Configuração automática de elementos em modelos heterogêneos de simulação

Alex Guimarães Cardoso de Sá[1], **Braulio Adriano de Mello**[2]

[1]Departamento de Ciência da Computação – Universidade Federal de Lavras (UFLA)
Lavras – MG – Brasil.

[2]Ciência da Computação – Universidade Federal da Fronteira do Sul (UFFS)
Chapecó – SC – Brasil.

{alexgsa@comp.ufla.br, braulio@uffs.edu.br}

Abstract. *The automatic configuration of heterogeneous simulation elements can improve the demanded time and reduce erros for the project of a model. However, this is a very complex task due to the variability of the elements used to make models. Thus, this work presents an automatic configuration tool for these kinds of elements performing inside the Distributed Co-simulation Backbone (DCB) architecture. For this job, the configurator handle informations about elements in a distributed data base to generate configuration files for each element which is integrated at the simulation model. The automatic building of these files is the focus of this work. This paper also presents a solution for integration of real elements in simulated models. This integration is advantaged when the elements have a complex behavior. During the study case, this paper shows a discussion about the correct variability precision representation to describe the systems behavior in the context of Precision Agriculture.*

Resumo. *A configuração automática de elementos heterogêneos de simulação contribui com agilidade e redução de erros na tarefa de construção de modelos. No entanto, esta é uma atividade complexa devido à variabilidade das características dos elementos utilizados na composição dos modelos. Este trabalho apresenta um módulo de configuração automática de elementos de simulação para a arquitetura DCB (Distributed Co-simulation Backbone). O configurador traduz informações do elemento, gerenciadas por um repositório distribuído de elementos, em arquivos de configuração, cujo formato é reconhecido pelos módulos de execução de modelos heterogêneos do DCB. A geração automática desses arquivos é o foco deste trabalho. O trabalho também apresenta uma solução para integração de elementos reais em modelos simulados. A integração de elementos reais é vantajosa quando a sua representação é muito complexa ou inviável. Nos estudos de caso o trabalho apresenta experimentos na representação da variabilidade de precisão em elementos usados na composição de modelos no contexto da Agricultura de Precisão.*

1. Introdução

As técnicas de modelagem e simulação de sistemas contribuem para identificar e corrigir problemas, diminuir custos de produção, detectar falhas antes da construção de protótipos (em sistemas novos) e validar o comportamento completo de sistemas. De acordo com os

recursos utilizados para a sua construção, modelos podem ser considerados homogêneos ou heterogêneos. Ao contrário de modelos homogêneos, os modelos heterogêneos utilizam partes que se diferenciam em termos de linguagem de descrição, interface e modo de tratamento do tempo. Essas características podem onerar a representação de sistemas reais que demandam a composição de partes (elementos) distintas. Essa dificuldade também está sofre influência da dinamicidade, distribuição e/ou precisão requeridas pelo modelo a ser construído.

Um dos grandes desafios encontrados na construção de modelos heterogêneos está na comunicação entre elementos simulados com tecnologias heterogêneas (distintas) de construção. Este desafio está afinado com o segundo dos grandes desafios da pesquisa em computação, a Modelagem de Sistemas Complexos. Grandes esforços podem ser gastos na tentativa de estabelecer meios de comunicação entre elementos que possuem partes distintas. A bibliografia retrata alguns estudos que propõem soluções para a simulação heterogênea. Dentre eles está a *High Level Architecture* (HLA) [IEEE 2009], que apresenta técnicas de interoperabilidade entre elementos (federados) de um modelo (federação). Porém, a interferência no código dos elementos, necessária para cooperação distribuída, pode gerar problemas difíceis ou impossíveis de tratar. Existem trabalhos, no contexto da HLA, que tem por objetivo evitar modificações nos códigos dos elementos [Tolk 2002].

Já a arquitetura do *Distributed Co-simulation Backbone* (DCB) [Mello et al. 2005], objeto de estudo desse trabalho, faz uso de técnicas de cooperação entre elementos heterogêneos (e/ou distribuídos) de simulação que preservam a integridade do elemento. A ideia central do DCB é que o elemento se torne independente do restante do modelo, comunicando-se apenas com seu *gateway* [Strassburger et al. 1998], que o preserva de primitivas de comunicação. O *gateway* é o módulo do DCB que controla a comunicação do elemento com o restante do modelo. A existência de um *gateway* privativo para cada elemento facilita o reuso mesmo sob restrições de propriedade intelectual [Page et al. 1999]. A partir de seus métodos é possível atribuir a responsabilidade das tarefas para os serviços internos do DCB. Outro aspecto que motiva o uso do DCB é a existência de um ambiente de suporte para gerenciamento de repositórios distribuídos de elementos [Mello and Parreiras 2009].

A solução apresentada neste trabalho permite gerar automaticamente os arquivos de configuração de elementos para composição de modelos reconhecidos pela arquitetura do DCB. Esses arquivos de configuração definem a cooperação realizada entre cada elemento do modelo do sistema. Entretanto, quando essa configuração é feita sem o uso de ferramentas que elevem o nível de abstração, a necessidade de conhecimentos sobre detalhes internos do ambiente de simulação consomem mais esforço do projetista para a integração de elementos. Isso torna a configuração manual trabalhosa e demorada, o que justifica o desenvolvimento de um configurador que reduz o esforço no trabalho de composição de elementos. O módulo de configuração apresentado neste trabalho está integrado ao Gerenciador de Repositórios Distribuídos de elementos [Mello and Parreiras 2009].

Como segunda parte da configuração de modelos, este trabalho apresenta um módulo que é responsável por gerar automaticamente os arquivos de controle de interface (*gateways*) de cada elemento. A construção desse segundo módulo é justificável pelo mesmo motivo, reduz a necessidade de conhecimentos mais detalhados da arquite-

tura do DCB, de programação e de tarefas realizadas pelos *gateways* (estrutura interna), para que possa construir um modelo de simulação. Além disso, esses arquivos preservam a implementação original dos elementos.

Porém, a representação de um sistema real através de modelos geralmente sofre redução de fidelidade ou perda de detalhes importantes. Há situações em que o sistema real possui partes com interface simples e com comportamento interno complexo de difícil representação. Nesses casos pode ser vantajoso integrar elementos reais em modelos onde as demais partes são simuladas, elevando o grau de heterogeneidade, além da composição de elementos simulados distintos em termos de interface, sincronização do tempo ou tecnologia de desenvolvimento.

Assim, de modo complementar à geração automática da configuração de modelos, este trabalho apresenta também um módulo que permite a integração de elementos reais em modelos simulados à luz da arquitetura do DCB. Experimentos realizados demonstram comparativos do comportamento entre as duas versões de um mesmo modelo, uma delas com todos os elementos simulados e a outra onde os elementos simulados cooperam com um elemento real.

O artigo esta organizado nas seguintes seções: a seção 2 explora as principais características da plataforma de simulação DCB e questões de projeto relacionadas ao Gerenciador de Repositórios Distribuídos. A seção 3 delimita os métodos de configuração mais comuns na literatura. Já na seção 4 é apresentado o configurador e, em seguida, seu estudo de caso realizado sobre o modelo específico para a Agricultura de Precisão. Considera-se na seção 5, a variabilidade dos níveis de precisão do sensor de passagens de sementes. Os estudos sobre a integração de elementos reais ao modelo simulado são apresentados na seção 6. Considerações finais e referências são apresentadas nas duas últimas seções.

2. Plataformas Utilizadas

Essa seção explora as principais características plataformas e seus respectivos componentes, no qual esse trabalho baseia-se. A primeira subseção define as características principais do DCB. E a segunda mostra as funcionalidades dos componentes presentes no Gerenciador de Repositórios Distribuídos de elementos.

2.1. O DCB

O DCB é uma plataforma de simulação que provê de recursos necessários para cooperação de elementos heterogêneos e/ou distribuídos. Todas suas atividades são transparentes ao componente simulado pelo fato de que todas elas são realizadas respeitando políticas internas do próprio DCB. Quatro módulos principais, identificados na Figura 1, são responsáveis pelo controle da cooperação entre elementos: o DCBS, o DCBR, o DCBK e o Gateway.

O DCBS realiza tarefas de envio de mensagens para outros elementos do modelo. O DCBR decodifica/recebe mensagens advindas de elementos origem. Já o DCBK realiza atividades para manter a comunicação entre elementos, respeitando o tempo global do modelo. A necessidade da utilização do Gateway está na comunicação de interfaces distintas dos componentes. A Figura 1 apresenta a estrutura geral do DCB. O único módulo

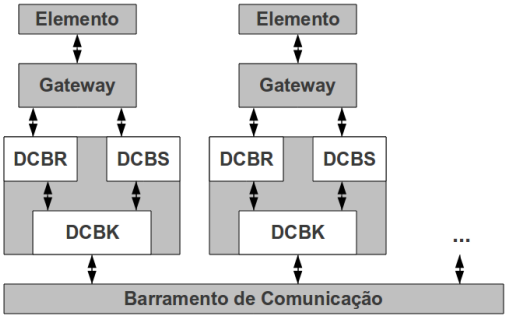

Figura 1. Arquitetura geral do DCB.

interno do DCB que executa operações remotas é o DCBK. Os demais realizam gerenciamento local para garantir a correta cooperação entre o elemento e o restante do modelo.

2.2. Gerenciador de Repositórios Distribuídos

O ambiente de gerenciamento de repositórios distribuídos de elementos foi desenvolvido com o principal objetivo de cadastrar, localizar e gerenciar elementos para facilitar a sua busca, identificação e seleção para que possam ser reutilizados em novos modelos. Os principais benefícios do gerenciador são a agilidade, a facilidade e a confiabilidade na construção e execução de modelos de simulação.

O cadastro de elementos no repositório é requisito para reuso na composição de novos modelos. Quando disponível no repositório, o elemento pode ser identificado pelas suas características e mais facilmente integrado/sincronizado ao modelo. Por isso, é recomendável que todos os elementos do sistema tenham código livre e aberto para ajuste de funcionalidades [Amory et al. 2002].

O módulo de cadastro está integrado ao Gerenciador de Repositórios Distribuídos e atende os requisitos da arquitetura do DCB. O trabalho de integração exigiu modificações no formato de armazenamento de informações sobre atributos e tipo de sincronização do elemento. A Figura 2 mostra a interface de cadastro de elementos e de seus respectivos atributos.

O registro de um elemento tem diversas informações associadas a ele, tais como: porta de entrada, IP, atributos de entrada e de saída e tipo de sincronização no tempo. As condições para geração automática do modelo dependem diretamente das informações sobre o elemento registradas no cadastro (repositório). O configurador, a partir dos dados de cadastro do elemento, gera arquivos em XML e *gateways* no formato identificado pelo DCB para carga e execução de modelos heterogêneos.

3. Métodos de Configuração

A configuração de elementos para a construção de modelos pode ser realizada manualmente ou empregando recursos computacionais para elevação do nível de abstração. A vantagem de recursos computacionais para esse fim recai principalmente sobre a redução de esforço e da necessidade de conhecimento técnico sobre o ambiente de simulação em uso.

Algumas das formas de automatizar etapas de configuração são, por exemplo a configuração por tabelas de vinculação, a configuração por meios de recursos gráficos e

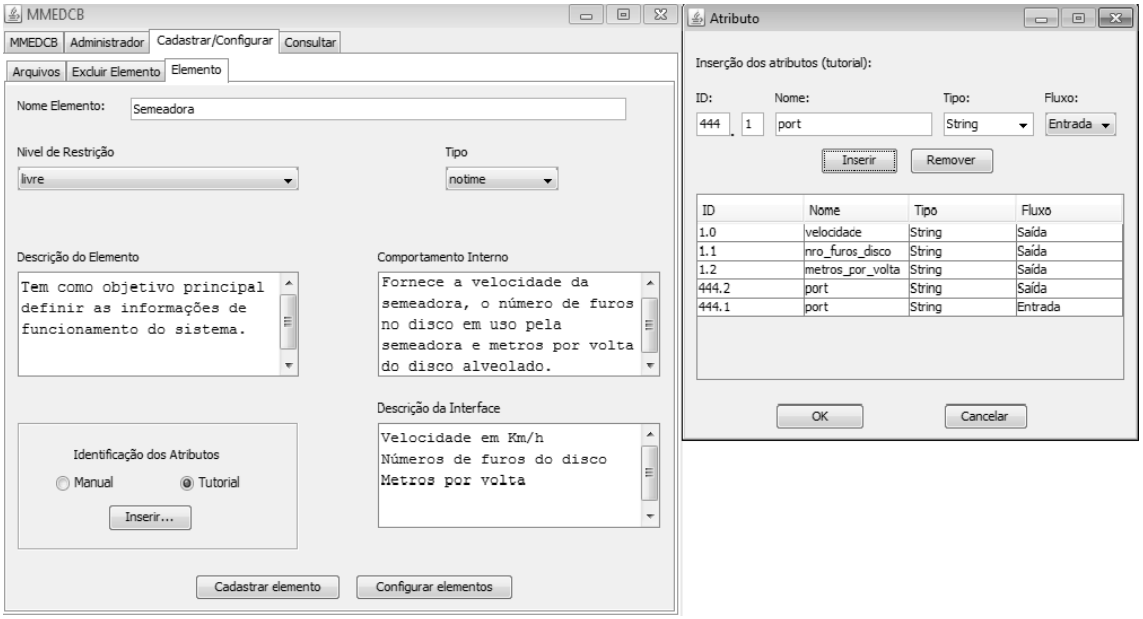

Figura 2. Cadastro de um elemento e de seus atributos no gerenciador.

a configuração por regras. O primeiro modo destaca-se por apresentar todas as peculiaridades do elemento a ser cadastrado em uma tabela, na qual define todas as vinculações de atributos de saída do elemento com os atributos de entrada de outros elementos. O configurador proposto nesse trabalho utiliza esse primeiro tipo de configuração. No segundo, os elementos são representados através de componentes gráficos e toda a configuração é feita por representações gráficas dos elementos e atributos que definem os esquemas para relacioná-los. São exemplos desse tipo de configuração o ambiente ASDA– Ambiente de Simulação Distribuída Automática [Bruschi 2002], o Tangram [Souza et al. 2003] e o wSimul [Merhi 2010] que visam, através de módulos gráficos, realizar a configuração do modelo. A terceira estratégia de configuração é o uso de um conjunto de regras as quais os elementos e os modelos devem respeitar para poder ser realizada a cooperação. A arquitetura HLA segue essa técnica.

4. Módulo de configuração automática de elementos

Esta seção apresenta o trabalho de especificação e construção do módulo de configuração automática. Um estudo de caso também é feito, onde o configurador foi utilizado para gerar os arquivos de configuração e os arquivos *gateways* do SEP (Supervisor Eletrônico de Plantio) [Mello and Caimi 2008]. Esses arquivos foram gerados para que o SEP possa ser executado no DCB usando apenas os arquivos gerados automaticamente pelo configurador. A possibilidade de geração automática do modelo completo sustenta os benefícios no uso de recursos computacionais com a finalidade de agregar agilidade, facilidade e correção aos modelos.

4.1. Persistência da Configuração de Elementos

O configurador permite gerenciar os atributos de entrada e saída dos elementos cadastrados no (s) repositório (s) reconhecidos pelo Gerenciador Repositórios Distribuídos, sejam eles remotos ou locais. Para que isso aconteça de forma transparente ao usuário projetista do modelo, uma interface gráfica é apresentada a fim de melhorar a integração

desse usuário com as tarefas do configurador. Assim, ações como a geração do arquivo em XML (configuração) do elemento que está sendo configurado são transparentes ao usuário. A Figura 3 mostra a interface gráfica de configuração de elementos.

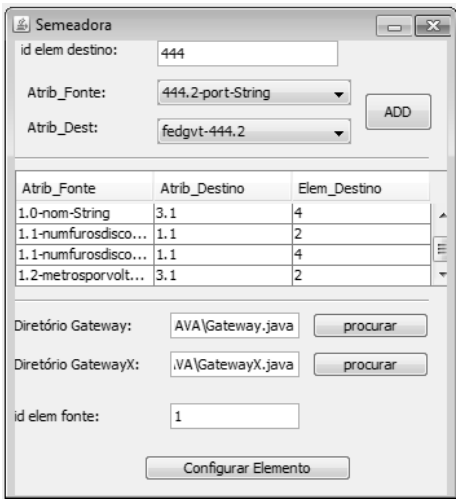

Figura 3. Interface Gráfica de configuração de elementos.

A configuração automática de elementos tem como requisito o cadastro prévio do elemento no banco de dados do Gerenciador de Repositórios Distribuídos de elementos. Esse banco de dados utiliza persistência em XML para registrar os novos elementos. Para os elementos serem configurados, os seus dados, em questão, são carregados pelo configurador que, então, permite ao usuário a visão dos atributos na forma de uma lista, como demonstrado na da Figura 3 e condiciona os dados dos elementos para que esses sejam manipulados corretamente. Para tornar a tarefa de configuração intuitiva, o configurador apresenta na interface, os atributos de saída do elemento em fase de configuração e os atributos de entrada dos demais elementos. O projetista vincula os atributos de saída do elemento (origem) aos atributos de entrada dos elementos (destino) conforme a especificação do modelo em construção. Essa vinculação tem por objetivo definir a cooperação entre dois ou mais elementos do modelo, que está em processo de design.

Para cada elemento escolhido pelo usuário, o configurador mostra uma nova interface como a da Figura 3. É possível, a partir disso, monitorar e controlar quais atributos de entrada estão conectados aos atributos de saída de cada elemento que está sendo configurado. A consistência das conexões entre atributos não é validada pelo configurador. O estudo, especificação e desenvolvimento de estratégias para garantir consistência das conexões é identificada como perspectiva de continuidade do trabalho.

A tarefa de selecionar o atributo fonte e destino pode ser feita repetidamente, para que ocorra vinculação de atributos e, consequentemente, a troca e mensagens entre esses elementos. O registro, das informações de identificação (id) dos elementos destino, é feito no momento da vinculação. Essas informações são adicionadas em uma tabela, para melhor visualização do projetista, com três colunas: a do atributo fonte (“Atrib_Fonte”), a do atributo destino (“Atrib_Dest”) e o id do elemento destino (“Elem_Destino”), que somente é válido se for um inteiro.

O configurador, após esse processo, já pode gerar o arquivo em XML de

configuração do elemento, necessitando apenas da identificação do elemento fonte (“id elem fonte”), essencial para cada elemento do modelo. Porém é ainda necessário esperar a geração dos arquivos de controle de interface para que o processo continue.

Este método de configuração de elementos é bem distinto do ASDA e de outros configuradores, que propõem submódulos de especificação gráfica de modelos. Porém, da mesma forma, esse ambiente busca afastar o usuário da tarefa de transcrição do modelo em um programa de simulação e também da análise estatística dos resultados mostrando as informações da configuração em uma simples tabela.

Todavia, como o módulo apresentado neste trabalho visa especificar detalhes de eventos peculiares à arquitetura do DCB. A composição das características de independência dos elementos, a distribuição geográfica sem que elementos necessitem executar ações de comunicação e a integração de elementos heterogêneos (reais ou simulados) identifica a contribuição do trabalho.

Um ponto relevante no estudo da arquitetura configurador proposto é sua similaridade em relação à outras aplicações que modelam eventos, como por exemplo o Composer [Guimarães et al. 2008], que é um *software* utilizado para mapear eventos para TV digital. O Composer também gera um arquivo no formato XML, bem como o configurador desenvolvido nesse trabalho, para definir quais são os eventos que irão ocorrer ou estão ocorrendo de acordo com a transmissora do canal. Esses eventos são feitos para promover a iteratividade entre usuário e TV. Esta característica demonstra que caso hajam modificações na estrutura de controle de eventos para o DCB, o configurador pode ser utilizado para outros fins.

4.2. Geração Automática dos Gateways

Além dos arquivos de configuração, o DCB requer o ajuste dos arquivos de interface denominados *gateways*. Cada elemento possui dois *gateways* particulares que reconhecem as portas de comunicação com o elemento identificadas na etapa de modelagem como atributos de entrada e saída. Para a geração automática dos *gateways* é necessário preencher os dois campos de diretórios, onde se encontram os *templates* dos *gateways*. Para cada novo elemento, o configurador deve gerar um arquivo GatewayX, onde X é o identificador do elemento, e realizar uma alteração no arquivo Gateway informando a existência do novo elemento. O DCB permite que múltiplos elementos sejam executados em *threads* distintas em uma mesma máquina virtual, por isso a existência de um arquivo Gateway para redirecionar mensagens ao destino correto, conforme a configuração do modelo.

Nos arquivos dos *templates* estão contidas informações que não se alteram dentro dos *gateways*, ou seja, esses arquivos são padrões ou modelos de *gateways*. O primeiro diretório contém o *template* do *gateway* genérico (Gateway), responsável pelas tarefas de inicialização e redirecionamento de tarefas para o *gateway* específico (GatewayX). O segundo diretório contém o *template* do *gateway* específico do elemento que cumprirá tarefas de atualização do tempo virtual local do elemento, conversão de tipos de atributos, caso seja necessário, e atualização de atributos. A ideia de *templates* de *gateways* foi inspirada dos trabalhos de [Sperb 2003] e de [Parreiras 2008]. O configurador também permite editar o conteúdo de ambos os arquivos de *gateways* para a inclusão de ações não previstas pelo configurador.

Preenchidos todos os campos de diretórios dos *templates* dos *gateways* e feita todas as conexões dos atributos fonte e destino, o elemento fonte pode ter sua identificação registrada em um campo da interface. O módulo tem em sua tela quase todas essas informações relevantes que podem ser passadas a um método. Com a adição dos atributos de entrada do elemento como parâmetro desse método é possível gerar o arquivo em XML de configuração desse elemento e seus respectivos *gateways*, seguindo um padrão de modelagem.

Ao final da configuração, um arquivo de saída é gerado para cada elemento, em um formato XML padronizado e dentro de um diretório previamente selecionado pelo usuário, onde constam as informações de cooperação entre elementos do modelo. Também são abertas duas janelas para cada elemento, uma com o *gateway* genérico e outra com o específico, para que possam ser feitas modificações quando os elementos apresentam características não previstas ou exceções. Neste caso é necessário intervenção do projetista no código dos *gateways*, antes de salvá-los como arquivos. Os códigos fontes dos *gateways* são escritos na linguagem de programação Java.

O uso do configurador reduz drasticamente a complexidade do trabalho de construção de modelos. Muitos dos detalhes relacionados ao modo como o vínculo entre atributos deve ser feito são mantidos consistentes pelo configurador. Isso reduz a probabilidade de erros. Outra vantagem é a possibilidade de configurar o comportamento (cooperação) entre múltiplos elementos simultaneamente. Não é necessário concluir a configuração de um elemento para iniciar a configuração do próximo. Esta característica adiciona flexibilidade sem perda de consistência ao trabalho do projetista de modelos.

4.3. Geração automática dos arquivos do modelo SEP

Para validar o configurador proposto, foram gerados automaticamente os arquivos de configuração e os arquivos *gateways* de um modelo de supervisor eletrônico de plantio, o SEP. Isso foi feito para que esse modelo fosse executado no DCB, utilizando somente os arquivos gerados pelo configurador. A Figura 4 mostra a arquitetura geral do SEP. O SEP possui quatro elementos principais: a Semeadora, o Disco Alveolado, o Sensor de Passagem de Sementes e o Centro de Controle.

O modelo construído para realização dos experimentos possui dez unidades do conjunto de elementos formados pelo disco alveolado e pelo sensor. Cada unidade possui autonomia na execução de ações e enviam dados de funcionamento do conjunto disco-sensor para o único elemento de controle (centro de controle) do sistema. Primeiramente, os elementos foram modelados e cadastrados no Gerenciador de Repositórios Distribuídos de elementos.

Em seguida, o configurador automático foi utilizado para compor as regras de cooperação entre os elementos do SEP e geração dos arquivos de configuração e interface (XML e *gateways*). Pequenos ajustes foram realizados no arquivo do DCB que identifica o número de diferentes elementos disparados em *threads* distintas da mesma JVM.

Depois de gerados os arquivos de configuração, o modelo foi executado no DCB tendo como entrada os dados cujos comportamentos eram conhecidos. Os resultados da execução confirmaram os resultados esperados. Além disso, a partir das abordagens utilizadas no configurador, os códigos dos elementos, os arquivos em XML e os arquivos *gateways* gerados com a configuração não foram modificados.

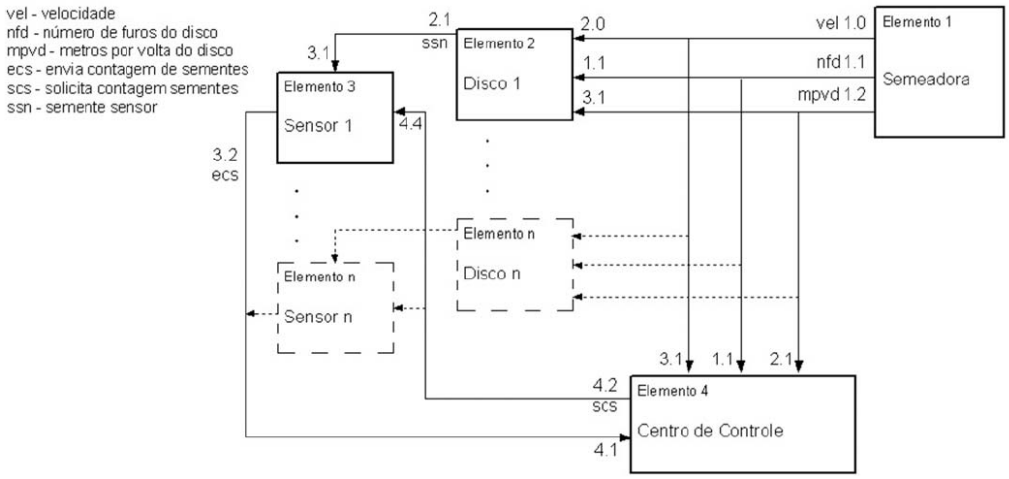

Figura 4. Arquitetura do SEP.

A configuração dos elementos foi feita também manualmente como indicador comparativo do esforço necessário em comparação à configuração automática de modelos. A partir disso pode-se comprovar a redução expressiva do esforço para o projetista do sistema simulado (ou usuário). Pode ser percebido também que a reutilização de componentes é uma das vantagens do gerenciador. A maior facilidade de incluir itens ao modelo a ser criado e em seguida configurá-los de acordo com as preferências do usuário beneficia a fidelidade em relação ao sistema real.

5. Variabilidade dos níveis de precisão do sensor de passagem

Um dos desafios na representação por modelos de sistemas para agricultura de precisão é, justamente, a avaliação dos níveis de precisão dos componentes do sistema. As condições severas de trabalho dos equipamentos e o excesso de resíduos (de qualquer tipo) comprometem expressivamente os níveis de precisão. Para isso, o modelo foi alterado para integrar recursos capazes de estimar a variabilidade na precisão das medidas. A independência entre elementos, característica da arquitetura do DCB, tornou esta tarefa simples. O elemento e o *backbone* do DCB não foram alterados.

O método responsável pelo envio de contagens parciais de sementes para o centro de controle recebeu uma alteração para monitorar e controlar a variabilidade da precisão usando distribuição de probabilidade. A distribuição de Poisson foi utilizada para representar a variabilidade de precisão. Em estudos que aplicam técnicas de simulação em sistemas que demandam altos custos, como a agricultura de precisão, essa distribuição de probabilidade tem sido sugerida [Chowdhury and Koval 2005].

Para o cálculo de lambda, parâmetro necessário ao modelo de Poisson, foram estimados 15 anos de atividade com tempo de inatividade em 30 porcento. Foram utilizados dados de estimativas de estudos já realizados sobre a avaliação de desempenho em equipamentos agrícolas [Pacheco 2000]. Neste caso, para o experimento realizado, o lambda teve valor igual a 3,33. Para o cálculo dos dos fatores que representam a perda de precisão do modelo foi utilizado o algoritmo da Figura 5.

O algoritmo de Poisson segue a ideia de iterar uma variável qualquer (“auxiliar”),

```
Algoritmo Poisson (lambda){

    limite = e^(-lambda);
    auxiliar  = 1;
    interações = 0;

    Faça{
      iterações = iterações + 1;
      auxiliar = auxiliar * Sorteia-Nro-Real(0,1);
    Enquanto (auxiliar > limite);

  Retorna (Iterações – 1);
}
```

Figura 5. Algoritmo para simular variabilidade na precisão dos sensores.

multiplicando-a por um número real aleatório entre zero e um (“Sorteia_Nro_Real(0,1)”), até que essa variável seja maior que o limite calculado durante a inicialização (“limite”). Para calcular esse limite, o algoritmo recebe como parâmetro o valor de lambda. O número de iterações (“iterações”) menos uma unidade é o valor do tempo de espera do sensor para enviar a contagem do número de sementes para o centro de controle. Esse fator representará a perda de precisão do sistema em campos agrícolas. No modelo, essa característica foi colocada em duas unidades disco-sensor.

Os dados utilizados na configuração da semeadora para o experimento foram: velocidade de 12km/h, 20 furos em cada disco alveolado e uma volta do disco significando 4,5 metros percorridos. Com isso, são 4,444 sementes esperadas por metro e média monitorada de 4,410 incluindo os recursos de variabilidade na precisão do sensor de sementes. Dessa forma, os testes realizados sobre o modelo do SEP obtiveram os resultados contidos na Tabela 1. A segunda coluna da tabela mostra os resultados para a representação da perda de precisão (RPP) do sensor em função das condições severas de trabalho. Esses resultados obtiveram uma média de 4 segundos e um desvio padrão de 1,414. Já para os tempos de quedas para 2000 sementes, a média dos tempos foi de 65,5 segundos e seu desvio padrão foi de 0,972.

Tabela 1. Testes de simulação do modelo.

Testes	RPP (s)	Tempo (s) p/ 2000 sementes	Perda de desempenho
1	4	67	10,00%
2	4	67	10,00%
3	2	65	8,00%
4	4	67	10,00%
5	4	67	10,00%
6	2	65	8,00%
7	6	68	12,00%
8	5	66	9,00%
9	6	67	12,00%
10	3	66	9,00%

Para calcular a porcentagem de perda de desempenho dos modelos (quarta coluna), considerou-se antes da inclusão do código que reduz a precisão do modelo, testes para prever qual é o tempo necessário para passarem 2000 sementes nos sensores. O tempo exato foi de 60 segundos.

6. Integração de elementos reais em modelos simulados

A integração de elementos reais com elementos simulados contribui com a fidelidade do modelo em relação ao sistema real e reduz o esforço de construção do modelo. O uso de elementos reais é especialmente apropriado quando a imitação do elemento é muito difícil ou inviável. A provável necessidade de ajustes proprietários na interface entre o elemento real e o modelo é um exemplo de dificuldade para integração de elementos reais. O protocolo de comunicação na interface do elemento real também não pode ser alterado ou simplificado. Se um elemento real gera um volume grande de dados constantemente informações importantes podem ser perdidas durante a transmissão.

Problemas de integração e tratamento de dados no uso de elementos reais podem ser resolvidos com recursos que certificam o recebimento e envio de dados. Um protocolo específico para este fim viabiliza esta solução. O uso de um protocolo padrão também contribui com soluções para detectar eventuais problemas de funcionamento do elemento real, por exemplo, implementando políticas de tempo limite de resposta do elemento (*timeout*). Em [Reynolds 1998] são apresentados alguns dos principais problemas envolvendo a integração de elementos reais em modelos de simulação.

A arquitetura do DCB favorece a integração de elementos reais, no entanto, exige a construção de uma interface específica entre o elemento e o *gateway* do DCB, para que ambos consigam se comunicar adequadamente. Esta seção apresenta uma solução para integração de elementos reais que combinam partes de *hardware* e *software* em modelos suportados pelo DCB. A solução é composta de uma controladora para interface de *hardware* e um protocolo de comunicação para gestão da cooperação entre sinais ou dados recebidos e/ou enviados pelo elemento real.

De acordo com [Fummi et al. 2009], não há metodologia consolidada para o uso de ambientes de simulação heterogênea (co-simulação) em função, essencialmente, do principal desafio que é a complexidade dos modelos [Oraw et al. 2007]. Em alguns casos, quando a representação do elemento real é inviável, a integração do elemento real ao modelo onde os demais elementos são simulados é uma alternativa que substitui o esforço de representação de um elemento complexo pelo trabalho de integração ao modelo.

A integração de elementos reais em modelos simulados ainda é pouco explorada como recurso para tratamento de problemas complexos. Por isso, os ambientes de simulação dedicam pouca ênfase para esta alternativa. O ambiente de interesse deste trabalho, o DCB, proporciona facilidades para a integração de elementos reais em modelos simulados, pois preserva a independência de elementos em relação ao seu comportamento interno e ao uso de recursos de comunicação para simulação distribuída. Toda cooperação do elemento com o ambiente externo ocorre localmente com o uso de portas de comunicação gerenciadas por componentes dedicados do DCB, exclusivas ao tratamento de interface. A flexibilidade para adequação do DCB à interface do elemento também é considerada.

No entanto, a integração de elementos reais exige recursos não comuns para a integração de elementos simulados. Sistemas embarcados, atuadores, sensores, entre outros, são exemplos de elementos reais que exigem recursos intermediários de interface não necessários para integração de elementos simulados essencialmente baseados em tecnologias de descrição de comportamento (linguagens).

6.1. Módulo para integração de elementos reais

No contexto do presente estudo foi desenvolvido um trabalho que demonstra a integração de elementos reais ao DCB [Chaves et al. 2009]. O principal objetivo foi construir um módulo de entrada e saída (*hardware* e *software*) capaz de cooperar com interfaces de comunicação tais como RS232, porta paralela e USB de um lado, com o *gateway* do DCB do outro. O recurso foi validado na integração de um sensor real de passagem de sementes do sistema de Supervisão Eletrônica de Plantio (SEP) já modelado.

O módulo possui uma controladora capaz de cooperar com elementos reais dentro de um conjunto reduzido de requisitos. A construção de um padrão genérico de controladora é muito difícil. Neste trabalho foi desenvolvida uma controladora que suporta a adaptação de sensores compostos por *leds* infravermelhos. Os *leds* geram tensão positiva quando detectam a passagem de uma semente.

Além da controladora, o módulo possui a parte de *software* que é um protocolo de comunicação entre elementos reais e DCB. Neste trabalho, o protocolo desenvolvido, baseado no MODBUS [Mobdus sd.], tem como objetivo principal padronizar as trocas de dados entre DCB e o elemento real. O protocolo permite controle sobre a comunicação, envio e recebimento de dados, execução de funções, e possui um padrão próprio de mensagem. Também é verificado pelo mesmo, a integridade das mensagens através do cálculo do CRC (*Cyclic Redundance Check*) [Griffiths and Stones 1987]. A Figura 6 apresenta o formato da mensagem utilizada pelo protocolo.

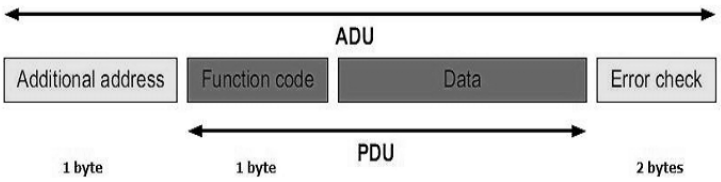

Figura 6. Formato das mensagens utilizada pelo protocolo.

O primeiro campo contém o endereço do elemento real de destino. O campo "Function code" carrega o código da função que será executada. O campo "Data" contém os dados necessários para a execução da função. E o campo "Error check" contém o código para verificação de erros.

O uso de um protocolo de comunicação para integração de elementos reais a modelos de representação facilita a integração de elementos, pois permite o reuso de soluções de integração para elementos similares ou que utilizam os mesmos padrões de comunicação. O uso de protocolo reduz a necessidade de modificações nos *gateways* do DCB e também o *overhead* na medida em que a parte de *hardware* do módulo de integração mantém uma tabela de endereços entre dois elementos reais permitindo que um elemento A possa cooperar com um elemento B sem interferência do DCB.

Experimentos foram realizados com a versão do SEP que integra um sensor real. Resultados equivalentes ao modelo simulado foram alcançados. Para isso, uma fonte de tensão foi utilizada como elemento real no lugar dos sensores infravermelhos como medida de controle sobre a frequência dos sinais enviados pelo elemento real.

7. Conclusões e trabalhos futuros

A configuração de modelos heterogêneos de simulação é uma tarefa complexa diante da necessidade de integração de elementos diferentes. A etapa de configuração define os critérios de cooperação entre elementos do modelo. Por exemplo, eventos que definem a troca de mensagens, especificação das origens e destino, tradução de dados para formato e tipos compatíveis com o destino, identificações dos elementos que receberão mensagens e atributos de saída serão utilizados. O configurador automático desenvolvido e apresentado neste trabalho aumenta o nível de abstração do trabalho de configuração de modelos para execução no DCB.

A relevância da configuração automática de elementos está em agilizar, facilitar e aumentar a confiabilidade do projeto de sistemas com o uso de recursos computacionais. Como resultado há redução de esforço do projetista, pois reduz a quantidade de detalhes no trabalho de configuração de elementos. Isso permite ao projetista preocupar-se com necessidades de maior importância, como por exemplo, a fidelidade do modelo em relação ao sistema real.

O configurador automático apresenta, ao projetista, uma interface contendo todas as informações sobre comportamento, atributos de interface, tipo de dados, e demais informações relevantes para o trabalho de modelagem. Assim, o configurador utiliza as relações de cooperação definidas pelo projetista na interface para gerar os arquivos de configuração em persistência XML e os arquivos *gateways* para tratamento da interface entre DCB e elemento.

Foram identificadas questões relevantes de projeto durante o desenvolvimento do gerador de arquivos de configuração (XML e *gateways*) e na construção do modelo SEP com dez unidades de disco-sensor realizada no estudo de caso. As soluções implementadas no configurador contribuíram com a geração dos arquivos de configuração com reduzida interferência do usuário projetista, facilitaram a modelagem e tratamento da variabilidade de precisão do modelo de simulação considerando o cenário da agricultura de precisão, criaram condições favoráveis para representar essa variabilidade e formas de gerenciar a quantidade de mensagens trocadas por tantos elementos em relação aos limites físicos do elemento real.

Estão previstos estudos para habilitar o módulo de configuração atual com funções para que ele também permita o cadastro de novas características nos *gateways*, reduzindo ou anulando a necessidade de modificações diretas no código. A versão atual do configurador gera os *gateways* de elementos simples. Outra funcionalidade relevante é a habilidade do configurador verificar se há inconsistências nas conexões entre atributos. Esta funcionalidade, posta como perspectiva irá contribuir para a geração de modelos mais corretos.

A integração de elementos reais ao modelo de simulação completa este trabalho. Pode ser necessário adaptar elementos reais ao modelo de simulação quando o elemento apresenta complexidade muito elevada. Com o desenvolvimento de um módulo que proporciona os recursos essenciais de comunicação ao modelo, essa integração foi realizada. Esse módulo possui uma controladora que oferece uma interface reconfigurável para cooperação de um elemento real com um simulado. A comunicação entre elementos reais e simulados é administrada por um protocolo que delimita regras para cooperação.

A utilização de um protocolo específico entre o modelo e o elemento real proporciona um fluxo de mensagens controlado entre emissor e receptor. Além disso, como a controladora desenvolvida possui uma interface configurável, ela permite que mais de um elemento real possa ser integrado ao modelo. Um estudo de caso comparando o modelo do SEP com elemento real e sem elemento real foi realizado. O comportamento de ambos foi equivalente.

No contexto dos grandes desafios da pesquisa em computação, um desafio específico é o esforço para simular mudanças ocorridas no mundo real em tempo de execução do modelo. O conjunto das contribuições apresentadas neste trabalho apontam, como perspectiva, para a busca de alternativas que permitam compor recursos ou ferramentas para viabilizar a construção, preferencialmente colaborativa, de modelos com condições de suportar a modificação dinâmica.

Referências

Amory, A. M., Moraes, F. G., Oliveira, L. A., Hessel, F. P., and Calazans, N. L. V. (2002). Desenvolvimento de um ambiente de co-simulação distribuído e heterogêneo. In *Proceedings of VIII Workshop Iberchip*.

Bruschi, S. (2002). *Um ambiente de simulação distribuída automático*. Tese de Doutorado, Instituto de Ciências Matemáticas e de Computação (ICMC) - Universidade de São Paulo (USP).

Chaves, T. M., Mello, B. A., and Caimi, L. L. (2009). Adaptação de elementos reais na interface do DCB para simulação de modelos heterogêneos. In *Proceedings of XXXV Latin-American Conference on Informatics - CLEI*.

Chowdhury, A. and Koval, D. (2005). Development of probabilistic models for computing optimal distribution substation spare transformers. In *IEEE Transactions on Industry Applications*, 41(6):1493 – 1498.

Fummi, F., Loghi, M., Poncino, M., and Pravadelli, G. (2009). A co-simulation methodology for HW/SW validation and performance estimation. In *ACM Transactions on Design Automation of Electronic Systems*, 14:23:1–23:32.

Griffiths, G. and Stones, G. C. (1987). The tea-leaf reader algorithm: An efficient implementation of CCR-16 and CCR-32. In *Communications of the ACM*, v.30:617-620.

Guimarães, R. L., Costa, Monteiro, R. M. R., and Soares, L. F. G. (2008). Composer: Authoring tool for itv programs. In *Proceedings of the 6th European conference on Changing Television Environments*, EUROITV '08, pages 61–71, Berlin, Heidelberg. Springer-Verlag.

IEEE. (2009). IEEE draft standard for Modeling and Simulation (M&S) – High Level Architecture (HLA) - Framework and Rules. *IEEE Unapproved Draft Std P1516/D5, Apr 2009*.

Mello, B. A. and Parreiras, A. A. (2009). Distributed management of elements for modeling and simulation of heterogeneous models. In *Summer Computer Simulation Conference (SCSC)*, pages: 49–53. Society for Modeling and Simulation International.

Mello, B. A., Souza, U. R. F., Sperb, J. K., and Wagner, F. R. (2005). Tangram: Virtual integration of IP components in a distributed co-simulation. In *IEEE Design and Test of Computers*, 22:462–471.

Mello, B. A. and Caimi, L. L. (2008). Simulação na validação de sistemas computacionais para a agricultura de precisão. *Revista Brasileira de Engenharia Agrícola e Ambiental*, 12:666 – 675.

Merhi, E. (2010). wsimul: Um sistema para simulação computacional. Em *Anais do Simpósio de Mecânica Computacional (SIMMEC)*.

Modbus (sd.). Modbus application protocol specification. Obtido em: www.mobdus.org.

Oraw, B., Choudhary, V., and Ayyanar, R. (2007). A co-simulation approach to model-based design for complex power electronics and digital control systems. In *Proceedings of the 2007 Summer Computer Simulation Conference*, SCSC, pages 157–164, San Diego, CA, USA. Society for Computer Simulation International.

Pacheco, E. P. (2000). Seleção e custo operacional de máquinas agrícolas. Relatório Técnico, Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA Acre), Ministéio da Agricultura e do Abastecimento.

Page, E. H., Buss, A., Fishwick, P. A., Healy, K., Nance, R. E., and Paul, R. J. (1999). Web-based simulation: Revolution or evolution? *In ACM Transactions on Modeling and Computer Simulation*, 10:3–17.

Parreiras, A. A. (2008). Geração automática da configuração de elementos para execução no DCB. Trabalho de conclusão de curso (graduação em Ciência da Computação), Universidade Regional Integrada do Alto Uruguai e das Missões.

Reynolds, P. F. (1998). Heterogeneous distributed simulation. In *Proceedings of the 1988 Winter Simulation Conference.*

Souza, U. R. F., Sperb, J. K., de Mello, B. A., and Wagner, F. R. (2003). Tangram – Virtual integration of heterogeneous ip components in a distributed co-simulation environment. In *Proceedings of the 16th Symposium on Integrated Circuits and Systems Design*, SBCCI '03, Washington, DC, USA. IEEE Computer Society.

Sperb, J. K. (2003). Geração de modelos de co-simulação distribuída para a arquitetura DCB. Dissertação de mestrado (Ciência da Computação), Universidade Federal do Rio Grande do Sul - UFRGS.

Strassburger, S., Schulze, T., Klein, U., Henriksen, J. O., and D-Magdeburg (1998). Internet-based simulation using off-the-shelf simulation tools and HLA. In *Proceedings of the 1998 Winter Simulation Conference*, pages 1669–1676.

Tolk, D. A. (2002). Avoiding another green elephant – A proposal for the next generation HLA based on the model driven architecture. In *Proceedings of the 2002 Fall Simulation Interoperability Workshop.*