

Towards an Application of FDT within OOAD

Dr. Eckhardt Holz
Humboldt-Universität zu Berlin, Institut für Informatik
A.-Springer-Str. 54a, 10117 Berlin
holz@informatik.hu-berlin.de

Formal Description Techniques and techniques of Object Oriented Analysis and Design are two major means for the development of complex, distributed and reliable software. Nevertheless, both techniques address the problem from a different baseline. The main focus of FDT's is to ensure the correctness of the final system by the provision of extensive validation and verification methods. OOAD techniques on the other side focus on an efficient development process with smooth transitions between the single stages. The paper investigates approaches to combine both techniques to increase the mutual benefits and to limit the shortcomings of them.

1. Preface

Within the last years object oriented analysis and design techniques (OOAD) have gained a widespread application within the area of design and development of complex and distributed software. On one side this is due to their intelligible and intuitive graphical notation and the wide availability of a range of supporting tools. On the other side object oriented paradigms have been accepted as the main means for the description and implementation of distributed systems. This is reflected as well by the languages applied (C++, Java, Smalltalk etc.) as by the infrastructures, which enable and support the distribution (OMG-CORBA, TINA-DPE).

One of the main shortcomings of OOAD techniques however is their weak and informal semantically foundation. Without accompanying explaining text the models may be ambiguous and the application of formal validation and verification techniques is severely restricted. Tools for OOAD limit themselves to a syntactical analysis (including verification of signatures) and the tracing of design informations over different stages of the development process or between different submodels of the overall model. The new trends towards a unification of the different techniques (UML, [2]) do only partially address this problem. The UML Semantics document does give a formalized definition of the structure of a UML model (i.e. graphical syntax), however, the static and dynamic semantics are once again only informally defined.

Formal description techniques (FDT) on the other side are characterized by their well defined formal syntax and semantics. As major techniques SDL, LOTOS, Estelle and Z have found industrial application due to the availability of methodologies and supporting tools for a verification and validation basing on the formal semantics. On hindrance for a much broader application of these techniques are the requirements which they set upon the user (strict syntax, high degree of abstraction, limited set of basic concepts). A direct reflection of object oriented concepts with FDT's requires often a substantial overhead. This does also make the transition from specification over verification/validation to the implementation less seamless as it is for the OOAD techniques.

This paper discusses different approaches for a combination of object oriented paradigms resp. of OOAD techniques with FDT's. Among these approaches are the direct introduction of object oriented concepts in the FDT, a translation approach between OOAD and FDT and the application of FDT to formalize submodels of OOAD.

2. Introduction of Object Oriented Concepts into FDT

To cope with the new requirements of the software design SDL has been extended within its last major language revision with object oriented concepts [1]. Important new features are the introduction of a type concept (corresponds to classes) and the possibility to build generalization/specialization hierarchies (inheritance). To be compatible with previous versions of SDL the semantics of these new extensions to the language are introduced by a mapping to existing concepts. This implies that the object oriented concepts are merely a syntactical extension, within the semantics definition they are not visible any more (cf. Fig. Figure 1). A direct verification of properties of these concepts is therefore also not possible, the verification will be

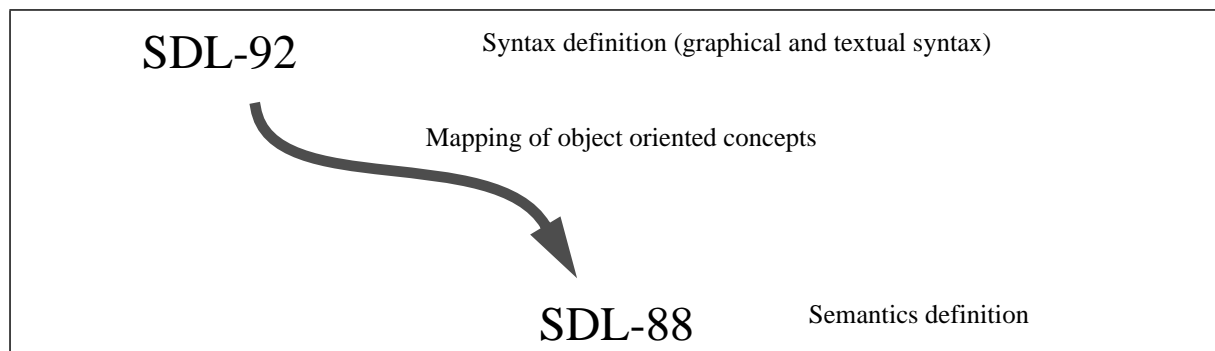


Figure 1 Definition of object oriented concepts in SDL

always based on the transformed specification and the results have to be traced back to the original specification. However, because SDL does not provide the full range of dynamic object oriented concepts (e.g. polymorphism, type based instantiation), the restrictions to the verification are limited. Additionally tools do support the direct generation of object oriented code (C++) from a object oriented SDL specification. This code can be used for a simulation or as an implementation prototype.

3. Combination of OOAD and FDT

Within different projects proposals for a combination of the OOAD technology OMT [3] and the FDT SDL have been made. Two different directions can be identified here:

- Translation of OMT to SDL ([8], [10])
- Link between SDL and OMT models [9].

The translation approach uses OMT and SDL at different stages within the development process: OMT is applied for the analysis and early system design, SDL is applied for the detailed design and as a high-level implementation language (cf. Fig. Figure 2). A translational semantics for OMT is defined by giving rules to map the different concepts of OMT to SDL concepts. Such a mapping can be supported by tools. The main base for the translation are the OMT class model and the behaviour model, the functional model is not considered. The resulting SDL specification consists of a configuration of blocks and processes (according to the OMT class structure) and the behaviour descriptions for the processes (process graphs) derived

from the OMT state machines. The translation approach does restrict the flexibility of OMT, because not each OMT description can be translated to an SDL specification. These restrictions concern the structure (e.g. no recursive aggregation), the semantics of OMT constructs (associations) and the behaviour part

The resulting SDL specification, which will be more or less complete - depending on the OMT description, can now be analysed with the tools and methodologies available for SDL. Finally the verified and completed specification serves as a starting point for the code generation. Unfortunately, the code generated from such an SDL specification bears less object orientation as code generated directly from a complete OMT description.

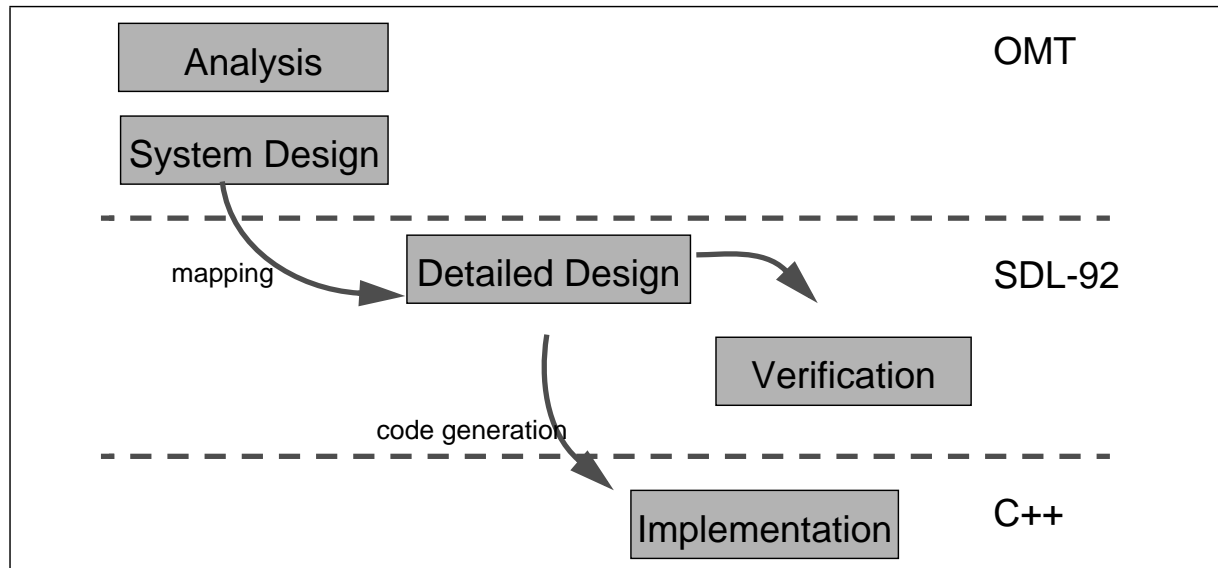


Figure 2 Translation approach

The second approach uses both techniques at the same level of the development process. The application of OMT is restricted to the specification of data types, whereas the overall system structure and behaviour is specified in SDL (cf. Fig. Figure 3). The connection between both specifications is made by so-called links. This enables the use of object oriented data types (defined in OMT) within a SDL specification. A formal definition of the semantics underlying the links is not given. The main idea behind this approach is the provision of an

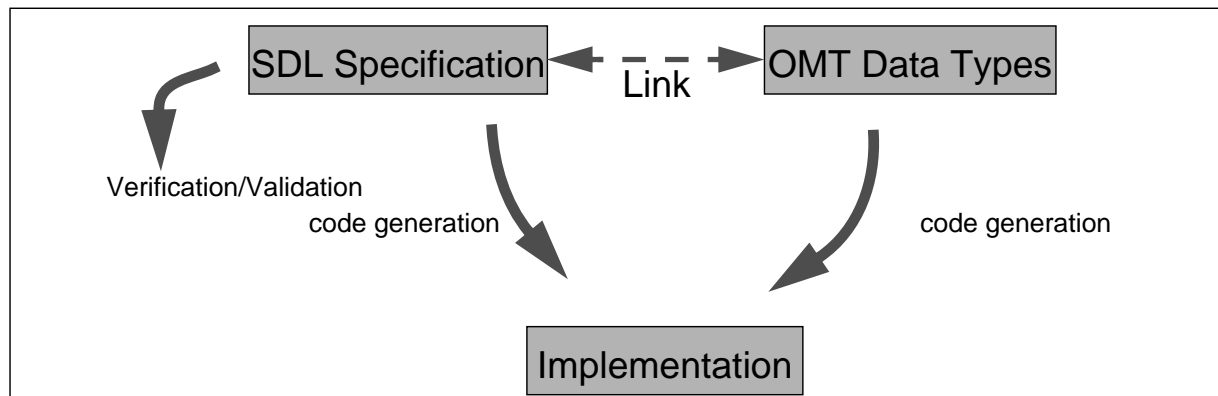


Figure 3 Link approach

external data type concept for SDL. The code generation takes both sub-specifications and generates code for them which can be then linked together.

One important fact which can be observed for both approaches is, the combination between OMT and SDL is not in first case made to enable a formal verification. The main focus is rather on a smooth transition from the specification to an implementation.

4. Formalizing OOAD?

The different ways presented in Sect. 2 and 3 to combine object orientation and formal description techniques do both try to give the FDT new concepts and features. On the other side, the of OOAD users outnumber by large the users of FDT's. The question should be therefore, how the advantages of a formal verification and validation can be made available to OOAD users without having them to give up their notation. In order to do so, first it has to be identified where and when a formal verification/validation is necessary within the scope of OOAD and where informality is sufficient enough.

Clearly two different areas for the application of FDT within OOAD can be identified:

- Support for a conformance test between models at different stages of the development process
- Verification/Validation of submodels and the overall model within a single stage.

A second fact, which can be observed is, that the degree of informality decreases from the analysis to the detailed design. However, even at the final stages OOAD models are usually not complete models, i.e. descriptions of trivial structures and behaviour are often omitted or directly given in the envisaged implementation language.

Transition between development stages

Due to the informality of OOAD and the variety of concepts used at different stages to describe the structure and the behaviour of the envisaged system, the practical area of formal conformance tests between stages is restricted to the later stages. What instead is required is a support to trace design decisions through the different stages of the development process from analysis downwards to the detailed design (and back). Such information can be stored in annotations to single model elements or by relations describing the links between model elements of different stages.

Verification/Validation of submodels

The first task here is a structural/syntactical analysis. This is usually already available for the tools for OOAD and comprises:

- check of the usage of model elements with respect to their definition,
- visibility and access.

What is missing here is a check of the dynamics of the model. Due to the possibility to describe the behaviour from different viewpoints and with different means (state-machines, sequence charts, collaborations and use cases) the check of the dynamics has to investigate two tasks:

- Does a submodel reflect the required behaviour?
- Do the different submodels of the same system component contradict?

Especially for the first task the FDT do provide the necessary tools and methodologies. Using extended finite state machines for the behavioural description in the same way as FDT's like SDL or Estelle allows to use the same tools and techniques which are applied for those languages. One necessary prerequisite is only to provide a mapping between the OOAD state machines to the FDT state machines (or to extend the FDT tools to accept Harel-state charts). The same holds also for the analysis of the sequence charts, which do have less or the same expressive power as the standardized Message Sequence Charts (MSC). The rules for the mapping from OOAD to the FDT are similar to the rules of the translation approach (cf. Sect. 3), however, additionally rules for the re-interpretation of the analysis results back to the OOAD are needed. The third kind of behaviour descriptions are collaborations. They are snapshots of the observable behaviour of a group of objects, concentrating on the order of operations. Here techniques known from LOTOS are a prime candidate for the analysis.

The second task is more difficult, because it requires to make statements between different semantic views of the same component. As state-machine models as well as collaboration models both have a strong relation to the sequence charts, the test whether the submodels contradict should be made sequence chart centred. This may also lead to the derivation of new sequence charts or the extension of existing ones.

5. Summary and Outlook

Formal description techniques and object oriented analysis and design techniques are two important instruments for the development of complex and distributed software systems. Whereas FDT's have been developed for the specification of unambiguous system descriptions and for the verification and validation, the focus of OOAD techniques is an efficient design process from early analysis stages down to the implementation. The question should be therefore not whether to use either FDT or OOAD, but how to combine the advantages of both techniques. Even more in the advent of new and upcoming description techniques (e.g.OMG-IDL, TINA-ODL) and with the need to specify systems from different viewpoints (ODP, [4]), ways to combine different techniques and to make statements about the relations between specifications of different viewpoints become increasingly important.

6. References

- 1 ITU-T: "Z1.100 - ITU-T Specification and Description Language SDL"; ITU-T, 1993.
- 2 Booch, G.; Jacobson, I.; Rumbaugh, J.:" The UML Notation Manual V.1.0", Rational,1997.
- 3 Rumbaugh, J.; Blaha, M.: "Object Oriented Modelling and Design", Prentice Hall, Englewood Cliffs, 1991.
- 4 ITU-T:"X.900 - The Basic Reference Model of Open Distributed Processing (RM-ODP)", ITU-T/ISO, 1996.
- 5 ISO:"LOTOS A formal description technique based on temporal ordering of observational behaviour", ISO, 1989.
- 6 OMG: "The Common Object Request Broker Architecture CORBA 2.0", OMG, 1995
- 7 TINA-C: "Guidelines into TINA", TINA-C, Red Bank, 1995.
- 8 Holz, E.; Wasowski, M.; Witaszek et.al.: "INSYDE Methodology", ESPRIT Project 8641, 1996.
- 9 Telelogic : "The SDT User Manual", Telelogic, 1996.
- 10 Gou, F.; MacKenzie, T.: "Translation of OMT to SDL-92", Proc. of SDL'95, Oslo, 1995.