

# A Report on Applications of the Use Case Points Method in Industry

Peter Merrick and Patrick Barrow  
University of East Anglia, Norwich, Norfolk, U.K.  
*peter.merrick@uea.ac.uk*

## Abstract

*This report examines issues intrinsic to the application of the Use Case Points Method of top down algorithmic effort estimation in the early stages of the development lifecycle. It begins by setting the method into context, suggests modifications that make it easier to apply, and examines the prerequisite that different Use Cases should be comparable by ensuring they are expressed at the same level of abstraction. Progress is reported on two applications of the method and the tests being undertaken designed to test whether the method is a good predictor of relative complexity.*

## 1. Introduction

Use Case modelling is a popular technique for representing requirements, normally employed in the analysis phase of the software engineering lifecycle. UML [1] specifies Use Case modelling as the basis of a development approach where Use Cases act as the input to design, as the basis of verification, validation and testing. There is less in the literature on the application of Use Cases in the process of effort estimation [2, 3]. In this paper, the practical steps required to begin employing Use Cases as the basis of a complexity measure and estimation method are reported. The original work employing Use Cases as an input to an algorithmic estimation approach was introduced by Karner [4, 5] (Use Case Points Method) and draws its inspiration from the better known methods; Function Point Analysis (FPA) [6], and COCOMO [7, 8]. This paper investigates the use of requirements (captured in Use Case form) to express system complexity and relative effort at an early stage in the engineering lifecycle in the formulation of estimates.

Preliminary results are presented from the application of a modified version of the Use Case Points Method (UCPM) to calibrate the method so that it may be employed in an industrial environment on

behalf of the Health and Safety Executive (HSE) [9]. This is part of ongoing work between the authors, the HSE and their suppliers to look at ways of improving effort estimation. Currently the HSE has a non-exclusive 'partnership' agreement with their IT suppliers, who employ a system of 'bottom-up' task based effort estimation that has attracted criticism.

## 2. Use Case Abstraction and Algorithmic Simplification

Use Cases were first introduced by Jacobson [10] and have since become part of the UML standard. They are a recognised and popular method for capturing user requirements. A Use Case is comprised of two elements, a graphical element and a textual element. Modelling for the purpose of procurement in this paper is constrained to the graphical element, a sub-set of a complete Use Case that Kulak [11] refers to as a 'façade'.

A prerequisite to the successful application of the UCPM is agreement as to the correct level of abstraction that is appropriate for Use Case representation. Within the requirements engineering (RE) community there has emerged the desirability of considering problem expression from the perspective of goals. Goals are represented according to a hierarchical scale of abstraction ranging from the representation of strategic concerns to low-level technical detail concerned with aspects of implementation [12]. In considering Use Cases however, Jacobson believed that they should be represented at the level of a user of the system, such that the user would recognise the Use Case as delivering observable value [10]. Cockburn [13] takes a broader view, stating that a Use Case exists on a goal centered scale, ranging from 'very high level' to 'too low' (very high summary, summary, user-goal, sub-function, too low) which aligns Use Case modelling more comfortably with the goal-driven

orthodoxy [12]. For the purposes of the work reported here, Use Case abstraction has been standardised according to guidelines presented in an earlier paper [14] which describes the same Use Case as represented at different levels of abstraction dependent on the need of the model's audience. Use Cases are represented at two levels of abstraction; the *summary* level and the *user-goal* level. Models presented in the figures (Fig. 1, Fig. 2) of the report are represented as *summary* use cases due to space restrictions, even though Use Cases represented at *user-goal* level were employed in the UCPM calculations.

The UCPM was developed with direct reference to FPA and COCOMO. The motivation behind the UCPM is to take advantage of the fact that it takes Use Cases as an input to its algorithm, artefacts that may be available at an early stage in the engineering lifecycle, whereas the inputs to the other methods are more readily available after the system in question has been built. This is likely to account for their unpopularity as early effort techniques [15].

Both FPA and COCOMO make reference to technical and environmental modifiers. A simplification of the method is employed that ignores both the technical complexity factor (TCF) and the environmental complexity factor (ECF). Karner himself recommends the adjustment factors be ignored where the detail is unavailable [4]. Further simplifications to the UCPM are made that reduce the number of categories available for the classification of actors and Use Cases from three (simple, average and complex) to two (simple and complex). This is done in recognition that the decision as to categorisation as proposed by Karner is potentially open to the same subjectivity with which Symons [16] criticises the FPA method. These changes render a method that is easier to apply but which remains substantively comparable to the original.

The simplified UCPM calculation is thus reduced to the sum of all the weighted actors plus the sum of all the weighted use cases. This can be expressed mathematically as:

$$\sum_{i=1}^4 n_i * w_i$$

Where  $i = 1$  to 4 represents each of the four categories: simple actor, complex actor, simple Use Case, complex Use Case.  $n_i$  is the number of items of type  $i$ , e.g. the total number of complex actors. The basis of the decision as to the category of a modelling element is given in the description of the requirements pattern language that follows.

### 3. Transforming Requirements into Models

Requirements can be represented in a variety of different ways ranging from the unstructured informality of natural language to the precise representation of the Z specification language. Use Cases fall somewhere in-between these extremes and are normally produced as part of the engineering lifecycle rather than during procurement. To overcome time as a constraint a series of requirements patterns employing Use Case modelling constructs were developed. This approach is intended to generate Use Cases in a predictable manner [17, 18] that conform to *de facto* standards of Use Case expression [19]. The requirements pattern language (RPL) is comprised of the following patterns: **Minimal Representation**, the **NewSearchModify (NSM)**, **Usual Suspects**, **Informed Management**, **Role Based Assignment**, and **Coherent Whole**. Collectively, their purpose is to produce an initial Use Case model that can be employed as the input to the UCPM to arrive at a complexity measure. The constituent patterns of the RPL are not presented in their entirety due to space restrictions, however an overview is given that introduces the principles governing the application of each pattern. A prerequisite to using this pattern language is that a *sufficient* requirements specification exists that should include a description which includes the nouns that will allow for the identification of entities as described by Abbott [20]. The first pattern is **Minimal Representation** which requires the construction of a logical minimal 'database-like' model sufficient to support the business. Care was taken to select only nouns that would represent entities while nouns that represented attributes were eliminated. Nouns that resolved *many:many* relationships were added to the models [21]. Once a candidate data model had been constructed, the next pattern of the RPL can be applied, known as **New\_Search\_Modify (NSM)** which defines a set of candidate Use Cases over each identified entity discovered by *Minimal Representation*. This pattern draws on the ability to build up complex behaviour from basic functions that are generally necessary in database systems, namely the ability to Create, Read, Update and Delete (CRUD) over data entities. Other authors have suggested that the identification of functionality based on data entities is likely to produce a valid candidate set [22, 23]. All Use Cases defined through the application of this pattern are categorised as *simple* Use Cases for the purpose of the ensuing calculation. The objective of the **Usual Suspects** pattern is to allow for quick identification of roles (known as actors [1, 10]) that

have a high likelihood of occurring in a 'transactionally-driven' software application offering a service to an end-user. All human actors defined through the application of this pattern are categorised as *complex* for the purpose of the complexity calculation. Non-human actors are defined as *simple* according to the assumption they represent easily addressable external systems following Karner [4]. After the application of the **Usual Suspects**, it is appropriate to apply the **Informed Management** pattern which generates reporting Use Cases, often thought of as providing the functionality associated with a management information system (MIS). Such Use Cases are categorised as *complex*. The pattern **Role Based Assignment** provides a way of grouping Use Cases and representing them in a manner that is logically coherent from the perspective of presentation. The final pattern, **Coherent Whole** provides a check to ensure that the system model generated represents a logical unit whose sum provides recognisable value to its cast of users (actors).

The RPL has been applied on a number of occasions and shown to have value in predicting the final shape of the finished application [9, 24]. Within the greater context of exploring the usefulness of the UCPM for effort estimation it is necessary to show that when the RPL predicts one system is more complex than another that this will be borne out in practice by requiring more resources in its fulfillment. To attempt this, two high-level models are presented of different IT applications of different complexities (according to the application of the UCPM). It is hoped to show that when these applications are built the total comparative amounts of time expended will show a correlation with the prediction. For example, if one application is predicted to have a complexity measure of 10 and another a measure of 15, we would firstly hope to show that the application of the greater complexity took longer to build (cost more in this example, as time is the principle component of cost). Secondly, we would like to show that the more complex project took close to 50% longer to build within an acceptable margin of error.

In the first half of the experiment the RPL was used to calibrate an initial *time/unit of functionality* metric that would allow a comparison to be made. To arrive at a figure it was decided to undertake an exercise in reverse modelling of a system that had recently been built for the management of planning applications near to hazardous sites. The resulting model would then be transformed into a value expressed in complexity points. As the amount charged by the supplier was known, this monetary value could be divided by the

number of complexity points to arrive at an initial figure sufficient to represent the cost of satisfying a single notional unit of function. The cost could be converted to time by further dividing the known total by the supplier's hourly rate (averaged). The authors recognise that this calculation serves only as a starting place and in the event the method shows some promise will require ongoing calibration.

In the second half of the experiment, the RPL was applied again to a new requirement for the representation of a fleet vehicle management system intended to replace a legacy application. In this iteration, a new complexity point's measure would be calculated, and multiplied by the *time/unit of functionality* metric calculated in the experiment's first half. Embedded in this cross-domain approach is the assumption that these systems are in some sense capable of comparison. The requirements language itself is designed to be independent of vertical business domains (e.g. banking, travel, retail) although it is constrained to predicting the shape of transactionally-oriented database driven applications, sometimes characterised as solutions that fit the 'workpiece' frame [22]. The Domain Theory [25] proposes that all software engineering problems can be described by a small set of fundamental abstractions. Sutcliffe suggests that one of the ways an application may be built is through specification in a high level requirements language leading to program generation. The ability to predict the ultimate shape of an application from its requirements expression would be valuable in expressing system size, as an input to effort estimates, in managing change and reporting progress.

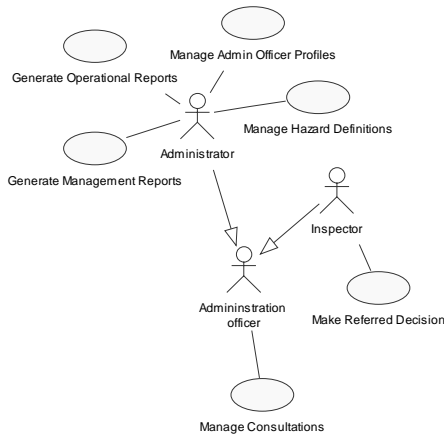
Because this experiment is in its early stages, and there are a great many potential variables, the criteria for success are modest. Success is defined as an outcome whereby the more complex system takes proportionally longer to build than the system characterised as less complex. Ideally, it is hoped to demonstrate that there is a correlation between the calculated system size and the amount of time it takes to build.

## 4. Modelling Results

As previously stated, two applications have been modelled. The first application supports the planning authorities in the U.K. who have a statutory duty to consult on certain developments that fall within a specified proximity to hazardous installations and pipelines. It was intended that the system would support the Land Division of HSE in the processing of land use planning applications. Every application will

ultimately receive either an ‘allow’ or a ‘refuse’ decision. Applications are made where a development is within the consultation distance of hazardous installations which are defined as being pipelines, quarries, explosive and nuclear sites. The basis upon which decisions are made is a set of rules that have been codified through many years of experience. The objective of undertaking the work was to allow a lower-grade of administrative officer to process routine consultations.

This project was a new-build bespoke application chosen by HSE for the purpose of reverse modelling because it was considered to have been successful. The project did suffer from some over-runs, but less than had been experienced on other projects. To perform the reverse modelling exercise, the authors worked from the original requirements statement produced by the customer without physically seeing the working application. When the model was complete, it was verified by an application inspection. Care was taken to limit the introduction of bias into the definition of the evaluation Use Cases. The working application was then modelled in its delivered state and the predicted model was compared with the model of the working program. This was done to ensure the RPL was capable of predicting the final form of the application with reasonable accuracy and is reported in [9].



**Fig. 1: Model of the HSE system for the management of planning applications near hazardous sites expressed employing summary level Use Cases.**

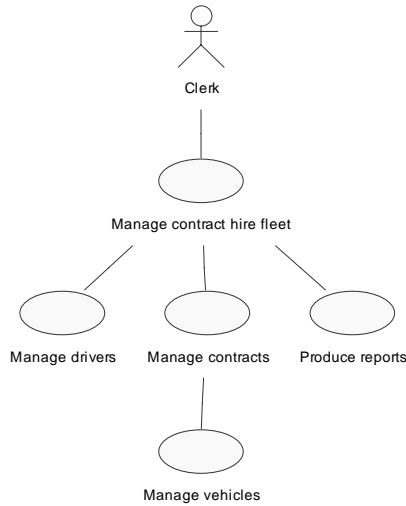
Component	Weight	Number	Value
Secondary Actor	1.5	0	0
Primary Actor	3	2	6
Simple use case	5	20	100
Complex use case	10	6	60
Total UCP			166

**Table 1: table representing the complexity of the HSE planning application system modelled at the level of user-goal (not shown) Use Cases as presented in [9].**

Due to the restrictions of commercial confidentiality the sums charged for delivering a system equivalent to 166 UCP are not reported. Instead a baseline is proposed whereby a system of 166 points is defined as taking 100 units of time.

The RPL was then applied to the new project to build a vehicle management system. This project was to replace an existing legacy application that had been built in an ad hoc manner by the users themselves. The model was built with the RPL that was applied by analysts who worked for the suppliers to the HSE after they were trained by the authors in a two day workshop. The authors built a model of the application independently for the purpose of comparison.

In comparing Fig. 1 and Fig. 2, it can be seen that one appears to be bigger than the other in terms of the number of actors and featured Use Cases. This is borne out by the complexity calculations, although it must be remembered that Use Cases at the user-goal level are employed as inputs to the method while the Use Cases in Fig. 1 and Fig. 2 are presented at the summary level. Therefore it is not possible to perform the calculation from the information presented in Fig. 1 and Fig. 2.



**Fig. 2: Model of the HSE system for the management of vehicle leasing employing summary level Use Cases.**

Component	Weight	Number	Value
Secondary Actor	1.5	0	0
Primary Actor	3	1	3
Simple use case	5	6	30
Complex use case	10	3	30
Total UCP			63

**Table 2: table representing the complexity of the HSE leasing vehicle system modelled at the level of user-goal Use Cases.**

According to the two complexity measures, the system characterised in Table 2 at 62 points is 37% of the size of the system described in Table 1 (166 points). According to the system of baselined time therefore, it is predicted this system will take 37 time units.

This top down estimate was compared with an existing bottom up estimate (whereby constituent tasks are assigned times and summed) and were found to be somewhat different. Applying the bottom up method the supplier estimated the project would take 24 nominal time units (or 1/4 of the time of the application in Fig. 1).

Which estimate is closest cannot yet be reported, because the work has not been completed. It is instructive to consider that the supplier's record of exceeding their estimates is a prime motivator for this research, although the top down estimate is fully 50% greater than the bottom-up estimate.

## 5. Comment

Any programme of estimation is likely to be improved by employing more than one method, by keeping appropriate metrics and by treating the results as estimates rather than targets [26]. Effort estimation algorithms are not used extensively in industry [15]. This leaves the practitioner with a range of choices for estimation that are not directly based on a statement of requirements (bottom-up, expert opinion etc. ). This report describes the efforts being made to allow the UCPM to be employed in an industrial setting while at the same time acknowledging that there are many variables which are difficult to control.

[1.] Fowler, M. and K. Scott, *UML Distilled - A Brief Guide to the Standard Object Modeling Language*. 1999, Upper Saddle River, N.J.: Addison Wesley Longman

[2.] Anda, B., D. H, S. D, and J. M. *Estimating Software Development Effort based on Use Cases - Experiences from Industry*. in *4th International Conference on the Unified Modeling Language*. 2001. Toronto, Canada: Springer-Verlag

[3.] Anda, B., E. Angelvik, and K. Ribu. *Improving Estimation Practices by Applying Use Case Models*. in *4th International Conference on Product Focused Software Process Improvement, Rovaniemi, Finland, December 9 - 11, pp. 383-397, LNCS 2559, Springer-Verlag*. 2002. Profes, Finland

[4.] Karner, G., *Resource Estimation for Objectory Projects*, 1993, Masters thesis, Linköping Institute of Technology, Linköping

[5.] Schneider, G. and J. Winters, *Applying Use Cases - A Practical Guide*. 1998: Addison Wesley Longman, Inc.

[6.] Albrecht, A.J. *Measuring Application Development Productivity*. in *IBM Applications Development Symposium*. 1979. Monterey, CA

[7.] Boehm, B., *Software Engineering Economics* Advances in Computing Science. 1981: Prentice Hall

[8.] Boehm, B. and E. Horowitz, *Software Cost Estimation with Cocomo II*. 2000: Pearson Education

[9.] Merrick, P. *Testing a Requirements Pattern Language Through Reverse Engineering*. in *INCOSE 2004*. 2004. Toulouse, France: accepted for presentation

[10.] Jacobson, I., Jonsson, P., Christerson, M., Overgaard, G., *Object-Oriented Software Engineering - A Use Case Driven Approach* ACM Press. 1992, Upper Saddle River, N.J.: Addison Wesley Longman

[11.] Kulak, D. and E. Guiney, *Use Cases - Requirements in Context*. 2000, Upper Saddle River, N.J.: Addison Wesley Longman

[12.] van Lamsweerde, A. *Goal-Oriented Requirements Engineering: A Guided Tour*. in *Requirements Engineering 2001 (RE01)*. 2001. Toronto: IEEE

[13.] Cockburn, A., *Writing Effective Use Cases*. 2001, Upper Saddle River, N.J.: Addison Wesley Longman

- [14.] Merrick, P. and P. Barrow. *Towards a Requirements Formalism in Procurement*. in *8th Annual Conference of United Kingdom Academy of Information Systems*. 2003. Warwick, England
- [15.] Taylor, A., *IT projects sink or swim*. The Computer Bulletin, 2001. **42**(1): p. 24-26, Oxford Journals.
- [16.] Symons, C.R., *Function Point Analysis : Difficulties and Improvements*. IEEE Transactions on Software Engineering, 1988. **14**(1): p. 2-11, IEEE.
- [17.] Biddle, R., J. Noble, and E. Tempero. *Patterns for Essential Use Cases*. in *KoalaPLoP*. 2001. Melbourne, Australia
- [18.] Biddle, R., J. Noble, and E. Tempero. *Essential use cases and responsibility in object-oriented development*. in *Twenty-Fifth Australasian Computer Science Conference (ACSC2002)*. 2002. Monash University, Melbourne, Victoria, Australia: Australian Computer Society Inc.
- [19.] Adolph, S., P. Bramble, A. Cockburn, and A. Pols, *Patterns for Effective Use Cases Agile Software Development*. 2002, Upper Saddle River, N.J.: Addison Wesley Longman
- [20.] Abbott, R.J., *Program design by informal English descriptions*. Communications of the ACM, 1983. **26**: p. 882-894, ACM Press.
- [21.] Connolly, T., A. Strachan, and C. Begg, *Database Systems - A Practical Approach to Design, Implementation and Management*. 1995, Harlow: Addison-Wesley Longman
- [22.] Jackson, M., *Problem frames: analysing and structuring software development problems*. 2001, Harlow: Pearson Education
- [23.] Armour, F. and G. Miller, *Advanced Use Case Modelling*. 2001: Addison Wesley
- [24.] Merrick, P., *Testing the Predictive Ability of a Requirements Pattern Language*. Requirements Engineering (accepted for publication), 2004 Springer-Verlag.
- [25.] Sutcliffe, A., *The Domain Theory - Patterns of Knowledge an Software Reuse*. 2002, London: Lawrence Erlbaum Associates, Inc.
- [26.] Fenton, N. and S.L. Pfleeger, *Software Metrics - A Rigorous & Practical Approach*. 1996, London: Thomson Computer Press