A Practical Approach for Reliable Pre-Project Effort Estimation

Carl Friedrich Kreß[1], Oliver Hummel[2] , Mahmudul Huq[1]

[1]Cost Xpert AG, Augsburg, Germany
{Carl.Friedrich.Kress,Mahmudul.Huq}@CostXpert.de
[2]Karlsruhe Institute of Technology (KIT), Germany
Hummel@KIT.edu

Abstract. Precise estimation is an important foundation for accurate project planning and staffing. However, state of practice methods (such as Function Points) typically require a thorough elicitation and analysis of system requirements, which is usually not available in an early project phase or even before a project begins. Use Case Points (UCP) are a possible solution for this dilemma and hence, in this paper we explain how a practically successful approach that combines UCP with COCOMO-based estimation can help to derive more accurate estimates. Moreover, we discuss how these estimates can be based on minimal "use case briefs" that allow estimation with less upfront effort than their "fully dressed" variant. The presented approach is thus applicable on a rudimentary understanding of requirements and helps to reduce the investment necessary for effort estimations before or soon after project initialization. Moreover, it is applicable by both, the customer as well as the IT service provider.

Keywords: Software cost estimation, COCOMO, pre-project, parametric cost estimation, Use Case Points, Function Points, project planning, project management, cost-benefit analysis, business case

1 Introduction

It is a well-established fact that every software development project should be backed up by a corporate business justification, which is a business case describing the expected financial return of the project [OGC 2009]. This is usually based on a Cost-Benefit Analysis (CBA) for which it is necessary to estimate the project's overall cost. This is directly related to the development effort that in turn depends on the functional size and complexity of the IT product and the project's environment. Consequently, a Functional Size Measurement (FSM) of the intended product is the main prerequisite for all estimates as it provides the input required for state of the art parametric estimation methods such as COCOMO II [Boehm 2000]. While there exist various other estimation approaches, such as expert judgments, which may be well applicable for small projects, they quickly reach their limits when it comes to larger undertakings. Their main weaknesses are that they are rather subjective and do not consider the diseconomies of scale that appear in large software development projects [Brooks 1995].

Current state of the art model-based estimation methods thus utilize a combination of Function Points as FSM and COCOMO II for the actual effort calculation [Boehm 2000]. From a pre-project perspective, however, Function Points are difficult, if not impossible to apply since they require a thorough understanding of the requirements and hence typically also initial analysis results such as the system operations that need to be implemented in the upcoming project. This information is usually only discovered within the project and thus Function Points are not well suited where merely a set of rudimentary requirements is available. In this research paper we propose combining COCOMO II with the relatively novel Use Case Points [Karner 1993] approach that will allow deriving comparatively more precise estimates based on the requirements of a project. Moreover, we will analyze the required minimum information content of preliminary use case briefs and sketch how Use Case Points can even be applied as a pre-project estimation method.

State of the Art and Limitations

As indicated above, it is impractical to use the proven combination of Function Points and COCOMO II in project preparation phases when merely a rudimentary description in the form of user stories or use case briefs is available. As a first step towards deriving estimates earlier, the UCP method was developed by [Karner 1993]. As it only requires use case descriptions that can be provided by business experts in early project phases, it is much more suited for this purpose than Function Points and is easier to learn even for non-technicians [Huskins et al. 2013]. However, the original UCP method proposes to directly derive effort estimates by enriching the counted number of so-called Unadjusted Use Case Points (UUCP) with project parameters (so-called cost drivers), which imposes various limitations on this approach:

- The parameters of the Use Case Points model have never been thoroughly calibrated by Karner due to a lack of available data.
- The same problem exists with Karner's so-called project productivity factor. This linear scaling factor in addition falls short of taking the non-linear diseconomies of scale into account that appear in large software development projects[1].
- Furthermore, Karner's Use Case Points rely on well formulated use cases (sometimes called "fully-dressed" [Larman 2003]), which are usually not available during project preparation.

[1] Thus Karner's model falls short of the two most important ingredients for an estimation model that can be classified as third-order model, that is the combination of a non-linear function (second order) and thoroughly calibrated linear cost drivers (third order). [Jensen et al. 2006]

In many modern iterative and incremental system development approaches (as e.g. described by [Larman 2003]), fully-dressed use cases are often refined iteratively from less formal notations called use case briefs and casual use cases. These different levels of formality can be distinguished as follows:

- Use Case – Brief: At this level of formality the use case is described in a one paragraph summary containing the main success scenario.
- Use Case – Casual: The use case is described in multiple informal paragraphs also covering alternative (such as other success and failure) scenarios.
- Use Case – Fully Dressed: Here, all scenarios of a use case are written in detail and enriched with various "metadata" (such as pre- and post-conditions, involved stakeholders etc.).

A use case brief is the degree of detail that we would normally expect from a pre-project requirement elicitation (if at all). They are typically described from a business standpoint, i.e. they treat the system as a black box and normally do not consider the interactions that are carried out by the system in its connections to other systems.

In summary, we can identify two major challenges for effort estimations on a rudimentary pre-project set of requirements, namely, first, a need for a thoroughly calibrated parametric estimation model, and, second, the necessary countable content of use cases needs must be clearly identified so that it can be captured during pre-project requirements elicitation without causing too much overhead.

Combining Use Cases and COCOMO II

We therefore suggest using a state of the art cost estimation approach like COCOMO II and aligning Use Case Points with the Function Point sizing method normally used there. This idea would allow a straightforward application of COCOMO II as presented by Boehm [Boehm 2000] with use cases, as summarized in the following:

1. Determine the unadjusted use case points and convert them into unadjusted function points.
2. Transform the function point value into equivalent SLOC (the input format of COCOMO II) by using an established backfiring table.
3. Apply the COCOMO II formula with its cost drivers and non-linear scaling factors that help capturing the project's context.

There has been some skepticism concerning the incommensurable nature of Use Case Points and Function Points [Smith 2003], also described as comparing apples with oranges [Probasco 2002]. However, this is only true concerning the initially proposed direct application of two methods for effort estimation: we believe that from a mere sizing point of view, both methods are comparable. After all, each method captures the functional size of the system in question and thus should yield the same functional quantity. Under this assumption, one can therefore utilize any FSM for application in COCOMO II by normalizing it to Function Points and in turn to Source Lines of Code (SLOC).

Applying this idea has been tested in industry for some time with good success; however, appropriate data is kept confidential for competitive reasons. To the best of our knowledge, in academia, the only publication addressing this idea was presented from [Schofield et al. 2013] very recently. They analyzed sizing data of a project collected after its completion by qualified experts and provide a first rough relation of Function Points to Use Case Points. Counting yielded a functional size of 161 Unadjusted Function Points and 71 Unadjusted Use Case Points (UUCP), which yields a preliminary normalization factor of 2.27 (that needs to be better calibrated for practical use, of course). We can therefore normalize Use Case Points to Function Points by multiplying with this value. In a next step the resulting value needs to be transformed into SLOC. In order to do so, backfiring tables as provided by [Boehm 2000] can be used. Backfiring values provide a conversion factor that describes how many lines of code (measured in SLOC) are needed to implement one Function Point. The corresponding value for Java is 53. The resulting SLOC value can then be fed into the COCOMO formula. For example, assuming a project with 250 UUCPs and Java as programming language this yields:

$$(250 \text{ UUCP} * 2.27 \text{ FP per UUCP}) * 53 \text{ SLOC} \approx 30 \text{ kSLOC} \quad (1)$$

Pre-Project Derivation of Use Case Points

Use Cases Points provide a relatively simple approach for transferring requirements into a FSM, as they merely count and weigh actors and use cases. To evaluate the complexity of the latter and thus its Use Case Point count one would have to count and weigh the number of user interactions, so-called "transactions", of human and non-human actors with the system under consideration as follows:

- 1-3 steps and less than 5 involved entities: Simple Use Case (weight: 5)
- 4-7 steps with less than 10 entities: Average Use Case (weight: 10)
- 7 steps or more and 10 or more entities: Complex Use Case (weight: 15).

In addition actors are evaluated by means of their interaction with the system:

- via APIs: Simple Actor (weight: 1)
- via protocols or the command line: Average Actor (weight: 2)
- through graphical user interfaces: Complex Actor (weight: 3).

All discovered use case points are weighted as described and then summed up to the unadjusted use case points. However, if, as discussed above, no fully-dressed use cases are available it is not possible to apply this approach without further ado.

If at least use case briefs are available during project preparation it will nevertheless still be possible to apply this approach as described in the following. As use case briefs only describe the "steps" that a human user needs to carry out, rather than all the "transactions" and the entities that occur within the underlying system, we propose the following extension for pre-project cost estimation:

- Count the use case transactions from a business standpoint as described by the use case briefs, i.e. treating the system as a black box.
- Improve the measurement by informally looking "inside" the black box to count additional Actors, that is, usually other systems which interact with the main system.
- Refine the measurement by examining those use cases that comprise 2 steps (upper limit of a simple use case) or 6 steps (upper limit of an average use case) as these use cases are close to the threshold of being weighted as average or complex respectively. Examine whether they interact with one of the systems that were identified as Actors before; if so change the complexity rating to the higher level.

Note that for a complex use case there is no possible higher rating anyway so that counting transactions can be stopped immediately when more than six transactions have been identified. Moreover, in the style of the Dutch Method for Function Points [NESMA 1997], entities are also not counted in order to further simplify the counting procedure and save effort. By following these three steps in the described order the estimation is adjusted to the time and resources available at pre-project phases. Moreover, we believe that experience should allow deriving an adjustment factor that will help capturing the additional effort caused by the ignored elements.

Once the functional size of a system under discussion has been estimated that way, it can be used as an input for the COCOMO II estimation method by applying the calculation described in the previous section. In order to yield results as precise as possible, it is finally necessary to choose appropriate values for the project's parameters (i.e. the COCOMO II cost drivers and scaling factors).

Conclusion

The presented parametric approach combining Use Case Points with COCOMO II can significantly help to simplify bids and proposals for software development projects. By feeding a SLOC value converted from use case points into the COCOMO II formula[2] one can even evaluate unfamiliar project environments by simply adjusting the

2 The USC CSSE COCOMO II Online Calculator can be used as a starting point, however the calibration data for the standard COCOMO II model does not include stratified data for different types of projects and business domains: http://csse.usc.edu/tools/COCOMOII.php

respective parameters. This is obviously beneficial in pre-project estimation and can be applied by both, the customer and the contractor. A customer would benefit from a better evaluation of bids from potential providers through creating his own cost price estimate. Hence he can check how realistic an offer is and avoid falling prey to an aggressive pricing to win strategy. On the other hand, a provider would be able to demonstrate that he is capable of conducting the project within time and budget by using the estimate derived by our proposed approach without investing too much effort into preliminary (and thus unstable requirements elicitation) and Function Point calculation. Although the proposed combination of Use Case Points and COCOMO II brings in an additional conversion stage, our practical experience from many customer projects indicates precise estimates when using a large enough calibration database to gain the conversion factor. The proposed simplified version for a pre-project application, however, although promising in theory so far, still needs to be field tested in practice so that we cannot give any quality indications yet.

References

[Boehm 2000] Barry Boehm: *Software Cost Estimation with COCOMO II*. Upper Saddle River 2000

[Brooks 1995] Frederick P. Brooks: *The Mythical Man Month. Essays on Software Engineering (2nd ed.)*. Reading, Mass. 1995

[Huskins et al. 2013] Michael Huskins, James Kaplan and Krish Krishnakanthan: *Enhancing the efficiency and effectiveness of application development*. In: *McKinsey Quarterly*, August 2013, URL: http://goo.gl/BWCpwt (last accessed: December 2013)

[Jensen et al. 2006] Randall W. Jensen, Alan W. Armentrout, Regina M. Trujillo: *Software Estimating Models: Three Viewpoints*. In: *CrossTalk*, February 2006, URL: http://goo.gl/ACfzDk (last accessed: December 2013)

[Karner 1993] Gustav Karner: *Metrics for Objectory*. Diploma thesis, Linköping University 1993.

[NESMA 1997] Netherlands Software Metrics Association: *Counting Guidelines for the Application of Function Point Analysis*, 1997.

[OGC 2009] Office of Government Commerce: *Managing successful projects with PRINCE2*, London 2009, section 2.1 *Continued business justification*

[Probasco 2002] Leslee Probasco: *Dear Dr. Use Case: What About Function Points and Use Cases?*, URL: http://goo.gl/lwkAVX (last accessed: December 2013)

[Schofield et al. 2013] Joe Schofield et al.: *Function Points, Use Case Points, Story Points: Observations from a Case Study*. In: *CrossTalk*, May/June 2013 URL: http://goo.gl/il44Kw (last accessed: December 2013)

[Smith 2003] John Smith: *The Estimation of Effort based on Use Cases*, IBM / Rational White Paper, 2003, URL: http://goo.gl/4c5B1E (last accessed: December 2013)