Design of Computing Systems:
Social and Ergonomic Considerations

Proceedings of the Seventh International Conference on Human-Computer Interaction, (HCI International '97), San Francisco, California, USA
August 24-29, 1997
Volume 2

Edited by

Michael J. Smith
University of Wisconsin, Madison, WI 53706, USA

Gavriel Salvendy
Purdue University, West Lafayette, IN 47907, USA

Richard J. Koubek
Purdue University, West Lafayette, IN 47907, USA

1997
ELSEVIER
Amsterdam - Lausanne - New York - Oxford - Shannon - Tokyo
Generating Adaptable Multimedia Software from Dynamic Object-Oriented Models: The OBJECTWAND Design Environment

Christian Márton and Michael Humpl

Fachhochschule Augsburg, Fachbereich Informatik, Baumgartnerstraße 16, D-86161 Augsburg, Germany, E-mail: maertin@informatik.fh-augsburg.de

Abstract

OBJECTWAND is a design environment for modeling and generating high-quality multimedia applications. It enables close cooperation between human developers and automatic construction tools. The system exploits abstract object-oriented models, which specify the internal features and the static and dynamic interrelationships of application domain classes. An automated design process gradually refines models into object-oriented source code. At each development stage models may be adapted to user-specific requirements.

1. INTRODUCTION

Since 1990 there has been tremendous progress in the area of user interface design environments. Today several model-based systems offer tools for specification, simulation and automatic generation of the interactive parts of application software [1-5, 8, 9, 11, 13, 14]. Most of these systems either produce flexible UI software from detailed design models or software with restricted UI structure and behavior from abstract domain models. The first approach is time-consuming and requires human developers with advanced UI design and programming skills. The second approach is limited to application domains with only modest UI requirements.

As a next step, therefore, we must provide both improved tool usability and mechanisms for designer-system collaboration in those situations where generators alone cannot achieve the required UI quality and functionality. OBJECTWAND, which is discussed in this paper, was developed with that kind of cooperative design capabilities in mind. OBJECTWAND, the successor of AME [6,7], is a model-based CASE environment for interactive applications. It provides advanced generation facilities and interactive tools for designing real world multimedia applications. The system supports:

- definition of OMT-based domain models [10], including message-based communication structure and design-specific requirements (e.g. for video, sound and speech objects);
- automatic translation into OOD models, which specify UI structure, action hierarchy, abstract interaction objects, dynamic behavior, domain functionality links;
- interactive model refinement;
- automatic adaptation to requirements of target users and environments;
- simulation of layout prototypes;
- generation of production quality target code for Delphi; and
- model reuse and synthesis based on analysis and design pattern recognition.
2. LIFE CYCLE SUPPORT FOR DESIGNER-SYSTEM COOPERATION

*OBJECTWAND* implements a cooperative design automation life cycle (figure 1). The result of each design activity is represented as an object-oriented internal model using *Application System Objects (ASOs)* [7]. In any development state automatic generators and interactive tools can be used alternatively or in combination to refine the model.

![Diagram](image)

Figure 1. *OBJECTWAND* software development life cycle activities

An *OBJECTWAND*-session starts with the definition of a domain object model for the target application. Attributes and operations of the application domain classes and their relationships (inheritance, aggregation, association) may be defined. For representing interactions between domain classes inter-class message channels can be created. They are used for sending and receiving data and activation records between the operations of different classes and for sequence control. Structural and dynamic aspects are encapsulated within a single model. Thus mapping problems during implementation can be avoided. At this stage the designer may also introduce detailed design objects or patterns [12] for modeling complex requirements, e.g. for multimedia resources. Figure 2 shows some available modeling primitives.

Many design patterns, however, can automatically be derived from analysis patterns in the domain model. An *object pattern interpreter* parses the aggregation, association and inheritance structure of the domain model and expands each known domain pattern to a pattern of related OOD classes. Such design patterns represent categories of complex UI structures like windows or dialog boxes.
Figure 2. Domain object modeling primitives and some design-specific class features

These patterns include the knowledge for modeling the static appearance and dynamic behavior of the represented UI structures in a given target environment (Windows 95 or 3.x). They cover such aspects as the default menu structure, dialog box layout constraints or method specifications for object communication, activation and deactivation. Unrecognized analysis patterns may be expanded interactively by the designer.

Figure 3. Generated Delphi example application

A rule-based component is used to generate an OOD class from each attribute declaration in a given domain class. Generated OOD classes become parts of their surrounding window or dialog box. Abstract interaction objects are automatically assigned to these OOD classes. They are selected by rules, which exploit attribute type, cardinality and additional meta-data. Each domain class operation declaration is mapped to an OOD class representing a standard or domain-specific menu entry or an action button. The number and types of interaction objects in a window determine its default layout style.
In order to improve generation results and use specific layout and presentation style rules, user profile objects can be exploited automatically. The complete animated layout is available within Borland Delphi and can be refined by the designer (figure 3). Communication channel primitives are automatically mapped to generated messages and method code within both the sending and receiving object instances. The code generator creates an object-oriented target application and automatically embeds existing domain functionality.

3. RESULTS

Experiments with building multimedia business applications and object-oriented front-ends for RDBMS applications yielded the following preliminary results:

- Mixing analysis and design classes and patterns in the initial object model may drastically shorten design time. It typically leads to better and more consistent target software than adding low-level design objects only later in the life cycle.
- The system frees developers from most routine work during design, implementation and system integration. The designer gains valuable time for domain modeling.
- To produce quality application source code, a number of life cycle iterations with gradually more interactive refinements toward later iterations are typically required.
- Future OBJECTWAND versions will include a mode that enables the system to learn previously unknown analysis patterns and their OOD mappings.

REFERENCES