

Software Metrics & Software Metrology

Alain Abran

Chapter 9 **Use Case Points: Analysis of their Design**


Agenda

This chapter covers:

- Overview of the Use Case Points (UCP):
 - origins & initial design.
- Analysis of the design of the UCP, including:
 - the entities and attributes measured,
 - the scale types
 - measurement units.
- Related work on the UCP relationships with development effort

Agenda

This chapter covers:

- Overview of the Use Case Points (UCP): 
 - origins & initial design.
- Analysis of the design of the UCP, including:
 - the entities and attributes measured,
 - the scale types
 - measurement units.
- Related work on the UCP relationships with development effort

Introduction

- With the increase in the popularity of development methodologies based on use cases (e.g. UML, Agile, RUP) has come a concomitant interest in effort estimation based on Use Case Points (UCP).
- The UCP sizing method was initially designed by Gustav Karner of the company Objectory AB in 1993.
 - UCP is an adaptation of FP for measuring the size of projects, the functional requirements specifications of which are described by a use-case model.

Use Case Points Description

Origins

- The use-case approach was developed by Ivar Jacobson [1987] while at Ericsson in the 1960s:
 - A use case is a technique for capturing the requirements of a software system.
 - Each use case is composed of a number of *scenarios*, written in an informal manner in the language of the business domain to describe the interactions between the actors and a system.

- In 1993, Gustav Karner proposed the UCP measurement method, an adaptation of Function Points, to measure the size of software developed with the use-case approach:
 - aimed at measuring software functional size as early as possible in the development cycle.

Use Case Points Description

Initial Design of UCP

- The initial design of UCP takes 3 aspects of a software project into account:
 - Use cases,
 - Technical qualities, and
 - Development resources.

- **1) Unadjusted Use Case Points – UUCP**
 - Use Cases are represented by a number of Unadjusted Use Case Points (UUCP).
 - Each actor & each use case is:
 - classified at a complexity level (simple, average, or complex) and
 - assigned a corresponding weight
 - from 1 to 3 points for the actors, and from 5 to 15 for the use cases.

Use Case Points Description

■ 2) Technical qualities

- The technical qualities of UCP are represented by a Technical Complexity Factor (TCF), which consists of 13 technical qualities, each with a specific weight, combined into a single factor.
 - To calculate the TCF: an expert must assess the relevance to the project of each technical quality, evaluated on a scale from 0 to 5 (where 0 is 'not applicable' and 5 is 'essential').
 - The weights are *balanced* in such a way that a relevant factor of 3 for each quality will produce a TCF equal to 1.
- The TCF is thus the sum of all the relevant factors (one for each technical quality) multiplied by their corresponding weights + 2 constants:
 - C2 (0.1) and C1 (0.6)
- Karner based his design for these weights on the constants and weights of the 1979 Function Points Value Adjustment Factors.

Use Case Points Description

■ 3) Development resources

- The development resources for UCP are represented by the Environment Factors (EF), also referred to as experience factors [Carroll 2005].
 - The UCP model identifies 8 factors contributing to the effectiveness of the development team.
- To calculate the EF, an expert must assess the importance of each factor and classify it on a scale from 0 to 5 (0 meaning 'very weak'; 5 meaning 'very strong').
- The selection of the weights is balanced such that a value of 3 for each factor will produce an EF of 1.
- The EF is the sum of all the factors multiplied by their weights and two constants, C2 (0.03) and C1 (1.4).

■ 4) Total UCP size


- The final size in terms of the number of UCP is the product of these three components:

$$\text{Total UCP size} = \text{UUCP} \times \text{TCTP} \times \text{EF}$$

- For estimation purposes, the number of UCP is combined with a productivity constant (referred to as Mean Resources – MR) representing the ratio of man-hours per UCP.
- Karner proposed that each organization require its own productivity constant.
- The MR constant for the Objectory projects in Karner's paper = approximately 20.

Agenda

This chapter covers:

- Overview of the Use Case Points (UCP):
 - origins & initial design.
 - Analysis of the design of the UCP, including:
 - **the entities and attributes measured,**
 - the scale types
 - measurement units.
 - Related work on the UCP relationships with development effort
- 

Analysis of the UCP Design

The measurand: the entities and the attributes measured

- UCP is aimed at measuring the functional size of a software system described by a functional requirement specification written in use-case form.
- However, the UCP design includes a measurement of 5 distinct types of entities:
 - Actor
 - Use case
 - Specification of requirements (functional and non functional)
 - Development team (a project-related entity)
 - Programming language (a project-related entity)
- For this set of entities, 11 different types of attributes are taken into account and measured by the UCP method described in [Carroll 1993] – see Figure 1 and Table 1.

Analysis of the UCP Design

Table 1: Entities, Attributes, and Measurement Rules

Entity	Attribute	Measurement rule
Actor	Complexity (of actor)	The type of complexity (simple, average, or complex) of the interaction between the actor and the system
Use case	Complexity (of use case)	The type of complexity (simple, average, or complex) measured in the number of transactions
Specification of requirements	Relevance of the technical quality requirements	The level of relevance (from 0 to 5) of each of the 13 known non-functional qualities
	Stability of the requirements	The level of stability (from 0 to 5) of the functional and non-functional requirements
Development team	Familiarity with the methodology	The level (from 0 to 5) of skills and knowledge of the development methodology in use for the project.
	Part-time status	The level (from 0 to 5) of part-time staff on the team
	Analysis capability	The level (from 0 to 5) of analysis capabilities of the development team with respect to project needs
	Application experience	The level (from 0 to 5) of team experience with the application domain of the system
	Object-oriented experience	The level (from 0 to 5) of team experience with object-oriented design
	Motivation	The level (from 0 to 5) of team motivation
Programming language	Difficulty	The level (from 0 to 5) of programming difficulty

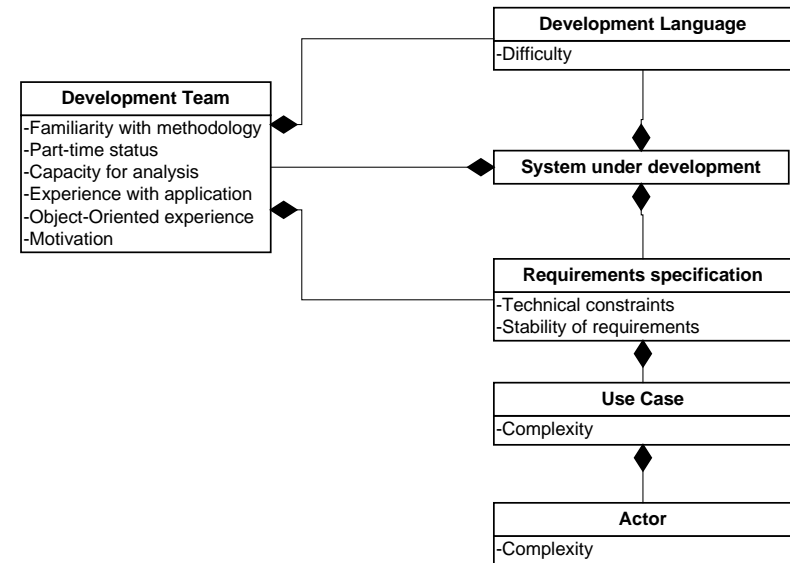



Figure 1:
The 5 types of entities &
11 types of attributes
measured in UCP

Agenda

This chapter covers:

- Overview of the Use Case Points (UCP):
 - origins & initial design.
- Analysis of the design of the UCP, including:
 - the entities and attributes measured,
 - **the scale types** 
 - measurement units
 - Measurement method
- Related work on the UCP relationships with development effort

Analysis of the UCP Design

Analysis of the numerical world: Issues in scale types

- These 11 types of attributes are quantified using a variety of scale types, and then they are combined.
- The ‘complexity’ attribute, assigned to actors & use cases, is at the core of UCP (collectively defined as the UUCP factor).
 - This complexity is categorized as being of the ordinal scale type using a scale of 3 values: simple, average, and complex.
 - Thus an actor categorized by the measurer as ‘simple’ is considered less complex than an ‘average’ actor, and an average actor less complex than a ‘complex’ actor.
 - The scale is similar for use cases: the same category labels are used (simple, average, and complex)
 - However, it cannot be assumed that the categories and the categorization process are similar, since different entity types are involved.

Analysis of the UCP Design

- The technical and resource factors are also all evaluated through a categorization process with integers from 0 to 5 (0, 1, 2, 3, 4, or 5):
 - It must be noted that these numbers do not represent numerical values on a ratio scale, but merely a category on an ordinal scale type:
 - that is, **they are merely ordered labels and not numbers.**

Measurement of a Programming Language

- A programming language assigned a difficulty level of 1:
 - is considered to be less difficult for the development team than a programming language of difficulty level 2,
 - but cannot be considered to be exactly 1 unit less difficult than one categorized as having a difficulty level of 2, because these levels are being measured on an ordinal scale.

Analysis of the UCP Design


- In fact, there is no justification provided in the description of the UCP model to support a ratio scale:
 - for example: nothing in the UCP design allows room to consider that a programming language of factor 4 is twice as difficult as a programming language of factor 2.

Levels in the quantification of the Attribute

- In the UCP design, each label for a level – see Table 1 – might represent a different interval, and each interval may not be, and does not need to be, regular within an attribute being measured.
 - Ex.: the measurement rules for the Value Adjustment Factor of Function Points in chapter 8, section 3.5.
- It should be noted that Karner had indicated that the EF constants and weights were preliminary and estimated:
 - more recent sources have not revisited this issue and have integrated the same values without modification into their own measurement models of UCP variants.

Agenda

This chapter covers:

- Overview of the Use Case Points (UCP):
 - origins & initial design.
- Analysis of the design of the UCP, including:
 - the entities and attributes measured,
 - the scale types
 - **measurement units** 
 - measurement method
- Related work on the UCP relationships with development effort

Analysis of the UCP Design

■ Issues in measurement units

- In measurement, each distinct type of attribute should have its corresponding measurement unit.
 - This is not the case with UCP, where each type of attribute is assigned some 'points' of an unspecified nature and for which the measurement unit is not described.
- The UCP measurement process clearly includes a mix of several entities (actors, use cases, requirements, team, programming language) for which distinct attributes are quantified & then combined:
 - It is obvious that the end-result cannot be expressed solely in terms of use cases.
 - The resulting units of measurement (i.e. UCP points) are of an unknown and unspecified nature, despite the UCP label.

Agenda

This chapter covers:

- Overview of the Use Case Points (UCP):
 - origins & initial design.
- Analysis of the design of the UCP, including:
 - the entities and attributes measured,
 - the scale types
 - measurement units
 - **measurement method**
- Related work on the UCP relationships with development effort



Analysis of the UCP Design

Issues in the measurement method

- Without a detailed measurement method for identifying the entities and attributes to be measured, the UCP measurement process also lacks strict repeatability, or even reproducibility

■ Use cases

- EX: the fairly flexible definition of a use case poses an important problem for the measurement of UCP [Smith 1999].
 - The way in which a use case is written is not standardized: a use case can be identified and documented at any level of granularity.
 - The UCP method does not take this into account and does not describe any means to ensure the consistency of granularity from one use case to another.
 - The impact is the following: quantities as the outcome of UCP are not necessarily comparable, and a poor basis for benchmarking and estimation.
- This raise a number of consistency questions with respect to the inputs to the measurement process, such as:
 - What is an elementary use case?
 - How can the level of granularity of a particular use case even be characterized?
 - Are use cases being characterized in a consistent manner by one or more analysts working on the same project, and across projects?

Analysis of the UCP Design

■ Technical and resource factors

- Both the technical factors & the resources are evaluated qualitatively by a measurer.

■ Technical qualities

- The UCP model does not propose any criteria for measuring the attribute values of technical qualities. The result is a subjective interpretation by a measurer.
 - To address the same kind of arbitrary assignment of values in the ‘technical adjustment factors’ of the FP model, the IFPUG organization worked out detailed evaluation criteria for each of the non functional characteristics of the system and documented these in its Counting Practices Manual.

■ Resource factors

- The UCP model does not propose any criteria for measuring resource factors.
 - Once again, this measurement is a subjective interpretation by a measurer.
 - The UCP method does not prescribe any rule for assigning a value to, for example, the level of familiarity with a methodology.

Analysis of the UCP Design

Relations between the empirical & numerical worlds

- The principle of homomorphism requires that the integrity of attribute ratios and relationships be maintained when translating from the empirical model to the numerical model.

- **Actor complexity:** The measurement procedure for assigning a value to the complexity attributes of actors is based on categorization rules.

Quantifying an Actor:

If an actor acted on the system by means of a graphical user interface, then the value assigned to it is labeled 'complex'.

- this value is on an ordinal scale.
 - If we accept that an application programming interface (API) represents less functionality than a command-line interface, which in turn represents less functionality than a graphical user interface, then it can be said that the categorization of the actors by their type of interface constitutes a valid correspondence.
 - But if, after assigning a weight of 1, 2, or 3 to the actor types, the model then uses these ordered values in sums and products, it is effectively making a translation from an ordinal scale (complexity categories) to a ratio scale (UCP weights) without justification, which is not a mathematically valid operation.
 - Furthermore, there are no documented data to demonstrate that a graphical user interface represents 3 times more 'functionality' than an application programming interface.

Analysis of the UCP Design

■ Use-case complexity:

- For use-case, the measurement process for assigning a value comes:
 - first, from counting the number of transactions or the number of analysis classes,
 - and then looking up the correspondence rules in a 2-dimensional table (transactions & analysis classes) to assign a corresponding ordered label of an ordinal-type scale.
- Ex: if a use case contains 4 transactions, it is assigned the category label 'simple'.
- This rule transforms the measurements of use-case complexity from a ratio-type scale (# of transactions or classes of analysis) into an ordinal-type scale (complexity categories), and then moves it up to a ratio-type scale (UCP weights).
 - The arbitrary assignment of the weights (5 for simple, 10 for average, and 15 for complex) could have been avoided if the number of transactions or classes of analysis had been kept as numbers on a ratio-type scale
 - (rather than losing this quantitative precision by mapping them to only 3 ordered categories with arbitrarily assigned values of 5, 10, and 15).
 - Both the constants and weights of UCP's technical quality calculation (TCF) are derived directly from the Albrecht's Function Points model
 - See Chapter 8 for a discussion on the related weaknesses in Function Points.

Analysis of the UCP Design

Inadmissible numerical operations

- The formula to calculate the number of UCP is as follows:

$$\text{UCP} = \text{UUCP} * \text{TCF} * \text{EF}$$

- The UUCP value is calculated by multiplying the number of use cases & actors of each type of complexity by their weights.
 - In doing so, the ordinal-scale values (categories of complexity) are transformed into interval-scale values.
 - These values on an interval-scale type are next multiplied, resulting in a value on a ratio scale, another algebraically inadmissible mathematical operation.
- This analysis applies equally to TCF and EF factor calculations, which, in addition to the transformations of ordinal-type scale into interval-type scale (confounding numerical values with the ordering of categories by numerical label), also introduce multiplications with ratio-scale constants.

Analysis of the UCP Design

- It was further pointed out in chapter 8 that the interaction between the technical factors would have been better accounted for by multiplying the factors together, rather than adding them. This analysis also applies for the EF factors.
 - *In summary*, TCF has inherited most of the defects of Function Points, and in some cases has compounded them by adding more inadmissible mathematical operations.

- The UCP measurement method assigns numbers to several entities (actors, use cases, requirements, etc.) and attributes (complexity, difficulty, etc.). Combining many concepts at once, as this method does, makes it a challenge to figure out what the end-result is from a measurement perspective.
 - The impact of this is that the end-result is of an unknown and unspecified entity type; that is, what has been measured is not known.
 - The UCP measurement method is based on the constants and weights of Albrecht's FP Value Adjustment Factors without supporting justification.
 - The evaluation of attributes is performed by a measurer without criteria or a guide to interpretation – a 5 for one measurer could be a 2 for another. This could result in very poor repeatability and reproducibility.
 - UCP method calculations are based on several algebraically inadmissible scale type transformations. It is unfortunate that this has not yet been challenged, either by UCP users or the designers of subsequent UCP variants.

Analysis of the UCP Design

“The measurement of functional size using use cases (therefore, very early in the development life cycle) constitutes an interesting opportunity for an industry which is increasingly using RUP and Agile use-case-oriented methodologies”.

“Unfortunately, the UCP method appears fundamentally defective: by adopting the basic structure of Function Points, UCP has – from a measurement perspective – inherited most of its structural defects”.

Agenda

This chapter covers:

- Overview of the Use Case Points (UCP):
 - origins & initial design.
- Analysis of the design of the UCP, including:
 - the entities and attributes measured,
 - the scale types
 - measurement units
 - measurement method
- **Related work on the UCP relationships with development effort**



The Modeling of Relationships of UCP with Project Effort

Analysis of 3 studies on the use of variants of UCP to analyze the relationship between UCP and project effort.

■ Nageswaran

- On the basis of a single case study, [Nageswaran 2001] reports that the UCP method can be more reliable than an estimation model with Function Points to predict testing effort.
- The approach described in [Nageswaran 2001] is a variant of Karner's, proposing:
 - 9 technical qualities (instead of 13) associated with testing activities (for example, test tools and test environment), and
 - ignoring the development resource factors (EF).
- For the single project studied, the effort estimated by the UCP method was reported to be only 6% lower than the actual effort.
- The author himself stresses that this “*estimation technique is not claimed to be rigorous.*”
- Of course, this single case study lacks generalization power.

The Modeling of Relationships of UCP with Project Effort

■ Mohagheghi

- The adapted UCP method described in [Mohagheghi 2005] suggests that iterative projects need special rules to account for the ongoing re-estimation of changing requirements during the course of a project.
 - Replaces the resource factor (EF) by a simple increase in the productivity constant (MR).
 - An overhead factor, is introduced to represent the effort of project management and other activities not directly related to functionality.
- For the 2 projects presented in [Mohagheghi 2005], the efforts estimated by UCP were 21% and 17% lower than the actual effort.
- Again, this study refers to only 2 projects and lacks generalization power.

The Modeling of Relationships of UCP with Project Effort

■ Carroll [Carroll 2005]

- In essence, the same as Karner's, but with the addition of a new 'risk coefficient', specific to each project, to account for risk factors not already covered by the basic model (for instance, 'reuse effort').
 - The risk coefficient is a simple percentage of effort added according to the opinion of an estimation expert.
- Reportedly used with over 200 projects over a 5-year period, and produced an average deviation of less than 9% between estimated effort (using the UCP method) and recorded effort.
- However, no information is provided about:
 - the dispersion across this average,
 - the statistical technique used to build the estimation model, or
 - the classical quality criteria of estimation models such as
 - the coefficient of determination (R^2) and Mean Relative Error.
- In brief, no documented details are given to analyze the claim about the accuracy of the estimates.

Summary

This chapter has presented:

- Overview of the Use Case Points (UCP):
 - origins & initial design.
- Analysis of numerous issues in the design of the UCP:
 - Disparities in the entities and attributes measured,
 - Incorrect usage of the scale types
 - Unknown nature of the UCP units
 - Subjectivity in the measurement method
- The limitations of related work on the UCP relationships with development effort