Projeto Integrado de Pesquisa

Desenvolvimento Formal de Componentes de Software Reutilizáveis

Silvio Romero Lemos Meira
Alexandre Marcos Lins de Vasconcelos
Paulo Henrique Monteiro Borba

DEPARTAMENTO DE INFORMÁTICA
UFPE

Recife, Julho de 1996

Índice

1 Introdução

Técnicas de análise, projeto e programação orientadas a objetos são extremamente eficientes para reuso, manutenção e desenvolvimento incremental de software, como tem sido constatado com o desenvolvimento de aplicações industriais [1, 2]. Isso é decorrente do suporte a conceitos bem estabelecidos como encapsulamento, abstração de dados, subtipos, herança e "dynamic binding".

Por outro lado, a importância de métodos formais para desenvolvimento de software é significativamente reconhecida [3], sendo justificada pelo alto nível de confiabilidade dos sistemas desenvolvidos com linguagens de especificação oferecendo um alto nível de abstração, tendo uma semântica matemática bem definida, e permitindo provas formais de que projetos e implementações satisfazem (refinam) especificações [4].

Certamente, a integração entre métodos formais e técnicas de orientação a objetos oferece muitas vantagens [5]. Em particular, essa integração provê recursos essenciais para suportar a industrialização e comercialização de componentes de software—isto é, programas com interfaces bem definidas e padronizadas podendo ser facilmente agrupados a outros programas, de várias maneiras, para formar componentes mais complexos. Por um lado, técnicas de orientação a objetos favorecem o reuso, a manutenção e a integração de componentes. Por outro lado, métodos formais favorecem a confiabilidade dos componentes, o que é muito importante já que a utilização de componentes defeituosos em larga escala pode causar sérios danos.

Com o intuito de integrar métodos formais e orientação a objetos, a linguagem MooZ [6, 7] vem sendo desenvolvida no Departamento de Informática da UFPE (DI-UFPE) desde 1990. MooZ é baseada em Z [8, 9], uma linguagem de especificação formal de software que vem sendo desenvolvida na Universidade de Oxford desde o início da década de 80. MooZ estende Z com características de orientação a objetos, oferecendo melhor suporte linguístico para estruturação de sistemas. A motivação principal para essa extensão é facilitar o entendimento, desenvolvimento, reuso e manutenção de software.

Nesse projeto de pesquisa propomos atividades que estão diretamente relacionadas ao desenvolvimento de MooZ e ferramentas de suporte a esta linguagem. Com isso visamos contribuir para que MooZ possa ser usada com sucesso para desenvolver formalmente componentes de software de alta qualidade e que possam ser facilmente reutilizados.

Este documento está organizado da seguinte maneira. Primeiro indicamos a motivação para a realização do projeto. Em seguida, apresentamos a missão do projeto e os planos de atividades dos participantes (isto é, os pesquisadores responsáveis e os bolsistas), incluindo tarefas, metodologia, cronograma e equipe executora. Depois disso, abordamos a relevância, originalidade, viabilidade técnica e os resultados esperados do projeto. Também indicamos a contrapartida do DI-UFPE para a realização do projeto. No final apresentamos o orçamento detalhado do projeto e indicamos outras fontes de financiamento.

2 Motivação

Computadores estão cada vez mais sendo utilizados para auxiliar e realizar diversas atividades humanas, incluindo produção industrial, prestação de serviços, comércio, pesquisa e ensino. Por isso a qualidade de software é muito importante, principalmente quando o correto funcionamento de um sistema garante a segurança de pessoas e a disponibilidade de serviços essenciais à população. Isso é claramente indicado por programas nacionais como o SOFTEX 2000, o qual é realizado pelo governo juntamente com a iniciativa privada e visa transformar o Brasil em um centro produtor e exportador de software de qualidade. Certamente, o processo de desenvolvimento de software de qualidade tem uma importância estratégica para o Brasil.

Infelizmente, vários problemas têm atingido a indústria de software. São inúmeros prejuízos e frustrações causados por grandes projetos de desenvolvimento fracassados. Como relatado em [10], estudos mostram que cerca de um quarto dos projetos de desenvolvimento de grandes sistemas é cancelado antes da conclusão. Em média, o tempo de desenvolvimento é bem maior do que o estimado. Três quartos dos grandes sistemas não são usados ou não funcionam como planejado. Além disso, a manutenção e reutilização de software são, em geral, extremamente difíceis e custosas; às vezes é mais vantajoso desenvolver um novo sistema do que modificar, estender ou usar a funcionalidade de um já existente. Esses aspectos afetam diretamente a produtividade da indústria de software, contrastando com a indústria de hardware que produz e comercializa em larga escala componentes padronizados.

De fato, muitos sistemas são mal projetados, mal estruturados, mal documentados e não correspondem às expectativas dos usuários. A causa principal desses problemas é a má qualidade das teorias (linguagens), metodologias e ferramentas de desenvolvimento utilizadas. Por isso, existe atualmente uma grande demanda social para a utilização de abordagens que visem o desenvolvimento de software de alta qualidade. Com base na experiência da indústria de hardware e outras engenharias, uma abordagem promissora é o desenvolvimento formal de componentes de software que sejam facilmente reutilizáveis [11].

Com a produção e reutilização de componentes de alta confiabilidade, a indústria de software pode melhorar consideravelmente a qualidade de seus produtos, além de reduzir os custos e o tempo de desenvolvimento e manutenção. Consequentemente, a competitividade das empresas que desenvolvem software aumentará, o que será importante para a sobrevivência das mesmas em um mercado cada vez mais globalizado e competitivo. Da mesma forma, as empresas que consumirem software de alta qualidade terão chances maiores de manter e conquistar mercados. Em geral, a sociedade poderá dispor de melhores produtos a um menor custo.

3 Missão

A missão desse projeto de pesquisa é contribuir para melhorar os processos de desenvolvimento, evolução e disponibilização de software de alta qualidade. Em particular, esperamos que os resultados do projeto ajudem a aumentar consideravelmente a confiabilidade, extensibilidade, adaptabilidade e reusabilidade de software. Por isso, pretendemos desenvolver novas teorias, metodologias e ferramentas para facilitar o desenvolvimento, reuso e manutenção de componentes de software altamente confiáveis e que possam ser facilmente reutilizados.

4 Plano de Atividades dos Pesquisadores e Bolsistas APB

As atividades de pesquisa deste projeto visam realizar três objetivos principais:

- Desenvolver uma ferramenta de apoio ao desenvolvimento de especificações formais escritas em MooZ.
- Desenvolver um sistema genérico de interconexão de módulos (componentes) que possibilite interconectar módulos de várias maneiras para formar módulos mais complexos. Em particular, queremos instanciar esse sistema para MooZ.
- Desenvolver a base de uma teoria de refinamento para MooZ.

Assim dividimos esta seção em três partes, detalhando os objetivos acima separadamente. Em cada seção encontra-se os planos de atividades dos Bolsistas APB e dos pesquisadores responsáveis pelo projeto.

4.1 Ferramenta para Desenvolvimento de Especificações em MooZ

Esta parte do projeto tem como principal objetivo estender a funcionalidade da ferramenta ForMooZ, implementada a nível de protótipo [29], de modo a criar um sistema integrado para o desenvolvimento de especificações escritas em linguagens de especificação formal com semântica relacionada à MooZ. Tal ambiente será suportado por uma máquina de hipertexto de propósito geral.

Parte do desenvolvimento da ferramenta ForMooZ já se encontra em andamento e está prevista para o início do primeiro semestre de 97 a disponibilização de um protótipo inicial em Eiffel[12] contendo suas características mais básicas, que são:

Interface com um SGBD-OO Comercial

Dentre os SGBD's comerciais disponíveis no DI-UFPE, foi escolhido um sistema que melhor se adequa à aplicação, ou seja, um SGBD orientado a objetos. Desta forma, o sistema ObjectStore foi escolhido e está sendo desenvolvida uma interface de comunicação entre ObjectStore e ForMooZ.

Máquina de Hipertexto Genérica

A máquina de hipertexto utiliza diretamente a interface de comunicação com o SGBD descrito anteriormente, e é a principal responsável pela integração dos diversos módulos de ForMooZ. O modelo de dados do hipertexto nesta nova versão de ForMooZ se baseia no modelo de referência Dexter [13] e na arquitetura do modelo Proxhy [14], utilizando conceitos de orientação a objetos (classes representando nós, *links* e âncoras do hipertexto) e mecanismos de comunicação (métodos destas classes são acessados a partir de uma interface básica de comunicação com a máquina de hipertexto). Esta máquina está sendo implementada em Eiffel de forma o mais independente possível dos outros módulos de ForMooZ, com o objetivo de facilitar a integração futura com outras ferramentas a partir da estrutura de hipertexto.

Compilador Batch

A fim de tornar o *compilador*[1] disponível o mais rápido possível, optamos por fazer sua implementação inicial de modo que especificações fossem processadas de forma *batch*.

4.1.1 Tarefas e Metodologia

Para estender o ambiente ForMooZ realizaremos as tarefas descritas a seguir. Estas tarefas serão efetuadas sobre o protótipo de ForMooZ desenvolvido em Eiffel.

[1] Neste contexto denominaremos de compilação as operações de verificação léxica e sintática, seguida de verificação de escopo e de tipos; ou seja, não existe a geração de código executável.

Tarefa FORM.1 — Explorar a Utilização do Protótipo Desenvolvido em Eiffel

Para explorar a utilização de ForMooZ serão realizados testes com alunos da disciplina de Especificação Formal ministrada no DI-UFPE durante o primeiro semestre de 1997. Para isto, serão incluídos na ferramenta estudos de casos de especificações desenvolvidas em MooZ. A partir do *feedback* destes usuários, serão realizadas prováveis correções no protótipo, sendo possível a validação das tarefas desenvolvidas até então.

Tarefa FORM.2 — Extensão da Máquina de Hipertexto

Esta tarefa prevê a extensão da interface de comunicação entre o hipertexto e o SGBD utilizado para permitir o uso de serviços mais avançados providos pelo SGBD, tais como o controle de versões, o uso de consultas, etc. Considerando esta extensão, serão então implementados nós com versões e criadas facilidades mais sofisticadas para navegação e recuperação de informação, utilizando a estrutura de hipertexto. Além disto, o hipertexto deve ser estendido para manipular, além de especificações formais, alguns dos documentos gerados por outras ferramentas. Será também estudada a possibilidade das interfaces de comunicação com o hipertexto seguirem a filosofia cliente-servidor.

Tarefa FORM.3 — Extensão da Interface com o Usuário

Com relação à interface, as extensões previstas estão relacionadas ao *feedback* dos usuários a partir dos testes de usabilidade realizados na Tarefa FORM.1 e às novas características a serem adicionadas ao ambiente a partir das demais tarefas. Para isto, será necessário um estudo para definir como melhor representar estas novas características na interface, onde novos testes de usabilidade, discussões entre a equipe e prototipagem das possíveis representações deverão ser realizados.

Tarefa FORM.4 — Desenvolvimento de um Compilador Incremental para MooZ

O mecanismo de compilação *batch* implica que, ao ser modificada uma especificação, será necessária a completa recompilação da mesma. Entretanto, a recompilação de todas as definições pode ser uma operação muito cara.

A eficiência do mecanismo de compilação pode ser melhorada através do uso de técnicas de compilação incremental semelhantes às desenvolvidas pelo Dr. Vasconcelos[30] num ambiente incremental de processamento de especificações escritas em Z.

Para que as técnicas extensivamente usadas pelo Dr. Vasconcelos possam ser aplicadas em ForMooZ, as mesmas devem ser adaptadas para a linguagem MooZ. Para tanto será necessário investigar quais os possíveis tipos de relacionamentos implícitos existentes entre trechos de especificações em MooZ (ex: uso de uma variável com sua declaração), bem como decidir qual o tratamento que será dado aos relacionamentos externos à especificação sendo compilada.

Tarefa FORM.5 — Transformação de ForMooZ em um Ambiente Instanciável

Sendo MooZ uma extensão de Z é natural pensar que o sistema ForMooZ deverá ser transformado em um sistema instanciável, capaz de dar suporte à construção de especificações em ambas linguagens. Em [15] é descrita uma proposta preliminar para uma semântica formal para MooZ. Este trabalho demonstra que Z pode ser utilizada para descrever a semântica de MooZ e portanto é possível a implementação de um conversor sintático de especificações MooZ em especificações Z.

A idéia de termos um ambiente instanciável tanto para Z quanto para MooZ pode ser estendida a longo prazo, de modo a termos um ambiente genérico ForX, onde X é uma linguagem de especificação formal baseada em modelos com uma semântica relacionada à de MooZ.

Tarefa FORM.6 — Disponibilização de ForMooZ através de Servidor WWW

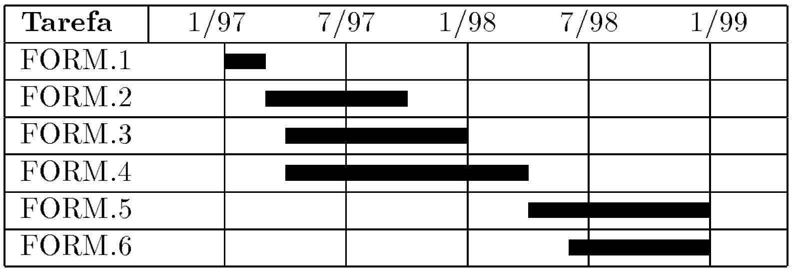
Está previsto também o estudo da possibilidade de implementar a interface de ForMooZ em HTML e/ou Java tornando possível a sua disponibilização através do servidor WWW do Departamento de Informática, permitindo assim a utilização remota de algumas funcionalidades de ForMooZ sem que o usuário tenha que obter a ferramenta (código executável) e instalá-la em seu ambiente.

4.1.2 Cronograma e Equipe Executora

A tabela a seguir mostra o esforço em homem-mês necessário para a realização das tarefas de extensão associadas ao desenvolvimento de ForMooZ. As tarefas serão realizadas no DI-UFPE, pelos Drs. Vasconcelos e Meira, por um bolsista APB e por quatro bolsistas de iniciação científica.

Tarefa	**Homem-mês**
FORM.1—Explorar a Utilização do Protótipo	2
FORM.2—Extensão da Máquina de Hipertexto	7
FORM.3—Extensão da Interface com o Usuário	6
FORM.4—Desenvolvimento do Compilador Incremental	5
FORM.5—Transformação em um Ambiente Instanciável	4
FORM.6—Disponibilização via WWW	8

As tarefas descritas acima serão realizadas de acordo com o seguinte cronograma.



4.2 Sistema de Interconexão de Módulos (SIM)

Com o desenvolvimento do sistema genérico de interconexão de módulos (SIM) visamos oferecer recursos lingüísticos e computacionais para estruturar e conectar os diversos documentos (isto é, requisitos, diagramas, especificações, programas, documentação, manuais, etc.) gerados durante o desenvolvimento de grandes sistemas. Os diferentes tipos de conexões entre os documentos serão armazenados como diferentes tipos de *links* do sistema de Hipertexto provido por ForMooZ, auxiliando e guiando a navegação dos documentos de um sistema.

Esse sistema será independente da linguagem utilizada para escrever o conteúdo de cada módulo. Dessa forma, será possível gerar facilmente sistemas de interconexão de módulos para as diversas linguagens utilizadas nas diferentes etapas do processo de desenvolvimento de software. Cada sistema gerado interconectará módulos contendo um tipo específico de documento; por exemplo, especificações em MooZ, programas em Java [16], diagramas em uma notação gráfica ou manuais em linguagem natural.

Para explorar a utilização do SIM, iremos utilizá-lo para gerar um sistema de interconexão de módulos para MooZ.

4.2.1 Tarefas e Metodologia

Para desenvolver o sistema de interconexão de módulos realizaremos as tarefas descritas a seguir. Descrevemos brevemente cada tarefa e a metodologia a ser utilizada para executá-las.

Tarefa SIM.1—Definir a Linguagem do SIM

Primeiro iremos definir informalmente a linguagem de interconexão de módulos que será suportada pelo sistema. Essencialmente, teremos que decidir que operações serão disponíveis para compor módulos com o objetivo de formar novos módulos, e transformar um módulo em outro. A escolha dessas operações indicará os requisitos básicos do sistema de interconexão de módulos.

Para realizar esta tarefa, nos basearemos nos elegantes projetos das linguagens CLEAR [17], OBJ [18], e LIL [19]. Nestas linguagens, um módulo agrupa a definição de vários conceitos relacionados e várias "interfaces" podem ser associadas a um mesmo módulo. Além disso, consideraremos as seguintes características:

- módulos genéricos (parametrizados) que podem ser instanciados por outros módulos, resultando em um novo módulo que oferece uma funcionalidade dependente da funcionalidade daquele usado para instanciação;
- operações para composição de módulos: união, importação e instanciação;
- operações para transformação de módulos: mudar o nome de funções e operações (métodos, atributos, etc.), esconder operações e tipos, e eliminar a definição de operações e tipos;
- distinção entre composição horizontal (i.e., o agrupamento de módulos para especificar um sistema) e composição vertical (i.e, o uso de um módulo para especificar outro módulo).

Com base na experiência da equipe encarregada de desenvolver o SIM, técnicas de reuso e estruturação baseadas nestas características são bem mais fundamentais, documentadas e eficazes do que as técnicas baseadas em classes, funções de alta ordem e polimorfismo encontrados em linguagens funcionais. Por isso justifica-se suportar tanto módulos quanto classes em uma linguagem como MooZ.

Tarefa SIM.2—Definir a Metodologia de Utilização do SIM

Uma vez definida a linguagem do SIM, iremos definir a metodologia de utilização desse sistema; isto é, iremos indicar como o SIM pode ser utilizado para estruturar os diversos documentos gerados durante o processo de desenvolvimento de software.

Para realizar esta tarefa, iremos nos basear na experiência do Dr. Borba com a utilização das linguagens de interconexão de módulos de OBJ para desenvolver aplicações reais de pequeno e médio porte. Para cada tipo de documento, apresentaremos um método de utilização específico.

Tarefa SIM.3—Especificar e Projetar o SIM

Com base em resultados sendo desenvolvidos para definir uma semântica concreta para um subconjunto da linguagem do SIM, derivaremos a primeira especificação do SIM em MooZ. Essa parte da tarefa será facilitada pelo fato de tanto MooZ quanto a semântica concreta da linguagem do SIM serem baseadas em Teoria dos Conjuntos. A partir dessa especificação, por um processo rigoroso de refinamento passo-a-passo, derivaremos várias especificações do sistema até obter uma especificação (projeto) que seja concreta o suficiente para ser diretamente implementada de forma eficiente em Java.

Para verificar a consistência (de sintaxe, tipos, etc.) das especificações, usaremos um protótipo de ForMooZ atualmente sendo desenvolvido no DI-UFPE. Para estruturar as diversas especificações do SIM, usaremos a linguagem do próprio SIM, ajudando assim a validar a proposta dessa linguagem. Em particular, usando módulos genéricos, obteremos especificações parametrizadas pelas características da linguagem utilizada para escrever os módulos. Instanciando uma dessas especificações, obteremos uma especificação de um sistema de interconexão de módulos para uma linguagem específica.

Tarefa SIM.4—Gerar Protótipos e Implementar o SIM

Derivaremos a implementação e protótipos do SIM a partir, respectivamente, do projeto final e de algumas especificações resultantes da Tarefa SIM.3. Utilizaremos os protótipos para validar as especificações. O SIM e os protótipos serão escritos em Java, independentemente do hardware e sistema operacional utilizados. Entretanto, para utilizar e validar o SIM, nesse projeto usaremos implementações baseadas na plataforma Windows-NT/IBM PC.

Obteremos um sistema de interconexão de módulos independente da linguagem utilizada para escrever os módulos. Instanciando esse sistema genérico, será possível obter sistemas de interconexão de módulos para diversas linguagens.

Tarefa SIM.5—Projetar e Implementar a Interface do SIM

Depois que a linguagem do SIM for especificada, projetaremos e implementaremos a interface desse sistema. Para garantir o impacto social do SIM, é essencial que esse sistema seja eficiente e fácil de usar, tendo uma interface gráfica multimídia amigável e intuitiva, com documentação e suporte acessíveis. Além disso, é importante que o SIM possa ser usado via WWW.

Para implementação da interface, usaremos e estenderemos classes pré-definidas das bibliotecas de Java, permitindo assim que o sistema também seja utilizado via Internet/WWW. Também implementaremos o acesso uniforme via WWW a diversos repositórios de módulos gerados pelo SIM. Essa interface será desenvolvida independentemente das partes principais do sistema; só na fase final de desenvolvimento do SIM é que a interface será integrada às outras partes do sistema.

Tarefa SIM.6—Explorar a Utilização do SIM

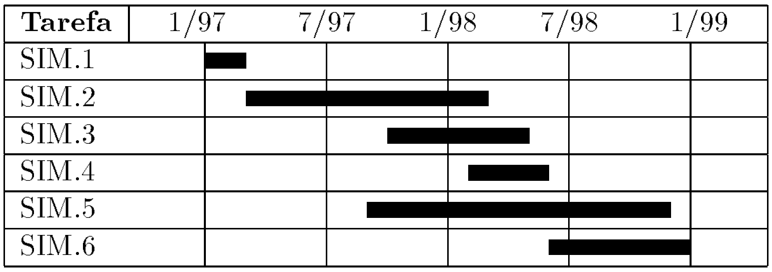
Para explorar a utilização do SIM, primeiro iremos utilizá-lo para gerar um sistema de interconexão de módulos para MooZ. Esse sistema será integrado a ForMooZ. Em seguida, iremos usar o sistema gerado para verificar automaticamente a consistência da estruturação das especificações do SIM em MooZ, resultantes da Tarefa SIM.3; assim estaremos validando tanto o sistema gerado quanto o SIM. Além disso, iremos utilizar o sistema de módulos para MooZ como suporte para desenvolver uma aplicação de engenharia reversa: derivar projetos estruturados (usando a linguagem do SIM) a partir de programas escritos em Java.

4.2.2 Cronograma e Equipe Executora

A tabela a seguir mostra o esforço estimado em homem-mês necessário para a realização das tarefas associadas ao desenvolvimento do SIM. As tarefas serão realizadas pelo Prof. Meira, Dr. Borba, por um bolsista APB e por quatro bolsistas de iniciação científica.

Tarefa	**Homem-mês**
SIM.1—Definir a Linguagem do SIM	2
SIM.2—Definir a Metodologia de Utilização do SIM	5
SIM.3—Especificar e Projetar o SIM	7
SIM.4—Gerar Protótipos e Implementar o SIM	2
SIM.5—Projetar e Implementar a Interface do SIM	8
SIM.6—Explorar a Utilização do SIM	5

As tarefas descritas acima serão realizadas de acordo com o seguinte cronograma.



4.3 Refinamento para MooZ

Com o desenvolvimento da base de uma teoria de refinamento para MooZ visamos oferecer o suporte mínimo para desenvolvimento formal, passo-a-passo, de software em MooZ. Em particular, essa base oferece recursos para provar-se formalmente, ou até mesmo verificar-se rigorosamente, que uma especificação concreta satisfaz (refina) uma especificação mais abstrata.

4.3.1 Tarefas e Metodologia

As principais tarefas associadas ao desenvolvimento da base de uma teoria de refinamento para MooZ são descritas a seguir, juntamente com as estratégias a serem utilizadas para executá-las.

Tarefa REF.1—Definir Noção de Refinamento

Primeiro definiremos formalmente a noção de que uma especificação em MooZ refina outra especificação em MooZ. Em termos gerais, uma especificação S é refinada por uma especificação T se T preserva as propriedades de S e (possivelmente) tem propriedades adicionais; dessa forma, podemos substituir S por T sem perda de significado.

Para definir a noção de refinamento, usaremos algumas idéias de [20] e [21], onde apresenta-se a base de uma teoria de refinamento para FOOPS [22, 23]—uma linguagem de especificação e programação orientada a objetos. Escolheremos um subconjunto significativo de MooZ, visando facilitar a definição da noção de refinamento. Essa definição será baseada no modelo matemático adotado para dar a semântica de MooZ. Como em [20], daremos noções (relações) de refinamento de estados, expressões e especificações.

Tarefa REF.2—Derivar Propriedades de Refinamento

Com base na formalização da definição de refinamento gerada pela Tarefa REF.1, iremos propor técnicas eficientes para provar formalmente que uma especificação é refinada por outra. Além disso, estabeleceremos propriedades da relação de refinamento, incluindo reflexividade, transitividade e congruência. Esses aspectos são essenciais para verificar refinamento em aplicações complexas, além de justificar o desenvolvimento de software passo-a-passo e por partes (que são compostas para formar o todo).

Para descobrir as técnicas de prova e as propriedades de refinamento consideraremos alguns resultados apresentados em [21], mas também tentaremos novas abordagens e a derivação de propriedades adicionais.

Tarefa REF.3—Explorar Noção e Propriedades de Refinamento

Usando a noção de refinamento, as técnicas de prova e as propriedades de refinamento desenvolveremos formalmente diversas aplicações de pequeno e médio porte para ilustrar e analisar a adequabilidade da nossa abordagem. Em particular, desenvolveremos alguns exemplos simples (como buffers e protocolos de comunicação) e partes do sistema de inteconexão de módulos proposto neste projeto.

Tarefa REF.4—Investigar Relação entre MooZ e Java

Por fim, investigaremos a relação entre a semântica de MooZ e a semântica de uma linguagem de programação orientada a objetos (provavelmente Java [16] ou Eiffel [12]). Esse é o primeiro passo para formalizar a noção de refinamento de especificações em MooZ para programas nesta linguagem.

Idealmente, a relação entre a semântica de MooZ e a semântica da linguagem de programação será indicada por um mapeamento entre os modelos semânticos das duas linguagens. Caso nenhum dos dois modelos seja suficientemente expressivo para dar significado às estruturas existentes nas duas linguagens[2], teremos que propor um modelo mais expressivo. Nesse caso, a relação entre as duas semânticas será indicada por mapeamentos dos modelos das linguagens para o modelo mais expressivo.

Uma vez estabelecidos os mapeamentos entre os modelos, a noção de refinamento de especificações em MooZ para programas na outra linguagem é dada pela noção de refinamento associada ao modelo que é o codomínio dos mapeamentos. Por isso é importante entender isoladamente as noções de refinamento de MooZ e da linguagem de programação antes de definir a noção de refinamento de uma linguagem para a outra.

4.3.2 Cronograma e Equipe Executora

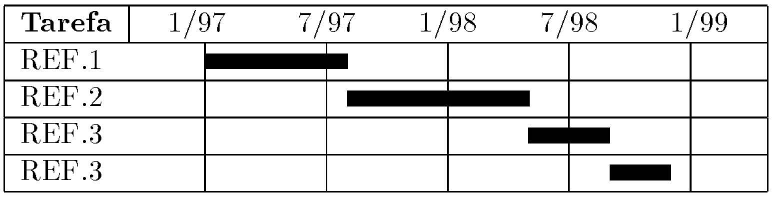
A tabela a seguir mostra o esforço estimado em homem-mês necessário para a realização das tarefas associadas ao desenvolvimento da base de uma teoria de refinamento para MooZ. Todas

[2]Provavelmente, o modelo semântico de MooZ não é concreto o suficiente para expressar a semântica de algumas características encontradas em linguagens orientadas a objetos, incluindo objetos com identificadores, compartilhamento de objetos, concorrência, e criação e remoção dinâmica de objetos.

as tarefas serão realizadas pelo Prof. Meira e pelo Dr. Borba.

Tarefa	**Homem-mês**
REF.1—Definir Noção de Refinamento	4
REF.2—Derivar Propriedades de Refinamento	5
REF.3—Explorar Noção e Propriedades de Refinamento	2
REF.4—Investigar Relação entre MooZ e Java	2

Essas tarefas serão realizadas de acordo com o seguinte cronograma.



5 Plano de Atividades dos Bolsistas de Iniciação Científica

Este projeto de pesquisa conta com a participação de oito bolsistas de iniciação científica. Essa participação será essencial para o desenvolvimento do projeto, trazendo várias vantagens tanto aos bolsistas quanto aos pesquisadores responsáveis pelo projeto. Através de reuniões, os pesquisadores responsáveis introduzirão aos bolsistas conhecimentos básicos para o desenvolvimento do projeto. Depois disso os bolsistas auxiliarão na realização de algumas das tarefas apresentadas na Seção 4. A seguir detalhamos as atividades que os bolsistas realizarão.

5.1 Utilização do SIM via WWW

Dois bolsistas participarão da execução da Tarefa SIM.5—projeto e implementação da interface do SIM—concentrando-se na utilização do SIM via WWW, incluindo o acesso remoto aos repositórios de módulos do SIM. Esses bolsistas já deverão ter familiaridade com conceitos de orientação a objetos.

5.1.1 Tarefas e Metodologia

Os bolsistas realizarão as seguintes (sub)tarefas associadas à execução da Tarefa SIM.5.

Tarefa IC1.1—Estudar Java e HTML

Para implementar o acesso aos repositórios de módulos do SIM via WWW, os bolsistas precisarão estudar em detalhe as linguagens Java [16] e HTML [24], além de padrões para acesso a banco de dados via WWW. Esse estudo será dirigido e acompanhado pelo Dr. Borba através de leituras específicas, prática em laboratório e reuniões semanais com os bolsistas.

Tarefa IC1.2—Projetar a Interface do SIM

Com base nos estudos da Tarefa IC1.1, o Prof. Meira, o Dr. Borba e os bolsistas projetarão informalmente a interface do SIM, no que concerne o acesso aos repositórios de módulos via WWW. Como resultado, será elaborado um documento descrevendo o projeto desta parte da interface.

Tarefa IC1.3—Estudar e Estender Bibliotecas de Java

Antes de implementar o acesso aos repositórios de módulos, os bolsistas analisarão e estenderão as biblioteca pré-definidas de Java que oferecem suporte a recursos gráficos e interação com Internet/WWW. A extensão dessas bibliotecas será fundamental tanto para auxiliar na implementação quanto para que os bolsistas tenham maior experiência prática com Java. Essa tarefa será dirigida e acompanhada pelo Dr. Borba através de reuniões quinzenais com os bolsistas.

Tarefa IC1.4—Implementar a Interface do SIM

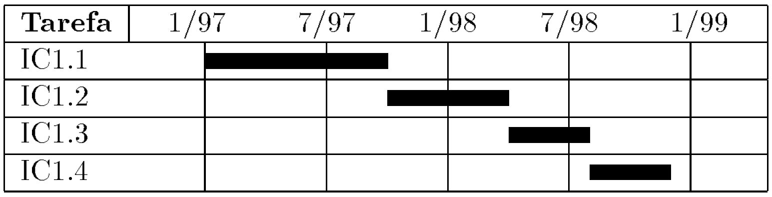
Para implementação do acesso aos repositórios de módulos do SIM, os bolsistas usarão as extensões das biblioteca de Java resultante da Tarefa IC1.3. Esse acesso será desenvolvido independentemente das partes principais do sistema; só na fase final de desenvolvimento do SIM é que o acesso será integrado às outras partes do sistema. Essa tarefa será acompanhada pelo Prof. Meira e pelo Dr. Borba através de reuniões quinzenais com o bolsista.

5.1.2 Cronograma e Equipe Executora

A tabela a seguir mostra o esforço estimado em homem-mês necessário para a realização das tarefas descritas na seção anterior. Note que as tarefas serão realizadas pelos bolsistas com orientação e auxílio do Prof. Meira e do Dr. Borba.

Tarefa	Homem-mês
IC1.1—Estudar Java e HTML	6
IC1.2—Projetar a Interface do SIM	4
IC1.3—Estudar e Estender Bibliotecas de Java	2
IC1.4—Implementar a Interface do SIM	2

As tarefas descritas acima serão realizadas de acordo com o seguinte cronograma.



5.2 Desenvolvimento do SIM

Dois bolsistas participarão da execução das tarefas SIM.3, SIM.4, e SIM.6 (veja Secão 4.2), correspondendo respectivamente a especificação, implementação e exploração do SIM. Esses bolsistas já deverão ter familiaridade com conceitos de orientação a objetos.

5.2.1 Tarefas e Metodologia

As principais atividades a serem realizadas pelos bolsistas são descritas a seguir, juntamente com as metodologias a serem utilizadas.

Tarefa IC2.1—Estudar OBJ e MooZ

Para desenvolver o SIM, os bolsistas precisarão estudar em detalhe o sistema de módulos de OBJ [18], aspectos de "parameterized programming" [25], MooZ e a linguagem resultante da Tarefa SIM.1 (veja Seção 4.2.1). Esse estudo será dirigido e acompanhado pelo Dr. Borba através de leituras específicas e reuniões semanais com os bolsistas.

Tarefa IC2.2—Especificar e Projetar o SIM

Apoio na realização da Tarefa SIM.3 (veja Seção 4.2.1). A participação dos bolsistas será acompanhada pelo Prof. Meira e pelo Dr. Borba através de reuniões semanais com os bolsistas.

Tarefa IC2.3—Gerar Protótipos e Implementar o SIM

Apoio na realização da Tarefa SIM.3 (veja Seção 4.2.1). A participação dos bolsistas será acompanhada pelo Dr. Borba através de reuniões quinzenais com os bolsistas.

Tarefa IC2.4—Explorar a Utilização do SIM

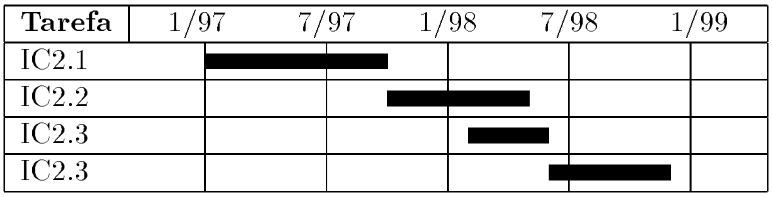
Apoio na realização da Tarefa SIM.3 (veja Seção 4.2.1). A participação dos bolsistas será acompanhada pelo Prof. Meira e pelo Dr. Borba através de reuniões quinzenais com os bolsistas.

5.2.2 Cronograma e Equipe Executora

A tabela a seguir mostra o esforço estimado em homem-mês necessário para a realização das tarefas apresentadas na seção anterior. Note que as tarefas serão realizadas pelos bolsistas com orientação e auxílio do Prof. Meira e do Dr. Borba.

Tarefa	**Homem-mês**
IC2.1—Estudar OBJ e MooZ	5
IC2.2—Especificar e Projetar o SIM	4
IC2.3—Gerar Protótipos e Implementar o SIM	2
IC2.4—Explorar a Utilização do SIM	3

As tarefas descritas acima serão realizadas de acordo com o seguinte cronograma.



5.3 Construção da Interface de ForMooZ

Dois bolsistas de IC participarão da execução da tarefa FORM.3 – Extensão da Interface com o Usuário de ForMooZ. Esses bolsistas já deverão ter familiaridade com conceitos de Engenharia de Software e de orientação a objetos.

5.3.1 Tarefas e Metodologia

Para que os bolsistas possam dar uma contribuição efetiva na construção da interface de ForMooZ, as seguintes (sub)tarefas associadas à execução da tarefa FORM.3 deverão ser desenvolvidas.

Tarefa IC3.1 – Análise da proposta para a interface de ForMooZ

Para a definição de um modelo de interface para ForMooZ, os bolsistas deverão estudar princípios de interação com ambientes e ferramentas CASE, a partir do trabalho apresentado em [31]. Neste trabalho, foi também proposto um modelo para a interface de ForMooZ, que deverá ser analisado e adaptado à nova versão do sistema, como descrito na tarefa IC3.2.

Tarefa IC3.2 – Definição da interface de ForMooZ

Com base nos estudos desenvolvidos em IC3.1, os bolsistas deverão participar da definição de uma interface gráfica para o ambiente ForMooZ.

Tarefa IC3.3 – Estudo das ferramentas para construção de interfaces

Inicialmente, os bolsistas irão estudar e praticar o uso da linguagem Eiffel. A seguir, deverão investigar a viabilidade de uso da ferramenta para geração de interfaces, DevGuide[32], assim como a sua integração com a linguagem Eiffel.

Tarefa IC3.4 – Implementação da interface de ForMooZ

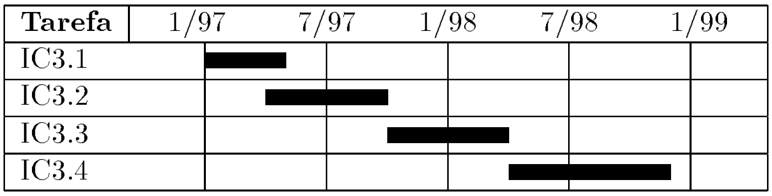
A partir das ferramentas estudadas na tarefa IC3.3, os bolsistas auxiliarão na implementação do modelo final para a interface de ForMooZ.

5.3.2 Cronograma e Equipe Executora

A tabela a seguir mostra o esforço estimado em homem-mês necessário para a realização das tarefas descritas na seção anterior. Note que as tarefas serão realizadas pelos bolsistas de IC, no DI-UFPE, com orientaçãoe auxílio dos Drs. Vasconcelos e Meira e de um bolsista APB.

Tarefa	Homem-mês
IC3.1 — Análise da proposta para a interface de ForMooZ	2
IC3.2 — Definição da interface de ForMooZ	3.5
IC3.3 — Estudo das ferramentas para construção de interfaces	2
IC3.4 — Implementação da interface de ForMooZ	3

As tarefas acima serão realizadas de acordo com o seguinte cronograma.



5.4 Desenvolvimento de um Compilador Incremental para MooZ

Dois bolsistas de IC participarão da execução da tarefa FORM.4 – Desenvolvimento de um Compilador Incremental para MooZ. Esses bolsistas já deverão ter familiaridade com conceitos de Engenharia de Software e de orientação a objetos.

5.4.1 Tarefas e Metodologia

Para que os bolsistas possam dar uma contribuição efetiva no desenvolvimento de um compilador incremental para MooZ, as seguintes (sub)tarefas associadas à execução da tarefa FORM.4 deverão ser desenvolvidas.

Tarefa IC4.1 – Estudar técnicas de compilação incremental

A fim de entender em detalhes os conceitos de compilação incremental, os bolsistas terão a necessidade de estudar as diversas técnicas existentes na literatura, as quais são detalhadas no trabalho apresentado em [30]. Neste trabalho, foi também descrita uma técnica de compilação incremental para a lingaugem Z. Esta técnica poderá vir a ser adaptada para manipular especificações escritas em MooZ.

Tarefa IC4.2 – Analisar os relacionamentos implícitos em especificações MooZ

De modo a adaptar a técnica de compilação incremental descrita em [30], será necessário analisar quais os possíveis tipos de relacionamentos implícitos (gerados automaticamente pelo sistema) existentes em especificações escritas em MooZ (ex: uso de uma variável e sua declaração, etc.). Para esta análise serão examinados os diversos estudos de caso de especificações MooZ existentes, bem como serão propostos novos estudos de caso.

Tarefa IC4.3 – Projetar o compilador incremental

O compilador atual de ForMooZ está sendo implementado de forma batch. A fim de transformá-lo em um compilador incremental, será necessário que os bolsistas entendam em detalhes a sua implementação.

Com base no compilador batch e nos tipos de relacionamentos implícitos encontrados em IC4.2, os bolsistas auxiliarão no projeto do compilador incremental para MooZ.

Tarefa IC4.4 – Estudar a linguagem Eiffel

Sendo Eiffel a principal linguagem de implementação de ForMooZ, caberá aos bolsistas dominar a linguagem antes de partirem para a implementação do compilador incremental proposto.

Tarefa IC4.5 – Implementar o compilador incremental

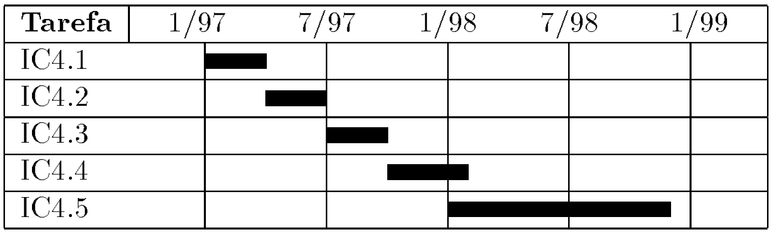
Com base no projeto proposto para o compilador incremental, caberá aos bolsistas auxiliar na implementação do mesmo.

5.4.2 Cronograma e Equipe Executora

A tabela a seguir mostra o esforço estimado em homem-mês necessário para a realização das tarefas descritas na seção anterior. Note que as tarefas serão realizadas pelos bolsistas de IC, no DI-UFPE, com orientação e auxílio dos Drs. Vasconcelos e Meira e de um bolsista APB.

Tarefa	**Homem-mês**
IC4.1 — Estudar técnicas de compilação incremental	2
IC4.2 — Analisar os relacionamentos implícitos	3
IC4.3 — Projetar o compilador incremental	3
IC4.4 — Estudar a linguagem Eiffel	2
IC4.5 — Implementar o compilador incremental	4

As tarefas acima serão realizadas de acordo com o seguinte cronograma.



6 Resultados Esperados

Os principais resultados a serem obtidos com o desenvolvimento do projeto são apresentados na tabela a seguir. Além de dar a descrição e o tipo de cada resultado, indicamos a data na qual o resultado será apresentado e as tarefas associadas à produção do mesmo. O tipo do resultado é especificado de acordo com a seguinte notação: S–Software, R–relatório técnico, E–Especificação, P–Publicação (em revistas ou em anais de simpósios e conferências).

Data	Tipo	Descrição	Tarefas
11/97	E, S, P	Extensão da máquina de hipertexto	FORM.2
1/98	R, S	Implementação da Interface com o Usuário	FORM.3, IC3.2, IC3.4
3/98	P, S	Implementação do compilador incremental	FORM.4, IC4.2, IC4.3, IC4.5
3/98	P	Descrição da linguagem e metodologia de utilização do SIM	SIM.1, SIM.2
4/98	S	Protótipo do SIM	SIM.4, IC2.3
5/98	R	Projeto da interface do SIM	SIM.5, IC1.2
5/98	E	Especificações e refinamentos do SIM	SIM.3, IC2.2
6/98	P	Noção e Propriedades de Refinamento para MooZ	REF.1, REF.2, REF.3
8/98	S	Extensão das bibliotecas de Java	IC1.3
12/98	S, P	Implementação, final do SIM e do sistema de interconexão de módulos de MooZ	SIM.4, SIM.6, IC2.3, IC2.4
12/98	S	Implementação da interface do SIM	SIM.5, IC1.4
1/99	R	Disponibilização via WWW	FORM6

Além desses resultados diretos, é importante notar que a participação dos bolsistas no projeto será de fundamental importância para a formação dos mesmos, já que eles obterão experiência em tecnologias de ponta: técnicas de programação orientada a objetos, métodos formais para desenvolvimento de software, WWW, técnicas de modularização e engenharia reversa de software. Em particular, no final do projeto os bolsistas serão especialistas em Java, que corresponde ao estado-da-arte em linguagens orientadas a objetos para o desenvolvimento de aplicações e interfaces seguras o suficiente para serem usadas via Internet/WWW.

7 Relevância e Originalidade

A disponibilização de uma ferramenta como ForMooZ é essencial para a difusão do uso de métodos formais, em particular MooZ. Tal ferramenta possibilitará tanto a análise sintática, a verificação de tipos de especificações escritas em MooZ, quanto o controle de versões e a navegação (com o uso do hipertexto) das mesmas. Desta forma, esperamos contribuir para uma maior disseminação desta cultura de desenvolvimento de software tanto no meio acadêmico como no meio empresarial.

Convém também ressaltar que o uso de ForMooZ pelos alunos da graduação e da pós-graduação, bem como a experiência adquirida durante o desenvolvimento deste projeto darão uma grande contribuição para o ensino de disciplinas relacionadas à Engenharia de Software.

O sistema de interconexão de módulos oferece poderosos recursos linguísticos e computacionais para estruturar a especificação de sistemas. Com a utilização do SIM define-se claramente as interfaces, dependências e ligações entre componentes de software, o que é essencial para entendê-los facilmente e reusá-los em larga escala. Dessa forma, esperamos aumentar consideravelmente a flexibilidade, adaptabilidade e reusabilidade de especificações em MooZ. Como conseqüência, esperamos facilitar a manutenção de componentes e sistemas desenvolvidos em MooZ.

Futuramente, resultados ainda mais promisores e originais poderão ser obtidos pela análise do contexto[3] no qual um modulo está situado. Isso permite o rastreamento de requerimentos, especificações, e outros documentos, o que facilita bastante o entendimento e manutenção de um sistema. Em particular, uma aplicação direta dessas técnicas é estruturar, através de um processo de engenharia reversa, programas e componentes que precisam ser modificados ou estendidos.

O acesso ao SIM e ForMooZ via Internet/WWW é extremamente relevante, tendo um enorme potencial econômico: componentes localizados remotamente poderão ser interconectados de qualquer localidade com acesso a WWW. Assim teremos uma tecnologia de suporte a "Global Yet Personal Software Libraries"[4] permitindo o acesso uniforme a bibliotecas de componentes de software localizadas remotamente em sistemas heterogêneos. Sem dúvida isso será essencial para comercialização de componentes de software em larga escala, o que tem grande potencial para acabar com sérios problemas da indústria de software [11].

O desenvolvimento da base de uma teoria de refinamento para MooZ possibilita, na prática, o desenvolvimento formal de software em MooZ. Assim, esperamos aumentar a confiabilidade de software desenvolvido em MooZ. Para a produção de componentes de software isto é muito importante já que a utilização de componentes defeituosos em larga escala pode ser catastrófica.

A investigação da relação entre os modelos semânticos de MooZ e de uma linguagem de programação é essencial para definir a noção de refinamento de especificações em MooZ para programas nessa linguagem. Essa noção poderá servir como base para documentar programas através de um processo de engenharia reversa: especificações em MooZ são derivadas a partir de programas, de forma que as especificações formalizam a funcionalidade dos programas. Assim será possível produzir uma documentação abstrata, precisa, e clara. Como benefício, acreditamos que o reuso e a manutenção de programas serão consideravelmente simplificados. Em especial, a documentação de componentes de software usando esta abordagem é bastante atrativa, já que facilitaria a localização de componentes desejados em bibliotecas.

A indústria de software certamente pode se beneficiar com os resultados do projeto de pesquisa proposto nesse plano de trabalho. Entretanto, um grande esforço será necessário para transferir *diretamente* a curto prazo alguns dos resultados para a indústria, já que as teorias e tecnologias propostas aqui diferem substancialmente das técnicas sendo atualmente usadas na indústria de sofware. Felizmente, alguns dos resultados podem ser aplicados através de um processo de engenharia reversa; esses resultados e as técnicas de modularização podem ser mais facilmente transferidos para a indústria a curto prazo.

Além destes aspectos, é importante notar que a formação de recursos humanos altamente qualificados, como proposto para os bolsistas deste projeto, tem uma importância estratégica para o Brasil. O projeto servirá também como plataforma de estudo para a realização de teses de mestrado e de doutorado.

8 Viabilidade Técnica

Como indicação de que o plano de trabalho proposto é realista e que resultados significativos podem ser obtidos, é importante notar que o Prof. Meira foi um dos pioneiros no estudo de

[3] Por exemplo, as relações de importação e refinamento entre os módulos de um sistema.

[4] No mesmo sentido de "Global Yet Personal Information Systems" [26].

hipertexto e métodos formais no Brasil, e na integração entre métodos formais e orientação a objetos, tendo desenvolvido vários trabalhos de relevância internacional na área do projeto [6, 7, 27, 28].

Dr. Vasconcelos tem boa experiência com métodos formais. Durante a realização do seu doutorado, a área de compilação incremental despontou como um dos principais enfoques de sua tese, na qual ele implementou e especificou formalmente uma série de técnicas de compilação incremental para a linguagem Z [30]. Em decorrência desta tese, o Dr. Vasconcelos teve a necessidade de aprofundar seus conhecimentos na área de orientação a objetos e de ferramentas e ambientes CASE. O envolvimento do Dr. Vasconcelos no projeto e implementação da ferramenta ForMooZ é, portanto, uma continuação natural do seu projeto de pesquisa durante a realização de seu doutorado.

Dr. Borba tem boa experiência em métodos formais, semântica e refinamento, linguagens orientadas a objetos, e sistemas de interconexão de módulos. Em particular, a base de uma teoria de refinamento para uma linguagem concorrente e orientada a objetos foi desenvolvida como parte da sua tese de doutorado [21]. Além disso, a linguagem considerada nesta tese oferece um suporte para módulos baseado em LIL [19], o modelo para definir a linguagem de interconexão de módulos do SIM.

Referências

[1] Kim Waldén and Jean-Marc Nerson. *Seamless Object-Oriented Software Architectures—Analysis and Design of Reliable Systems.* Prentice Hall, 1995.

[2] Grady Booch. *Object-Oriented Analysis and Design with Applications.* Benjamin/Cummings, second edition, 1994.

[3] George Cleland and Donald MacKenzie. Inhibiting factors, market structure and the industrial uptake of formal methods. In *International Workshop on Industrial Strength Formal Specification Techniques.* IEEE Computer Society Press, Boca Raton, Florida, April 1995.

[4] Carroll Morgan. *Programming from Specifications.* Prentice Hall, second edition, 1994.

[5] Cliff Jones. Does the O-O community need formal methods? In *TOOLS4: Technology of O-O Languages and Systems*, pages 15–22. Prentice Hall, 1991.

[6] Silvio Meira and Ana Cavalcanti. Modular object-oriented Z specifications. In Prof. C.J. van Rijsbergen, editor, *Workshop on Computing Series.* Springer-Verlag, Oxford, December 1990.

[7] Silvio Meira and Ana Cavalcanti. MooZ case studies. In S. Stepney, editor, *Comparative Study of Object Orientation in Z.* Springer-Verlag, 1992.

[8] J. Michael Spivey. *The Z Notation: A Reference Manual.* Prentice-Hall, 1989.

[9] Ian Hayes. *Specification Case Studies.* Prentice-Hall, 1987.

[10] W. Wayt Gibbs. Software's chronic crisis. *Scientific American*, September 1994.

[11] Brad Cox. No silver bullet reconsidered. *American Programmer*, November 1995.

[12] Bertrand Meyer. *Object-Oriented Software Construction.* Prentice Hall, 1988.

[13] F. Halasz and M. Schwartz. The Dexter Hypertext Reference Model. *NIST Hypertext Standardization Workshop*, 1990.

[14] C.J. Kacmar and J.J. Legget. PROXHY: A Process-Oriented Extensible Hypertext Architecture. *ACM Transactions on Information Systems*, 9(4), October 1991.

[15] Lin Tse Min. A Semântica Formal de MooZ. Master's thesis, Departamento de Informática - Universidade Federal de Pernambuco, 1993.

[16] Ken Arnold and James Gosling. *The Java Programming Language*. Addison Wesley, 1996.

[17] Rod Burstall and Joseph Goguen. Putting theories together to make specifications. In Raj Reddy, editor, *Proceedings, Fifth International Joint Conference on Artificial Intelligence*, pages 1045–1058. Department of Computer Science, Carnegie-Mellon University, 1977.

[18] Joseph Goguen, Claude Kirchner, Hélène Kirchner, Aristide Mégrelis, and José Meseguer. An introduction to OBJ3. In Jean-Pierre Jouannaud and Stephane Kaplan, editors, *Proceedings, Conference on Conditional Term Rewriting*, pages 258–263. Springer, 1988. Lecture Notes in Computer Science, Volume 308.

[19] Joseph Goguen. Reusing and interconnecting software components. *Computer*, 19(2):16–28, February 1986. Reprinted in *Tutorial: Software Reusability*, Peter Freeman, editor, IEEE Computer Society, 1987, pages 251–263, and in *Domain Analysis and Software Systems Modelling*, Ruben Prieto-Diaz and Guillermo Arango, editors, IEEE Computer Society, 1991, pages 125–137.

[20] Paulo Borba and Joseph Goguen. Refinement of concurrent object oriented programs. In Stephen Goldsack and Stuart Kent, editors, *Formal Methods and Object Technology*, Chapter 11. Springer-Verlag, 1996. Also appeared as Technical Report TR-18-95, Oxford University, Computing Laboratory, Programming Research Group, November 1995.

[21] Paulo Borba. *Semantics and Refinement for a Concurrent Object Oriented Language*. PhD thesis, Oxford University, Computing Laboratory, Programming Research Group, July 1995.

[22] Joseph Goguen and José Meseguer. Unifying functional, object-oriented and relational programming, with logical semantics. In Bruce Shriver and Peter Wegner, editors, *Research Directions in Object-Oriented Programming*, pages 417–477. MIT, 1987.

[23] Paulo Borba and Joseph Goguen. An operational semantics for FOOPS. In Roel Wieringa and Remco Feenstra, editors, *International Workshop on Information Systems—Correctness and Reusability, IS-CORE'94*. Vrije Universiteit, Amsterdam, September 1994. A longer version appeared as Technical Monograph PRG-115, Oxford University, Computing Laboratory, Programming Research Group, November 1994.

[24] Chuck Musciano and Bill Kennedy. *HTML: The Definitive Guide*. O'Reilly & Associates, 1996.

[25] Joseph Goguen. Parameterized programming. *Transactions on Software Engineering*, SE–10(5):528–543, September 1984.

[26] John Gurd and Cliff Jones. The global-yet-personal information system. In Ian Wand and Robin Milner, editors, *Computing Tomorrow*. Cambridge University Press.

[27] Paulo Borba and Silvio Meira. From VDM specifications to functional prototypes. *Journal of Systems and Software*, 21(3):267–78, June 1993.

[28] Paulo Borba and Silvio Meira. A system for translating executable VDM specifications into Lazy ML. *Software—Practice & Experience*, To appear 1997.

[29] Santos, C. S. *Um Ambiente Multi-Usuário Baseado em Hipertexto de Suporte à Construção de Especificações Formais Orientadas a Objeto.* Dissertação de Mestrado, Universidade Federal de Pernambuco, Departamento de Informática, Recife, 1992.

[30] Vasconcelos, A. M. L. *Incremental Processing of Z Specifications.* DPhil. Thesis, University of York, Department of Compu ter Science, York, 1993.

[31] Jansen, S. *Paradigmas de Interface em Sistemas de Hipertexto : Análise de Caso e Síntese de um Sistema Real.* Dissertação de Mestrado, Universidade Federal de Pernambuco, Departamento de Informática, Recife, 1995.

[32] Sun Microsystems, Inc. *Open Windows Developer's Guide 1.1*, 1990.