IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITB12840

This paper appears in the book, *Emerging Trends and Challenges in Information Technology Management, Volume 1 and Volume 2* edited by Mehdi Khosrow-Pour © 2006, Idea Group Inc.

A Method of Translating Business Use Cases into System Use Cases

George Abraham & Il-Yeol Song
College of Information Science & Technology, Drexel University, Philadelphia, PA 19104, T: (215)895-5912, {sga72, song}@drexel.edu

ABSTRACT

Business use cases (BUC) capture functional business requirements at an organization level, while system use cases (SUC) capture functional system requirements at a system level. Thus, a technique of translating business use cases into system use cases provides poor traceability between the business requirements and corresponding system functionality. In this paper we present a 7-Step method for translating a business use case into possible system use cases. A BUC template approach is shown for documenting the process. In doing so, we show a way to achieve much needed traceability between the business requirements and system functionality.

1. INTRODUCTION

IS architecture has to be business-driven, and changes in business processes have to be reflected in the Information system. One of the problems faced is poor traceability between the business requirements and corresponding system functionality. Traceability is essential for an organization to swiftly make changes to the IS functionality based on business demands. The use-case driven approach [Jac92] has become the de-facto standard in determining the functional requirements of the system. In this paper we employ the business use case (BUC) approach as defined in the business modeling discipline of the Rational Unified Process (RUP) [Kru03, Jac94, and Ng02]. The goal of this paper is to present a method for systematically translating a BUC in to possible system use cases (SUC). In doing this we would present a way to ensure traceability between them by using a BUC template for documenting the process. The advantage of using the BUC approach is that each business process of value is linked to BUC. Any change in the business process would trigger a change in associated BUC with a semantically similar name. Since the SUC are derived directly from a BUC, the analyst can quickly identify the system functionality that might be affected.

In the business use case (BUC) approach, a BUC represents a sequence of actions a business performs to yield an "observable result of value" to the *Business actor* (external to the business). A business-use-case realization describes the interaction between *Business entities*, and *Business workers* who are internal to a business process. A *Business entity* [Kru04] is a passive object and does not initiate interactions on its own. Business entities are usually information/data objects that are manipulated during the business process by actors/workers. A *Business worker* [Kru04] is someone who is internal to the business process and is shown only during business use case realization. A system-use-case on the other hand is defined as a sequence of actions a system performs and yields an observable result of value to a particular actor [Kru03; Jac92]. The distinction between the two kinds of use cases is that, a business goal sets the boundary for a business-use-case. BUC exist at an organizational unit level and may include manual processes. A system-use-case boundary depends on a particular system being developed actor includes only automated operations [Jac92].

The paper is organized as follows: Section 2 reveals relevant research on goal driven modeling effort. In Section 3 we present our method for translating business-use-cases to system-use-cases, and explain with an example, Section 4 concludes our paper and presents our future work.

2. RELATED RESEARCH

Business modeling depends on business goals, and systems modeling depend on user/actor goals to derive functional requirements [Fow97, Lam95, Lar05, RUP, and Jac92]. A review of goal oriented requirements engineering literature [Lam01, Kav04] lists some of the popular techniques employed for determining goals. Among the approaches, abstraction [Lou97] and refinement of goals have been used to derive goal trees. The goal tree is then mapped to the processes that help achieve the goals at the various levels. The *i** framework [Yu93; 97] shows a way to model dependencies (task and resource dependencies) among goals which are then paired with agents who accomplish it. This framework has been extended in research to derive system use cases [San02]. Another approach involves a modification of Role Activity Diagrams (RAD) called Systematic Technique for Role and Interaction Modeling (STRIM) [Oul93]. Here, role is a set of responsibilities carried out by a human, a system or a group of people, and for each role there are a set of activities which need to be carried out in a certain order.

The research on using BUC for deriving SUCs is rare. The benefit of the use case approach when compared to other techniques is the level of detail which is captured through simple language and documented using a well-defined system use case template. Since the entire system design depends on business requirements, it's essential that the requirements captured at the business level are unambiguous. In the next section, we present our methodology for deriving system use cases from a business use case. We discuss briefly as to how a business use case could be selected, and how it is detailed in the later steps.

3. METHOD FOR TRANSLATING A BUSINESS USE CASE TO SYSTEM USE CASES

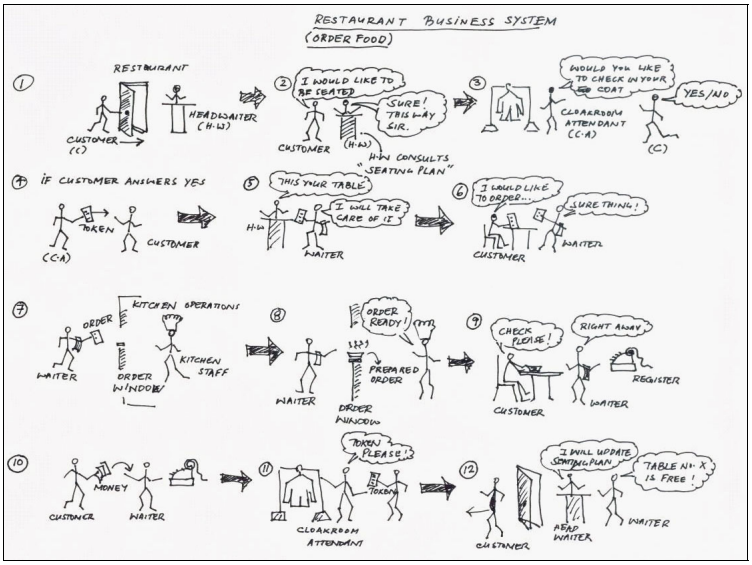
In this section we explain our method for translating business-use-cases to system-use-cases. This method (Table 1) assumes that a business use case model is already available.

For demonstrating the methodology we choose a restaurant system, and modeled one of its core business processes, *Order Food*. The 7-step

Table 1. Method overview

1. Select a BUC
2. Describe internal business process
3. Assign Role responsibilities
4. Map entity-role interaction
5. Task flow categorization and process innovation
6. Develop system-use-case diagram
7. Generate system-use-case description

Figure 1. Storyboard for Order Food process



ıre

method begins with identifying a business use case, and ends by generating a system use case model for a possible system.

The Method:

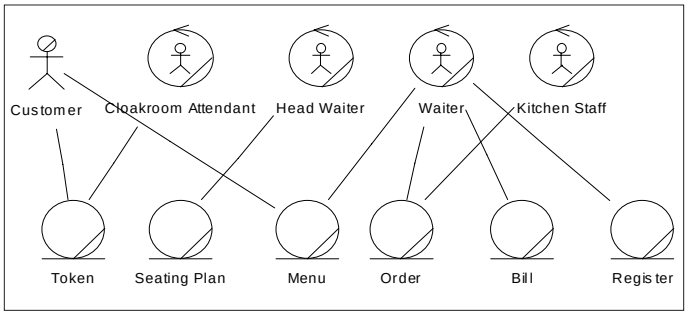
Step-1. Select a business use case (BUC), and set the scope based on the strategic business goal: a) Identify the business actors, b) Determine the business actor goals.

The core processes support a customer directly, and are the most visible processes to someone external to the business. In this case a business actor would be a *Customer*. For the *Customer* business actor,

an important business relationship with the restaurant would be *Order Food.* This would translate into the *business goal* for the *Customer.*

Business USE CASE # New	BUC 1.0	
BUC Name	Order Food	
ACTOR	Customer	
BUSINESS WORKER New	Head waiter, Cloakroom attendant, Kitchen staff, waiter.	
Purpose (1 phrase)	This use case shows how the order food business process functions.	
Overview and scope	This business use case documents…….customer leaves the restaurant after paying (if meal was ordered).	
Level New	*Core/Value added process, support level*	
Pre-conditions	The restaurant is open for business. They are serving customers.	
Post conditions in words	The customer pays for his/her meal and leaves the restaurant.	
Trigger	Customer enters the dining room.	
MAIN SUCCESSFUL SCENARIO in numbered sequence	**Actor Action**	***Business System Action*** New
	1. Customer enters the restaurant	2. the head waiter greets the customers and inquires about seating preferences.
		3. Cloakroom attendant enquires if the customer wishes to check in his/her coat
	4. The customer hands over the coat.	5. the attendant attaches a token to the coat and puts it up in the hanger and hands a copy of the token to the customer
		6. The headwaiter assigns a table and assigns a waiter to the table.
	7……	8……
OTHER SUCCESSFUL SCENARIOS	**Step**	**Branching Action**
	2a	<<Branching action of step 2 above>>
UNSUCCESSFUL SCENARIOS		
STRATEGIC DEPENDENCIES	New <<*Resource*>>Resource dependency: table availability, waiter availability	
BUSINESS RULES	<< *Rules that would dictate the flow*>>	

Figure 2. Entity/Object interaction diagram



Step-2. Identify internal processes that would support the externally visible business process: a) Generate a process flow for the business process (order food); b) Document this process using a business use case template.

For our example problem [Jac94] we choose the "order food" business use case for detailed analysis. The illustration (Fig.1) is optional and can be substituted with a process summary. Other visualizations like process flow diagrams can also be used. We chose story boards because they are simpler to illustrate and does not require much training.

A detailed business process description is captured using a business use case template shown in Table 2, which has been derived from a system use case template. "BUC#" field would help in linking a business use case template to a system use case template at a later stage (Step 7) for addressing the traceability issue. The BUC template helps in separating *business actor* actions from *business system* actions. The *new* tags shown in the business template is to show how the business use case template differs from a system use case template [Son04a].

Step-3. Assign roles to the tasks identified in Step-2: Develop a sequence diagram to show assignment of tasks.

Step-2 helped in identifying business workers who are internal to the business use case. In Step-3 we map the activity (in Table 2) to the roles as follows: *Head waiter:* Assign Table, Assign table waiter; *Cloak room attendant*: Check in coat, Check out coat; *Table Waiter*: Present Menu, Take order, Place order, Compute bill, Process payment; *Kitchen Staff*: Notify waiter, prepare order.

Step-4. Identify the business entities and interactions with the business workers/actors.

In this example, by looking at the story board Fig.1 we can identify the explicit business entities. The object interaction diagram (refer Fig. 2) displays the entities that are manipulated or created. Song et al [Son04b] present a detailed method for identifying objects in the use case descriptions and process summaries.

Business entities identified: *Token, Seating plan, Menu, Order, bill, transaction register*

Step-5. Aim for process innovation by categorizing task and information flow:

a) Task flow categorizations involves making decisions about tasks that could be automated, computer-supported, or manual, b) Generate a system sequence diagram [Lar05].

This step has a degree of subjectivity involved because the choice to automate something has to be justified from a business stand-point, and a technical/systems stand-point. The new design should aim to make the business process more efficient and effective.

The systems sequence diagram shows which activities can now be displaced/delegated to the external computer system. This would help in visualizing the system input and also in identifying system actors in the system use case diagram.

Step-6. Develop a system use case diagram: a) The system actors can be identified from the system sequence diagram, b) The actor goals are represented by the system use cases in the use case diagram.

The system sequence diagram reveals candidate actors for the system being designed. The tasks being accomplished by the system become system use cases in the use case diagram.

System actors –and actor goals

Head waiter- Assign waiter, Assign table, *Customer*- Process coats, *Kitchen Staff* -Fulfill orders, *Waiter*- Process order.

Step-7. Develop a system use case description for the use cases identified from the business use case: a) Use the system use case template for documentation. b) Do not name the system till other business use cases have been translated. It might yield additional system use cases.

A system use case template is used to document the system functionality. An additional field (Row 3, Table 3) is included to link this system use case to the source business use case. This helps in ensuring traceability.

Figure 3. System sequence diagram for new business process

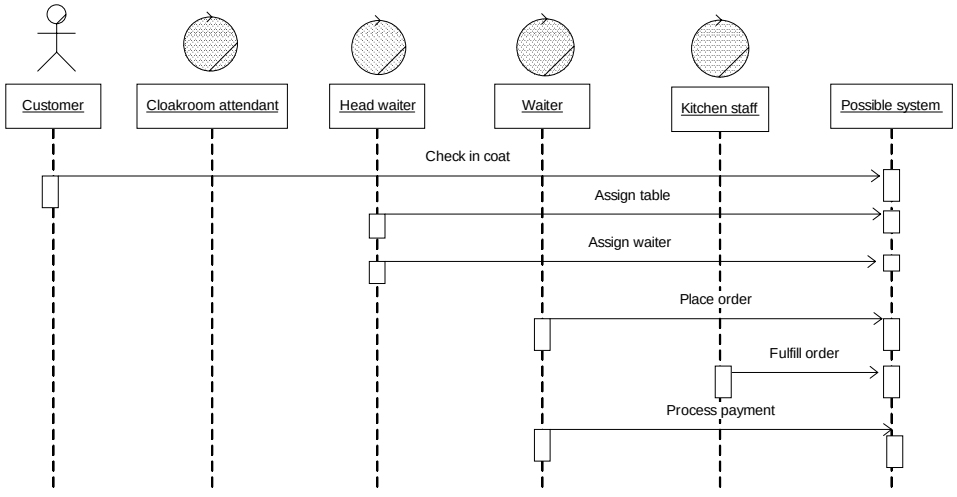


Figure 4. Partial System use case model based on one business use case

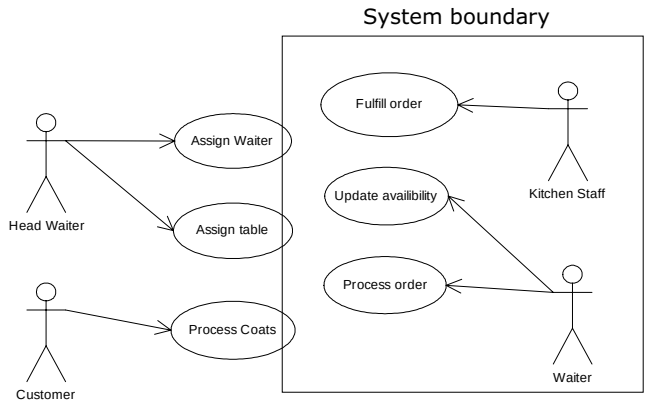


Table 3. System use case template [Son04a]

USE CASE #	SUC 1.0
USE CASE Name	Process order
Derived from Business use case	BUC 1.0 ⇐ **New**
ACTOR	
Purpose (1 phrase)	
…….	……
……..	……
Creation Date	
Other comments	

4. CONCLUSION

In this paper we presented a method for translating a business use case into system use cases. This approach helps traceability of requirements. We presented a technique for documenting the business use case using a business use case template derived from the system use case template. UML 2.0 diagrams were used to represent the operationalization of the steps. Future research would involve testing the 7-step method in terms of perception and performance [Moo03] in multiple domains, by analysts with varying levels of expertise to validate its effectiveness. We would like to mention that some amount of subjectivity exists in what is automated and what is not. This can at times significantly change the final output of the method.

REFERENCES

[Cur92] Curtis, W., Kellner, M. I., and Over, J., (1992) Process Modeling. *Communications of the ACM*, Vol. 35, No. 9, pp. 75–90.

[Fow97] Fowler, M. (1997) *UML distilled*, Addison-Wesley

[Gia01] Giaglis, G.M. (2001) A taxonomy of business process modeling and information systems modeling techniques. *International Journal of Flexible Manufacturing Systems*, 13(2): 209-228.

[Jac92] Jacobson, I., et al.(1992). *Object oriented software engineering: a use case driven approach*. Wokingham UK: Addison-Wesley.

[Jac94] Jacobson, I., Ericsson, M., Jacobson, A. (1994). *The Object Advantage, Business Process Reengineering with Object Technology*, Addison-Wesley Publishing Company.

[Kru03] Krutchen, P (2003) *The rational unified process an introductions*. Addison-Wesley

[Kav04] Kavakli, E. (2004) Modeling organization goals: Analysis of current methods, *ACM Symposium on applied computing*.

[Lam01] van Lamsweerde, A. (2001) Goal oriented requirements engineering: A guided tour in the *5th IEEE international symposium on requirements engineering. RE'01*. Toronto: Springer

[Lar05] Larman, Craig. (2005). *Applying UML and Patterns: An introduction to object oriented analysis and design*, Upper saddle River, Prentice Hall

[Ng 02] Ng,Pan-Wei (2002) Effective Business Modeling with UML: Describing Business use Cases and Realizations from http://www-128.ibm.com/developerworks/rational/library/905.html

[Oul 95] Ould M.A,(1995) *Business processes- Modeling analysis for re-engineering and improvement.* John Wiley and Sons.

[RUP] Jacobson,I et. al (1999) *The unified software development process*, Reading MA: Addison Wesley.

[San02] Santander, V. and Castro, J. (2002). Deriving use cases from organizational modeling. In *Proceedings of the IEEE Joint International Conference on Requirements Engineering*, pp.32–39, Essen, Germany.

[Son04a] Song, I. (2004). *Object oriented analysis and design using UML: A practical guide*. Lecture notebook, Pearson Custom Publishing.

[Son04b] Song, I. et al (2004). A taxonomic class modeling methodology for object oriented analysis. In J. Krogstie et al (Eds.), *Information modeling methods and methodologies* , Idea group publishing.

[Vas01] Vasconcelos, A., et al. (2001). A framework for modeling strategy, business processes and information systems from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=950424

[Yu93] Yu, E. (1993) Modeling organization for information systems requirements engineering., *Proc Re'93, IEEE*, pp. 34-41

[Yu97] Yu, E. (1997) Towards modeling and reasoning support for early phase requirements engineering. *Proc. RE'97 3rd international symposium in requirements engineering, Annapolis*, pp. 226-235