Expert Systems with Applications 39 (2012) 406–418

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A rule-based approach for estimating software development cost using function point and goal and scenario based requirements

Soonhwang Choi [a,1], Sooyong Park [b,1], Vijayan Sugumaran [c,d,*]

[a] DMC R&D Center, Samsung Electronics, Suwon, Gyeonggi-do 443-742, Republic of Korea
[b] Department of Computer Science, Sogang University, Seoul 121-742, Republic of Korea
[c] Department of Decision and Information Sciences, School of Business Administration, Oakland University, Rochester, MI 48309, United States
[d] Department of Service Systems Management and Engineering, Sogang Business School, Sogang University, Seoul 121-742, Republic of Korea

ARTICLE INFO

Keywords:
Project management
Requirements triage
Cost estimation
Function point
Goal
Scenario

ABSTRACT

Function point is a method used to measure software size and estimate the development cost. However, for large complex systems, cost estimation is difficult because of the large number of requirements expressed in natural language. In this paper we propose a rule-based approach for estimating software development cost in the requirements analysis phase. It combines goal and scenario based requirements analysis with function point based cost estimation. In our proposed approach, Context Analysis Guiding rules, Data Function Extraction Guiding rules, and Transaction Function Extraction Guiding rules have been developed to identify function points from text based goal and scenario descriptions. These rules are established based on a linguistic approach. The contribution of the proposed approach is to help project managers decide which requirements should be realized.

1. Introduction

Developing software systems that meet stakeholders' needs and expectations is the ultimate goal of any software provider seeking a competitive edge. To achieve this, we must effectively and accurately manage our stakeholders' system requirements: the features, functions, and attributes they need in their software system (Davis, 1993; Şen & Baraçlı, 2010). Once we agree on these requirements, we can use them as a focal point for the development process and produce a software system that meets the expectations of both customers and users. However, in real world software development, there are usually more requirements than we can implement given stakeholders' time and resource constraints. Thus, project managers face the following dilemma: how to select a subset of the customers' requirements and still produce a system that meets their needs (Karlsson & Ryan, 1997)?

Software requirements triage is the process of determining which requirements a product should satisfy given the time and resources available (Davis, 2003). The practice of triage increases the likelihood that a product will meet customers' needs, and thus contributes significantly to the economic impact of that product on the company's bottom line. Yet despite the potential benefits of requirements triage, not much research has been conducted, and the descriptions that do appear are brief (Davis & Zweig, 2000; Leffingwell & Widrig, 2000; Wiegers, 1999; Yourdon, 1997). This is because triage is a difficult task, fraught with political and financial dangers—politically dangerous because both technical and marketing personnel claim the tasks as part of their responsibility; financially dangerous because a mistake could trigger a major loss of revenue. Davis (2003) has studied the requirements practices of approximately 100 companies and organizations over 25 years. He reports that one of the 14 key recommendations is to contain "Cost". Boehm (1981) and Boehm and In (1996) argues that project managers should be able to manage the impact of changing requirements on software cost and schedule. Therefore we need to identify the scope of project and develop an appropriate plan for project at the initial phase of software development.

Recently, some proposals have been made for estimating cost from requirements. Larvet and Vallée (2002) have defined an estimator based on the information available at the requirements stage, mainly the text of requirements. They advocate a set of textual metrics that could be predictors; it includes four distinct metrics, namely, TNW, NKW, AUTO and MANU. It is possible to quantify the complexity of a project through simple linguistic metrics that are directly issued from the requirements. Auer, Becker, Rauber, and Biffl (2005) proposed a transparent way of visualizing the similarity of use cases without the need for explicit data collection; an implicit analogy metric is obtained by using textual similarity. Lavazza and Valetto (1981) have presented a case study that

* Corresponding author at: Department of Decision and Information Sciences, School of Business Administration, Oakland University, Rochester, MI 48309, United States. Tel.: +1 248 370 2831/+82 2 705 8845.
E-mail addresses: soonhwang.choi@samsung.com (S. Choi), sypark@mail.sogang.ac.kr (S. Park), sugumara@oakland.edu (V. Sugumaran).
[1] Tel.: +82 2 705 8928.

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.07.029

aims at quantitative assessment of the impact of requirements changes, and estimation of the cost of development activities that must be carried out to accomplish those changes. They adopted a very simple approach to requirements quantification: they just counted the labels placed on textual requirements for identifying them and tracing them to design and code elements.

Although there are several different approaches for estimating software project efforts (Li, Xie, & Goh, 2009; Park & Baek, 2008; Pendharkar, 2010), few of them are actually applied successfully in typical industrial environments. One of the main reasons is that each estimation technique usually relies on its own unique way that is subjective. They need to be combined with general cost estimation method like function point. However, they are not very conducive to be formally used with general cost estimation methods, and do not provide any methodological guidelines for how to estimate cost from requirements. Finally, these approaches do not directly help project managers in selecting a sub set of the requirements for implementation.

In this paper, we propose a rule-based approach to count function point from textual requirements through goal and scenario based linguistic approach (Kim, Park, & Sugumaran, 2004, 2006). Thus, our main objective is to develop a methodology for counting function point from textual requirements. Two characteristics of the proposed approach contribute to the achievement of this objective.

First, textual requirements are specified in terms of goal and scenario. We adopt goal and scenario based approach for requirements engineering from prior work (Kim et al., 2004, 2006). The second characteristic of the approach is the concept of extraction rules to count function point. These rules guide the user to count function point. There is a key notion embodied in the extraction rules. The idea is that the goal and scenario at the interaction level describes the interaction between user or external application and the target application. It includes data for interaction, which can be used to derive *data functions*, and behavior for data processing that leads to *transaction functions*.

The rest of the paper is organized as follows. Section 2 describes the related work regarding the basic concepts of both goal and scenario approach and function point. In Section 3, our approach to count function point from textual requirements in terms of goal and scenario is discussed using a case example. Then, Section 4 describes the validation of our approach. The last section concludes the paper and describes future work.

2. Prior work

2.1. Goal and scenario based requirements analysis

Goal and scenario based requirements analysis is used to elicit and analyze requirements before function point is counted. After requirements are specified in terms of goal and scenario, function point can be calculated based on goal and scenario. In this paper we adopt the goal and scenario approach discussed in Kim et al. (2004, 2006), which is briefly described below.

The goal and scenario approach has been used to elicit initial requirements and to refine requirements from a higher level to a lower level (Dardenne, Van Lamsweerde, & Fickas, 1993; Kim, Park, Sugumaran, & Yang, 2007; Kim et al., 2004, 2006; Rolland, Souveyet, & Achour, 1998). The goal and scenario approach provides multiple abstraction levels which help in the separation of concerns in requirements elicitation. We use four abstraction levels, namely, business, service, interaction and internal level. The aim of the business level is to identify the ultimate purpose of a system. At this level, the overall system goal is specified by the organization or a particular user. The aim of the service level is to identify the services that a system should provide to an organization and their rationale. At the system interaction level the focus is on the interaction between the system and its agents. The internal level focuses on what the system needs to perform the interactions selected at the system interaction level. A goal is generated at each level and scenarios are created to achieve that goal. Goals at lower level are derived from scenarios at higher level. This mechanism supports the elicitation of requirements through goal and scenario, and helps to refine the requirements.

We also use goal and scenario authoring rules. A Goal is authored as a template $\langle Verb + Target + Direction + Way \rangle$, where Verb is an active verb, Target is a conceptual or a physical object, Direction is either source or destination, and Way is the way in which the goal is to be achieved. In general each goal is expressed as a simple sentence with '*Verb*' and '*Target*' parameter as mandatory. Sometimes 'direction' and 'way' can be omitted. For example, the goal, 'Withdraw cash from the ATM' is represented as follows: '$(\text{Withdraw})_{\text{Verb}}$ $(\text{Cash})_{\text{Target}}$ $(\text{From the ATM})_{\text{Dir}}$'.

The scenarios capture real requirements since they describe real situations or concrete behaviors, and goals can be achieved through the execution of scenarios. Thus, scenarios have their goals, and typically, goals are achieved by scenarios. In other words, just as goals can help in scenario discovery, scenarios can also help in goal discovery. As each individual goal is discovered, a scenario can be authored for it. Once a scenario has been authored, it can be explored to yield further goals. All scenarios should be authored using the following format:

$$\text{'}Subject : Agent + Verb + Target : Object + Direction : (Source, Destination) + Way\text{'}.$$

The expected scenario prose is a description of a single course of action. This course of action should be an illustration of fulfillment of the goal. It should describe the course of actions that are expected, not the actions that are not expected, impossible, and not relevant with regard to the problem domain. Although this format may be perceived to be a bit simplistic, it is sufficient to proceed with modeling goals and scenarios. The indirect objects can be filled in the slot of *Direction*. For example, 'Tom gives me a book' can be rewritten, according to our rules as, '$(\text{Tom})_{Agent}$ $(\text{gives})_{Verb}$ $(\text{a book})_{Object}$ $(\text{to me})_{Direction}$'. The 'Direction' and 'way' are optional in a scenario.

2.2. Function point

Function point is a method for measuring software size which was developed in 1979 (Albrecht, 1979). The method is widely used and adopted as an international standard, namely, ISO14143-1 (ISO-IEC, 1998). It is also adopted by ministry of information and communication in Korea (MIC, 2004). The IFPUG (International Function Point User Group) was established in 1984 and has published FP CPM (Function Point Counting Practice Manual) (IFPUG, 2000). Functions of a software system consist of the following components according to IFPUG's Counting Practice Manual:

- **ILF (Internal Logical File):** data or control information maintained through one or more elementary process of the target application
- **EIF (External Interface File):** data or control information referenced through one or more elementary processes
- **EI (External Input):** an elementary process to maintain an ILF or alter the behavior of the application
- **EO (External Output):** a process that presents information to user through processing logic
- **EQ (External inQuiry):** an elementary process for presenting information to the user through retrieval of data or control information from an ILF or EIF

The elementary process mentioned above represents the smallest process that a user can identify. ILF and EIF are data functions. EI, EO and EQ are transaction functions. CPM (Counting Practice Manual) of IFPUG provides a method for identifying the functions and counting points based on these functions.

The IFPUG issues a license and certifies experts for counting function point. Generally the expert identifies ILF, EIF, EI, EO and EQ from requirements specification and counts function point according to IFPUG's CPM. However this approach needs expert's help each time the requirements are changed because there are no systematic methods linking function points with requirements specification. This approach also has problems with respect to recalculating function points or tracking cost when requirements change.

3. Approach for counting function point from requirements

This section describes our approach for counting function point from goal and scenario based textual requirements. We assume that the requirements have already been analyzed in terms of goal and scenario (for more details, please refer Kim et al. (2006)). Section 3.1 discusses the main idea of our approach and the next subsections describe some rules that help in counting function point from requirements. Additional details about the process are provided in the case study discussed in Section 4.

3.1. Proposed approach

Our approach focuses on extracting function point from goals and scenarios at the interaction level. The main reason for this is that the goal and scenario at the interaction level describes the interaction between user/external application and the target application (Kim et al., 2006). It includes data for interaction and behavior for data processing. Data for interaction can be used to derive data functions (ILF or EIF) maintained in the target application or external application. Behavior for data processing can drive transaction functions (EI, EO or EQ), which are behaviors dependent on input, output or inquiry. The overall concept of the proposed method is represented in Fig. 1. The initial requirements are described using goal and scenario approach. Function point is extracted through function point Extraction rules such as Context Analysis rules, Data Function Analysis rules, and Transaction Function Analysis rules. Finally, based on function point, cost or schedule is calculated by cost estimation model such as COCOMO II (Boehm et al., 2000).

3.2. Function point extraction rules

As stated earlier, one can think of function point extraction rules as a set of guidelines that help identify function point. They are based on linguistic techniques. First, we discuss the structure of function point extraction rules and then each rule in detail.

3.2.1. The structure of function point extraction rules

We propose 10 rules to help identify function point. These rules consist of three different types, namely, Context Analysis (CA) rules, Data Function Analysis (DFA) rules, and Transaction Function Analysis (TFA) rules. CA rules are for scoping the application boundary and extracting context elements. DFA rules are for separating data functions into ILF and EIF. TFA rules are for identifying transaction functions that represent the functionality provided to the user for the processing of data by the target system.

As mentioned above, all the rules are based on textual requirements in terms of goal and scenario. In fact, these requirements are often represented as goal & scenario tree and the description of goals and scenarios. Hence, function point extraction rules are preceded by the inspection of the contents of goals and scenarios. Table 1 shows the relationship between goal & scenario based textural requirements and function point Extraction rules.

For example, the rule CA2 can be executed through scenarios because agents are identified by the interaction between *subject* and *target* in the scenario template (more details provided in next section).

3.2.2. Function point extraction rules

We propose the following Function Point Extraction rules. We have developed a domain analysis approach to identify common extraction patterns and their constraints. The formalization of these patterns results in the current set of extraction rules. In this

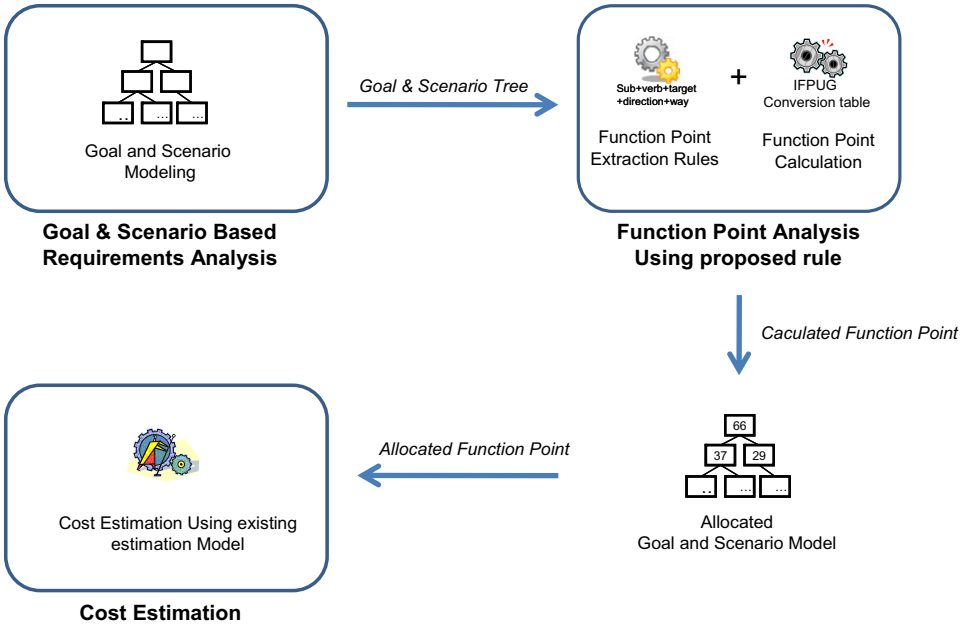

Fig. 1. Overview of our approach.

Table 1
The relationship between rules and goal and scenario.

Rule category	Id	Relationship	
		Goal	Scenario
Context analysis	1	N/A	N/A
	2	N/A	Inspected
	3	N/A	N/A
Data Function Analysis	1	Inspected	N/A
	2	N/A	Inspected
	3	N/A	inspected
Transaction Function Analysis	1	Inspected	N/A
	2	N/A	Inspected
	3	N/A	Inspected
	4	N/A	Inspected
	5	N/A	Inspected
	6	N/A	Inspected

section, each rule is introduced using the following template ⟨Definition, Comment, Example⟩. The definition explains the contents of the rule. The comment is expressed as items to be considered when applying the rule. The example component shows a representative example (we show an example from the Ordering Processing System).

3.2.2.1. Context Analysis Guiding rule 1 (CA1). Definition: If the target system can be decomposed into several applications, the applications should be decomposed according to business goals. Individual application is refined into service goal.

Comment: Application decomposition is determined by business functions.

Example: Fig. 2 shows that 'order processing system' is decomposed into two applications because business functions can be decomposed.

3.2.2.2. Context Analysis Guiding rule 2 (CA2). Definition: The Agent represented in '*subject*' interacts with another agent in '*direction*'. If the agent is human, then it must be a user. The other agent that is interacting with the human agent is the target application. Likewise, the agent interacting with the target application can also be an external application.

Example: In the example below, accounting manager is the user, and the order management application (OMA) is the target application. Accounting system is external application.

1. $(\text{Accounting manager})_{\text{sub}}$ $(\text{add})_{\text{verb}}$ $(\text{customer name})_{\text{target}}$ $(\text{to OMA})_{\text{direction}}$

3.2.2.3. Context Analysis Guiding rule 3(CA3). Definition: If target system is decomposed into several applications, the decomposed applications are designed separately. When one of applications is designed, the other can be regarded as external applications.

Example: In Fig. 2, we should consider 'order management application' and 'warehouse management application' independently. When we design 'order management application', 'warehouse management application' is regarded as external application at the point of 'order management application's view.

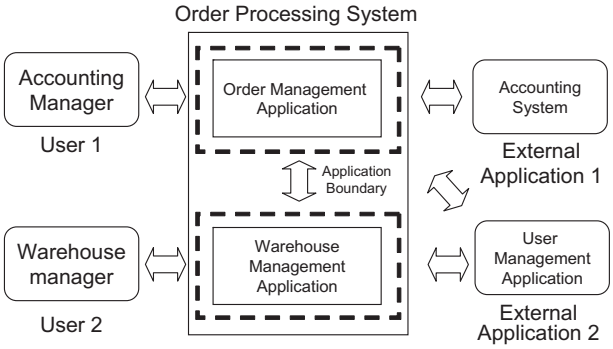

Fig. 3. Context diagram for order processing system.

Using the rules above, we can extract context elements from goal and scenario model. For the order processing system example, the above rules have been applied to the goal and scenario model and the following agents have been identified – accounting manager, warehouse manager, user management application and accounting system. Order processing system is decomposed into 'order management application' (OMA) and 'warehouse management application' (WMA). Fig. 3 shows the context diagram for order processing system.

After the context analysis, the data functions should be identified. Data functions are characterized by ILF and EIF. Concerning goals and scenarios at the interaction level, the '*target*' expresses data or control information of transmission between target application and the user or external application. This data can be maintained in the target application or the external application. If the data is maintained in the target application, it is ILF. If the data is maintained in the external application, then it is EIF.

After identifying the data functions, the complexity of the data functions should be determined. Complexity of the data function is determined from RET (Record Element Type) and DET (Date Element Type) (IFPUG, 2000). DET is a unique user recognizable and unrepeated field in the ILF or EIF. RET is a user recognizable subgroup of data elements within an ILF or EIF. The '*Target*' component of a scenario expresses the refined data of the goal. Thus, the refined data in several requirements chunks (G, Sc) can be grouped. We can count the number of refined data elements and the subgroups. Complexity of data function is measured by the number of refined data elements, represented as DETC (DET Count) and the number of subgroups, represented as RETC (RET Count) (IFPUG, 2000). Explanation and examples are given in the following rules.

3.2.2.4. Data Function Extraction Guiding rule 1 (DF1). Definition: The '*target*' of a goal and the group of the '*target*' in the scenario which is not related to the '*target*' of such a goal is ILF or EIF candidate.

Comment: The candidate should be a data element maintained in application and duplication regarded as one. The maintained application can be found in '*subject*' or '*direction*'.

Example: In the following sentences, the first sentence is about a goal. The second and third sentences are about partial scenarios

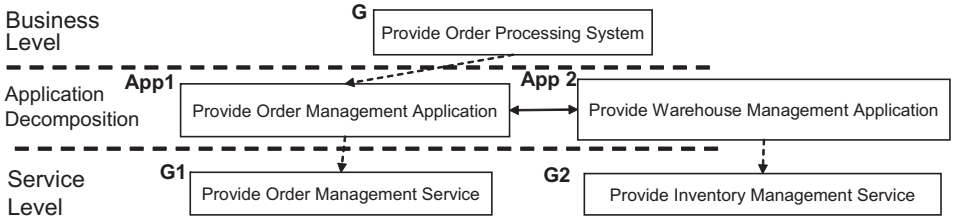

Fig. 2. Application decomposition.

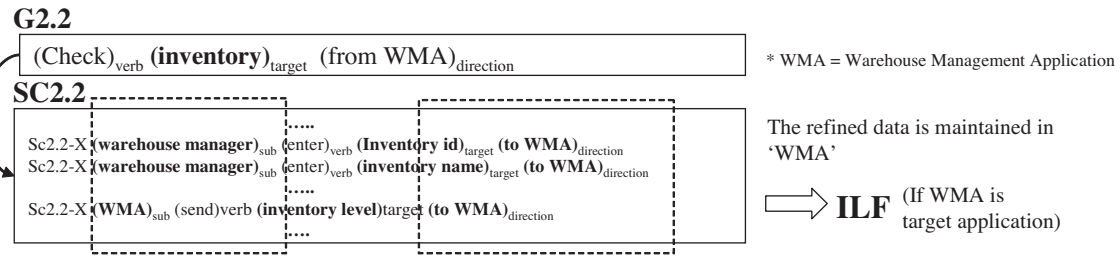

Fig. 4. Example of extracting data function from (G, Sc) 1.

refining the goal. 'Order' is maintained in Order management application. 'Invoice id' and 'invoice contents' are not refined data for 'order', and they can be grouped into 'invoice information' maintained in accounting system. So 'order' and 'invoice information' are ILF or EIF.

1. $(\text{Add})_{\text{verb}}$ **(Order)** target (to OMA)direction.
2. $(\text{OMA})_{\text{sub}}$ $(\text{receive})_{\text{verb}}$ **(invoice id)**$_{\text{target}}$ $(\text{from Accounting System})_{\text{direction}}$.
3. (OMA)sub $(\text{receive})_{\text{verb}}$ **(invoice contents)**$_{\text{target}}$ $(\text{from Accounting System})_{\text{direction}}$.

3.2.2.5. Data Function Extraction Guiding rule 2 (DF2). Definition: Scenario includes refined data of ILF or EIF. If the refined data are maintained in target application, the data function is ILF. If the refined data are maintained in external application, the data function is EIF.

Comment: All goals which include same data function should be considered. The maintained application can be found in '*subject*' or '*direction*'.

Example: In Fig. 4, the refined data for 'inventory' are maintained in WMA. If the WMA is target application, it is ILF and if the WMA is not the target application, it is EIF.

3.2.2.6. Data Function Extraction Guiding rule 3 (DF3). Definition: The identified data function contains refined data. The refined data can be grouped into subgroups. The number of refined data elements is denoted by DETC and number of subgroup is denoted by RETC.

Comment: When the refined data is grouped, all the goals which include the data should be considered. Final complexity is determined by RET/DET complexity matrix for data functions in IFPUG CPM (Counting Practice Manual) (IFPUG, 2000).

Example: In Fig. 5, 'order' is the data function and it is common data for G1.1, G1.2 and G2.1. It includes ten different data elements such as customer name, etc. They can be grouped into two parts, which are 'customer information' and 'order item information'. We can count the number of unique data elements as well as the subgroups. The former represents the DETC and the latter is RETC. Finally we can determine the complexity according to RET/DET complexity matrix.

After identifying the data functions, the transaction functions should be extracted. Transaction functions represent the functionality provided to the user for the processing of data by the target system. They are EI (External Input), EO (External Output), and EQ (External inQuiry). They exhibit the behavior for processing data. In the goal and scenario based approach, the interaction level contains 'verbs' which can represent the behavior for processing data.

In our approach, we identify transaction functions from 'verb' and classify transaction functions using 'verb type' and authoring format. We name the transaction function using the 'verb + target + (direction)' fragment in the goal. This paper restricts the 'verb' in a goal and scenario to active voice. The 'verb type' is defined as the 'data sending type verb' and the 'data receiving type verb'. The main intent of the 'data sending type verb' is sending data, and 'data receiving type verb' is receiving data. Examples of commonly used verbs in requirements statements are given in Table 2. The authoring format of the goal used for classification is described in the rules.

Complexity of transaction function is calculated from FTR (File Type Reference) and DET (Data Element Type) (IFPUG, 2000). FTRC (File Type Reference Count) is the number of ILF or EIF which are referred to by the transaction function. DETC for the transaction function is the number of data elements which are referred to by the transaction + (error, confirm, or complete message) + starting method, etc. We can determine the value of DETC from the 'target' in a given scenario. We can also count the FTR from checking the related ILF or EIF and determine the value of FTRC. Detailed explanation and some examples are given in the following rules.

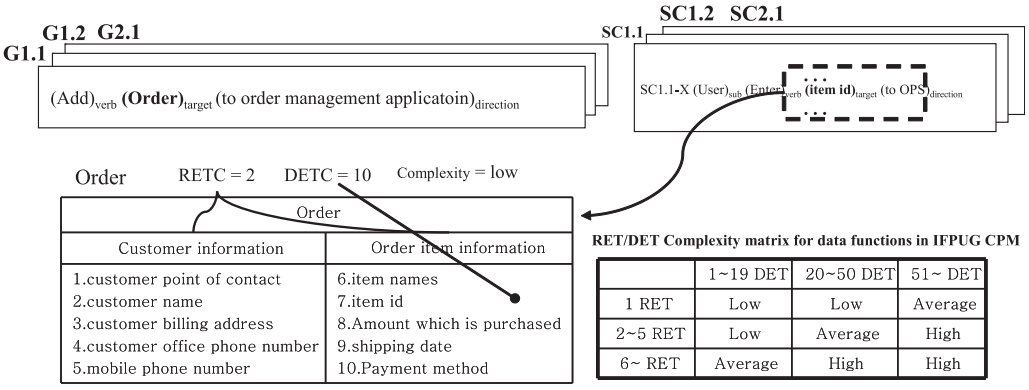

Fig. 5. Example of counting complexity of data function from (G, Sc).

Table 2
Commonly used verbs in requirements state.

Data sending type verb	Data receiving type verb
Send, display, add, dispatch, forward, direct, convey, remit , input, request, modify, fill, output, enter, list	Receive, get, accept, be given, pick up, collect, obtain, acquire, take, validate, check

3.2.2.7. Transaction Function Extraction Guiding rule 1 (TF1).

Definition: '*verb* + *target* + (*direction*)' components of a goal lead to EI, EO, and EQ candidates. '*Direction*' can be omitted in the name of a transaction. If several goals are connected with an 'or' relation and only the '*way*' component is different, then one transaction is identified among the several goals.

Example: In the two goals given below, 'add order' and 'add order through backorder' are transaction candidates. But only the '*way*' component is different. So, these two goals lead to one transaction.

$(\text{Add})_{\text{verb}}$ $(\text{order})_{\text{target}}$ $(\text{to OMA})_{\text{direction}}$
$(\text{Add})_{\text{verb}}$ $(\text{order})_{\text{target}}$ $(\text{to OMA})_{\text{direction}}$ $(\text{through backorder})_{\text{way}}$

3.2.2.8. Transaction Function Extraction Guiding rule 2 (TF2).

Definition: Goal does not have a subject but the subject is regarded as a subject in the scenario that derives the goal. In a goal from which a transaction is derived through TF1, if the authoring format is $\langle(\textit{User or External application})_{sub} + (\textit{Data}$ sending type verb$)_{verb} + (\textit{Target data})_{Target} + (\textit{target application})_{direction}\rangle$ or $\langle(\textit{target application})_{sub} + (\textit{Data}$ receiving type verb$)_{verb} + (\textit{Target data})_{Target} + (\textit{User}$ or External application$)_{direction}\rangle$, then the transaction target is EI.

Comment: If there are same format goals except '*way*', the goals are regarded as one transaction function.

Example: In the requirement below, the '*subject*' is user and the '*direction*' is target application. The 'Verb type' is 'data sending type verb' and the main intent is sending order data to the target application. So, 'Add order' is EI for the application.

$(\text{Accounting manager})_{\text{sub}}$ **(add)**$_{\text{verb}}$ **(order)**$_{\text{target}}$ $(\text{to OMA})_{\text{direction}}$

3.2.2.9. Transaction Function Extraction Guiding rule 3 (TF3).

Definition: In a goal which leads to a transaction through TF1, if the authoring format is $\langle(\text{User or External application})_{\text{sub}} + (\text{Data receiving type verb})_{\text{verb}} + (\text{Target data})\text{Target} + (\text{target application})_{\text{direction}}\rangle$ or $\langle(\text{target application})_{\text{sub}} + (\text{Data sending type verb})_{\text{verb}} + (\text{Target data})_{\text{Target}} + (\text{User or External application})_{\text{direction}}\rangle$, then the transaction which the goal relates to has EO or EQ, which is the behavior for output or inquiry.

Comment: If there are same format goals except '*way*,' the goals are regarded as one transaction function.

Example: In the requirement given below, the '*subject*' is user and the '*direction*' is target application. 'Verb type' is 'data receiving type verb' whose main intent is receiving inventory data and checking. So, 'check inventory' is EO or EQ.

$(\text{Warehouse})_{\text{sub}}$ **(check)**$_{\text{verb}}$ **(inventory)**$_{\text{target}}$ $(\text{from WMA})_{\text{direction}}$

3.2.2.10. Transaction Function Extraction Guiding rule 4 (TF4).

Definition: EO or EQ from TF3 can be classified using this rule. If it includes mathematical calculation, measurement, derived data, changes system behavior or maintaining data function, it is EO. Otherwise, it is EQ.

Example: In the following requirement, 'check inventory' is EO or EQ according to TF3. However, 'check inventory information' does not include mathematical calculation, measurement, derived data, change system behavior, or maintain data function. So 'check inventory' is EQ.

$(\text{Warehouse manager})_{\text{sub}}$ **(check)**$_{\text{verb}}$ **(inventory)**$_{\text{target}}$ $(\text{from WMA})_{\text{direction}}$

3.2.2.11. Transaction Function Extraction Guiding rule 5 (TF5).

Definition: DETC for transaction functions is the sum of the following:

(The number of unrepeated data elements in '*target*' of the scenario to achieve the transaction) + (number of error, confirm, or complete message) + 1 (if there are starting methods for transactions such as button or hot key)

Comment: Unrepeated data in '*target*' of a scenario should be transmitted between target applications and outside and also be in ILF or EIF. If the transaction has 'or' relation goal with different '*way*' (like example of TF1), the '*target*' of the scenario of the 'or' relation goals should also be considered.

Example: In Fig. 6, 'add order' is EI. In the scenario, there are 14 related data items (10 for 'order', 2 for 'invoice' and 2 for 'inventory') and confirming message. We assume there is a starting method for 'ok' button. So it's DETC is 16 (10 + 2 + 2 + 1 + 1).

3.2.2.12. Transaction Function Extraction Guiding rule 6 (TF6).

Definition: FTRC of a transaction is the number of ILF or EIF which is referred by the transaction. If the data elements used

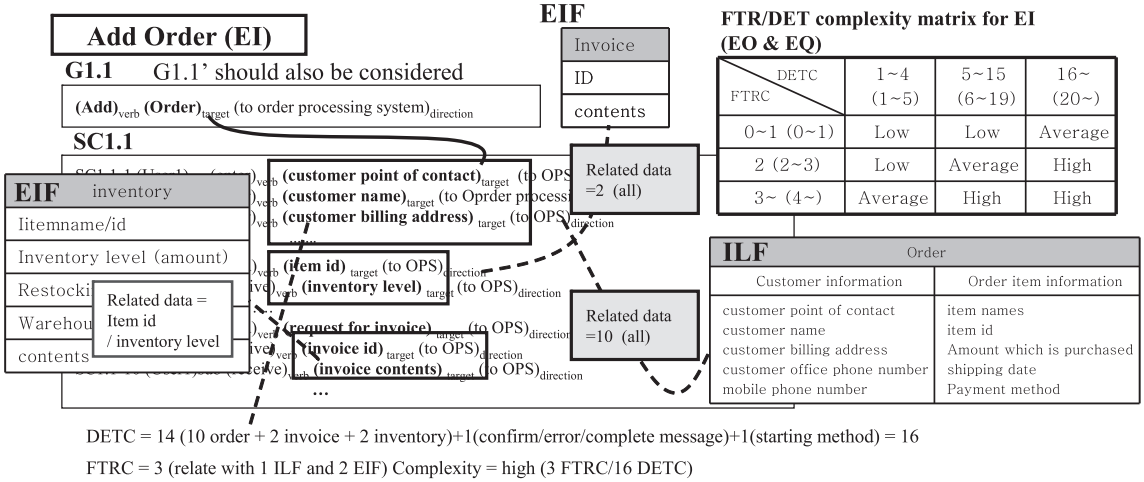

Fig. 6. Determining complexity of transaction function.

for determining DETC is one of the elements of a certain ILF or EIF, then the ILF or EIF is regarded as being referred by the transaction.

Comment: Final complexity is determined by the FTR/DET complexity matrix for EI, EO and EQ, which is provided by the IFPUG CPM (IFPUG, 2000).

Example: In Fig. 6, 'add order' is EI. The data elements used for determining DETC is included in one ILF named 'Order' and two EIFs named 'invoice' and 'inventory'. Therefore, there is one ILF and two EIFs which are referred by 'add order'. So, the FTRC is 3. The final complexity of 'add order' is determined as 'high' by using the FTR/DET matrix.

4. Case study: course registration system

To illustrate the feasibility of our approach, a case study has been conducted using a course registration system. The main purpose of this system is to manage data. Because the function point technique defines functions according to data exchange type and data location, data centered system is more suitable for a case study than others such as embedded systems.

'Course registration system' is a system for registering for courses using the Internet. Major services are registering courses by students, managing courses for professors and managing the system by staff. This case study consists of goal and scenario modeling, extracting context elements, extracting data function, extracting transaction function, calculating unadjusted function point and estimating cost using previous cost model. The following subsection describes each activity in detail.

4.1. Goal and scenario modeling

Fig. 7 represents a partial view of requirements in terms of goal and scenario. The business goal, which is given from organizational strategy and marketing plan, is 'build course registration system'. Service goals to satisfy the business goal are 'provide course registration service for student', 'provide course management service for professor' and 'provide system management service for staff'. Interaction goals corresponding to service goals are 'view list of course', 'add course to register', 'modify course to register', 'check authority', 'select course to teach', 'submit grade', 'view grade info', 'add user information', 'modify user information', 'add course information', 'modify course information', 'close course', 'check authority in normal way', 'check authority with invalid id or pw' and 'send billing information'. These interaction goals should be refined into internal goals which satisfy their own parent goal. As mentioned above, more details about the goal and scenario based requirements analysis is discussed by Kim et al. (2004), Kim et al. (2006).

4.2. Extracting context elements

Our approach uses goal and scenario at the interaction level because it describes the interaction between user or external application and the target application (Kim et al., 2006). It includes data for interaction and behavior for data processing. Data for interaction can be used to derive data functions (ILF or EIF) maintained in the target application or external application. Behavior for data processing can drive transaction functions (EI, EO or EQ), which are behaviors dependent on input, output or inquiry. Thus, Context elements are extracted from 'subject' and 'direction' at the interaction level according to the authoring rule. 'Course registration system consists of one application. It means that application decomposition is not performed, and CA1 is not applied. External agents such as student, professor, staff, and billing system are extracted using CA2. Fig. 8 represents the context analysis of course registration system.

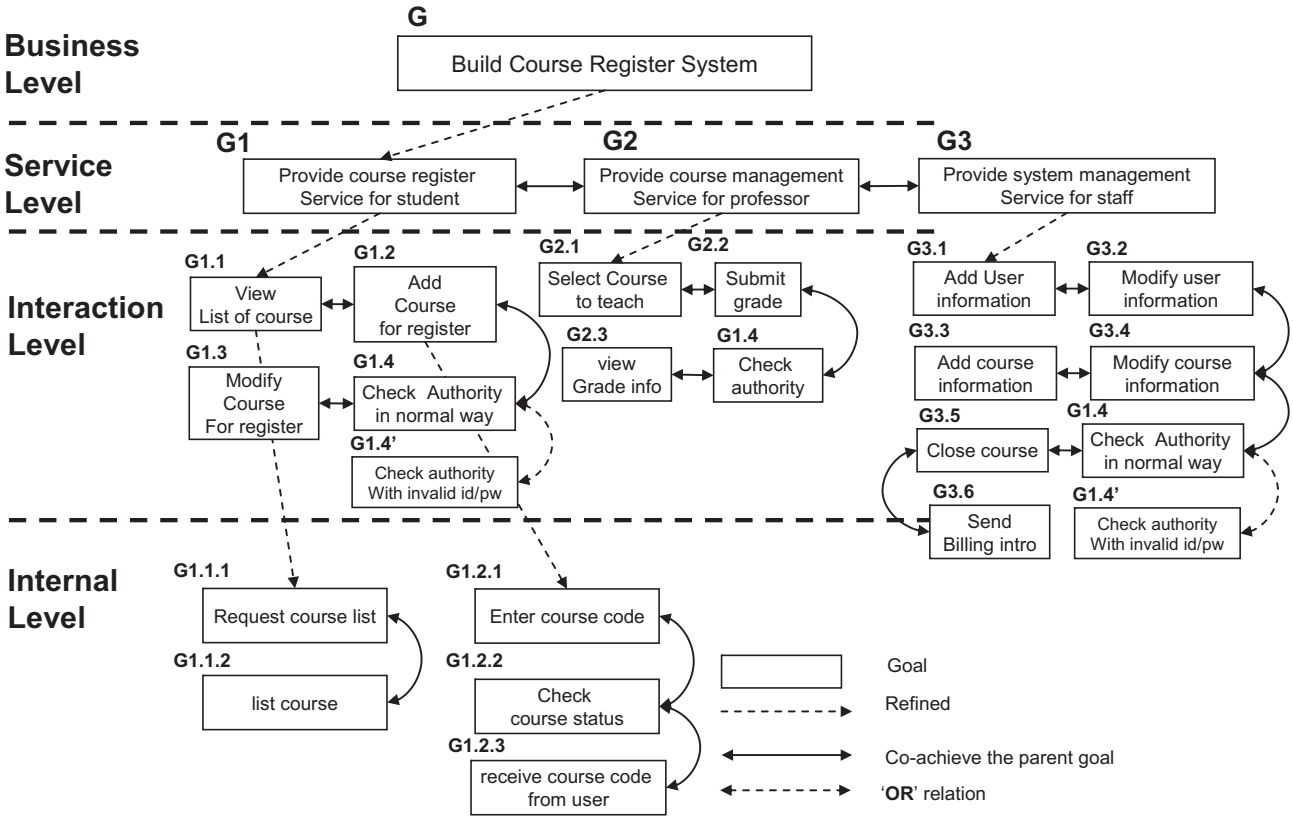

Fig. 7. Partial goal and scenario model of course registration system.

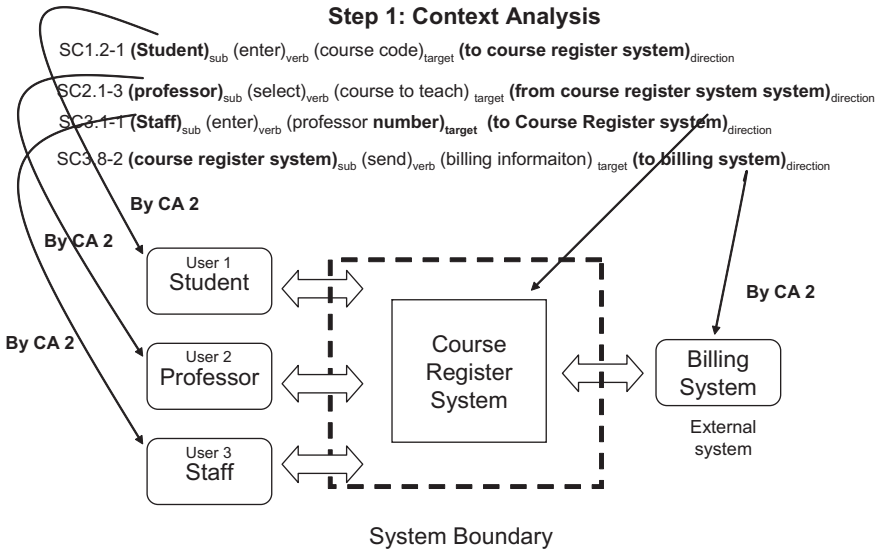

Fig. 8. Context analysis of course registration system.

Table 3
Data functions and complexity of course registration system.

Data F.	Function name	RET	DET	CPX
ILF	Course	1	7	Low
	Grade	1	4	Low
	User	3	14	Low
EIF	Billing	1	5	Low

4.3. Determining data functions and complexity

After defining the scope of the system, ILF and EIF as data functions should be identified. Course, Grade information, user and Billing information can be identified from the interaction level goal and scenario according to DF1. These data functions are identified from '*target*' of the scenario and the name is determined by grouping. These data functions are classified into ILF and EIF using DF3. Billing information is EIF because it is maintained outside the system boundary. The other data function is ILF because it is maintained within the system boundary.

Complexity of data function is determined by DF3, by using DETC and RETC. DETC is the count of data elements in the data function and RETC is the count of data group that is meaningful to user. If we can count DETC and RETC, we can determine its complexity by the RET/DET matrix which is provided by IFPUG's CPM. The result of data complexity is represented in Table 3.

4.4. Determining transaction functions and complexity

Transaction functions such as EI, EO and EQ are behavior related to input, output, inquiry data or control information which is distinguished by the user. This can be extracted from the '*verb*' of the goal at the interaction level. In case of the course registration system, we can identify 14 transaction functions according to TF1. The identified functions are: add course to register, modify course to register, select course to teach, submit grade, add user, modify user, add course, modify course, close course, view list of course, check authority, view grade information and send billing information, which are candidates for transaction function. We can classify these functions into EI, EO or EQ according to TF2, TF3 and TF4. In our case, view list of courses, check authority, view grade information, send billing information are EQ, and the others are EI. Complexity of transaction function is determined by the FTR/DET matrix of IFPUG's CPM. DETC and FTRC are determined by TF6. The result is represented in Table 4.

Table 4
Transaction functions and complexity of course registration system.

Transaction F.	Function name	FTRC	DETC	CPX
EI	Add course for register	2	5	Average
	Modify course for register	2	5	Average
	Select course to teach	2	3	Low
	Submit grade	1	5	Low
	Add user	1	15	Low
	Modify user	1	15	Low
	create course for teaching	1	8	Low
	Modify course for teaching	1	9	Low
	Close course	1	4	Low
EQ	View list of course	1	9	Low
	Check authority	1	3	Low
	View grade information	1	6	Low
	Send billing information	1	6	Low

4.5. Calculating unadjusted function point

After measuring EI, EO, EQ, ILF and EIF, we should calculate the UFP (Unadjusted Function Point). IFPUG's CPM provides conversion table for this, which is represented in Table 5 (IFPUG, 2000). For example, 'add course to register' is EI and complexity is low so its UFP is 3 according to the conversion table. If we multiply UFP by the adjustment factor, we can get AFP (Adjusted Function Point). However, most cost estimation models such as COCOMO, SEER, and PRICE accept only UFP as an input parameter (ISPA, 2007). Only UFP is accepted as international standard (ISO-IEC, 1998). Therefore, we deal with only UFP. The result of measuring UFP for the 'course registration system' is represented in Table 6. It has three ILF, one EIF, nine EI and four EQ. The final value of UFP is 67.

4.6. Estimating cost using cost model

This paper deals with extracting function point from goal and scenario based requirements. The extracted function point can

Table 5
UFP conversion table.

Func.	CPX		
	Low	Average	High
EI & EQ	3	4	6
EO	4	5	6
ILF	7	10	15
EIF	5	7	10

Table 6
Complexity/UFP of course registration system.

Data F.	Function name	CPX	UFP
ILF	Course	Low	7
	Grade	Low	7
	User	Low	7
EIF	Billing	Low	5
Transaction F.	Function name	CPX.	UFP
EI	Add course for register	Average	4
	Modify course for register	Average	4
	Select course to teach	Low	3
	Submit grade	Low	3
	Add user	Low	3
	Modify user	Low	3
	Add course	Low	3
	Modify course	Low	3
	Close course	Low	3
EQ	View list of course	Low	3
	Check authority	Low	3
	View grade information	Low	3
	Send billing information	Low	3
Total UFP			67

then be input into cost estimation models such as SEER, COCOMO, PRICE, etc. (ISPA, 2007). If both the proposed method and cost estimation model are used together, cost tracking and management can be performed.

To validate this, we input our UFP result into the SEER model, which is a popular cost estimation model. To estimate cost using SEER, we input SEER specific parameters, which are knowledge base parameters. The SEER model has four KBase (Knowledge Base) attributes which are Platform, Application, Acquisition Method, and Development Method (Galorath., 2010). In this case study, the values for these parameters are: Platform = 'Business and Non-critical MIS', Application = 'Transaction Processing', Acquisition Method = 'new development' and Development Method = 'OOD and OOP.' A detailed explanation of SEER is beyond the scope of this paper. Using SEER, we can estimate the effort to be 6.32 Effort Month. If we assume average labor rate to be $14,700, the estimated development cost is $92,966. Figs. 9 and 10 represent the input and the result of cost estimation from SEER.

5. Evaluation

We have proposed a method for extracting function point from goal and scenario. The proposed method is applied to course registration system and three other projects. We chose these case studies because function point technique is generally applied to a data centered domain which is concerned mainly with exchanging data.

To validate efficiency and effectiveness of our method, we analyzed the case studies and compared our results to the result provided by a specialist. We also found several advantages in supporting traceability activities. Section 5.1 describes the accuracy of extracting function point and Section 5.2 describes the advantages such as supporting traceability activities.

5.1. Accuracy of extracting function point

The proposed method systematically supports extracting function point from goal and scenario based requirements. To validate the usefulness of the method we request a function point specialist who is not an author of this paper to extract function point from our two case studies and two other real world projects. One of the real projects is the manhole management system, which is part of the waterworks system of Seoul, South Korea. The other is the SDRA project, which is a partial system of inventory documents management system of IBM Korea. We provided the specialist with the same specification and compared the results. The results are shown in Table 7.

The 'order processing system' and 'manhole management system' have the same results for our method and the specialist. However, the 'Course registrations system' and the 'SDRA' project' have slightly different results for our method vis-à-vis the FP specialist. The difference is about 5%. The reason for the difference is as follows.

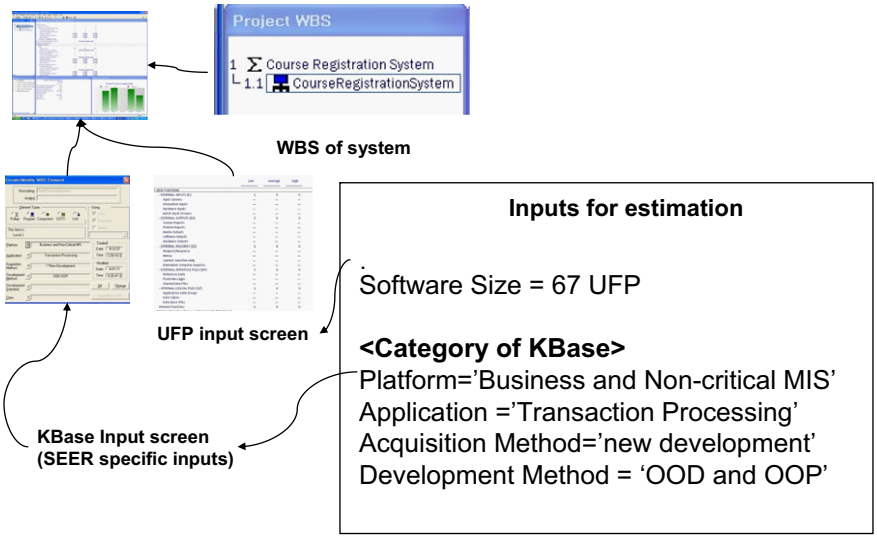

Fig. 9. Input for estimation model.

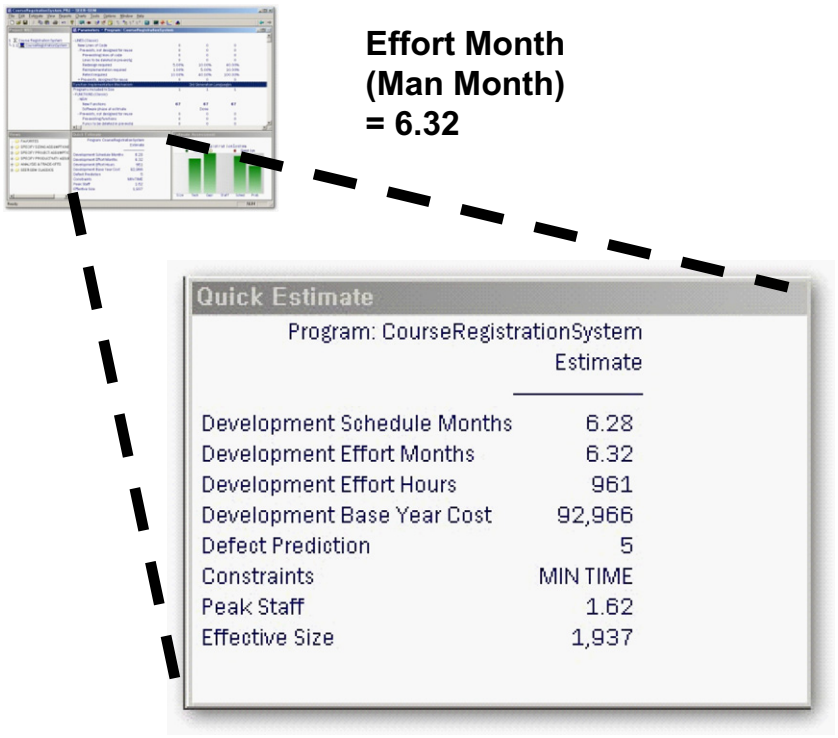

Fig. 10. Cost estimation result.

First, the 'counting error of starting method' causes the difference. The Specialist thought that some transaction functions had starting methods although the given specification does not describe it. They determined it by looking at the user interface design. They counted it as an additional DET. Because the difference of DETC is small, UFPs of only partial transactions are affected. So, some projects such as 'order processing system' and 'manhole management system' are not affected by this.

Second, the 'classification error of transaction' also causes the difference. The specialist thought that the 'view list of course' and 'view grade information' are EO though they are EQ in our case study. The specialist thought that the transactions had mathematical calculation. So, they classified the transaction into EO. EO has more UFP than EQ in case of same complexity. So, UFP of the specialist is more than the UFP of the proposed method in the 'course registration system'.

Third, the 'identification error of transaction' also causes the difference. The specialist thought that one of the EQ transactions of SDRA project is not a transaction although the transaction is identified by our rule. So the specialist did not count it and regarded it as a simple function. So, UFP of the specialist is less than UFP of the proposed method in the 'SDRA project'.

As described above, the accuracy is more than 94% and the reason for the error is caused by human mistake or lack of specification. The 'Starting method' can be described in the specification. Classification error is caused by human mistake. Identification error can also be described in the specification.

The proposed method helps even a beginner to extract function point quite accurately and also has several advantages such as supporting traceability activities. The following section further describes these advantages.

5.2. Advantages supporting traceability activities

The proposed method provides traceability link between requirements and cost. It is possible because the proposed method allocates function point to interaction goal and links the interaction goal with data functions. This traceability link can support traceability activities which are impact analysis, derivation analysis and coverage analysis. Impact analysis is the activity that analyzes cost difference due to requirements change. Derivation analysis is the activity that finds or traces changed requirements in response to cost. Coverage analysis is the activity that analyzes cost rate of selected requirements. This section describes an example of traceability activities using the proposed method and highlights its advantages.

Table 7
Comparison between proposed method and FP specialist.

Case study	Function	Proposed method	FP specialist
Course registration system	Data F.	26	26
	Transaction F.	41	45
	Total UFP	67	71
Order processing system	Data F.	39	39
	Transaction F.	29	29
	Total UFP	68	68
Manhole management system	Data F.	17	17
	Transaction F.	36	36
	Total UFP	53	53
SDRA project	Data F.	39	39
	Transaction F.	43	39
	Total UFP	82	78

Figs. 11 and 12 represent the allocated function point value in the goal and scenario model for the course registration system and order processing system. The following three examples explain how to perform traceability activities using the proposed method.

5.2.1. Example 1: impact analysis in course registration system

Let's assume that the following change request is issued: "We do not need authority check, remove the authority function".

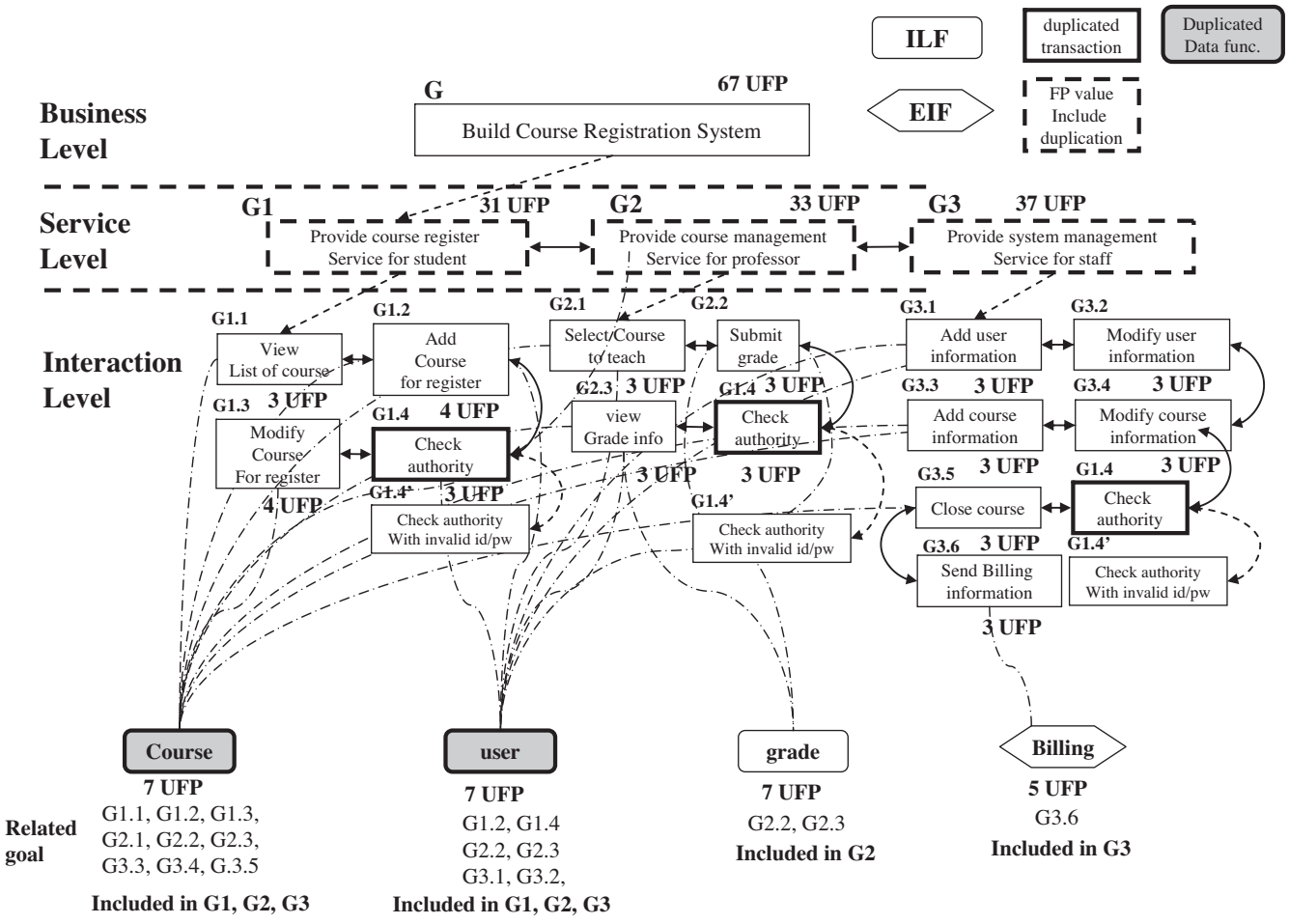

Fig. 11. Allocated function point value for the course registration system.

To do impact analysis we can check the allocated function point for 'authority check' and related data function. Calculating the difference can be performed easily. In Figs. 11 and 3 UFP is allocated for 'check authority' and 7 UFP is allocated for 'user' that is related to ILF. Function point is calculated based on the interaction level goal and related data functions. Therefore, 10 UFP is subtracted from each service goal because 'check authority' is included in G1, G2 and G3. However only 10 UFP is subtracted from the total UFP because duplicated goal is regarded as the same in the business level.

5.2.2. Example 2: derivation analysis in course registration system

Let's assume that the following derivation analysis request is issued: "We need to expand course data. What are the related goals and their function point?"

To do derivation analysis we can find the data function called 'course' and the related goals which are G1.1, G1.2, G1.3, G2.1, G2.2, G2.3, G3.3, G3.4, G3.5, G1, G2 and G3 in Fig. 11. We can easily calculate the summation of their function point, that is, 3(G1.1) + 4(G1.2) + 4(G.13) + 3(G2.1) + 3(G2.2) + 3(G2.3) + 3(G3.3) + 3(G3.4) + 3(G3.5) = 29 UFP.

5.2.3. Example 3: coverage analysis in order processing system

Let's assume that the following coverage analysis request is issued: "If we complete G1.1 and related data functions, what is the cost coverage in the order processing system?"

In Fig. 12, G1.1 is related with 3 data functions which are order, invoice and inventory. Summation of the allocated function point of G1.1 and the related data functions is 23 UFP. The order processing system's total UFP is 68. Therefore cost coverage of G1.1 and related data functions is 23/68 = 33.82% of total FP.

The above examples show that traceability activities can be performed by finding the linked node and performing simple calculations. Even a novice can perform impact analysis, derivation analysis and coverage analysis. Without the proposed method, these activities would be separated. Function point expert and requirements expert should work together. We are also planning to develop a supporting case tool to facilitate this process. With the supporting tool, we expect that the traceability management activities can be performed more efficiently. Table 8 succinctly summarizes the advantages of our proposed method.

6. Conclusion and future work

Requirements and development cost are closely related. If requirements are changed, cost also changes. If the cost condition is changed, it can affect the requirements. This paper has discussed a function point extraction method from goal and scenario based requirements and it has been applied to the 'course registration system' and 'order processing system'. The proposed method can provide the intermediation between requirements and cost. It provides rules for extracting function point from goal and scenario based requirements and thus, links requirements with function point. Function point is an input parameter for existing cost models. The goal and scenario model provides different abstraction levels for tracking and linking requirements with cost by allocating function point to goal model.

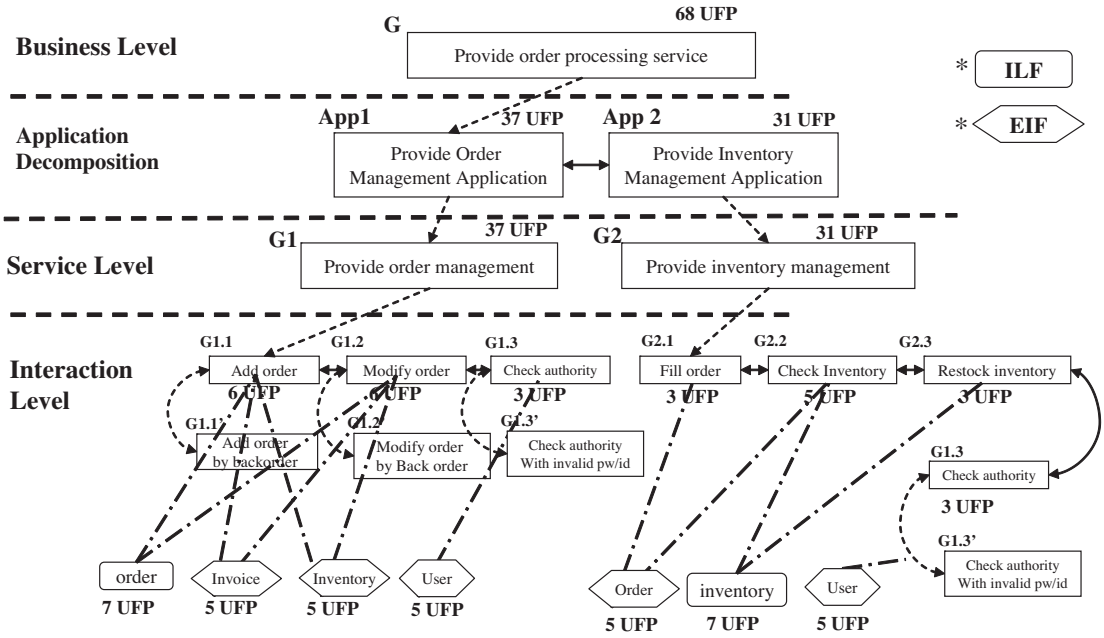

Fig. 12. Allocated function point value for order processing system.

Table 8
Advantages of the proposed method.

Advantages	Description
Easy to do traceability activities	It can be performed by finding linked node and simple calculating
Integrated traceability management	Requirements traceability activities and cost analysis can be performed together
Tool support	Tool support is possible and planned

Table 9
Comparison between proposed method and previous method.

Item	FP from UML	Proposed method
Easy estimation	Yes	Yes
Early estimation	No	Yes
Accuracy	High	High
Traceability support	No	Yes
Tool support	Yes	No (but planned)

Because existing approaches do not provide the linkage between requirements and cost, traceability management activities need expert's help. However, the proposed method helps traceability management activities to be performed systematically. Even a beginner can estimate and manage cost using our approach. Some previous approaches which are about function point extraction from UML specification also provide cost estimation (Cantone, Pace, & Calavaro, 2004; Harput, Kaindl, & Kramer, 2005; Uemura, Kusumoto, & Inoue, 1999). Nevertheless, they do not support traceability activities and early estimation is difficult as shown in Table 9.

A limitation of the proposed method is that it is not appropriate for embedded systems because function point is not suitable for embedded systems. Our future work involves developing a supporting tool for estimating function point automatically and refining the rules for extracting function points. This automated tool will use ontologies for implementing the rules, since the tool should be able to judge the meaning of words. We also plan to integrate this supporting tool with the goal and scenario based architecture modeling tool.

Acknowledgments

This work has been partly supported by Sogang Business School's World Class University Program (R31-20002) funded by Korea Research Foundation, and MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by NIPA (National IT Industry Promotion Agency (NIPA-2011-(C1090-1131-0008)).

References

Albrecht, A. (1979). *Measuring application development productivity*. *Proceedings of the joint SHARE/GUIDE IBM applications development symposium*. CA: Monterey, pp. 83–92.

Auer, M., Becker, C., Rauber, A., & Biffl, S. (2005). Implicit analogy-based cost estimation using textual use case similarities. In *Proceedings of 2nd international conference of intelligent computing and information systems (ICICIS 2005)* (pp. 369–376). ACM Press.

Boehm, B. (1981). *Software engineering economics*. Prentice Hall.

Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B. H., Horowitz, E., et al. (2000). *Software cost estimation with Cocomo II*. Upper Saddle River, New Jersey: Prentice Hall.

Boehm, B., & In, H. (1996). Software cost option strategy tool (S-COST). In *Proceedings of 20th international computer software and applications conference (COMPSAC '96)* (pp. 15–20).

Cantone, G., Pace D., & Calavaro G. (2004). Applying function point to unified modeling language: conversion model and pilot study. In *Proceedings of the 10th international symposium on software metrics*.

Dardenne, A., Van Lamsweerde, A., & Fickas, S. (1993). Goal-directed requirements acquisition. *Science of Computer Programming, 20*(1-2), 3–50.

Davis, A. (1993). *Software requirements: Objects, functions and states*. Prentice-Hall.

Davis, A. (2003). The art of requirements triage. *IEEE Computer, 36*(3), 42–49.

Davis, A., & Zweig, A. (2000). Editor's corner: The missing piece of software development. *Journal of Systems and Software*, 205–206.

Galorath. (2010). *SEER for software*. URL: <http://www.galorath.com/index.php/products/software/C5/> (last accessed 20.07.2011).

Harput, V., Kaindl, H., & Kramer, S. (2005). Extending function point analysis to object-oriented requirements specifications. In *Software metrics 2005 11th IEEE international symposium* (pp. 10–39).

IFPUG. (2000). *Function point counting practice manual release 4.1.1*. International Function Point User Group.

ISO-IEC. (1998). *International ISO/IEC standard 14143-1, information technology – software measurement – functional size (part 1: definition of concepts)*.

ISPA, International Society of Parametric Analysts. (2007). *Parametric estimating handbook* (4th ed.).
Karlsson, J., & Ryan, K. (1997). A cost-value approach for prioritizing requirements. *IEEE Software, 14*(5), 67–74.
Kim, J., Park, S., & Sugumaran, V. (2006). Improving use case driven analysis using goal and scenario authoring: A linguistics-based approach. *Data and Knowledge Engineering, 58*(1), 21–46.
Kim, J., Park, S., & Sugumaran, V. (2004). *A linguistics-based approach for use case driven analysis using goal and scenario authoring. NLDB 2004, LNCS* (Vol. 3136, pp. 159–170). Springer-Verlag.
Kim, M., Park, S., Sugumaran, V., & Yang, H. (2007). Managing requirements conflicts in software product lines: A goal and scenario based approach. *Data and Knowledge Engineering, 61*(3), 417–432.
Larvet, P., & Vallée, F. (2002). *UML developments: Cost estimation from requirements, ECSQ 2002, LNCS* (Vol. 2349, pp. 156–164).
Lavazza, L., & Valetto, G. (1981). Requirements-based estimation of change costs. *Empirical Software Engineering* 5(3), 229–243.
Leffingwell, D., & Widrig, D. (2000). *Managing software requirements*. Addison-Wesley.
Li, Y., Xie, M., & Goh, T. (2009). A study of mutual information based feature selection for case based reasoning in software cost estimation. *Expert Systems with Applications, 36*(3), 5921–5931.
MIC, Ministry of information and communication republic of Korea. (2004). *Criterion for remuneration of software business, MIC-set No. 2004–2008*.
Park, H., & Baek, S. (2008). An empirical validation of a neural network model for software effort estimation. *Expert Systems with Applications, 35*(3), 929–937.
Pendharkar, P. (2010). Probabilistic estimation of software size and effort. *Expert Systems with Applications, 37*(6), 4435–4440.
Rolland, C., Souveyet, C., & Achour, C. (1998). Guiding goal modeling using scenarios. *IEEE Transactions on Software Engineering, 24*(12), 1055–1071.
Şen, C., & Baraçlı, H. (2010). Fuzzy quality function deployment based methodology for acquiring enterprise software selection requirements. *Expert Systems with Applications, 37*(4), 3415–3426.
Uemura, T., Kusumoto, S., & Inoue, K. (1999). *Function point measurement tool for UML design specification. Software metrics symposium 1999 proceedings sixth international*. IEEE Computer Society, pp. 62–69.
Wiegers, K. (1999). *Software requirements*. Redmond, Washington: Microsoft Press.
Yourdon, E. (1997). *Death march*. Upper Saddle River, New Jersey: Prentice Hall.