Integrating Lightweight Systems Analysis into the United Process by Using Service Responsibility Tables

Xin Tan
Fairleigh Dickinson University, xtan@fdu.edu

Steven Alter
University of San Francisco, alter@usfca.edu

Keng Siau
University of Nebraska - Lincoln, siauk@mst.edu

Integrating Lightweight Systems Analysis into the Unified Process by Using Service Responsibility Tables

Xin Tan
Fairleigh Dickinson University
xtan@fdu.edu

Steven Alter
University of San Francisco
alter@usfca.edu

Keng Siau
University of Nebraska - Lincoln
ksiau@unl.edu

ABSTRACT

This paper is a step toward establishing direct, but non-automatic links between lightweight (semi-formal) analysis methods for business professionals and heavyweight analysis methods for IT professionals. After noting the importance of user involvement in obtaining accurate and meaningful user requirements, the paper summarizes the Unified Process, a software development methodology that employs Unified Modeling Language (UML). Another section in the paper summarizes previous extensions of the work system method that produced a lightweight analysis tool called Service Responsibility Tables (SRTs). This paper uses a straightforward example to demonstrate a set of heuristics for translating between service responsibility tables produced by business professionals and UML diagrams that IT professionals can use as a partial basis for programming. This type of guideline-based link between lightweight and heavyweight methods could lead to more effective user involvement in requirements determination and reduce failure rate in IT projects.

Keywords

Requirements, lightweight analysis, work system, the Unified Process.

INTRODUCTION

The rate of failure in information systems (IS) development projects is disappointingly high (Johnson, Boucher, Connors, and Robinson, 2001). An important aspect of this problem is that managers and other business professionals often do not think about these systems with enough depth and rigor, and often are not adequately engaged in the analysis and design of these systems. One of many reasons for this inadequate engagement is the lack of organized, simple, and easy-to-use methods that managers and business professionals can use for understanding systems in their own terms and at whatever level of detail is appropriate for a particular situation (Alter, 2007a).

Methods for business professionals should differ substantially from analysis and design methods for information technology (IT) professionals whose goal is the production of software. Production of software that is traceable to rigorous user specifications requires formal, "heavyweight" specifications that are rigorous, precise, and complete. In contrast, business professionals would probably do best with lightweight methods that are semi-formal and have only a few rules and practices that are easy to follow. Lightweight methods for business professionals would help them think about a system, establish a personal understanding, and identify questions and issues. These methods would not have the same level of rigor and depth of detail as methods that IT professionals use for producing software that is verifiably linked to complete user requirements. Nonetheless, such methods would help in in-depth analysis, system-related communication and collaboration with other business professionals and with IT professionals.

The distinction between lightweight and heavyweight methods has existed for a number of years in methods for IT professionals. Examples of lightweight methodologies for professional programmers include agile modeling, extreme programming, adaptive software development, the Crystal family of methodologies, feature driven development, and the ICONIX process (Wikipedia, 2008).

This paper is a step toward establishing direct, but non-automatic links between lightweight methods for business professionals and heavyweight methods for IT professionals. It adds to a recent extension of an existing lightweight method for business professionals called the work system method (Alter, 2003, 2006). The recent extension involved the introduction of the service value chain framework and related tools called service responsibility tables (SRTs) (Alter, 2007a, 2008). This paper demonstrates that heuristics can be used to link several types of service responsibility tables to related use case and class diagrams in Unified Modeling Language (UML), which is the recommended analysis and modeling method in the Unified Process (UP), a popular software development methodology. Extensions of this paper's contribution would add links between other types of service responsibility tables and other UML diagrams, leading toward practical approaches for linking lightweight analysis for business professionals with heavyweight analysis for IT professionals.

This paper proceeds as follows. First, it introduces the requirements determination issues in the UP to reveal the need for lightweight analysis methods for business professionals. After summarizing the service value chain framework and basic ideas about service responsibility tables as a lightweight analysis method, it uses a straightforward example to demonstrate heuristics for translating between service responsibility tables produced by business professionals and UML diagrams that IT professionals can use as a partial basis for programming.

REQUIREMENTS DETERMINATION AND THE UNIFIED PROCESS

A set of complete and accurate requirements are critical for the ultimate success of IS development projects. This is the reason why requirements determination is a crucial activity recognized by IS practitioners and researchers (Browne and Ramesh, 2002; Pohl, 1994).

Requirements Determination

Requirements determination is a set of activities that assess the functionality required in both the work system that is being supported and the software itself. Pohl (1996) identified four subtasks within requirements determination: requirements specification, requirements negotiation, requirements representation, and requirements validation.

- Requirements specification is to understand the organizational situation that the system under consideration aims to improve and describe the needs and constraints of the system under development
- Requirements negotiation is to establish an agreement on the requirements of the system among the various stakeholders involved in the process
- Requirements representation is to develop a mapping of real-world needs onto a requirements model.
- Requirements validation is to ensure that the derived specification corresponds to the original stakeholder needs and conforms to the internal and/or external constraints set by the enterprise and its environment.

Requirements determination for information systems is a communication-intensive, iterative, and creative activity. Holtzblatt and Beyer (1995) claimed that the success of information requirements determination is ultimately dependent on how well people can communicate and understand each other in the process. Although communication issues and cognitive problems are important in information requirements determination (Siau and Tan, 2003, 2005, 2006), they are often downplayed or even neglected by systems development methodologies.

The Unified Process

With the pervasive adoption of object-oriented modeling and programming, the Unified Process (UP), which is based on a spiral model, has had an enormous influence on IS development. The UP is described as a use-case driven, architecture centric, iterative, and incremental methodology for object-oriented systems development (Jacobson, Booch, and Rumbaugh, 1999). Within the UP, each software life cycle contains four phases – Inception, Elaboration, Construction, and Transition (Jacobson et al., 1999). In addition, five core workflows cut across the set of four phases. The five core workflows are Requirements, Analysis, Design, Implementation, and Test. Each of the UP phases is divided into iterations, or mini-projects. A typical iteration crosses all five of the core workflows to a greater or lesser extent. For instance, an iteration during the Elaboration phase may involve more effort during the Requirements and Analysis workflows, while another iteration during the Construction phase may focus heavily on Design, Implementation, and Test workflows (Jacobson et al., 1999).

The UP provides a general framework for developing an information system iteratively. Even though it does not specify the detailed procedures and techniques for conducting the activities in the workflows and phases, the majority of UP practitioners suggest the use of the Unified Modeling Language (UML) as a primary modeling method in the UP. UML has emerged as the computer industry's standard for object-oriented modeling language (Kobryn, 1999). The Object Management Group (OMG), a computer industry consortium, is responsible for UML standards. The current version 2.0 of UML has thirteen types of diagrams that can be used in various modeling activities throughout systems development.

In the UP, the Requirements workflow is used to determine the requirements of the system being built. The fundamental objective of the Requirements workflow is to work toward building the right system. Iterations in the Inception and Elaboration phases focus heavily on the Requirements workflow.

Two UML diagrams, the use case diagram and class diagram, are used extensively in the Requirements workflow in the Unified Process. (Other UML diagrams are used in other workflows such as Analysis and Design, and Implementation.) Use cases are used to capture user requirements. Each use case diagram contains a number of use cases and each use case describes a major business process of the software system. The sum of these use cases defines the total functionality of the software system. The software is designed and implemented to support these use cases.

Class diagrams are used in conjunction with use cases to specify the classes in the software system. The UP employs UML class diagrams for capturing the contents of the domain model, which describes the important things and concepts associated with the system. Within the UP, these things and concepts are represented as classes and the various kinds of relationships among classes.

Systems analysts and other IT professionals use the formal notation of UML to document the requirements and analysis outcomes for designing and implementing IT-based solutions. However, empirical studies have indicated that UML has a number of shortcomings as a tool for requirements analysis. For instance, Glinz (2000) identified nine deficiencies in UML in analyzing the requirements for a medical care system. Some other studies found that business people have difficulties in learning and using formal UML diagrams (Siau, Erickson, and Lee, 2005; Siau and Loo, 2006). In addition, UML is not easily understood by business professionals due to its emphasis on the use of technical artifacts such as classes and use cases. Consequently, these formal analysis and modeling methods may not be appropriate for users of the to-be-built software.

In conclusion, class diagrams, use case diagrams, and other UML diagrams are basically heavyweight analysis methods used by IT professionals for formal documentation of requirements. They are far from ideal for business professionals to use as lightweight analysis methods.

SERVICE RESPONSIBILITY TABLES AS A LIGHTWEIGHT ANALYSIS METHOD

UML is a heavyweight analysis method that is too complicated for business professionals to learn and use in a short period of time. The systems analysis and design process would benefit greatly if a form of lightweight analysis could help business professionals participate actively in the requirements specification process. Although business professionals should be able to obtain help from IT professionals during the requirements specification phase, some form of lightweight analysis for business professionals would be helpful if it would 1) enable business professionals to participate actively in the analysis process and 2) provide a means through which business professionals could provide significant inputs into the more precise and rigorous heavyweight analysis needed to produce testable computer programs and IT systems. The work system approach (Alter, 2003, 2006) was developed specifically to help business professionals understand, evaluate, and analyze systems in organizations at a level of detail that is appropriate for their particular situation, regardless of whether IT plays a central role.

The work system approach and service value chain framework

The work system approach developed over the last decade has generated a number of concepts, frameworks, and tools that are at least potentially useful for business professionals. These include: the work system framework, work system life cycle model, work system snapshot, and work system principles, all of which have been described a number of times (Alter, 2003, 2006, 2008). More recently, widespread interest and concern about services and the service economy led to an attempt to extend the work system approach to incorporate the unique characteristics of services. The main products to date of those efforts are the service value chain framework and service responsibility tables (Alter, 2007a, 2007b, 2008).

The service value chain framework (Figure 1) was developed to depict generic activities and responsibilities of both service providers and customers (Alter, 2007b). These activities and responsibilities may occur before, while, and after a specific service is delivered to a specific customer. The form of the service value chain framework is based on the widely accepted observation that services tend to be co-produced by service providers and service consumers. The service value chain framework represents concepts related to services, such as coproduction of value, value capture, service interactions, back stage and front stage during the production and consumption of services, negotiations leading to service level agreements, provider and consumer preparation prior to instances of service delivery, negotiation of requests during specific instances of service provision, service fulfillment, and service follow-up (Alter, 2007a, 2008).

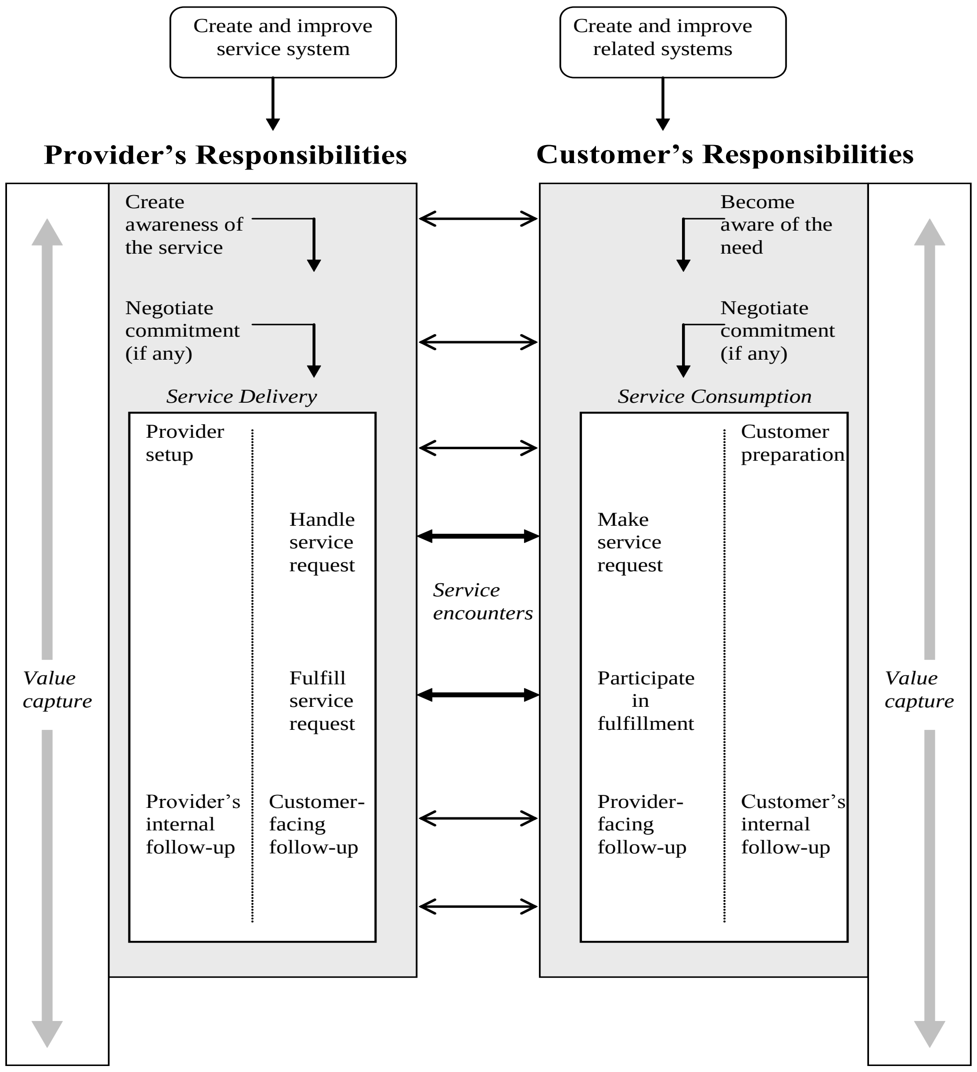
Create and improve service system
Create and improve related systems
Provider's Responsibilities
Customer's Responsibilities
Create awareness of the service
Negotiate commitment (if any)
Service Delivery
Provider setup
Handle service request
Fulfill service request
Provider's internal follow-up
Customer-facing follow-up
Become aware of the need
Negotiate commitment (if any)
Service Consumption
Customer preparation
Make service request
Participate in fulfillment
Provider-facing follow-up
Customer's internal follow-up
Service encounters
Value capture
Value capture

Figure 1. Service Value Chain Framework (Alter, 2007a, 2008) slightly updated

Service Responsibility Tables

The two-sided format of the service value chain framework translates directly into a useful and flexible analysis tool called Service Responsibility Table (SRT). SRTs strip out the terminology about generic service steps and emphasize coproduction of value by identifying active or passive responsibilities of both service providers and service consumers throughout a service process. As illustrated in the first two columns of Table 1, the simplest form of SRT resembles a two-column swimlane diagram, with one column for providers and one for customers, and with specific provider and customer roles indicated clearly. The entries in the first two columns of Table 1 are all activities, although some SRT entries can be responsibilities, such as a patient's responsibilities during a physical exam or a traveler's responsibilities during an airplane flight. It is easy to extend a two-column SRT into a three-column SRT that adds a new column for any of a number of topics that might be important for analyzing a particular system. The third column in Table 1 identifies information needed or generated at each step. The example is based on the illustrative example from Alter (2006) that combines aspects of a number of real-world loan approval processes.

Initial empirical evidence from classroom usage by MBA and EMBA students suggests that SRTs are potentially useful tools (Alter, 2007a). Use of an SRT early in an analysis can serve several purposes:

- Clarifying scope and context of the service without requiring mastery of details that will be clarified later in the analysis by using detailed representations of workflow and logic.
- Focusing attention on activities and responsibilities rather than on details of technology and information.
- Identifying job roles that are involved.
- Bringing customer responsibilities into the analysis.
- Identifying service interactions (rows with both provider and customer responsibilities) and other steps that are not visible to customers.

Provider Activity or Responsibility	Customer Activity or Responsibility	Information Needed or Generated
Loan officer identifies businesses that might need a commercial loan.		Profiles of businesses
Loan officer contacts potential loan applicant.	Potential **loan applicant** agrees to discuss the possibility of receiving a loan.	Contact information
Loan officer discusses loan applicant's financing needs and possible terms of the proposed loan.	Potential **loan applicant** discusses financing needs.	Possible terms of the loan
Loan officer helps loan applicant compiles a loan application.	**Loan applicant** compiles loan application.	Loan application
Loan officer and senior credit officer meet to verify that the loan application has no glaring flaws.		Loan application
Credit analyst prepares a "loan write-up" summarizing the client's financial history, providing projections of sources of funds for loan payments, etc.		Loan write-up, including financial history, projection of loan payments
Loan officer presents the loan write-up to a senior credit officer or loan committee.		Loan write-up
Senior credit officer makes approval decision.		Decision on the loan application
Loan officer informs loan applicant of the decision.	**Loan applicant** accepts or declines an approved loan.	Decision on the loan application
Loan administration clerk produces loan documents for an approved loan that the client accepts.		Accepted loan application

Table 1. Three-Column Service Responsibility Table (SRT)

LINKING SRTS TO THE UNIFIED PROCESS

The IS field has a long history of trying to develop links between different levels of specification and analysis. One of the earlier attempts occurred in MIT's Automatic Programming Group, which worked on initial concepts for linking different levels of language, with the ultimate, but still unrealized goal of translating automatically from natural language specifications to code (Ruth et al, 1980). In a much more recent example, IBM researchers have developed and used in an approach called "model driven business transformation," which operates through links between a strategy model, an operations model, an execution model, and an implementation model (Lee, 2005).

An overarching goal of our research is to establish links between lightweight analysis by business professionals and heavyweight analysis by IT professionals. The business professionals would use SRTs to clarify their understanding of the business situation while the IT professionals would use UML diagrams as a rigorous basis for programming. As a step toward this link between lightweight and heavyweight analysis, we propose a set of heuristics for transforming two types of SRTs into corresponding UML class diagrams and use case diagrams. A consistent way to perform that translation could be a step toward empowering business professionals to participate more fully in the requirements specification and representation. This type of translation would help integrate the lightweight analysis with the heavyweight analysis in the Unified Process.

Two heuristics are discussed below to demonstrate the transformation of SRTs into use case and class diagrams.

Heuristic for transforming an SRT into a use case diagram:

(i) Produce a two-column SRT that summarizes activities and responsibilities of service providers and customers.

(ii) Identify the various types of service providers and customers in the SRT and consider them as actors in a use case diagram.

(iii) Identify the various activities and responsibilities associated with each type of service provider and customer. Treat these activities and responsibilities as use cases in a use case diagram.

(iv) Link actors with corresponding use cases; show "extends" or "includes" relationships among use cases based on the SRT.

As an example, the use case diagram in Figure 2 was produced from the SRT in Table 1 by following the above heuristic.

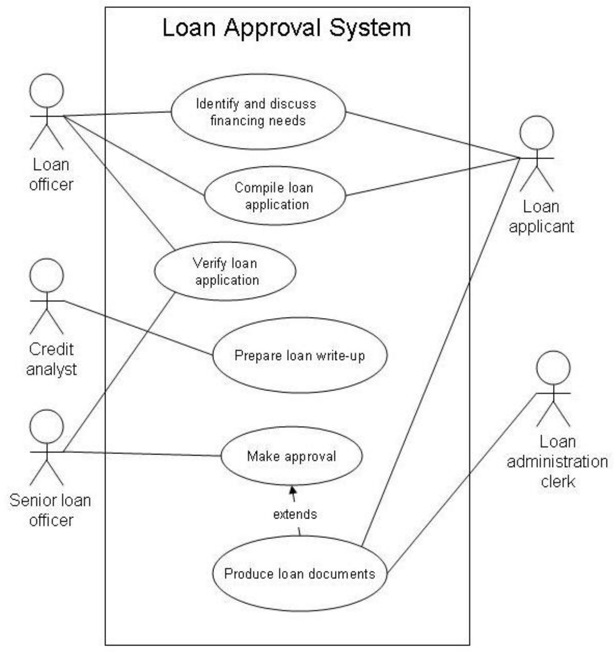

Figure 2. A Use Case Diagram transformed from the SRT for a loan approval system

Heuristic for transforming an SRT into a class diagram:

(i) Add a third column to the two-column SRT that is used to create the use case diagram. The third column lists information created and used in each step. Typically the third column will identify "business artifacts" (Nigam and Caswell, 2003) such as orders, invoices, contracts, and other documents containing many fields of data that can be identified later.

(ii) Identify all of the actors mentioned in the first two columns and all of the business artifacts and information items mentioned in the third column of the SRT.

(iii) Decide what are the relevant entity types, what are the related attributes, and what are the relationships between the entity types.

(iv) Fill in missing entity types, attributes, and relationships that may not have been mentioned explicitly but are required for a complete class diagram.

(v) Produce class diagrams and ensure that they capture the information requirements expressed by the SRT.

The figure below is an initial class diagram transformed from the SRT in Table 1. Methods and multiplicities among classes are not depicted in this class diagram because it mainly serves as a domain model.

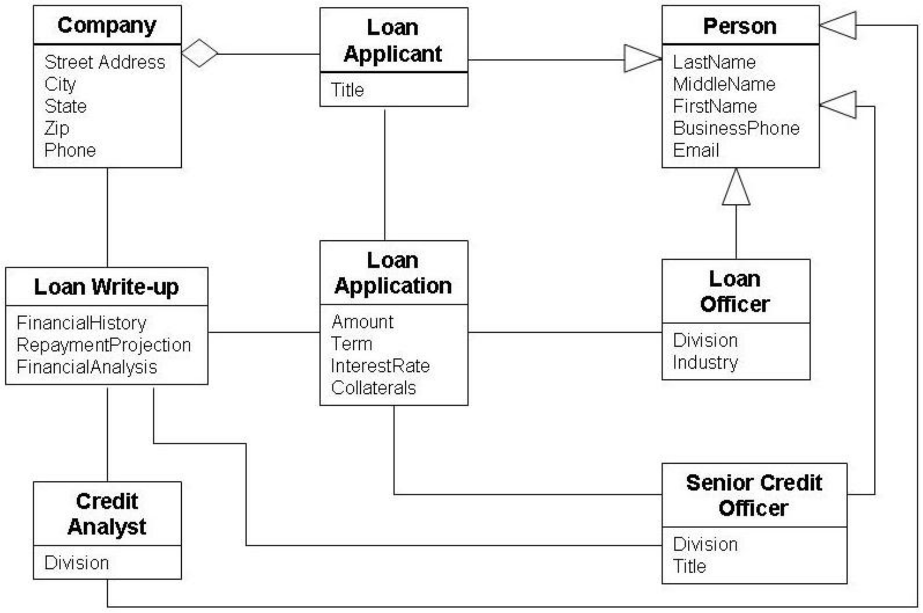

Figure 3. A Class Diagram transformed from the SRT for a loan approval system

The goal of this research is not to make SRTs as rigorous or complex as the corresponding UML diagrams. SRTs should be as easy to use and as informal as possible. They should remain a lightweight analysis method that can be used and understood by business professionals with little or no training. The translation between SRTs and UML diagrams is meant to be done manually by IT professionals who will recognize and fill in missing details based on implicit relationships, everyday business knowledge, and discussions with business professionals. Our attempt to develop this approach further will strive to balance simplicity of use and completeness of information.

OTHER USES OF SRTS IN THE UNIFIED PROCESS

The format of an SRT facilitates easy reuse as the analysis proceeds. For example, it is easy to generate a set of three-column SRTs by reusing the first two columns and including in each additional SRT a new column for any of a number of topics that might be important for analyzing a particular system (See Table 2).

The tabular structure of SRTs is also conducive to creation of various standard versions that can be defined and used with database or spreadsheet software. For example, systems analysis efforts in a particular firm might establish the common practice of beginning its systems analysis efforts by using SRTs with specific third columns such as issues and problems, customer benefits, key performance gaps, or constraints. Business professionals in that firm would use the standard two-column SRT to define the scope of the system to be analyzed, and then use SRTs with additional third columns as a starting point for exploring additional topics and issues.

The use of SRTs can be extended to other analytical activities in systems development. For example, if it is important to remember that certain groups of steps occur in parallel, it is possible to number the activities and use simple numerical conventions to indicate activities that occur in parallel. If it is important to record non-sequential precedence relationships in a SRT (rather than in other documentation), it is possible to add two numeric columns, one that numbers each activity and another that identifies one or more direct predecessors of each activity.

Topics related to problems or issues (by step)	**Topics related to the system's structure and requirements (by step)**	**Topics related to performance metrics (by step)**
Issues and problems	Goals and requirements	Activity rate
Participant or interpersonal issues	Pre-conditions	Duration (cycle time)
Information issues	Triggers	Delay between steps
Technology issues	Business rules	Defect rate
Confusion or training issues	Business or legal constraints	Rework rate
Points of friction	Post-conditions	Downtime
Reasons for delays, errors, rework	Special cases	Provider cost
Communication issues	Significant exceptions	Customer cost
Conflicts with culture or policies	Alternative paths or methods	Customer complaints
Legal or regulatory issues	Knowledge or skill requirements for participants	Information accuracy
External dependencies	Participant incentives	Information timeliness
Conflicts with other systems	Information used	Information availability
	Information generated	Information security
	Technology used	Technology performance
	Products and services produced (and used in other systems by customers or provider organizations)	Key performance gaps for important steps (Gap = desired vs. current value of an important metric)
	Possibilities for change	
	Things that cannot change	
	Benefits provided to customers	

Table 2: Examples of Typical Topics (by Step) for Additional Columns of a SRT (Alter, 2007a, 2008)

Requirements negotiation and validation

SRTs with a variety of third columns can be used for requirements negotiation and validation. Requirements negotiation requires stakeholders of the development process to reach consensus on system requirements. Many of the topics in Table 2 could be included in a series of SRTs that might clarify and summarize the issues. For example, SRTs containing columns about topics such as conflicts with culture or policies or points of friction could help stakeholders identify and discuss important issues that might be overlooked in purely UML-based analysis. Thus, such SRTs can help in requirements negotiation.

Requirements validation requires links between system specifications and the original stakeholder needs and internal and/or external constraints set by the enterprise and its environment. SRTs related to topics such as business constraints, legal constraints, or significant exceptions could aid in requirements validation.

CONCLUSION

The iterative and incremental nature of the Unified Process has great potential in encouraging active user involvement and accommodating the evolutionary nature of users' requirements. Although we fully recognize that UML's use case diagrams and class diagrams are key deliverables related to requirements definition in the Unified Process, we highlight that they are most appropriate for heavyweight analysis by IT professionals and far less appropriate for lightweight analysis by business professionals. Since IT professionals and business professionals have different background knowledge related to IT artifacts and notations, it is desirable to separate and link lightweight analysis by business professionals and heavyweight analysis by IT professionals for requirements determination. Our proposal to use SRTs in lightweight analysis and use heuristics to link the SRTs to heavyweight analysis is an attempt to use different techniques for each group's analysis purposes.

As we continue to develop these ideas, we hope to achieve the following goals:

- Understand the benefits and practical limitations of lightweight analysis such as the approach based on SRTs.
- Develop heuristics that can be used to transform other types of SRTs into UML diagrams that can be used for more precise documentation and analysis contributing to the development of testable software.
- Develop ways to complement and supplement UML diagram creation by using SRTs to capture ideas, issues, and requirements that are outside of UML's scope.

REFERENCES

1. Alter, S. (2003) 18 Reasons why IT-Reliant Work Systems Should Replace the IT Artifact as the Core Subject Matter of the IS Field, *Communications of the AIS,* 12, 23, 365-394.
2. Alter, S. (2006) *The Work System Method: Connecting People, Processes, and IT for Business Results*. Larkspur, CA: The Work System Press.
3. Alter, S. (2007a) Service Responsibility Tables: A New Tool for Analyzing and Designing Systems, in *Proceedings of the Thirteenth Americas Conference on Information Systems*, Keystone, CO.
4. Alter, S. (2007b) Systems as Services: Viewing the Customer as Co-Producer within the System Being Analyzed, in *Proceedings of the Seventh AIS SIGSAND (SIG Systems Analysis and Design) Symposium*, Tulsa, OK.
5. Alter, S. (2008) Service system fundamentals: work system, value chain, and life cycle, *IBM Systems Journal,* 47, 1, 71-85.
6. Browne, G. J., and Ramesh, V. (2002) Improving information requirements determination: a cognitive perspective, *Information and Management,* 39, 8, 625-645.
7. Glinz, M. (2000) Problems and deficiencies of UML as a requirements specification language, in *Proceedings of the 10th International Workshop on Software Specification and Design (IWSSD-10),* San Diego, CA.
8. Holtzblatt, K., and Beyer, H. R. (1995) Requirements gathering: The Human Factor, *Communications of the ACM,* 38, 5, 30-32.
9. Jacobson, I., Booch, G., and Rumbaugh, J. (1999) *The Unified Software Development Process*. Boston, MA: Addison-Wesley.
10. Kobryn, C. (1999) UML 2001: A standardization odyssey, *Communications of the ACM,* 42, 10, 29-37.
11. Lee, J. (2005) Model-Driven Business Transformation and the Semantic Web, *Communications of the ACM*, 48, 12, 75-77
12. Nigam, A. and Caswel, P. (2003) Business artifacts, an approach to operational specification, *IBM Systems Journal*, 42, 3, 428-445.
13. Pohl, K. (1994) The three dimensions of requirements engineering: a framework and its applications, *Information Systems,* 19, 3, 243 - 258.
14. Ruth, G, Alter, S., and Martin, W. (1980) A Very High Level Language for Business Data Processing, Technical Report, MIT Laboratory for Computer Science, MIT/LCS/TR-254.
15. Siau, K., Erickson, J., and Lee, L. (2005) Theoretical versus Practical Complexity: The Case of UML, *Journal of Database Management,* 16, 3, 40-57.

16. Siau, K., and Loo, P. (2006) Identifying the Learning Difficulties with Unified Modeling Language (UML), *Information Systems Management*, 23, 3, 43-51
17. Siau, K., and Tan, X. (2003) Information systems requirements determination and analysis: A mental modeling approach, in *Proceedings of the Ninth Americas Conference on Information Systems*, Tampa, FL.
18. Siau, K., and Tan, X. (2005) Improving the quality of conceptual modeling using cognitive mapping techniques, *Data and Knowledge Engineering*, 55, 3, 343-365.
19. Siau, K., and Tan, X. (2006) Using Cognitive Mapping Techniques to Supplement UML and UP in Information Requirements Determination, *Journal of Computer Information Systems*, 47, 59-66.
20. Wikipedia (2008) Lightweight methodology, retrieved on April 17, 2008 at http://en.wikipedia.org/wiki/Lightweight_methodology