ESTIMATING SOFTWARE PROJECT EFFORT BASED ON THE DEVELOPMENT PROCESS MODEL

Ursula Passing, Susanne Strahringer

Department of Information Systems
European Business School
Oestrich-Winkel, Germany
{ursula.passing, susanne.strahringer}@ebs.de

ABSTRACT

A new, process-oriented effort estimation approach for custom software development projects is presented. It provides a methodological framework to support "subjective" expert judgements and analogy estimating, the most popular estimating techniques in practice, in order to make estimates transparent and repeatable. The proposed approach combines bottom-up estimation with a prescriptive estimation procedure. It is based on a standard decomposition of the software development process. Three dovetailed methods are provided on how to estimate the items in the decomposition, depending on the time of the estimation, the availability of project details, the desired estimation accuracy and the desired prescriptiveness of the estimation method.

Keywords. Effort estimation, Rational Unified Process, software cost, software project management.

1. INTRODUCTION

Effort estimation continues to be an important issue in custom software development project management today. Although researchers have been working on estimation models for decades, subjective and unstructured estimation practices, such as the so-called expert judgements, still seem to prevail in day-to-day effort estimating.

The effort estimation approach proposed in this paper provides a methodological framework to support expert judgement and analogy estimating, the most popular estimating techniques in practice. The aim is to make estimates transparent and repeatable.

The proposed approach is process-oriented: The effort is estimated based on the activities performed in a project.

The paper is organized as follows: First, existing estimation techniques are shortly introduced and classified in order to illustrate the positioning of the new approach. Then, the fundamental principles and the structure of the estimation approach are presented in detail. Finally, the approach is discussed and conclusions are drawn.

2. EXISTING ESTIMATION METHODS

The existing effort estimation techniques can be classified into three types: Expert judgement, analogy estimation and algorithmic estimation models.

Expert judgement denominates estimation methods that are solely based on judgements of people who have extensive knowledge in the project-relevant area.[1] These methods can be further split up into macro and micro judgements. For a macro expert judgement, the expert is asked to estimate the effort needed for the whole project. For micro estimates, the expert estimates a part of the project, e.g., a specific task. The estimates of the parts are added up to form the estimate of the whole project.

Analogy estimation is in some respect a systematic form of expert judgement.[2] To estimate the effort for a new project, a similar completed project is identified and its actual cost is used as a basis for the estimate of the new project.[1]

In algorithmic models, the development effort is estimated as a function of variables representing the most important cost drivers in the project. Usually, the variables are identified by correlation analysis of data on completed projects.[1, 3] A more recent algorithmic approach are neural networks. They also work with combinations of parameters, but use artificial intelligence learning algorithms in order to identify the relations between them.[4] Algorithmic estimation models typically use the estimated size of the software to be developed in the project as the key parameter. Examples of algorithmic models are *COCOMO* [1, 5] and *Function Point Analysis (FPA)*[6].

Estimation methods can be distinguished by their degree of prescriptiveness and their use of decomposition.

The degree of prescriptiveness of an estimation method indicates to what extent the method prescribes a repeatable estimation procedure which produces estimates that are transparent and comprehensible to outsiders (i.e., outsiders can reconstruct the reasoning behind the estimate). Methods involving a low degree of prescriptiveness allow more flexibility, but the estimates may more easily be despised as untrustworthy "guesses".

Decomposition means splitting the project up into smaller parts for estimation, and adding the component estimates up to form the overall estimate. This is also called bottom-up estimating. The contrary is usually called top-down estimating: The effort is estimated for the whole project at once.[1] Decomposition reduces the uncertainty inherent in the estimation by reducing the complexity of the estimation task – the probability of gross estimating errors is reduced because the sum of the errors in the component estimates will likely be smaller than the error in an overall estimate. Moreover, the additive connection between the component estimates limits error propagation. Top-down estimating may allow faster estimates but involves higher uncertainty.

This implies a classification of the estimation methods as shown in Figure 1.

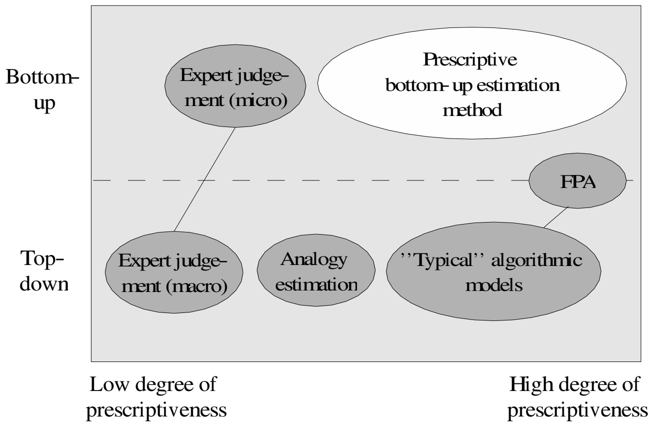

Fig. 1. Classification of effort estimation methods

Expert judgement is the least prescriptive of the three types of estimation methods because it does not provide a repeatable estimation procedure. It can be applied as a bottom-up (micro expert judgements) or a top-down technique (macro expert judgements).

Analogy estimates are considered more prescriptive than expert judgements because the estimator is instructed to use comparable projects to estimate the effort. Usually, analogy estimating represents a top-down approach because whole projects are compared and estimated.

Algorithmic estimation models have been classified to be the most prescriptive. The projects are characterized by parameters which have to be quantified – this represents a repeatable procedure. However, algorithmic models can be distinguished into approaches that either provide a defined quantification procedure for all their parameters (more prescriptive) or leave the quantification of certain parameters, e.g. the software size, open to the estimator (less prescriptive). Furthermore, most algorithmic models represent top-down approaches: All parameters relate to the project as a whole. In FPA however, the size estimate involves a decomposition of the software product into functions (external inputs, outputs etc.) As the estimating result is still an estimate of the whole project, FPA is to be considered a hybrid method.

The area at the upper right corner in the matrix is not populated by existing estimation approaches, it would host a prescriptive bottom-up method, i.e. a decomposition approach that would at the same time provide a well structured estimation procedure.[7] Such a method would combine the advantages of a prescriptive estimation method (repeatable and transparent estimates) with the strengths of bottom-up estimating (reduction of uncertainty and estimation error). In the following, a prescriptive bottom-up estimation approach is proposed.

Empirical investigations justify this research direction: In practice, micro expert judgement and analogy estimation are the most frequently applied estimation methods, top-down algorithmic estimation methods seem to be rarely used.[3, 8, 9] However, especially expert judgements are discredited as subjective because they lack a methodological foundation and do not produce repeatable and comprehensible estimates.[10] It has been frequently suggested that expert judgements, supported by a transparent and repeatable procedure, would be welcomed as the most appropriate estimation technique in practice.[11, 10]

3. PROCESS-ORIENTED EFFORT ESTIMATION

3.1. Fundamental principles

The proposed estimation approach relies on four fundamental principles:

1. *Decomposition:* The estimation model is based on a standard decomposition of the project, used as a template for all estimations. This makes estimates more comparable and saves estimation effort. Decomposition is used because it reduces uncertainty and the potential of gross estimation errors. Also, it is already frequently used in practice. Since at different stages in the project, different levels of decomposition detail may be appropriate, the proposed decomposition structure is variable to accomodate different levels of detail.

2. *Process-orientation:* The approach is based on a decomposition of the software development process, not of the software product, as usual. The reason is that the effort needed to complete the project is not induced by the software product itself, but by the activities that are performed to produce it. Also, the product view would hardly allow a standard decomposition because software products are too heterogeneous. The development process is broken down into workflows and activities for estimation. The component estimates are summed up to yield the total effort estimate. This implies that, as opposed to most algorithmic models, an estimate of the system size is no prerequisite for estimating the project effort; the influence of the size estimate on total effort is reduced significantly. This is an advantage because the estimation of the system size is very difficult, especially in early project phases, and represents the main source of estimation errors in algorithmic estimation models.[3, 12] Furthermore, process-orientation supports project management: Activity-specific estimates enable better pro-

ject control which increases the chances that the project will meet the targets.

3. *Prescriptiveness:* The estimation approach provides a structured estimation procedure and therefore allows to produce transparent and repeatable estimates. Three levels of prescriptiveness are offered.

4. *Expert judgement:* The model essentially relies on expert judgements and analogies. It methodologically supports experts in producing an estimate.

3.2. The estimation model

Figure 2 gives an overview of the estimation approach: The model relies on a "2×3" structure: There are two levels of detail in the decomposition of the software development process, which is the basis of the estimation model. The decomposition defines *what* is to be estimated: The items in the decomposition are the objects of estimation. The estimation method then provides three alternatives *how* these objects can be estimated. These estimation procedures represent three levels of prescriptiveness – basic, intermediate, and advanced.

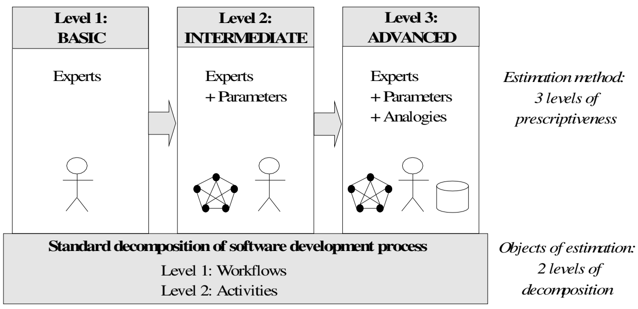

Fig. 2. Model overview

3.2.1. The objects of estimation: Two levels of process decomposition

As shown in Figure 2, the foundation of the proposed estimation approach is the standard decomposition of the software development process. The decomposition defines the objects of estimation (*what* is to be estimated?): In this case process components, i.e. workflows and activities.

During the development of the model, the Rational Unified Process (RUP) will be used as the underlying process model, but generally, any other software development process model could be used.[13]

The RUP represents a standard breakdown of the software development process into workflows and, more detailed, into activities. This standard decomposition is used throughout the estimation model, at varying levels of detail: The first level of the decomposition are the workflows – the RUP defines nine of them for the software development process: Business modeling, requirements, analysis and design, implementation, test, deployment, project management, configuration management, and environment. The second level of detail is the activity level: Each workflow is composed of eight to fifteen activities.

The estimation model incorporates these different levels of detail because during the lifecycle of a project, estimates of differing levels of detail are made: At the very beginning of the project, an estimate can only be very rough, because little detail on the project is known. As the project proceeds, more and more detail is available – if the later project estimates are to be more accurate than the rough up-front estimates, they have to be more detailed. With two levels of decomposition detail, it is possible to estimate those parts of the project on which enough information is available on an activity basis, and the parts that are still rather unclear on a workflow basis. That is, the decomposition and with it the whole estimation model can be used flexibly at all times of estimation.

The result of the decomposition is a table which includes the workflows and activities the software development process is composed of. Depending on the level of detail selected, the objects of the effort estimation are workflows or activities. The estimation methods described in the following section provide the procedures as to how the objects are to be estimated.

3.2.2. The estimation method: Three levels of prescriptiveness

An estimation method prescribes *how* the objects of estimation are to be estimated. The proposed model offers three alternatives to do so, involving increasing degrees of prescriptiveness: A basic, an intermediate, and an advanced estimation method. These three levels build on each other: The intermediate method cannot be used without having already implemented the basic method, and at the same time represents the prerequisite for using the advanced estimation method.

Basic estimation method

The first level is called *Basic*. The items of the decomposition are estimated directly by experts. In Figure 2, the foundation of the method on expert knowledge is symbolized by the stick figure.

The estimating procedure in the basic method is depicted in Figure 3: Based on the standard process decomposition, several experts, independent from each other, estimate the process component effort. They then decide on a consensus estimate for each process component. The consensus component estimates are added up to form the total project effort estimate. This procedure is usually called "Delphi" technique.[1]

An example illustrates the procedure: Assuming the level of decomposition appropriate for the time of the estimation is the workflow level. The decomposition provides nine workflows. The experts estimate how much effort each workflow will involve. For each workflow, the experts discuss their estimates and agree on a consensus estimate. The nine workflow estimates are added up to form the project estimate.

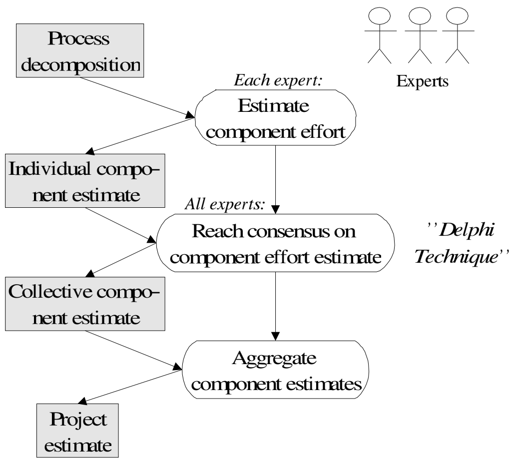

Fig. 3. The basic estimation method

The basic estimation method provides a low degree of prescriptiveness. Structure is only provided by the standard decomposition. Therefore, the method is relatively simple and flexible. At the same time, it provides basic comprehensibility and repeatability due to the standard decomposition. However, the procedure how the experts come up with their component estimates remains intransparent to outsiders. This can be remedied by a sound documentation of the estimation but still represents a weakness of the method.

If a consistent estimation method has never been used at all in an organization, the basic estimation method represents a first step into structured effort estimation that is superior to "guessing". The method is also appropriate for situations where detailed information on the project is not available yet.

The basic estimation method can be used with both levels of detail in the decomposition, i.e. also on an activity-level, so that it is applicable throughout the lifecycle of a project. The more detailed the decomposition, the more accurate the estimates will likely be because the objects of estimation are smaller. But, the quality of the estimates produced with this method always depends on the estimating capability of the experts – the experts have no methodological support beside the decomposition.

If an organization is used to formal estimation approaches or if detailed information on the project is available in an estimating situation, it should be considered to seize this knowledge using a more sophisticated estimation method that provides more comprehensible and repeatable estimates and is apt to produce also more accurate estimates, as described in the following.

Intermediate estimation method

The *intermediate* estimation method improves the support given to the experts when estimating the items of the decomposition. This is achieved in two ways: First, the objects of estimation are structured further: Parameters are identified that drive the effort of the workflows and activities and therefore indicate their "size". Second, a more structured procedure is defined for how the experts' estimates should be treated to form a consensus estimate. In Figure 2, this additional support has been symbolized by a pentagonal structure for the parameter structure.

In the basic estimation method, the experts could only build on their experience when estimating the effort of a given item in the process decomposition. Now, an effort driver is introduced for each process component as an indicator of its size. This parameter is intended to facilitate the mental process of inferring the effort needed for a given component. To be really helpful in the estimation, it is important that the parameter is easier to quantify than the process component as a whole. Therefore, the parameter must either be a measurable quantity, or easier to estimate than the process component itself.

An example illustrates the nature of the effort drivers: Assuming the appropriate level of decomposition in an estimation is the activity level, an activity to be estimated is "Elicit stakeholder requests" in the RUP "Requirements" workflow. To support the estimator in assigning an effort to this activity, a possible "size" parameter is the identified number of stakeholders. The number of stakeholders involved in the project may either be known very early, then the parameter would be measurable. Otherwise, it can relatively easily be estimated by considering the scope of the project. It is assumed that estimating the effort needed for eliciting the stakeholder requests is facilitated if it is known whether the project will involve three or 25 different stakeholders.

Our research on identifying meaningful parameters that drive process component effort and are easier to quantify than the component effort itself is still ongoing. Preliminarily, about 30 parameters have been identified, with one parameter influencing several workflows or activities. Most parameters are of quantitative nature, i.e. amounts like number of stakeholders, number of use cases, number of interfaces. Others are qualitative and have to be assessed by ratings, like personnel experience or the required reliability of the software. All parameters together characterize the project.

However, it is not intended to establish a functional relationship between a parameter and the component effort. The parameters merely serve as hints and it is the task of the estimator to infer the component effort from the parameter value. This has been designed so by intent, in order to make the model applicable to different environments and not to tailor it to a particular empirical data basis.

The estimating procedure, which is more repeatable and comprehensible to outsiders than the "Delphi" technique used above, is illustrated in Figure 4: After the parameters in the decomposition have been quantified, each expert provides a minimum, most likely and maximum effort estimate for each process component, given the "process component size" indicated by the parameter. The three estimates are weighted and aggregated among the experts to form the expected value and the standard deviation for the collective process component estimate. The

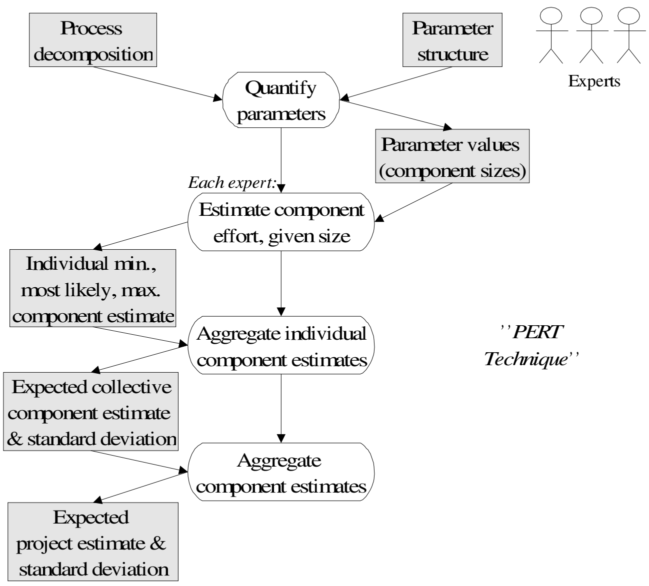

Fig. 4. The intermediate estimation method

collective component estimates are then aggregated, yielding an expected value and the standard deviation for the whole project estimate. The procedure is also known as "PERT" technique [14, 15] and can be easily managed with a spreadsheet prepared in advance.

The improvements over the basic method are the following: The estimation of the process components is facilitated and the component estimate is more comprehensible due to the "size" parameter. Furthermore, an explicit, repeatable procedure is provided to handle diverging expert estimates. Moreover, the final estimate is not a point estimate, but a range indicated by the expected value of the estimate and the standard deviation; this gives an impression of the uncertainty inherent in the estimate.

The intermediate method is more prescriptive than the basic method; it increases the comprehensibility and repeatability of estimates. The method is also more complicated and may involve a little more time for the estimation. It is suited for organizations that desire better repeatability and comprehensibility of estimates. The method can be used with both levels of detail in the decomposition, it is therefore applicable at all stages in the project.

Advanced estimation method

The *advanced* estimation method is a further enhancement of the intermediate method. Whereas it relies on the same structure as the intermediate method (process decomposition and parameter structure), the repeatability and comprehensibility of the estimating procedure are further improved: When estimating the process component effort, the experts are supported by an analogy database. The analogy support in the advanced estimation method is indicated in Figure 2 by the database symbol.

The estimating procedure in the advanced method is depicted in Figure 5: In the beginning, it is equivalent to the intermediate method. The experts estimate the effort of the process components, given the values of the "size" parameters. Whereas in the intermediate method, it was up to the expert how much effort to assign to a process component given its parameter value, this decision is now supported by historical data. The parameter values, the component estimates and actuals, as well as the estimated and actual total effort of completed projects are stored in a database. This means that analogies are provided at the process component (micro) level, not at the project (macro) level as in other estimation techniques. A process component is characterized by its size parameter. The expert searches the database for projects with similar parameter values. The estimated and actual effort of the analogical process component gives an indication of the effort required for the process component to be estimated. The expert's experience is then "only" needed for the decision whether to assign the same, less, or more effort to the new component compared to the historical project.

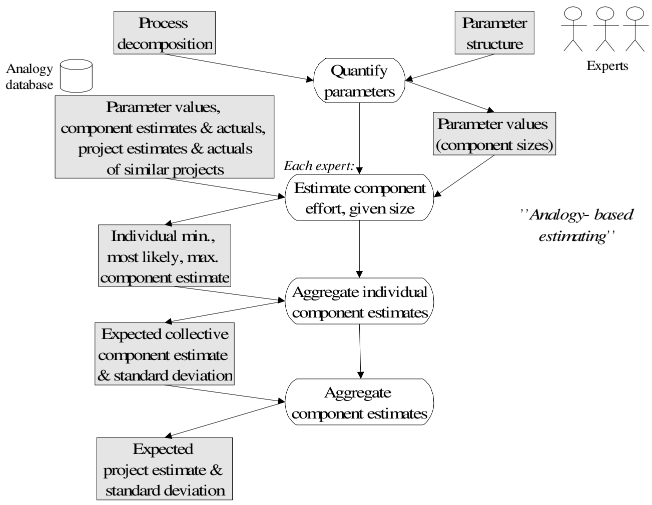

Fig. 5. The advanced estimation method

The remaining steps in the procedure are equivalent to the intermediate method: The experts provide three effort estimates per process component, a minimum, most likely, and maximum estimate, which are aggregated to yield an expected collective component estimate and its standard deviation. The collective component estimates are then aggregated, the result is again an expected project estimate and a standard deviation which indicates the uncertainty in the estimate.

Compared with the intermediate method, the advanced estimation method offers several advantages: The estimation process becomes more transparent and more repeatable by the structured use of analogies because the influence of (subjective) expert experience has been reduced. Further more, the method enables estimators to improve their estimates from project to project because for new estimates, lessons learned from past projects can be seized. Past evidence may also reduce the standard deviation in the final aggregated estimate because it provides the experts with a common starting point.

The advanced estimation method can only be applied when a database with a sufficient number of projects has been established. As long as such a database is not available, the intermediate estimation method can be used: The estimation results achieved with this method are entered in a database; when the project is completed, the actuals are added. The intermediate method is used until a sufficient number of projects has been recorded in the database to allow the application of the advanced method.

The advanced estimation method is the most prescriptive of the proposed methods and offers the best comprehensibility and repeatability. However, this requires preparation. It will take months or even years until the database provides enough analogy projects to really support new estimates. Therefore, the method is appropriate for organizations that desire a structured estimation method and are willing to invest time in establishing it. Again, the method can be applied with all levels of the decomposition.

4. DISCUSSION

The proposed estimation model represents a new approach in effort estimation modeling in several respects.

First, the prescriptive bottom-up estimating approach is unprecedented in the effort estimation domain. The usual decomposition estimating techniques (micro expert judgements) do not use a standard decomposition and do not provide a methodological procedure on how the items in the decomposition should be estimated. FPA uses decomposition only for the size estimate. By providing varying levels of detail in the decomposition, the approach can be applied at different stages in the project. The potential of estimation errors is reduced by both the decomposition and the additive connection between the component estimates. Most top-down algorithmic estimation methods use multiplicative connections between their parameters so that small parameter estimation errors have multiplicative effects on the total estimate.

The second major innovation is process-orientation. It was argued that not the software system to be produced induces the effort, but the activities performed to produce it. Also, process decomposition allows a standard decomposition. Focusing on processes instead of the software product also saves the estimator from estimating the software system size as a prerequisite for an effort estimate. The influence of the software size estimate on the whole effort estimate is significantly reduced – and with it the propagation of errors. Furthermore, the process decomposition enables better project control.

However, this also implies that an implemented software development process model is the prerequisite for applying the approach. Whereas the three estimation methods can be used with any process model, the decomposition and parameter structure have to be adapted if a process model other than the RUP is used. They also have to be adapted in case the development process is modified.

The approach seizes the knowledge of experts and provides a methodology that reduces the subjectivity of the estimates. The "2×3" structure of the model accounts for different estimation requirements and levels of experience in organizations. It allows to establish a structured estimation method from scratch and to enhance it step by step.

The model is theoretically founded, it has not been designed to fit empirical data, as is the case with most algorithmic estimation models. The model therefore is generally applicable in any software development organization, as long as a process model is available. It has been designed to account for the needs of medium-sized software development companies who cannot afford to invest in effort estimation techniques like large organizations.

Apart from the *basic* estimation method, the estimation model provides estimate ranges rather than point estimates, which reveals the uncertainty in the estimate instead of simulating accuracy.

However, to evaluate the usability of the estimation method and the accuracy of the estimates, the proposed estimation approach has to be tested in real projects.

5. CONCLUSION

A new effort estimation approach for custom software development projects has been presented. It provides a methodological framework to support expert judgement and analogy estimating, the most popular estimating techniques in practice, in order to make estimates transparent and repeatable.

The proposed approach is process-oriented and combines bottom-up estimation with a prescriptive estimation procedure. To account for different levels of detail available at different stages in the project, the approach provides two levels of decomposition. To meet different requirements concerning the repeatability and transparency of estimates, three different estimation methods, involving varying degrees of prescriptiveness, are offered.

REFERENCES

[1] Barry Boehm, *Software Engineering Economics*, Englewood Cliffs, 1981.

[2] Martin Shepperd, Chris Schofield, and Barbara Kitchenham, "Effort estimation using analogy," in *Proceedings of the 18th International Conference on Software Engineering (ICSE)*, 1996, pp. 170–178.

[3] Fred Heemstra, "Software cost estimation," *Information and Software Technology*, vol. 34, no. 10, pp. 627–639, 1992.

[4] Andrew Gray and Stephen MacDonell, "A comparison of techniques for developing predictive models of software metrics," *Information and Software Technology*, vol. 39, no. 6, pp. 425–437, 1997.

[5] Barry Boehm et al., *Software Cost Estimation with COCOMO II*, Upper Saddle River, 2000.

[6] Allan Albrecht and John Gaffney, “Software function, source lines of code, and development effort prediction: A software science validation,” *IEEE Transactions on Software Engineering*, vol. SE-9, no. 6, pp. 639–648, 1983.

[7] Barry Boehm and Richard Fairley, “Software estimation perspectives,” *IEEE Software*, vol. 17, no. 6, pp. 22–26, 2000.

[8] Albert L. Lederer and Jayesh Prasad, “Information systems software cost estimating: A current assessment,” *Journal of Information Technology*, vol. 8, no. 1, pp. 22–33, 1993.

[9] Fred Heemstra and Rob Kusters, “Software cost estimation in the Netherlands: 10 years later,” in *Proceedings of the European Software Control and Metrics Symposium (ESCOM-SCOPE)*, 1999, pp. 23–31.

[10] Robert T. Hughes, “Expert judgement as an estimating method,” *Information and Software Technology*, vol. 38, no. 1, pp. 67–75, 1996.

[11] Robert Park, Wolfhart Goethert, and Todd Webb, “Software cost and schedule estimating: A process improvement initiative,” Special Report CMU/SEI-94-SR-3, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, URL: `http://www.sei.cmu.edu/publications/documents/94.reports/94.sr.003.html`, 1994, retrieved 12.11.2001.

[12] Barbara Kitchenham, “Empirical studies of assumptions that underlie software cost-estimation models,” *Information and Software Technology*, vol. 34, no. 4, pp. 211–218, 1992.

[13] Philippe Kruchten, *Der Rational Unified Process*, Reading/Mass., 1999.

[14] Lawrence Putnam and Ann Fitzsimmons, “Estimating software costs (Part 1),” *Datamation*, vol. 25, no. 9, pp. 189–198, 1979.

[15] Martin Höst and Claes Wohlin, “An experimental study of individual subjective effort estimations and combinations of the estimates,” in *Early Evaluation of Software Process Change Proposals*, Martin Höst, Ed. 1999, pp. 103–122, Diss., Department of Communication Systems, Lund University, Sweden, URL: `http://www.tts.lth.se/Personal/martinh/phdthesis.pdf`, retrieved 14.02.2002.

ABOUT THE AUTHORS

Ursula Passing is a doctoral student at the Department of Information Systems at the European Business School at Oestrich-Winkel. She is currently working on a Ph.D. thesis on effort estimation for software projects.

Prof. Dr. Susanne Strahringer is chair of this department. Her research focus covers different topics in the area of object-oriented application development.