Chapter 1

Introduction

Over the last forty years various formal and semi-formal specification techniques have been proposed supporting different levels, areas, and activities of software design. Each specification technique shows particular advantages and shortcomings often inherited from the underlying formal framework.

The algebraic specification [EM85, EM90] of abstract data types is a well established specification technique. It supports the representation-independent description of classical data structures [LEW96] and more general of software and hardware systems [AKKB99]. The axiomatic approach to functional system description offers a rigorous formalism at a high level of abstraction. The approach hardly scales up to large and complex systems without adequate structuring concepts [BG77].

Object-oriented descriptions [JCJÖ92] promise an intuitive approach to systems modeling. Object-oriented concepts are often used in an informal way, since a precise and tractable semantics is not obvious [BHH+97, HR04]. The semi-formal object-oriented specification with UML [BRJ97, Obj07a] has become a de facto standard in software engineering used for developing large component-based systems [Szy99].

A top-down approach clearly separating the level of functional system modeling with algebraic specification techniques from the level of state-based modeling with object-oriented techniques would benefit from the advantages of both approaches for system development [BR07].

To bridge the gap between functional system modeling and state-based modeling, this thesis studies the refinement of algebraic specifications into object-oriented specifications. Moreover, to put the approach into practice, we mechanize the refinement step in a transformation system [AC96].

In the following, we first describe the contribution, then we overview related work, and finally we outline the structure of this thesis.

1.1 Contributions

In this thesis, we present a tool-supported formal method for refining algebraic specifications into object-oriented specifications illustrated in Figure 1.1. The transformation highlights the conceptual differences between algebraic specifications and object-oriented

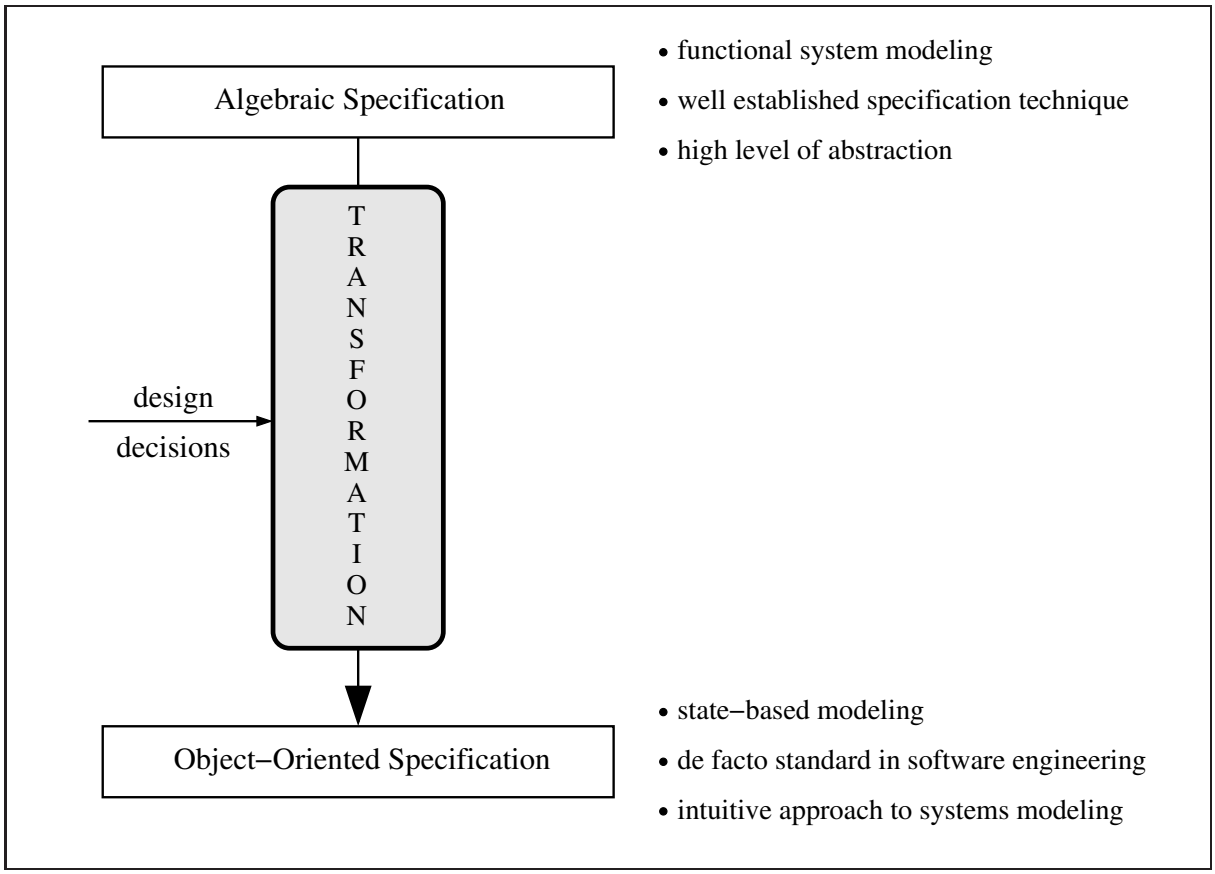

Figure 1.1: Transformation of an algebraic specification to an object-oriented specification

modeling. Particular emphasis is laid on explicating the design decisions underlying this refinement step [BvW98] . The correctness of the refinement step is established by preserving the models of the algebraic specification respecting a natural semantic relation between algebraic and object-oriented specifications.

The transformation provides a safe pathway from functional specifications to object-oriented specifications.

This thesis contributes a variety of results which are of general interest to software engineers. The transformation of algebraic specifications into object-oriented specifications unveils general insights into the concepts of functional and object-oriented modeling techniques. Thus it contributes to a better understanding of object-oriented software construction.

In the functional world, the elements of the carriers are static items possessing a unique value. In the object-oriented world, the objects are activated during system execution. During their lifetime, objects encapsulate a hidden state which results from the history of updates effected by the sequence of methods applied to the object after activation. The transformation of algebraic specifications to object-oriented specifications reflects a clear separation of static versus dynamic concepts, of value-oriented versus state-based concerns, and of interaction versus lifecycle methods.

The transformation unveils an object-oriented counterpart to terms over a functional signature which are structures built by the sequential and interleaving composition of method applications. Thereby, the subterm relation of the term becomes a set of causal depen-

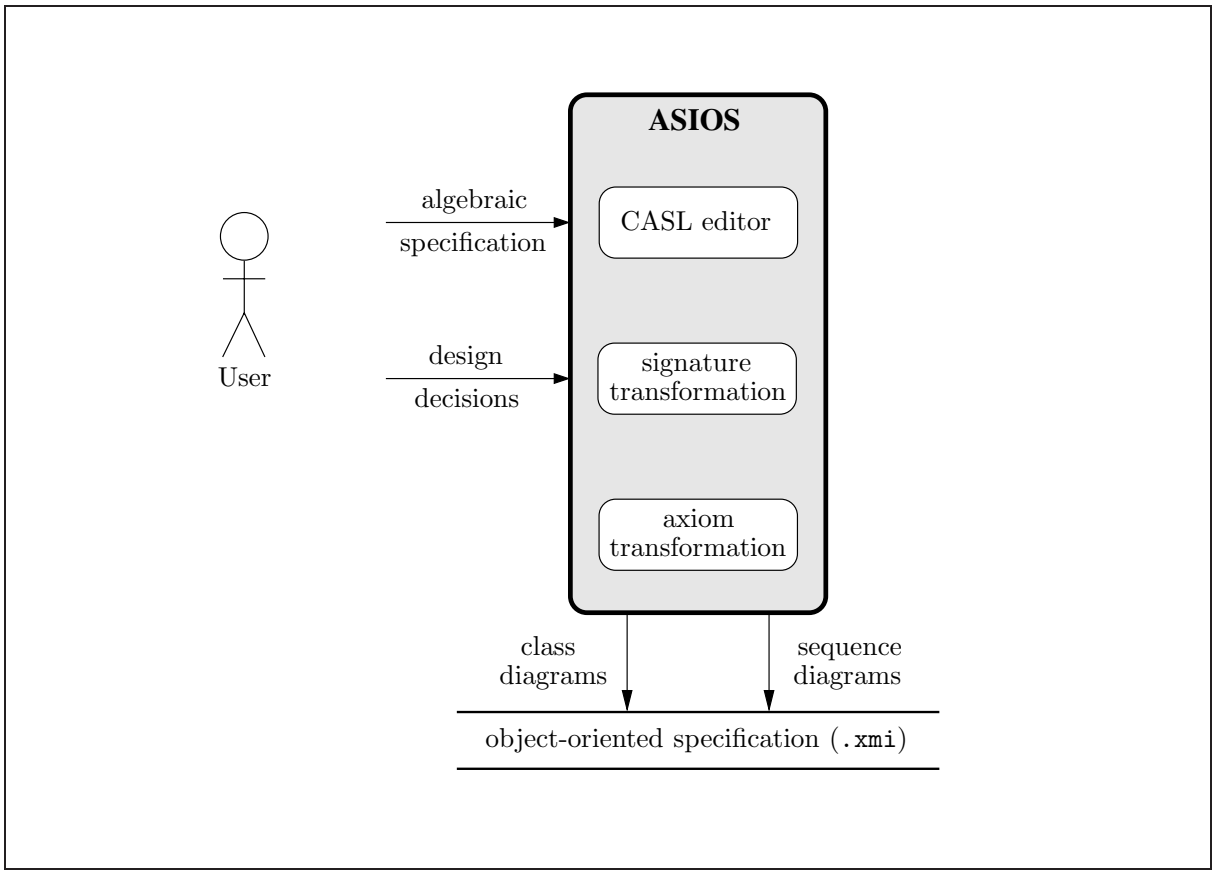

Figure 1.2: Tool ASIOS

dencies between the method applications. Furthermore, the transformation shows that in object-oriented specifications an equation between two terms turns into two structures of method applications and a set of pairs of variables. Each pair specifies that its variables address objects with the same state after each possible evaluation of the two structures of method applications.

The approach bridges the gap between functional system modeling and state-based modeling. This combination of system views is an essential ingredient [BR07] for scientifically founded software development.

Furthermore, the thesis introduces a basic system model for object-oriented systems, since there exists no generally accepted system model. This basic system model serves to define the semantics of object-oriented specifications in a precise way, and thus enables to prove the correctness of the transformation.

The complete transformation has been implemented in the prototype system ASIOS, cf. Figure 1.2. The tool presents the necessary design decisions for the transformation through a graphical user interface. It consists of three subsystems: an editor for entering algebraic specifications in CASL notation [ABK+02], a subsystem for the transformation of the functional signature, and a subsystem realizing the transformation of the axioms of an algebraic specification. The object-oriented specifications resulting from the transformation consist of UML class diagrams and sequence diagrams exported as XMI files [Obj05]. In this way, they can be further refined using UML tools.

1.2 Related Work

A relation between the algebraic and the object-oriented approach can be accomplished in different ways.

Both approaches can be melted into a new specification technique [Aig04, Bre91, Ehr99, GH94] . Alternatively an algebraic specification can be refined into an object-oriented specification. Vice versa, an algebraic specification can be abstracted from an object-oriented specification.

Following the *top-down direction*, algebraic specifications were transformed into implementations in functional programming languages [AS02, FM97, THK88] , in imperative programming languages [Lin93, Lom87] , and object-oriented programming languages [AH00, Pri93] . In contrast to the transformation proposed in this thesis, these approaches make most of the design decisions in a predefined way and result with one possible implementation of an algebraic specification in the considered programming language.

Various tools have been developed supporting the refinement of algebraic specifications into functional programs and imperative programs. For example, the tool LTS [DM01] supports the refinement of higher order algebraic specifications following sound transformation rules ending up with an algorithmic specification that can be automatically compiled into Standard ML code [MTH89] . The tool RefStep [HH95] offers the refinement of algebraic specifications into imperative programs using refinement rules [Mor94] .

Following the *bottom-up direction*, an algebraic specification was extracted from an object-oriented specification [HCRT99] and from an object-oriented implementation [OSI00, HD03] .

1.3 Outline

The thesis starts with an introductory example in Chapter 2. The rest of the thesis is structured into three main parts.

Part I surveys the theoretical foundations of this work. In Chapter 3, we present the syntax and semantics of algebraic specifications. Chapter 4 deals with object-oriented specifications. We define a system model for object-oriented systems, and describe the syntax and semantics of object-oriented specifications. Finally we represent object-oriented specifications with UML .

Part II presents the transformation of algebraic specifications into object-oriented specifications. Chapter 5 transforms the signature of an algebraic specification to a collection of object-oriented class signatures. We explore possible design decisions and formalize them as choice mappings. Chapter 6 converts terms and equations between terms into their object-oriented counterparts. Moreover, it points out important properties of the transformation establishing its correctness.

In Part III, we show a comprehensive case study and how the mechanization of the transformation of algebraic specifications into object-oriented specifications proceeds. Chapter 7 illustrates the transformation using the well-known abstract data type “stacks of natural numbers” . We explore the design space and investigate the transformation for characteristic design choices. Thereby, we graphically represent the resulting object-oriented

specifications with UML. Furthermore, we show object-oriented classes which provide interpretations for them. In Chapter 8, we survey the tool ASIOS which supports the systematic refinement of algebraic specifications into object-oriented specifications. The key features of the system are shown followed by a description of the architecture, the subsystems, and the usage of the tool.

Finally, we conclude the thesis with a summary and a discussion of our results, we sketch an important area of application of the transformation, and we identify prospects of future work.

The appendix of the thesis consists of three chapters. Appendix A introduces the mathematical terms and notations which are used in this thesis. In particular, it explains notations on sets, (partial) mappings, sequences, and indexed families. Appendix B contains an implementation of each of the object-oriented classes described in this thesis using the object-oriented programming language Java. All proofs are collected at the end of the thesis in Appendix C.