A Web-based Animator for Object Specifications in a Persistent Environment

Mark Richters and Martin Gogolla

Universität Bremen
FB 3 Mathematik und Informatik
Arbeitsgruppe Datenbanksysteme
Postfach 330 440
D-28334 Bremen
e-mail: {mr|gogolla}@informatik.uni-bremen.de

Abstract. We present an animation tool for the formal specification language TROLL *light*. The system allows the manipulation and querying of objects and navigation through object hierarchies. A Web-based user interface simplifies the usage of the system.

1 Introduction

Formal approaches are important for the development of correct and reliable software systems. But, formal methods often are difficult to understand and use for end users. In the following, we present an animation tool for the object-oriented specification language TROLL *light*. The animation system allows the validation of conceptual models while ease of use is guaranteed by a Web-based user interface.

2 Object Specification with TROLL *light*

The language TROLL *light* is employed for describing static and dynamic properties of objects. We achieve this by offering language features to specify object structure as well as object behavior [2]. Following the object paradigm has the advantage that all relevant information concerning one object can be found within one single unit and is not distributed over a variety of locations. Object descriptions are called templates in TROLL *light* and show the following general structure.

```
TEMPLATE name of the template
  DATA TYPES   data types used in current template
  TEMPLATES    other templates used in current template
  SUBOBJECTS   slots for sub-objects
  ATTRIBUTES   slots for attributes
  EVENTS       event generators
  CONSTRAINTS  restricting conditions on object states
  VALUATION    effect of event occurrences on attributes
  DERIVATION   rules for derived attributes
  INTERACTION  synchronization of events in different objects
  BEHAVIOR     description of object behavior by a CSP-like process
END TEMPLATE
```

Speaking in rough terms, the `DATA TYPES` and `TEMPLATES` sections are the interfaces to other templates, the `SUBOBJECTS`, `ATTRIBUTES`, and `EVENTS` sections constitute the template signature, and in the remaining sections axioms concerning static (`CONSTRAINTS` and `DERIVATION`) and dynamic (`VALUATION`, `INTERACTION`, and `BEHAVIOR`) properties are specified. For more details we have to refer to [2]. As an example, Fig. 1 shows an author template in the left window whereas the right window displays properties of an author object. Both documents are generated by the animation tool.

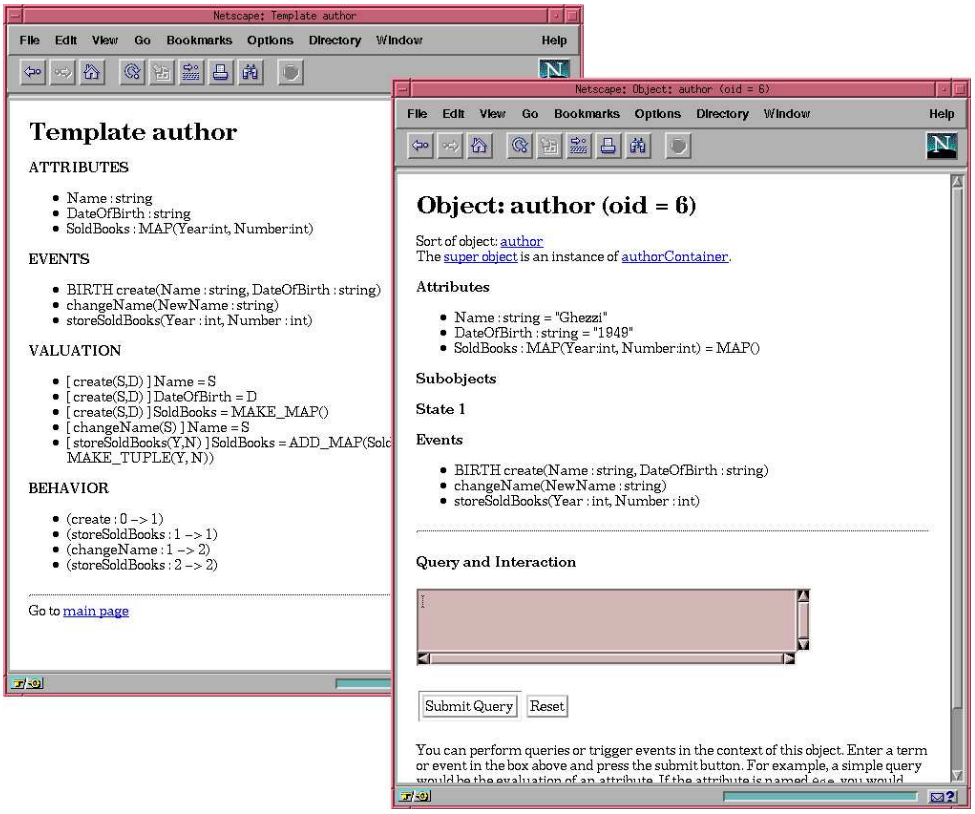

Fig. 1. A template for author objects and a concrete state of an author object.

3 Animation of Object Specifications

As part of the specification process in the database and information systems fields, usually a conceptual model is developed. Due to the fact that not all requirements can be formalized, we also need to validate the model against *informal* user requirements.

This can be achieved by testing the specification in an animation system. In the following, we describe the general architecture of our animation tool.

An animation system allows instantiation and manipulation of objects in a user controlled session. Though animation does not ensure the correctness of a specification, it provides a way to get a first impression of the designed model, and to eliminate obvious errors or design flaws. The main tasks of our animation system are: (1) the exploration of actual states of objects, (2) the specification of event occurrences for initiating state transitions, and (3) the visualization of state changes [3]. In contrast to the system proposed in [3], our animation tool is completely designed for and implemented in the persistent environment Tycoon. We also decided to switch to a Web-based user interface.

The animation system itself consists of the following components:

- persistent representations of object specifications, object states and complex values,
- evaluation of expressions as part of a calculus of complex values,
- execution of state transitions (object creation, change of object attributes and behavior states, object destruction), and
- a user interface for visualization of object states and accepting user requests for events and ad-hoc-queries.

Specifications and object states need to be available in several successive sessions. Therefore, we need persistent representations without restrictions regarding their lifetime. Also, the complex nature of abstract syntax representations requires a powerful means for data modeling. As shown in the next section, the persistent programming environment Tycoon provides an appropriate basis for these tasks.

4 The Persistent Programming Environment Tycoon

Tycoon is a persistent programming environment that improves the construction, maintenance and operation of persistent application systems (PAS) [4]. A PAS is a software system that gives its users a flexible, problem oriented and safe access to large sets of long-lived objects of application-specific types [1]. This characterization perfectly fits to our application domain, where object specifications and instances have to be considered "long-lived objects".

The scalable Tycoon architecture integrates persistent data, programs and threads. It strictly separates tasks of storage, manipulation, modeling and representation into well-defined system layers [5]. The underlying programming language TL has a rich type system and allows for generic programming and external communication. It provides orthogonal persistence, type completeness, and higher order functions. The language TL is strictly typed and is neutral with respect to the data model.

Our animation system is completely written in TL. The implementation effort was significantly reduced by applying Tycoon's advanced concepts of expressive orthogonal language constructs and persistence.

5 The Web User Interface

In this section, we concentrate on the user interface of the animation system. The primary purpose of the user interface is to visualize the description and current state of objects. As another important aspect it provides a uniform way for the user to trigger events and to specify queries for further exploration.

A Web browser is employed for realizing the client part of the system. The visualization of information about objects can easily be managed by generating HTML-documents. Because the content of documents depends on the current state of objects, documents are always generated dynamically on request. A request may result from the selection of a hypertext link in a document, for example, to explore a different object that is referenced by an attribute in the currently selected object. By employing a hypertext system we can model relationships among objects in a natural way by providing hypertext links. Thus, the user can explore the object hierarchy by simply following the corresponding hypertext links.

The request mechanism for the dynamic generation of documents is realized by using the Common Gateway Interface (CGI) on the Web server side. Requests are embedded in an HTML-anchor by augmenting the URL with necessary information, i.e. the kind of request and any arguments like oid's of related objects or names of specifications. To provide an entry point to an animation session, it is sufficient to keep a single static document (a file) with two links, the first one pointing to the set of available specifications, the second one leading to the root object of a hierarchy of instances.

A user can manipulate objects by initiating one or more, possibly parameterized, events. Furthermore, the TROLL *light*-language allows for SQL-like queries that are evaluated in context of the currently selected object. Both, queries and event descriptions, are specified in a form field (see Fig. 1). After submitting a request, the evaluation takes place and the results are shown in a new document.

References

1. M.P. Atkinson and R. Morrison. Orthogonally persistent object systems. *VLDB Journal*, 4(3):319–401, 1995.
2. M. Gogolla, S. Conrad, G. Denker, R. Herzig, N. Vlachantonis, and H.-D. Ehrich. TROLL light – The Language and Its Development Environment. In M. Broy and S. Jähnichen, editors, *KORSO – Methods, Languages, and Tools for the Construction of Correct Software (KORSO'95)*, volume 1009 of *Lecture Notes in Computer Science*, pages 204–220. Springer, Berlin, 1995.
3. R. Herzig and M. Gogolla. An Animator for the Object Specification Language TROLL light. In Vangalur S. Alagar and Rokia Missaoui, editors, *Object-Oriented Technology for Database and Software Systems, Proc. Colloquium on Object Orientation in Databases and Software Engineering (COODBSE'94)*, pages 156–170. World Scientific, River Edge (NJ), 1995.
4. F. Matthes. *Persistente Objektsysteme: Integrierte Datenbankentwicklung und Programmerstellung*. Springer-Verlag, 1993.
5. F. Matthes, G. Schröder, and J.W. Schmidt. Tycoon: A scalable and interoperable persistent system environment. In M.P. Atkinson, editor, *Fully Integrated Data Environments*. Springer-Verlag (to appear), 1995.

This article was processed using the LaTeX macro package with LLNCS style