

# Linking Lightweight and Heavyweight Systems Analysis by Converting Service Responsibility Tables into UML Diagrams

## **Xin Tan**

Fairleigh Dickinson University  
1000 River Road, H-DH2-06  
Teaneck, NJ 07666  
Phone: 201-692-7283  
Fax: 201-692-7219  
Email: [xtan@fd.edu](mailto:xtan@fd.edu)

## **Steven Alter**

University of San Francisco  
2130 Fulton Street, MH-314  
San Francisco, CA 94117-1045  
Phone: 415-422-6383  
Fax: 415-422-2502  
Email: [alter@usfca.edu](mailto:alter@usfca.edu)

## **Keng Siau**

University of Nebraska-Lincoln  
209 CBA, Department of Management  
Lincoln, NE 68588-0491  
Phone: 402-472-3078  
Fax: 402-472-5855  
Email: [ksiau@unl.edu](mailto:ksiau@unl.edu)

## **Abstract**

Heavyweight systems analysis approaches such as the use of Unified Modeling Language (UML) are inappropriate for business professionals who nonetheless need to participate actively in systems analysis and design processes to ensure that the system requirements reflect their needs. This paper proposes the use of a lightweight analysis approach based on Service Responsibility Tables (SRTs) to serve as a front-end to UML diagrams. Business professionals (with or without the help of IT professionals) can use this lightweight approach to specify at least part of system requirements. Subsequently, IT professionals can perform heavyweight analysis for the design and implementation of hardware and software. This work-in-progress paper presents heuristics for transforming two types of SRTs into the two primary UML diagrams. The research is on-going, and a plan for future research is presented.

## Introduction

Eliciting and documenting users' requirements for the system being built is a crucial activity in system development projects. Despite widespread agreement on the importance of user involvement during systems analysis, the level and quality of user involvement is still often inadequate. Users often have difficulty identifying and describing the capabilities and features they want. Even if the software totally reflects what they requested, the system often omits important capabilities that the users failed or forgot to request. Davis (1982) listed four sources of difficulties in information requirements determination: (i) the constraints on humans as information processors and problem solvers, (ii) the variety and complexity of information requirements, (iii) the complex patterns of interaction among users and analysts in defining requirements, and (iv) the unwillingness of users to provide requirements. Valusek and Fryback (1987) labeled these difficulties as communication obstacles "within" individual users, "among" users, and "between" users and analysts. Siau and Tan (2003, 2005) elaborated further on the cognitive underpinnings of such communication problems.

One way to improve user involvement and business/IT alignment is to empower business professionals by providing frameworks, terminologies, and methods that can help them contribute more effectively to IT-related projects. Instead of assuming that users and analysts should use the same methods and tools, we propose differentiating between lightweight analysis by business professionals and heavyweight analysis by IT professionals and then providing different techniques for different purposes.

Lightweight techniques are designed to help business professionals think through their own views of IT-reliant systems that they care about, with or without the help of IT professionals. The goal of lightweight analysis by business professionals is to understand the business situation and potential implications of any proposed change regarding process flow, information usage, and aspects of systems that are visible to them and their colleagues.

Heavyweight analysis is directed toward building software that fits well with the understanding generated during the lightweight analysis. Heavyweight analysis techniques are used by IT professionals, especially system analysts, to formally document the requirements and analysis outcomes for designing and implementing IT-based solutions.

With this division of analysis responsibilities, an organized approach for linking lightweight analysis and heavyweight analysis is desirable. We propose the use of Service Responsibility Tables (SRTs) as a tool for lightweight analysis by business professionals. We also suggest a set of heuristics to transform SRTs to UML diagrams, which are used as modeling techniques for heavyweight analysis by IT professionals.

## SRTs as Lightweight Analysis Tools

The work system approach was developed specifically to help business professionals understand, evaluate, and analyze systems in organizations at whatever level of detail is appropriate for their particular situation, regardless of whether IT plays a central role. Typically they would obtain help from IT professionals if changes in hardware and software might be needed. We believe that a practical, easily learned form of lightweight analysis for business professionals would be a good compromise between 1) assuming that they are incapable of analyzing anything in an organized manner and 2) assuming that the only legitimate form of analysis is precise, rigorous analysis needed to produce testable computer programs and IT systems.

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 extensions of the work system approach to incorporate issues and characteristics related to 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) depicts generic activities and responsibilities of both service providers and customers. 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 common observation that services tend to be coproduced by service providers and service consumers. The service value chain framework includes a large number of concepts related to services, such as coproduction of value, value capture, service interactions, differentiation between 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).

The two-sided format of the service value chain framework translates directly into a useful and flexible analysis tool called a Service Responsibility Table (SRT). SRTs strip out the terminology about steps in services per se, and emphasize the issue of coproduction by identifying the (active or passive) responsibilities of both service providers and service consumers throughout a service process. As shown 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 problems and issues at each step. Many other examples of topics for the third column are listed in Table 2.

An SRT serves some of the same purposes as a work system snapshot, and therefore could be used instead of a work system snapshot early in an analysis. Use of an SRT early in an analysis serves several purposes:

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

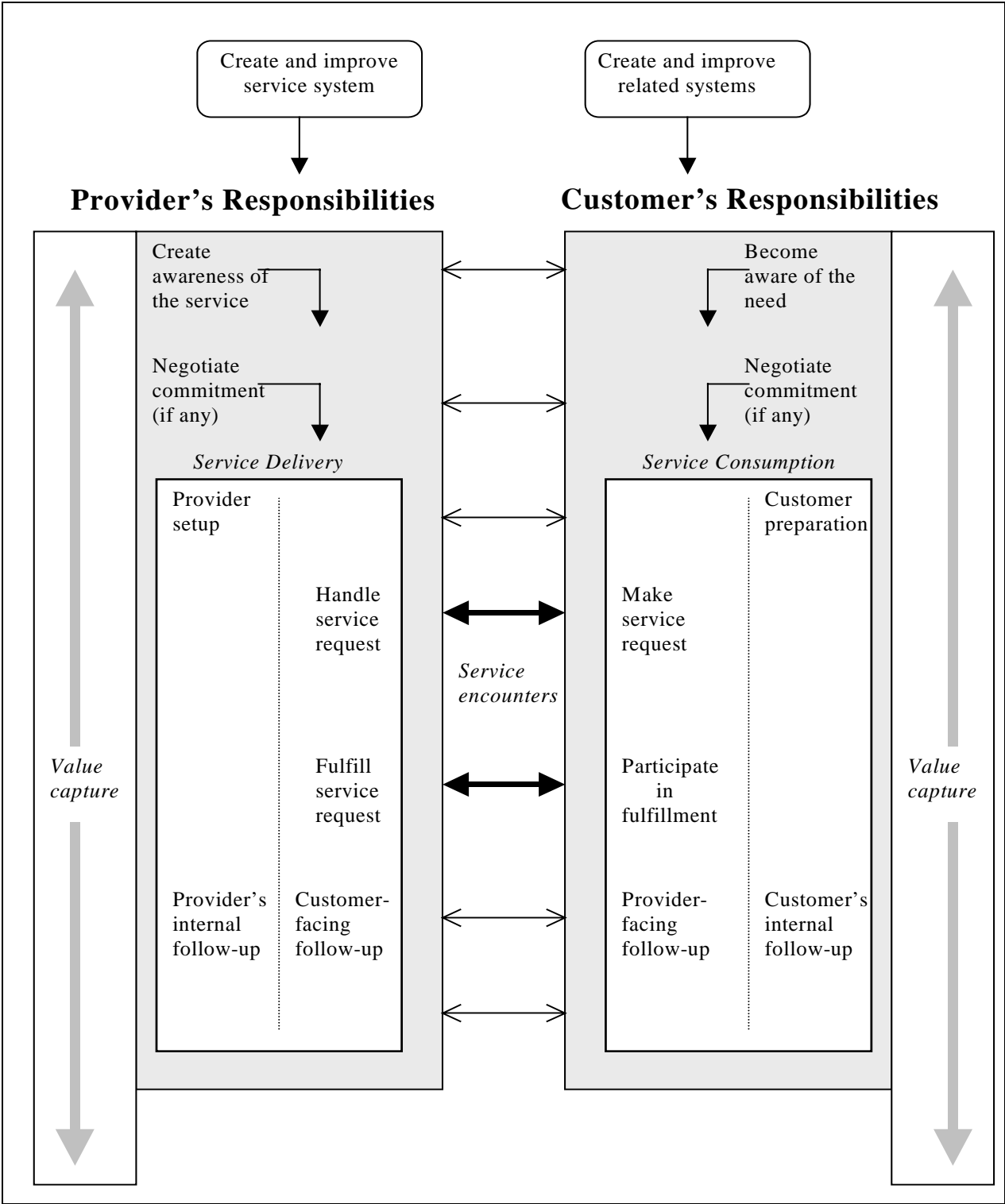


Figure 1: Service Value Chain Framework (Alter, 2008) slightly updated

Table 1: Three-Column Service Responsibility Table (SRT), Including a Column for Problems and Issues

Provider Activity or Responsibility	Customer Activity or Responsibility	Problems or Issues
<b>Loan officer</b> identifies businesses that might need a commercial loan.		Loan officers are not finding enough leads.
<b>Loan officer</b> contacts potential loan applicant.	<b>Potential loan applicant</b> agrees to discuss the possibility of receiving a loan	
<b>Loan officer</b> discusses loan applicant's financing needs and possible terms of the proposed loan.	<b>Potential loan applicant</b> discusses financing needs.	Loan officer is not able to be specific about loan terms, which are determined during the approval step, which occurs later.
<b>Loan officer</b> helps loan applicant compile a loan application	<b>Loan applicant</b> compiles loan application.	Loan applicant and loan officer sometimes exaggerate the applicant's financial strength and prospects.
<b>Loan officer</b> and <b>senior credit officer</b> meet to verify that the loan application has no glaring flaws.		20% of loans applications have glaring flaws.
<b>Credit analyst</b> prepares a "loan write-up" summarizing the client's financial history, providing projections of sources of funds for loan payments, etc.		10% rate of significant errors, partly because credit analysts use an error prone combination of several spreadsheets and a word processing program. Much rework due to inexperience of credit analysts.
<b>Loan officer</b> presents the loan write-up to a senior credit officer or loan committee.		Meetings not scheduled in a timely manner. Questions about exaggerated statements by some loan officers.
<b>Senior credit officer</b> or <b>loan committee</b> makes approval decision.		Excessive level of non-performing loans. Rationale for approval or refusal not recorded for future analysis.
<b>Loan officer</b> informs loan applicant of the decision	<b>Loan applicant</b> accepts or declines an approved loan.	25% of refused applicants complain reason is unclear. 30% of applicants complain the process takes too long.
<b>Loan administration clerk</b> produces loan documents for an approved loan that the client accepts		

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). Initial empirical evidence from classroom usage by MBA and EMBA students suggests that SRTs are potentially useful tools (Alter, 2007a).

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

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

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)
<ul style="list-style-type: none"> <li>• Issues and problems (See Table 1)</li> <li>• Participant or interpersonal issues</li> <li>• Information issues</li> <li>• Technology issues</li> <li>• Confusion or training issues</li> <li>• Points of friction</li> <li>• Reasons for delays, errors, rework</li> <li>• Communication issues</li> <li>• Conflicts with culture or policies</li> <li>• Legal or regulatory issues</li> <li>• External dependencies</li> <li>• Conflicts with other systems</li> </ul>	<ul style="list-style-type: none"> <li>• Goals and requirements</li> <li>• Pre-conditions</li> <li>• Triggers</li> <li>• Business rules</li> <li>• Business or legal constraints</li> <li>• Post-conditions</li> <li>• Special cases</li> <li>• Significant exceptions</li> <li>• Alternative paths or methods</li> <li>• Knowledge or skill requirements for participants</li> <li>• Participant incentives</li> <li>• Information used</li> <li>• Information generated</li> <li>• Technology used</li> <li>• Products and services produced (and used in other systems by customers or provider organizations)</li> <li>• Possibilities for change</li> <li>• Things that cannot change</li> <li>• Benefits provided to customers</li> </ul>	<ul style="list-style-type: none"> <li>• Activity rate</li> <li>• Duration (cycle time)</li> <li>• Delay between steps</li> <li>• Defect rate</li> <li>• Rework rate</li> <li>• Downtime</li> <li>• Provider cost</li> <li>• Customer cost</li> <li>• Customer complaints</li> <li>• Information accuracy</li> <li>• Information timeliness</li> <li>• Information availability</li> <li>• Information security</li> <li>• Technology performance</li> <li>• Key performance gaps for important steps (Gap = desired vs. current value of an important metric.)</li> </ul>

Many additional variations are consistent with a SRT's basic structure. 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 an 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.

## Transforming SRTs into UML Diagrams

Unified Modeling Language is the industry standard modeling language for developing information systems (Object Management Group, 2005). It is widely adopted by IT professionals and supported by CASE tools (Kobryn, 1999). UML 2.0 consists of 13 different diagrams and the complexity of UML is an issue (Siau and Cao, 2001). Despite its status as an OMG standard, the full acceptance of UML in practice is somewhat problematic. For example, many organizations that are using UML actually use only a small set of diagrams such as use cases and class diagrams (Dobing & Parsons, 2006; Grossman, Aronson, & McCarthy, 2005). Some empirical studies have indicated that UML has a number of shortcomings in systems analysis (Dawson & Swatman, 1999; Glinz, 2000). In addition, with its emphasis on the use of technical artifacts, such as classes and use cases, UML is not easily understood by business professionals. Several empirical studies have found that business people have difficulties in learning and using UML (Dawson & Swatman, 1999; Siau & Loo, 2006). Thus, the set of UML diagrams that is a key deliverable in many software projects is far from ideal for business professionals.

One way to address this disconnect is to translate from SRTs produced by business professionals to corresponding UML diagrams that IT professionals can use for the more precise analysis and documentation needed to produce testable software. A consistent way to perform that translation could be a step toward empowering business professionals to participate more fully in the analysis and design of IT-based systems.

A first step in this direction is to show how to translate from SRTs to use case diagrams and class diagrams, the two fundamental diagrams in UML (Booch, Rumbaugh, & Jacobson, 1999; Dennis, Wixom, & Tegarden, 2001). Two sets of heuristics are discussed below to transform SRTs into use case and class diagrams.

### ***Heuristic for transforming an SRT into a use case diagram***

1. Produce a two-column SRT that summarizes activities and responsibilities of service providers and customers.
2. Identify the various types of service providers and customers in the SRT and consider them as actors in a use case diagram.
3. 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.
4. 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.

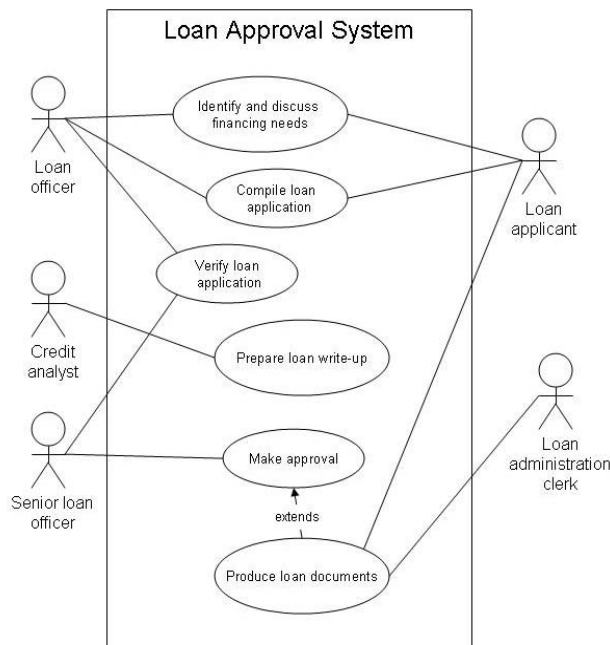


Figure 2. A Use Case Diagram generated from the SRT for a loan approval system

### ***Heuristic for transforming an SRT into a class diagram***

1. 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.

2. 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.
3. Decide what are the relevant entity types, what are the related attributes, and what are the relationships between the entity types.
4. Fill in missing entity types, attributes, and relationships that may not have been mentioned explicitly but are required for a complete class diagram.
5. 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.

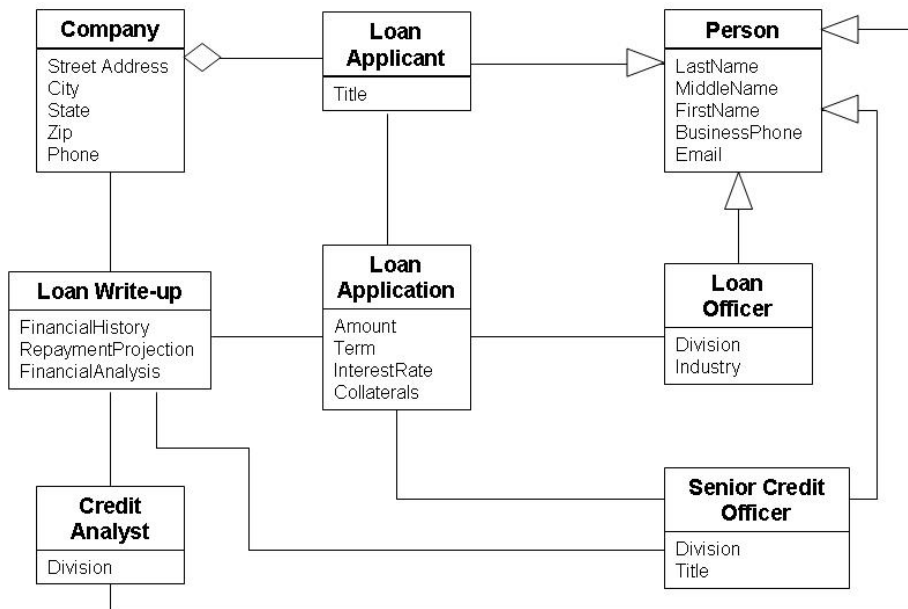


Figure 3. A Class Diagram generated from the SRT for a loan approval system

## Future Research Plan

We plan to take the following steps to identify and verify a comprehensive set of heuristics for transforming SRTs to UML diagrams:

- We will identify worked out UML examples in textbooks or other sources. These will be examples that are more or less completely worked out for a particular problem domain and include most of the UML diagrams, as opposed to individual diagrams that illustrate specific points about UML per se.
- For each fully worked out UML example, we will create a set of SRTs that captures as much information as possible from the UML diagrams. In a way, this is a “reverse-engineering” approach.
- For each fully worked out example, we will examine the corresponding SRTs and propose heuristic guidelines for converting these SRTs to the UML diagrams.
- We will review Table 2 to identify typically important topics that are not addressed or captured in UML.
- We will test our heuristics in classroom exercises involving systems analysis and design students. The students we divided into two groups. One group will attempt to produce UML diagrams directly from

descriptions of business situations. The other group will use SRTs as lightweight analysis and will then use the guidelines to convert those SRTs to UML diagrams.

- We will compare the results to determine whether the step of producing the SRTs was useful for the students.
- We will apply the results to extend our ideas about using SRT-based analysis as a step towards creating UML diagrams.

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.

## Conclusion

In this paper, we differentiate between lightweight analysis by business professionals and heavyweight analysis by IT professionals, and propose using different techniques for each group's purposes. In particular, we propose the use of Service Responsibility Tables as a tool for lightweight analysis by business professionals and suggest a set of heuristics to transform SRTs to UML diagrams, which are used as modeling techniques for heavyweight analysis by IT professionals. 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 SRTs from lightweight analysis 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

- 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.
- Alter, S. (2006). *The Work System Method: Connecting People, Processes, and IT for Business Results*. Larkspur, CA: The Work System Press.
- Alter, S. (2007a). *Service Responsibility Tables: A New Tool for Analyzing and Designing Systems*. Paper presented at the Thirteenth Americas Conference on Information Systems, Keystone, CO.
- Alter, S. (2007b). *Systems as Services: Viewing the Customer as Co-Producer within the System Being Analyzed*. Paper presented at the Seventh AIS SIGSAND (SIG Systems Analysis and Design) Symposium, Tulsa, OK.
- Alter, S. (2008). Service system fundamentals: work system, value chain, and life cycle. *IBM Systems Journal*, 47(1), 71-85.
- Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Reading, Mass.: Addison-Wesley.
- Davis, G. B. (1982). Strategies for Information Requirements Determination. *IBM Systems Journal*, 21(1), 4-30.
- Dawson, L., & Swatman, P. (1999). *The use of object-oriented models in requirements engineering: a field study*. Paper presented at the Twentieth International Conference on Information Systems, Charlotte, NC.

- Dennis, A., Wixom, B. H., & Tegarden, D. (2001). *Systems Analysis and Design : An Object-Oriented Approach with UML*. New York: Wiley.
- Dobing, B., & Parsons, J. (2006). How UML is used. *Communications of the ACM*, 49(5), 109-113.
- Glinz, M. (2000). *Problems and deficiencies of UML as a requirements specification language*. Paper presented at the 10th International Workshop on Software Specification and Design (IWSSD-10), San Diego.
- Grossman, M., Aronson, J. E., & McCarthy, R. V. (2005). Does UML make the grade? Insights from the software development community. *Information and Software Technology*, 47, 383-397.
- Kobryn, C. (1999). UML 2001: A standardization odyssey. *Communications of the ACM*, 42(10), 29-37.
- OMG. (2005). Introduction to OMG's Unified Modeling Language (UML) (Publication. Retrieved January 15, 2005: [http://www.omg.org/gettingstarted/what\\_is\\_uml.htm](http://www.omg.org/gettingstarted/what_is_uml.htm)
- Siau, K. & Cao, Q. (2001). Unified Modeling Language – A Complexity Analysis. *Journal of Database Management*, Vol. 12, No. 1, Jan-Mar 2001, pp. 26-34.
- Siau, K., & Loo, P. (2006). Identifying the Learning Difficulties with Unified Modeling Language (UML). *Information Systems Management*, 23(3).
- Siau, K., & Tan, X. (2003). *Information systems requirements determination and analysis: A mental modeling approach*. Paper presented at the Ninth Americas Conference on Information Systems (AMCIS'03), Tampa, FL.
- Siau, K., & Tan, X. (2005). Improving the quality of conceptual modeling using cognitive mapping techniques. *Data & Knowledge Engineering*, 55(3), 343-365.
- Valusek, J. R., & Fryback, D. G. (1987). Information Requirements Determination: Obstacles Within, Among, and Between Participants. In R. Galliers (Ed.), *Information analysis: Selected Readings* (pp. 139-151). Reading, MA: Addison Wesley.