FUNCTION POINT SIZE ESTIMATION FOR OBJECT ORIENTED SOFTWARE BASED ON USE CASE MODEL

A. Chamundeswari and Chitra Babu
SSN College of Engineering, Rajiv Gandhi Salai, SSN Nagar – 603110, India

Keywords: Object-Oriented Software, Use Case Model, Function Point, Object Model.

Abstract: Precise size estimation earlier in the software development life cycle has always been a challenge for the software industry. In the context of object oriented software, Use Case Model (UCM) is widely used to capture the functionality addressed in the software. Existing size estimation techniques such as use case points and use case size points do not adhere to any standard. Consequently, lots of variations are possible, leading to inaccurate size estimation. On the other hand, Function Point Analysis (FPA) has been standardized. However, the current estimation approaches based on FPA employ object modeling that happens later in the software development life cycle rather than the UCM. In order to gain the advantages of FPA as well as UCM, this paper proposes a new approach for size estimation of object oriented software. This approach is based on the UCM by adapting it to FPA. Mapping rules are proposed for proper identification and classification of various components from UCM to FPA. Estimation results obtained using the proposed approach are compared with those using finer granular level object model which adapts FPA at design phase. The close agreement between these two results indicates that the proposed approach is suitable for accurate software size estimation earlier in the software development life cycle.

1 INTRODUCTION

Software size estimation is a challenging activity in software development life cycle (Kusumoto et al., 2004). One of the widely used size estimation techniques is FPA. It was initially proposed (Albrecht, 1979) to estimate the software size based on the functionality from requirements specification, independent of the technology used to build the software. It contains five components namely External Interface File (EIF), Internal Logical File (ILF), External Input (EI), External Output (EO) and External Inquiry (EQ). All these five components are estimated based on the functionality. Fourteen Technical Complexity Factors (TCF) are evaluated based on the non-functional requirements. FPA has been approved by International Function Point User Group (IFPUG) and it became a standard. It is widely accepted in software industry as a superior metric compared to the naïve Lines of Code counting for size estimation.

As object oriented methodology was embraced by several software organizations, there was a need to suitably adapt the FPA approach. Many researchers have proposed the adaptation of FPA approach for object oriented software size estimation using object model (Antoniol et al., 1999, Ayman et al., 2006 , Zivkovic et al., 2005). These approaches estimate size of software by mapping the various key notions of object model to the FPA components. However, UCM has certain distinct advantages in capturing the system requirements earlier in software development life cycle (Kusumoto et al., 2004). Some researchers have contributed to size estimation based on UCM in terms of use case points or use case size points (Edward, 2005, Kusumoto et al., 2004, Marico et al, 2006). Use case points size estimation technique is an extension of FPA and Mark II FPA (Marico et al., 2006). However, these approaches are not standardized as opposed to FPA which is governed by IFPUG.

Furthermore, the UCM captures different granularities such as brief, fully-specified and refinements in the requirement analysis phase (Cockburn, 2001) in detail. This paved the way to the formulation of the hypothesis that the more detailed the UCM, the more accurate will be the anticipated system's predicted size (Ayman et al., 2006).

Chamundeswari A. and Babu C. (2008).
FUNCTION POINT SIZE ESTIMATION FOR OBJECT ORIENTED SOFTWARE BASED ON USE CASE MODEL.
In *Proceedings of the Third International Conference on Software and Data Technologies - SE/GSDCA/MUSE*, pages 139-145
DOI: 10.5220/0001879701390145

The main motivation for this research stems from the hypothesis that UCM is more appropriate in capturing the system requirements earlier in the software development life cycle and consequently provides accurate size estimation. Even though use case points and use case size points are based on the UCM, they do not adhere to any standard. On the other hand, FPA is standardized. In order to combine the advantages of UCM as well as FPA, this paper proposes a new size estimation technique based on the UCM.

The objective of this research work is to provide accurate size estimation using FPA technique. This has been achieved as follows:

1. Mapping the UCM components to FPA components during the analysis phase.
2. Estimating size of object oriented software in terms of function points by applying this FPA mapping.
3. Comparing the estimated size with the existing object model size estimation technique.

The proposed approach is substantiated with finer granularity level object model during design phase.

The remainder of the paper is organized as follows. Section 2 surveys work related to size estimation of object oriented software. In Section 3, a size estimation model which applies FPA on UCM is presented. Section 4 discusses the results. Section 5 concludes and suggests future work.

2 RELATED WORK

The research work on quantitative size estimation for object oriented software has been the focus of many researchers. Size estimation has been dealt during the various phases of the software development life cycle such as analysis (Fetcke et al., 1997, Antoniol et al., 2003, Fernandez et al., 2004, Harput et al., 2005, Zivkovic et al., 2005), design (Ram et al., 2000, Uemura et al., 2001, Antoniol et al., 2003, Zivkovic et al., 2005) and development (Whitmire, 1992, Schooneveldt, 1995, Minkiewicz, 1997). FPA technique cannot be directly used for estimating size of object oriented software (Fetcke et al., 1997, Antoniol et al., 1999, Ram et al., 2000, Harput et al., 2005, Zivkovic et al., 2005). Hence mapping rules were framed, to adapt FPA for estimating the size of object oriented software.

Use case point method for estimating size of projects developed with object oriented methodology was first proposed by (Karner, 1993). However, it has been tested only on a few small projects. Therefore, more research is needed to establish the general usefulness of the method. FPA technique was applied on OOSE Jacobson method (Fetcke et al., 1997). This was the first attempt to apply FPA on object oriented software to estimate the size.

A measure on UCM and object model was proposed by (Zivkovic et al., 2005). In their study, the approach of ISBSG statistical tool kit was adapted for calculating the size estimation based on use case diagram.

Use case point size estimation was proposed considering the various parameters for UCM such as actors, use cases, technical factors and eight experience factors by (Edward, 2005). An analysis of the performance with empirical data based on use case point, it was shown that there was a deviation in effort from planned to actual by -41.43%.

Use case size points estimation from UCM was also proposed by (Marico et al., 2006). Finer granularity such as classification of actors, preconditions, main scenarios, alternative scenarios, exceptions, post conditions, TCF and environment adjustment factor were considered for estimation. Manual measurement was done using original FPA. Error rate of the estimate showed no significant difference between FPA and use case size point.

However, size estimation techniques such as use case point and use case size point based on UCM follow different procedures and hence produce varied results. In addition, these techniques also have the following shortcomings:

1. Focus on the internal structure alone.
2. Lack of boundary identification.
3. Lack of identifying interaction with other external use case.
4. Not considering overlapping use cases that capture the same functionality.

In order to address these shortcomings, this paper proposes a new estimation technique based on UCM by applying FPA standard.

3 PROPOSED SIZE ESTIMATION MODEL

Figure 1 gives a pictorial overview of the proposed size estimation model. Software comprises of many applications and it is essential to identify the boundaries of different applications. Boundary of an

application is classified into two categories: internal and external.

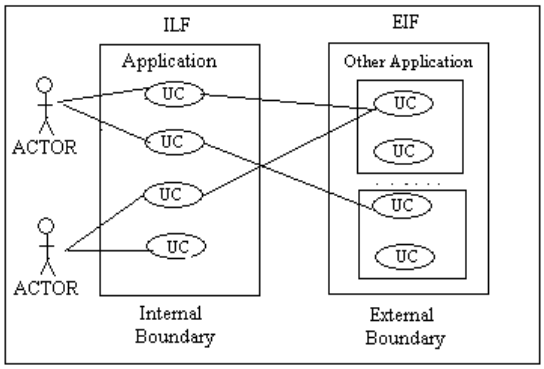

Figure 1: An overview of size estimation model.

An internal boundary contains actors which directly invoke a set of use cases in an application. An external boundary contains use cases of different applications, which can be referenced by use cases from an internal boundary. Logical files in FPA are of two categories: ILF and EIF. ILF are identified based on the artifacts that affect the internal boundaries of the system, while EIF are identified based on the artifacts that affect the external boundary of the system. Thus the artifacts related to an internal boundary of UCM are mapped to ILF components, while the artifacts related to an external boundary is mapped to EIF components of FPA.

Transactions in the FPA technique are classified as EI, EO and EQ. The classifications EO and EQ are not easily applicable in UCM, because scenarios are the main transactions in use case and are defined as Transaction Function (TF). TF of UCM is mapped to EI components of FPA.

The detailed size estimation procedure consists of four steps. The first step is identification of the boundaries (internal and external). The second step is identification of files and transactions within these boundaries. The third step is to assign weights for files based on their classification. The fourth step is to count the size of the software application. The following subsection discusses these steps in detail.

3.1 Mapping UCM to FPA

3.1.1 Boundary Identification

The main objective of FPA is to determine the size based on functional requirements of the software application. Identification of boundary is essential to determine the artifacts under estimation. The boundary indicates the border between different applications. The various parameters in UCM are boundaries, abstract use case, concrete use case, active actors, passive actors, devices and scenarios. The following mapping rules are proposed for proper identification of internal and external boundaries in UCM.

1. Active actors who are directly communicating with the use cases within the system boundary are internal.
2. Passive actors who are directly communicating with the use cases for data storage purpose within the system boundary are internal.
3. Devices which are directly interacting with the use case within the system boundary are internal.
4. All other external references are external.

3.1.2 Identification & Classification of Logical Files

Use cases are the main candidates in the UCM based on which the estimation is carried out. A set of use cases form the candidates for logical files. An actor that directly invokes one or more use cases are grouped as an ILF. The following mapping rules are proposed for proper identification of ILF in UCM.

5. Select use cases that have direct connection to an actor or device as stated in rules 1, 2 or 3.
6. Group the use cases with their respective actor or device separately.
7. Reject all other use cases that have relationship *uses* or *extends.*

Classification of ILF depends on two parameters namely Record Element Type (RET) and Data Element Type (DET). RET represents a user recognizable group of logically related data. DET represents a simple unique user recognizable, non-recursive data in RET. Each use case invoked by an actor is counted as an RET. Data that flows from actor to use case or from use case to actor is counted as a DET. When RET and DET parameters are classified and measured, ILF complexity table defined as in IFPUG is used for classifying complexity as low, average and high.

In the case of EIF, internal use case within the internal boundary refers to external use case maintained by other applications. An actor through internal use case can invoke indirectly one or more external use cases from another application. An actor with external use cases is grouped as an EIF. The following mapping rules are proposed for proper identification of EIF in UCM.

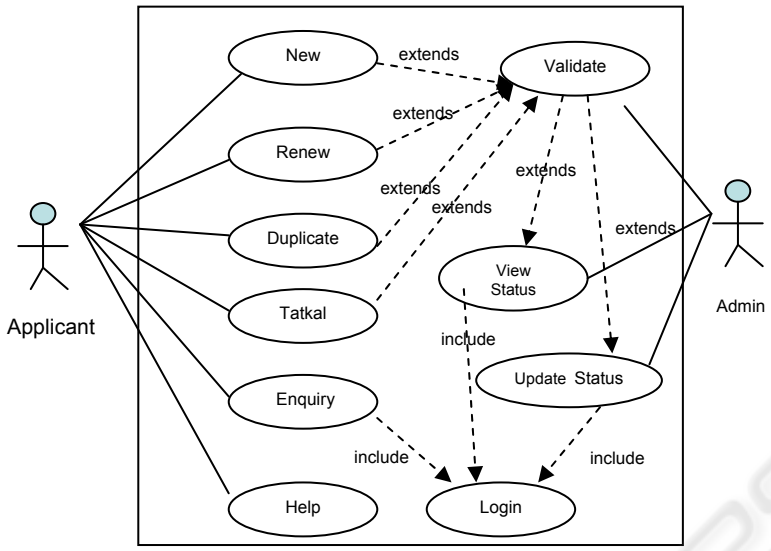

Figure 2: Typical Use Case diagram.

8. Select internal use cases that refer external use cases.
9. Identify the related actor or device of internal use cases that invoke the external use cases.
10. Group the actor or device with the corresponding external use cases.
11. Reject all other external use cases.

Classification of EIF also depends on the two parameters namely RET and DET. Each external use case that is invoked though an actor is counted as an RET. Data that flows from actor to external use case or from external use case to actor is counted as a DET. When RET and DET parameters are classified and measured, EIF complexity table defined as in IFPUG is used for classifying complexity as low, average and high.

3.1.3 Identification & Classification of Transactions

The following mapping rules are proposed for proper identification of TF in UCM.

12. Select concrete use case and identify the scenarios for each.
13. Identify the various transaction messages for each scenario.
14. Identify the unique transaction messages for different scenarios in each use case.
15. Reject all overlapping transaction messages for different scenarios in each use case.

Classification of transactions depends on two parameters, namely, File type referenced (FTR) and DET. FTR represents a transaction in ILF or EIF. DET represents a simple unique user recognizable and non-recursive data in FTR. Each use case scenario is counted as an FTR and each transaction message is counted as a DET. When the DET and FTR parameters are classified and measured, EI complexity table defined as in IFPUG is used for classifying complexity as low, average and high.

3.2 Size Estimation

The size estimation of a software application can be determined by applying all the above 15 proposed rules. The proposed estimation, UCM_{fp}, is calculated as shown below.

$$UCM_{fp} = Unadj\ FP_{usecase\text{-}estimation} * TCF$$

where

$$Unadj\ FP_{usecase\text{-}estimation} = EIF_{size\text{-}complexity} + ILF_{size\text{-}complexity} + TF_{size\text{-}complexity}$$

$$EIF_{size\text{-}complexity} = f\ (RET, DET)$$

$$ILF_{size\text{-}complexity} = f\ (RET, DET)$$

$$TF_{size\text{-}complexity} = f\ (FTR, DET)$$

$$TCF = 0.65 + 0.01 * \sum_{i=0}^{14} t_i$$

UCM_{fp} is determined from four components namely $EIF_{size\text{-}complexity}$, $ILF_{size\text{-}complexity}$, $TF_{size\text{-}complexity}$ and TCF. By determining the complexities of the corresponding parameters RET and DET the size complexity of $EIF_{size\text{-}complexity}$ and $ILF_{size\text{-}complexity}$ is calculated. Parameters FTR and DET determine the complexity of the $TF_{size\text{-}complexity}$. TCF is determined from t_i for fourteen characteristics, as defined in (Albrecht, 1979). A case study for which the proposed size estimation technique is applied is described in the next section.

4 RESULTS AND DISCUSSION

As a case study for validating the proposed estimation model, passport automation system developed in the Software Engineering Laboratory in our Institute has been considered. Fifteen mapping rules as specified in section 3.1.1, 3.1.2, 3.1.3 and the formula in section 3.2 are applied for validation. Figure 2 shows the UCM of this application. This case study consists of all activities in software development life cycle such as analysis, design, implementation and testing. The total number of use cases is 10 with 2 active actors and 3 passive actors. Number of data provided or received from the use cases are 85. Number of scenarios addressed by all these use cases is 18 with message flow of 100.

By applying rules 1, 2, 3 and 4 the boundaries of the application is identified. By applying rules 5, 6 and 7 the internal files are identified and classified to determine RET and DET. Table 1 shows the $ILF_{size\text{-}complexity}$ for the passport automation system. By applying rules 12, 13, 14 and 15 the scenarios and unique transaction messages are identified.

Considering the use case diagram shown in Figure 2, '*new*' use case addresses two scenarios: *'completed form'* and *'uncompleted form'*. An identified transaction message for two scenarios is represented as a list of sequential steps in Figure 3. This is also represented using sequence diagrams as shown in Figure 4(a) and 4(b).

Table 1: ILF Size Complexity.

Actor	*RET*	*DET*	*COMP*
Applicant	*6*	*45*	*15*
Administrator	*3*	*15*	*7*
Passport_details	*2*	*15*	*7*
Passport_renew	*1*	*5*	*7*
Passport_status	*1*	*5*	*7*
		Total	*43*

1. An applicant request new form
2. Obtain the form
3. New passport application form
4. Display new passport application form
5. Fill and submit the application form
6. Is form filled? Yes go to step 9
7. Some fields not entered
8. Display message and go to step 4
9. Form filled
10. Request for payment
11. Make payment
12. Assign applicant id

Figure 3: Transaction messages of *new* use case.

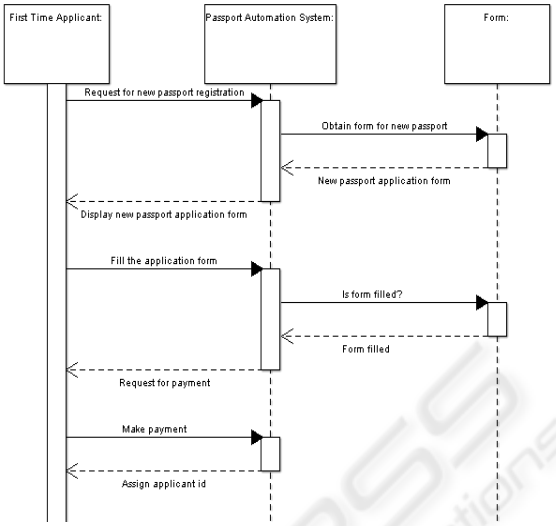

Figure 4a: View of '*complete form*' scenario.

Through the sequence diagrams it is clear that the '*completed form*' scenario has 10 transaction messages and '*uncompleted form*' scenario has 8 transaction messages. However 6 messages are overlapping out of the total 18 messages. Rejecting the overlapping messages, DET is calculated as 12 and RET is calculated as 2. Table 2 shows the $TF_{size\text{-}complexity}$ for the passport automation system. TCF is calculated as 1.07.

Unadjusted FP = ILF + TF
= 43+41 = 84
Adjusted FP = Unadjusted FP * TCF
= 84 * 1.07 = 89.88 FP

The proposed size estimation technique, UCM_{fp} is also applied on different projects developed in our software engineering laboratory. For the same set of projects, fine granular level object model size estimation technique, $object\_model_{fp}$ which adapts FPA during design phase proposed by (Antoniol et al., 1999) is also applied.

Table 2: TF Size Complexity.

Use case	*FTR*	*DET*	*COMP*
New	*2*	*12*	*4*
Renew	*2*	*9*	*4*
Duplicate	*2*	*9*	*4*
Talkaal	*2*	*9*	*4*
StatusEnquiry	*2*	*5*	*4*
Help	*3*	*3*	*4*
Validate	*1*	*4*	*3*
ViewStatus	*1*	*5*	*4*
UpdateStatus	*1*	*6*	*4*
Login	*3*	*6*	*6*
		Total	*41*

The comparison of projects using UCM_{fp} and $object\_model_{fp}$ is tabulated in Table 3. It is observed

from Table 3 that passport automation system project UCM_{fp} predicts more number of FPs when compared to $object\_model_{fp}$. In the case of other projects, $object\_model_{fp}$ is predicting higher number of FPs than UCM_{fp}. The accuracy evaluation of the estimation models are assessed using Magnitude of Relative Error (MRE), which is defined as

$$MRE = | UCM_{fp} - object\_model_{fp}| / UCM_{fp}$$

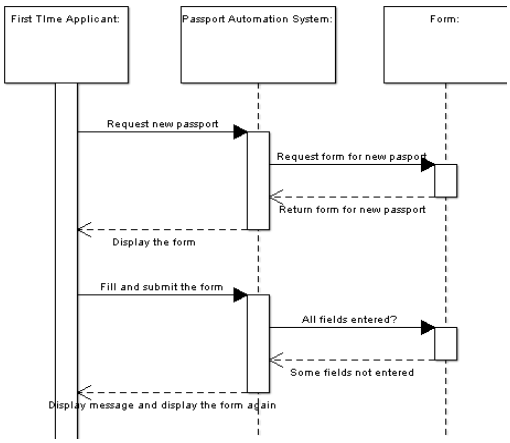

Figure 4b: View of '*uncompleted form*' scenario.

Table 3: Comparison of FP Size Estimation.

Project Name	UCM_{fp}	$object\_model_{fp}$	*MRE*
Passport automation System	89.88	83.46	0.071
e-book management system	83.46	87.74	0.069
On-line photo sharing and indexing	92.02	93.09	0.011
Foreign exchange system	108.07	115.56	0.069

Although there are some differences in the results obtained from the existing model to the proposed one, it is clear that FPA can be applied on UCM_{fp}. Also, the MRE between two techniques is found to be very low. This result shows that FPA predicts size accurately during early analysis phase when UCM components are mapped to FPA components for size prediction.

The advantages of the proposed estimation technique can be summarized as follows:

1. Size estimation using FPA can be applied at an early stage of software development life cycle using UCM.
2. Coarse granular size prediction can be achieved using UCM_{fp} and more fine granular size estimation can be done using $object\ model_{fp}$.

5 CONCLUSIONS AND FUTURE WORK

A new size estimation technique is proposed for estimating the size of object oriented software by applying FPA at the early analysis phase based on UCM. This approach involves mapping UCM components to FPA components. Clear and precise rules are proposed for identification and classification of various FPA components for size estimation. The results obtained from different projects by applying this technique is compared with the existing object model size estimation which applies the principles of FPA. The MRE achieved through these results are minimal and it signifies that FPA can be applied at an early stage based on UCM. Future work is to empirically validate this size estimation model by applying it to real projects from software industry.

REFERENCES

Albrecht, A., "Measuring application development productivity", IBM Application Development Symposium, 1979, pp. 83-92.

Al-Hajri, M.A., Ghani, A.A.A., Sulaiman, M.N., Selamat M.H., Modification of standard function point complexity weights system, Journal of Systems and Software volume 2, 2004, pp. 195–206.

Antoniol, G., Lokan, C., Caldiera, G., Fiutem, R., "A function point-like measure for object oriented software", Journal of Empirical Software Engineering, volume 4 Sep 1999, pp. 263-287.

Antoniol, G., Calzolari, F., Cristoforetti, L., Fiutem, R., Caldiera, G., "Adapting function points to object oriented information systems", Lecture notes in computer science, Advanced Information System Engineering, volume 13, 1998, pp. 59-76.

Antoniol, G., Fietum, R., Lokan, C., "Object oriented function points: An empirical validation", Journal of Empirical Software Engineering, volume 8, no 3, Sep 2003, pp. 225-254.

Ayman, I., Mohammed, O., David, C., "Software cost estimation using use case models: a critical evaluation", Proceedings of Second conference on Information and Communication Technologies, volume 2, April 2006, pp. 2766-2771.

Cockburn, A., "Writing effective use cases", Boston, London: Addison-Wesley, 2001.

Edward, R.C., "Estimating software based on use case points", Conference on Object Oriented Programming Systems Languages and Applications, 2005, pp. 257-265.

Fernandez, N.C., Abrahao, S., Pastor, O., "Towards a functional size measure for object oriented systems from requirements specifications", Proceedings of the

Fourth International Conference on Quality Software, volume 00, 2004, pp. 94-101.

Fetcke, T., Abran, A., Nguyen, T.H., "Mapping the OO-Jacobson approach into function point analysis", Proceedings of IFPUG, 1997 Spring Conference, 1997, pp. 134-142.

Harput, V., Kaindl, H., Kramer, S., "Extending function point analysis of object oriented requirements specifications", Eleventh IEEE International Software Metrics Symposium, volume 0, 2005, pp. 39-49.

Karner, G., "Resource Estimation for Objectory Projects", Objectory Systems, 1993.

Kusumoto, S., Matukawa, F., Inoue, K., Hanabusa, S., Maegawa, Y., "Estimating effort by use case points: method, tool and case study", Proceedings of the Sixth International Symposium on Software Metrics, Sep 2004, pp. 292 – 299.

Marico, R.B., Silvia, R.V., "Software effort estimation based on use cases", Proceedings of the thirty Annual International Computer Software and Applications Conference, Sep 2006, pp. 221 - 228

Minkiewicz, A.F., "Measuring object oriented Software with predictive object points", Proceedings of the Conference on Applications in Software Measurements, Oct 1997.

Ram, D.J., Raju, S.V.G.K., "Object oriented design function points", Proceedings of the First Asia Pacific Conference on Quality Software, Hong Kong, 2000, pp. 121-126.

Schooneveldt, M., "Measuring the size of object oriented systems", Proceedings of the Second Australian Conference on Software Metrics, Metrics Association, Nov 1995, pp. 168-177.

Uemura, T., Kusumoto, S., Inoue, K., "Function point analysis using design specification based on the Unified Modeling Language", Journal of Software Maintenance Evolution-Research Practice, volume 13, 2001, pp. 223-243.

Whitmire, A.S., "Applying function points to object oriented software models", Software Engineering Productivity Handbook, Mc Graw-Hill, New York, 1992, pp. 229-244.

Zivkovic, A., Hericko, M., "Tips for estimating software size with FPA method", Proceedings of the IASTED International Conference on Software Engineering, Acta Press, 2004, pp. 515-519.

Zivkovic, A., Hericko, M., Brumen, B., Beloglavec, S., Rozman, I., "The impact of details in the class diagram on software size estimation", Informatica (Lithuania), volume 16, no 2, 2005, pp. 1-18.

Zivkovic, A., Hericko, M., Kralj, T., "Empirical assessment of methods for software size estimation", Informatica (Lithuania) volume 4, 2003, pp. 425-432.

Zivkovic, A., Rozman, I., Hericko, M., "Automated software size estimation based on function points using UML models", Journal of Information and Software Technology, volume 47, 2005, pp. 881-890.