

Effort Estimation of Use Cases for Incremental Large-Scale Software Development

Parastoo Mohagheghi, Norwegian University of Science and Technology (NTNU) and Agder University College

Bente Anda, Simula Research Laboratory

Reidar Conradi, NTNU, Simula Research Laboratory

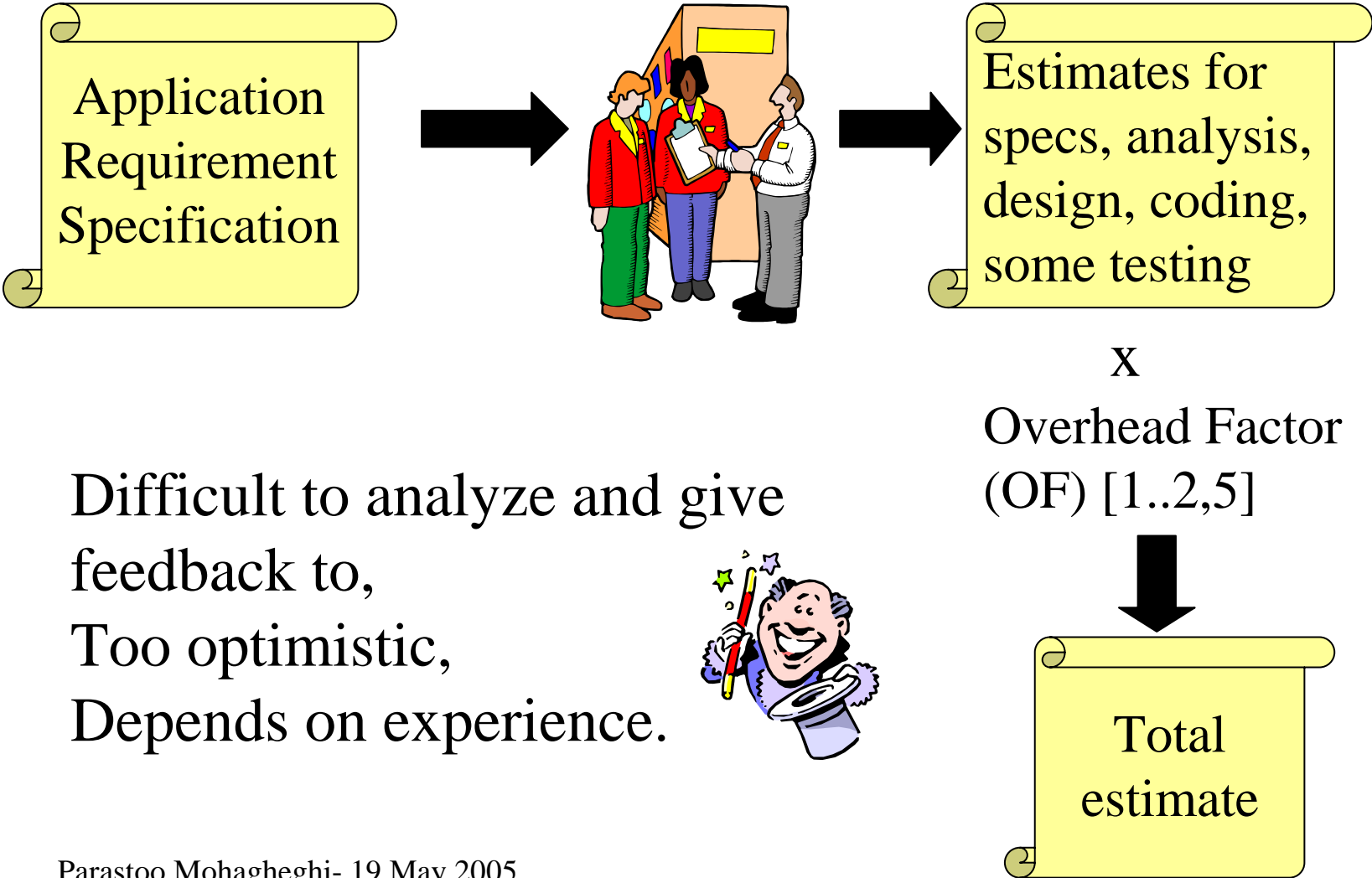
Contents

- A modification of an effort estimation method based on use cases, the Use case Points (UCP) is tested on a large industrial system.
- This presentation:
 - Why and how the estimation method is modified,
 - Estimates and comparison with actual data,
 - Lessons learned from this modification,
 - Relation to previous work and plans for future work.

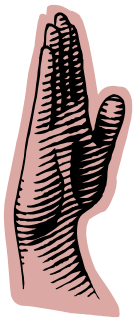
Background

- Ericsson in Grimstad-Norway has developed a large-scale telecom system (~470 Kilo Non-commented Source Lines of Code) as part of a product family:
 - Software process is an adaptation of RUP.
 - Each release typically has 5-7 iterations, each of 2-3 months.
 - Requirements, initially defined as features, are captured in use cases and supplementary specifications for non-functional requirements.
 - Up to 200 developers involved.

Current estimation method



Alternative: Top-down rule-based estimation methods



+ Organization & product specific factors



Estimated product size
in SLOC,FP, OP or UCP

Example: The Constructive Cost

Model (COCOMO): $E = A * (Size)^B$

Total
estimate
+ range

Use Case Points (UCP) estimation method

- Because estimating SLOC, FP or OP is difficult and modern systems are often developed using the UML, UML-based software sizing approaches are proposed using use cases, classes, or FP estimated from UML diagrams.
- The UCP estimation method introduced in 1993 by Karner estimates effort in person-hours based on use cases.
- We decided to evaluate the method because of our earlier experience with it and access to use cases.

Steps in UCP: 1. Estimate size of the system

1	<p>Classify actors:</p> <ul style="list-style-type: none"> a) Simple, WF = 1, b) Average, WF = 2, c) Complex, WF = 3. 	<p>Unadjusted Actor Weights (UAW) = \sum (#Actors in each group*WF)</p>
2	<p>Classify use cases:</p> <ul style="list-style-type: none"> a) Simple (<=3 transactions), WF = 5, b) Average (4-7 transactions), WF = 10, c) Complex (>7 transactions), WF= 15. 	<p>Unadjusted Use Case Weights (UUCW) = \sum (#use cases in each group*WF)</p>
3	<p>Calculate the Unadjusted UCP (UUCP).</p>	<p>UUCP = UAW + UUCW</p>

Steps in UCP: 2. Adjust to the project and 3. estimate effort.

4	Assign values to the technical and environmental factors (0..5) and multiply by weights (-1..2). Calculate the weighted sums (TFactor and EFactor). Calculate TCF and EF as shown.	<p>Technical Complexity Factor (TCF)= $0.6 + (0.01 * TFactor)$,</p> <p>Environmental Factor (EF) = $1.4 + (-0.03 * EFactor)$</p>
5	Calculate the adjusted UCP.	$UCP = UUCP * TCF * EF$
6	Estimate effort in person-hours.	$E = UCP * \text{Person-Hours per UCP (PHperUCP)}$

Comparing earlier work and this study

- Earlier studies:
 - UCP is earlier tested on relatively small projects with detailed use cases.
 - Previous projects developed software from scratch.
- This study:
 - *Much larger* product (diseconomy of scale?) and *more complex use cases with several main flows*.
 - Develops software *incrementally* (based on a previous release) and *use cases are modified between releases*.

Research questions

- **RQ1.** Does the UCP method scale up for a large industrial product?
- **RQ2.** Is it possible to apply the UCP method to incremental changes in use cases?
- **RQ3.** How can effort to modify software from a previous release be estimated?
- **RQ4.** Does the method produce reasonable results in this industrial setting?

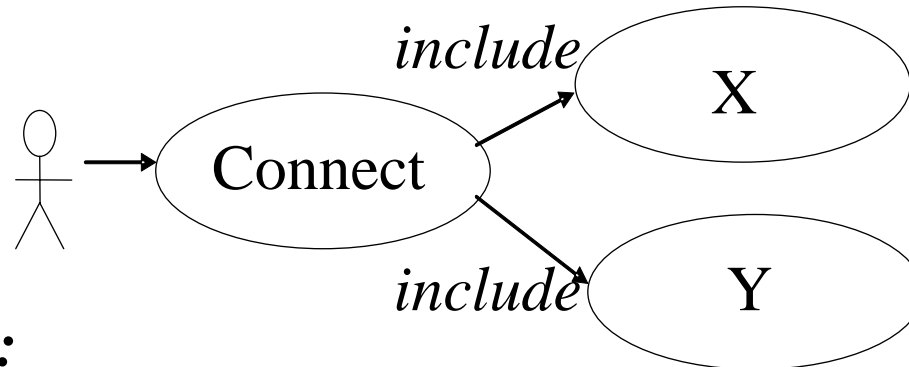
First trial with UCP on release 1

- Data from two releases were available.
- Of the 23 use cases in release 1, seven use cases were not modified, one use case was new, while 15 use cases were modified.
- All use cases were classified as Complex.

$$23 * 15 \text{ (WF)} * 36 = 12\,420 \text{ Person-Hours}$$

- Less than 10% of actual effort, even if all use cases were developed from scratch!
- > **The method must be adapted for size and complexity of use cases, and modifications in use cases.**

An example of an use case



Main Flow:

M1. Request Connection. A request message is received from user.

M2. The load of node is checked. Include X. A1 as alternative.

M3. Validate user identity:

1. The user should give one of the allowed ID type.

- 2. The ID is analyzed.**

- 3. Get necessary data from other nodes.**

3. The user is authenticated. Include Y.

Alternative flow:

A1. Too high load. A reject message is sent.

Adaptation steps (1): Size

1. Assume all actors as average complex.
2. Break down use cases in smaller ones:
 - Count each transaction in the main flow as a single use case.
 - Count each alternative flow as a single use case.
 - Count *included* and *extended* use cases as other use cases.
 - Modify weighting rules to reflect complexity.

-> **Calculate total system size (UUCP).**
3. **Calculate modified system size (MUUCP)** by counting modifications in use cases as use cases.

Back to our example

Main Flow:

M1. Request connection.

M2. Check load. Include X.

M3. Validate ID.

1. Check allowed ID.

2. Analyze ID.

3. Get data.

4. Authenticate. Include Y.

Alternative flow:

A1. Too high load. Reject.

UUCP

Actor $1*2=2$

Simple UC (M1, M2,A1)

$3*5=15$

Average UC (M3) $1*10=10$

Total 27

MUUCP

Simple UC (M3 with 2 steps)

$1*5=5$

Total for modifications 5

Adaptation steps (2): Factors

4. Assume average project $TCF=EF=1$:
 - The impact of TCF is small and it does not cover the non-functional requirements either.
 - The impact of EF may be larger but there are few changes to this factor from one release to another.
 - An adjusted Person-Hours per UCP has accounted for these factors.
5. Adjusted UCP = UUCP, Modified Adjusted UCP = MUCP.

Adaptation steps (3): Incremental development

6. Use Case Points should be multiplied by PHperUCP. There are two mechanisms that consume effort in our model:

E_primary estimates effort for realizing new and modified use cases:

$$E_{\text{primary}} = \text{MUCP} * \text{PHperUCP}$$

E_secondary estimates effort for *secondary (indirect) changes* of software:

$$E_{\text{secondary}} = (\text{UCP} - \text{MUCP}) * \mathbf{EMF} * \text{PHperUCP}$$

Equivalent Modification Factor

Equivalent Modification Factor (EMF)

- EMF counts for modification of software delivered in a previous release
 - To integrate new functionality,
 - Improve quality attributes such as availability,
 - Restructure or refactor (also to improve quality)
 - Correct bugs.
- It is calculated using COCOMO II model for software reuse:

$$\text{Equivalent SLOC} = \text{Adapted SLOC} * \text{AAM}$$

Adaptation Adjustment Modifier

Estimation results

- Release 1: 288 use cases, 77 modified
- Release 2: 254 use cases, 108 modified
- Estimated effort with 36 PHperUCP covered only development before system test. The estimated effort must be multiplied by 2 to cover all development effort (the **Overhead Factor**). Alternatively, we could use 72 PHperUCP but using OF includes the impact of effort distribution on the estimation method.
- **Estimates were 21% lower for Release 1 and 17% lower for Release 2 than the actual effort.**
- The system in this study is 20 times larger than earlier studies measured in UCP and 50 times larger in effort.

The Overhead Factor (OF)

- OF is calculated using data from the two release:

	Development before system test (specs, analysis, design, unit/integ. test, use case test)	System test	Project management	Other (CM, RUP adapt, doc, travel)
Release 1	50%	25%	10%	15%
Release 2	55%	19%	11%	15%

Discussion

- Advantages:
 - Easy to learn. I learned the method in one day. Adaptation took a couple of days.
 - Rule-based and can be adapted.
 - Does not depend on any tool.
 - Useful to produce early estimates.
 - May be used by non-technical staff or in addition to expert estimates.
- Limitations
 - Complexity assessment is rule-based, but factors are subjective.
 - Depends on high quality use cases, but can also promote this.

Our contributions to the UCP method

Generic changes for every project (modified UCP):

- Size and the level of detail in use cases,
- Technical and environmental factors.

1. Changes specific to incremental development (added to UCP):

- Use case points for modifications of use cases,
- The Equivalent Modification Factor (EMF) for building on a previous release. We have not identified similar factor in other methods.

2. Changes specific to the development organization (extended UCP): The Overhead Factor (OF).

Answers to research questions

- **RQ1.** Does the UCP method scale up? It did when we broke down the use cases.
- **RQ2.** Applicable to changes in use cases? Yes, by counting modifications in use cases.
- **RQ3.** How can effort to modify software from a previous release be estimated? We used the COCOMO II formula for adapted software and calculated EMF-
- **RQ4.** Reasonable results in this industrial setting? Fitted well into the adapted RUP process. The method is cheap and understandable.

Relation to earlier work

- Work in effort estimation for *initial* software development has been reported since the mid-sixties.
- Work has also been done to predict effort needed for *software maintenance* (unplanned changes).
- Effort estimation based on UML models (use cases, class diagrams etc.) is reported (*Fast&Serious*, Ashman, UCP), but not for incremental development.
- COCOMO assumes incremental development, but estimates effort for the whole project.

Lessons learned and conclusion

- Use cases may be used to estimate the size of software (Chen et al.). However, the method needs adaptation for large or complex products.
- Few studies on estimation in incremental development:
 - Software is now continuously modified, both planned and unplanned changes.
 - Effort needed to modify software and build on it should be better studied. EMF can be adapted to the context.

Future work and acknowledgements

- Study the effort in incremental development of software in other projects.
- Use case modification, relation to design or code
- Student experiments planned by Reidar.
- We thank Ericsson in Grimstad for data.
- The study was performed in the context of INCO project (Incremental and Component-based Software development).

Questions?