

Use Case Points Effort Estimation Based on Different Specification Formats

Stephan Frohnhoff, Karsten Kehler
sd&m AG, Berliner Str. 76, D-63065 Offenbach
frohnhoff@sdm.de

Summary:

The Use Case Point method (UCP method) allows early, easy estimation of the anticipated effort during a software development project. The basis for such an estimation in real industrial cases is frequently a number of rough specifications in different formats and of differing granularity. The success of the UCP method and comparability of the results depends above all on whether, and how, it is possible to map the existing specification to use cases in the sense of the UCP method. This article summarizes a suggested guideline that has been derived from real software specification cases by the software developer sd&m.

Keywords

project estimation, top-down estimation, effort estimation, use case points, UCP, specification

1 Introduction

The Use Case Point method (UCP method) has already been described in detail in [1]. It is a top-down estimation method that classifies the functional requirements of the specification into countable units (use cases), to which points are assigned (use case points) according to their estimated complexity. The estimated project effort is then proportional to these use case points. An important point of criticism about this method is that use cases can be described in different granularity, which could directly influence the UCP estimation result.

Based on the projects described in [1], we have drawn up a guideline on how to “cut up” use cases for the purpose of estimating the functional scope using the UCP method. This comprehensive document shall be published as a preprint [2]. The following article gives a summary of the most important findings. It is not the aim of this article to define a uniform approach to describing use cases.

The reference material for this article are software development projects for company information systems developed by the software house sd&m AG as service provider for clients from different industries. The effort estimations are based on different rough specifications given by the clients and follow no uniform format; these can often be UML-like descriptions or purely text specifications.

2 Terminology

As a uniform definition of its level of complexity, we shall evaluate a use case as either easy, medium or complex according to the number of its *scenarios*, *steps* and *dialogs*. These definitions are as follows:

Easy: maximum 3 scenarios, steps and dialogs => 5 points

Medium: maximum 7 scenarios, steps and dialogs => 10 points

Complex: at least 8 scenarios, steps or dialogs => 15 points

The total points over all such weighted use cases are then summed up as an unweighted UCP measure (i.e. without accounting for the complexity factors). See [1] for a detailed illustration of this.

2.1 Use Case

We use the following definition as our basis: a use case itemizes precisely one activity out of the system-supporting activities of a business process. It describes the behaviour and interaction of a system as the reaction to a specific query or action of an actor. The description of the use case explains both the external, visible system behaviour and the detailed, internal system behaviour in the user's language and from the user's perspective. Where there is a use case, you know that the system performs a useful service for the user or achieves a useful result.

A crucial point here is that a specific **functional target** can always be achieved with a use case. Wherever this target is not achieved, one can only talk of application functions or steps of a use case. These could be, for example, all function-oriented descriptions or even dialog-oriented descriptions. For example, using a search dialog is most often not a separate, concluded use case because the actual functional target has not yet been achieved when the hits are found and shown, since there will still be further actions taken on a certain record after the search.

2.2 Scenarios

A use case scenario is the functional sequence of actions performed by the actors and system in an error-free success case, where the functional target of the use case is achieved. As a rule, every use case has just such a success scenario.

Besides the success scenario, there are **alternative scenarios** that can be specified. These describe separate, special functional cases; so if there are ever special conditions or exceptional cases, these are the scenarios in which the use case is still successfully concluded.

2.3 Steps

What is decisive when weighting a use case is the number of steps required to reach the functional target. Unfortunately, there is no generally recognized definition of these steps either; these are frequently referred to as actions, operations or sequences of operations or activities. System-side actions are also referred to as application functions, especially if the description already includes a certain functional decomposition of the use case.

A **step** in the process of a use case is a self-contained functional part of the use case, which is clearly distinct from the next step and previous step, either upon a change in actor or the processing “layer” (e.g. Input into dialog by user => Processing of input on server => Display of result at interface), or upon the production of a defined (intermediate) result (e.g. creation of print documents), or upon splitting up a new scenario.

In use case modelling, it is important to look always at the top level of the breakdown into steps from the perspective of the use case. Every additional hierarchical refinement is then no longer relevant for estimation by UCP. So when defining a use case, if there are certain activities described that in turn consist of operations, then only the activities themselves are counted here. When counting, one must take into account:

- The number of all steps in all scenarios; this is a summation over all scenarios, where any steps used in more than one scenario are only counted once
- And the same goes for user actions (steps) and system-side actions also

Typical **examples** of steps would be, say, inputting one or more values into a dialog (without this leading to a server roundtrip), calling application functions, server transactions, performing functional tests, or creating documents. An actor making a trivial selection from a display would not be counted as a separate functional step (e.g. “select address”), although this could lead to new scenarios depending on the selection.

When defining steps, things get a little different when there are very complex processes that have to be processed in a single step according to the abovementioned criteria. One runs the risk of underestimating such steps if one defines them as described above. These would include, for example, tests performed in a single step, complex calculations, or perhaps even the creation/modification of a large number of entities. This category also includes extensive bodies of rules or even functional or decision matrices. In such cases, we suggest comparative weighting against other (“classical”) use cases from the project, in order to break them down into smaller, individual steps that produce functionally delimitable partial results.

2.4 Dialogs

Another parameter that goes into the estimation is the number of different dialogs in a use case. For the purpose of UCP estimation, **dialog** is the broad term used for every interaction interface. Dialogs are counted as follows:

- Every tab or mask of a dialog (with significant functional differences) is counted as a separate dialog,
- Every frame of a website (with significant control elements) is counted as a separate dialog,
- Trivial pop-up messages, confirmations and menus are not counted

In addition to such “classical” dialogs, the following interaction interfaces are also counted as dialogs:

- Complex interfaces that are made available to neighbouring systems (in addition to processing)
- Printouts and other specially created, formatted documents

3 Insights into "Cutting Up" Use Cases

The following sections deal with the rules for “cutting” and as such structuring use cases for the purpose of applying the UCP method. The aim is to ensure a standardized and uniform applicability of the UCP method for independent evaluators. These rules have been drawn from interviews with about 20 experienced project managers and chief architects on various projects. These insights have then been verified by estimations with specifications of all different kinds (use cases, dialogs, textual descriptions and mixed variants). There has yet to be extensive statistical verification of these rules, in particular with regard to the intended reproducibility of the use case point counting.

3.1 Granularity

Finding the right cut for the use case is a crucial part of the estimation. It is not possible, however, to give an exact measure for a balanced use case description, so we shall provide some guides here.

The size of the descriptions in question in the specification document can give an indication of whether the use cases are too complex (or too simple). We have various criteria that can give you an idea of whether your chosen scenario is too extensive:

- The textual description of a scenario covers more than one DIN A4 page, **or**
- A scenario contains more than 12 steps, **or**
- A use case includes more than seven branchings/scenarios. In this case, when you consider the weighting, the individual scenarios should preferably be treated as separate use cases.

If a use case significantly exceeds these values, then it is too extensive for the estimation and should be broken down into smaller components. These values are drawn from practical project experience and, as with all given limits, should not be blindly followed in every specific case, rather checked against your own experience or other use cases.

Very small use cases, i.e. those with a very small number of steps, scenarios and dialogs, are an indication of the wrong choice of “cut”. Signs for this would be:

- The description of a use case is only a few lines long (careful: in a rough specification, this can very frequently be the case if, say, a lot of steps are simply listed out in one sentence; in which case the choice of cut could still be correct)
- The use case does not have any dialogs **and**
- The use case has only one scenario **and**
- The use case has only one or two steps

In this case, one should reconsider whether one is in fact dealing with an application function or a sub-step of a higher-level use case (e.g. “Search”).

One must nevertheless bear in mind that “runaways” are allowed as a rule. Again, a useful measure would be: approximately 80 to 90 percent of all existing use cases should be kept within these limits. The rest are exceptions that are either very large or very small, but which “fit in” with the others in terms of the cutting. Overall, a balanced distribution of small, medium or large use cases is a sign of a “good” cut. For the reproducibility and consistency of an estimation, it is crucial for the criteria for finding use cases and steps to be uniform within the individual estimation and for different estimations. It is therefore recommended to compare the “cut” of use cases and steps repeatedly during a UCP estimation.

3.2 Accounting for “Hidden” Functionality

It is very often the case that, in addition to the pure functionality, additional “secondary” functionality is described in separate chapters of the requirements specification, or is required at any rate. This could be, for example, clean-up runs, migration programs, administrative functions, or management and configuration interfaces.

This functionality must of course also be factored in. This could be a use case such as: “nightly correction of account differences”, “migration of address data”, or “configuration of metadata”.

What is important is that all specified (and implicit) application functionality must be fully covered by the created use case.

3.3 Calling Other Use Cases

The use case method for specification of an application always allows you to define a use case call tree by creating a hierarchical order. For the sake of a good, readable specification, branches to individual scenarios or steps of a different use case are forbidden. The calling use case makes sure the preconditions of the called use case are fulfilled, and that the entire (sub) use case is run through. Once the called use case has been run through completely, its result must be valid. Then, the calling use case continues its processing.

The conditions for using such called use cases in the UCP method can be relatively easily stated as follows:

- Calling the subordinate use case and creating the preconditions for it counts as one step for the calling use case
- The scenarios, steps and dialogs of the called use case are only counted within that use case, and not in the calling use case (and are thus only counted once in the overall scheme of the whole application)

If it turns out that use case call trees have to be used extensively in a specification, this could be (but is not necessarily) a sign that the “cut” of the use case is still not right for that particular estimation.

3.4 Handling Batch Functionality

Quite often, complex server-side functions or even batch processes that are performed without any interaction with other processes or actors are not specified as classical use cases. But these often *are* use cases: they have a definite start with preconditions (which are often even easier to determine here than for other use cases); there are process scenarios; and there is a specific functional target.

The challenge in these cases is to break down the process into the right sized individual steps for the estimation. In principle, the same advice applies for all steps here as to other use cases.

- If very complex: break down into sub-use cases according to functional blocks
- Steps can also be found based on functional blocks, access to interfaces or the processed entities
- An individual step could in this case have greater power (=covered functionality) than a step in a non-batch use case, since as a rule there will be no actor intervention, for example, and thus no complex interface or level change within the application architecture
- The general rule is that the steps within a use case should be approximately the same size across all use cases in an estimation and (ideally) across all UCP estimations.

If such a breakdown has been found for a batch use case, it will be counted just the same as a “normal” use case, i.e. scenarios, steps and dialogs will be created (in this case more likely in the form of printouts or complex output files).

4 Evaluation of Ways to Describe Use Cases for the UCP Method

There are various different methods for documenting use cases that have proven successful in practice. These have been investigated in many projects, although the experience from this can only be summarized here. Specifications that exist in the form of flow charts or activity diagrams (UML) are very suitable.

Figure 1 shows the use case “Create address” very well as a flow chart. The different scenarios, dialogs and steps are represented as colour coded and numbered captions. It must be noted that displays (dialogs) must also be counted as steps here; see for example the *suggestion list*: Dialog 2 = Step 4. Splitting in the case alternatives leads to new scenarios according to the number of new branch-offs (no combinations with previous scenarios). Trivial error scenarios can be illustrated as separate scenarios with steps, but should not be counted as such.

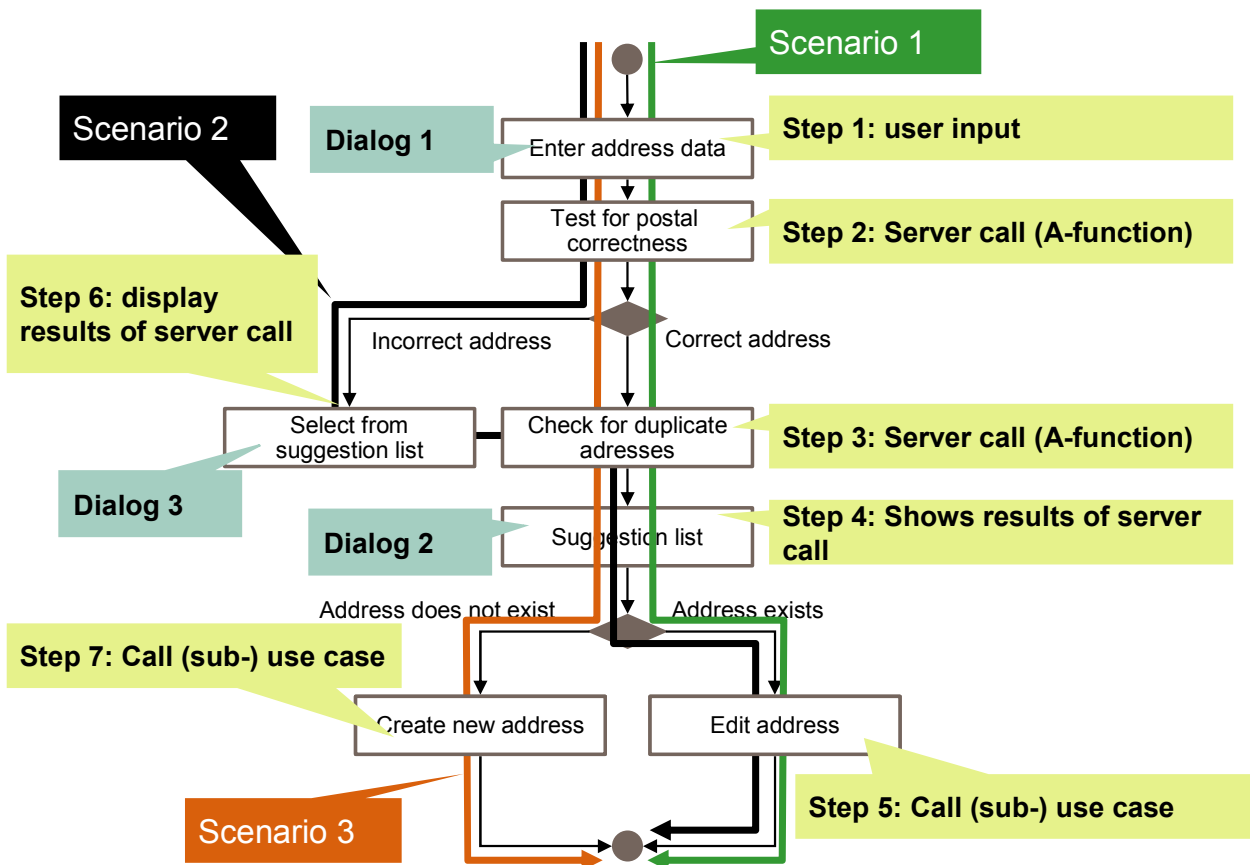


Figure 1: Flow-diagram as an example for UCP counting

We thus determine 3 scenarios, 3 dialogs and 7 steps. In accordance with Chapter 2, it follows that this use case is of medium complexity, i.e. this use case is to be awarded 10 points, unweighted.

Here, we have described UCP counting using the example of a flow chart. Other types of specification and their “counting” can be treated analogously for the UCP method.

A detailed illustration of this would go beyond the scope of this article, and is given in [2] instead. The result of this analysis, however, is summarized in Table 1.

Type of specification	Suitability for the UCP method	Advantages	Risks
Activity diagram or flow chart	Very good	Direct adoption of use cases and counting of scenarios and steps; quick and easy	Risk of neglecting to check against the textual description; Activity diagrams can be too fine
Textual or tabular description	Very good	List-outs can be directly adopted, scenarios and steps are already described	
Rough (textural) description	Good	List-outs can be directly adopted, scenarios and steps are already roughly described	Underestimation due to too little information on scenarios/steps; underestimation due to forgotten entire use cases
Dialog description	Good	Use cases and scenarios are relatively easy to find, good coverage of the application	Hidden steps; application logic is not described, therefore always observe additional texts
Sequence diagram	Good if all use cases are described, otherwise average	Cut for use case already given; cuts and scenarios clearly shown	Danger of too fine granularity; usually incomplete coverage of the application; therefore recommended to compare with additional texts
Functional description	Good, depending on content	Description of the steps and functions, good coverage of the required functionality of the application	Composition into scenarios and use cases can be difficult in certain circumstances; risk of estimating in terms of functions and therefore in too much detail
State diagram	Average		Different "cut" by the application; use cases overlooked, only useful in conjunction with additional text!
CRUD diagram (Create/Read/Update/Delete)	Average	Most use cases have already been found	Steps still have to be found; use cases overlooked; only recommended in conjunction with additional text
Business processes as rough specification	Average, depending on level of detail	List-out of use cases can be used directly or easily integrated, good overview of the application	Underestimation due to too little information on scenarios/steps

Table 1: Evaluation of different types of specification for the UCP method

5 Conclusion and Outlook

With the help of this rough guideline, UCP estimations can be performed based on requirement specifications of different kinds and granularities. This guideline provides large organizations with a foundation upon which different estimators can make their estimations according to clear rules, thus ensuring that all estimations are comparable. The effort involved in introducing the method described here into an organization such as sd&m, with experienced software engineers, is reduced to a study of this guideline. The method is just as easily applied at other software companies.

The success of the UCP method and comparability of the results depends above all on whether, and how, it is possible to illustrate the existing specification using the kinds of use cases described here. Ideally, the specification meets the mentioned requirements, so that there is only the need for quantitative evaluation (weighting + counting).

The estimation requires no additional detailing of the use case, and it can and should be done at a purely functional level. That way, a rough specification that only gives a list-out and briefly names these components could be an even better basis for a UCP estimation than a detailed description (because it is easier to record). In general, flow charts, activity diagrams and textual or tabular descriptions are very suitable. However, good results have also been achieved with dialog descriptions for company information systems.

If there is any difficulty in finding or evaluating the use case, that generally indicates that there is not enough information for estimating the functional scope, and in such cases the UCP method cannot deliver any reliable values.

Each different type of specification described in this guideline has been evaluated on the basis of two to three expert appraisals. The next step should be to investigate the reproducibility of the different UCP countings when following this guideline, as a function of the type of specification investigated, in that a larger group of estimators each perform a UCP estimation independently. Low statistical variance of the use case points would then be a measure for a specification's suitability for the UCP method.

Literature

1. Frohnhoff, S.; Jung, V.; Engels, G.: "Use Case Points in der industriellen Praxis" In "Applied Software Measurement - Proceedings of the International Workshop on Software Metrics and DASMA Software Metrik Kongress", Abran, A. et al. Eds. Shaker Verlag, 2006, pp. 511-526.
2. Frohnhoff, S.; Kehler, K., Dumke, R.: "Leitfaden zum Finden von Anwendungsfällen für die Use Case Points Methode", Preprint, Fakultät für Informatik, Universität Magdeburg, 2007