Because Effort Matters!

A Mapping Model for Assessing Project Effort in Requirements Engineering

Project effort is critical for the success of software development projects. However, although the requirements have an influence on the resulting effort, requirements engineering (RE) methods are not capable of assessing project effort adequately. We present a mapping model that integrates function point analyses into RE and thereby supports assessing project effort from requirements.

DOI 10.1007/s12599-010-0103-y

The Authors

Frank Zickert M.Sc. (✉)
Prof. Dr. Roman Beck
Chair of Business Administration,
esp. E-Finance and Services Science
Goethe University Frankfurt
Grüneburgplatz 1
60323 Frankfurt
Germany
mail@frankzickert.de
rbeck@wiwi.uni-frankfurt.de

Received: 2009-08-02
Accepted: 2010-03-04
Accepted after one revision by the editors of the special focus.
Published online: 2010-04-23

This article is also available in German in print and via http://www.wirtschaftsinformatik.de: Zickert F, Beck R (2010) Weil Aufwand wichtig ist! Ein Zuordnungsmodell zur Bewertung des Projektaufwands im Requirements Engineering. WIRTSCHAFTSINFORMATIK. doi: 10.1007/s11576-010-0225-3.

1 Introduction

Project effort is a critical factor in software development projects and significantly affected by underlying requirements. There is, for instance, a difference in project effort between the requirements "the system should provide a basic calculator" and "the system should provide full support for company accounting". What is astonishing is the fact that the resulting project effort is not incorporated into requirements engineering (RE). In fact, current RE techniques are not capable of assessing project effort appropriately. While most techniques only assess what the system under construction has to accomplish (van Lamsweerde 2001, p. 250), project effort is conceptually different (Glinz 2007, p. 24). It comprises the work that has to be done in a project (Abdel-Hamid and Madnick 1991, p. 82), which can be measured by the time and number of staff that is needed to complete the project (Albrecht 1979, p. 85; Cheung et al. 1999, p. 278). Project effort is thus critical for answering the question whether the given constraints in time and budget can be complied with. Recognizing the effects requirements have on project effort is essential, because "engineering is not just about solving problems; it is about solving problems with economical use of resources, including money" (Shaw 1990, p. 15).

In this paper, our objective is to incorporate the assessment of project effort resulting from requirements for software development projects into RE by developing a Mapping Model for Assessing Project Effort (MMAPE). It is a mapping of semantics used in the RE method KAOS onto patterns that are counted in function point analyses (FPA). KAOS comprises a requirements notation language that supports requirement elicitation and evaluation in a structured way. FPA provides information on project effort by measuring the system's functional size.

We then apply MMAPE in a software development project for a large financial institution. The results indicate that the integration of measuring system size into KAOS provides useful information for assessing both the satisfaction of system-related goals as well as project effort. By providing additional information, MMAPE increases the utility of KAOS for requirements engineers.

The remainder of this paper is structured as follows: In the Sect. 2 we briefly provide an overview on KAOS and FPA as base for our MMAPE, which is presented in Sect. 3. In Sect. 4, we describe the empirical case study where we applied MMAPE. This section is subdivided into method, description of the case, and results. We conclude with a summary and outlook (Sect. 5).

2 Related Research

The assessment of project effort requires a structured representation of requirements. Requirements are usually stated by stakeholders in natural language (Kotonya and Sommerville 1998, p. 19). In this form they comprise "unstated assumptions that reflect the shared ("common sense") knowledge of people familiar with the social, business and technical contexts within which the proposed system will operate" (Ryan 1993, p. 1). Common sense makes people recognize that there are differences between building a basic calculator and an accounting system. However, analyzing natural language "may be wrong and misleading, not reflecting the actual meaning of the requirement" (Natt och Dag et al. 2002, p. 27). Assessment of project effort is

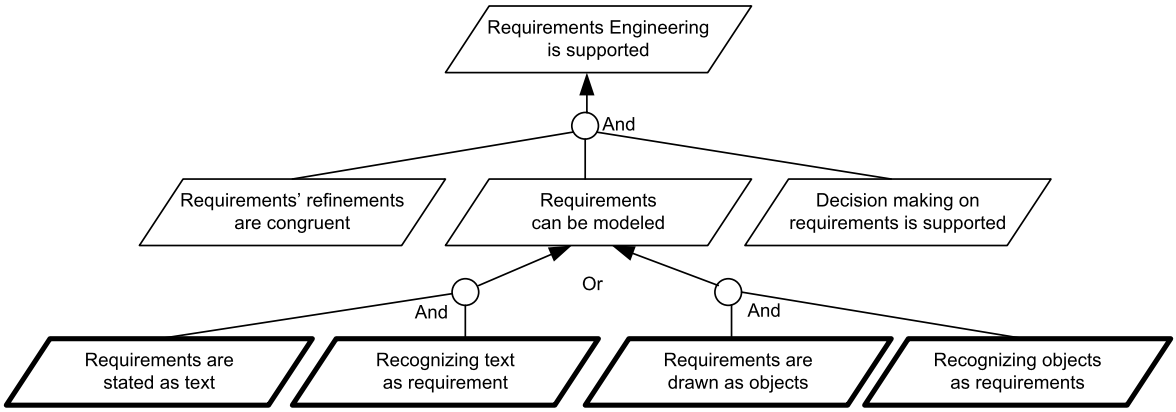

Fig. 1 KAOS goal model

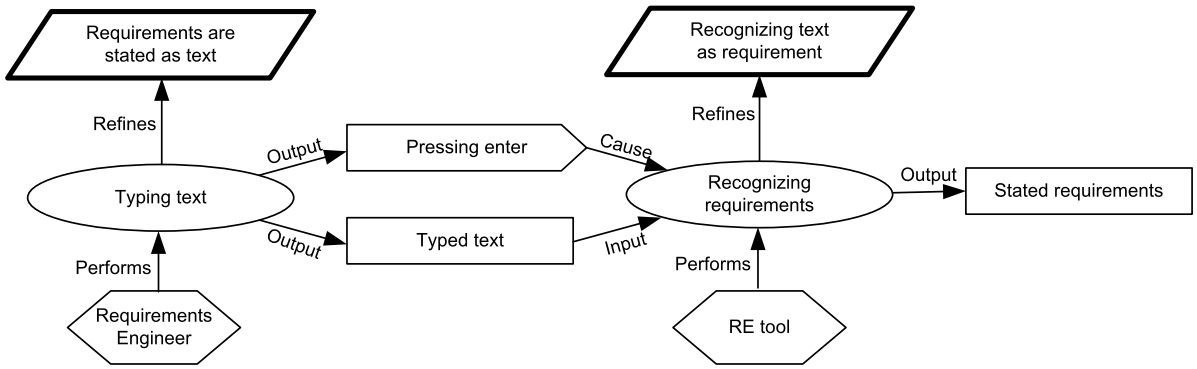

Fig. 2 Derived specifications

thus better based on structured requirements with notation languages, as used in KAOS.

KAOS is one of the most important RE methods supporting requirement elicitation and allows reasoning about goal satisfaction (van Lamsweerde 2004, p. 5). Its notation language comprises an outer semantic layer and an optional inner formal assertion layer (van Lamsweerde et al. 1998, p. 911). The outer layer is used for conceptual modeling of requirements, attributes, and relations of requirements through a graphical syntax. The formal layer utilizes temporal logic (Dardenne et al. 1993) for inferring specifications from requirements (van Lamsweerde and Willemet 1998) and reasoning about system-related goal satisfaction (Letier and van Lamsweerde 2004). MMAPE is based upon the semantic layer since it provides sufficient information for analyzing requirement structures.

KAOS comprises four models that are iteratively prepared: (1) a goal model in which goals to be achieved by the system are described; (2) an object model in which objects involved in the system are described; (3) an agent model in which responsibilities are assigned to agents; and (4) an operation model in which input-output relationships among operationalizations of requirements and identified objects are described (van Lamsweerde 2001, p. 256; Letier and van Lamsweerde 2002, pp. 120–122).

At first, goals are incrementally elaborated and refined by asking "how" and "why" questions (Yu 1997, p. 229). Goals provide precise criteria for sufficient completeness and pertinence of requirements (van Lamsweerde 2001, p. 251). They are the rationale for building the system under construction. **Fig. 1** provides an exemplary goal model in KAOS. Each circle represents a refinement of a goal into sub-goals or requirements, whereby all connected sub-goals need to be satisfied to satisfy the superordinate goal. Thus, it represents an AND-refinement. Different refinements (circles) of one (sub-) goal imply an OR-relationship in which either one or another refinement must be satisfied.

After the goals are elaborated, the object and agent models are prepared. The object model collects objects, attributes, and relationships among them while in the agent model, agents are assigned responsibility for the requirement implementation. Agents and objects are used when the operation model is prepared. In this last step, specifications are derived from requirements. Specifications consist of operations (represented as ovals) and the identified objects, such as events (arrowed rectangle) or entities (rectangles). Operations can be interpreted as a behavior of the system in a specific situation. That situation is determined by events that cause the operation and entities that serve as information input. Similarly, operations may produce either events, entities, or both as outputs. Altogether, this set of objects and relations represent the specifications that need to be fulfilled in order to satisfy the requirements (Letier and van Lamsweerde 2002, p. 121). Each operation must be performed by an agent (hexagon). **Fig. 2** provides an example on two derived specifications.

All goals, requirements, and specifications in KAOS describe the system under construction (van Lamsweerde 2001,

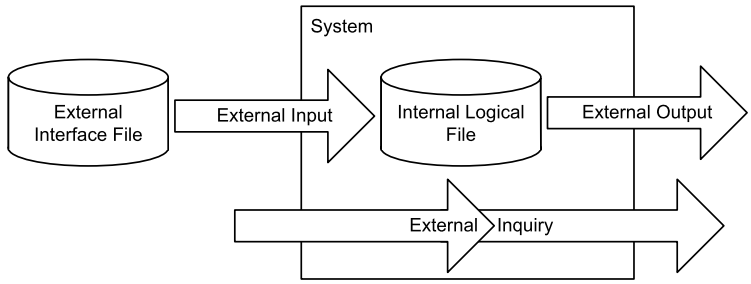

Fig. 3 Function Classes

p. 250). Since project effort is not something the system is about, it cannot be modeled in KAOS. It is a result of performing activities for building the system (Boehm 1984, p. 205). With increasing size of the system, more effort must be spent on the activities (Abdel-Hamid and Madnick 1991, p. 82). For example, all functions of the system must be modeled in KAOS and/or be described in the requirements document. It is more effort to describe all functions of a large system like a stock trading system than it is for a small system like a simple address book. Writing the code for that large system is also more laborious than it is for the smaller one. Consequently, system size is often measured in either functionality or lines of code (Heemstra 1992, p. 631; Gencel and Demirors 2008, p. 15:3). Since requirements describe the functions of a system (Kotonya and Sommerville 1998, p. 3), assessing effort that results from them is best based on size in terms of functionality. Functional size can be measured by FPA, which was initially developed by Albrecht (1979), and is constantly updated by the International Function Point Users Group (IFPUG).

Counting function points (FP) is primarily based on identification of function classes in the software's logical design (IFPUG 1999, p. 2–2). There are two major function classes: transactional functions and data functions. Transactional functions represent external inputs (EI), external outputs (EO), and external inquiries (EQ). Data functions in FPA are either internal logical files (ILF) or external interface files (EIF).

An EI is an input that originates from the user or another software component outside the counted system boundary. It may also use data from outside the system boundary (EIF) and it updates data inside the system boundary (ILF). For instance, the user entering data into the system is an EI. The entered data becomes an ILF when it is stored in the system. If the data is not entered by a user but sent from another system, the function is still an EI. The data also becomes an ILF when it is received and stored by the system. However, in this case, the origin of the EI is an EIF, which is the file saved at the other system when it is sent.

An EO works the other way round. An EO is an output that originates within the boundary of the counted system and uses data (ILF) from inside the system that is transmitted to a user or another software component outside the system. In the example, the system forwarding the received file to another system is an EO. It uses the data stored in the system (ILF) for sending data outside the system boundary.

An EQ is an online input that results in an intermediate software response. Its primary intent is presenting information to a user through retrieval of data. It does not update an ILF but provides a direct response outside the system boundary.

An ILF is a logical group of data that resides within the boundary of the software under construction. An EIF is a logical group of data that resides outside the software boundaries and that provides data for the software.

The IFPUG manual (IFPUG 1999) provides a more detailed description of these structures. **Fig. 3** graphically illustrates the five major function classes. **Table 1** summarizes the commonly used abbreviations.

Each identified function is assessed with a complexity factor. The determination of complexity is subjective and often guided by criteria developed from experience (Abran and Robillard 1994, p. 180). These criteria are likely to be specific to the individual organization or even department. Complexity in this context is classified as "simple", "average", or "complex". Once the complexity is assessed, each function is weighted with a factor that depends on the function class and the complexity. **Table 2** provides the weighting factors in the form of a calculation sheet that is frequently used for counting FP (Kemerer 1993, p. 86).

3 Mapping Model for Assessing Project Effort

MMAPE incorporates FPA into KAOS in two steps. Firstly, function classes counted in FPA are identified in a KAOS model. For this purpose, we define patterns in KAOS that map onto FPA function classes. Secondly, complexity is assessed for each identified function by measuring number and distance of objects and operations involved in the respective function. **Fig. 4** summarizes how MMAPE counts FP from KAOS models. The steps are more thoroughly explained in the following.

In FPA, all function classes are identified primarily based on the software's logical design (IFPUG 1999, p. 2–2). This design is represented in the KAOS operation model that contains the specifications that determine the behavior of the system under construction (Letier and van Lamsweerde 2002, p. 121; Jackson and Zave 1995, p. 15). Firstly, the system boundary must be defined. Since RE techniques such as KAOS are rather used for large composite systems where applications or functions need to interact to deliver functionality for the user (IFPUG 1999, p. 5–5), it is reasonable to regard each function within this composition separately. Functions in this sense are represented by operations in KAOS, because these describe the behavior of the composite system in specific situations. Thus, generally, an operation's perspective is applied for identification of function classes in KAOS models.

From an operation's perspective, transactional functions refer to the operation's input/cause or output connections. Data functions refer to objects (entities or events) that are connected to the operation by the transactional functions.

An EO is an output connection from the operation. It connects to an object that is an ILF for the operation. The operation determines the state of that object. An ILF thus resides within the system boundaries. Moreover, the respective output connection is not yet an EO unless the connected ILF serves as the input/cause for another operation.

Similarly, an EI is an input connection to the operation. It connects from an object that is an EIF for the operation. An

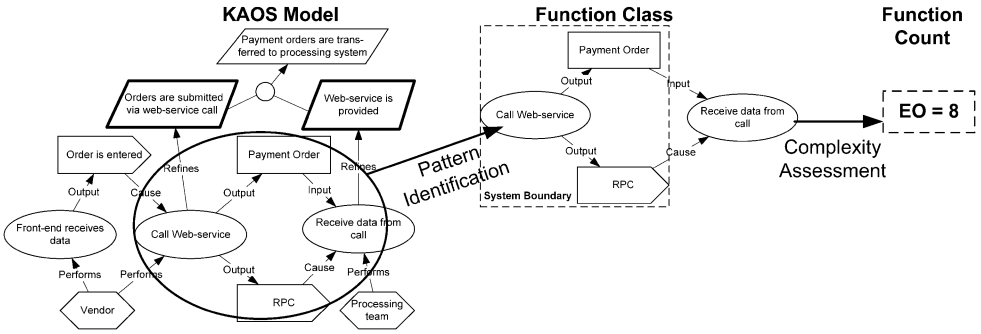

Fig. 4 Steps of MMAPE for counting FP from KAOS models

Table 1 Abbreviations

EI	External Input	Input that originates outside the system and is used for updating an ILF
EO	External Output	Output created by the system and transmitted to an outside component
EQ	External Inquiry	Online input that originated outside the system and results in intermediate system response
ILF	Internal Logical File	Logical group of data that resides within the system
EIF	External Interface File	Logical group of data that resides outside the system but is used by the system

Table 2 Weighting factors used in FPA

	Simple	Average	Complex
External Inputs (EI)	3	4	6
External Outputs (EO)	4	5	7
External Inquiries (EQ)	3	4	6
Internal Logical Files (ILF)	7	10	15
External Interface Files (EIF)	5	7	10

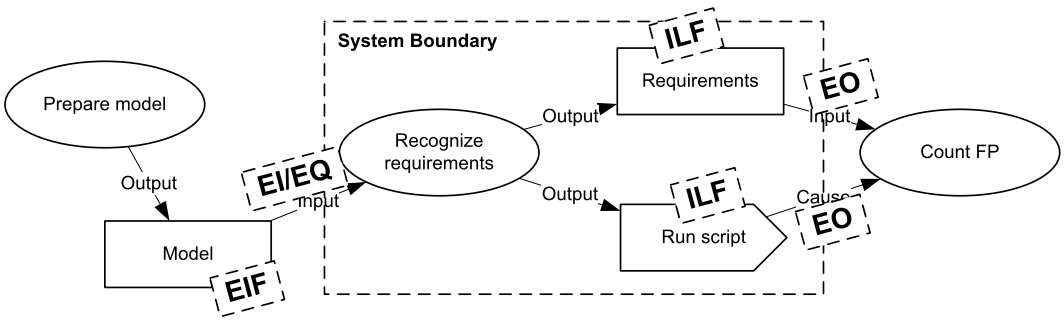

Fig. 5 Patterns

EIF is an object that originates from outside of the operation. Consequently, an additional condition for the input connection to be counted as EI is that the connected object is an output of another operation.

Although it is recommended to distinguish between EQ and EI, the structures that are counted are similar. Both refer to input connections to the operation. The major difference is whether an ILF is updated (EI) or not (EQ). An EQ would require the operation to provide an output response directly to another operation. Since direct connections between different operations are not allowed in KAOS, there is no equivalent to EQ and thus, they cannot be distinguished from EI. However, since both EQ and EI are treated equally in FPA (use of the same weighting factors (Kemerer 1993, p. 86)), MMAPE does not differentiate between both function classes. **Fig. 5** provides a graphical representation of all the patterns.

Having identified patterns that need to be counted, the next step is assessing their complexity. While this is a rather subjective assessment in FPA (Abran and Robillard 1994, p. 180), the KAOS models provide additional information about the context of operations and objects. The complexity of building a system is determined by the interrelations among its elements (Campbell 1988, p. 42). In KAOS models, these interrelations are explicitly described. Moreover, when building the system, effort results from the coordina-

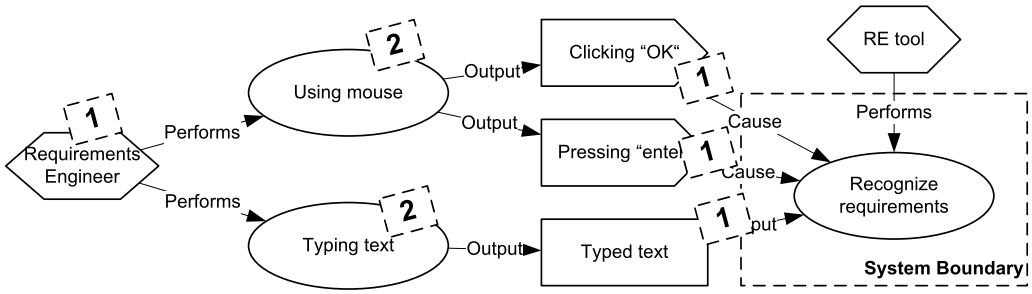

Fig. 6 Example of EI

tion that is required for coping with the interrelations (McCann and Ferry 1979, p. 116).

We measure this coordination complexity for transactional functions (EI/EQ, EO) using two dimensions: the number of interrelations and the distance in terms of division of labor (Marengoa and Dosi 2005, p. 305). The number can be easily assessed by counting entities and events that are connected with the operation. We weight each connected entity or event with the value 1.

Division of labor is a simple two-level construct. The first level is functional division (e.g., in KAOS, each operation serves a function). Coordination of functions becomes increasingly complex with increasing number of functions (Mihm et al. 2003, p. 747). It is of secondary interest whether the functions exchange more or less data. For instance, coordinating the interplay of five functions that share small pieces of data is more complex than two functions that exchange a bunch of data. In MMAPE, each operation that is connected with the counted operation is weighted with 2, regardless of the number of objects through which the operations are connected to each other.

The second level is organizational division. In KAOS, each operation is assigned to a stakeholder who is responsible for its implementation. If two related operations are assigned to the same stakeholder, there is no increased coordination complexity. Changing the interface between two operations can be done solely by concentrating on the functional issues. On the contrary, operations that are related across boundaries of stakeholders' responsibility need to be explicitly coordinated. Depending on the stakeholders this may involve different understandings, as between business and IT departments, different languages, or even different cultural habits as in offshore relationships. We weight each different stakeholder responsible for a related operation with 1, since in the case that we present in Sect. 5 all stakeholders are located within a country and share the same language and culture. This value may be increased in other settings.

Fig. 6 provides an example of counting an EI that comprises two events and one entity. The objects result from two different operations but from one stakeholder. The total count is: $1 + 1(\text{events}) + 1(\text{entity}) + 2 + 2(\text{operations}) + 1(\text{other stakeholder}) = 8$ in total.

For data function complexity assessment, we rely on the notion of overlapped sets of data elements. This means with an increasing number of operations using or determining values of a data element, the less flexible this construct is and thus, the more coordination is required for its implementation. For instance, there is no overlap for an object that is determined by an operation but not used by any other operation. It does not matter if the values within that object change. On the contrary, an object that is used by many other operations is very constricted. If any of these operations requires a value to change, it will have consequences for all operations that also use this value.

We weight each operation that is connected with the data with a value of 1 and each stakeholder with 2. Here, all operations and stakeholders are counted. For instance, data that is touched by three operations that two different stakeholders are responsible for has a complexity of $3 \times 1 + 2 \times 2 = 7$. This value is added to each connected operation's data function count, either as ILF or as EIF.

Finally, the FP counts of EI/EQ, EO, ILF, and EIF are summed for each operation.

4 Application of MMAPE in a Software Development Project

4.1 Method

We experimentally applied MMAPE in a software development project within a large financial institution to assure that it is applicable for a real-world problem and to illustrate that the information gathered is useful for assessing project effort. Various stakeholders were involved in the project. Each stakeholder stated different goals. This offered us the opportunity to calculate and evaluate modularized components that stakeholders requested that others be responsible for.

In the software development project at hand, we observed two situations in which alternative requirements were negotiated among stakeholders, because one stakeholder rejected requirements that another supported and vice versa. We concentrated evaluation on these two situations, because these were extensively analyzed by stakeholders in the project for argumentation and, consequently, the most detailed information could be gathered for these. Sources of information included interrogations, attendance of meetings, and access to e-mails, concepts, and decision documents. Data was both partly quantitative and partly qualitative. For increasing the reliability of our data, we reduced all data to a qualitative assessment of which discussed alternatives better satisfied each stated goal.

We base evaluation of MMAPE on its correct assessment of which of discussed alternatives better satisfies stated goals. For this purpose, we compare deductions of the alternative that better satisfies goals in a situation, which are based on information gathered using MMAPE on the one hand and observations we made in the case on the other hand. Since MMAPE utilizes KAOS models that

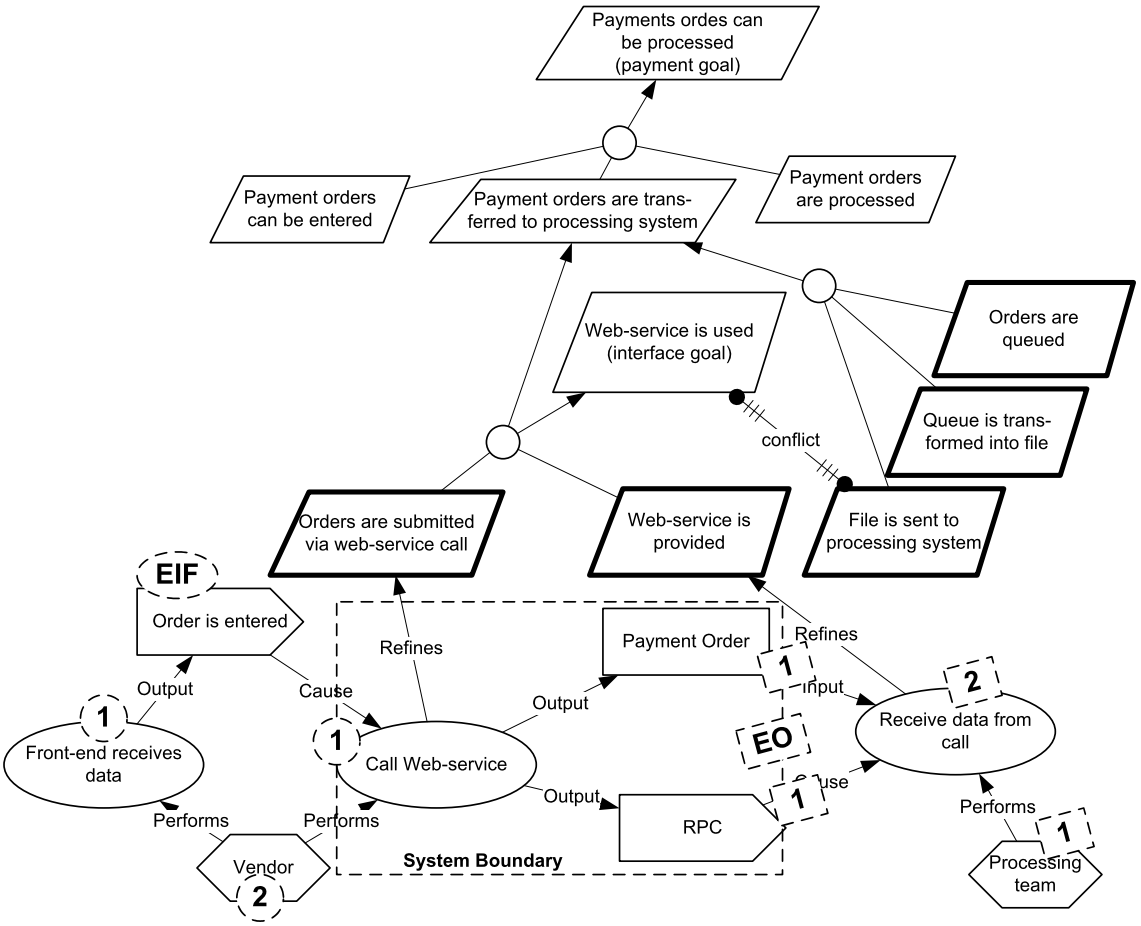

Fig. 7 Web-service alternative

represent the system under construction, it is supposed to correctly assess satisfaction of system-related goals. Moreover, MMAPE is also supposed to correctly assess satisfaction of goals that refer to compliance with given constraints of time and budget, since the integrated FPA provides information on functional size and thus, project effort (Albrecht 1979, p. 85), development time (Cheung et al. 1999, p. 278), and cost (Boehm 1984, p. 212), MMAPE.

4.2 The Software Development Case

We were able to investigate a software development project within a large financial institution. The project's purpose was to integrate an existing front-end system with a recently built payment processing system. Before the project started, payment order entering was already offered at the front-end. These orders were transferred by a routing system to a legacy processing system. Since the legacy system was intended to be deactivated, it has been requested that entered orders be routed to the recently built processing system. In addition, some front-end modifications need to be made to comply with the order format used by the new processing system that differed from the old format. Since order entering at the front-end was re-designed on this occasion, additional goals were stated. Firstly, entered orders should be checked for correctness before they were accepted by the front-end system. Secondly, to minimize maintenance costs in the future, the front-end interface to back-end systems should be upgraded to a service-oriented architecture (SOA).

Due to the organization's size and the inter-departmental setting of that project, various stakeholders were involved. The goals of the internal customer (*customer*) have been enabling payment entering at the front-end (*payment goal*) and checking for correctness (*correctness goal*). Responsible for front-end function delivery is the front-end system team (*front-end team*). This team also set the goal of upgrading the interface to SOA (*interface goal*). Although not explicitly stated by the *customer*, *performance-goals* have also been added by the *front-end team*, because the *front-end team* expected the *customer* to not accept a system which lacks sufficient performance. Project lead was within the *front-end team*. Interaction with the processing system team (*processing team*) was required. Both teams are part of the organizational information technology (IT) department. Moreover, some front-end components are provided by an external vendor (*vendor*) as customized standard software.

Since the deadline for the legacy system deactivation was set for August 2009, the project had a fixed time constraint. The budget had been fixed prior to the project start based on an initial feasibility analysis carried out in 2008. Moreover, the budget was allocated to all three teams responsible for development: the *front-end team*, the *processing team*, and the *vendor*. Since all teams were responsible for compliance with their respective budget constraints, each team stated a goal for limiting effort (*front-end effort goal, processing*

Table 3 Web-service

Operation	EI/EQ	EO	ILF	EIF	Total	Stakeholder	Comment
Frontend receives data	0	4	8	0	12	Vendor	
Call webservice	4	5	12	8	29	Vendor	
Receive data from call	5	4	8	12	29	Processing team	
Process order	4	4	6	8	22	Processing team	

Table 4 Routing system

Operation	EI/EQ	EO	ILF	EIF	Total	Stakeholder	Comment
Frontend receives data	0	4	8	0	12	Vendor	
Send order	4	5	12	8	29	Vendor	
Receive and collect order	5	3	4	12	24	Front-end team	
Create and send file	6	5	12	8	31	Front-end team	Existing
Route file	5	5	12	12	34	Front-end team	Existing
Process order	5	4	6	12	27	Processing team	

Table 5 Result of the web-service vs. routing system situation

Goal	Information provided by MMAPE			Data from the case
	Web-service	Routing system	Suggestion	
Payment goal	Satisfied	Satisfied	Even	Even
Interface goal	Satisfied	Unsatisfied	Web-service	Web-service
Front-end effort goal	0 FP	24 FP	Web-service	Web-service
Vendor effort goal	41 FP	41 FP	Even	Even
Processing effort goal	51 FP	27 FP	Routing system	Routing system

Table 6 Result of the download vs. online situation

	Information provided by MMAPE			Data from the case
	Download	Online	Suggestion	
Correctness goal	Satisfied	Satisfied	Even	Even
Vendor effort goal	43 FP	34 FP	Online	Online
Performance goal	Satisfied	Partly satisfied	Download	Download
Front-end effort goal	152 FP	150 FP	Even	n/a

effort goal, vendor effort goal). The project started in January 2009 and was finished within time and budget. However, due to the criticality of the old processing system's deactivation, time reserves that had been included in the initial project schedule were completely consumed.

Firstly, requirements on how to transfer orders to the processing system were created. Since the *front-end team* was responsible for both designing the front-end connection to the processing system and was a major stakeholder of the interface-goal, requirements and the derived design specifications were created that described a web-service interface between the front-end and the processing system. Subsequently, an analysis was conducted and the *processing team* rejected the requirements regarding a web-service. The p*rocessing team* did not accept responsibility for developing this new interface. They invoked their core competency to be processing issues not routing issues. The *processing team* proposed using the routing system that was currently in place between the front-end and the legacy processing system. The *front-end team* rejected that alternative by pointing out the unsatisfied *interface goal*.. The conflict was resolved by an architectural board decision to use the routing system.

For satisfaction of the *correctness-goal*, the recipient bank code entered at the front-end needed to be checked with existing bank codes. With regard to *performance-goals*, the *front-end team* stated requirements for downloading information about existing bank codes to the front-end. The *vendor* rejected that requirement by proposing an online interface that would check bank codes on demand. The *front-end team* did not accept that rejection. Although some performance concerns were discarded, it remained unclear whether the current performance level would have been reached using the online interface. The *front-end team* decided to implement the download alternative and enforced that decision by pointing out being the client who pays for the implementation.

4.3 Results of Applying MMAPE

We prepared KAOS models for all alternatives in both situations. These models served as the basis for assessing the satisfaction of each system-related goal.

Abstract

Frank Zickert, Roman Beck

Because Effort Matters!

A Mapping Model for Assessing Project Effort in Requirements Engineering

Project effort is critical for the success of software development projects. It has a major impact on whether constraints in time and budget can be complied with. But although requirements affect project effort, requirements engineering (RE) methods are not capable of assessing project effort.

In this paper, we present our mapping model for assessing project effort (MMAPE). MMAPE incorporates into RE the assessment of project effort resulting from requirements for software development projects. It maps semantics of the RE method KAOS onto structures that are counted in function point analyses. We applied MMAPE in a case study on a software development project within a large financial institution. The validity of MMAPE is supported, since we found throughout consistent statements between information provided by MMAPE and data gathered from the case.

Keywords: Requirements engineering, Project effort, KAOS, Function Point Analysis

For example, the *payment goal* was refined into three sub-goals. The discussion about web-service vs. use of routing system was about refining the sub-goal on transferring orders to the processing system. Only the web-service alternative was suited for refining the *interface goal*, whereas using the routing system conflicted with it. **Fig. 7** provides an excerpt of the goal and operation model with an exemplary FP count for an EO and an EIF of the *call web-service* operation. Here, EO counts are represented as rectangles, EIF counts as circles.

Table 3 provides the counts for each operation of the web-service alternative. It also includes information on responsible stakeholders. **Table 4** provides the counts for the alternative set of requirements for use of the routing system. Here, it has to be recognized that *create and send file* and *route file* operations were already implemented and had previously been used to transfer orders to the old processing system.

Table 5 provides information on the satisfaction of each goal as assessed using MMAPE, a suggestion based on this as to which alternative better satisfied the goal, and the respective observation in the case.

In the prepared models, the *payment goal* was satisfied with both alternatives. There were no obstacles which would have inhibited goal satisfaction. The observations support this. The routing system was implemented and proved that the goal was satisfied. Although the web-service alternative had not been implemented, there was well specified documentation on that alternative in the project that also did not disclose any obstacles.

The *interface goal* was only satisfied in the web-service alternative, because this goal explicitly mentioned use of a web-service which was not used in the file transfer via the routing system. The suggestion provided by MMAPE also complied with the observation regarding the *front-end effort goal*. While the *front-end team* would not have had any development task in the web-service alternative, it had to redesign the interface to the routing system.

MMAPE provides clear statements regarding the *vendor effort goal* and the *processing effort goal*. Assuming that stakeholders prefer the alternative that best satisfies their goals (Robinson 1990, p. 270; Simon 1996, p. 29), we find support in our observations here, as well. Due to similar effort, there was no reason for the *vendor* to reject any of both alternatives. The *processing team* had a preference for the routing system and rejected the web-service. This supports the model's suggestion that use of the routing system resulted in significantly less effort for the *processing team*.

Table 6 provides the same information for the situation of download vs. online. The download alternative had been implemented and proven to satisfy both the *correctness goal* and the *performance goal*. For the online alternative, design documents from the case suggested that the *correctness goal* would have been satisfied. If there was any doubt in the project concerning this, we would have particularly expected the *front-end team* or the *customer* to raise respective concerns. That was not the case. In fact, it was even approved by the *front-end team* who however rejected that alternative because it did not satisfy the *performance goal*. We find that this observation supports results from our models. However, it has to be noticed that the conceptual KAOS model did not provide a clear statement regarding the *performance goal*. Thus, we utilized the formal layer for an in depth investigation. We recognized that formal derivation of specifications from the *correctness goal* required some assumptions about the front-end use. The verification of these assumptions with data from the project disclosed that the *performance goal* was satisfied for average usage but not for high peaks. Finally, data on satisfaction of the *front-end effort goal* was not available in the case and thus, respective information provided by MMAPE cannot be evaluated.

We found consistent statements between information provided by MMAPE and data gathered from the case about which alternative better satisfied each goal for all goals for which we had data.

5 Summary and Outlook

In this paper, we present MMAPE, a mapping model of the semantics of the RE method KAOS onto structures that are counted in FPA. Using measuring size in terms of functionality of the system under construction, it provides the basis for assessing project effort. Since most RE techniques are not capable of assessing project effort appropriately, our mapping model adds to RE by filling this gap.

We applied MMAPE on a software development project within a large financial institution. Thereby, we illustrate that it is applicable and provides meaningful information. This information complied with observations we made in our case on both system-related goals and project effort. Moreover, by using MMAPE, we were able to gather information on the *front-end effort goal*, whereas there was no information within the case.

MMAPE contributes to the field of RE by increasing the utility of KAOS. It provides information for assessing project effort that otherwise would need to be gathered separately. We made observations of stakeholders whose major interest was compliance with given constraints in time and budget. They rejected requirements which would have ended up in higher effort, although the requirements satisfied system-related goals. We find rejection by stakeholders understandable because software development is not only about solving problem, but about solving problems with economical use of resources (Shaw 1990). Requirement rejection resulted in negotiations among stakeholders about which requirements had to be used. In situations where stakeholders disagree on requirements, the RE process becomes inevitably political (Bergman et al. 2002, p. 158). Each stakeholder tries achieving his own set of goals (Robinson 1990, p. 270). Information on project effort, as provided by MMAPE, supports the requirements engineer selecting requirements that are not rejected due to unforeseen effort. Thereby, it supports engineers in their task, which is not overcoming resistance, but avoiding it (Markus 1983, p. 441).

Our study has some limitations that future work needs to address. Firstly, although we initially validated our mapping model, further empirical validation is required for assuring reliability and precision of the measures used for assessing project effort. Secondly, we did not explicitly deal with non-functional requirement effects on project effort. Although these can be modeled in KAOS, our mapping does not take into consideration whether they have any specific effect on effort. Finally, it has to be noticed that FPA is primarily intended for assessing interactive software. Depending on the type of system under construction, there may be other suited techniques for assessing project effort. However, there are three major arguments for assessing project effort via integrating KAOS and FPA: (1) both methods are well established; (2) both methods rely on simple constructs, which limits the number of assumptions required for integration; and (3) requirements describe functions of the system under construction and FPA assesses project effort based on measured size in terms of functionality. Thus, there is a conceptual fit of what is being measured.

Acknowledgements

This work was developed as part of a research project of the E-Finance Lab at Goethe University Frankfurt. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the E-Finance Lab or its supporting partners. The authors are indebted to the participating universities and gratefully acknowledge all support of their industry partners.

References

Abdel-Hamid T, Madnick S (1991) Software project dynamics. Prentice Hall, Englewood Cliffs

Abran A, Robillard PN (1994) Function points: a study of their measurement processes and scale transformations. J Syst Softw 25:171–184

Albrecht AA (1979) Measuring application development productivity. In: Proc IBM App Dev Symp

Bergman M, King JL, Lyytinen K (2002) Large-scale requirements analysis revisited: the need for understanding the political ecology of requirements engineering. Requir Eng J 7(3):152–171

Boehm BW (1984) Software engineering economics. IEEE Trans Soft Eng 10(1):4–21

Campbell DJ (1988) Task complexity: a review and analysis. Acad Manage Rev 13(1):40–52

Cheung Y, Willis R, Milne B (1999) Software benchmarks using function point analysis. Benchmarking Int J 6(3):269–279

Dardenne A, van Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. Sci Comput Program 20(1–2):3–50

Gencel C, Demirors O (2008) Functional size measurement revisited. ACM Trans Softw Eng Methodol 17(8):15–36

Glinz M (2007) On non-functional requirements. In: Proc IEEE Joint Int Conf Req Eng, Delhi

Heemstra FJ (1992) Software cost estimation. Inform Softw Techol 34(10):627–639

IFPUG (1999) Function point counting practices manual release 4.1. International Function Point Users Group, Westerville

Jackson M Zave P (1995) Deriving specifications from requirements: an example. In: Proc 17th Int Conf Softw Eng (ICSE'95). ACM Press, New York, pp 15–24

Kemerer CF (1993) Reliability of function points measurement. Commun ACM 36(2):85–97

Kotonya G, Sommerville I (1998) Requirements engineering: processes and techniques. Wiley, New York

Letier E, van Lamsweerde A (2002) Deriving operational software specifications from system goals. In: Proc 10th ACM, SIGSOFT Symp Found Softw Eng, Charleston

Letier E, van Lamsweerde A (2004) Reasoning about partial goal satisfaction for requirements and design engineering. ACM SIGSOFT Softw Eng Notes 29(6):53–62

Marengoa L, Dosi G (2005) Division of labor, organizational coordination and market mechanisms in collective problem-solving. J Econ Behav Org 58:303–326

Markus ML (1983) Power, politics, and MIS implementation. Commun ACM 26(6):430–444

McCann JE, Ferry DL (1979) An approach for assessing and managing inter-unit interdependence. Acad Manage Rev 4(1):113–119

Mihm J, Loch C, Huchzermeier A (2003) Problem-solving oscillations in complex engineering projects. Manage Sci 49(6):733–750

Natt och Dag J, Regnell B, Carlshamre P, Andersson M, Karlsson J (2002) A feasibility study of automated natural language requirements analysis in market-driven development. Requir Eng 7(1):20–33

Robinson WN (1990) Negotiation behavior during requirement specification. In: Proc 12th Int Conf Softw Eng, Nice

Ryan K (1993) The role of natural language in requirements engineering. In: Proc IEEE Int Symp Req Eng, San Diego

Shaw M (1990) Prospect for an engineering discipline of software. IEEE Softw 7(6):15–24

Simon HA (1996) The sciences of the artificial. MIT Press, Cambridge

van Lamsweerde A (2001) Goal-oriented requirements engineering: a guided tour. Invited Paper. In: 5th IEEE Int Symp Req Eng, Toronto

van Lamsweerde A (2004) Goal-oriented requirements engineering: a roundtrip from research to practice. In: 12th IEEE Int Req Eng Conf, Kyoto

van Lamsweerde A, Willemet L (1998) Inferring declarative requirements specifications from operational scenarios. IEEE Trans Sofw Eng 24(12):1089–1114

van Lamsweerde A, Darimont R, Letier E (1998) Managing conflicts in goal-driven requirements engineering. IEEE Trans Softw Eng 24(11):908–926

Yu E (1997) Towards modelling and reasoning support for early-phase requirements engineering. In: Proc 3rd IEEE Int Symp Req Eng