer-aided software engineering (ICASE) maturity and experience levels These are the two dynamic parameters that affect the productivity rate of the team and the estimation project.

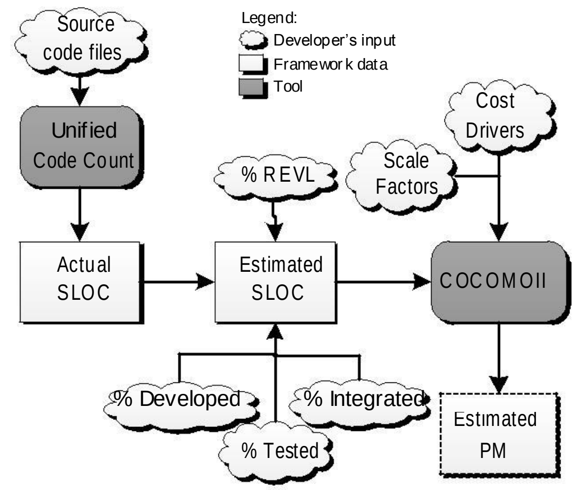


Figure 3. Workflow for development projects

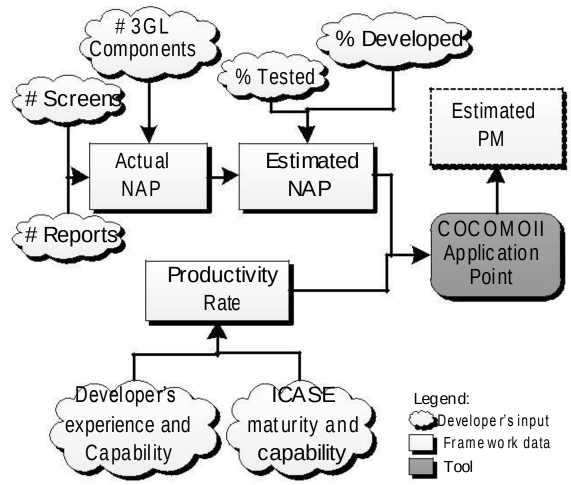


Figure 4. Workflow for NDI-intensive projects

C. Tool Support

Having an effective tool to support the framework is essential in enabling software teams to utilize the framework to its potential. The COPEPMO tool has been developed using the IBM Jazz technology in [18] to support the framework. The Jazz platform has been chosen for its capabilities and extensibility including support for team and user managements as well as high collaborative environment. Detailed discussion of the tool is beyond the scope of this paper.

C. Tool Support

Having an effective tool to support the framework is essential in enabling software teams to utilize the framework to its potential. The COPEPMO tool has been developed using the IBM Jazz technology in [18] to support the framework. The Jazz platform has been chosen for its capabilities and extensibility including support for team and user managements as well as high collaborative environment. Detailed discussion of the tool is beyond the scope of this paper.

VII. OBTAINING THE DATA

The application of the framework was done in a Project Development environment using the data obtained from various

Software Company. In this various project development team of various companies experience how to use good software engineering practices to develop software systems from the exploration to the operations phases.

Each year, teams of Seven or Eight Employee are formed to develop projects for real-client. The some employee come from undergraduate programs with less than 2 years of working experience. Other are full-time professionals with at least 5 years of experience. The clients are various software companies, neighbor-hood corporations, and nonprofit organizations. Typically, the fresher employee act as operational concept engineers, requirements engineers, software architects, UML modelers, coders, life cycle planners, and feasibility analysts, while the senior employee act as Integrated Independent Verification and Validation (IIV&V) personnel, quality assurance personnel, and testers.

Depending on the scopes and complexities, the projects are completed either within a 12-week or 24-week schedule. After the teams gathered and negotiated their requirements, they used the COPEPMO tool to define number of iterations and milestones. We specified the length of each iteration to be one half week long, while the milestones correspond to the Incremental Commitment Spiral Model milestones in [4]. The teams used this tool weekly to report the development progress as part of their progress report. For each week, each team member answered a general project progress survey. For each milestone, a more in depth survey is generated in order to assess the milestone achievements and performances. As the projects progressed in the semester, the teams continuously recalibrate the cost and schedule estimations based on the improvements suggested by the COPEPMO tool in order to reflect the teams' statuses and performances.

The COPEPMO framework and tool have been deployed at various software companies. The various project have 79 software engineers making up 13 project teams of which 5 were development projects and 8 were NDI-intensive projects. By the end of 12th week, 5 projects were completed with products completely delivered to the clients, while the remaining projects continued. The Employee were surveyed to observe the feedbacks and effectiveness of the use of the framework and tool.

VIII. ANALYSIS

Fig. 6 shows the results of the 79 employee surveyed. The surveys focused on the way the software employee reflected on their teams in the aspects on synchronization, stabilization, and strengths. We asked the individual team members to rate his/her own team on the following categories:

- Level of team synchronization in understanding and knowledge
- Level of uncertainties in project
- Team's strength and performance
- Level of effort reduction in resolving risks, team inconsistencies, and project issues

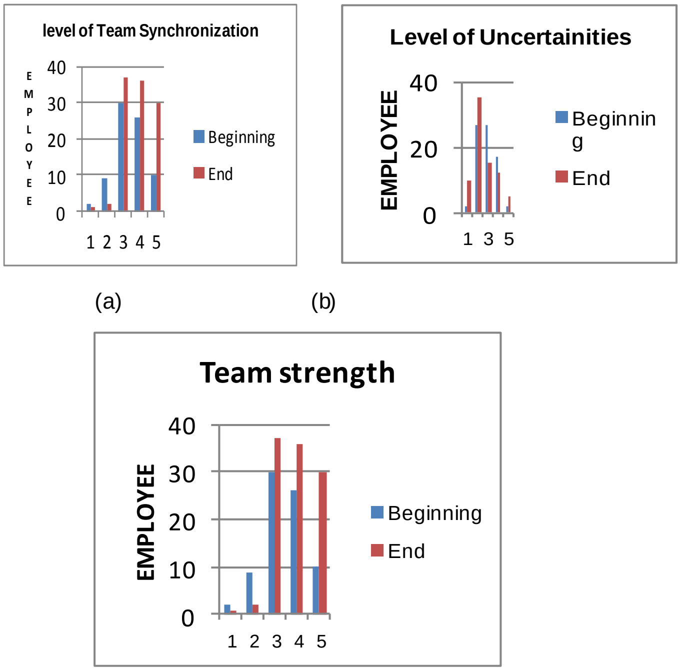
As shown in Fig. 6(a) and Fig. 6(b), the majority of the team members felt that their teams had been much more synchronized as the project progressed, while the levels of uncertainties that existed within the teams and the projects had reduced. Additionally, with better synchronization and stabilization process, Fig. 6(c) clearly shows that the teams' strengths had significantly increased at the end of the Project phases as compared to the beginning. Of all the Employee surveyed, 80% of them said that they were more satisfied with the team and the project at the end , 15% were indifferent, and 5% were less satisfied possibly due to unresolvable issues among the team members. Currently, we only focused on how the framework improved the members' perspectives on their team's status and project performance over time. We have yet to compare these results with the historical data.

Furthermore, Fig. 6 shows that effort required by the teams to resolve project related issues have been remarkably reduced as the teams continued to use the COPEPMO tool throughout the project phases. The software employee were asked to evaluate the level of effort required to address and resolve issues in the areas of a) risk resolution, b) communication and understandings, and c) project issues and defects. The assessment method of COPEPMO allowed the teams to quickly identify problems that existed within the teams as well as gaps in knowledge and understandings that existed among the team members. It allowed the teams to address those issues more effectively because they could be detected early before becoming critical problems

.

In addition to the positive feedback received from employee in 2013 we have also compared the use of the COPEPMO tool with the previous projects during Fall 2011 and 2012 semesters when the tool was not utilized. For every week during the project life cycle, each team member was required to report the effort, in hours, they spent on the project activities in the following categories: 1) operational concepts development, 2) requirements engineering, 3) de-sign and architecture, 4) planning and control, 5) feasibility evidence analysis, 6) quality management, 7) testing, 8) communication and synchronization, and 9) performance control. Shown in Fig. 7 is the average effort spent by each person on the project during each week in the semester. The effort required for the projects in Fall 2011 was significantly reduced in each week compared to that of the previous years. Moreover, we also analyzed the average of the total effort spent on each of the activities shown in Fig. 8

. In many of the activities such as operational concepts development, requirements engineering, planning and control, and feasibility evidence analysis, the average effort spent were slightly lower than the previous years. Since the amount of work required to perform these activities remain the same across all three years, it is expected that the average efforts only show some slight reduction. However, the most significant reduction in the average effort was in the areas of communication and synchronization. The level of effort required for the team members to synchronize with each other and to stabilize the team's knowledge and for the projects in 2013 was significantly understanding had greatly decreased. The COPEPMO tool and framework provided the teams with an effective mechanism to detect inconsistencies within the team and help reduce knowledge gaps that existed.



(a) (b)

(c)

Figure 5. Survey results of 50 employees (13 projects). (a) shows the level of synchronization within the teams (1 = unsynchronized, 5 = highly synchronized). (b) shows the level of uncertainties that existed within the team (1 = low uncertainties, 5 = high uncertainties). (c) shows the level of team strengths (1 = low, 5 = high).

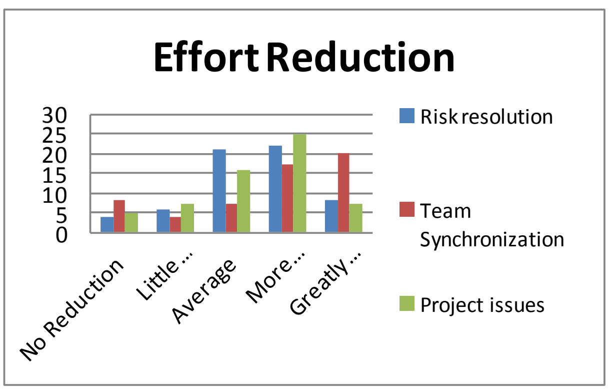


Figure 6. Reduction of effort required to address/resolve 1) risk resolution, 2) team synchronization, and 3) project issues

With this, the teams could quickly resolve those issues and focus more attention on completing the project instead having to waste effort in trying to work in an unsynchronized tea.

Finally, the clients were surveyed at the end of various phases for their satisfactions on the teams and the projects. Based on their evaluations, more clients were satisfied with the project overall in 2013 compared to the previous years. More importantly, one project in 2013 was initially planned for a 24-week schedule, but based on the progress tracking and re-estimations reported by the COPEPMO tool, they were able to determine that the project only required half the resources and could be completed within a 12-week schedule instead. The project immediately proceeded with the development and the product was de-livered to the client with 100% of end user functionalities implemented. In the previous years, when projects had to switch from a 24-week to 12-week schedule, they required major re-scoping of features and capabilities in order to meet the new deadline

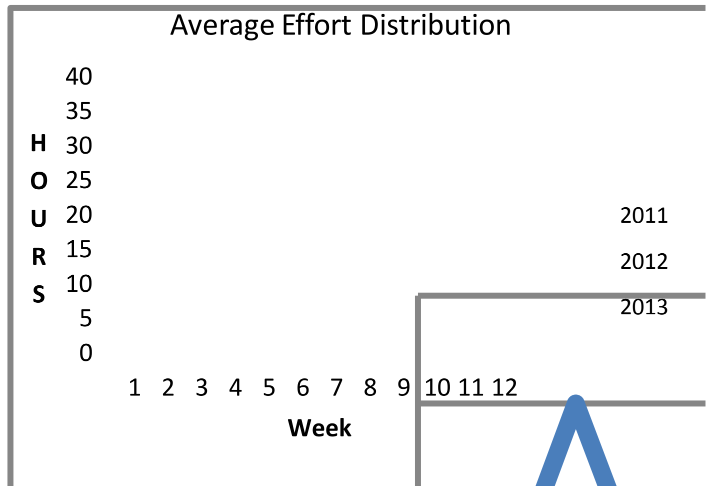


Figure 7. The average effort by week

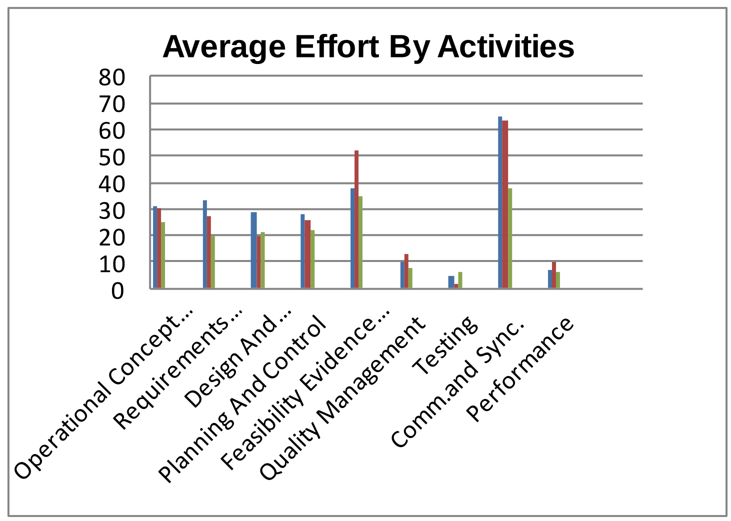


Figure 8. The average effort by activities

IX. THREATS TO VALIDITY

Representativeness of projects. Most projects were small-sized e-services projects, which may not represent the industry at a larger scale. Nonetheless, the projects were done for real clients with real fixed schedules and costs. Also, all projects followed the same incremental development process and project activities that are used in the industry.

Representativeness of personnel. The majority of the project teams consisted of members with less than 2 years of industry experience. the off-campus students and clients were working professionals. Furthermore, the verification and validation processes were done by the off-campus students to help ensure the integrity of the project artifacts.

Validity of assessment data. Since many of the survey questions were derived from strengths, weaknesses, and issues observed from software engineering students, the assessed data may not be valid in the industry. We will be verifying the assessment questions with the experts in the industry as part of our future w ork.

Changes in life cycle process from 2011 - 2013. The life cycle process remained largely the same from 2011 to 2013. A minor change in the requirement negotiation process w as introduced during the Fall 2013 w ith the use of a new negotiation tool and Planning Poker concept in the requirement prioritization process. How ever, the core process and practice were still based on the WinWin process in [19], [20], and [21].

Additionally, the team size had reduced to 6 team members instead of 7 members from the previous years. This may have slightly affected the level of synchronization required among the team members. How ever, observing the level of effort reduced in communication and synchronization, w e feel that the decrease in 1 team member w ould not have had such significant impact.

X. CONCLUSIONS AND FUTURE WORK

We have developed a team improvement framew ork for continuous assessment to aid in the team synchronization

Another target for our future w ork is to experiment the COPEPMO framew ork and tool in the industry to observe the validity of the framew ork. The survey assessment framew ork w ill be verified and validated by experts in the industry, and the whole framew ork w ill be tested w ith projects of different sizes and domains. Since the majority of projects w ere e-services projects w ith maximum of 7 team members, it w ould be valuable to observe the effectiveness of the COPEPMO and stabilization process by reducing the levels of uncertain-ties that exist in the project and among the team members. The unknow ns and uncertainties in the project can be greatly reduced w ith the use of proper assessment mechanisms. How ever, since team assessments can be tedious, labor intensive, and require high level of expertise and time for data analysis, they are often overlooked as effective means for team improvements.

The A COntinious Project Evaluation Process MOdel, or COPEPMO, w as introduced to help softw are development teams reach a higher level of synchronization and stabilization w ithout having to go through complex processes. It consists of three main parts:

1) Project progress tracking
2) Continuous team assessment
3) COCOMO II estimation adjustment

The process framew ork provides mechanisms for the team to quickly track their project progress based on the amount of development completed and to detect issues and know ledge gaps w ithin the team through its quick assessment method. As the team continuously performs the assessment through-out the project life cycle, uncertainties are reduced, w hile team and project understandings increase. Additionally, the framew ork provides suggestions to the adjustments that should be made to the COCOMO II estimations created by the team. This allow s the development team to continuously monitor the accuracy of their project estimates and make rational adjustments to them as necessary.

we introduced the COPEPMO framew ork and deployed the tool to the various softw are companies consisting of 79 graduate students and 13 projects. As show n in our analysis, the utilization of the COPEPMO framew ork and tool provided the teams w ith the mechanis and stabilization process by reducing the levels of uncertain-ties that exist in the project and among the team members. The unknow ns and uncertainties in the project can be greatly reduced w ith the use of proper assessment mechanisms. How ever, since team assessments can be tedious, labor intensive, and require high level of expertise and time for data analysis, they are often overlooked as effective means for team improvements.

The framew ork provides strong support for both development projects and NDI-intensive projects. For development projects, the COPEPMO framew ork relies on the UCC tool to report the project progress based on the SLOC developed and uses the COCOMO II estimation model for effort con-versions. For NDI-intensive projects, on the other hand, the framew ork uses the COCOMO II Application Point model to track the number of screens, reports, and 3GL components completed by the developers. The assessment framew ork of COPEPMO analyzes the team’s survey assessment data and provides improvement suggestions to the parameters used for estimation calculation in both COCOMO II models.

We introduced the COPEPMO framew ork and deployed the tool to the various softw are companies consisting of 79 graduate students and 13 projects.. As show n in our analysis, the utilization of the COPEPMO framew ork and tool provided the teams w ith the mechanism to effectively synchronize and stabilize the teams in various areas such as communications, understandings, and performance. With simple and effective assessments, the teams’ performance had greatly improved w ith reduced uncertainties, w hile the effort required for the project had substantially decreased. With better estimates and effective project tracking mechanisms, the teams w ere able to constantly monitor the progress and the feasibility of their to effectively synchronize and stabilize the teams in various areas such as communications, understandings, and performance. With simple and effective assessments, the teams’ performance had greatly improved w ith reduced uncertainties, w hile the effort required for the project had substantially decreased. With better estimates and effective project tracking mechanisms, the teams w ere able to constantly monitor the progress and the feasibility of their projects ensuring that the scopes can be delivered w ithin the

Framew ork used in larger teams w here team synchronization and stabilization tend to be much more complex.

ACKNOWLEDGMENTS

The authors w ish to thank Deepak Kuhlare for their effort in helping w ith the COPEPMO tool development.

REFERENCES

[1] B. Boehm, Softw are Engineering Economics. Prentice-Hall, 1981.

[2] B. Boehm et al., Softw are Cost Estimation w ith COCOMO II. Prentice-Hall, 2000.

[3] "Unified CodeCount." Internet: http://sunset.usc.edu/research/ CODECOUNT/, [Jan. 20, 2012].

[4] S. Koolmanojwong and B. Boehm, "The Incremental Com-mitment Model process patterns for rapid-fielding projects," in Proc. of the 2010 Int. Conf. on New Modeling Concepts for Today's Software Processes: Software Process (ICSP'10), Paderborn, Germany, 2010, pp. 150–162.

[5] B. Boehm et al., "Using The Spiral Model and MBASE to generate new acquisition process models: SAIV, CAIV, and SCQAIV," CrossTalk, pp. 20–25, Jan. 2002.

[6] W. Krebs, P. Kroll, and E. Richard, "Un-assessments reflec-tions by the team, for the team," in Proc. of the Agile 2008, Washington, DC, USA, Aug. 2008, pp. 384 –389.

[7] M. Cohn, Agile Estimating and Planning. Prentice-Hall, 2006.

[8] J. Grenning. "Planning poker," Internet: http:// renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf, Apr., 2002 [Jan. 20, 2012].

[9] L. Putnam and A. Fitzsimmons, "Estimating software costs," Datamation, pp. 189–198, Sep. 1979.

[10] J. A. Farquhar, "A preliminary inquiry into the software estimation process," The Rand Corporation, Santa Monica, CA, Tech. Rep. RM-7271-PR, 1970.

[11] D. Yang et al., "COCOMO-U: An extension of COCOMO II for cost estimation with uncertainty," in Software Process Change, ser. Lecture Notes in Computer Science, Q. Wang, D. Pfahl, D. Raffo, and P. Wernick, Eds. Springer Berlin / Heidelberg, 2006, vol. 3966, pp. 132–141.

[12] J. D. Wiest and F. K. Levy, A Management Guide to PERT/CPM. Englewood Press: Prentice-Hall, 1977.

[13] V. R. Basili, "Applying the Goal/Question/Metric

[14] P. Kroll and W. Krebs, "Introducing IBM Rational Self Check for software teams," Internet: http://www.ibm.com/ developer works/rational/library/edge/08/may08/kroll krebs, May 3, 2008 [Jan. 20, 2012].

[15] B. Boehm et al., "Early identification of SE-related program risks," Systems Engineering Research Center, Tech. Rep. SERC-2009-TR-001, Sep. 2009.

[16] "Jazz Foundation." Internet https://jazz.net/projects/jazz-foundation/[Jan.20,2012].

[17] B. Boehm et al., "Software requirements negotiation and renegotiation aids," in Proc. of the 17th Int. Conf. on Software Engineering (ICSE '95), Seattle, Washington, United States, 1995, pp. 243–253.

[18] M. J. Lee, "Foundations of the WinWin requirements ne-gotiation system," Ph.D. dissertation, University of Southern California, 1996.

[19] B. Boehm et al., "Using the WinWin Spiral Model: A case study," Computer, vol. 31, pp. 33–44, Jul. 1998.

[20] W. S. Humphrey, A Discipline for Software Engineering. Addison-Wesley, 1995.

[21] W. S. Humphrey, Introduction to the Team Software Process. Addison-Wesley, 1