Chapter

69 *SOMT Introduction*

This chapter describes the techniques that the SOMT method is based on. It also gives a first brief description of the method and the activities it consists of.

Last in the chapter is a list of references, which are referred to throughout this volume.

Background

Object-oriented analysis is a well-known and popular technique for understanding a problem and analyzing a system. There are many different versions of object-oriented analysis published as different methods in various books. The Object Management Group (OMG) has agreed upon a standard language for object-oriented methods and tools - the *Unified Modeling Language* (UML) and its notation ([36], [37]).

SDL [23] and MSC [25], on the other hand, are standardized notations used for specification and design of systems. They have a solid foundation in terms of support for design, formal verification and code generation. Especially for distributed, reactive, real-time applications SDL and MSC has a successful track record.

The scope of this volume is to present a method that integrates object-oriented analysis and SDL design with a special focus on application areas where SDL is known to be suitable. The intention is of course to provide developers with the best of both worlds: The strength of the object oriented analysis in the early phases together with the strong back-end provided by SDL and SDL tools.

The method has been given the name SOMT (short for *SDL-oriented Object Modeling Technique).* As the name indicates the method is essentially an adaptation of OMT to the requirements given by the special application area of distributed, reactive, real-time systems together with the usage of SDL for the design.

Another object-oriented analysis method that has provided input to SOMT is the Objectory method by Jacobsson [18]. In particular, the idea of use cases that originated from this method has been incorporated into SOMT.

A special focus of SOMT is to provide solutions that can be directly applied in a development project. For instance, it provides guidelines about how to apply OMT in a way that enables a smooth transition between analysis and design.

The volume is focused on how to use the different notations and concepts of object-oriented analysis and SDL and does not provide a detailed description on the specific notations. It is therefore recommended that you are at least reasonably familiar with both the general concepts of object-oriented analysis, the OMT notation and SDL before reading this report. [20] is a good introduction to the OMT notation and there exist several text books on SDL, for example [26].

The rest of this document is structured as follows:

- This chapter, *SOMT Introduction*, gives an overview of the SOMT method and introduces the different activities
- chapter 70, *SOMT Concepts and Notations* describes briefly the different concepts and notations that SOMT uses
- chapter 71, *Requirements Analysis* covers the requirements analysis activity
- chapter 72, *System Analysis* presents the system analysis activity
- chapter 73, *From Analysis to Design* discusses the steps and considerations when moving from analysis to design
- chapter 74, *System Design* moves on to system design
- chapter 75, *Object Design* deals with object design
- chapter 76, *SOMT Implementation* ends the SOMT activities with a description of the implementation activity
- chapter 77, *SOMT Projects* focuses on how to organize a SOMT project

Overview of the SOMT Method

The SOMT method provides a framework that shows how to use object-oriented analysis and SDL-based design together in a coherent way. The framework is based on describing the analysis and design of a system as a number of activities. Each activity deals with some specific aspects of the development process. The work done in the activities is centered around a number of models, that document the result of the activities. As an integral part of the SOMT method, guidelines are given for the transition between the different models. The activities and the main models used in SOMT are illustrated in Figure 604.

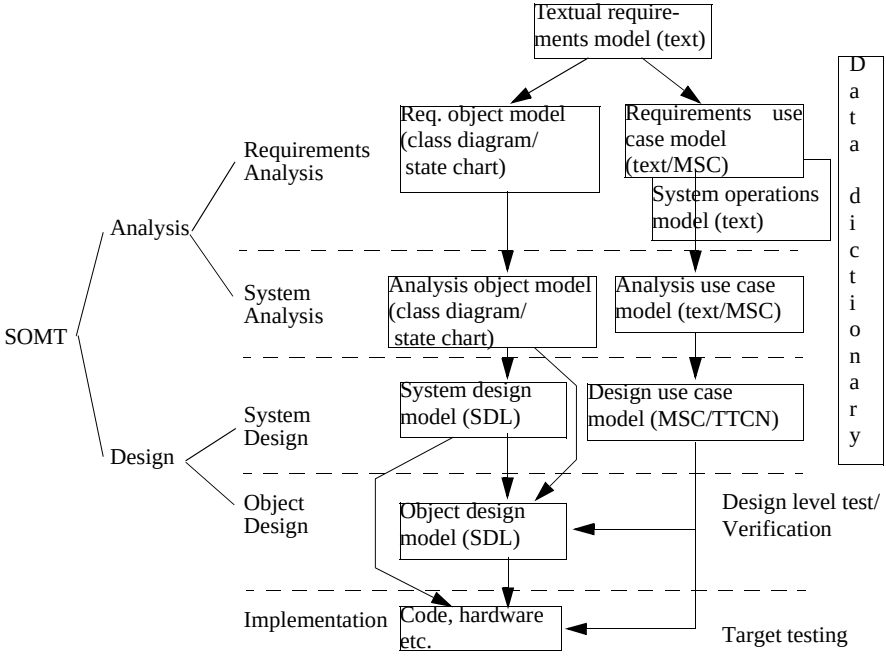

Figure 604: SOMT activities and main models

As can be seen in Figure 604, SOMT consists of five major activities:

- *Requirements analysis.* The purpose of the requirements analysis is to analyze the problem domain and the requirements on the system to be built, essentially by analyzing the system as a black-box in its intended environment.
- *System analysis.* The purpose of the system analysis is to identify the architecture and most important objects that must be represented in the system to achieve the required functionality.
- *System design.* The purpose of the system design phase is to precisely define the architecture of the system including the detailed interfaces between different parts. In the system design the architecture of the system is also analyzed in terms of implementation strategies and decomposition into work packages for different development teams
- *Object design.* The purpose of the object design is to define in detail the functionality including the behavior of all objects.
- *Implementation.* The purpose of this phase is to create the executing application that implements the requirements.

As seen above, SOMT uses a number of different models to describe different aspects of the system. In many cases there are relations between objects in the different models. One especially important relation is that in some cases one object can be seen as an implementation of another object. In SOMT this type of relation is called *implink* (short for implementation link) and corresponds to a design decision taken during the development. The creation and maintenance of implinks forms an important part of the SOMT method. Two points can be made:

- Since the transition between different models is a creative process involving a number of engineering decisions, this is a manual, but tool supported, activity.
- The major tool support is a *Paste As* functionality that will allow you to copy an object in one model and paste it into another model while automatically creating an implink. For example: an object model class from the analysis object model may be pasted as an SDL process type in the system design model.

Note that the *Paste As* functionality can be seen as an implementation of transformation rules between the different models, e.g. from object models to SDL design models.

Another important aspect of implinks is that they facilitate consistency checks between the different models.

It is important to note that even though the description in SOMT of the different activities is a sequential description, this does not mean that in practice these activities can or should be carried out in sequential order. On the contrary, depending on the size and complexity of the application, the different activities can be organized in a number of different ways. This is the topic of the chapter 77, *SOMT Projects*.

Scope of the SOMT Method

The primary focus of the SOMT method is on system development using object oriented analysis and SDL design, with extra support for interface definitions in other notations like ASN.1 [22]. Nevertheless, an important issue in SOMT is the possibility to use the analysis part of SOMT together with design using other notations. Some examples:

- The system to be built is composed of several different parts, where some parts are not suitable to design using SDL. A simple example might be where the system includes a window based user interface that is easier to design using a special purpose tool.
- Another example is when the system includes a data base component. The analysis of the data is suitable to perform using object oriented analysis, but the data base component is preferably designed and implemented using a commercial data base tool.
- Different parts of the system may be designed in different parts of the world and for some reason there has been a choice to use SDL-based development for some parts and other methods for other parts.

The scope of the support given by SOMT when using a design notation other than SDL is, as can be seen in Figure 605, basically:

- The requirements and system analysis of the system are still valid.
- Parts of the system design activity are still supported by SOMT.
- The SDL-based design of components is of course supported.

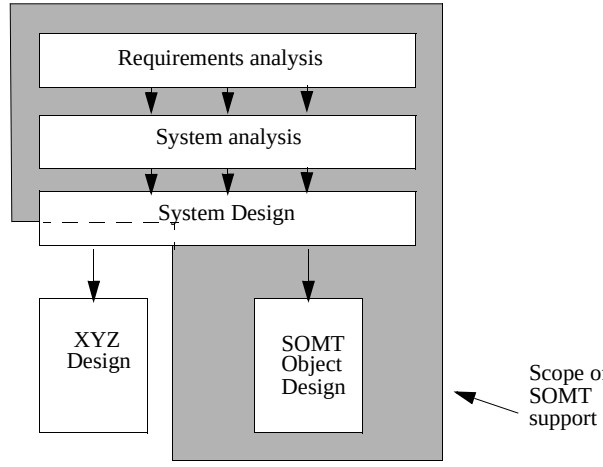

Figure 605: Using SOMT analysis and design together with design using other methods/notations

How much of the system design part that is supported by SOMT is of course very much depending on the “XYZ” method used (see Figure 605). However, the support given by SOMT when using “external” components is at least the definition of the interface of the “external” components. This includes definitions of both static and dynamic aspects as will be seen in chapter 74, *System Design*.

Requirements Analysis

The requirements analysis is the first activity in the SOMT method. The purpose of the activity is, as stated above, to capture and analyze the problem domain and the user requirements on the system to be built. For this purpose the system is viewed as a black-box and only the objects and concepts visible on the system boundary and outside the system are modeled.

There are five major models used in this phase:

- A textual requirements model
- A use case model
- A requirements object model
- A data dictionary

- A system operation model

The textual requirement model is a conventional textual requirements specification. This may come from the customers as input to the development project or it may be created as part of the requirements analysis as a complement to the other models.

The use case model consists of a set of use cases, each described using structured text and/or using MSCs and HMSCs. The purpose of this model is to capture and validate requirements from a user's point of view to make sure that the system will solve the right problem.

The requirements object model is a conventional object model, i.e. one or more diagrams illustrating a number of objects and their relations (including inheritance and aggregation relations). State charts may also be included in the object model to provide high level descriptions of the behavior of important objects. The purpose of the model is two-fold:

- To use context diagrams to describe the system and the external actors that interact with the system
- To document all the concepts found during the requirements analysis and the relations between them in order to ensure that developers and users have a common understanding of the problem domain

The system operation model is a description of the atomic transactions the system must be able to perform. The system operations can be viewed as more detailed specifications of the events that are part of the use cases.

During the requirements phase a data dictionary is also created. The data dictionary is a list of the concepts identified in this phase together with brief explanations of the concepts. It is used and refined throughout all the development phases.

System Analysis

In the system analysis phase the system to be built is analyzed using an object-oriented method. The purpose of the phase is to describe the architecture of the system and identify the objects that are needed to implement the required functionality. The models used in this phase are:

- An analysis object model, to describe the architecture of the system and represent the objects in the system

- An analysis use case model where the interaction between some of the objects is described
- Textual analysis documents to document decisions and architecture related requirements not suitable to be expressed in the object models and use cases

The analysis object model is a conventional object model, consisting of class diagrams and possibly state charts, that forms the input to the object design phase. The analysts should in this model be concerned with identifying the objects that must be present in the system. These objects may come from the requirements object model produced in the requirements analysis, or they may be added in this phase for different reasons. For example, objects may be needed to represent the control logic described by the use cases from the requirements analysis. The differences between the object model in the system analysis phase and the object model in the requirements analysis phase are:

- The requirements analysis object model consists of objects visible on the border of the system and outside the system, e.g. users of the system, while the system analysis object model is focused on the internal object structure of the system.
- The purpose of the object model in the requirements analysis is different from the purpose in the system analysis. The purpose in the requirements analysis is to describe the problem that the system is to solve, while the purpose in the system analysis is to describe the system itself.

System Design

The purpose of the system design is to define the implementation structure of the system and to identify overall design strategies. The models used in this activity are mainly:

- A design module structure, describing the source modules of the design

- An architecture definition, containing
  - SDL system/block diagrams that define the architecture of the resulting application
  - SDL interface definitions of the components of the system using signals or remote procedures
- A design use case model, described by MSC and HMSC diagrams, to define the dynamic interfaces between the different components in the system
- Textual design specifications, describing aspects of the components not suitable for formalization like user interface definitions and other non-functional requirements

The designer of the system will have several different aspects to cover within the system design activity:

- The decomposition of the system into work items, maybe to be implemented by different development teams
- Reuse aspects of the design

Object Design

The purpose of the object design is to create a complete and formally verified description of the behavior of the system. The models used in this phase are essentially SDL diagrams, in particular SDL process graphs that are used to define the behavior of active objects.

One very important aspect of this phase is usage of the *Paste As* concept that will take a user from object models to SDL diagrams. Essentially the designer of the system will in this phase have to consider two major aspects when creating the design model:

- The representation of the analysis object model concepts in the design

  This involves issues like to decide for each object if it is an active or passive object.

- The dynamic behavior of the design objects

The object design also includes a testing/verification task. The purpose of this is to verify that the requirements on the system, expressed e.g. in the design use cases from system design, are correctly implemented in the design.

Implementation

The implementation and testing activities are aimed at producing the final application, i.e. executable software and hardware. The activities in this phase are very much depending on the execution environment of the application but may include:

- Using an automatic code generation tool to produce the code from the SDL design
- Adapting the generated code (the running SDL system) to its environment i.e. regarding signal handling
- Integrating the code to the hardware requirements by means of using real-time operating systems and cross-compilers to generate the executable for the hardware
- Implementing and executing test cases in the target environment based on the design use cases from the system design activity

Summary

SOMT is a method specially designed to suit the development of distributed, reactive, real-time systems. It is based on well-established and industrial-strength techniques:

- object models for analysis
- SDL for design

SOMT recognizes not only a number of models to be made, but also the importance of having a glue between these models. Therefore, SOMT introduces the *Paste As* concept to provide a smooth and well-defined transition between the different models, e.g. between the analysis and design, enabling traceability using *implinks.*

The benefit for a development project is that the analysts and developers all can use the notations best suited for each phase of a development project. The object oriented analysis strength when analyzing requirements and creating conceptual analysis models is combined with the strong back-end given by SDL tools for design, verification and code generation.

References

[17] I. Jacobson, S. Bylund, P. Jonsson, S. Ehnebom, Using contracts and use cases to build plugable architectures, JOOP, May 1995.

[18] I. Jacobson et al, “Object-Oriented Software Engineering”, Addison-Wesley, 1992.

[19] J. Rumbaugh, Getting Started, using use cases to capture requirements, JOOP, September 1994.

[20] J. Rumbaugh et al, “Object-Oriented Modeling and Design”, Prentice Hall, Englewood Cliffs, NJ, 1991.

[21] J. Rumbaugh, OMT: The Development Process, JOOP, May 1995.

[22] ITU Recommendation X.680-683, Abstract Syntax Notation One (ASN.1), 1994.

[23] ITU Recommendation Z.100, Specification and Description Language (SDL), 1994.

[24] ITU Recommendation Z.105, SDL Combined with ASN.1 (SDL/ASN.1), 1995.

[25] ITU Recommendation Z.120, Message Sequence Charts (MSC).

[26] J. Ellsberger, D. Hogrefe, A. Sarma, “SDL – Formal Object-oriented Language for Communicating Systems”, Prentice Hall Europe, 1997, ISBN 0-13-632886-5.

[27] A. Olsen, O. Færgemand, B. Møller-Pedersen, R. Reed, J.R.W. Smith, “Systems Engineering using SDL-92”, North-Holland, 1994, ISBN 0-444-89872-7.

[28] F. Belina, D. Hogrefe, A. Sarma, “SDL with Applications from Protocol Specification”, Carl Hansiger Verlag and Prentice Hall International, 1991, ISBN 0-13-785890-6.

[29] D. Coleman et al, “Object-oriented Development - The Fusion Method”, Prentice-Hall, 1994.

[30] OMG, Common Object Request Broker: Architecture and Specification, Revision 2.0, Object Management Group, July 1995.

[31] Telelogic AB: Telelogic Tau 4.5 User’s Manual, 1999.

[32] G. Booch, “Object Solutions - Managing the Object Oriented Project”, Addison-Wesley, 1996.

[33] ISO Tree and Tabular Combined Notation ISO/IEC 9646-3 1992

[34] Unified Modeling Language, UML Notation Guide version 1.0, (document ad/97-01-09), Object Management Group, January 1997.

[35] David Harel, “Statecharts: a visual formalism for complex systems”, Science of Computer programming (8), 1987.

[36] Unified Modeling Language, UML Notation Guide version 1.1 (document ad/97-08-05), Object Management Group, September 1997.

[37] Unified Modeling Language, UML Semantics version 1.1 (document ad/97-08-04), Object Management Group, September 1997.

Chapter 69 SOMT Introduction