Uma Arquitetura de *Software* Distribuída, Configurável e Adaptável Aplicada às Várias Missões de Controle de Satélites (SICSDA)

Adriana Cursino Thomé
Instituto Nacional de Pesquisas Espaciais
drithome@bol.com.br

Mauricio G. V. Ferreira
Instituto Nacional de Pesquisas Espaciais
mauricio@ccs.inpe.br

João Bosco S. Cunha
Universidade Federal de Itajubá
bosco@unifei.edu.br

Resumo

Propõe-se neste trabalho uma arquitetura distribuída, configurável e adaptável para o software de controle de satélites (SICSDA). O objetivo de se propor esta arquitetura é permitir que o controle dos vários satélites possa ser feito usando-se o mesmo conjunto de máquinas, possibilitando que se possa escolher qual dos satélites deseja-se monitorar em um determinado instante. Outro fator importante é a necessidade de se ter uma arquitetura que permita que uma nova missão possa ser acomodada sem a necessidade de se criar um sistema específico para o satélite a ser lançado, fazendo com que o esforço necessário para adaptar o sistema a esse novo requisito seja minimizado.

Abstract

This work proposes the creation of an adaptive configurable distributed software architecture to be applied for satellite control missions (SICSDA). The main purpose of this architecture is to control a satellite constellation through one set of machines, enabling the choice of each satellite to be monitored in any given period of time. This architecture allows a new mission to be settled without the need for the creation and addition of a specific software component for the satellite being launched, thus minimizing the effort needed to adapt the complete system to the new requirement.

1. Introdução

A grande extensão do país e a existência de imensas áreas com baixa densidade populacional, especialmente na região Amazônica, são características predominantes para justificar o uso de satélites como instrumentos de integração do território nacional, através de suas redes de comunicação, serviços de previsão de tempo e acompanhamento dos processos de uso do solo. Estas condições fisiográficas, aliadas à prática adquirida no estudo e na utilização de técnicas espaciais, foram motivos suficientes para se iniciar um programa de desenvolvimento da tecnologia espacial no Brasil.

O Instituto Nacional de Pesquisas Espacias (INPE), como uma das principais organizações envolvidas na evolução tecnológica espacial brasileira, assumiu a responsabilidade do lançamento e controle dos primeiros satélites brasileiros. O programa espacial brasileiro compreende o lançamento de quatro satélites, sendo os dois primeiros utilizados para coleta de dados, e os outros, para sensoriamento remoto (Ferreira,2001).

Destes quatro satélites, dois satélites de coleta de dados, o SCD1 (Sistema de Coleta de Dados 1) e o SCD2 (Sistema de Coleta de Dados 2) já foram lançados; e um satélite de sensoriamento remoto, o CBERS1 *(China Brazil Earth Research Satellite 1)*, também já foi lançado. O próximo satélite a ser lançado será o satélite de sensoriamento remoto CBERS2 *(China Brazil Earth Research Satellite 2)*, que tem seu lançamento previsto para o ano de 2003.

Para gerir todas as peculiaridades inerentes ao controle de um satélite, o INPE criou uma infra-estrutura robusta, composta pelo Centro de Controle de Satélites (CCS), por duas estações de rastreamento (Estações de Cuiabá e Alcântara), por uma rede de comunicação (RECDAS) e por aplicativos de software para o controle de satélites (SICS), como mostra a Figura 1. As estações remotas localizadas em Cuiabá e Alcântara oferecem, juntamente com o CCS, o suporte em terra para o contro1e dos satélites em órbita.

Para cada satélite já lançado foi desenvolvido um aplicativo de software que permite o seu monitoramento em terra. Isso é necessário, pois cada satélite tem características próprias, que normalmente variam, mesmo que sutilmente, de um satélite para outro. O acesso a esses aplicativos está restrito aos controladores de satélites fisicamente localizados no CCS no INPE em São José dos Campos.

Cada satélite lançado necessita, portanto, que seja destinada a ele uma máquina ou um conjunto de máquinas específicas onde um aplicativo específico para aquele determinado satélite é executado, auxiliando no recebimento de seus dados e monitoramento de seu estado interno.

Isto significa que para cada novo satélite a ser lançado, um aplicativo deverá ser desenvolvido ou adaptado para aquele satélite em especial, e máquinas deverão ser destinadas à execução desse *software* específico, implicando em um custo de desenvolvimento adicional a cada novo lançamento, tanto em termos de *hardware*, quanto em termos de *software*.

Este contexto nos leva a pensar na criação de um único sistema de *software* para o controle de satélites, que permita que os diferentes tipos de satélites possam ser monitorados de uma mesma máquina ou de um mesmo conjunto de máquinas.

Ainda assim, a necessidade de uma adaptação no sistema, por exemplo, para incorporar características de um novo satélite a ser monitorado, traria uma série de dificuldades para a adaptação do software, ocasionando um grande esforço despendido para que as novas características pudessem ser incorporadas ao sistema de forma que a qualidade do mesmo fosse mantida.

Todas estas questões aliadas ao desejo crescente de se obter aplicações que evoluam à medida que o domínio evolui, fez com que se pensasse em construir aplicações mais configuráveis, flexíveis e adaptáveis, permitindo que o sistema pudesse se adaptar, mais facilmente, às novas necessidades do domínio, acompanhando a evolução dos requisitos, porém, mantendo sua qualidade.

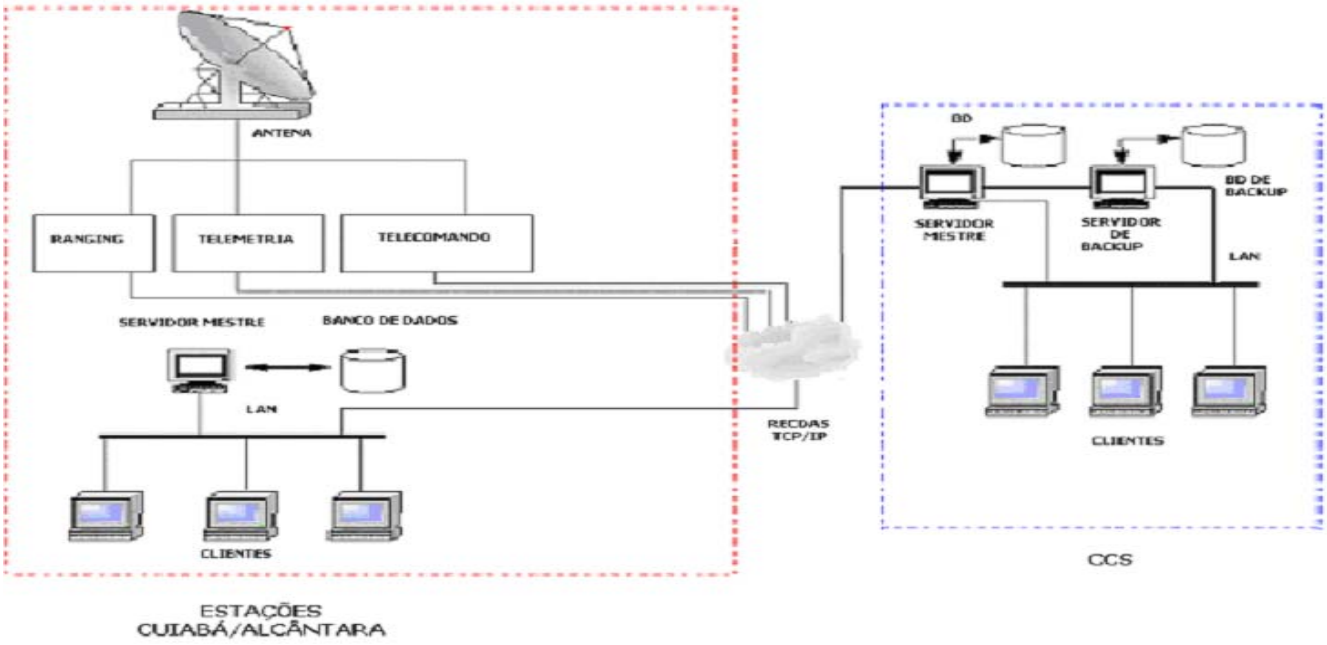

Figura 1- Arquitetura simplificada do sistema de rastreio e controle de satélites.
FONTE: adaptada de Rozenfeld(2002).

Uma forma de se conseguir isto é mover certos aspectos do sistema, como regras de negócio, por exemplo, para o banco de dados, fazendo com que dessa forma, elas possam ser facilmente modificadas. O modelo resultante permite que o sistema possa se adaptar rapidamente às novas necessidades do domínio através de modificações nos valores armazenados no banco de dados, ao invés de modificações no código (Yoder,2001).

Isto encoraja o desenvolvimento de ferramentas que permitam que os tomadores de decisão e administradores introduzam novos elementos ao *software* sem a necessidade de programação adicional, e que façam mudanças em seus modelos de domínio em tempo de execução, reduzindo significantemente o tempo para incorporação de novas idéias ao *software*.

Arquiteturas que podem dinamicamente se adaptar em tempo de execução a novos requisitos de usuários são chamadas de "arquiteturas reflexivas" ou "meta-arquiteturas". A filosofia básica de projeto para uma arquitetura reflexiva é expor aos desenvolvedores, ao invés de esconder, os elementos que compõe a estrutura do sistema. Um sistema reflexivo é introspectivo; ou seja, ele tem uma representação de si mesmo que pode ser observada (auto-representação). Freqüentemente, essa auto-representação é expressa em termos de entidades abstratas que podem ser manipuladas para modificar o

comportamento do sistema. Então, um sistema reflexivo promove a escrita de componentes genéricos e reusáveis que manipulam essa auto-representação (Nguyen-Tuong,1999).

O modelo reflexixo sugere um novo paradigma para o desenvolvimento de sistemas. Nesse novo paradigma, um sistema é decomposto em pelo menos dois níveis: **(1)** o nível base, onde estão agrupadas as funções relacionadas exclusivamente ao domínio do sistema e o **(2)** nível reflexivo, que agrupa o código que supervisiona, adapta e retorna informações sobre o nível base (Stehling,1999).

Uma "arquitetura de modelos de objetos adaptáveis *(Adaptive Object_Model Architecture)*" é um tipo particular de arquitetura reflexiva que abrange sistemas orientados a objeto que gerenciam elementos de algum tipo, e que podem ser estendidos para adicionar novos elementos. Sistemas que têm este tipo de arquitetura podem ser chamados de "Modelos de Objetos Ativos *(Active Object-Models)*", "Modelos de Objetos Dinâmicos *(Dynamic Object Models)*", ou "Modelos de Objetos Adaptáveis *(Adaptive Object-Models)*" (Yoder,2001).

Dessa forma, um modelo de objetos adaptável é um sistema que representa classes, atributos e relacionamentos como sendo metadados. Os usuários modificam os metadados (modelo de objetos) para refletir as mudanças no domínio. Essas mudanças modificam o comportamento do sistema. Em outras palavras, o sistema armazena o modelo de objetos em um banco de dados e o interpreta. Conseqüentemente, o modelo de objetos é ativo, e quando ele é modificado, o sistema muda imediatamente (Yoder,2001).

Assim sendo, os metadados são usados em modelos de objetos adaptáveis para descrever o modelo em si (auto-respresentação). Desde que o sistema consiga interpretar os metadados para construir e manipular as descrições das classes do modelo em tempo de execução, torna-se fácil adicionar novas classes ao modelo de objetos adaptável, e torná-las imediatamente disponíveis para os usuários (Yoder,2000).

Usar a abordagem dos modelos de objetos adaptáveis (AOMs) no desenvolvimento de sistemas pode amenizar alguns dos problemas que vêm sendo encontrados pelos desenvolvedores de *software*, principalmente em relação à flexibilidade, evolução e manutenção do sistema, permitindo que o custo total do desenvolvimento e manutenção possa ser reduzido drasticamente (Ledeczi,2000).

Baseando-se nos fatores anteriormente descritos, elaborou-se a proposta de Tese de Doutorado em Computação Aplicada descrita neste trabalho, intitulada "Uma Arquitetura de *Software* Distribuída, Configurável e Adaptável Aplicada às Várias Missões de Controle de Satélites (SICSDA)".

2. Objetivo da Arquitetura SICSDA

O objetivo desta arquitetura é permitir que o controle dos vários satélites possa ser feito usando-se o mesmo conjunto de máquinas, possibilitando que se possa escolher qual dos satélites deseja-se monitorar em um determinado instante. É importante ressaltar que não é possível, pelo menos por enquanto, monitorar mais de um satélite por vez, já que apenas a estação de Cuiabá encontra-se em operação.

Outro fator importante é a necessidade de se ter uma arquitetura que permita que uma nova missão possa ser acomodada sem a necessidade de se criar um sistema específico para o satélite a ser lançado, fazendo com que o esforço necessário para adaptar o sistema a esse novo requisito seja minimizado.

Adicionalmente, deseja-se que especialistas do domínio e usuários possam configurar a interface humana do sistema. Dessa maneira, a interface pode ser moldada para o formato mais apropriado ao perfil de cada um, adequando-se, assim, às suas necessidades.

2. A Arquitetura SICSDA

A arquitetura SICSDA modela a aplicação para o controle de satélites baseando-se nos modelos de objetos adaptáveis (AOMs). Isso significa que nesta arquitetura os objetos do domínio do problema, por exemplo; telemetria, telecomando, "*ranging*"; ao invés de estarem localizados no código que implementa a aplicação, estarão implementados em um banco de dados orientado a objetos para que possam ser instanciados em tempo de execução. Isto significa que o sistema terá um código genérico implementado em uma linguagem orientada a objetos que será capaz de acomodar os diferentes modelos de objetos dos vários satélites.

A arquitetura SICSDA, principalmente por questões de tolerância a falhas, será distribuída, e as funcionalidades oferecidas pela aplicação; por exemplo, visualização de telemetria e emissão de telecomando; poderão estar distribuídas dentro de um domínio de rede pré-definido. Isto significa que os objetos da aplicação e poderão ser instanciados em máquinas diferentes na rede, ocasionando, portanto, uma distribuição do código do sistema. O serviço de carga do sistema definirá a localização desses objetos, ou seja, em que máquina da rede eles estarão disponíveis.

Isto significa que cada máquina da rede poderá ter uma "visão" do metadado armazenado no banco de dados. Entende-se por uma "visão", neste contexto, como sendo o trecho do modelo de objetos adaptável que será instanciado naquela máquina; ou seja, uma determinada máquina da rede terá acesso apenas à parte do modelo de objetos adaptável que lhe cabe, instanciando apenas os objetos pré-definidos pelo serviço de carga do sistema.

Além disso, a arquitetura sugere também o uso de bancos de dados distribuídos através da replicação do banco de dados em um ou mais nós da rede. É importante lembrar que será necessário também que se tenha mecanismos que auxiliem o controle e o armazenamento do histórico dos metadados, e que auxiliem o controle de versões.

Pode-se dizer que a arquitetura proposta será configurável porque os especialistas do domínio e usuários poderão re-configurar os objetos de interação humana, moldando a interface para o formato mais apropriado ao perfil de cada um, adequando-a, dessa forma, às suas necessidades.

Finalmente, a arquitetura SICSDA será adaptável porque será capaz de acomodar possíveis mudanças no domínio do problema, permitindo que se possa acompanhar a evolução dos requisitos do negócio, e adaptando-se às necessidades dos usuários. Ou seja, porque especialistas do domínio e desenvolvedores poderão adaptar o sistema para acomodar novas classes através da criação, em tempo de execução, dessas classes com seus atributos e métodos; e porque é possível, em tempo de execução, alternar entre os metadados dos diversos satélites, ocasionando uma instancialização de um novo modelo de objetos em cima do código genérico cada vez que uma troca de contexto desse tipo for requisitada pelo usuário, ou seja, cada vez que se desejar controlar outro satélite.

A Figura 2 ilustra a estrutura e o funcionamento da arquitetura SICSDA. A seguir são detalhados os elementos que aparecem na Figura 2:

- **Objetos da Aplicação:** objetos do *software* aplicativo que realiza o controle dos satélites (telecomando, *ranging*, telemetria). O objeto "telemetria" encapsula o estado interno do satélite, como por exemplo, a voltagem de um determinado circuito, o posicionamento de uma determinada chave (ligada ou desligada), como também os métodos inerentes ao objeto, tais como processar telemetria, visualizar telemetria, etc. O objeto "telecomando" encapsula os telecomandos que podem ser enviados para o satélite, bem como os métodos pertinentes ao objeto, tais como enviar telecomando, visualizar telecomandos, etc. O objeto "*ranging*" encapsula as medidas de distância do satélite em relação à Terra, bem como os métodos responsáveis por esse cálculo.
- **Serviço de Persistência:** responsável por armazenar e recuperar do banco de dados os metamodelos da aplicação e os metadados do sistema.
- **Serviço de Configuração:** responsável por manter os metamodelos da aplicação e os metadados do sistema, por definir as visões dos objetos distribuídos, e por garantir a integridade dos objetos distribuídos.
- **Simulador de Satélites:** *software* que simula a interação com os satélites, ou seja, que simula a chegada e envio de dados dos/para os satélites.
- **Serviço de Carga:** responsável por carregar os metamodelos dos satélites, necessitando para isso saber qual satélite será monitorado em um determinado instante. Será implementado através de um código genérico.
- **Serviço de Usuário:** responsável por oferecer ao usuário que utilizará a aplicação de controle de satélites a interface adequada.
- **Serviço de Conexão:** será responsável por prover a localidade de um objeto dentro da rede.

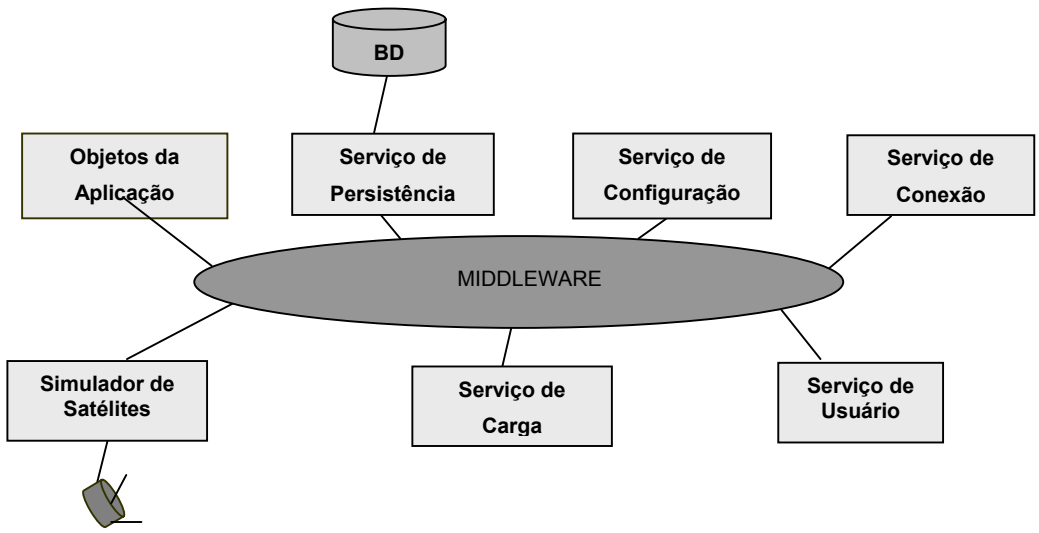

Fig.2 - Arquitetura SICSDA.

Como ilustra a Figura 2, o serviço de carga do sistema carregará o metamodelo do satélite que se deseja monitorar em um determinado instante. Para tal ele acionará o serviço de configuração que será responsável por definir as visões dos objetos distribuídos em cada nó para aquele determinado satélite. O serviço de

configuração acionará o serviço de persistência que será responsável por recuperar os metadados do banco de dados, povoando a aplicação com os objetos do satélite em questão, montando desta forma o modelo de objetos real da aplicação para aquele determinado satélite. O usuário poderá interagir com os objetos da aplicação através do "*middleware*". Os dados referentes aos satélites poderão ser obtidos através do Simulador de Satélites. Os objetos da aplicação estarão distribuídos, e o serviço de conexão será responsável por prover a localização de um determinado objeto no sistema. Quando o usuário desejar monitorar outro satélite ele informará o sistema, e este acionará o serviço de carga, que imediatamente carregará o novo metamodelo em cima do código genérico realizando as mesmas tarefas para povoar a aplicação com objetos.

Se os metamodelos do sistema precisarem ser alterados para refletir mudanças, o usuário configurador poderá, através de uma interface específica, realizar tais alterações. Quando estas alterações forem realizadas, o serviço de configuração deverá ser acionado, pois ele é responsável por manter os metadados. Para fazer isto ele acionará o serviço de persistência que imediatamente refletirá no banco de dados tais alterações.

Atualmente, após terem sido definidas a linguagem de programação orientada a objetos que será usada para construir o código genérico *(Java)* e a estrutura de armazenamento para os metadados *(InterSystems Caché)*, iniciou-se a fase de construção de um protótipo que implementa o código genérico que instancializa uma determinada aplicação cujos objetos de negócio estão armazenados no banco de dados. Carregados os objetos da aplicação, pode-se em tempo de execução, criar novas classes, atributos e métodos para essas classes.

A próxima etapa do trabalho, a curto prazo, será definir a plataforma de distribuição que será adotada para que se possa distribuir o código da aplicação, e assim que possível, incorporar as características inerentes à distribuição dos objetos no protótipo. Provavelmente, será utilizada a plataforma *J2EE*.

3. Resultados Esperados

Ao final do trabalho aqui proposto espera-se ter elaborado uma arquitetura distribuída, configurável e adaptável para o *software* de controle de satélites tendo como base os modelos de objetos adaptáveis.

As arquiteturas baseadas em modelos de objetos adaptáveis usualmente apresentam um número menor de classes do que as arquiteturas tradicionais, pois a informação que estava contida no código, agora está contida no banco de dados. Essa nova estrutura de classes não muda à medida que o sistema evolui, ao invés disso, as mudanças na especificação levam a mudanças no conteúdo do banco de dados.

Uma das maiores vantagens de se usar uma arquitetura baseada em modelos de objetos adaptáveis é a facilidade de se fazer mudanças no sistema, permitindo que, além dos desenvolvedores, os próprios especialistas do domínio estendam seu sistema dinamicamente, ou seja, permitindo que eles "programem sem programação".

Por outro lado, a possibilidade de se criar novas regras de negócio sem a necessidade de se intervir no código do sistema cria algumas dificuldades. Por exemplo, para que isso possa ser feito é necessário que seja construída uma linguagem específica do domínio, fazendo com que os desenvolvedores dos AOMs herdem todos os problemas associados ao desenvolvimento de qualquer linguagem, como por exemplo, construção de "*debuggers*", realização do controle de versões, e construção de ferramentas para documentação.

Além disso, para que os desenvolvedores e especialistas do domínio possam criar essas novas regras em tempo de execução, é necessário que se tenha uma interface interativa, sendo necessário que se tenha ferramentas e suporte para interfaces gráficas.

Em adição, as arquiteturas baseadas em modelos de objetos adaptáveis requerem a construção de um código genérico, que, na maioria das vezes, é difícil de ser construído, já que é necessário que se tenha um alto grau de abstração.

Outra questão importante é o desempenho. Como o modelo de objetos adaptável tem que ser interpretado para que se possa montar o modelo de objetos real e para que se possa refletir as mudanças realizadas, conseqüentemente, a execução do sistema é afetada, requerendo suporte de *software* e *hardware* adequado.

Adicionalmente, desenvolvedores não familiarizados com esse tipo de arquitetura tendem a achar que elas são mais difíceis de se entender e de se manter, já que existem dois sistemas co-existindo: o código genérico escrito em uma linguagem de programação orientada a objetos e o AOM que é interpretado. Por outro lado, desenvolvedores familiarizados com esse tipo de arquitetura acreditam que ela torna os sistemas mais fáceis de se manter, uma vez que eles têm menos linhas de código e que, tipicamente, uma pequena mudança no sistema pode causar uma grande mudança no sistema sendo executado.

Apesar de muitos aspectos apontarem para o fato de que, pelos menos inicialmente, as arquiteturas baseadas em AOMs exigem mais esforço para serem construídas, pretende-se com este trabalho dar um passo significativo em direção a reusabilidade de um sistema, já que os sistemas adaptáveis têm a característica de acompanhar a evolução dos requisitos do negócio, e já que têm uma arquitetura generalista, o que os torna altamente adaptáveis.

Outro passo importante pode ser dado em relação a alterabilidade de um sistema. Isso acontece porque o esforço para se realizar alterações no sistema pode ser

bastante minimizado, já que alterações no código podem ser reduzidas substancialmente. Adicionalmente, com este tipo de arquitetura pode-se permitir que os próprios especialistas do domínio realizem algumas alterações, melhorando a alterabilidade e diminuindo a intervenção dos desenvolvedores na evolução do sistema.

Além disso, espera-se também dar um passo em relação a economicidade, uma vez que futuras missões poderão aproveitar todo o investimento de *hardware* e *software* já feito em outras missões.

Embora a computação genérica e os modelos reflexivos já venham sendo explorados há algum tempo, os modelos de objetos adaptáveis ainda são pouco conhecidos, e apresentam, portanto, um grande potencial para ser explorado, especialmente, quando aliados ao conceito de distribuição. Isto acontece porque, apesar de algumas aplicações baseadas em modelos de objetos adaptáveis terem sido desenvolvidas, nenhuma das aplicações pesquisadas explorou o conceito de distribuição do código genérico; embora algumas tivessem citado a possibilidade de se usar bancos de dados distribuídos. Exemplos de aplicações já desenvolvidas usando-se os AOMs podem ser encontrados em Johnson(2002) e Yoder(2001).

Mais um indicativo de que os AOMs têm potencial para ser explorado foi dado em Poole(2001). Neste artigo os modelos de objetos adaptáveis são colocados como uma possível evolução à "*Model Driven Architecture (MDA)*", recentemente proposta pelo *OMG (Object Management Group)*. Mais detalhes sobre a *MDA* podem ser encontrados em Breton(2001) e OMG(2002).

Outro indicativo foi dado em Stehling(1999). Neste artigo o modelo reflexivo é visto como um novo paradigma para o desenvolvimento de sistemas. Assim sendo, existe potencial para explorar esse tipo de modelo sob o ponto de vista da engenharia de *software*, ou seja, provavelmente será necessário que se conceba uma nova forma de se desenvolver esse tipo de sistema, adequando metodologias e processos de desenvolvimento propostos pela área de engenharia de *software* a esse novo contexto.

O trabalho proposto une, portanto, áreas que até então eram exploradas de forma independente, trazendo para si uma característica multidisciplinar, ou seja, possibilitando a integração e colaboração de várias áreas. Além disso, o trabalho proposto aproveita esforços do passado através da elaboração de uma arquitetura que modela a aplicação para o controle de satélites com base nas arquiteturas SOFTBOARD e SICSD, propostas em Cunha(1997) e Ferreira(2001), respectivamente.

Espera-se, desta forma, colaborar, mesmo que de forma modesta, para o avanço da pesquisa no Brasil, e para o sucesso da missão espacial brasileira, oferecendo uma nova alternativa para a arquitetura do *software* para o controle de satélites, e, principalmente, abrindo novos campos de estudo em direção aos sistemas adaptáveis.

4. Referências Bibliográficas

[1]Breton, E.; Bézivin, J. Model driven process engeneering. **Proceedings of the IEEE Computer Society´s International Computer Software and Applications Conference.** USA. pg. 225-230. 2001.

[2]Cunha, J. B. S. **Uma abordagem de qualidade e produtividade para desenvolvimento de sistemas de software complexos utilizando a arquitetura de placa de software – SOFTBOARD.** Tese de Doutorado, Computação Aplicada. INPE. 1997.

[3]Ferreira, M. G. V. **Uma arquitetura flexível e dinâmica para objetos distribuídos aplicada ao software de controle de satélites.** Tese de Doutorado, Computação Aplicada. INPE. 2001.

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

[5]Ledeczi, A.; et al. Synthesis of self-adaptive software. **IEEE Areospace Conference Proceedings.** USA. V 4, pg. 501-507. 2000.

[6]Nguyen-Tuong, A.; Grimshaw, A. S. Using reflection for incorporating fault-tolerance techniques into distributed applications. **Parallel Processing Letters.** Vol. 9, N. 2, pg. 291-301. 1999.

[7]OMG. **Model Driven Architecture (MDA):** document number ormsc/2001-07-01. **On-line.** (http://www.omg.org). March of 2002.

[8]Poole, J. D. **Model Driven Architecture:** vision, standards and emerging technologies. **ECOOP´2001 Workshop on Metamodeling and Adaptive Object Models.** April of 2001.

[9]Rozenfeld, P.; Orlando, V.; Ferreira, M.G.V. Applying the 21th century technology to the 20th century mission control. **SPACE OPS 2002.** USA. October of 2002.

[10]Stehling, R. O. **Projeto e implementação de uma arquitetura de software reflexiva para linguagem Xchart.** Dissertação de Mestrado. Instituto de Computação da Universidade Estadual de Campinas. Março de 1999.

[11]Yoder, J. W.; et al. Architecture and design of adaptive object-models. **ACM Sigplan Notices.** Vol. 36, Fasc. 12, pg. 50-60, December of 2001.