Estrutura de Dados e Regras de Negócio Configuráveis pelo Usuário Final

Paulo Eduardo Cardoso
Instituto Nacional de Pesquisas Espaciais
paulinho@dss.inpe.br

Mauricio Gonçalves Vieira Ferreira
Instituto Nacional de Pesquisas Espaciais
mauricio@ccs.inpe.br

Resumo

A Divisão de Desenvolvimento de Sistemas de Solo (DSS) sempre desenvolveu os sistemas de controle com arquiteturas que permitisse que os mesmos fossem re-usados por futuros satélites com um conjunto reduzido de mudanças. O desafio que se apresenta é construir um sistema de controle adaptativo usando a tecnologia de Objetos Dinâmicos de modo que tais sistemas possam atender futuros requisitos sem necessidade de mudanças no código. De acordo com esta tecnologia as estruturas dos objetos e os seus comportamentos são armazenados em banco de dados de modo que os usuários finais possam modificá-los usando ferramentas de configuração e possivelmente uma linguagem especifica ao domínio do problema. Este trabalho mostra uma parte da pesquisa realizada para se atingir o objetivo proposto. A pesquisa produziu um modelo para representação e armazenamento de estruturas de dados e comportamentos que em principio pode ser utilizado por qualquer domínio de problema.

Palavras chaves: Modelo de Objetos Dinâmicos, Sistemas Adaptativos, Metadados, *Design Patterns*

1. Introdução

Até o momento o INPE esta operando três satélites: SCD1, SCD2 e CBERS-2. Os satélites de coleta de dados (SCD1 e SCD2) são parte da Missão Espacial Completa Brasileira (MECB) e têm como objetivo receber dados transmitidos por plataformas de coleta de dados distribuídas pelo território brasileiro, processar estes dados e deixá-los disponíveis para a comunidade. O satélite SCD1 está em operação desde 1993 e o SCD2 desde 1988. O satélite CBERS-2 é o segundo satélite de sensoriamento remoto desenvolvido com China dentro do programa "China Brazil Earth Resource Satellites". O primeiro satélite CBERS foi lançado em 1999 e foi controlado pela China e pelo Brasil até o final de sua vida (2003). Atualmente, os seguintes programas espaciais estão em execução: programa de micro-satélites científicos, o qual prevê o lançamento dos satélites EQUARS (Equatorial Atmosphere Research Satellite) e MIRAX (X-Ray Astronomy Satellite); e o programa de uma plataforma multi-missão que será utilizada no projeto dos satélites de sensoriamento remoto SSR-1 e SSR-2. Além disso, o acordo com a China está sendo renovado para o desenvolvimento de mais dois satélites de sensoriamento remoto (CBERS-3 e CBERS-4).

O desenvolvimento dos sistemas de controle de satélites esta sob a responsabilidade da Divisão de Desenvolvimento de Sistemas de Solo (DSS) do INPE. Desde o início de suas atividades a DSS sempre buscou desenvolver sistemas cuja arquitetura permitisse o máximo de re-uso para operar outros satélites. A DSS já desenvolveu 2 sistemas de controle para os satélites dos programas MECB [9] e CBERS [2][3] e atualmente esta desenvolvendo um terceiro sistema para o controle dos micro satélites científicos. Em cada novo sistema novos processos têm sido adicionados de modo a melhorar reusabilidade dos sistemas [5]. Que este objetivo tem sido, pelo menos parcialmente, alcançado pode ser comprovado pelo fato de que, hoje o sistema desenvolvido para os satélites CBERS também é responsável pelo controle dos satélites da MECB.

A cada nova missão, a equipe de desenvolvimento tem adquirido mais experiência para definir sistemas mais flexíveis que podem ser reusados por outras missões com um número reduzido de modificações.

O atual desafio é buscar uma solução que permita a adição de novas funcionalidades nos sistemas sem que seja necessário modificar o código, ou pelo menos reduzir esta modificação ao mínimo possível. O objetivo é que os sistemas possam ser re-configurados o máximo possível pelos próprios engenheiros de operação dos satélites. Deste modo, haveria uma grande redução no custo e tempo de desenvolvimento para atender os requisitos de cada nova missão espacial

Uma das pesquisas realizadas no sentido de vencer este desafio é a utilização da tecnologia de Objetos Dinâmicos. De acordo com esta tecnologia as estruturas dos objetos e os seus comportamentos são armazenados em banco de dados de modo que os usuários finais possam modificá-los usando ferramentas de configuração e possivelmente uma linguagem especifica ao domínio do problema.

Este trabalho mostra uma parte da pesquisa realizada para se atingir o objetivo proposto. A pesquisa produziu um modelo para representação e armazenamento de estruturas de dados e comportamentos que em principio pode ser utilizado por qualquer domínio de problema.

2. Arquitetura para um sistema altamente re-usável

Embora cada nova missão espacial estabeleça novos requisitos, a maior parte deles permanecem os mesmos. Desta forma é muito importante identificar quais partes são mais sujeitas a modificações para projetar sistemas mais fáceis de serem adaptados para outras missões espaciais para atendimento de novos requisitos. O ideal seria que os próprios engenheiros de operação pudessem mudar o sistema de acordo com novos requisitos sem haver necessidade de escrever um novo código.

Uma forma de alcançar este objetivo é mover certos aspectos do sistema, como por exemplo, as regras do negócio, para dentro de um banco de dados. Os sistemas podem então ser projetados para permitir mudanças nas necessidades do negócio através de modificações dos valores da base de dados. Deste modo, é possível introduzir novos produtos sem re-programação e até mesmo fazer mudanças nos modelos do negócio em tempo de execução. Através desta nova abordagem, é possível reduzir o custo e o tempo de desenvolvimento dos produtos.

Uma solução para implementar um sistema com estas características é utilizar a tecnologia de “arquiteturas reflexivas” ou “meta-arquiteturas”. Sistemas construídos com este tipo de arquitetura também são chamados de Modelo de Objetos Dinâmicos (DOM) ou Modelo de Objetos Adaptativos.

Sistemas DOM normalmente são uma evolução de *frameworks* de domínios específicos que alcançaram uma elevada maturidade. À medida que o *framework* evolui, os desenvolvedores adquirem um melhor entendimento do domínio do problema e podem reconhecer quais as partes que são mais sujeitas a mudanças.

2.1 Visão resumida do Modelo de Objetos Dinâmicos

Um Modelo de Objetos Dinâmicos é um sistema que representa classes, atributos e relacionamentos como metadados. Usuários mudam os metadados de modo a refletir mudanças no domínio do problema. Estas mudanças modificam o comportamento dos sistemas. Em outras palavras, o modelo de objetos é armazenado em banco de dados e interpretado em tempo de execução. [10] Deste modo, futuras mudanças no comportamento não exigem mudanças no código do sistema.

Esta arquitetura é baseada nos seguintes padrões (*design patterns*): *TypeObject* [6], *Property* e *Strategy* [4].

O padrão *Type-Object* torna explícito o relacionamento de classificação classe-instância. Instâncias de um tipo de classe substituem as sub-classes da classe original. Isto permite que os usuários tenham completo controle deste relacionamento. Eles podem até mesmo modificá-lo em tempo de execução. Este padrão divide o sistema em Entidades e “Tipos de Entidades”. Entidades têm Atributos, sendo que cada qual deles tem um “Tipo de Atributo”. Cada “Tipo de Entidade” especifica os “Tipos de Atributos” de suas entidades. O “Tipo de Entidade” também pode armazenar um conjunto de estratégias como parte de suas propriedades [8].

O padrão *Property* declara uma única variável de uma instância para armazenar uma coleção de atributos, em lugar de declarar uma variável para cada atributo. Cada atributo é associado com uma chave única. Os usuários podem utilizar estas chaves para acessar, adicionar, modificar ou remover atributos em tempo de execução [8].

O padrão *Strategy* torna possível representar o comportamento através de objetos que representam regras primárias ou combinação de regras tais como repetições, condições, seqüências e ramificações. Ele define uma interface padrão para uma família de algoritmos. Os clientes podem trabalhar com qualquer algoritmo de uma dada família. Se o comportamento de um objeto é definido por uma ou mais estratégias, então aquele comportamento pode ser facilmente mudado por um mecanismo acessível pelos usuários [10].

O Núcleo de uma arquitetura DOM é a combinação dos três padrões descritos. Eles permitem que o usuário controle o estado, o comportamento e o relacionamento de classificação A arquitetura resultante é criada aplicando-se o padrão *Type-Object* duas vezes juntamente com o

padrão *Property*. O Sistema é dividido em Entidades e "Tipos de Entidades". As Entidades tem atributos que podem ser definidos utilizando-se o padrão *Property*. Cada estado (atributo) de uma Entidade é associado a um objeto que define o tipo do estado. O tipo do estado, por sua vez, está contido em um conjunto de tipos válidos associados ao objeto que define o tipo da Entidade. Finalmente, o padrão *Strategy* ou *RuleObjects* [1] responsável pela representação do comportamento de cada tipo de Entidade [10] é adicionado. A composição destes padrões forma o padrão conhecido como *"Type Square With Rules*", o qual está representado na Figura 1

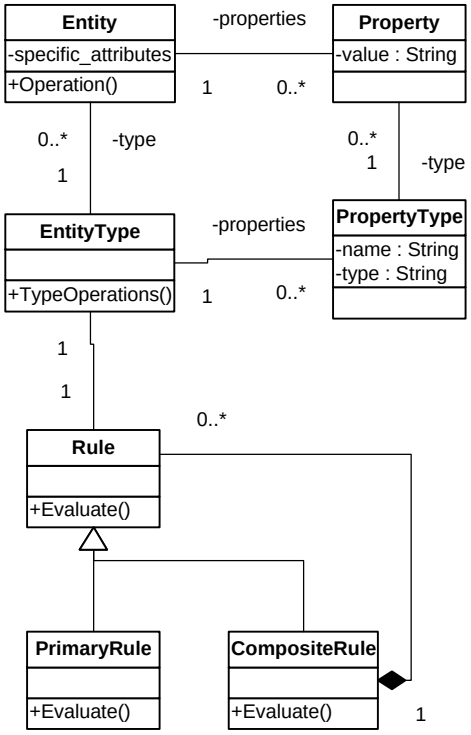

Figura 1 - *Type Square With Rules*

Os padrões *TypeObject*, *Property* e *Strategy* são os blocos de construção da arquitetura. Sistemas DOM representam o restante do modelo de objetos como dados de configuração, ou metadados. Portanto, modificações dos metadados mudam o modelo de objetos e conseqüentemente o comportamento do sistema.

Os metadados podem ser recuperados da base de dados onde estão armazenados e construir o DOM que representa o modelo de negócio real. Se um banco de dados orientado a objetos é usado, os tipos de objetos e seus relacionamentos podem ser construídos pela simples instanciação dos *TypeObjects*, *Properties* e *RuleObjects*. Caso contrário, os metadados podem ser lidos da base de dados para construir estes objetos, usando os padrões *Interpreter* e *Builder* [7][8].

Ferramentas especiais e linguagens visuais devem ser implementadas para auxiliar os usuários finais uma vez que a complexidade do sistema é transferida para os dados de configuração e as decisões de configuração são delegadas a estes usuários. Desta forma os usuários se tornam programadores especializados na linguagem específica definida pelos metadados.

Dois fatores contribuem para o aumento da flexibilidade da arquitetura DOM. Primeiro, somente a parte genérica (*Type Square With Rules*) necessita ser codificada usando os meios tradicionais de programação. Segundo, a parte variável é transferida para os metadados de modo que os usuários podem adaptar o modelo de objetos de acordo com suas necessidades.

3. Solução para representação dos objetos dinâmicos

Analisando-se o padrão "*Type Square and Rules*" notou-se que a maior dificuldade de uma implementação DOM esta na definição de regras de negócio configuráveis através da base de dados. Esta definição pode se tornar tão mais complexa quanto maior for a flexibilidade a ser oferecida aos usuários finais.

O princípio para a configuração das regras é que os usuários devem poder combinar regras básicas pré-definidas (funções primitivas) para construir uma regra de maior complexidade. Fazendo-se uma analogia com as linguagens de programação, esta regra pode ter um tipo de retorno e usar parâmetros para que um dado contexto de Entidades possa ser utilizado e/ou atualizado pela combinação de funções primitivas que compõe o procedimento a ser executado. Este procedimento, entre outras funcionalidades, pode comportar a execução de seqüências simples, condicional e iterativa de comandos. Um comando, por sua vez, pode ser uma função primitiva ou um dos tipos de seqüência.

Observou-se então que uma regra pode ser modelada como um "Tipo de Entidade" que tem "Tipos de Propriedades" tais como tipo de retorno, parâmetros e procedimento. Um procedimento é um "Tipo de Entidade" que tem comandos. Um comando está associado a um dos seguintes "Tipos de Propriedades": função primitiva, função de outro tipo definido pelo usuário, seqüência simples, seqüência condicional e seqüência iterativa. E assim por diante.

A generalização desta idéia levou-nos a idealizar uma arvore de "Tipos de Entidades" e "Tipos de Propriedades" para definir o modelo de uma regra. Qualquer regra pode então ser instanciada com base neste modelo.

Em seguida esta concepção foi estendida para toda a estrutura do Modelo de Objetos Dinâmicos com a especificação completa do modelo de Tipos de Objetos e também do modelo dos Objetos que podem ser criados. O modelo de um Tipo de Objeto contém os modelos dos atributos e das regras. O

modelo de um atributo permite a definição da sua identidade e do seu tipo. O modelo de um Objeto permite a definição do Tipo de Objeto e dos valores dos seus atributos.

Baseado neste modelo o usuário final pode: alterar as regras do negócio instanciando novos tipos ou modificando os tipos existentes; ou executar as regras existentes através da instanciação de objetos dos tipos existentes.

3.1 Implementação do modelo

Um protótipo deste modelo esta sendo implementado usando a linguagem Java, o gerenciador de base de dados Microsoft Access e a ferramenta Sun ONE Studio.

Como mostrado na figura 2, este protótipo oferece uma interface amigável apresentando o modelo como uma arvore cuja raiz se ramifica em listas de tipos e listas de objetos.

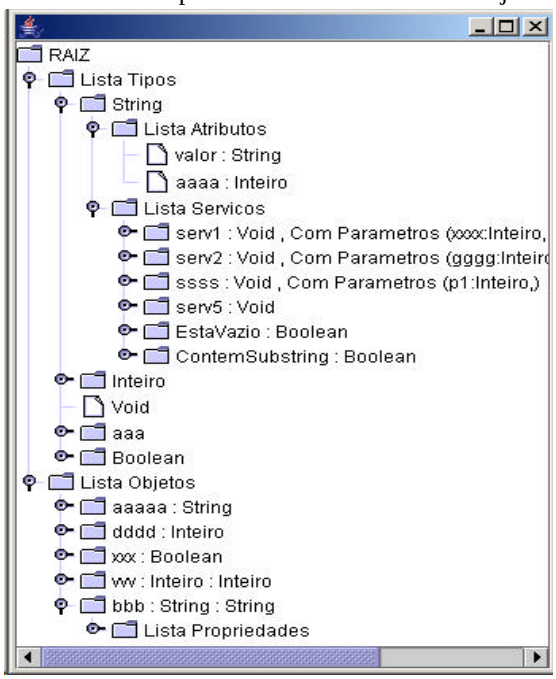

Figura 2 - Editor

Ao selecionar um item da arvore com o botão direito do *mouse* um menu é exibido, permitindo que o usuário, por exemplo, possa inserir um novo item ou modificar / eliminar o item selecionado.

Usando o ramo "Lista Tipos", mostrado na figura 2, o usuário pode adicionar novos tipos de objetos, seus atributos, serviços e procedimentos a serem executados pelos serviços. É também possível modificar ou eliminar um tipo ou alguma de suas propriedades.

O ramo "Lista Objetos" permite que o usuário possa adicionar novos objetos dos tipos definidos, definindo o valor de suas propriedades. De modo semelhante ao caso dos tipos é também possível eliminar um objeto ou modificar suas propriedades.

Todas estas operações instanciam, eliminam ou modificam objetos na memória e simultaneamente atualizam a base de dados.

Além da funcionalidade de edição do modelo este protótipo permite que o usuário possa também executar qualquer um dos serviços definidos pelo tipo do objeto. Neste caso, como mostrado na figura 3, são ativados seqüencialmente o serviço (SV), procedimento (PRC) e comandos (CMD) correspondentes. Lembrando que um comando pode ser um dos 3 tipos de seqüência (SEQ), uma função primitiva (PRI) ou um serviço de outro tipo de objeto definido pelo usuário (SV2).

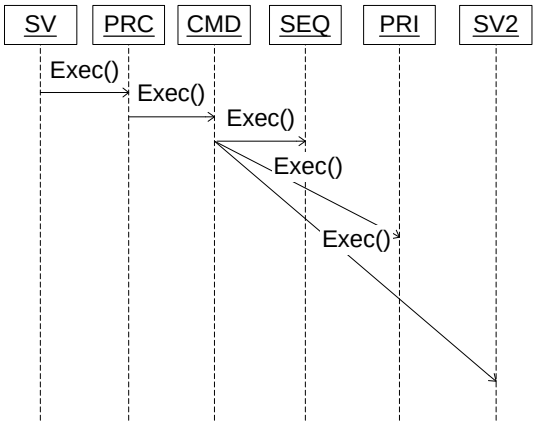

Figura 3 – Execução de um serviço

Na execução das funções primitivas é utilizado o pacote de classes reflexivas do Java. Isto possibilita que para adicionar ao protótipo uma nova primitiva seja suficiente ter o .class da nova classe primitiva e inserir um novo tipo de objeto, de acordo com os nomes da classe, dos serviços, e dos tipos de retorno e parâmetros definidos no arquivo .java correspondente.

A seguir são mostradas duas figuras, com objetivo de exemplificar as facilidades de edição que são oferecidas ao usuário.

A figura 4 mostra o dialogo que é exibido ao usuário quando ele deseja incluir um novo atributo a um tipo de objeto. Para definir o tipo do novo atributo ele pode selecionar somente um tipo já existente.

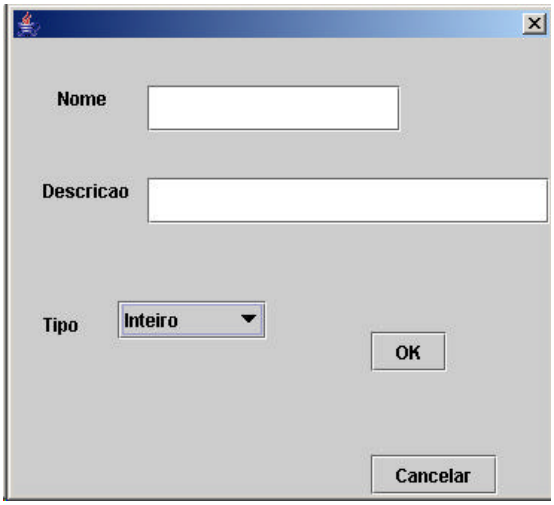

Figura 4 – Dialogo de Inclusão de Atributo

A figura 5 mostra um exemplo das facilidades oferecidas para definir o procedimento a ser executado por uma nova regra de negócio definida pelo usuário.

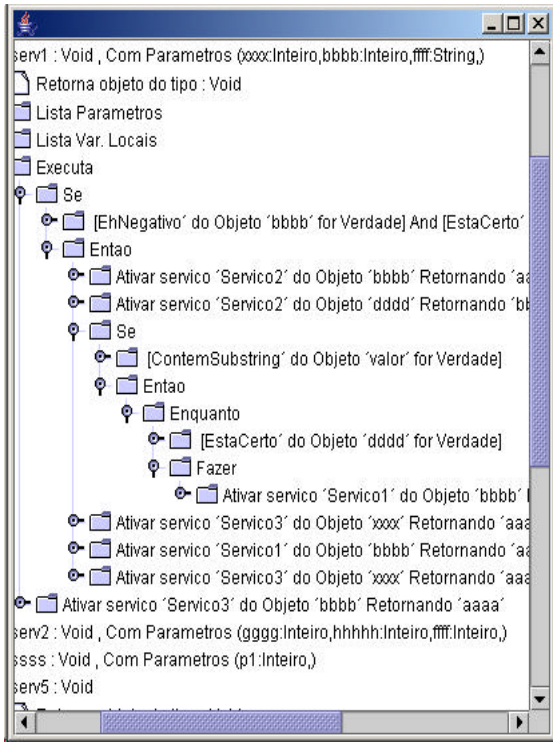

Figura 4 – Procedimento

4. Conclusão

A proposta de utilização da arquitetura DOM nos Sistemas de Controle do INPE esta apoiada na existência de um *framework* maduro e uma equipe com vários anos de experiência no domínio de aplicação.

A estrutura de objetos proposta e o protótipo em implementação indicam que estamos em um bom caminho no sentido de oferecer aos usuários finais uma ferramenta que lhes permita modificar objetos e seus comportamentos da forma mais natural possível.

O próximo passo deste projeto será a definição de primitivas de um sistema de processamento de telemetrias.

Adicionalmente o trabalho realizado gerou um *framework* do Modelo Objetos Dinâmicos que pode ser utilizado, em princípio, por qualquer domínio de problema pela simples adição das funções primitivas adequadas.

5. Referências

[1] A. Arsanjani, “Rule Object 2001: A Pattern Language for Adaptive and Scalable Business Rule Construction”, PLoP'2000 - Pattern Languages of Programming Conference, 2001.

[2] P.E. Cardoso et al, “Brazilian experiences in Upgrading a Satellite Control System”, IV International Symposium on Space Mission Operations and Ground Data Systems, Munich, Germany, 1996.

[3] P.E. Cardoso et al, “Lessons Learned in Adopting PCs at the Brazilian Satellite Control Center”, V International Symposium on Space Mission Operations and Ground Data Systems, SPACEOPS98, Tokyo, Japan, 1998.

[4] E. Gamma, et al, “Design Patterns – Elements of Reusable Object-Oriented Software”, Addison-Wesley 1995.

[5] L.S.C. Gonçalves et al, “Satellite Control Center: solution for an Adaptable System”, IV International Symposium of Small Satellites Systems and Services – SSSS, Antibes, France, 1998.

[6] R. E. Johnson, B. Woolf, “The Type Object Pattern”, PloP’ 96 - Pattern Languages of Programming Conference, 1996.

[7] R. Johnson, J. W. Yoder, “The adaptive object-model architectural style”, IEEE/IFIP Conference on Software Architecture 2002 (WICSA3´02). Canada, August 2002.

[8] D. A. Manolescu, R. E. Johnson, “Dynamic Object Model and Adaptive Workflow”, OOPSLA'99 Metadata and Active Object-Model Pattern Mining Workshop, 1999.

[9] W. Yamaguti et al, “Satellite Control System Nucleus for the Brazilian Complete Space Mission”, In Proceeding of the International Symposium on Ground Data Systems for Spacecraft Control, Darmstadt, June 26-29, 1990.

[10] J. W. Yoder, et al, “Architecture and Design of Adaptive Object-Models”, ACM Sigplan Notices, Vol. 36, pg 50-60, December, 2001.