Sistema de Gerenciamento Distribuído de Interfaces Homem-Computador

Fábio Nogueira de Lucena[+]
Hans K.E. Liesenberg[*]

[+]Instituto de Informática (UFG)
fabio@inf.ufg.br
[*] Instituto de Computação (Unicamp)
hans@dcc.unicamp.br

Resumo

Interfaces concorrentes e distribuídas são cada vez mais comuns. Embora ferramentas para apoiar o desenvolvimento deste nicho de software sejam promissoras, elas ainda são poucas, limitadas funcionalmente, impõem requisitos que não são comuns em ambientes de produção e oferecem poucas abstrações. O gerenciador de diálogo é um dos componentes mais complexos de tais interfaces. O Sistema de Gerenciamento Distribuído de Interfaces (SGDI) Xchart é uma proposta que emprega uma linguagem de alto nível (Xchart) para capturar este gerenciador, assume a existência de recursos normalmente disponíveis em ambientes de produção e apóia a execução deste componente, o que inclui o suporte ao modelo de múltiplos agentes oferecido por Xchart. Neste texto são apresentadas as principais idéias subjacentes ao SGDI Xchart e como ele provê os recursos oferecidos.

Palavras Chave: **ferramenta de desenvolvimento de interfaces, arquitetura de software, especificação de diálogo**

1. INTRODUÇÃO

Certos tipos de sistemas interativos devem evidenciar aos usuários a concorrência inerente às funções que realiza ou a concorrência necessária na interface por questões de usabilidade. Também são comuns sistemas interativos onde as suas interfaces operam de forma assíncrona com as respectivas aplicações (componente responsável pela semântica). Em geral, nestes sistemas, a aplicação pode envolver longos processamentos que não devem exigir que a interface permaneça "congelada" durante este período, ou ainda, a aplicação continuamente produz dados que devem ser exibidos concorrentemente com as entradas fornecidas pelos usuários. Há casos onde um mesmo conjunto de dados pode ser observado sob pontos de vista distintos e qualquer alteração nos dados deve repercutir em todas as visões. Ainda existem sistemas cujo desenvolvimento é praticamente inviável sem o emprego de concorrência. Sistemas multimodais, por exemplo, muitas vezes empregam vários computadores interligados, com funções bem definidas, para prover a interação com o usuário [7]. Nestes exemplos, concorrência é a forma mais natural, algumas vezes a única, de abordar o desenvolvimento de suas interfaces.

Em sistemas que empregam concorrência, o gerenciador de diálogo (comentado adiante), tem a responsabilidade de realizar tarefas complexas. A linguagem Xchart foi especialmente desenvolvida para capturar a funcionalidade deste componente de interfaces. O SGDI Xchart apóia o emprego da linguagem Xchart durante todo o ciclo de vida do desenvolvimento deste componente e, em especial, em tempo de execução, através de vários recursos exigidos pela execução possivelmente distribuída de Xchart.

Este texto comenta trabalhos correlatos e define SGDI na próxima seção. Posteriormente é apresentada a linguagem Xchart, que é inovadora, entre outros motivos, por permitir a coexistência harmônica de conceitos das duas linhas de pesquisas mais exploradas nesta área: diagramas de transição de estados e modelo de eventos. Para identificar o que é suscetível de ser descrito nesta linguagem, uma seção posterior propõe uma arquitetura de software para sistemas interativos. Posteriormente é apresentada a arquitetura do SGDI Xchart e o processo de desenvolvimento apoiado por este sistema, siguidos das considerações finais.

É natural que o leitor sinta falta de detalhes acerca do SGDI Xchart, que podem ser obtidos no endereço URL do projeto, onde estão disponíveis documentos que cobrem os vários aspectos de Xchart.

2. INTERFACES E FERRAMENTAS DE APOIO

Interfaces são componentes importantes de sistemas interativos e de desenvolvimento complexo, onde custos elevados e vários desafios persistem [18], apesar de contínuos progressos da área. Atualmente existe um amplo mercado para ferramentas que enfrentam tais desafios. Entretanto, as dificuldades de confecção de tais ferramentas as remetem para domínios especializados ou fazem com que os recursos oferecidos contemplem apenas parte dos desafios. Além disso, ainda existem nichos na área pouco explorados, embora promissores, tais como interfaces para sistemas cooperativos e distribuídos [17], onde o desenvolvimento de interfaces não conta com o apoio de ferramentas especializadas para enfrentar de forma sistemática os problemas peculiares.

Ferramentas de apoio à construção de interfaces são conhecidas por SDIs (Sistemas de Desenvolvimento de Interfaces) ou ainda SGIs (Sistemas de Gerenciamento de Interfaces). Em inglês os termos empregados são UIDS (*User Interface Development System*) e UIMS (*User Interface Management System*), respectivamente. Tanto SGIs quanto SDIs estão relacionados, entre outros, a notações, métodos, técnicas e ferramentas voltados para o projeto e implementação de interfaces. Contudo, um SGI difere de um SDI por apoiar, especialmente, a fase de execução de uma interface. O presente texto discorre sobre a proposta Xchart, que foi devidamente talhada para dar suporte à concorrência no gerenciador de diálogo. Este suporte é um recurso inexistente na grande maioria dos SGIs. Esta concorrência pode ser explorada através de uma execução distribuída do gerenciador de diálogo sobre nós distintos de uma rede e, em conseqüência, a proposta é designada de Sistema de Gerenciamento Distribuído de Interfaces (SGDI).

As técnicas e notações pertinentes a um SGI auxiliam na partição, organização e descrição do software de interfaces de acordo com uma arquitetura alvo adotada pelo SGI. Dentre as várias propostas existentes de arquiteturas, prevalece a divisão do software de uma interface nos componentes de apresentação, gerenciamento de diálogo e acoplamento com a aplicação.

Conforme Morse [17], o gerenciador de diálogo é um dos componentes mais complexos e onde menos progressos foram registrados, ao contrário do que ocorreu com a apresentação. A maior parte das ferramentas existentes limitam-se a fornecer recursos para o desenvolvimento da apresentação[1] e deixam a cargo do programador o desenvolvimento "manual" do código correspondente ao gerenciador de diálogo. Por exemplo, Visual Basic (Microsoft) e Delphi (Borland) enquadram-se neste cenário. Elas não fornecem mecanismos apropriados para implementar gerenciadores de diálogo e tampouco dão suporte à concorrência. *O SGDI Xchart é uma proposta voltada para o desenvolvimento de gerenciadores de diálogo: modelagem, organização, implementação e execução deste software, que pode estar dividido em objetos (agentes) distribuídos sobre nós de uma rede.*

Ao contrário de muitas abordagens disponíveis, o SGDI Xchart transfere para si a responsabilidade funcional do gerenciador de diálogo capturado através de modelo executável e de alto nível descrito em Xchart. Neste sentido, a proposta aqyui apresentada assemelha-se àquelas baseadas em modelos [20].

A ferramenta mais conhecida para desenvolver interfaces concorrentes é SASSAFRAS [9]. Ela é acadêmica, assume um ambiente de produção não convencional, emprega uma linguagem baseada no modelo de eventos e não permite a construção de interfaces realmente distribuídas. O moderno e comercial SGI *Alpha* [13] tem características bem similares as de SASSAFRAS. A linguagem e a arquitetura Rendezvous foram propostos para simplificar a construção de aplicações multimidia distribuídas e, em particular, interfaces gráficas que empregam manipulação direta [10].

O SGDI Xchart dinstingue-se das propostas acima pela linguagem empregada, que permite definir regras no mesmo sentido que as outras propostas sem perder o apelo visual de uma linguagem gráfica, pelos recursos que dão suporte à distribuição e pela arquitetura de software promovida. Cada um destes elementos são discutidos a seguir.

[1] `http://www.cs.cmu.edu/afs/cs.cmu.edu/user/bam/www/toolnames.html`.

3. NOÇÕES DE XCHART

Embora a notação de Diagramas de Transição de Estados (DTEs) seja a mais estudada para a especificação de gerenciadores de diálogo e já vem sendo usada nesse sentido desde 1968 [19], existem limitações bem conhecidas com esta proposta. Algumas foram eliminadas pela notação *Statecharts* [8], que possui vários indicadores positivos para ser empregada com esta finalidade [23, 24]. Contudo, inexiste um empreendimento mais sério envolvendo Statecharts.

O intuito de preencher esta lacuna levou à realização de vários estudos, que permitiram identificar dificuldades de Statecharts quando empregada neste contexto [16]. A eliminação de tais dificuldades deram origem a Xchart, que toma emprestado várias características de Statecharts (léxicas, sintáticas e semânticas). Alguns recursos foram acrescentados, removidos e outros sofreram ajustes. Em outras palavras, Xchart é uma proposta com o objetivo de melhorar a qualidade com que gerenciadores de diálogos podem ser capturados em Statecharts.

A sofisticada semântica de Xchart está formalmente definida [14]. A semântica formal é imprescindível à execução de especificações em Xchart. Uma descrição informal de Xchart, exemplos, a definição da semântica formal e as ferramentas do ambiente podem ser obtidos via WWW.

Algumas características inovadoras de Xchart, em relação a outras linguagem empregadas para descrever gerenciadores de diálogo são apresentadas abaixo.

Modelo cooperativo. Múltiplos agentes concorrentes, criados e destruídos dinamicamente, possivelmente distribuídos, compartilhamento de memória e sinalização de eventos distribuídos. Estes agentes permitem modelar tarefas que podem ser executadas concorrentemente. Cada agente é representado por um diagrama em Xchart. Um diagrama em Xchart também pode recursivamente chamar outra diagrama no mesmo sentido que *Recursive Transition Networks.*

Hierarquia de eventos, instâncias de eventos e dados acoplados a eventos. Eventos são instâncias de classes especiais, que podem ser estruturadas em forma de uma hierarquia. Dados podem ser associados a eventos, que serão visíveis onde o evento for visível.

Regras. A combinação de um evento, uma guarda e uma ação forma uma regra. Regras rotulam transições ou ocorrerem no interior de estados. Este recurso permite a definição de mecanismos tão poderosos quanto os tratadores de eventos de abordagens baseadas no modelo de eventos.

Não determinismo. A semântica de Xchart foi reformulada com o intuito de aumentar o determinismo de especificações nesta linguagem. Embora torne a semântica mais intuitiva, este recurso implica em dificuldades na definição formal, conforme [12].

3.1. Sistemas Interativos Típicos para o emprego de Xchart

Sistemas interativos onde *o usuário pode conduzir várias tarefas simultaneamente*; sistemas cujas *interfaces são intrinsecamente complexas* devido ao domínio da aplicação (por exemplo, editoração eletrônica, CAD). As interfaces de tais sistemas apresentam comportamentos que requerem uma notação adequada para descrevê-los; sistemas responsáveis por *manter várias apresentações distintas de uma mesma informação*, em geral, ao atuar sobre uma das apresentações, as demais também devem refletir as alterações efetuadas; sistemas nos quais há uma *interação contínua com o usuário, enquanto a aplicação é executada concorrentemente* (por exemplo, a interface deve fornecer alguma forma de realimentação contínua da aplicação, mesmo enquanto o usuário permanece interagindo com o sistema); e *sistemas cooperativos e jogos* são sistemas cujos gerenciadores de diálogo de suas interfaces podem se beneficiar dos recursos providos pelo SGDI Xchart.

3.2. Exemplo

Na Figura 1 vemos cinco dos quase 80 diagramas em Xchart que compõem a especificação do gerenciador de diálogo do SPRING (sistema de informação geográfica desenvolvido pelo INPE). Este sistema compreende mais de 250.000 linhas de código, que adicionadas à complexidade das tarefas que podem ser realizadas pelos usuários tornam-no em um excelente meio para avaliar o emprego da linguagem e do ambiente Xchart.

A especificação completa em Xchart do gerenciador de dialogo do SPRING pode ser obtida através do URL do projeto Xchart. Infelizamente está além do escopo deste trabalho e do espaço disponível apresentar

detalhes desta especificação. Exemplos comentados de Xcharts também podem ser obtidos através do URL do projeto.

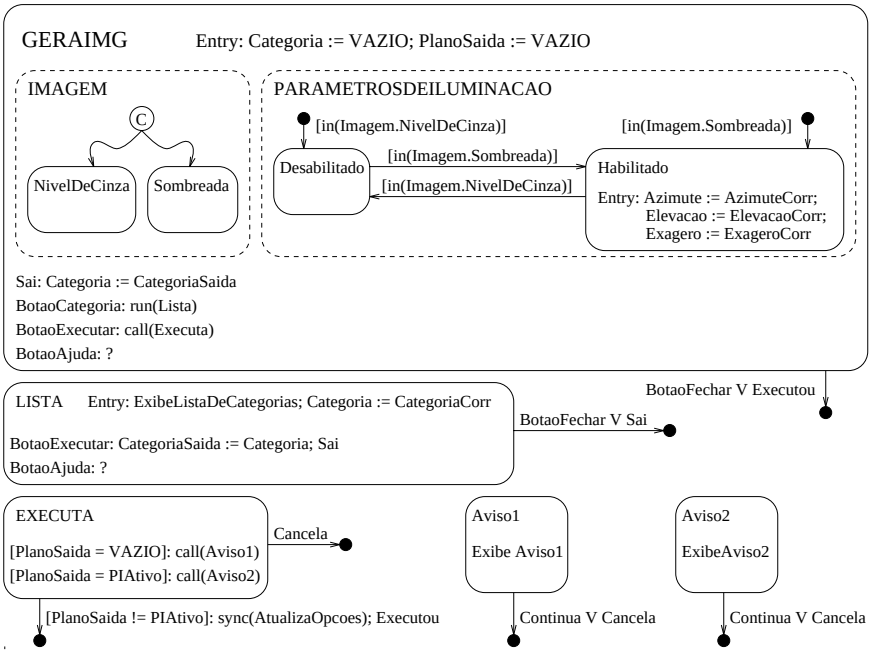

Figura 1: Diagramas em Xchart extraídos da especificação do SPRING.

4. ARQUITETURA DE SOFTWARE PARA SISTEMAS INTERATIVOS

Muitas arquiteturas de software têm sido propostas para sistemas interativos. Contudo, os objetivos conflitantes e compromissos, que devem ser estabelecidos na escolha de uma arquitetura, ainda não permitiram obter um consenso acerca de uma arquitetura ideal, que deveria atender requisitos não somente relacionados com o domínio de engenharia de software, mas também aqueles pertinentes à interação homem-computador [2].

No contexto de Xchart, um sistema interativo pode ser simples, isto é, centralizado e seqüencial, ou complexo, isto é, distribuído e concorrente. Um sistema interativo é estruturado em três subsistemas (Figura 2): (1) o componente de apresentação, que captura eventos gerados pelo usuário e exibe, para o usuário, resultados produzidos, além de implementar objetos de apresentação (por exemplo, reta e hachuras em um sistema para arquitetos); (2) a aplicação implementa a funcionalidade do sistema interativo, ou seja, conceitos dependentes de um domínio (por exemplo, parede e pintura correspondentes aos objetos de apresentação citados anteriormente); e (3) o gerenciador de diálogo, que implementa um protocolo de comunicação entre os outros componentes, converte conceitos da aplicação (por exemplo, temperatura) em objetos de apresentação (por exemplo, marcador de um termômetro) e vice-versa. O gerenciador de diálogo é responsável por servir de ligação entre estes dois componentes que fazem uso de elementos de naturezas distintas. De um lado temos objetos de apresentação (barras de rolagem, botões, apitos sonoros), de outro, temos conceitos (idade, sexo, ponte, estrada, perigo) e entre eles o gerenciador.

O gerenciador de diálogo é organizado em componentes menores (agentes), denominados objetos neste trabalho. Objeto é a unidade de implementação de interface no ambiente Xchart, trata-se de uma instância de um diagrama. O SGDI Xchart é responsável pela comunicação e sincronização entre estes objetos, que podem estar distribuídos, compartilhar memória e sinalizar a ocorrência de eventos. Este componente é composto, normalmente, por uma coleção de objetos que encapsulam e determinam, em conjunto, o comportamento da interface. Tipicamente, um objeto detecta requisições do usuário, através de objetos de apresentação

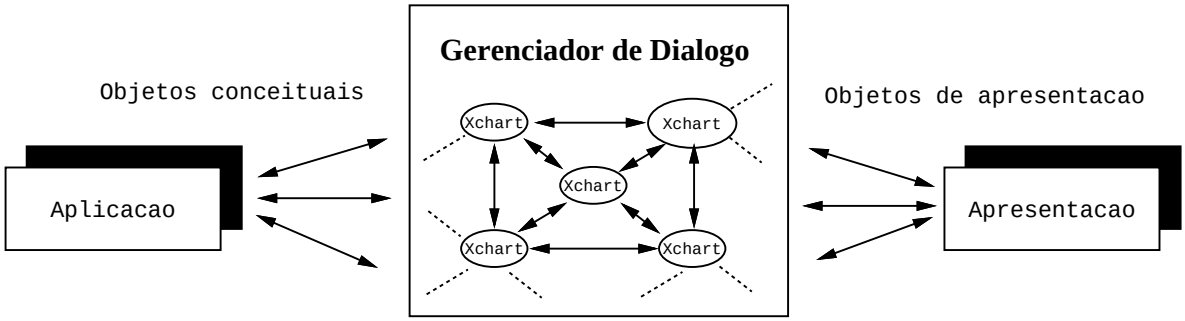

Figura 2: Modelo de um sistema interativo em Xchart.

providos por um *toolkit*, bem como aciona operações da aplicação, através de objetos conceituais, de acordo com o seu padrão de comportamento definido em Xchart.

Esta decomposição de um sistema interativo é bem diferente da criticada interpretação lingüística do modelo de Seeheim [6]. O modelo aqui apresentado é similar ao PAC-Amodeus [21], que é fruto de toda uma seqüência de evoluções de modelos. No presente modelo, contudo, os agentes (objetos) não necessariamente precisam formar uma hierarquia, ao contrário do anterior. A linguagem Xchart, comentada anteriormente, permite descrever estes objetos, a organização entre eles e como eles se comunicam com os demais componentes do modelo (apresentação e aplicação). O SGDI Xchart dá suporte a execução de especificações em Xchart. Isto inclui gerenciamento de memória compartilhada e eventos, que são os recursos disponíveis para comunicação entre interface e aplicação.

5. SISTEMA DE EXECUÇÃO DE XCHART

O Sistema de Execução (SE) fornece suporte à execução de Xcharts e estende uma versão simplificada e sem suporte a concorrência descrita em outro trabalho [15]. Nesta seção são identificados os componentes que fornecem a funcionalidade do SE de Xchart.

Objetos, dispersos sobre nós de uma rede, ativam/desativam, através de reações, outros objetos e geram eventos que podem provocar transições de estados em outros objetos. Cada objeto está associado a um único processo, que pode conter vários objetos. O SGDI Xchart fornece um substrato, a partir da qual estes objetos podem ser modelados e implementados. As funções do SE são desempenhadas pela cooperação entre o Núcleo Reativo (NR) e o Gerente de Distribuição (GD).

Núcleo Reativo (NR)

Interpreta especificações descritas em Xchart conforme a semântica desta linguagem. Este componente estabelece a reação de um objeto dado a ocorrência de um evento de acordo com a configuração corrente (conjunto de estados ativos) e outras informações de controle (mantidos todos na forma de atributos de instância) em que se encontra e o seu respectivo padrão de comportamento (mantido como atributo compartilhado a nível de classe). O NR normalmente faz uso dos serviços providos pelo GD (por exemplo, requisita um *broadcast* de um evento). A proposta de implementação do NR difere substancialmente daquela em [22]. O NR está sendo desenvolvido em C/C++, assume a existência de um ambiente operacional convencional (Windows NT e Unix, por exemplo) e funciona como uma máquina abstrata no mesmo sentido em que ESTEREL é implementada [3].

Gerente de Distribuição (GD)

Componente encarregado de fornecer suporte à concorrência: controle de concorrência, suporte à comunicação em grupo, localização de recursos, atomicidade e outros. Os serviços providos pelo GD ocultam a possível distribuição e concorrência entre objetos. Problemas envolvidos com a execução distribuída de Xcharts são comentados em [1].

Interface de Programação de Xchart (IPX)

A IPX é uma interface que oferece acesso aos recursos oferecidos pelo SE. Esta interface é utilizada tanto pelo código de apresentação quanto pelo código da aplicação de um sistema interativo. O programador destes componentes utiliza esta interface para criar instâncias de Xcharts (objetos) e sinalizar a ocorrência de eventos, por exemplo.

Smart — Editor Gráfico de Xcharts

Permite manipular descrições em Xchart. O usuário da linguagem Xchart interage com esta ferramenta para criar Xcharts, documentá-los, realizar análise de consistência das especificações criadas e experimentar uma especificação através de simulações.[2] O Smart gera automaticamente o leiaute de um Xchart dadas a hierarquia de estados e as transições entre eles e as regras.

Grace — Editor de Apresentação

Grace ou *interface builder* permite a definição interativa dos objetos de apresentação.

Estágio atual de desenvolvimento (implementação)

As ferramentas citadas acima encontram-se em desenvolvimento. Muitas delas estão divididas em subsistemas. Por exemplo, parte do GD responsável pela comunicação de baixo nível entre componentes do modelo Xchart já está operacional no ambiente Windows NT [1] assim como uma camada que utiliza estes recursos de baixo nível para prover atomicidade [5]. O Smart já realiza parte substancial das suas funções, contudo, a difícil tarefa de realizar o leiaute automático de Xcharts ainda exige mudanças no algoritmo empregado para esta finalidade.

Desenvolvimento usando Xchart

Várias atividades são realizadas ao longo do desenvolvimento de interfaces usando Xchart. O projeto de interação da interface, definição do "look and feel" da interface é apoiado pelo emprego das ferramentas Grace e Smart. Em tempo de execução, conforme comentado anteriormente, várias ferramentas oferecem o apoio necessário.

6. CONSIDERAÇÕES FINAIS

Idéias subjacentes ao SGDI Xchart foram apresentadas. Em particular, esta ferramenta dá ênfase ao desenvolvimento de interfaces concorrentes, comuns em sistemas de trabalho cooperativo, jogos, sistemas multimodais e outros. Contudo, poucas são as ferramentas existentes para auxiliar a implementação de tais interfaces e, em especial, aquelas existentes oferecem recursos de baixo nível de abstração. Xchart emprega uma abordagem distinta, onde uma linguagem de alto nível foi especificamente burilada para oferecer recursos de alto nível para descrever esta complexa parte do software de um sistema interativo que, em geral, é manualmente implementado por programadores.

Este trabalho abordada questões ainda em aberto e propõe um conjunto de funções para o SGDI Xchart, que é factível na arquitetura proposta e mais sofisticado do que aquele de sistemas como o SGI *Alpha* [13]. Esperamos que a aplicação desta proposta em sistemas reais e complexos (por exemplo, SPRING), ofereça subsídios para eventuais melhorias na linguagem Xchart e no ambiente de suporte.

Xchart não é apresentada como a solução definitiva para descrever a funcionalidade de gerenciadores de diálogo. Acreditamos que notações que refletem o ponto de vista do usuário (por exemplo, UAN [11]) são promissoras. No âmbito do projeto estamos investigando esta questão, assim como o emprego de Xchart para a especificação de controle complexo em geral.

[2] Simulação aqui deve ser entendia de forma mais restrita do que aquela em [4].

REFERÊNCIAS

[1] Edilmar L. Alves. *Port System:* Sistema de Comunicação em Grupo para o Ambiente Xchart. Master's thesis, DCC/IMECC/UNICAMP, Campinas/SP, Fevereiro 1996.

[2] Len Bass and Prasun Dewan, editors. *User Interface Software*. Trends in Software. John Wiley & Sons Ltd, 1993.

[3] Gérard Berry and Georges Gonthier. The ESTEREL Synchronous Programming Language: design, semantics, implementation. *Science of Computer Programming*, 19(2):87–152, November 1992.

[4] João W. L. Cangussu, Paulo C. Masiero, and José C. Maldonado. Execução Programada de Statecharts. *Revista Brasileira de Computação*, 7(2):3–14, Jun 1994.

[5] Luciana de Paula Brito. Suporte de Transação Atômica para a Linguagem Xchart. Master's thesis, DCC/IMECC/UNICAMP, Campinas/SP, 1996.

[6] Mark Green. Report on Dialogue Specification Tools. In Günther E. Pfaff, editor, *User Interface Management Systems*, pages 9–20. Springer-Verlag, 1985.

[7] Mark Green and Robert Jacob. SIGGRAPH'90 Workshop Report: Software Architectures and Metaphors for Non-WIMP User Interfaces. *Computer Graphics*, 25(3):229–235, July 1991.

[8] D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[9] Ralph D. Hill. Supporting Concurrency, Communication, and Synchronization in Human-Computer Interaction — The Sassafras UIMS. *ACM TOG*, 5(3):179–210, July 1986.

[10] Ralph D. Hill, Tom Brinck, Steven L. Rohall, John F. Patterson, and Wayne Wilner. The Rendezvous Architecture and Language for Constructing Multiuser Applications. *Transactions on Computer-Human Interaction*, 1(2):81–125, June 1994.

[11] Deborah Hix and H. Rex Hartson. *Developing User Interfaces: Ensuring Usability Through Product & Process*. John Wiles & Sons, Inc., 1993. ISBN 0471-57813-4.

[12] C. Huizing and W. P. de Roever. Introduction to Design Choices in the Semantics of Statecharts. *Information Processing Letters*, 37:205–213, 1991.

[13] Daniel Klein. Developing Applications with the Alpha UIMS. *Interactions*, 2(4):48–65, October 1995.

[14] Fábio N. Lucena, Mário Massato Harada, and Hans K.E. Liesenberg. Operational Semantics of Extended Statecharts (XCHARTS). *3rd Workshop on Logic, Language, Information and Computation (WoLLIC'96)*, 1996. Salvador/BA, May 8-10.

[15] Fábio N. Lucena and Hans K. E. Liesenberg. A Statechart Engine to Support Implementations of Complex Behaviour. *XXI Semish*, pages 177–191, 1994. Caxambu/MG, Brazil.

[16] Fábio N. Lucena and Hans K.E. Liesenberg. Reflections on Using Statecharts to Capture User Interface Behaviour. *Proceedings of XIV Int. Conf. of the Chilean CSS*, October 1994.

[17] Alan Morse and George Reynolds. Overcome Current Growth Limits in User Interface Development. *Communications of the ACM*, 36(4):73–81, April 1993.

[18] Brad A. Myers. Challenges of HCI Design and Implementation. *Interactions*, 1(1):73–83, 1994.

[19] William M. Newman. A System for Interactive Graphical Programming. *Proceedings of the Spring Joint Computer Conference*, pages 47–54, 1968.

[20] A. R. Puerta. A Model-Based Interface Development Environment. *IEEE Software*, 14(4):40–47, August 1997.

[21] D. Salber, L. Nigay, and Joëlle Coutaz. Extending the Scope of PAC-Amodeus to Cooperative Systems. *Proceedings of CSCW*, pages 22–26, October 1994.

[22] Rosemeire Shibuya, Rosângela D. Penteado, and Paulo César Masiero. Geração de Código a partir de Modelos Comportamentais Especificados por Statecharts. *VIII Simpósio Brasileiro de Engenharia de Software*, pages 253–267, October 1994.

[23] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, third edition, 1998.

[24] Pierre D. Wellner. Statemaster: A UIMS based on Statecharts for Prototyping and Target Implementation. In *Proceedings SIGCHI'89*, pages 177–182, April 1989.