Engenharia Reversa Orientada a Objetos do Ambiente StatSim: método utilizado e resultados obtidos

Rosângela A. D. Penteado (*)
e-mail: rosangel@icmsc.sc.usp.br
Fernão S. R. Germano (**)
Paulo C. Masiero (**)(***)
e-mail: masiero@icmsc.sc.usp.br

(*) UFSCar/IFSC - USP
C.P. 676 - 13565-905 - São Carlos - SP

(**) ICMSC - USP
C.P. 668 - 13560-970 - São Carlos - SP

(***) Apoio CNPq

RESUMO

Um método para engenharia reversa de sistemas implementados sem usar tecnologia de orientação a objetos é apresentado, visando à criação de um modelo de análise orientado a objetos. O modelo de análise desenvolvido está baseado no Método Fusion, para análise e projeto orientados a objetos. O método foi aplicado para a engenharia reversa do ambiente StatSim e partes do modelo produzido são apresentadas. Faz-se uma avaliação do método proposto e discutem-se alguns resultados obtidos que são de interesse geral.

ABSTRACT

A reverse engineering method for systems implemented without using object oriented technology is presented to create an object oriented analysis model. The model developed is based on the Fusion Method for object oriented analysis and design. The method has been applied to the reverse engineering of the StatSim environment and parts of the model produced are presented. An evaluation of the proposed method is done and some of the results obtained that are of wide interest are discussed.

1. Introdução

No ambiente das aplicações comerciais pode-se dizer que hoje são poucas as ocasiões em que o desenvolvimento de um novo sistema não envolve também o reaproveitamento de código herdado de implementação existente, não raro em situação de completa ausência de documentação atualizada sobre a arquitetura do sistema. Talvez em menores proporções, essa é uma preocupação comum a todos os domínios de aplicações e tem recebido cada vez mais atenção tanto da comunidade usuária como de pesquisadores da área.

O termo "engenharia reversa" refere-se à recuperação da arquitetura e do modelo de análise de um sistema existente, baseada em observações sobre o comportamento do sistema, seu código, conhecimento de agentes relacionados com o sistema e outros documentos eventualmente disponíveis. Quando essa atividade refere-se principalmente à revitalização da documentação existente, sem criação de um modelo lógico, ela é denominada redocumentação [Ch90]. Quando, a partir do modelo produzido por uma atividade de engenharia reversa, o sistema é desenvolvido com um processo normal de engenharia, tem-se então uma atividade de reengenharia de sistema.

Foram duas as motivações para a realização deste trabalho. Uma delas envolve a necessidade de evolução do ambiente StatSim, desenvolvido pelo grupo de Engenharia de Software do ICMSC em ambiente Unix, para edição e simulação de statecharts. Esse sistema foi desenvolvido como um protótipo ao longo de cerca de cinco anos (1989-1993) a partir de projetos de mestrado e iniciação científica. A versão atual, que se encontra em forma de protótipo operacional desde 1993 possui cerca de 30000 linhas de código C e, além do código fonte, conta com descrições sobre o projeto espalhadas por várias dissertações, relatórios e artigos [Ma91, Ma94, Ca95]. Há interesse do grupo em evoluir o StatSim de forma mais organizada e foi eleita a abordagem orientada a objetos tanto para o desenvolvimento de novos módulos quanto para uma eventual reengenharia para ambiente DOS/Windows.

Há, por outro lado, como afirmado em [Co94], a suposição de que é possível fazer a reengenharia de um sistema não orientado a objetos para um modelo orientado a objetos, sendo que apenas algumas tentativas têm sido feitas nesse sentido, sem haver ainda propostas concretas e detalhadas de como conduzir esse processo. Coleman afirma em seu livro que o método Fusion seria adequado para isso, mas nenhuma diretriz existe de como fazê-lo. Assim, a segunda motivação deste trabalho foi avaliar a adequação do método Fusion para essa atividade e estabelecer uma abordagem concreta de como realizá-la. A engenharia reversa é a parte crítica de qualquer reengenharia, pois uma vez obtido um modelo do sistema, os métodos conhecidos de engenharia "avante" de software podem ser aplicados.

O objetivo deste artigo é apresentar uma abordagem concreta de engenharia reversa utilizando o método Fusion, a partir de um sistema implementado usando tecnologia não orientada a objetos, que se derivou com base na experiência de tê-la aplicado à reengenharia

do ambiente Statsim. A abordagem aplicada à Reengenharia do ambiente StatSim é descrita, sendo ilustrada sucintamente com alguns excertos do modelo produzido. Faz-se também uma avaliação da abordagem proposta e algumas descobertas relativas à implementação atual, que acredita-se serem de interesse geral, são comentadas.

A reengenharia de sistemas vem sendo tratada por diversos pesquisadores, sendo propostas técnicas que facilitam a alteração parcial ou total de tais sistemas. Em [Ja91] são apresentados três cenários para reengenharia de sistemas não orientados a objetos para serem modelados com a orientação a objetos. O exemplo de um sistema de faturamento é tomado para ilustrar os cenários e três estudos de caso são citados por terem servido de base para o desenvolvimento da proposta. Entretanto, a abordagem é apresentada apenas em linhas gerais dificultando qualquer tentativa de uso. A fase da recuperação das informações do projeto é chamada de redocumentação estrutural em [Wo95] e considerada como o ponto chave da engenharia reversa. Nesse sentido os autores apresentam um ambiente automatizado, chamado Rigi, que apoia um método para identificar, construir e documentar subsistemas hierárquicos.

No Brasil, alguns trabalhos na área de engenharia reversa foram publicados. Guedes e Staa apresentam a reengenharia de sistemas sob o enfoque econômico, propondo mecanismos de redução de custos e aumento de qualidade. Todo o processo tem apoio de ferramentas computadorizadas sendo que a ferramenta CASE Talisman é utilizada para a reengenharia do sistema [Gu93]. Leite e outros apresentam a reengenharia do ambiente DRACO-PUC, enfocando seu analisador sintático. O método SADT é utilizado para apresentar a estratégia de reengenharia e também para especificação das novas funcionalidades do sistema. Diagramas do método JSD são utilizados para representar a estrutura de procedimentos recuperados [Le92]. Ambos os trabalhos enfocam mais a recuperação e o projeto de programas, enquanto o trabalho relatado neste artigo preocupa-se com a recriação do modelo de análise do sistema.

O restante deste artigo está organizado da seguinte forma: na seção 2 é apresentado o método Fusion; na seção 3 o método de engenharia reversa orientado a objetos é descrito enfatizando passo a passo a análise realizada e o produto obtido; na seção 4 o ambiente StatSim serve de base para a aplicação passo a passo do método descrito na seção anterior. A avaliação dos resultados é apresentada na seção 5.

2. O método Fusion

O método Fusion foi criado por Coleman e outros, reunindo as que, em sua opinião, são as melhores técnicas propostas por outros métodos existentes [Co94]. Três fases distintas compreendem o método: análise, projeto e implementação. Os autores propõem três modelos de análise: modelo de objetos, modelo de ciclo de vida e modelo de operações.

O modelo de objetos tem por objetivo representar os conceitos existentes no domínio do problema e suas relações. Ele permite representar classes, atributos e relações entre classes,

incluindo relações de agregação e especialização/generalização. A notação é bastante semelhante à do modelo de objetos do OMT [Ru91].

O modelo de ciclo de vida permite especificar através de expressões regulares as seqüências permitidas de interações com o ambiente em que um sistema pode participar. Para facilitar a elaboração do modelo de ciclo de vida são construídos cenários do sistema mostrando o seu comportamento de acordo com os eventos gerados pelos agentes do ambiente e as respostas (eventos de saída) dadas pelo sistema a esses eventos de entrada. O ciclo de vida é definido globalmente para o sistema.

O modelo de operações é decorrente do modelo de ciclo de vida. A cada evento de entrada corresponde uma operação que é especificada através de um gabarito textual. Nesse gabarito há entradas para descrever informalmente a operação e para especificar: as informações que são fornecidas junto com os eventos de entrada ou estão disponíveis globalmente; objetos a que a operação pode ter acesso, mudando ou não seu estado; os eventos de saída que a operação gera e os agentes que os recebem; pré-condições que devem ser satisfeitas e resultados da operação, descritos como uma pós-condição. Diagramas de cenários, semelhantes a OMT, auxiliam a estudar a interação do sistema com seu ambiente e, conseqüentemente, definir os eventos de entrada que dão origem às operações.

A fase de projeto do método Fusion é baseada nos métodos CRC [Wi90] e de Booch [Bo91]. As decisões importantes de projeto podem ser documentadas usando Grafos de Interação de Objetos, Grafos de Visibilidade e Grafos de Herança, visando à Descrição de Classes. Na implementação traduz-se cada classe descrita na fase de projeto para a sintaxe e semântica de uma linguagem de programação em particular. O método oferece diretrizes para essa tradução, visando principalmente as linguagens C++, Eiffel e Smalltalk. Entre os aspectos discutidos pode-se destacar a tradução das classes, o desenvolvimento do corpo dos métodos, tratamento de erros e satisfação de requisitos de desempenho.

3. Método de Engenharia Reversa Orientado a Objetos

O método de engenharia reversa orientado a objetos proposto é resumido na Tabela 1. As grandes atividades recomendadas em cada passo seguem aquelas recomendadas por Jacobson em [Ja91]. A seqüência ideal para execução dos passos é a apresentada na tabela, sendo que os sub-passos dos passos 2 e 3, com exceção do 2.1, são conduzidos com bastante interseção e interação. A versão completa do método e dos experimentos realizados com ele pode ser encontrada em [Pe95]. Cada passo é descrito com mais detalhes no restante desta Seção.

Passo	Objetivo/Produto
1. Revitalizar a Arquitetura do Sistema com base na documentação existente	Obter informações relacionadas à arquitetura do sistema para o seu entendimento elaborando-se a lista de todos os procedimentos, sua descrição e relacionamentos.
2. Recuperar o Modelo de Análise da Solução Atual	Obter um modelo de análise considerando somente os aspectos físicos
2.1. Definir Temas	Modelar em temas as informações armazenadas produzindo a Lista de Temas
2.2. Desenvolver o Modelo de Objetos	Elaborar um Modelo de Objetos com as classes e seus relacionamentos, extraídos dos tipos abstratos de dados que compõem a estrutura do sistema; a lista de atributos, procedimentos associados às classes e a lista das anomalias existentes
2.3. Definir o Ciclo de Vida	Mostrar o comportamento global do sistema elaborando o Modelo de Ciclo de Vida
2.4. Abstrair e Desenvolver as Operações	Obter as operações realizadas pelo sistema construindo-se o Modelo de Operações
3. Abstrair o Modelo de Análise do Sistema	Obter um modelo de análise do sistema considerando os aspectos do domínio de aplicação
3.1. Desenvolver o Modelo de Objetos	Elaborar um Modelo de Objetos considerando as classes e seus relacionamentos que devem ser tratados pelo sistema. Para cada classe elaborar a lista de atributos e métodos
3.2. Elaborar o Modelo de Ciclo de Vida	Elaborar o Modelo de Ciclo de Vida, fornecendo uma visão global do comportamento do sistema a partir da abstração realizada.
3.3. Elaborar o Modelo de Operações	Elaborar o Modelo de Operações, descrevendo como as operações devem ser realizadas
4. Mapear o Modelo de Análise do Sistema Atual para o Modelo de Análise do Sistema	Descrever a relação entre os Modelos de Análise do Sistema atual e novo através do Mapeamento das Classes, dos Métodos, Adição/Retirada de Métodos às classes.

Tabela 1 - Método de Engenharia Reversa Orientado a Objetos

3.1 Revitalizar a Arquitetura do Sistema, com base na documentação existente.

O objetivo é recuperar as informações relacionadas à arquitetura do sistema para que ele possa ser entendido/estendido pelo engenheiro de software, facilitando o desenvolvimento dos passos posteriores. Caso o sistema possua documentação adequada, basta apenas identificá-la e organizá-la.

A documentação pode ser recuperada semi-automaticamente, com apoio de uma ferramenta, como aquela descrita em [Wo95], ou manualmente. De ambas as formas, deve-se descrever a função de cada um dos procedimentos utilizados para implementação do sistema,

fazendo-se referência cruzada entre eles: os procedimentos utilizados (chama) e os procedimentos utilizadores (chamado por). Essas informações são obtidas a partir do código fonte existente.

O produto deste passo é uma lista com as informações de todos procedimentos do sistema, ordenados alfabeticamente para facilitar recuperação manual e funcionando como um indice para o código fonte e as relações chama/chamado por .

3.2. Recuperar o Modelo de Análise da Solução Atual

Deve-se desenvolver neste passo um modelo de análise do sistema atual considerando-se os seus aspectos fisicos, ou seja, deve-se considerar somente a implementação atual. Os resultados deste passo são os modelos de objetos, de operações e de ciclo de vida do Fusion, que compõe o Modelo de Análise do Sistema Atual (MASA).

3.2.1. Definir Temas

O objetivo é modelar os temas relativos às informações que o sistema manipula. Temas são "grandes" assuntos, relacionados a um subsistema. O critério para classificação dos temas é fisico e baseado na implementação atual: os módulos fisicos do sistema, a interface (entradas, saídas) e os dados armazenados permanente e temporariamente.

3.2.2. Desenvolver o Modelo de Objetos

Para cada um dos temas considerados no item anterior deve-se listar inicialmente todas as classes existentes, identificadas a partir das estruturas de dados implementadas. Assim, os tipos de dados relevantes são considerados como classes. Dados existentes apenas internamente a um procedimento, por exemplo, raramente são uma classe do sistema. O relacionamento entre essas classes é obtido através da própria relação expressa entre os tipos de dados, isto é, os ponteiros existentes entre as diversas estruturas ou o esquema da base de dados. A partir das classes de objetos deve-se listar todos os atributos da classe. Os atributos são os mesmos que os existentes em cada uma das estruturas de dados selecionadas.

Os procedimentos são obtidos com base no conhecimento do engenheiro de software do sistema e dos documentos obtidos no passo 1. Deve-se analisar cada um dos procedimentos e verificar qual estrutura de dados está associada a ele. Além disso deve-se analisar a forma de associação entre a classe e o procedimento. Adotou-se a convenção c para construtor, quando o procedimento altera a estrutura de dados, e o para observador, quando o procedimento somente consulta a estrutura.

Como o sistema não tem uma implementação orientada a objetos, a análise dos procedimentos é complexa. A experiência com a análise do sistema StatSim levou os autores a desenvolver uma classificação de anomalias. As mais comuns se referem a procedimentos que são:

- observador de uma classe e construtor de outra (**oc**)
- observador de uma classe e construtor de mais de uma classe (**oc+**)
- observador de mais de uma classe e construtor de outra (**o+c**)
- observador de mais de uma classe e construtor de mais de uma classe (**o+c+**)
- construtor de duas classes (ou mais) (**c+**)
- observador de duas classes (ou mais) (**o+**)
- procedimentos que não se referem a classe alguma, dependentes da implementação. relacionados à interface ou controlador, etc. (**i**)

Quando não há anomalia o procedimento é classificado simplesmente como (**o**) ou (**c**). Como produto deste passo tem-se o MASA, com as classes, atributos e procedimentos identificados e a classificação dos procedimentos.

3.2.3. Definir o Ciclo de Vida

O objetivo é definir o ciclo de vida do sistema. A observação da interface do sistema atual é importante para definir todos os eventos de entrada e de saída que o sistema aceita. Interação com o sistema para definir seqüências relevantes de interações permitidas também é importante, dessa forma recuperando cenários que, reunidos, são a base para a definição do ciclo de vida. Manuais de uso também podem ser usados nesta atividade.

3.2.4. Abstrair e Desenvolver as Operações

Para cada operação identificada constrói-se um modelo de operação seguindo o gabarito descrito na seção 2. A operação é normalmente implementada pelo procedimento chamado pelo controlador da interface a partir da ocorrência de um evento de entrada. Muitas vezes há uma cadeia de chamadas até se chegar ao procedimento que realmente executa a operação, geralmente auxiliado por um conjunto de outros procedimentos subordinados. Esse deve ser o procedimento base para a especificação da operação, pois os demais cuidam de aspectos da interface e são dependentes da implementação. Consistências efetuadas dão origem a pré-condições.

As abstrações relevantes para a definição das operações são encontradas geralmente até no penúltimo nível da hierarquia de chamada de procedimentos a partir da interface. Essas operações são parte daquelas classificadas como (**i**), no sub-passo 2.2. O nível mais baixo

contém os procedimentos que constroem ou observam objetos, dando origem aos demais tipos da classificação do sub-passo 2.2.

3.3. Abstração para o Modelo de Análise do Sistema

Até o momento trabalhou-se com informações disponíveis na implementação existente, construindo-se o Modelo de Análise do Sistema Atual. A partir do passo 3 realiza-se a abstração da visão física, criando-se a visão lógica do sistema. A visão lógica abstrai da visão física, sempre que possível, aspectos que deveriam ter sido especificados anteriormente. O produto gerado é semelhante aos documentos gerados no passo 2, com exceção da lista de temas, dando-se enfoque ao domínio da aplicação e não à implementação. Um Dicionário de Dados, manual ou automatizado, deve ser usado para armazenar informações sobre o MAS, como recomendado em Fusion.

3.3.1. Desenvolver o Modelo de Objetos

As classes do MAS são abstraídas das classes do MASA. É comum que um conjunto de classes e relações do MASA dê origem a uma classe do MAS, pois os objetos de implementação escolhidos (listas, vetores de ponteiros, etc.) devem ser abstraídos para um conceito do sistema. Pode ocorrer também que uma classe (tipos) do MASA com mais de uma instanciação concreta (objetos) dê origem a mais de uma classe no MAS, correspondentes aos conceitos do sistema.

Os atributos são analisados e normalmente têm seu nome modificado para um mnemônico mais próximo do conceito que ele representa no mundo real. Alguns atributos são desnecessários porque são dependentes da implementação e os que apontam para outras classes são substituídos por relacionamentos. Alguns métodos canônicos, como aqueles que criam a classe e permitem acesso a atributos da classe (para atualizar ou consultar) podem ser incorporados ao MAS neste passo, mas o método Fusion não recomenda que isso seja feito, podendo ser deixado para as fases de projeto ou de implementação.

É importante notar que os métodos não canônicos, em Fusion, são incorporados ao Modelo de Objetos na fase de projeto, a partir dos Grafos de Interação de Objetos. Como na engenharia reversa os procedimentos são recuperados a partir da implementação atual, eles também podem ser acrescentados ao modelo de objetos, eliminando-se as anomalias identificadas. Por exemplo, procedimentos sem anomalias (classificados no sub-passo 2.2. como **o** ou **c**) geralmente serão métodos do MAS, a menos de possível adaptação do nome; um procedimento com anomalia **oc** deve dar origem a dois métodos: um relacionado à classe da qual o procedimento é observador e outro relacionado à classe da qual o procedimento é

construtor e assim por diante para as demais anomalias. Os procedimentos classificados como (i) não dão origem a métodos.

3.3.2. Elaborar o Modelo de Ciclo de Vida

O modelo de ciclo de vida é praticamente o mesmo que o produzido no passo 2. 3, mas tomando-se o cuidado de alterar os nomes dos eventos de entrada e de saída para nomes mais significativos. Cada alteração realizada exige compatibilização com os nomes do modelo de operações. A ampliação de funcionalidade, que não é tratada neste passo, implicaria em outras mudanças que fogem ao escopo deste trabalho mas podem ser encontradas em [Pe95].

3.3.3. Desenvolver o Modelo de Operações

Para cada uma das operações do modelo de operação desenvolvido no sub-passo 2.4, possivelmente com seu nome alterado no sub-passo 3.2, deve-se re-escrever o modelo de operação. Deve-se respeitar a funcionalidade especificada mas atualizar a notação para o estilo do Fusion e considerar agora as classes, relacionamentos e atributos do MAS.

3.4. Mapear o Modelo de Análise do Sistema Atual para o Modelo de Análise do Sistema

Este mapeamento é fundamental para o desenvolvimento futuro de todo ou de substituição de partes do sistema atual, facilitando o reuso. O próprio desenvolvimento do sub-passo 3.1. produz o mapeamento entre o MASA e o MAS, que deve ser cuidadosamente documentado e checado em termos de completitude neste passo. Esse mapeamento corresponde à relação "implementado por", entre as classes do MAS e do MASA, e entre os métodos do MAS e os procedimentos do MASA. Em termos de notação isso pode ser feito por tabelas, mas a informação pode também estar armazenada no Dicionário de Dados.

4. Estudo de Caso: Engenharia Reversa do Ambiente Statsim

A abordagem apresentada na Seção anterior foi aplicada ao ambiente StatSim. Alguns pontos principais dessa atividade são comentados e ilustrados com alguns dos produtos produzidos em cada passo.

4.1 Revitalização da Arquitetura

Como foi mencionado anteriormente, as informações disponiveis para o ambiente eram o código fonte, as estruturas de dados, relatórios e dissertações dos trabalhos realizados. Não foram utilizadas ferramentas neste passo, sendo que a maior parte das informação foi extraida do código fonte. Foram analisados todos os procedimentos que compõem o sistema, que foram agrupados em módulos de acordo com as funções que realizam. Assim, para cada um dos módulos existentes foram listados todos os procedimentos e construida a relação chama/chamado por.

4.2 Recuperação do MASA

Com base nas informações obtidas anteriormente e com o conhecimento adquirido pelo uso do ambiente, este modelo foi construido completamente. Os resultados de cada sub-passo são comentados a seguir.

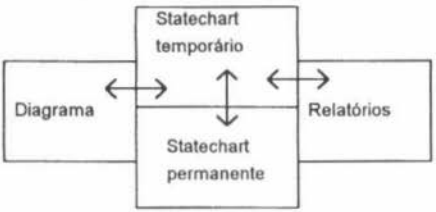

Figura 1 - Temas

Temas: seguindo as diretrizes dos sub-passo 2.1 os diagramas (de statecharts) foram identificados como um tema correspondente a entrada/saida, os relatórios produzidos pelos vários tipos de simulação, como um tema relacionado a saida, statechart permanente (que correspondem a informações armazenadas em disco, que são carregadas para processamento, pois não se usou um SGBD na implementação atual) e statechart temporário (quando o statechart é carregado do disco para a memória RAM) como os temas relacionados a armazenamento permanente e temporário, respectivamente. A Figura 1 apresenta o esquema de relacionamento dos temas.

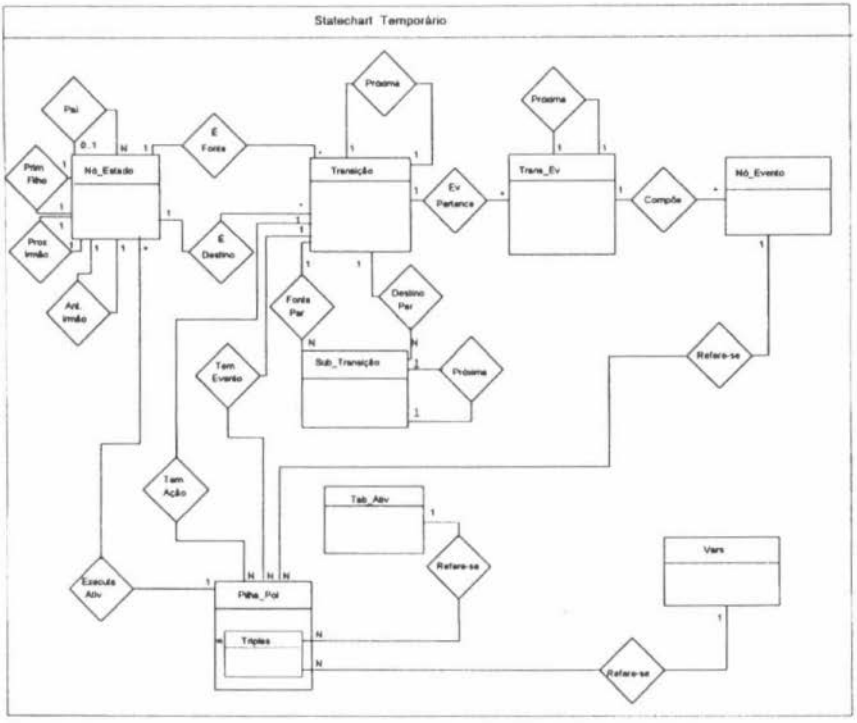

Figura 2 - Modelo de Objetos do MASA para o Tema Statechart Temporário

Classificação	Quantidade
observadores (o)	102
construtores (c)	115
anomalia (oc)	24
anomalia (oc+)	21
anomalia (o+c)	11
anomalia (o+c+)	0
anomalia (c+)	32
anomalia (o+)	15
anomalia (i)	34
Total	353

Tabela 2 - Classificação dos Procedimentos do Ambiente StatSim

Modelo de Objetos: para os temas selecionados foram recuperadas as informações que cada um deles modela e foram criadas as classes de objetos, como recomendado no sub-passo 2.2. Nem sempre as informações contidas no documento de revitalização da arquitetura do sistema foram suficientes para a classificação dos procedimentos, sendo necessárias consultas diretamente ao código fonte. Quando do desenvolvimento do StatSim, o método adotado foi usar a linguagem C com a idéia de tipos abstratos de dados: definir estruturas de dados implementando entidades do problema e procedimentos agindo sobre essas estruturas. Isso facilitou a identificação das estruturas relevantes para o modelo de objetos. O modelo de objetos produzido para o tema Statechart temporário é mostrado na Figura 2. Notem que neste modelo todas as cardinalidades são simples, não existindo M:N. Foram analisados e classificados 353 procedimentos neste sub-passo 2.2., como é ilustrado na Tabela 2.

Ciclo de vida: o ambiente StatSim é composto de um modo de edição e outro de simulação. A edição é realizada sobre o diagrama e corresponde a operações para criar, remover e consultar estados e transições. Na simulação pode-se escolher o tipo de simulação, interagir de acordo com esse tipo e obter relatórios da simulação. O Modelo de Ciclo de Vida foi um resultado relativamente simples de ser produzido devido à interface guiada por menus e janelas do StatSim, sendo que a expressão regular produzida traduz fielmente as interações permitidas na interface atual. Por exemplo, o ambiente é expresso pela expressão:

StatSim = (Edit | Simulate)

Edit = (editar_estados | carregar_statechart | editar_arquivos.carregar_statechart).
(editar_arquivos | editar_estados | editar_atividades | editar_variáveis |
carregar_statechart | ocultar_eventos)* . #statechart_editado

cada expressão é então definida por uma sub-expressão, como Edit, e assim por diante.

Operações: as operações foram especificadas a partir dos procedimentos chamados pela interface e a hierarquia de chamadas foi analisada até os procedimentos folha.

4.3- Abstração para o Modelo de Análise do Sistema:

Modelo de Objetos: a Figura 3 exibe o modelo de objetos do MAS, gerado a partir do modelo de objetos obtido no sub-passo 2.2. O nome é apenas Statechart porque não há mais necessidade no modelo de análise de distinção entre armazenamento temporário e permanente. Dessa forma, não há necessidade de criar um modelo de objetos para statechart permanente. A Figura 4 mostra a classe **Estado** com os seus atributos e alguns de seus métodos. Como essa classe é uma agregação de outras três, como pode ser visto na Figura 3, os atributos e métodos associados são aqueles relacionados às características gerais da classe.

Ciclo de Vida: nova versão foi produzida, apenas com alterações quanto aos nomes dos eventos de entrada e de saída, sem incluir ou eliminar eventos de entrada e saída na seqüência de eventos.

Operações: todas as operações identificadas no passo 2 foram re-especificadas, de forma a se adequarem aos novos nomes dos eventos de entrada e saída e ao MAS.

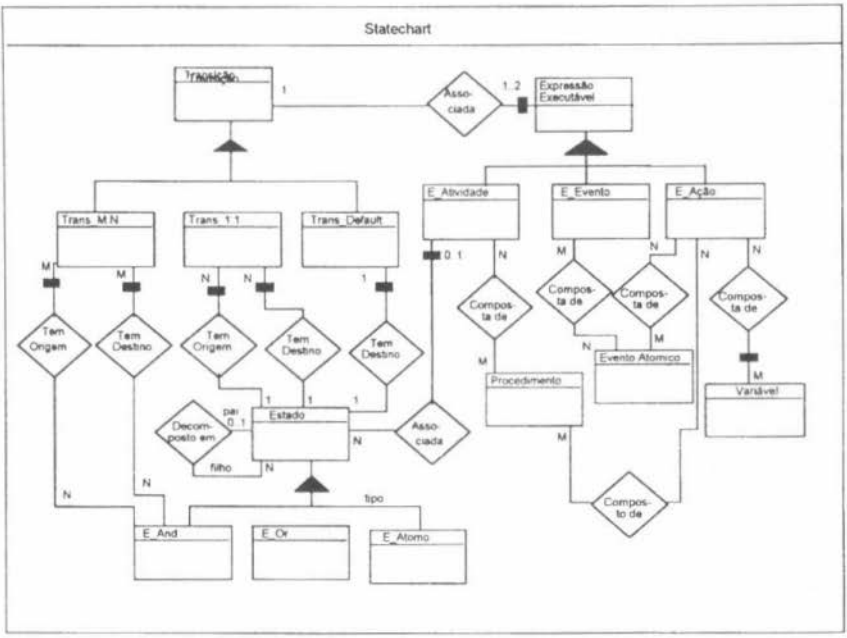

Figura 3 - Modelo de Objetos do MAS para o Tema Statechart

Estado		
Atributos:	nome	passos_consecutivos
	nivel_visual	num_médio_ativ_consecutivas
	nivel	pai
	ativação	filho
	cópia_ativação	atividade
	última_ativação	sentinela
	passos_ativos	default
Métodos:	criar_estado	
	colocar_infs_gráficas_estado	
	confirmar_estado	
	obter_infs_estado	
	remover_estado	

Figura 4 - Lista de atributos e métodos (parcial) da Classe Estado do MAS

4.4 Mapeamento do MASA para o MAS

A tabela 3 apresenta parcialmente a correspondência entre as classes de objetos extraídas do passo 2 e a abstração realizada no passo 3.

A classe **Estado** do MAS é descrita como uma agregação das classes **E_And, E_Or** e **E_Átomo**, englobando a classe **Nó_Estado** do MASA. Isso ocorreu porque na implementação realizada não houve preocupação em fazer procedimentos específicos ao tipo do estado. O mesmo aconteceu com a classe **Transição** do MAS, com os três tipos de transições agregados a ela. No MASA a classe **Sub_Trans** é que trata das transições M para N. **Pilha_Pol**, por outro lado, é uma estrutura de dados que implementa todos os tipos de expressões executáveis.

Classes de Objetos do MASA	Classe de Objetos do MAS
Nó_Estado	Estado: E_And E_Or E_Átomo
Transição, Trans_Ev Sub_Transição	Transição, Trans 1:1 Trans_Default Trans_M:N
Nó_Evento	Evento_Atômico
Pilha_Pol	E_Atividade, E_Evento, E-Ação
Tab_Ativ	Procedimento
Vars	Variável

Tabela 3 - Classes de Objetos do MASA e MAS

Procedimentos do MASA	Métodos do MAS	Classe / Tema
desenha_bolha (c+)	desenhar_estado (c) colocar_infs_gráficas_estado (c)	Bolha / Diagrama Estado/Statechart
aloca_bolha (c)	criar_estado (c)	Estado/ Statechart
confirma_bolha (c)	confirmar_estado (c)	Estado/ Statechart
cria_panel_nro_filhos (o)	receber_nro_filhos (o)	E_And / Statechart
identifica_bolha (o+)	obter_infs_estado (o) identificar_bolha (o)	Estado/Statechart Bolha/Diagrama
desaloca_nós_e_transição (oc+)	remover_estado (c) remover_transição_estado(c) remover_atívids_estado (c) consultar_bolha (o)	Estado/Statechart Transição/Statechart E_Atividade/Statechart Bolha/Diagrama

Tabela 4 - Procedimentos do MASA e métodos correspondentes do MAS

A tabela 4 apresenta o mapeamento para alguns procedimentos. Os procedimentos foram analisados quanto às anomalias e os métodos resultantes foram acrescentados ao modelo de objetos seguindo as diretrizes recomendadas na Seção 3.4. Para simplificar, nessa tabela, não são mostrados os parâmetros dos métodos e procedimentos.

Pode-se observar, por exemplo, que o procedimento aloca_bolha do MASA corresponde a um método de criação de um novo objeto Estado no MAS (método canônico). Alguns procedimentos sofrem apenas alteração (adaptação) de nome como é o caso de confirma_bolha (MASA). O método cria_panel_nro_filhos (MAS) não está associado à classe Estado e sim à classe E_And por estar relacionado aos estados tipo AND. O procedimento desaloca_nós_e_transição, com anomalia (oc+) deu origem a quatro métodos associados a diferentes classes. Esse é um exemplo típico dessa anomalia, onde um procedimento engloba todas as modificações de estado do sistema, envolvidas em uma operação.

5. Avaliação dos Resultados Obtidos e Conclusões

O objetivo de ter um modelo orientado a objetos do ambiente StatSim foi plenamente atingido. Com base na implementação atual, foram derivados os modelos de objeto, de ciclo de vida e de operações, que serão utilizados a partir de agora para evolução do ambiente. O experimento serviu também para reforçar a suposição já testada por alguns autores, de que é possível derivar um modelo orientado a objetos com base em uma implementação não orientada a objetos.

O estudo realizado, por outro lado, permitiu descobrir problemas importantes da implementação atual, que dificultam a manutenção e violam regras de modularização bem fundamentadas. As anomalias detectadas na implementação dos procedimentos podem ser consideradas uma métrica de qualidade da implementação atual. Como pode ser visto na tabela 2, 39% dos métodos apresentam anomalias. Dentre esses, as anomalias mais frequente foram (c+) e (i). A anomalia (c+) é menos grave pois os procedimentos não alteram o estado do sistema e podem ser convertidas com relativa facilidade em futuras evoluções; a anomalia (i) também não altera o estado do sistema; sendo que alguns desses procedimentos poderiam ser corrigidos (longa hierarquia de chamadas, por exemplo), mas outros existem em função da implementação. É interessante notar que não foram encontradas anomalias do tipo (o+c+). Ter que modificar duas ou mais estruturas de dados e também ter acesso a duas ou mais estruturas dentro de um mesmo procedimento torna-o muito grande, levando a sua divisão, o que pode explicar a baixa ocorrência dessa anomalia.

Parte dos problemas pode ser creditado ao fato de que padrões rígidos de projeto e codificação não foram seguidos durante todo o desenvolvimento, mas outros fatores certamente contribuíram para essa dificuldade: a falta de formação/treinamento adequado dos

estudantes que participaram do projeto, a liberdade permitida pela linguagem, o trabalho quase sempre individual, dificultando procedimentos de revisão e a não conscientização de que o sistema foi feito para evoluir e que outras pessoas deveriam interagir com os itens de configuração do sistema.

O objetivo de ter um método concreto, fundamentado e testado na prática foi atingido. O uso do método Fusion na engenharia reversa é original e mostrou-se bastante viável. O detalhamento do método de engenharia reversa utilizado e sua documentação permitem que o processo possa ser reutilizado. Como continuidade deste projeto, pretende-se investigar empiricamente como a evolução futura do ambiente StatSim será facilitada pelo novo modelo construido. A aplicação do método em outras situações também será feita. Ferramentas para apoio a esse trabalho serão investigadas.

Referências Bibliográficas

[Bo91] Booch, G. - Object-Oriented Design with Applications. Benjamin Cummings, CA, 1991.

[Ca95] Cangussu,J.W.L.; Penteado,R.D.; Masiero,P.C.; Maldonado,J.C. - Validation of Statecharts Based on Programmed Execution. 7 th International Conference On Computer And Information, Peterborough, Ontario Canada, July, 1995.

[Ch90] Chikofsky,J.E., Cross,J.H.- Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software 1990, january, 1990.

[Co94] Coleman,D. et ali - Object-Oriented Development - The Fusion Method, Prentice Hall, 1994.

[Gu93] Guedes,L.C.; Staa, A.- Um Processo de Re-engenharia Econômico e Eficaz. VII Simpósio Brasileiro de Engenharia de Software, Rio de Janeiro, 1993.

[Ja91] Jacobson, I.; Lindström, F. - Re-engineering of Old Systems to an Object-Oriented Architecture. ACM OOPSLA'91 Conference Proceedings, October 1991.

[Le92] Leite,J.C.S.P; Prado,A.F.; Sant'ana, M. - Draco-PUC, Experiências e Resultados de Re-Engenharia de Software. VI Simpósio Brasileiro de Engenharia de Software, Gramado, Rio Grande do Sul, 1992.

[Ma91] Masiero,P.C.; Fortes,R.P.M.; Batista,J.E.S. - Edição e Simulação de Aspectos Comportamentais de Sistemas de Tempo Real, XVIII Seminário Integrado de Software e Hardware, Santos, Brazil. 1991.

[Ma94] Masiero, P.C.; Maldonado, J.C; Boaventura, I. A. G. - A Reachability Tree for Statecharts and Analysis of Some Properties. Information and Software Technology, v. 36, n. 10, 1994.

[Pe95] Penteado, R. D. - Uso, Evolução e Engenharia Reversa de um Ambiente de Apoio ao Desenvolvimento de Sistemas Reativos. Tese de doutorado, IFSC - USP, 1995 (em preparação).

[Ru91] Rumbaugh, J. et al - Object-Oriented Modeling and Design. Prentice Hall International, Englewood Cliffs, 1991.

[Wi90] Wifgs-Brock ; Wilkerson, V.; Wiener, L. - Designing Object-Oriented Software, Prentice-Hall, Englewood Cliffs, NJ, 1990.

[Wo95] Wong, K.; et al - Structural Redocumentation: A Case Study. IEEE Software, january 1995.