Combining Troll with the Object Modeling Technique[‡]

Ralf Jungclaus[¶]
Roel J. Wieringa[§]
Peter Hartel[*]
Gunter Saake[*]
Thorsten Hartmann[*]

Abstract

The focus of this paper is the development of a formally based object-oriented modeling formalism called omtroll by using features ¿from mostly informal object-oriented modeling approaches (mainly OMT) and from a formal object-oriented specification approach (Troll). The goals of our approach are to improve popular informal modeling techniques by giving formal semantics to modeling constructs and to improve the applicability of formally-based specification approaches. Based on a brief analysis of OMT against Troll we will present omtroll using examples.

1 Introduction

In the past few years, there has been considerable activity in the area of object-oriented (OO) analysis and of OO formal specification. OO analysis methods like OMT [RBP+91], life cycle analysis [SM92], and Fusion [CAB+94] have been proposed mainly with a background in software development practice. On the other hand, formal OO specification languages like Oblog [CSS89], FOOPS [GM87], Troll [JSHS91], LCM [FW93], TROLL*light* [CGH92], object-oriented extensions to Z [SBC92] are being developed mainly with a background in mathematical theory. These formal languages have their roots in logic, algebra, set theory or category theory and provide a rigorous framework within which to specify OO systems. The OO analysis methods have a background in system development practice and do not reach the level of formality achieved by formal specification languages. Formal languages are based on academic research and are only now beginning to be applied to a few practical software development projects. This suggests that a marriage between the two approaches to modeling and development could be fruitful.

There are a number of advantages to be gained by such a marriage:

[§]Faculty of Mathematics and Computer Science, Free University, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands. Email: roelw@cs.vu.nl.

[*]Abt. Datenbanken, TU Braunschweig, Postfach 3329, D–38023 Braunschweig, Germany. Email: {hartel|hartmann|saake}@idb.cs.tu-bs.de

[¶]Deutsche Telekom, TD42a, Postfach 2000, D–53105 Bonn, Germany. Email: jungclau@u9000mst.nez.telekom.de

[‡]Partially supported by ESPRIT BRA WG 6071 IS-CORE, DFG under Sa 465/1-3, and OBLOG Software S.A., Lisboa.

in: B. Wolfinger (ed.). Innovation bei Rechen- und Kommunikationssystemen. Springer-Verlag, p.35-42, 1994

- It could help in discovering and eliminating ambiguity and vagueness in the informal representation techniques of OO analysis methods.
- Combining a practical analysis method with a formal specification language helps in making the formal specification language fit for use.
- Formal specification languages usually come with logical inference systems, and combining this with a practical analysis method makes powerful techniques like automated theorem proving and animation available for CASE tools used in analysis.

In this paper, we make an attempt to use concepts of the Object Modeling Technique (OMT) [RBP+91] (and some things borrowed from [SM92]) to develop specifications of object systems in the TROLL language [JSHS91, HSJ+94]. We concentrate mainly on the analysis phase in the system development process. Analysis is also called *conceptual modeling* in this paper since it results in a *conceptual model* of the system to be implemented.

The achievements of the approach presented here are

- The TROLL specification is used for the sound integration of different models (or better *views*) found in OMT and similar approaches.
- The notations of OMT are analyzed and improved on the basis of TROLL concepts.
- Concepts and specification issues of OMT are made more precise.
- A graphical notation that helps in developing system specifications is presented.

The point is that graphical notations will be used to make TROLL specifications more manageable. Significant portions of the TROLL language will remain present even in OMTROLL specifications, since there are properties that cannot be specified graphically in a suitable manner. Among these properties are specifications of effects of event occurrences on the state of objects, (temporal) constraints, and global rules. OMT has been chosen because of its popularity and the familiar notations that are used.

The paper is structured as follows: In the next section, we will give a brief introduction to the TROLL language and OMT and will point out some problems revealed by formal analysis of the techniques used in the conceptual modeling stage of OMT. The improved version of OMT is then illustrated in Section 3 by means of an example followed by a sketch of the translation of OMTROLL constructs to TROLL. Section 5 winds up the paper with conclusions and topics for further research.

2 TROLL versus OMT

2.1 TROLL

TROLL is a formal language for the specification of object systems. The approach puts emphasis on the modeling of dynamic behavior over time of objects. Objects can be structured along the lines of semantic data models.

An object is considered to evolve in a discrete manner by the occurrence of events. Objects come into existence by the occurrence of birth events and may cease to exist by the occurrence of a death event. Attributes are used to represent observations about the current state of an object. Objects may interact by synchronizing on events with the possibility of exchanging informations while interacting (similar to process models). Objects are related in various ways. A role describes a temporal specialization of an object having additional properties and/or restricted behavior. A specialization is considered to be a permanent role. Objects may be components of other objects which itself are called composite objects. In TROLL, the composition of composite objects may change dynamically. Separately defined objects may be connected through global interactions.

Over objects, we may define views that contain only derived information over the current states of objects. The current version of Troll is described in detail in [HSJ+94]. Examples of Troll specifications can be found also in Section 4.

The central idea in the formalization of Troll is the interpretation of an object as an observable sequential process [SSE87]. A life cycle of an object is a sequence of event occurrences in this object. All possible object life cycles define a process which is the formal description of object behavior. An object state depends on the complete object history, i.e. its actual life cycle. The state can be calculated as a mapping from a sequence of event occurrences to a sequence of attribute observations.

[Jun93] gives a formal definition of Troll based on the *Object Specification Logic* (OSL) [SSC92]. This logic allows the reasoning over properties of Troll specifications.

2.2 OMT

An OMT model of a system consists of an object model, a dynamic model, and a functional model. We neither want to present here OMT nor discuss its limitations and drawbacks in detail. The reader intereseted in OMT is referred to [RBP+91].

The *object model* represents the structure of the objects in the system. The diagram technique used is a variation and extension of ER representation techniques. The *dynamic model* represents aspects of the system concerned with control, including time, sequencing of operations, and interaction of objects. For each class whose instances have interesting behavior, a state chart [Har87] is made that describes all or part of the behavior of one object of a given class. OMT supports several mechanisms to model interactions between objects. However, OMT does only know asynchronous communication. The *functional model* defines the meaning of operations by showing how values are transformed by the system. The representation technique used is the data flow diagram (DFD).

The most important observation is that the three OMT models are poorly integrated. The most severe points to support this argument are:

- The definition of operations is spread over all three models.
- Constraints can be specified in the object model and in the functional model.
- Guards in the Dynamic Model may refer to properties defined in other models.
- The relationship between events, actions, activities (as elements of the Dynamic Model) and operations (as elements of the Object Model) is not well defined.
- Communications/Interactions can only be described in the Functional Model. For Dynamic Models, only a weak syntactic technique is given.

3 The omTroll Notation

The goals in developing the omTroll notation can be summarized as follows:

- Due to the popularity of OMT we want to save as much as possible of the OMT notation. However, clarity must be paid by changes in the notation.
- Where necessary, introduce improved notations, make prunings in OMT, or introduce new notations.
- Use OMT notations only where possible, i.e. graphical notations are regarded to be auxiliary notations to make Troll specifications better manageable. This implies that omTroll specifications usually contain textual constraints written in Troll.

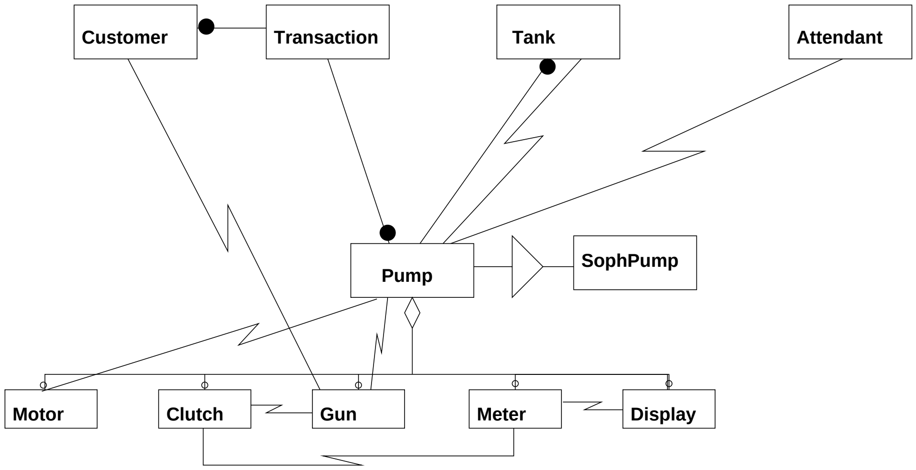

Figure 1: Object Model

3.1 The Example

We are using the gas station example which was used in [CAB+94]. A brief informal description is the following:

A computer-based system is required to control the dispensing of gas, to handle customer payment, and to monitor tank levels. Before a customer can use the self-service pumps, the pump must be enabled by the attendant. When a pump is enabled, the pump motor is started, if it is not already on, with the pump clutch free. When the trigger in the gun is depressed the clutch is engaged and gas is pumped. When it is released, the clutch is freed. The holster in which the gun is kept prevents gas being pumped until the gun is taken out. Once the gun is replaced in the holster, the delivery is deemed to be completed and the pump disabled. Further depressions of the trigger in the gun cannot dispense more gas. Displays on the pump show the amount dispensed and the cost so far. There are two kinds of pump. The normal kind allows the user to dispense gas ad lib. The sophisticated pumps allow the customer to preset either an amount or a volume of gas. Transactions are stored until the customer pays. At present, two grades of gas, stored in different tanks are dispensed.

3.2 Object Modeling

The OMTROLL notation for the object model does not change very much from the OMT notation. A feature that we added is the *interaction association* which is noted as a flash between the interacting object classes.

Figure 1 shows the overall view of the object model for the above example. Object classes are depicted by boxes, we have like in OMT associations (lines between boxes), composite classes (like the class Pump) and specialization classes to define taxonomic relationships between object classes (like SophPump). Association can have cardinalities – here, a black dot denotes that there is exactly one link. An extension is the fact that TROLL supports roles. In TROLL roles are the general case of taxonomic relationships – thus, static specializations are a derived concept and are depicted as dashed lines.

Each object class definition is detailed according to the OMT notation. A class definition defines the signature for the attributes and the events (or operations as they are called in OMT). The Figure 2 gives an example for the declarations of two object classes in the example object model.

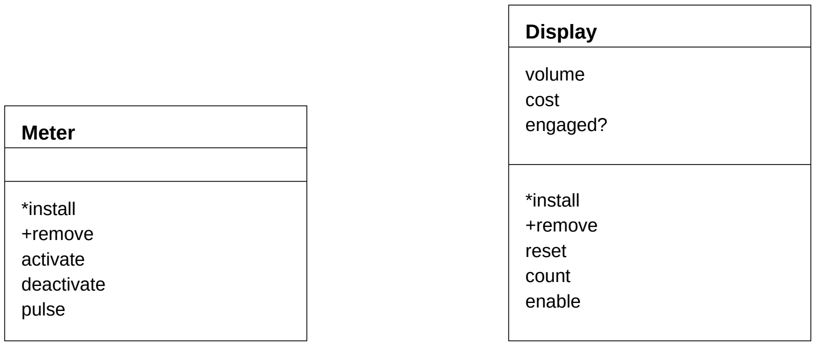

Figure 2: Meter Class, Display Class

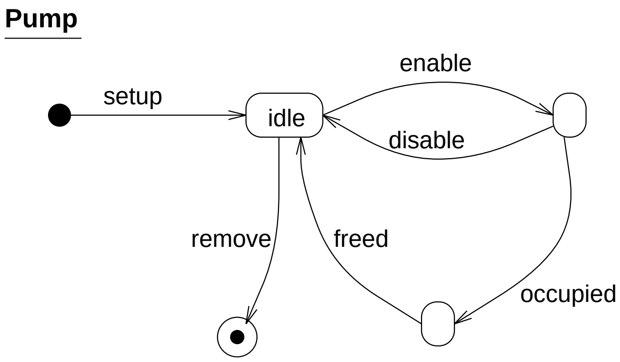

Figure 3: Pump Dynamic Model

3.3 Dynamic Modeling

In the dynamic model, the behavior over time of objects is defined. In OMT, state diagrams are used for this purpose (Figure 3). TROLL does not support the explicit definition of states. Thus, in OMTROLL states are used for auxiliary purposes – of more importance are the state transitions in OMTROLL. The states in the state diagram constrain the possible next transitions. A dot depicts an initial state, a circled dot depicts the postmortem state, i.e. it means that the object has died. In OMTROLL we may furthermore define additional enabling conditions for events (i.e. possible transitions). They are noted in textual form according to the TROLL-syntax.

3.4 Communications Modeling

Since OMT does not support the explicit specification of interactions between objects, we have introduced a new type of diagram in OMTROLL. We call those diagrams *communications diagrams* (Figure 4). The boxes representing object classes are equipped with boxes representing events. These event boxes are connected by arrows. There may be more than one arrow origination in an event box. Preconditions may be attached to connections.

A special case are communications inside composite objects. In this case, the components are included in a box that also includes the object classes specification of the local properties of the composite objects. Figure 5 shows an example.

4 Translation to TROLL

OMTROLL specifications can be translated to TROLL. By this translation we are giving a rigorous semantics to OMTROLL specifications. In many cases, the translation is straightfor-

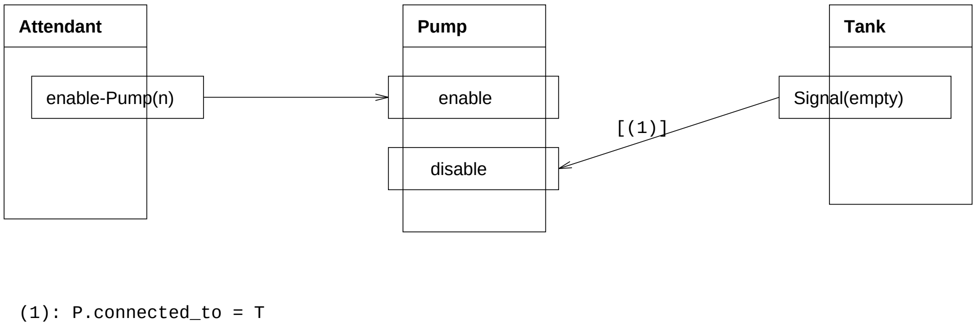

Figure 4: Communications with the Pump

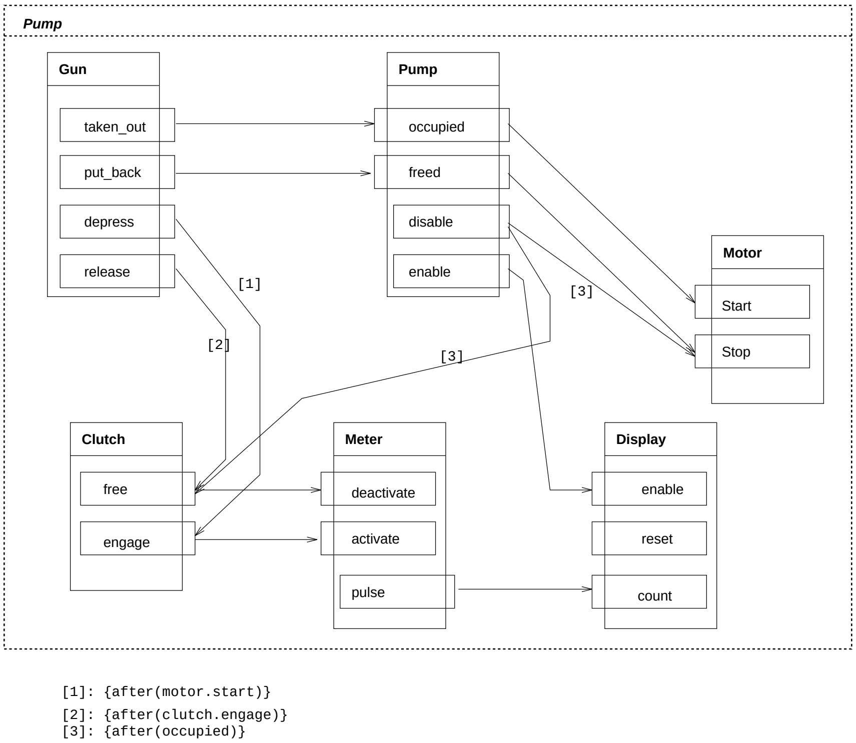

Figure 5: Communication inside Composite Object Pump

ward. Class specifications in OMTROLL are directly translated into skeletons of TROLL class specifications. Enabling conditions for events can be derived from the dynamic model.

The following TROLL-specification fragments are the result of the translation of the OMTROLL object class specifications and the OMTROLL dynamic models along with the integration of the textual parts.

```
object class Display
  attributes
    Volume:real  restricted Volume > 0.0 and Volume < 1000.0  initialized 0.
    Cost:money.
    engaged?:bool  initialized false.
  events
    install  birth.
    remove  death enabled after(install) or after(reset).
    reset  enabled after(enable) or after(count)  changing engaged := false.
    enable  enabled after(install) or after(reset)
      changing engaged := true; Volume := 0; Cost := 0.
    ...
```

```
end object class Display
```

Communication diagrams can result in two types of TROLL specifications. The first type are relationship specifications for interactions between otherwise unrelated objects. The following specifications is part of the translation of the diagram of Figure 4.

```
relationship Att-Control between Attendant A, Pump
   interaction
      A.enable_Pump(n) >> Pump(n).enable;
end relationship Att-Control
```

Finally, communications diagrams specify the interactions inside composite objects. The translation of the specifications of aspects of `Pump` yields the following specification of `Pump`:

```
object class Pump
   attributes
      Number:nat constant.
   events
      enable  enabled after(set_up) or after(freed) or after(disabled).
      disable.
      occupied  enabled after(enable).
      freed  enabled sometime(after(occupied)) since last after(enable).
      ...
   components
      motor:Motor.
      clutch:Clutch.
      gun:Gun.
      ...
    interaction
      gun.taken_out >> occupied;
      occupied >> motor.start;
      {after(motor.start)} gun.depress >> clutch.engage;
      ...
end object class Pump
```

5 Conclusions and Further Work

We simplified OMT graphical techniques in a number of respects:

- The functional model is dropped, because it is imprecise and has a different philosophy than the other two models. The effect of events on objects is specified declaratively in TROLL.
- We added interaction diagrams, which make the communication structure of the model explicit and allows one to specify multicasts as well as preconditions for communications.
- We dropped the representation of propagation of operations from the object model, and we dropped the representation of event calling from the dynamic model.
- Activities have been dropped, since they violate a basic atomicity principle for system transactions. They can be replaced by statecharts that consist of atomic events only.
- We added the concept of a role, which is a process that an object can perform only when it is in a certain state.

We did some minor adaptations of OMT techniques. We dropped qualification and ordering of associations. Guards were separated from tests and we do not distinguish between actions and events.

Returning to our starting point, this paper illustrates that combining a formal specification technique with a practical analysis method leads to an improvement of both. We have been able to simplify OMT techniques while at the same time adding expressive power, such as the ability to specify roles and to represent multicasts explicitly. We think the increased simplicity of the method makes it easier to use. In the future, we plan to look at other analysis methods, such as life cycle analysis [SM92]. We also plan to formalize the translation process from OMTROLL specification to TROLL. A recent work of one of our students describes the semantics of OMT using OSL [Fil94]. This work will help us in the verification of our integration of TROLL and OMT.

References

[CAB+94] D. Coleman, P. Arnold, S. Bodoff, S. Dollin, H. Gilchrist, F. Hayes, and P. Jeremes. *Object-oriented Development - The Fusion Method.* Prentice-Hall, 1994.

[CGH92] S. Conrad, M. Gogolla, and R. Herzig. TROLL *light*: A Core Language for Specifying Objects. Informatik-Bericht 92–02, TU Braunschweig, 1992.

[CSS89] J.F. Costa, A. Sernadas, and C. Sernadas. *OBL-89 User's Manual, version 2.3.* Instituto Superior Técnico, Lisbon, May 1989.

[Fil94] Juliana Küster Filipe. *Describing Semantics of OMT using OSL.* Diploma Thesis, TU Braunschweig, 1994.

[FW93] R.B. Feenstra and R.J. Wieringa. LCM 3.0: a language for describing conceptual models. Technical Report IR-344, Faculty of Mathematics and Computer Science, Vrije Universiteit, Amsterdam, December 1993.

[GM87] J. A. Goguen and J. Meseguer. Unifying Functional, Object-Oriented and Relational Programming with Logical Semantics. In B. Shriver and P. Wegner, editors, *Research Directions in Object-Oriented Programming*, pages 417–477. MIT Press, 1987.

[Har87] D. Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[HSJ+94] T. Hartmann, G. Saake, R. Jungclaus, P. Hartel, and J. Kusch. Revised Version of the Modelling Language TROLL (Version 2.0). Informatik-Bericht 94–03, Technische Universität Braunschweig, 1994.

[JSHS91] R. Jungclaus, G. Saake, T. Hartmann, and C. Sernadas. Object-Oriented Specification of Information Systems: The TROLL Language. Informatik-Bericht 91-04, TU Braunschweig, 1991.

[Jun93] R. Jungclaus. *Modeling of Dynamic Object Systems—A Logic-Based Approach.* Advanced Studies in Computer Science. Vieweg Verlag, Braunschweig/Wiesbaden, 1993.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall, Englewood Cliffs, NJ, 1991.

[SBC92] S. Stepney, R. Barden, and D. Cooper, editors. *Object Orientation in Z.* Springer, 1992.

[SM92] S. Shlaer and S. J. Mellor. *Object Lifecycles: Modeling the World in States.* Yourdon Press/Prentice Hall, Englewood Cliffs, NJ, 1992.

[SSC92] A. Sernadas, C. Sernadas, and J. F. Costa. Object Specification Logic. Research report, INESC/DMIST, Lisbon (P), 1992. *To appear in Journal of Logic and Computation.*

[SSE87] A. Sernadas, C. Sernadas, and H.-D. Ehrich. Object-Oriented Specification of Databases: An Algebraic Approach. In P.M. Stoecker and W. Kent, editors, *Proc.*

13th Int. Conf. on Very Large Databases VLDB'87, pages 107–116. VLDB Endowment Press, Saratoga (CA), 1987.