



## UM FRAMEWORK ORIENTADO A OBJETOS PARA CONSTRUÇÃO DE SIMULADORES DE SISTEMAS DINÂMICOS

ALEX BOSSOLAN<sup>1</sup>; FELIPE ROSA<sup>2</sup>; ADAUTO MANCINI<sup>3</sup>; LUÍS BARIONI<sup>4</sup>

Nº 12613

### • RESUMO

O objetivo deste trabalho foi desenvolver um framework genérico e simples, que permita desenvolvimento modular, hierárquico e orientado a objetos de simuladores de sistemas dinâmicos. A primeira fase do trabalho constituiu no levantamento dos requisitos do software, o qual foi realizado no âmbito de dois projetos com necessidade de implementação de simuladores de sistemas complexos na agricultura, o projeto PECUS (<http://www.agropediabrasilis.cnptia.embrapa.br/web/pecus>) e o projeto ANIMALCHANGE (<http://www.animalchange.eu/>), procurando, a partir de exemplos da literatura, generalizar a estrutura para a simulação de sistemas dinâmicos. O framework permite que modelos e simuladores específicos possam ser implementados a partir de classes básicas que dão suporte à hierarquização, sincronização, transferência de dados por meio de conexões e portas entre os componentes do(s) modelo(s) e eventos do simulador. O framework suporta simuladores contínuos, orientados a eventos e híbridos. Eventos possuem ponteiros para métodos específicos de um objeto-alvo. Simulações podem ser configuradas por meio de uma interface que permite a definição de dados de entrada e eventos de forma genérica a partir do caminho hierárquico de um modelo componente ou para uma porta do modelo. O framework foi desenvolvido para ser compilado separadamente de outros componentes de softwares de simulação tais como banco de dados e interface gráfica.

---

<sup>1</sup> Estagiário Embrapa: Graduação em Análise e desenvolvimento de Software, FT - Unicamp, Limeira-SP, [abossolan@gmail.com](mailto:abossolan@gmail.com).

<sup>2</sup> Estagiário Embrapa: Graduação em Eng<sup>a</sup> da Computação, Unicamp, Campinas-SP, [felipe.rosa.sd@gmail.com](mailto:felipe.rosa.sd@gmail.com).

<sup>3</sup> Orientador: Pesquisador, Embrapa(CNPTIA), Campinas-SP, [adauto@cnptia.embrapa.br](mailto:adauto@cnptia.embrapa.br).

<sup>4</sup> Orientador: Pesquisador, Embrapa(CNPTIA), Campinas-SP, [barioni@cnptia.embrapa.br](mailto:barioni@cnptia.embrapa.br).



- **ABSTRACT**

The objective of this work was to develop a simple and generic framework, which allows modular and hierarchical objected oriented development of dynamic systems simulators. The current version of the framework was driven by the requirements raised in the scope of two research projects which involve simulation of agricultural systems: PECUS (<http://www.agropediabrasilis.cnptia.embrapa.br/web/pecus>), and; ANIMALCHANGE (<http://www.animalchange.eu/>). In the development, generalization of the structure was also achieved through examples from the literature. The framework reported herein allows that specific models and simulators are implemented through basic classes that support hierarchization, synchronization, data transfer through ports and connections between model components and simulation events. The framework supports continuous, event-oriented and hybrid simulations. Events have method pointer for an specific target object. Simulations inputs, events and properties are defined through an interface that allows generic definitions through the information of a hierarchical path of a component model or port in the model. The framework was developed to be compiled independently from other components of simulation software such as databases and graphical user interface.

- **INTRODUÇÃO**

Modelos de simulação são um meio conveniente para representar em termos abstratos uma parte da realidade consistente com nossos interesses específicos em aperfeiçoar nosso entendimento sobre um sistema. Representação é a palavra-chave: modelos de simulação devem prover-nos com uma representação abstrata da realidade. Como nós representamos o modelo conceitual que derivamos da observação da realidade deve então ser de considerável interesse para o modelista. Aqui é onde linguagens de programação e ferramentas de simulação desempenham um papel central.

Na formulação e descrição destes modelos, é útil e comum pensar em termos de modelos e submodelos (ou modelos componentes). Se uma tentativa é feita para se ter a estrutura de um programa de computador assemelhando-se à estrutura do modelo conceitual, usando sub-rotinas por exemplo, geralmente as partes do programa são fortemente amarradas e não podem ser substituídas facilmente por

partes equivalentes de outros programas desenvolvidos separadamente. Técnicas de orientação a objetos, em particular desenvolvimento de software baseado em componente, prometem obter um alto grau de correspondência entre modelos conceituais e sua implementação em código-fonte. Isto tem levado pesquisadores a desenvolver modelos orientados a objetos em uma variedade de domínios. (HILLYER, 2003).

Um framework orientado a objetos é um conjunto de classes colaborativas projetadas para serem expandidas para formar aplicações relacionadas. Um framework de simulação (FS), então, é um conjunto de classes a serem expandidas para criar programas de simulação. Tal FS pode descrever o fluxo de execução, padrões de comunicação, ou estruturas de dados usadas por elementos de simulação. A construção de um FS visa atender 2 propósitos: (1) Criar uma separação de interesses do cliente e do projetista do FS, pela segregação entre (a) as partes da aplicação-cliente (AC) específicas de simulação dos modelos do cliente e (b) as partes de código de infraestrutura do FS, contendo o núcleo para realizar tarefas comuns empregadas por muitas simulações, aumentando o reuso de código, e; (2) Criar um caminho claro para construir uma simulação. Definindo quais elementos do FS efetivamente contêm a implementação do modelo e como estes elementos são usados, um projetista vislumbra um caminho claro do modelo conceitual para a simulação. Definindo como os componentes do modelo interagem com outros componentes, um FS melhora a reusabilidade de modelos componentes, permite o desenvolvimento de facilidades robustas de metamodelagem (estimadores de parâmetros, capacidades de análise estocástica) e acelera a montagem e análise de modelos complexos. (HILLYER, 2003).

Neste contexto, com o objetivo de prover uma ferramenta de simulação para fácil implementação de modelos, reuso de componentes e também de fácil manutenção de código para incorporação de novas funcionalidades, está sendo construído o framework de simulação orientado a objetos detalhado nas seções seguintes. Este FS, em desenvolvimento no Laboratório de Matemática Computacional da Embrapa Informática Agropecuária, tem por objetivo geral ser uma ferramenta para uso genérico em simulação, e por objetivo específico, ser utilizado para a implementação de modelos de sistemas de produção pecuária considerando os fluxos de gases de efeito estufa por vários sistemas de produção pecuária nos diversos biomas brasileiros.



## • MATERIAL E MÉTODOS

A implementação do FS foi baseada em vários conceitos de simulação orientada a objetos descritos por Bolte (1998) e Ziegler (2000). Esses autores sugerem que a implementação de um modelo se dê a partir de unidades fundamentais, i.e. objetos de simulação, que representem entidades ou processos. Cada uma dessas unidades é implementada a partir da especialização de uma classe abstrata que padroniza a interface de comunicação e define o comportamento dos objetos da simulação. Essa abordagem prevê o estabelecimento de protocolos para comunicação e a interação entre os objetos, de forma que objetos de simulação possam ser adicionados ou retirados sem impacto significativo sobre o restante da estrutura do simulador. A existência de uma classe abstrata permite também que classes controladoras da infraestrutura, que sincronizam a execução da simulação e gerenciam eventos, por exemplo, gerenciem os objetos de simulação de forma genérica. Finalmente, tais classes estabelecem formas padronizadas para o acesso aos dados gerados ao longo da simulação, para geração de relatórios e gráficos, por exemplo, e de comunicação de exceções durante a execução da simulação.

O projeto foi elaborado de forma colaborativa, por meio de reuniões frequentes e apresentações, de forma que membros não-permanentes, que não possuíam conhecimento prévio sobre simulação, pudessem adquirir entendimento de conceitos relacionados à simulação orientada a objetos. A modelagem de classes durante o projeto do FS foi auxiliada por diagramas UML<sup>5</sup> (Unified Modeling Language), particularmente os diagramas de classes e sequência. O processo de desenvolvimento foi iterativo, sendo que duas versões intermediárias foram geradas. Padrões de codificação e especificação de modelos foram definidos com o objetivo de facilitar a compreensão e manutenção do código. Utilizou-se um software gerenciador do projeto para planejamento e acompanhamento das atividades. A implementação está sendo feita em linguagem C++ utilizando o IDE NetBeans<sup>6</sup> e o controle de versões pela ferramenta subversion<sup>7</sup>.

<sup>5</sup> Linguagem de modelagem não proprietária que procura padronizar a representação visual de elementos de software auxiliando a visualizar a comunicação entre os mesmos.

<sup>6</sup> <http://netbeans.org/>

<sup>7</sup> <http://subversion.apache.org>

Embora os requisitos levantados nos projetos de pesquisa PECUS e ANIMALCHANGE indiquem a necessidade predominante de simulação contínua (a partir de equações diferenciais ordinárias ou de diferença), o FS foi projetado para também suportar simulações orientada a eventos e híbrida (ZEIGLER, 2000). Para suportar simulações orientadas a eventos, implementou-se uma lista de eventos futuros (LEF), a partir de uma estrutura de dados do tipo **heap**, na qual os eventos são ordenados pelo tempo de ocorrência.

O FS foi concebido de forma que simuladores construídos a partir dele pudessem ser compilados como bibliotecas dinâmicas a serem acessadas por uma AC. Essa abordagem permite desacoplar a simulação propriamente dita do armazenamento e manipulação de dados e da interface gráfica com o usuário. Dessa forma, o desenvolvedor da AC pode definir a linguagem e tecnologia para a implementação da interface gráfica e a forma de armazenamento de dados (e.g. memória, arquivos ou em banco de dados), possibilitando grande flexibilidade de integração entre diferentes sistemas. Essa abordagem difere daquela descrita por Bolte (1998), na qual o armazenamento de dados é feita pelos próprios objetos de simulação e, conseqüentemente, gera acoplamento entre esses objetos e os dados do modelo.

Para que a AC possa referir-se a um dado objeto da simulação, cada objeto (modelos componentes e variáveis) é associado a um caminho único na hierarquia, definido por uma string, na qual cada elemento do caminho corresponde a um nível da hierarquia do modelo. Desta forma, a AC pode solicitar o valor de qualquer variável do modelo, no passo de tempo corrente durante a execução da simulação, informando ao FS o caminho da variável na hierarquia do modelo por meio de um argumento do tipo string. Criou-se ainda, uma estrutura de dados do tipo `map`<sup>8</sup> para permitir o desacoplamento entre as classes que implementam o controle da simulação do FS, e aumentar a eficiência na localização dos componentes e variáveis .

A comunicação entre um simulador gerado a partir do FS e a AC durante a execução da simulação, é viabilizada por meio de *callbacks*<sup>9</sup>, isto é chamadas à funções da AC por meio de ponteiros de função. As *callbacks* viabilizam o controle da simulação e o acesso ao estado corrente das variáveis durante a simulação. Exemplos

<sup>8</sup> <http://www.cplusplus.com/reference/stl/map/>

<sup>9</sup> Referência a uma parte executável de código que é passada como argumento para outra parte do código. Isso permite a uma camada de software de nível inferior chame sub-rotinas de uma camada superior.

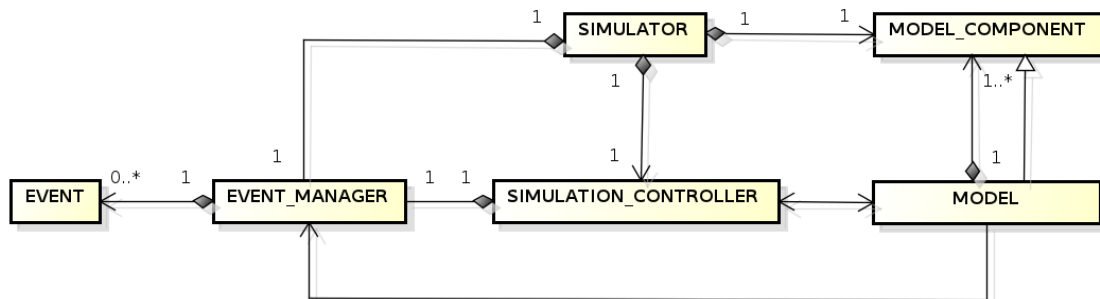
de callbacks incluem erros de configuração, disparo da execução, o final de uma iteração, a execução de um evento, o final da execução da simulação e erros ocorridos durante a simulação.

Para representar séries temporais (trajetórias) de dados de entrada para a simulação, dado que os componentes não armazenam dados, decidiu-se por cadastrar as mudanças de valor de entrada como eventos na LEF. A abordagem adotada é, ao conhecimento dos autores, uma inovação em FS.

Um requisito para o FS foi o de que as unidades de simulação pudessem ser implementadas hierarquicamente. Isso exigiu a implementação de relações de agregação entre as unidades de simulação.

## RESULTADOS E DISCUSSÃO

As principais classes que compõe a arquitetura do FS são apresentadas na forma de diagrama de classes da UML na Figura 1.



**FIGURA 1.** Diagrama de classes do Framework.

A interface entre o FS e a AC é implementada por meio da classe SIMULATOR, que é responsável por construir e destruir o modelo do cliente e as classes da infraestrutura de controle do FS (i.e. as classes SIMULATION\_CONTROLLER e EVENT\_MANAGER) e permitir à AC definir: a) o tempo de início, duração e passo de tempo da simulação; b) o valor inicial das variáveis de estado dos modelos componentes; c) a trajetória das variáveis de entrada; d) os eventos de ocorrência pré-agendada durante a simulação.(e) informar ao FS os ponteiros das funções de callback da AC.

Além disso a classe `SIMULATOR` permite à AC comandar o início, pausa e término antecipado da simulação e acessar o tempo corrente de simulação e os valores das variáveis de saída.

A classe `SIMULATION_CONTROLLER` contém o relógio do sistema e o gerente de eventos (`EVENT_MANAGER`). Essa classe é a responsável por: a) incrementar o tempo de simulação; b) solicitar ao modelo a execução da função de transição, a qual é responsável pela atualização do estado do sistema; c) solicitar ao `EVENT_MANAGER` que dispare os eventos com tempo menor ou igual ao tempo corrente; d) disparar callbacks prévias e posteriores à execução da função de transição e de término da simulação.

A classe `EVENT`, base de todos os tipos de evento, possui como atributos o tempo e ponteiros para o objeto-alvo e o método da classe desse objeto responsável pela execução do evento.

A classe `EVENT_MANAGER` é responsável por criar instâncias de `EVENT` a partir de dados fornecidos pela AC ou pela função de transição do modelo. Quando um `EVENT` é criado, e tanto o objeto-alvo quanto o método-alvo são localizados na estrutura map e seu ponteiro atribuído como atributo ao novo evento.

A classe `MODEL_COMPONENT` é uma classe abstrata que permite a interação dos modelos com as classes `SIMULATOR`, `SIMULATION_CONTROLLER`, `EVENT_MANAGER`. Além disso, possui portas de entrada e saída que permitem a transferência de dados entre modelos componentes.

A classe `MODEL` é também uma classe abstrata especializada a partir da classe `MODEL_COMPONENT`. Ela permite que modelos hierárquicos sejam construídos. Para isso, a classe `MODEL` foi construída como um container de `MODEL_COMPONENT` (i.e. ela possui uma lista de instâncias da classe `MODEL_COMPONENT`). Ela é responsável por construir e destruir o modelo com seus componentes (i.e. criar e destruir as instâncias de `MODEL_COMPONENT` e das conexões entre eles). Essa classe é também responsável por propagar a chamada à função de transição para seus componentes na ordem adequada.

Novos simuladores são construídos a partir dessa infraestrutura por meio da especialização das classes `MODEL` e `MODEL_COMPONENT`, nas quais são implementadas as funções de transição (normalmente um conjunto de equações executadas a cada iteração em resposta ao evento *iterate*). O nível mais agregado do modelo, normalmente uma instância da classe `MODEL` é agregada ao `SIMULATOR`.

Eventualmente, a classe SIMULATOR pode ser especializada para atender à funcionalidade demandada por uma AC específica.

A simulação é executada de acordo com o descrito no diagrama da Figura 2.

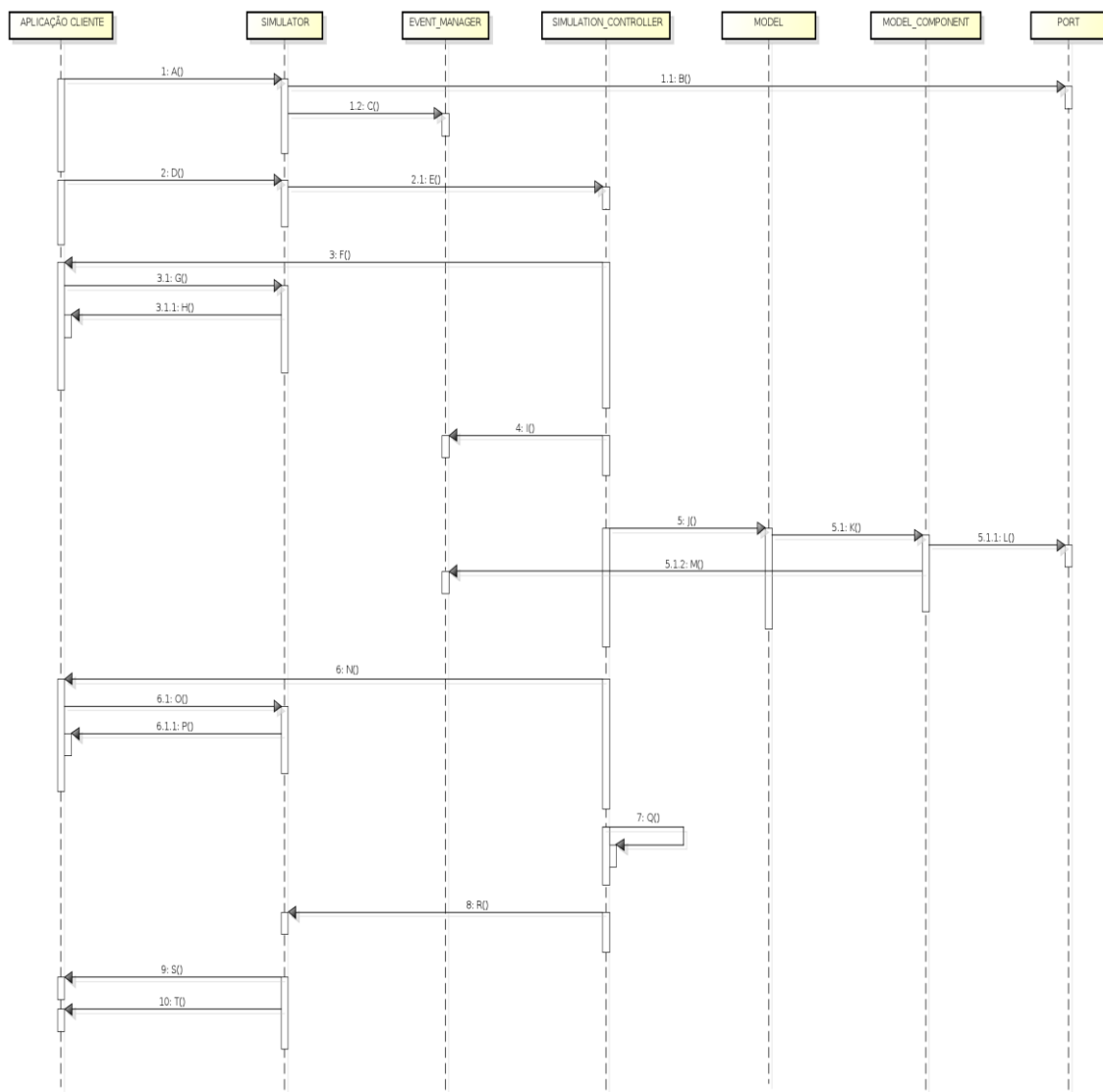


FIGURA 2. Diagrama de execução de uma simulação no FS





Há a seguinte correspondência entre as indicações no diagrama e as operações a serem realizadas:

- |                                |   |
|--------------------------------|---|
| A. Dados de Entrada            | L. Atribui Valor                              |
| B. Altera Valor                | M. Agendar Eventos Futuros                    |
| C. Cadastra Eventos            | N. Callback AfterIterate                      |
| D. Inicia Execução             | O. Consulta                                   |
| E. Inicia Execução             | P. Informa                                    |
| F. Callback BeforeIterate      | Q. Incrementa Tempo                           |
| G. Consulta Dados de Iteração  | R. Retorna Controle                           |
| H. Informa Dados de Iteração   | S. Callback AfterSimulate                     |
| I. Executar Eventos            | T. Retorna Controle para<br>Aplicação Cliente |
| J. Executa Função de Transição |   |
| K. Executa Função de Transição |   |

## CONCLUSÃO

O FS aqui descrito é compatível com a simulação de modelos para simulação contínua, discreta e híbrida em geral e atende os requisitos dos modelos a serem implementados no âmbito dos projetos PECUS e ANIMALCHANGE em particular.

A arquitetura do FS conseguiu atingir um alto grau de generalidade com uma estrutura bastante simples de classes. Esta simplicidade, aliada à padronização de codificação, permitirá uma fácil compreensão e manutenção do FS por novos membros da equipe. Ademais, o FS facilitará a implementação e modificação dos modelos componentes em um processo de desenvolvimento incremental de simuladores.

Como os modelos são construídos a partir da especialização das classes básicas do FS e tal especialização é feita na própria linguagem de programação C++, o modelador conta com todos os recursos da linguagem de programação disponíveis para a implementação dos modelos, bem como a chamada de outras bibliotecas desenvolvidas por terceiros.

Vislumbra-se como trabalhos futuros a integração do FS:

1. A criação de interface gráfica ou integração a outros sistemas, permitindo ao usuário a execução de simulações de forma interativa, alterando dados de entrada, acompanhando resultados parciais da simulação e visualizando relatórios e estatísticas da simulação
2. A criação de um repositório de modelos de processos relacionados a sistemas agropecuários, facilitando o reuso desses modelos.



3. Implementação e inclusão de: (a) algoritmos para integração numérica; (b) algoritmos ajuste de parâmetros, e; (c) bibliotecas matemáticas e estatísticas.

- **AGRADECIMENTOS**

A Embrapa, pela oportunidade de estágio.

- **REFERÊNCIAS**

BOLTE, J. Object-oriented programming for decision systems. In: PEART, R. M.; CURRY, R. B. (Ed.). **Agricultural systems modeling and simulation**. New York: Marcel Dekker, 1998. cap. 17, p. 629 – 650.

HILLYER, C.; BOLTE, J.; VAN EVERT, F.; LAMAKER, A. The ModCom modular simulation system. **European Journal Agronomy**, v. 18, p.333-343, 2003.

ZEIGLER, B. P.; PRAEHOFER, H.; KIM, T. G. **Theory of Modeling and Simulation**. 2nd Ed. San Diego: Academic Press, 2000. 510 p.