PROCEEDINGS

Southern African Computer Symposium

6th de

1991

Rekendarsimposium van Suider Afrika

1991

DE

OVERBERGER

HOTEL,

CALEDON

2 - 3 JULY 1991

SPONSORED

BY

ISM

FRD

GENMIN

EDITED BY

M H Linck

DEPARTMENT OF COMPUTER SCIENCE • UNIVERSITY OF CAPE TOWN
PROCEEDINGS / KONGRESOPSOMMINGS

6th
SOUTHERN AFRICAN COMPUTER
SYMPOSIUM

6de
SUIDELIKE-AFRIKAANSE
REKENAARSIMPOSIUM

De Overberger Hotel, Caledon
2 - 3 JULY 1991

SPONSORED by
ISM
FRD
GENMIN

EDITED by
M H LINCK
Department of Computer Science
University of Cape Town
TABLE OF CONTENTS

Foreword 1

Organising Committee 2

Referees 3

Program 5

Papers (In order of presentation) 9

"A value can belong to many types"
B H Venter, University of Fort Hare 10

"A Transputer Based Embedded Controller Development System"
M R Webster, R G Harley, D C Levy & D R Woodward,
University of Natal 16

"Improving a Control and Sequencing Language"
G Smit & C Fair, University of Cape Town 25

"Design of an Object Orientated Framework for Optimistic
Parallel Simulation on Shared-Memory Computers"
P Machanick, University of Witwatersrand 40

"Using Statecharts to Design and Specify the GMA
Direct-Manipulation User Interface"
L van Zijl & D Mitton, University of Stellenbosch 51

"Product Form Solutions for Multiserver Centres
with Hierarchical Classes of Customers"
A Krzesinski, University of Stellenbosch and
R Schassberger, Technische Universität Braunschweig 69

"A Reusable Kernel for the Development of Control Software"
W Fouche and P de Villiers, University of Stellenbosch 83

"An Implementation of Linda Tuple Space
under the Helios Operating System"
P G Clayton, E P Wentworth, G C Wells and F de Heer-Menlah,
Rhodes University 95

"The Design and Analysis of Distributed Virtual Memory
Consistency Protocols in an Object Orientated Operating System"
K Macgregor, University of Cape Town & R Campbell University
of Illinois at Urbana-Champaign 107
"Concurrency Control Mechanisms for Multidatabase Systems"
A Deacon, University of Stellenbosch 118

"Extending Local Recovery Techniques for Distributed Databases"
H L Victor & M H Rennhackkamp, University of Stellenbosch 135

"Analysing Routing Strategies in Sporadic Networks"
S Melville, University of Natal 148

"The Design of a Speech Synthesis System for Afrikaans"
M J Wagener, University of Port Elizabeth 167

"Expert Systems for Management Control: A Multiexpert Architecture"
V Ram, University of Natal 177

"Integrating Similarity-Based and Explanation-Based Learning"
G D Oosthuizen and C Avenant, University of Pretoria 187

"Efficient Evaluation of Regular Path Programs"
P Wood, University of Cape Town 201

"Object Orientation in Relational Databases"
M Rennhackkamp, University of Stellenbosch 211

"Building a secure database using self-protecting objects"
M Olivier and S H von Solms, Rand Afrikaans University 228

"Modelling the Algebra of Weakest Preconditions"
C Brink and I Rewitsky, University of Cape Town 242

"A Model Checker for Transition Systems"
P de Villiers, University of Stellenbosch. 262

"A New Algorithm for Finding an Upper Bound of the Genus of a Graph"
D I Carson and O R Oellermann, University of Natal 276
FOREWORD

The 6th Computer Symposium, organised under the auspices of SAICS, carries on the tradition of providing an opportunity for the South African scientific computing community to present research material to their peers.

It was heartening that 31 papers were offered for consideration. As before all these papers were refereed. Thereafter a selection committee chose 21 for presentation at the Symposium.

Several new dimensions are present in the 1991 symposium:

* The Symposium has been arranged for the day immediately after the SACLA conference.
* It is being run over only 1 day in contrast to the 2-3 days of previous symposia.
* I believe that it is first time that a Symposium has been held outside of the Transvaal.
* Over 85 people will be attending. Nearly all will have attended both events.
* A Sponsorship package for both SACLA and the Research Symposium was obtained. (This led to reduced hotel costs compared to previous symposia)

A major expense is the production of the Proceedings of the Symposium. To ensure financial soundness authors have had to pay the page charge of R20 per page.

A thought for the future would be consideration of a poster session at the Symposium. This could provide an alternative approach to presenting ideas or work.

I would sincerely hope that the twinning of SACLA and the Research Symposium is considered successful enough for this combination survive. As to whether a Research Symposium should be run each year after SACLA, or only every second year, is a matter of need and taste.

A challenge for the future is to encourage an even greater number of MSc & PhD students to attend the Symposium. Unlike this year, I would recommend that they be accommodated at the same cost as everyone else. Only if it is financially necessary should the sponsored number of students be limited.

I would like to thank the other members of the organising committee and my colleagues at UCT for all the help that they have given me. A special word of thanks goes to Prof. Pieter Kritzinger who has provided me with invaluable help and ideas throughout the organisation of this 6th Research Symposium.

M H Linck
Symposium Chairman
SYMPOSIUM CHAIRMAN

M H Linck, University of Cape Town

ORGANISING COMMITTEE

D Kourie, Pretoria University.
P S Kritzinger, University of Cape Town.
M H Linck, University of Cape Town.

SPONSORS

ISM
GENMIN
FRD
## LIST OF REFEREES FOR 6th RESEARCH SYMPOSIUM

<table>
<thead>
<tr>
<th>NAME</th>
<th>INSTITUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barnard, E</td>
<td>Pretoria</td>
</tr>
<tr>
<td>Becker, Ronnie</td>
<td>UCT</td>
</tr>
<tr>
<td>Berman S</td>
<td>UCT</td>
</tr>
<tr>
<td>Bishop, Judy</td>
<td>Wits</td>
</tr>
<tr>
<td>Berman, Sonia</td>
<td>UCT</td>
</tr>
<tr>
<td>Brink, Chris</td>
<td>UCT</td>
</tr>
<tr>
<td>Bodde, Ryn</td>
<td>Networks Systems</td>
</tr>
<tr>
<td>Bornman, Chris</td>
<td>UNISA</td>
</tr>
<tr>
<td>Bruwer, Piet</td>
<td>UOFS</td>
</tr>
<tr>
<td>Cherenack, Paul</td>
<td>UCT</td>
</tr>
<tr>
<td>Cook Donald</td>
<td>UCT</td>
</tr>
<tr>
<td>de Jaeger, Gerhard</td>
<td>UCT</td>
</tr>
<tr>
<td>de Villiers, Pieter</td>
<td>Stellenbosch</td>
</tr>
<tr>
<td>Ehlers, Elize</td>
<td>RAU</td>
</tr>
<tr>
<td>Eloff, Jan</td>
<td>RAU</td>
</tr>
<tr>
<td>Finnie, Gavin</td>
<td>Natal</td>
</tr>
<tr>
<td>Gaynor, N</td>
<td>AECI</td>
</tr>
<tr>
<td>Hutchinson, Andrew</td>
<td>UCT</td>
</tr>
<tr>
<td>Jourdan, D</td>
<td>Pretoria</td>
</tr>
<tr>
<td>Kourie Derrick</td>
<td>Pretoria</td>
</tr>
<tr>
<td>Kritzinger, Pieter</td>
<td>UCT</td>
</tr>
<tr>
<td>Krzesinski, Tony</td>
<td>Stellenbosch</td>
</tr>
<tr>
<td>Laing, Doug</td>
<td>ISM</td>
</tr>
<tr>
<td>Labuschagne, Willem</td>
<td>UNISA</td>
</tr>
<tr>
<td>Levy, Dave</td>
<td>Natal</td>
</tr>
<tr>
<td>Name</td>
<td>Institution</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>MacGregor, Ken</td>
<td>UCT</td>
</tr>
<tr>
<td>Machanick, Philip</td>
<td>Wits</td>
</tr>
<tr>
<td>Mattison Keith</td>
<td>UCT</td>
</tr>
<tr>
<td>Messerschmidt, Hans</td>
<td>UOFS</td>
</tr>
<tr>
<td>Mutch, Laurie</td>
<td>Shell</td>
</tr>
<tr>
<td>Neishlos, N</td>
<td>Wits</td>
</tr>
<tr>
<td>Oosthuizen, Deon</td>
<td>Pretoria</td>
</tr>
<tr>
<td>Peters Joseph</td>
<td>Simon Fraser</td>
</tr>
<tr>
<td>Ram, V</td>
<td>Natal, Pmb.</td>
</tr>
<tr>
<td>Postma, Stef</td>
<td>Natal, Pmb</td>
</tr>
<tr>
<td>Rennhackkamp, Martin</td>
<td>Stellenbôsch</td>
</tr>
<tr>
<td>Shochot, John</td>
<td>Wits</td>
</tr>
<tr>
<td>Silverberg, Roger</td>
<td>Council for Mineral Technology</td>
</tr>
<tr>
<td>Smit, Riel</td>
<td>UCT</td>
</tr>
<tr>
<td>Smith, Dereck</td>
<td>UCT</td>
</tr>
<tr>
<td>Terry, Pat</td>
<td>Rhodes</td>
</tr>
<tr>
<td>van den Heever, Roelf</td>
<td>UP</td>
</tr>
<tr>
<td>van Zijl, Lynette</td>
<td>Stellenbosch</td>
</tr>
<tr>
<td>Venter, Herman</td>
<td>Fort Hare</td>
</tr>
<tr>
<td>Victor, Hema</td>
<td>Stellenbosch</td>
</tr>
<tr>
<td>von Solms, Basie</td>
<td>RAU</td>
</tr>
<tr>
<td>Wagenaar, M</td>
<td>UPE</td>
</tr>
<tr>
<td>Wentworth, Peter</td>
<td>Rhodes</td>
</tr>
<tr>
<td>Wheeler, Graham</td>
<td>UCT</td>
</tr>
<tr>
<td>Wood, Peter</td>
<td>UCT</td>
</tr>
</tbody>
</table>
6TH RESEARCH SYMPOSIUM - 1991

FINAL PROGRAM

TUESDAY 2nd July 1991

10h00 - 13h00 Registration
13h00 - 13h50 PUB LUNCH

14h00 - 15h30 SESSION 1A
Venue: Hassner
Chairman: Prof Basie von Solms

14h00 - 14h30
"A value can belong to many types."
B H Venter, University of Fort Hare

14h30 - 15h00
"A Transputer Based Embedded Controller Development System"
M R Webster, R G Harley, D C Levy & D R Woodward, University of Natal

15h00 - 15h30
"Improving a Control and Sequencing Language"
G Smit and C Fair, University of Cape Town

SESSION 1B
Venue: Hassner C
Chairman: Prof Roelf v d Heever

14h00 - 14h30
"Design of an Object Orientated Framework for Optimistic Parallel Simulation on Shared-Memory Computers" P Machanick, University of Witwatersrand

14h30 - 15h00
"Using Statecharts to Design and Specify the GMA Direct-Manipulation User Interface" L van Zijl & D Mitton, University of Stellenbosch

15h00 - 15h30
"Product Form Solutions for Multiserver Centres with Hierarchical Classes of Customers" A Krzesinski, University of Stellenbosch and R Schassberger, Technische Universität Braunschweig

15h30 - 16h00 TEA
16h00 - 17h30 SESSION 2A

Venue: Hassner
Chairman: Prof Derrick Kourie

16h00 - 16h30
"A Reusable Kernel for the Development of Control Software" W Fouche and P de Villiers, University of Stellenbosch

16h30 - 17h00
"An Implementation of Linda Tuple Space under the Helios Operating System" P G Clayton, E P Wentworth, G C Wells and F de-Heer-Menlah, Rhodes University

17h00 - 17h30
"The Design and Analysis of Distributed Virtual Memory Consistency Protocols in an Object Orientated Operating System" K MacGregor, University of Cape Town & R Campbell, University of Illinois at Urbana-Champaign

19h30 PRE-DINNER DRINKS

20h00 GALA CAPE DINNER
(Men: Jackets & ties)
**WEDNESDAY 3rd July 1991**

7h00 - 8h15  BREAKFAST

8h15 - 9h45  SESSION 3A

Venue: Hassner

Chairman: Assoc Prof P Wood

8h15 - 8h45  "Concurrency Control Mechanisms for Multidatabase Systems" A Deacon, University of Stellenbosch

8h45 - 9h15  "Extending Local Recovery Techniques for Distributed Databases" H L Victor & M H Rennhackkamp, University of Stellenbosch

9h15 - 9h45  "Analysing Routing Strategies in Sporadic Networks" S Melville, University of Natal

9h45 - 10h15  TEA

10h15 - 11h00  SESSION 4

Venue: Hassner

Chairman: Prof P S Kritzinger

Invited paper: E Coffman

11h00 - 11h10  BREAK

SESSION 3B

Venue: Hassner C

Chairman: Prof G Finnie

8h15 - 8h45  "The Design of a Speech Synthesis System for Afrikaans" M J Wagener, University of Port Elizabeth

8h45 - 9h15  "Expert Systems for Management Control: A Multiexpert Architecture" V Ram, University of Natal

9h15 - 9h45  "Integrating Similarity-Based and Explanation-Based Learning" G D Oosthuizen and C Avenant, University of Pretoria
11h10 - 12h40 SESSION 5A

Venue: Hassner

Chairman: Prof C Bornman

11h10 - 11h40
"Efficient Evaluation of Regular Path Programs"
P Wood, University of Cape Town

11h40 - 12h10
"Object Orientation in Relational Databases"
M Rennhackkamp, University of Stellenbosch

12h10 - 12h40
"Building a secure database using self-protecting objects" M Olivier and S H von Solms, Rand Afrikaans University

SESSION 5B

Venue: Hassner C

Chairman: Prof A Krzesinski

11h10 - 11h40
"Modelling the Algebra of Weakest Preconditions"
C Brink & I Rewitsky, University of Cape Town

11h40 - 12h10
"A Model Checker for Transition Systems"
P de Villiers, University of Stellenbosch

12h10 - 12h40
"A New Algorithm for Finding an Upper Bound of the Genus of a Graph"
D I Carson and O R Oellermann, University of Natal

12h45-12h55 GENERAL MEETING of RESEARCH SYMPOSIUM ATTENDEES

Venue: Hassner

Chairman: Dr M H Linck

13h00 - 14h00 LUNCH

FINIS 6th COMPUTER SYMPOSIUM
PAPERS
of the
6TH RESEARCH SYMPOSIUM
Using Statecharts to Design and Specify a Direct-Manipulation User Interface

Lynette van Zijl and Deon Mitton
Institute for Applied Computer Science,
University of Stellenbosch, 7600 Stellenbosch,
Republic of South Africa

Abstract

Statecharts were developed by Harel et al [10] to specify complex reactive systems. In this paper we report on our application of statecharts as a design and specification tool for an X-Windows based Graphical User Interface for the telephone network performance modelling tool GMA.

1 Introduction

The design and specification of a user interface is often perceived as a straightforward form layout task. However, now that graphical direct-manipulation interfaces have become the norm, the complexity of the design, specification and coding of the user interface equals that of any other system component. This complexity led to a renewed interest in the specification of user interfaces over the last five years [4]. At the Institute for Applied Computer Science (ITR), where we develop commercial performance evaluation packages, we experienced the need for a modern specification methodology for direct-manipulation graphical user interfaces.

Before we discuss specification methodologies for user interfaces, let us first recapture on the structure of a typical user interface. One of the best-known models for a user interface is the Seeheim model [6]. In this model, the interface is conceived as having three components: The Presentation Component, the Dialogue Component and the Application-Interface Component (see figure 1). The Presentation Component concerns the visual aspects of the interface as presented to the user; the Dialogue Component concerns the dialogue between the user and the interface; and the Application-Interface Component concerns the connection between the application and the interface. Issues relating to the Presentation and the Application-Interface Components are less well known than the issues relating to the Dialogue Component, for which a number of specification methodologies have been proposed.

Specification methodologies can be loosely classified as being either formal or informal, and either being graphical or non-graphical. There are valid arguments for and against the choice of any of these languages; we opted for a formal graphical specification language (see section 3 for our motivations). Graphical specification languages that are based on state transition diagrams are powerful yet easy to understand and use. However, state transition diagrams suffer from two major problems when used to specify user interfaces. The first problem is the state explosion that occurs in any sizable system, and the second problem is the inability of the standard transition diagram to model concurrency. Many extensions/adaptations to state transition diagrams have been suggested in attempts to overcome these problems. One such adaptation is Harel's statecharts (see [10]).

We applied the statechart methodology in the design and specification of a large direct-manipulation interactive user interface. In our experience we found that Harel's statecharts overcome the problems in transition diagram specifications, in that they are suited to the specification of user interfaces based on asynchronous events (such as the industry standard X-
Windows system). Additionally, we found that statecharts can be applied to design all three of the Seeheim user interface components. As such, the statecharts provide a unified approach to interactive user interface design and specification.

The rest of this paper is organised as follows: Section 2 provides a brief introduction to the syntax and semantics of statecharts. Section 3 sketches the background to the project for which the user interface has been designed, and section 4 contains a discussion of the actual design and specification (with ample examples).

2 Statecharts Revisited

Statecharts were developed by David Harel in the mid 1980s, and are described in a number of papers [10, 12, 13]. Harel developed the statecharts as an answer to the challenge of Green [8] who claimed that there are no adequate graphical specification tools available to model interactive behaviour. Finite state machines and their transition diagrams, for example, are easy to understand, but suffer from an exponential state explosion problem.

Harel developed the generalised concept of a higraph [11] which is a diagramming object that combines the properties of hypergraphs and Venn diagrams. The higraph can therefore be used to represent a set of elements (with a special relation on them) and then to represent a collection of such sets with structural relationships among them. The interested reader can consult [11] for more information concerning higraphs; let it suffice at the moment to note that the higraph is a formally defined mathematical object, with a visual representation of its topological characteristics. The statechart is just an application of a higraph; it is essentially a higraph-based version of a finite state machine.

Since the statechart is based on a finite state machine, it is empowered with all the ergodicity of a state transition diagram. Moreover, it solves the state explosion problem found in finite state machines by having the properties of depth and orthogonality. The depth property allows for hierarchical modelling by clustering or refinement, so that atomic elements can be clustered graphically into a single composite element, or a composite element graphically refined into several smaller elements. The orthogonality property can be used to model independence and concurrency, by combining elements graphically in a representation of their Cartesian product.

Before illuminating these concepts with an example, let us first highlight the graphical symbols used to construct a statechart. Statecharts graphically represent the states and events in a system. The symbol used for a state is a rounded rectangle, with the name of that state inside the rectangle. Hierarchies of states are represented by enclosure of states inside outer states (see figure 2). Events are represented by arrows, with the arrow labelled by the name of the event and any condition which may guard that event. There are different types of arrows
available. The standard arrow is that which originates on the perimeter of one state and ends on the perimeter of another state (see Event-1 in figure 3). The default entry state is labelled by a default arrow (such as Event-0 in figure 3). The hierarchical arrow indicates that an event originates in a lower level state inside the current state. For example, in figure 3 State1 is a hierarchical state with substates on a lower level. Here Event-2 originates from one of the states within State1 and not from State1 as such.

Now that we have the symbols for a statechart available, let us use them to model a small example. Suppose that the system to be specified is a chocolate vending machine called C (the original vending machine idea is due to Hoare[14]). The vending machine is always willing to accept money, irrespective of whether it has the chocolates to uphold its pledge! However, it is at least honest – if it has no more chocolates to exchange for the money, it returns your coin. It is also a fairly clever vending machine; the slot in the machine allows only the correctly sized coin to be inserted, and other objects entered by dishonest or hungry customers are simply ignored. The first step in the statechart specification is to identify the states and the events in
the system. To the outside observer, there are two visible states: The READY state in which the machine is willing to accept money, and the BUSY state in which it processes a request for a chocolate. There are three relevant events: The event of a coin insertion into the machine, the event of a chocolate returned to the customer, and the event of a coin returned to the customer. This can be represented in a statechart as depicted in figure 4. Note the different types of arrows used to represent the different types of events.

One can now continue to refine the system C. Clearly, there must be some busy state in which C has chocolates to return to the customer, and some busy state in which no chocolates are available. Note in figure 5 how the enclosure of the states NO-CHOCS and MORE-CHOCS in the state BUSY indicates that only one of those states can hold at any one time (that is, encapsulation enforces an XOR).

It now remains to specify the event(s) which lead to the states NO-CHOCS and MORE-CHOCS from the READY state. Obviously, it is the same event in both cases, namely, the event coin-accepted. Since the same event leads to all the states enclosed in the BUSY state, we can use one arrow ending at the perimeter of the BUSY state (see figure 6).

Suppose now that there is a vending machine with exactly the same functionality as C, but which dispenses toffees. Let us call this machine T. We can now design a vending machine called CT from the specifications of C and T, which operates as follows: There are two slots for entering coins, one for a toffee and one for a chocolate (the price is the same). Suppose that a customer wants to buy a toffee and enters a coin in that slot. As before, if there are any toffees available, the machine delivers a toffee. However, if there are no toffees available, CT first checks whether there are any chocolates available. If there are, it returns a chocolate instead of a toffee. If there are neither toffees nor chocolates available, it returns the customer's coin. A similar return policy applies if a customer requests a chocolate. Graphically CT can be specified by a statechart as depicted in figure 7. Note that the dashed line indicates the Cartesian product of the two machines, so that it is in principle possible to simultaneously enter a coin for a toffee and a coin for a chocolate. The dashed line thus enforces an AND condition for the states in the system.

Another useful modelling mechanism is the selection arrow. The selection arrow is indicated by a circled S on the arrowhead of an event arrow. It can be interpreted as a generic event, with a number of clearly defined values to indicate which state is triggered by the event. For example in a menu driven system, this mechanism can be used to indicate the event of selecting one of many possible menu items. Detailed examples using this mechanism are given in section 4.
Figure 5: Chocolate Vending Machine Stage 2

Figure 6: Chocolate Vending Machine
The explanation above covers only those aspects of statecharts that we need to design and specify the GMA example in section 4. The reader is again urged to consult the original papers for more information.

3 The Graphical User Interface for GMA

The GMA (Graphical Modelling and Analysis) package is a performance modelling tool for telecommunications networks such as the South African public data network Saponet-P and the South African telephone network. The package has already been described in other papers ([3, 17]) and this section provides a summary of the overall project followed by more detail on the aspects relevant to this paper.

The GMA package runs under Unix, with graphics based on the X-Windows system using the X Toolkit and OSF/Motif. The structure of the GMA package is shown diagrammatically in figure 8. The package consists of three independent modular units: The Data Extraction Software, the Model Solution Kernel and the Graphical User Interface. Both the data extraction software and the model solution kernel are independent of the graphical interface; a standard interface to each module serves to hide its implementation details. This facilitates one of the main advantages of the package, namely its utility in modelling diverse computer network implementations. Such diverse modelling is achieved by modifying the front-end modules to cater for installation-specific features. The graphical interface controls the extraction of configuration and traffic data from the communication network under study. It presents a diagram of the network to the user and constructs the performance model of the network. The model is solved by the solution kernel, and results are presented graphically on the diagram.

One of the main goals of the GMA project was to produce a modelling package which could be used by network engineers with no training in the construction of statistical performance evaluation models. This implies that we had to design an interface that would present a graphical view of the network in a familiar form. This presented no problems in the case of data communication networks, as GMA could simply present the user with a geographical and/or a

![Figure 7: Chocolate and Toffee Vending Machine](image)
schematic view of the network nodes. The user can then switch between the geographical and schematic view, or zoom into a specific node of the network (see figure 9).

The interface for the telephone network did however present immediate problems. Typical telephone networks are hierarchical in nature, with calls routed among the different levels, as depicted in figure 10. Moreover, for historical reasons the typical telephone network actually consists of various overlay networks, with the different networks connected by crossover points (see figure 11). In addition, the characteristics of the network differ depending on geographical location—for example, the trunking and routing decisions for metropolitan areas differ from that of rural areas. Telephone networks also contain many more nodes than the typical data communication network. For example, the South African telephone network contains approximately 1100 nodes, while Saponet-P contains 31 nodes.

A graphical interface for the modelling of a telephone network thus has to take all the above factors into account. The user should be presented with a hierarchical outline view, a view of only one level of the hierarchy, and localized geographical area views. A facility must also be available to zoom in on the small representation of a node, the small representation being necessitated by the large number of nodes in the network. It should be possible to switch between representation modes without affecting the network information known to the system.

A user-friendly interface for the telephone network modelling in GMA is therefore clearly a complex system. The design of such an interface has to take into account all the events possible in any one mode, and the effects of such events. We found a natural language specification of such an interface to be open to too much ambiguity, and started a search for a formal design and specification tool suited to our needs. Axiomatic (for example, [2]) and other textual specifications (such as BNF, see [15]) were discarded, because they lead to specifications that are difficult to read and maintain for large complex interfaces such as the GMA interface. We thus set out to find a graphical specification method whereby we could design and specify an event-driven interactive user interface. Specification systems without a formal underlying methodology (see [9]) were discarded, mainly because of our interest in the provable correctness of the GMA system ([19] discusses this issue). The need for a formal graphical specification method pointed to the use of some extension or variation of state transition diagrams, such as that of Wasserman [20], Zave[21] or Harel[10]. Wasserman’s USE methodology uses ATN’s (Augmented Transition
Figure 9: The Schematic and Geographical Modes in GMA
3 The Graphical User Interface for GMA

Figure 10: The Hierarchical Levels in a Telephone Network

Figure 11: Two Overlay Networks in a Telephone Network
The Design and Specification of the GMA Interface

In this section we show how the GMA interface was progressively designed and specified using statecharts. Throughout the discussion we point out how statecharts can be used to design all three the components of the user interface, and we comment on the ease of use of its concurrency specification mechanism. We shall follow a standard procedure in the design and specification of the GMA user interface. We always start with the design of the first component of the Seeheim model (i.e. the Presentation Component) and given the Presentation design, we proceed to specify the Dialogue Component in more detail. The Application Interface Component can then be designed using the information about the external events in the Dialogue Component.

The GMA package should present the user with a logo on startup time, after which the working screen should be displayed. So, a global view of the overall system can be specified as in figure 12 where there are two non-concurrent states: The LOGO state and the GMA-PROPER state. Note how the design for the actual presentation of the screens (the Seeheim Presentation Component) is visible from the statechart: the XOR of the two states LOGO and GMA-PROPER with the given events implies that those two windows/screens cannot be visible simultaneously. The default entry point to the system is the LOGO state, through the external event gma. The event ok places the system in the GMA-PROPER state, from which the event CTRL-Q will cause a quit from the system. Since there is no event back from GMA-PROPER to LOGO, it is not possible to switch between these two visual presentations. The ok event together with the LOGO and GMA-PROPER states define the Dialogue Component, while the external events gma and CTRL-Q define the Application-Interface Component.

Let us look at the state GMA-PROPER in more detail. The visual layout of the window must comply with the OSF/Motif style guide and our house rules on screen layout and as such consist of three display areas: A Status Display Area, a Menubar and a Working Area. All three these areas should be visible simultaneously. The window layout (Presentation Component) can for example be presented as in figure 13. The corresponding statechart, with the relevant events, is given in figure 14. Note that, although the design of the Presentation Component of the user interface is clearly only informal, the statecharts do allow the designer to form a preliminary picture of the layout of the window as the design progresses. As before, the design of the Application-Interface Component can be derived from the (only) external event ok.

Figure 12: A global view of the GMA system

Nets), and suffer from the state explosion problem. Although Zave's sequence diagrams have the same expressive power as statecharts, we found the explicit expression on concurrency easier to understand and use. We thus set out to design and specify the GMA interface using statecharts.
Figure 13: The Main Window Layout for GMA

Figure 14: GMA-PROPER
The Design and Specification of the GMA Interface

4

Each of the three display areas can now be specified separately. The Status Display Area is the simplest of the three in terms of the possible events. It is an example of a display area where the user cannot enter any information but the application can set or display status information. The events that can be input to the Status Display Area are all of the same generic type (i.e. set one of the status flags) and we use the selection arrow to specify these events (see figure 15). Note that 'illegal' events are ignored by the statusbar, in that there is simply no reaction.

The second area of the GMA-PROPER window is the Menubar. The Menubar Area consists of a constantly visible menu bar, drop-down menus and pop-up selection boxes. Again, the selection of a menu item from the menu bar is specified using a selection arrow. Notice how an event can be aliased to hide implementation detail: The OptionSelected event can be implemented to allow multiple methods to select an option, such as a click on the mouse pointer and keyboard input (see figure 16). The MenuBar state itself is simple to specify, as illustrated in figure 17.

For the state DropdownMenus, we specify a generic dropdown menu to illustrate the applicability of statecharts to this menu type. Consider any drop-down menu system with $n$ dropdown menus. Each menu is either visible (when it is selected) or not visible. However, more than one menu cannot be visible simultaneously. The user can move between menus by dragging the pointer to the left or the right, or by exiting a menu and selecting a new option (on the menu bar). The statechart that models state DropdownMenus is depicted in figure 18.

When a dropdown menu is visible, a number of options are displayed, with the pointer indicating the current (highlighted) option. One can move the pointer to switch options, and
then click to select a new option. Alternatively, one can use mnemonics or accelerators from the keyboard to choose any of the options in that particular drop-down menu. Figure 19 contains the statechart for the visible state of a drop-down menu.

As a typical example of a popup menu, we selected to design and specify a Threshold Editor box. In the GMA context objects are colour coded to indicate their 'safety' level according to a certain threshold. The system has default numerical values for all thresholds, but the user is allowed to change them with the Threshold Editor. The editing should occur graphically and/or numerically, and the changes in the values should be visible in colour and in numerical values.

We use a scale with three different colours, where the boundaries of the coloured areas of the scale may be dragged to edit the thresholds. As usual in graphical user interface applications, three standard buttons are provided to commit the changes, cancel any changes, or to provide help. The Presentation Component of this popup menu is given in figure 20, with the corresponding statechart in figure 21.

The easiest specification in the Threshold Editor is the BUTTONS state. The mouse pointer may be in the buttons area of the Threshold Editor box, without being on any of the buttons. In that case, none of the buttons are selectable. We call this state the IDLE state. If the mouse pointer is on any of the buttons, then the buttons are potentially selectable. We call this state, for each button, its ACTIVE state. When the mouse pointer is clicked on a button, its corresponding reaction is activated. This is the SELECTED state for each button. So, the statechart looks like figure 22.

The statechart for the BUTTONS state in figure 22 is a clear example of how the Application-
Figure 19: A Visible Drop-down Menu

Figure 20: The Layout for the Threshold Editor

Figure 21: The Statechart for the Threshold Editor
The Design and Specification of the GMA Interface

4 The Design and Specification of the GMA Interface

Interface Component can be designed using statecharts. The external events from each of the button SELECTED states defines the (in X-Windows terminology) 'hook' between the interface and the application.

The NUMERALS and SCALE states can be specified in a straightforward fashion, reminiscent of the statechart for the Status Display Area in figure 15. The statecharts are given in figure 23.

Up to this point, we have illustrated how to design some standard graphical user interface features using statecharts. However, the third area of the GMA-PROPER state is the Working Area, which consists mostly of application-specific graphics as opposed to the general menu system. We will now show how the statecharts measured up to their potential in the design of the Working Area.

As explained in section 3, the Working Area must present to the user of the system both a geographical and a schematic view of the network. Additionally, the user must be able to zoom in on a specific geographical area, and switch between geographical and schematic views as required. Moreover the user should be able to select the hierarchy of the network currently visible (refer again to figure 10). The statechart to model these requirements are given in figure 24.

Each of the states in figure 24 can now be refined into a more detailed statechart. For example, let us consider the AREA-GEOGRAPHICAL state (see figure 25). Upon state entrance, a selection determines which geographical area should be displayed. The corresponding set of nodes are displayed (the NODE state indicates the graphical display of a node with its corre-
Figure 24: The GMA Working Area Statecharts

Figure 25: The Area Geographical Statechart

sponding links). For any of these nodes, the node can be active or selected. Note again that, as in the Threshold Editor Box, the system can be in an idle state. Further refinements can now be designed according to the action taken upon node selection.

The reader will notice that, in the more detailed statecharts, how the user interactions are modelled by the events among states. In the implementation of these statechart designs, these events usually specify the third component of the Seeheim model, namely, the Application-Interface Component.

This completes the detail of our design of the GMA direct-manipulation user interface with statecharts.

We believe that this section supports our claim on the superiority of statecharts as a specification mechanism for interactive event-driven systems.

5 Conclusion

In this paper we set out to show that statecharts are well-suited to the formal specification of event-driven user interfaces based on X. We believe that this claim is substantiated by the empirical results illustrated in section 4. In that section we discussed the design of a non-trivial direct-manipulation graphical user interface, and showed how the statecharts can be applied to design all three components of the Seeheim model. In summary, the main advantages of the statechart approach in our experience are the following:
• The statechart itself has been formalized completely, increasing the feasibility of its use in systems where formal correctness is an issue.

• Of the graphical design methodologies that we considered, the statecharts provide the easiest and most natural way to model concurrency. This is essential in the case of interfaces based on X-Windows.

• The statechart design can proceed hierarchical, ensuring that the designer can visually modularize the design so that the detail required at any stage is minimal.

• The statechart is a natural formalism for most programmers, so that the learning curve for the methodology is remarkably short.

• All the components of the Seeheim user interface model can be designed with the statecharts, providing a uniform approach to the design and specification of the user interface.

One of the only disadvantages of the statechart approach that we encountered, was that (specifically in object-orientated user interfaces) the state-based approach was not always the most natural approach to describe graphical objects. However, after a few examples we found that we could exploit the generic similarity between graphical object behaviour, so that most of our designs are adaptations of previous designs. This is in fact a hidden advantage, since one can thereby ensure that the behaviour of the interface is constant throughout the system.

References


5 Conclusion


