Towards Objective Estimations of Software Implementation Progress

Camargo Cruz Ana Erika
Prof. Ochimizu Koichiro
March, 2011
1. Motivation and Goal
2. Background and Related Work
3. Our Approach
4. Experimental Results
5. Conclusions and Future Work
Software implementation progress is often measured subjectively by software developers or project managers. Such reports are often open to misconception leading to inaccurate schedule estimations, which is considered as one of the major causes of schedule slippage.
To devise a methodology to estimate software implementation progress objectively and timely
A Use Case Based EVA Approach

“Monitoring software projects with earned value analysis and use case points”, Jinhua Li et al., 7th IEEE/ACIS International Conference on Computer and Information Science, 2008

What?

They propose the use of “Use Case Points” (UCP) and Earned Value Analysis (EVA), in order to consistently express project baselines and measure technical progress of software projects.

How it works?

• A project is estimated in terms of dollars and UCPs

• Then every use case of the system is also valued in UCP and easily translated into dollars.

• Earned Values (EV) assignment: task begins ⇒ 50% of the task budget is earned, task is completed ⇒ 100% of the budget is earned
A Use Case Based EVA Approach

“Monitoring software projects with earned value analysis and use case points”, Jinhua Li et al., 7th IEEE/ACIS International Conference on Computer and Information Science, 2008

Team: 1 mentor, 3 students
System: Automobile Rent system
Time: 20 weeks
Total cost of the System: $4500
12 Use cases with a total of 75 UCP

<table>
<thead>
<tr>
<th>WBS ELEMENTS</th>
<th>% BUDGET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>10.00%</td>
</tr>
<tr>
<td>Requirements</td>
<td>15.00%</td>
</tr>
<tr>
<td>Design</td>
<td>20.00%</td>
</tr>
<tr>
<td>Implementation</td>
<td>30.00%</td>
</tr>
<tr>
<td>Assessment</td>
<td>25.00%</td>
</tr>
</tbody>
</table>

- In order to plan and measure the progress in conventional way, they converted the EV of completed UCP into dollars:
  \[
  \frac{4500}{75} \text{ UCP} = \$60 \text{ per UCP}
  \]
- Since the management activity (communication and coordination) spreads over the whole project life cycle, its cost is distributed averagely. The EV (management) of one completed UCP is:
  \[
  60 \times 10\% / 20 \text{ weeks} = \$0.3
  \]
- Student 'A' reported in a performance record sheet that the design and test data for use case "X", which worths 6 UCP, is finished. The EV of this work is, because it is completed 100% of:
  \[
  6 \text{ UCP} \times 20\% \times \$60 + \frac{\text{Share Management Cost}}{3 \text{ students}} = \$72 + \frac{\$0.3}{3 \text{ students}} = \$72.1
  \]
A Use Case Based EVA Approach

"Monitoring software projects with earned value analysis and use case points", Jinhua Li et al., 7th IEEE/ACIS International Conference on Computer and Information Science, 2008

Conclusion

- This approach offers a more objective baseline measure to monitor and report progress consistently using UCP and EVA.

- Yet, to assess how much has been earned during the development of the software is rather subjective:
  - based on formulas reporting only 50% or 100% progress. (50% at starting time and 100% at completion time)
CK Degree of change used for Tracking and Software Development Progress


What?
They suggest the use of eight product measures to assist project managers to track change and progress within a release and to determine readiness.

How?
• By applying Phase Analysis, monthly measurements of eight product metrics were taken through two major releases cycles and a beta release of a commercial C++ application.
• The Chidamber and Kemerer metrics were considered among the selected eight product metrics.
• The measures represent degree of change from one month to another.
CK Degree of change used for Tracking and Software Development Progress


Their results

- 83% of classes showing profound change, also showed a period of consolidation
- They indicate that if a developer is reporting his/her task near completion, but the measures do not show the class stabilizing, the project manager should not take the assurances at the “face value”
- They conclude that the measures used are potential useful indicators of project progress during the release cycle.
Background and Related Work

CK Degree of change used for Tracking and Software Development Progress


Conclusion

• Although the suggested measures can help to monitor implementation progress and can be useful to support near completion reports from developers,

• accurate and objective measures of implementation progress at any point in time are not available.
Our Approach

Introduction

- Metrics that probably can be used cannot be assessed prior to the completion of the software (e.g. SLoC).
- A software is meant to accomplish certain specific functionalities, which are achieved through the relationship of its various components.
- If the most important components and their relationships are detailed in UML diagrams, they could be measured and used as a baseline to monitor implementation progress.
Our Approach

What?

We propose a methodology to estimate and monitor software implementation progress based on CBO measures:

- CBO measures Coupling Between Objects
- A UML approximation of the final value of CBO of the system can be obtained prior to the implementation
- CBO can be systematically measured at any point in time from code repositories
- CBO has been suggested as potential useful indicator of project progress
Our Approach

UML CBO

Class A{
    B b;
    void dosth(){
        int flag;
        flag = b.request_sh_toB();
    }
    int do_sthelse(){
        int status = 0;
        int temp;
        temp = d.request_sth_toD();
        if(a>0) status = 1;
        return(status);
    }
}

• Code CBO (A)
  Number of the different classes couple to class A, CBO (A) = 2

• UML CBO (A)
  Number of the different objects which receive any message from
  any object of class A, UCBO (A) = 2
Our Approach

UML CBO Evaluation

• We collected code CBO measures of four small-size software projects developed by students of JAIST: BNS-A, BNS-B, ECS-A, ECS-B (*)

• UML CBO measures were collected from the collaboration diagrams detailed in [1]

• Using normalization, we found that UML CBO measures closely approximated their respective code measures at the end of the implementation

Average Absolute Errors: 0.18, 0.26, 0.11 and 0.09 for the BNS-A, BNS-B, ECS-A and ECS-B respectively

(*) ECS-A, ECS-B: Are two different implementations (A and B) of an e-commerce system developed by two different groups of students of JAIST

BNS-A, BNS-B: Are two different implementations (A and B) of a banking system developed by two different groups of students of JAIST

Our Approach

CBO Code Evolution vs UML CBO

Moreover, we collected code CBO measures from the same projects FROM the day the project started TO the day was completed and contrasted them with their respective UML CBO measures.

![ECS-A: CBO Evolution](image)
Our Approach

CBO Code Evolution vs UML CBO

ECS-A: CBO Evolution

UML: UCBO (ver2)
DAY 19
DAY 20
DAY 24
DAY 25
Our Approach

CBO Code Evolution vs UML CBO

ECS-A: CBO Evolution

UML: UCBO (ver2)
DÁY 41

Class

CBO

0 2 4 6 8 10 12 14

0 1 2 3 4 5 6
How To Measure Progress?

- We observed that as the software development was progressing, its CBO code metrics closely approximate to their respective UML metrics.

- Therefore, to measure progress, we suggest an approach based on normalization and error approximations of the code CBO measures of the classes of the project to their corresponding UML CBO measures.

\[
\text{ERROR} \quad \text{PROGRESS} \quad \text{EARNED}
\]

As the error approximation of the code to its design decreases, some progress is earned.
**Our Approach**

**How To Measure Progress?**

- **STEP 1.** UML CBO measurement for the all classes (N) of the project:

  \[ UCBO[j], \ j=1 \ldots N \]

- **STEP 2.** PRODUCT VALUES of each class is measure as follows:

  \[ PV[j] = \frac{UCBO[j]}{\sum UCBO[j]} \]

- **STEP 3.** Normalization of UML measures: NUCBO[j]

- **STEP 4.** Code CBO measures at Time i (Ti): CBO_Ti[j]

- **STEP 5.** Normalization of code measures: NCBO_Ti[j]

- **STEP 6.** Calculation of Error Approximation at Ti:

  \[ \text{ERROR}_\text{Ti}[j] = \text{NUCBO}[j] - \text{NCBO}_\text{Ti}[j] \]
OUR APPROACH

How To Measure Progress?

- **STEP 7.** Earned Progress Assignation at Ti
  
  IF $\text{ERROR}_\text{Ti}[j] \geq 0.71$ THEN
  
  $\text{EP}_\text{Ti}[j] = 0$;

  IF $\text{ERROR}_\text{Ti}[j] \geq 0.5$ and $\text{ERROR}_\text{Ti}[j] < 0.71$ THEN
  
  $\text{EP}_\text{Ti}[j] = 0.1 \times \text{PV}[j]$;

  IF $\text{ERROR}_\text{Ti}[j] \geq 0.31$ and $\text{ERROR}_\text{Ti}[j] < 0.5$ THEN
  
  $\text{EP}_\text{Ti}[j] = 0.5 \times \text{PV}[j]$;

  IF $\text{ERROR}_\text{Ti}[j] < 0.31$ THEN
  
  $\text{EP}_\text{Ti}[j] = \text{PV}[j]$;

- **STEP 8.** Assessment of Implementation Progress

  Total Progress at Time $i = \sum \text{EP}_\text{Ti}[j] \times 100$
Results

Progress Percentages Patterns

Using our Approach (UCBO), CBO Progress Percentages can be estimated as soon as the implementation of the project starts.

Progress Percentages based on Bytes, LOC and code CBO cannot be obtained prior to the completion of the project.
Results

Progress Percentages Patterns

ECS-B: Progress Percentages

UCBO Baseline
Bytes
LoC
CBO

Days

%
Results

Progress Percentages Patterns

BNS-A: Progress Percentages

Days

%
Results

Progress Percentages Patterns

BNS-B: Progress Percentages

Days

%
Conclusions and Future Work

• The proposed approach effectively helped to determine when the project is near completion (in terms of CBO measures).

• Some challenges need to be addressed: to avoid sudden picks and falls of the resultant progress pattern.

• Although we cannot tell precisely to which extent our methodology is useful, we think can help mainly at monitoring and reporting progress of software implementation of first releases.