Padrões de Projeto Aplicados a Modelos de Simulação do Crescimento e Desenvolvimento de Culturas

Willingthon Pavan [1]
José Maurício Cunha Fernandes [2]
Carlos Amaral Hölbig [1]
Clyde William Fraisse [3]

Resumo: O desenvolvimento de modelos de simulação na agricultura, assim como em outras áreas, requer uma análise profunda do sistema em estudo, sendo crucial o emprego de métodos e ferramentas de engenharia de software. A modularidade e a forma genérica são os termos que descrevem a nova e amplamente aceita metodologia para superar as complexidades no desenvolvimento e reúso de modelos. A arquitetura MVC (Model-View-Controler), um padrão de projeto amplamente aceito no desenvolvimento de software, auxilia na definição das regras e responsabilidades dos objetos num sistema, possibilitando o desenvolvimento de aplicações com fluxo simplificado, robusto, flexível e de fácil manutenção. Um dos principais aspectos do MVC reside na separação entre os módulos da aplicação, de forma a possibilitar o reaproveitamento e o desenvolvimento genérico de modelos. Este artigo tem como objetivo apresentar uma estrutura para o desenvolvimento de simuladores de doenças de plantas, acoplados a modelos de crescimento, com a utilização extensiva de padrões de projeto orientados a objetos. Com os resultados, observou-se que a aplicação de técnicas e padrões no desenvolvimento de sistemas complexos, como os modelos de simulação, torna-os mais robustos, bem documentados, reusáveis, compatíveis e facilmente acopláveis a outras soluções já existentes.

Palavras-chave: Modelos de simulação. Simulador de doenças de plantas. Computação aplicada à agricultura.

Abstract: *The development of simulation models for agriculture, as well as for other areas, requires a deep analysis of the system in study, being crucial the choice of methods and tools of software engineering. Modularity and generic form are terms that describe the new and widely accepted methodology to surpass the complexities in the development and re-use of models. The MVC (Model-View-Controler) architecture, a project design pattern widely accepted in the software development industry, assists in the rules definition and properties of the objects in a system, allowing the development of simpler, robust applications, with flexible flow and easy maintenance. One of the main aspects of the MVC architecture resides in the separation of the application modules allowing the development of generic and re-usable models. The main objective of this study was to present a structure for the development of plant disease simulators coupled to crop growth models with extensive use of object oriented design pattern. It demonstrated that the use of design pattern techniques for development of simulators of complex systems results in more robust, well documented and reusable simulation models. In addition, the models were easily connected to other already existing applications.*

Keywords: *Simulation models. Plant disease simulator. Applied computing in agriculture.*

[1]Universidade de Passo Fundo, Curso de Ciência da Computação. Campus 1, Rodovia BR 285, Bairro São José. Passo Fundo (RS), Brasil
{pavan,holbig@upf.br}
[2]Embrapa Trigo. Rodovia BR 285, km 294. Passo Fundo (RS), Brasil
{mauricio@cnpt.embrapa.br}
[3]University of Florida, Agricultural & Biological Engineering. 271 Frazier Rogers Hall. P.O. Box 110570. Gainesville (FL) 32611-0570, USA
{cfraisse@ufl.edu}

doi: 10.5335/rbca.2009.003

1 Introdução

O desenvolvimento de modelos de simulação envolvendo sistemas de produção agrícola requer uma análise profunda do sistema, geração e acúmulo de conhecimento e uma ampla base de dados experimentais. Porém, é também crucial o emprego de métodos e ferramentas de tecnologia de software. A modularidade e a forma genérica são os termos que descrevem a nova e amplamente aceita metodologia para superar as complexidades que surgem ao construir, manter e reusar modelos em partes ou como um todo [26].

Na agricultura, como em outros modelos dinâmicos, o sistema pode ser analisado em termos de variáveis de estado, fluxo e auxiliares [9]. Essas variáveis, em geral, acabam fazendo parte de um conjunto de equações diferenciais que devem ser integradas numericamente, dada a complexidade do sistema. Esta estrutura comumente usada permite que seja possível desenhar e codificar um modelo com finalidades de entradas e saídas arranjadas com ferramentas de software.

Desde o surgimento dos primeiros computadores, o homem vem desenvolvendo técnicas para levar para dentro dos computadores o que encontra no mundo real, as linguagens de programação. Essas possibilitam que os computadores sigam ordens predefinidas e reportem os resultados obtidos. Nas últimas décadas, muitas linguagens foram criadas e em muitos paradigmas foram desenvolvidas, tentando fazer com que os computadores se pareçam cada vez mais com o cérebro humano. Dentre os paradigmas desenvolvidos estão o imperativo, o lógico, o funcional e o orientado a objetos, cada um tentando resolver os problemas de comunicação de forma diferente.

Modelos podem ser definidos como uma representação simplificada, por necessidade, do que se percebe ser realidade, podendo ser um objeto, uma ideia ou um sistema [30]. São uma descrição matemática das diversas causas e efeitos envolvidos num sistema real; para serem perceptíveis é necessário que as relações entre causas e efeitos sejam claras e limitadas [18]. São provenientes de aproximações realizadas para se poder entender melhor um determinado fenômeno, mesmo que essas aproximações não condigam totalmente com a realidade. De qualquer forma, um modelo retrata, ainda que de maneira simplificada, os aspectos da situação pesquisada [20].

Modelos de simulação do crescimento e do desenvolvimento de culturas têm sido projetados e desenvolvidos em várias partes do mundo [35, 16, 21, 37, 17, 22]. Dentre as culturas, pode-se citar o desenvolvimento para o trigo, soja, milho, cana-de-açúcar, mandioca, etc., esses para fins acadêmicos e práticos. Como um exemplo prático pode-se citar o uso do APSIM (Agricultural Production Systems SIMulator) e do DSSAT (Decision Support System for Agrotechnology Transfer).

O APSIM foi desenvolvido na Austrália para simular processos biológicos em sistemas agrícolas, relacionando os resultados econômicos e biológicos das práticas gerenciais em face do risco climático [17]. Este modelo está integrado a um sistema de suporte à tomada de decisão chamado “Whopper Cropper”, o qual tem como objetivo fornecer orientações no manejo de culturas, utilizando dados climatológicos observados e de previsão sazonal [6].

O DSSAT, desenvolvido por um grupo de pesquisadores das universidades da Georgia, Flórida, Hawaii, Guelph e Iowa State, além do centro internacional para fertilidade do solo e desenvolvimento agrícola (IFDC - International Center for Soil Fertility and Agricultural Development), simula o crescimento de culturas, produtividade, necessidades de água e nutrientes e o impacto ambiental na produção agrícola, estando atualmente na versão 4.0.

A maior parte dos programas de simulação do crescimento e desenvolvimento de culturas foi escrita em Fortran, razão por que muitos esforços têm sido feitos para que sejam reestruturados, visando aos aspectos fundamentais de modularidade. Uma vez que muitos programas foram desenvolvidos de forma independente, a incompatibilidade representa uma dificuldade para aqueles que se encontram fora do ambiente de desenvolvimento. Assim, nos dias de hoje há um consenso sobre a necessidade de se desenvolverem modelos mais compatíveis e eficientes, o que leva a que novas técnicas sejam utilizadas para minimizar esta heterogeneidade.

Na busca por técnicas de desenvolvimento que contemplem as necessidades de compatibilidade, interatividade e expansibilidade de modelos, novas tecnologias foram criadas a fim de possibilitar a aplicação da ideia de modularidade no desenvolvimento, como o paradigma de orientação a objetos [23]. A programação orientada a objetos tem revolucionado a maneira como os softwares são concebidos, escritos e mantidos. A orientação a objetos simplesmente significa a visualização de um problema em termos de objetos envolvidos com este, permitindo

que os softwares não apenas modelem o mundo real, mas permitam que o código seja escrito de uma forma mais organizada e consistente [28].

Características de reaproveitamento são usadas com grande frequencia na programação orientada a objetos, pela facilidade e rapidez no desenvolvimento, podendo ser aproveitadas para desenvolver sistemas de simulação complexos. A orientação a objetos possibilita inserir novos elementos no ambiente sem que haja a necessidade de recodificação, permitindo que os componentes envolvidos possam trocar informações necessárias para o funcionamento do modelo; assim, possibilita-se o desenvolvimento de softwares modulares capazes de se acoplar na busca de um objetivo específico. Busca-se, dessa forma, utilizar essas tecnologias a fim de obter um módulo que agregue diversos outros submódulos capazes de gerar os resultados esperados.

Na busca por padrões que viessem a auxiliar os desenvolvedores surgiram os design patterns, os quais são geralmente tidos como soluções já testadas e bem-sucedidas para problemas encontrados. O princípio básico é que, à medida que os problemas vão ocorrendo, é preciso que sejam documentados para que no momento em que voltem a ocorrer seja possível saber qual providência tomar. Assim, pode-se entender como design patterns a descrição de como resolver da melhor forma possível um determinado problema que se repete [5].

A utilização de programação multicamadas permite que as aplicações se tornem independentes da arquitetura do sistema, oferecendo uma facilidade maior quando da sua manutenção. Com esta técnica, é possível realizar a separação das camadas de apresentação e das regras de negócio, além de fazer a separação destas da camada de armazenamento, utilizando bancos de dados, XML ou outros; assim, facilita-se o desenvolvimento de softwares capazes de serem manutenidos, evitando a propagação para outros módulos. Quando o objetivo é o desenvolvimento de softwares robustos, o uso de multicamadas torna-se indispensável, em virtude do grande impacto que as mudanças causam num software.

Desde o final da década de 1960 até os dias atuais, pesquisadores têm desenvolvido modelos computacionais para a simulação do crescimento e do desenvolvimento de culturas, incluindo o rendimento [4]. Observa-se que na maior parte desses modelos de simulação, como o CropSim-Wheat, são raros ou inexitentes os que apresentam alguma forma de contabilização do impacto das pragas numa planta.

Segundo o conceito estabelecido pela FAO (Food and Agriculture Organization of the United Nations), entende-se como praga qualquer espécie, raça ou biótipo de vegetais, animais ou agentes patogênicos nocivos aos vegetais ou produtos vegetais. Dessa forma, o termo compreende animais (insetos, ácaros e nematóides), doenças (causadas por fungos, bactérias, fitoplasma, vírus e viróides) e plantas daninhas (plantas invasoras ou plantas espontâneas) [19].

Em virtude da inexistência da contabilização do impacto das pragas por grande parte dos modelos de simulação de culturas, este projeto foi originado buscando criar uma estrutura capaz de integrar os diferentes sistemas de simulação, de forma facilitada e com o uso de modernas técnicas de desenvolvimento, além da possibilidade de uso de uma base de dados centralizada.

Dessa forma, pela união das técnicas de programação orientada a objetos e uso de padrões de desenvolvimento, pode-se alcançar um alto nível na criação de sistemas de modelagem e simulação, tanto nas áreas do crescimento e desenvolvimento de plantas como no de submodelos, como os das pragas. O objetivo deste trabalho é propor uma estrutura para o desenvolvimento de simuladores acoplados a modelos de crescimento de plantas, utilizando programação orientada a objetos numa plataforma MVC.

2 Desenvolvimento de Modelos de Simulação: técnicas e tecnologias

Para o desenvolvimento deste trabalho utilizaram-se diferentes tipos de técnicas e tecnologias, desde aquelas desenvolvidas e utilizadas nos primórdios da computação até as existentes nos dias atuais, como a modelagem orientada a objetos, o design pattern MVC, as linguagens de programação orientadas a objetos, a separação das tecnologias na forma de multicamadas e muitas outras. Do conhecimento legado utilizaram-se daqueles construídos desde o final da década de 1960 até os dias de hoje, que são os modelos de simulação do crescimento e desenvolvimento de plantas escritos em linguagem Fortran.

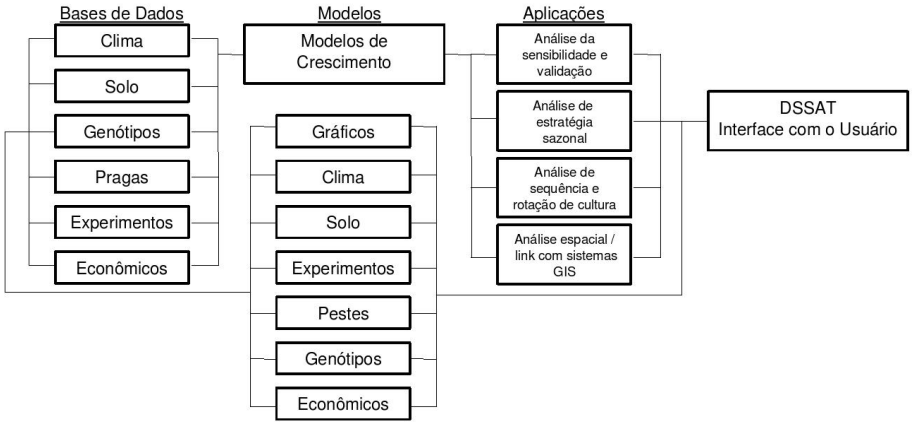
2.1 DSSAT

O DSSAT é um sistema composto por diversos modelos de simulação, esses orientados a processos, projetados para aplicações globais e que trabalham independentemente de local, estação e cultivar [37]. É apropriado para que os estudos a longo prazo avaliem as estratégias eficientes da gerência da cultura e otimizem a produção [10]. O DSSAT combina dados de solo e de clima com modelos de culturas e aplicativos a fim de simular os resultados de vários anos de estratégias no manejo de culturas [14]. É o resultado do trabalho colaborativo de muitos cientistas de diversas universidades e centros de pesquisa que fazem parte do consórcio ICASA (International Consortium for Agricultural Systems Applications – http://icasa.net).

Como um pacote de software, que integra os efeitos do solo, características da espécie, coeficientes genéticos, dados climáticos e opções de manejo, o DSSAT permite que os usuários possam solicitar resposta a perguntas como "O que ... se ...?" e simular os resultados de experimentos em minutos em um computador, os quais poderiam consumir uma parte significativa da vida profissional de um agrônomo [14]. A comparação das saídas dos modelos com os resultados observados faz parte da validação de um modelo.

O DSSAT é uma coleção de programas independentes que interagem entre si, no qual os modelos de simulação de culturas ficam ao centro (Figura 1), enquanto as bases de dados descrevem o clima, solo, observações e condições experimentais, além de informações sobre genótipos para aplicar nos modelos em diferentes situações [15]. Os componentes de software são escritos em:

- Fortran (modelos);
- C (shell);
- Pascal (gráficos);
- Dbase (base de dados) e
- Basic (programas de gerenciamento de estratégia e risco).



Fonte: Adaptado de [15]

Figura 1. Estrutura do Modelo DSSAT

2.1.1 CropSim-Wheat

Dentre os vários modelos de simulação que fazem parte da suíte do DSSAT escolheu-se o modelo Cropsim-Wheat, o qual é desenvolvido e mantido por Leslie A. Hunt [13], como modelo de simulação do crescimento e desenvolvimento do trigo, para ser inserido na estrutura aqui apresentada.

Dentre os 27 modelos de crescimento de culturas incorporados no DSSAT [14], o CropSim-Wheat apresenta-se como responsável pela simulação do crescimento e desenvolvimento do trigo (Figura 1, Modelos de Crescimento). Consiste do módulo planta de trigo que se conecta com os módulos de clima e de solo, pertencentes à suite do DSSAT, os quais computam a energia e a água disponível para o crescimento da planta de trigo, ao passo que o módulo planta de trigo simula os eventos fenológicos, expansão foliar, acúmulo de carboidratos e a partição entre a parte aérea e as raízes.

A alimentação do modelo consiste em características da planta do trigo (cultivares), dados de clima (radiação solar, temperatura mínima, temperatura máxima e precipitação), assim como práticas agronômicas, data de semeadura e espaçamento [12].

2.2 Linguagem R

Para possibilitar a integração entre os diversos módulos dos sistemas desenvolvidos em diferentes linguagens foi utilizada a tecnologia R (http://www.r-project.org), a qual é uma linguagem e ambiente para computação estatística e geração de gráficos, similar à linguagem e ambiente S, desenvolvida no Bell Laboratories por John Chambers e colegas [27, 25].

O R disponibiliza uma grande variedade de métodos estatísticos e técnicas gráficas. Um dos seus pontos fortes é a facilidade com que gráficos de alta qualidade podem ser produzidos. É uma linguagem bem desenvolvida, possuindo características herdadas de outras, como estruturas de controle, funções definidas pelo usuário (programação estruturada e modular), recursividade, facilidade de manipulação de arquivos e programas externos, orientação a objetos, entre outras [27, 31].

O R é disponibilizado sob os termos da GNU (General Public License) da Free Software Foundation (http://www.fsf.org) na forma de código aberto; roda num grande número de plataformas Unix e similares (incluindo FreeBSD e Linux), podendo também ser compilado e rodar em Windows 9x/NT/2000 e MacOS [27, 31].

R é um ambiente integrado de facilidades de software para a manipulação de dados, cálculos e geração de gráficos; inclui um efetivo tratamento de dados e facilidades de armazenamento, tanto em arquivos como em bancos de dados; pode ser expandido com a adição de novas funcionalidades pela definição de novas funções. Como em muitas outras linguagens, a maior parte das bibliotecas existentes na linguagem é desenvolvida na própria linguagem e disponibilizada na forma de pacotes, tornando-o dinâmico e muito poderoso [25].

Para tarefas que exigem um alto poder de processamento, linguagens como C, C++ e Fortran podem ser utilizadas para desenvolvimento de blocos de código e integrados ao R, podendo ser chamados em tempo de execução, assim como códigos podem ser escritos nestas linguagens para manipular diretamente objetos em R [25].

Com as potencialidades e características da linguagem, aliado à enorme quantidade de pacotes disponíveis (http://cran.r-project.org/src/contrib/PACKAGES.html) para os mais variados fins, torna-se possível a implementação de qualquer tipo de solução computacional, o que conduziu a sua escolha como linguagem para a realização da ponte entre os modelos de simulação escritos em Fortran e os em Java.

2.3 Java

Para o desenvolvimento dos componentes, sistemas e modelos de simulação foi utilizada a linguagem de programação Java (http://java.sun.com), a qual é uma linguagem de distribuição gratuita, capaz de produzir softwares robustos e multiplataforma que podem rodar em diversos tipos de microcomputadores e dispositivos de pequeno porte.

Java é uma linguagem orientada a objetos, de alto nível, influenciada de várias formas por C, C++ e Smalltalk, assim como por boas ideias implementadas em outras linguagens. Foi desenvolvida por uma equipe de desenvolvedores da Sun Microsystems no início da década de 1990, sendo composta por uma gama de produtos baseados no poder da rede e na ideia de que um software deve ser capaz de rodar em diferentes máquinas, sistemas e dispositivos, como handhelds, PDAs (Palm), celulares, entre outros [34, 32].

Java é dividida em três plataformas: Standard Edition (SE), Enterprise Edition (EE) e Micro Edition (ME).

A plataforma SE é voltada para o desenvolvimento de aplicações Desktop, ao passo que a EE é voltada para desenvolvimento Web; já a ME foi criada para o desenvolvimento de aplicações para equipamentos de pequeno porte [1].

Uma plataforma pode ser definida como um ambiente de hardware ou software em que programas são executados, ou seja, uma combinação de sistema operacional e hardware. As plataformas em Java são compostas por dois componentes: a máquina virtual Java (JVM) e a API Java (Java Application Programming Interface). A JVM é a base para as plataformas Java, sendo portada para vários tipos de hardwares, ao passo que a API Java é uma grande coleção de componentes de softwares que fornecem muitas potencialidades úteis, agrupadas em bibliotecas de classes e interfaces, conhecidas como pacotes (packages) [34].

Java destaca-se entre as demais linguagens pelo fato de ser dinâmica, isto é, por ser uma linguagem interpretada e poder ser carregada a qualquer momento, mesmo quando o interpretador já estiver rodando, permitindo que estas classes sejam carregadas dinamicamente e instanciadas conforme o fluxo de execução. A uniformidade dos conceitos de Java permite a integração de conceitos modernos, como componentes, invocação remota, reflexão/introspecção, validação, conectividade a bancos de dados e muitos outros.

Com relação à conectividade a banco de dados, Java possui integração com um grande número de SGBDs relacionais e orientados a objetos. É detentora de uma poderosa API (Application Program Interface), a JDBC-ODBC (Java DataBase Conectivity and Open Database Conectivity), escrita inteiramente em Java, a qual permite o envio de comandos SQL para um driver de banco de dados, responsável por acessar o banco e devolver os MSP (Modelo de Simulação de Pragas). Permite, dessa forma, que a aplicação não fique presa a um banco de dados específico, o que pode ser substituído a qualquer instante.

2.4 Rserve

Para a realização da integração entre as tecnologias (Java, R, C e Fortran) utilizou-se o Rserve, um servidor TCP/IP desenvolvido por Simon Urbanek sob licença GPL, que permite que outros programas se utilizem das facilidades da linguagem R ou façam ligações com a biblioteca do R [36, 29].

Todas as conecções realizadas através do Rserve possuem uma workspace (área de trabalho) e um diretório de trabalho separado. As implementações funcionam como aplicações cliente-servidor, permitindo que linguagens como C/C++ e Java possam se utilizar dos recursos da linguagem R [36]. O Rserve suporta conecções remotas, autenticação e transferência de arquivos. O seu uso típico é para integrar o R (retaguarda) com sistemas para computação de modelos estatísticos, gráficos, entre outros [36, 29].

Para o desenvolvimento com a linguagem Java, utilizou-se a biblioteca JRclient, um completo conjunto de ferramentas que permite que qualquer aplicação Java acesse o Rserve, sendo inteiramente desenvolvido em Java. Fornece uma transformação automática de tipos entre o Java e o R, como a transformação dos `int`, `double`, `arrays`, `string`, vetores e classes para os objetos específicos do R, como `RBool`, `RList`, etc. [36, 29].

2.5 Banco de dados

Os dados necessários para "alimentar" o sistema foram dispostos, em forma de tabelas, numa base de dados relacional implementada no servidor de banco de dados PostgreSql 8.1 (http://www.postgresql.org), sob o servidor Linux Ubuntu Dapper (http://www.ubuntu.com).

A base de dados foi modelada a fim de armazenar as informações climáticas (dados observados e dados de prognósticos), os dados de entrada para os modelos (tipos de solo, espécies, cultivares, tratamentos, etc.) e os dados resultantes da execução dos modelos simulação, possibilitando a análise e a validação dos modelos.

2.6 Infraestrutura

Tendo em vista a necessidade de seguir regras bem estabelecidas e de forma clara, utilizaram-se padrões, os quais são maneiras de descrever as melhores práticas, os bons projetos e a captura de experiências, de forma que seja possível, para outros, reutilizá-las [11]. O estabelecimento de padrões auxilia o desenvolvedor a criar sistemas, aproveitando-se de técnicas já conhecidas e bem estudadas com o intuito de modelar o comportamento

das aplicações.

Atualmente, podem-se encontrar diversos tipos de "padrões de projeto", cada um com uma finalidade específica, nomeando, abstraindo e identificando os aspectos-chave de uma estrutura. A descrição de soluções já testadas para determinados problemas de projeto forma uma base sólida de soluções, as quais atuam como blocos de software pré-fabricados para a construção de softwares complexos. A qualidade do software está intimamente relacionada com a utilização, por parte do profissional, desses padrões.

Dentre os inúmeros padrões existentes, como o Business Delegate, Composite Entity, Composite View, Data Access Object, etc., destaca-se o MVC (Model-View-Controller – Modelo-Visão-Controle), sendo o desing pattern mais conhecido e utilizado. É responsável por separar acesso a dados, lógica de negócio e apresentação de dados e interação de usuário.

2.7 MVC

Para obter uma estrutura clara e que possibilitasse a reutilização e acoplamento de novos módulos e modelos utilizou-se o padrão MVC, o qual é um modelo de desenvolvimento de aplicações que utiliza as características da programação em camadas, sendo divididas em três as camadas ou áreas funcionais: Modelo (Model), Visão (View) e Controle (Controller).

2.7.1 Modelo

O modelo é a parte que representa o estado e o comportamento do componente, gerenciando e conduzindo todas as transformações, não tendo conhecimento específico sobre o controlador nem sobre a visão; é o próprio sistema que mantém links entre o modelo e a visão, notificando a visão quando há mudanças no seu estado. É um repositório de dados e de lógicas de negócio, ou seja, uma representação do banco de dados. Geralmente, em aplicações baseadas em banco de dados parte da função do modelo é retornar dados de consultas ao banco e persisti-los.

A implementação da camada do modelo deu-se pelo uso de JavaBeans (beans) e Enterprise JavaBeans (EJB), os quais oferecem escalabilidade, concorrência, balanceamento de cargas, gerência de recursos automática, além de outros benefícios. Os beans fornecem o acesso rápido aos dados, ao passo que os EJBs fornecem acesso à lógica e aos dados compartilhados do negócio [33].

2.7.2 Visão

A visão é a responsável por disponibilizar os dados produzidos pelo modelo, gerenciando o que pode ser visto do estado representado pelo modelo. Os componentes da visão, também conhecidos como componentes de apresentação, em aplicações Web, geralmente são páginas que geram conteúdo dinâmico e seus MSP podem incluir conteúdo HTML, arquivos PDF, arquivos XML, imagens, gráficos, etc.

Para a implementação da visão utilizaram-se as tecnologias JSP (JavaServer Pages) e Servlets (classes Java executadas no servidor), acessando a camada de modelo e gerando, de forma dinâmica, o conteúdo estático a ser devolvido ao cliente, em formatos texto, XML, WML, gráfico e HTML. Como lógica de interação, quando o usuário final executar alguma ação dentro das páginas HTML criadas pela visão, um evento é submetido ao controlador, que terá a função de verificar o que fazer com essa ação.

2.7.3 Controle

O controle é responsável por receber as informações passadas pelo usuário através das visões e, baseado no conteúdo desta requisição, combinado com a programação ou o meta dado, decidir o que deve ser feito [5]. Em outras palavras, gerencia a interação do usuário/sistema com o modelo, fornecendo o mecanismo pelo qual as mudanças são feitas no estado do modelo.

Para a implementação do controle utilizou-se o Apache Struts 2.0 (http://struts.apache.org), o qual é um framework livre, de código aberto, destinado à criação de aplicações Web na linguagem de programação Java. É composto de diversas tecnologias, possibilitando aos desenvolvedores Web criar aplicações baseadas em padrões

de fácil construção, com facilidades de extensão e de fácil manutenção. Além do MVC, o Struts adere a uma série de outros padrões e boas práticas, possibilitando o desenvolvimento de forma transparente [2].

3 Resultados

O modelo MSP (Modelo de Simulação de Pragas) foi desenvolvido utilizando as técnicas de MVC, multicamadas e orientação a objetos, além de diversas outras tecnologias mencionadas, dividindo-se em módulos de software responsáveis pela manipulação de dados, interface com o usuário e controle da "lógica de negócio", que está por trás da interface com o usuário. A Figura 2 mostra o relacionamento entre as camadas do modelo.

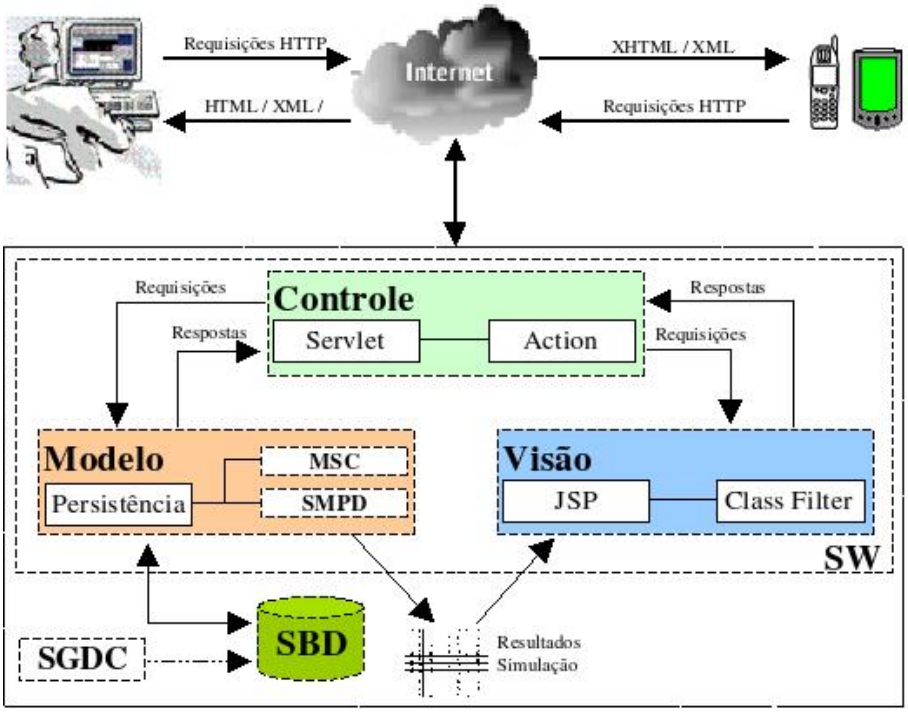


Figura 2. Modelo de simulação de doenças de plantas (MSP)

Para a execução do modelo MSP foram projetados servidores com responsabilidades distintas de processar as requisições feitas pelos demais. Estes servidores foram divididos em: (i) servidor de gerenciamento de dados climáticos (SGDC), (ii) servidor de banco de dados (SBD), (iii) servidor de modelos de previsão de doença (SMPD), (iv) servidor Web (SW) e (v) servidor de modelo de simulação de culturas (MSC).

A maneira como são organizados os componentes dentro da estrutura MVC oferece uma grande confiabilidade, além de facilitar o desenvolvimento e capacitar a execução em cada um separadamente, possibilitando a substituição de um determinado componente a qualquer instante sem que haja a necessidade de alteração no sistema como um todo. Por exemplo, se houver a necessidade de substituição de um determinado modelo de simulação, isso pode ser feito rapidamente e sem muito esforço.

O desenvolvimento da estrutura foi baseado no framework Struts, dividindo-se a programação em três camadas: camada de controle, camada de visão e camada de modelo. Cada componente deste sistema foi programado através de classes Java, encapsulando as operações e atributos de cada um.

O funcionamento do sistema dá-se de forma paralela e dinâmica, com cada componente sendo controlado e gerenciado pelas regras definidas em cada servidor. Alguns componentes são invocados por agendamentos feitos pelos programadores, ao passo que outros obedecem aos eventos gerados por outros componentes ou pelos usuários do sistema, em tempo de execução. Com a estrutura adotada, observa-se uma certa autonomia, característica esta buscada pela tecnologia de objetos.

O fluxo da execução do sistema é mediado pelo controlador central do Struts, o qual é responsável por delegar as requisições para os tratadores apropriados, localizados no modelo, os quais representam a lógica e o estado do sistema. A requisição é respondida através do controlador e apresentada na visão de maneira adequada, sendo as respostas orientadas por meio de mapeamentos, os quais são carregados de arquivos de configuração XML, levando a que haja dependência entre a visão e o modelo, auxiliando na criação, expansão e manutenção do sistema.

Com a visão de que um sistema deva ser decomposto em partes coesas, mas fracamente acopladas (modularidade), e que permita a substituição de diferentes versões dos módulos a qualquer tempo, todas as classes desenvolvidas seguiram um padrão de modularidade, o qual foi definido e descrito por [24]. Cada um dos módulos foi subdividido a fim de que refletisse melhor as necessidades do modelo, separando, dessa forma, o código em inicialização, cálculo de taxas, integração e saída (Figura 3).

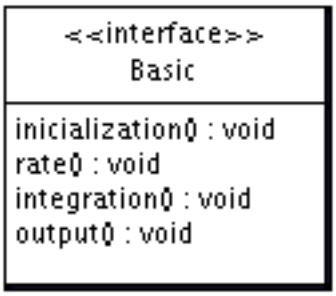


Figura 3. Interface implementada pelas classes do sistema

Tendo como princípio básico a utilização de linguagem orientada a objetos neste projeto, a persistência dos dados foi realizada utilizando-se em todos os módulos o framework Hibernate, o qual é responsável por armazenar e recuperar objetos do banco de dados, possibilitando uma maior transparência na camada de persistência e deixando o sistema mais flexível quanto a uma substituição do banco de dados [3].

3.1 SGDC

O servidor SGDC é responsável pela obtenção e armazenamento de dados climáticos, disponibilizados de diversas formas e locais, como pelas estações meteorológicas automáticas dispostas no território nacional, tanto de propriedade privada como de propriedade do governo brasileiro.

Os dados coletados são encontrados em diferentes formatos, desde XML (Extensible Markup Language), texto, HTML (HyperText Markup Language), arquivos binários, entre outros. Esses dados são processados a fim de se analisar sua veracidade e integridade; após esse processamento, são catalogados, indexados e armazenados no banco de dados com o objetivo de serem facilmente recuperados.

Este servidor possui uma estrutura composta por uma variedade de pequenos aplicativos, cada um com uma tarefa específica a realizar, como os "robôs de busca de dados" disponíveis na web, os coletores de dados de estações meteorológicas automáticas, os verificadores de integridade e veracidade dos dados, os sumarizadores de variáveis climáticas e os transportadores de dados para o SBD (Figura 4). Todos os aplicativos são gerenciados por um agendador de tarefas, o qual se utiliza do cron[4] do sistema operacional Linux para controlar a ordem e o horário de cada tarefa.

Um aplicativo especial destaca-se neste servidor, o coletor de dados de prognóstico climático, o qual é composto por módulos responsáveis por efetuar o download de arquivos binários, resultantes da execução dos modelos de circulação global (modelo Eta[5]), e processar o seu conteúdo de acordo com coodenadas geográficas

[4]Recurso padrão dos sistemas Unix (ou Linux) que permite o agendamento de tarefas para serem executadas num momento específico ou em intervalos regulares.

[5]ETA - Modelo de mesoescala, em ponto de grade, de equações primitivas. Nome oriundo da letra grega η (Eta). No Brasil, roda operacionalmente no Centro de Previsão de Tempo e Estudos Climáticos (CPTEC), sendo hidrostático e cobre a maior parte da América do Sul e oceanos adjacentes. A sua resolução horizontal atual é de 40 km e a vertical, de 38 camadas [7].

previamente estabelecidas (estações meteorológicas e municípios), obtendo os dados horários necessários para os modelos de simulação. As variáveis prognósticas do modelo são: precipitação pluviométrica, temperatura do ar, umidade relativa, pressão atmosférica e velocidade e direção do vento.

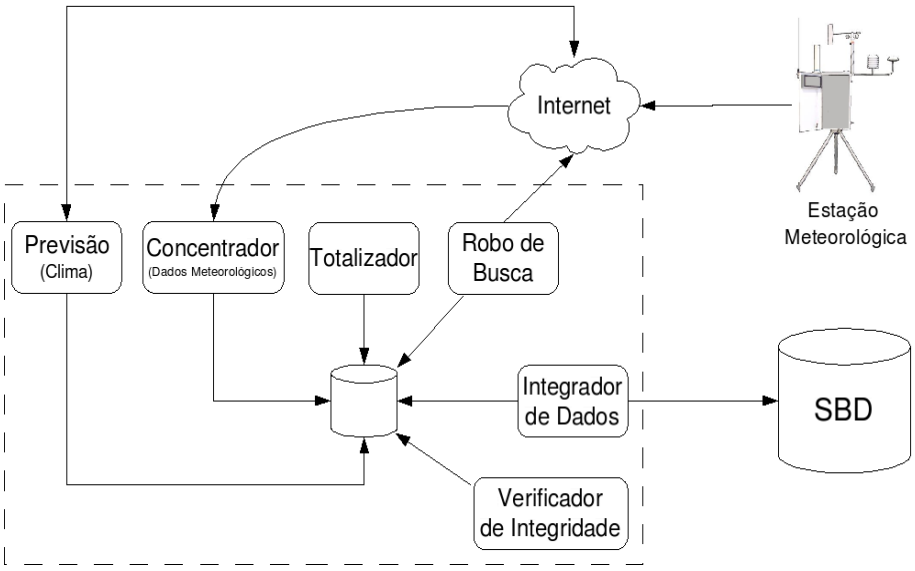


Figura 4. Servidor de Gerenciamento de Dados Climáticos (SGDC)

O SGDC, por meio de rotinas específicas, oferece aos modelos de simulação o suporte necessário para a realização da importação temporária de dados climáticos, disponibilizados na forma de arquivos texto, no momento da execução dos modelos, sendo eliminados ao término da seção do usuário ou armazenados permanentemente, se assim for a decisão do usuário. Esses dados, no momento da importação, são repassados diretamente para o SBD.

3.2 SBD

O servidor SBD é responsável pela disponibilização dos dados armazenados aos demais servidores e modelos através de requisições SQL (Structured Query Language), sendo projetado e organizado para armazenar informações advindas das mais variadas fontes, formas e usuários. Os dados podem ser públicos, disponíveis a todos os usuários, ou de uso privado, apenas para o "proprietário" dos dados.

O SBD foi preparado para ser capaz de gerenciar todos os dados de entrada necessários para a execução dos modelos de simulação, além de armazenar as saídas dos modelos. Gatilhos foram programados para realizar a limpeza do banco de dados, eliminando-se os dados não mais necessários e mantendo-se apenas aqueles selecionados pelo próprio usuário.

Para garantir o direito de propriedade dos dados de cada usuário o SBD foi organizado de tal maneira que para toda e qualquer informação inserida no banco de dados seja necessário informar a quem pertence, possibilitando, dessa forma, que as buscas por dados para as futuras execuções dos modelos sejam facilitadas.

Todas as conexões ao banco foram limitadas àquelas que estejam sendo realizadas na própria máquina ou dentro da estrutura definida (rede interna), evitando, assim, a tentativa de acesso indevido aos dados. Políticas de segurança, como criptografia de senhas e relacionamentos entre as entidades modeladas, foram implementadas com o intuito de minimizar as possibilidades de exceções.

3.3 SMPD

O SMPD é um servidor que tem como responsabilidade fornecer informações sobre o resultado da simulação sobre determinadas pragas de uma cultura; pode ser substituído por qualquer modelo de simulação que siga o

padrão estabelecido, possuindo os métodos necessários para o acoplamento com os demais módulos.

Pode-se citar como exemplo o modelo criado e desenvolvido por Emerson Del Ponte [8], o qual simula epidemias de uma doença que se destaca no cultivo do trigo, conhecida por giberela, causada pelo fungo *Gibberella zeae*, a qual está relacionada com eventos climáticos como temperatura, precipitação e umidade relativa, principalmente no estádio fenológico de floração.

Este modelo segue o padrão definido e explicado anteriormente, possuindo a interface básica para que pudese ser acoplado ao sistema. O modelo, a cada passo da aplicação, no período que compreende a fase da floração, executa rotinas internas calculando e fornecendo informações sobre a severidade da doença. Esta informação é de fundamental importância para alimentar o simulador da cultura (MSC), afetando processos internos e ocasionando modificações nos resultados simulados.

Estratégias de controle podem ser definidas no sistema, de modo a servir de entrada para os modelos, interferindo nos processos de simulação. Um exemplo dessas estratégias de controle é o uso de fungicidas sob certas condições de pressão da epidemia [8].

3.4 SW

O servidor SW é o responsável por disponibilizar acesso via Web, através do servidor de aplicações Apache Tomcat (http://www.apache.org), de forma restrita ou irrestrita, de acordo com as regras implementadas para cada módulo do sistema. É um servidor de aplicações desenvolvido pela Fundação Apache utilizando tecnologia Java, tendo a habilidade de converter, automaticamente, qualquer programa JSP num Servlet equivalente, isto é, capaz de criar código fonte Java a partir de um documento HTML.

O desenvolvimento foi feito sob a arquitetura MVC e implementado sob o framework Struts, possibilitando o controle e a separação das camadas da aplicação. Cada camada é tratada de forma independente, podendo ser reaproveitada caso já tenha sido desenvolvida.

Este servidor, quando se trata de interface web, é o responsável por prover todo acesso às aplicações, possibilitando a sua adequada execução. Quando uma requisição do usuário é realizada, o Tomcat repassa-a para o controle do Struts, deixando a cargo deste o fluxo da execução da aplicação, permanecendo à espera de alguma nova requisição externa ou por parte do Struts.

Com relação à visão, foi implementada utilizando-se tecnologias como JSP, JSF, TagLibs, JavaScript, AJAX, entre outras, possibilitando uma boa organização de código, boa qualidade na apresentação e na execução das mesmas, oferecendo aos usuários um sistema dinâmico e reciclável, atualizando apenas as informações necessárias. O uso destas tecnologias levou a que o sistema ficasse amigável e leve (tempo de download e upload curto).

3.5 MSC

O servidor MSC é responsável por interagir com o servidor SMPD, recebendo informações sobre a dinâmica da praga e fornecendo os dados sobre os estádios da cultura, além de outras informações, como a área foliar existente num determinado momento. Essa interação é de fundamental importância, tendo em vista que os processos da dinâmica de populações e de desenvolvimento da planta dependem de informações como estas.

A implementação do MSC deu-se com a realização de alterações no modelo de simulação Cropsim-Wheat, tendo como objetivo a disponibilização de seus serviços na Web, executando-o em ambiente Linux, sabendo-se que a sua versão original é para execução em sistema Windows® e com processamento local. No modelo Cropsim-Wheat foram realizadas as alterações necessárias para que pudesse ser executado e manipulado por meio da linguagem R, a qual é a ponte para os demais servidores deste sistema (Figura 5).

Para possibilitar a integração entre a linguagem Java e a linguagem R, utilizou-se o Rserve, um servidor TCP/IP para a linguagem R, pelo qual se tornou possível que os módulos desenvolvidos em Java pudessem se utilizar das facilidades da linguagem R sem precisar iniciar uma seção do R ou ligá-los a alguma biblioteca específica (Figura 6). Todas as conexões realizadas são separadas em espaços de trabalho (diretórios) diferentes no servidor. Com o uso do Rserve, possibilitou-se fazer conexões remotas ao R com o intuito de realizar a computação de

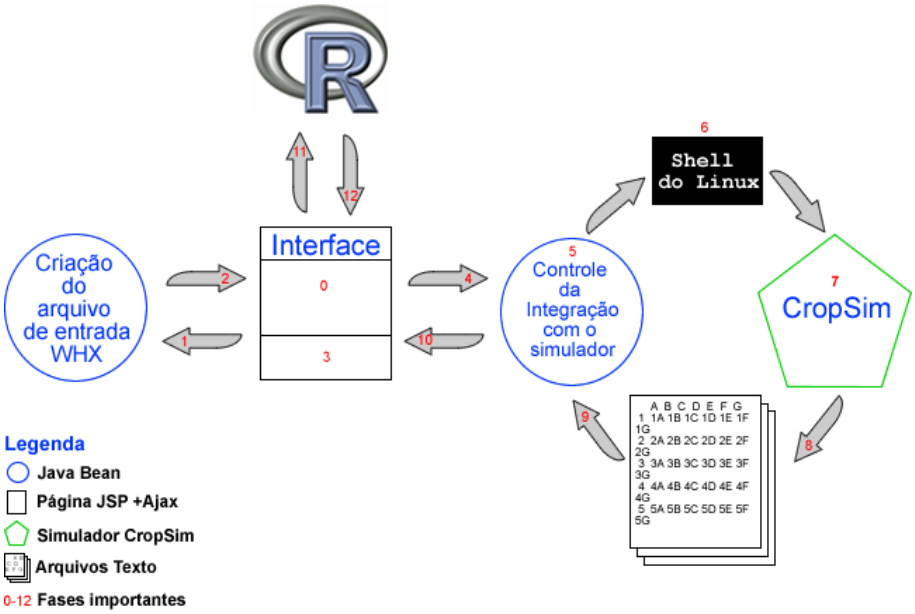


Figura 5. Integração R x CropSim-Wheat

modelos estatísticos, geração de gráficos e, principalmente, executar os modelos de simulação desenvolvidos em Fortran.

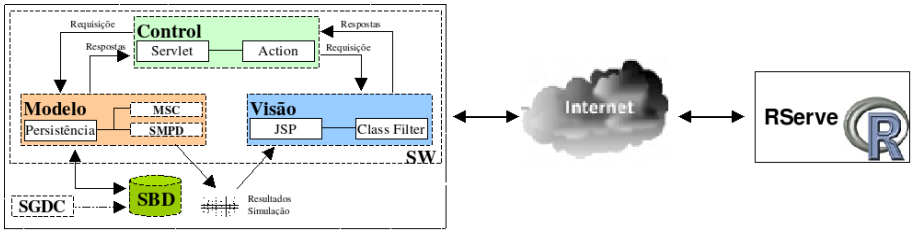


Figura 6. Integração Modelo x RServe

Para a realização das ligações entre os módulos dos servidores com o RServe utilizou-se uma biblioteca de classes denominada Rcliente, a qual realiza a integração da linguagem Java com o servidor TCP/IP RServe (Figura 7), responsável pela execução dos comandos da linguagem R, permitindo a ligação entre a heterogênea lista de tecnologias envolvidas. Obtiveram-se como resultado os dados necessários para certos servidores, como área foliar, estádios fenológicos etc.

Além de conteúdos resultantes da integração entre os servidores e os modelos, a geração de gráficos bem acabados tornou-se possível em tempo de execução, para publicação direta. Com a interação entre tecnologias como AJAX e as de ligação dinâmica, já mencionadas, tornou-se possível a construção de interfaces amigáveis e de fácil manipulação.

4 Discussão e Trabalhos em Andamento

Com a estrutura aqui apresentada, foi possível organizar e reutilizar conhecimentos construídos durante as últimas décadas sem a necessidade de alterações bruscas nos mesmos, melhorando a performance e a forma de disponibilização, não mais centrada em aplicações fechadas, mas abrangendo ambientes abertos e de acesso em massa.

A reutilização de modelos de simulação amplamente testados e validados torna as aplicações mais robustas

```
// Conexão com o RServe
Rconnection c = new Rconnection();
// Execução de operações entre vetores
ArrayList sumArray = new ArrayList();
c.eval("x=0:10");
c.eval("y=40:50");
sumArray.add(eval("x+y"));
int max = c.eval("max(x)");
```

Figura 7. Executando código R na linguagem Java.

no que diz respeito à simulação do crescimento e desenvolvimento de culturas, tendo em vista que o foco dos estudos fica direcionado ao problema em si, não à estrutura computacional.

Verificou-se a importância da aplicação de técnicas e padrões no desenvolvimento de sistemas complexos como os modelos de simulação, tornando-os mais robustos, bem documentados, reusáveis, compatíveis e facilmente acopláveis a outras soluções já existentes. A fácil manutenção e/ou substituição de módulos problemáticos assegura a qualidade e confiabilidade dos resultados. As tecnologias utilizadas permitem uma fácil conexão com bancos de dados, serviços disponíveis na Web, geradores de interfaces gráficas, entre outras.

A estrutura desenvolvida e apresentada permite a sua reutilização e aplicação para qualquer outro modelo de simulação integrante da suíte do DSSAT, tornando possível a integração entre diferentes modelos, maximizando a capacidade de desenvolvimento e facilitando a disponibilização dos resultados.

Devido a esses fatores, esta pesquisa está voltada, atualmente, ao desenvolvimento de um simulador genérico de culturas, aliado a modelos de doenças/epidemias. Este simulador visará à elaboração de um modelo genérico de simulação de culturas que utilizará dados meteorológicos obtidos de estações automáticas e de prognósticos de tempo, processando as informações por meio de seus modelos simples ou complexos, para a simulação da cultura e de suas doenças, podendo gerar alertas de risco de epidemias, distribuindo a informação aos usuários do sistema. Com os dados observados, o sistema fornecerá informações sobre o comportamento passado ou recente das doenças. Com os prognósticos de tempo é possível a predição antecipada de fenômenos na cultura e de doenças. Com isso, pretende-se desenvolver e implementar um sistema de previsão de doenças para as culturas que integram a Produção Integrada, nas diferentes regiões produtoras do Brasil, visando contribuir para racionalizar o uso de agrotóxicos, para a obtenção de produtos com qualidade certificada, aumentando a competitividade da cadeia produtiva e disponibilizando alimentos seguros para o consumidor.

Referências

[1] ADAMSON, C. What is java. Disponível em: <http://www.onjava.com/pub/a/onjava/2006/03/08/what-is-java.html?page=1>. Acesso em: 13 abr. 2007.

[2] APACHE Software Fundation. Apache Struts. Disponível em: <http://struts.apache.org/>. Acesso em 01 abr. 2007.

[3] BAUER, C.; KING G. **Java Persistence With Hibernate**. Manning, 2005.

[4] BECK, H. W. et al. Object-oriented approach to crop modeling: Concepts and issues. In: WORD CONGRESS OF COMPUTERS IN AGRICULTURE AND NATURAL RESOURCES, 2002. **Proceedings...** 2002. p. 297-305.

[5] CORAZZA D. Utilizando design pattern mvc no desenvolvimento de aplicações web. Monografia de Trabalho de Conclusão de Curso. 2002. Disponível em: <http://inf.upf.br:8080/bibdig/salvarArquivo.jsp?chave=59>. Acesso em: 19 mar. 2006.

[6] COX, H. User's guide to whoppercropper. 2006. Disponível em: <http://www.apsru.gov.au/apsru/>. Acesso em: 17 abr. 2007.

[7] CPTEC – Centro de Previsão de Tempo e Estudos Climáticos. Portal previsões numéricas. Disponível em: <http://www.cptec.inpe.br/prevnum/exp_eta.shtml>. Acesso em: 11 mai. 2007.

[8] DELPONTE, E. M. et al. Giberela do trigo: aspectos epidemiológicos e modelos de previsão. **Fitopatologia Brasileira**, v. 29, n. 6, p. 587-605, 2004.

[9] DE WIT, C. T. Coordination of models. In: PENNING DE VRIES, F.W.T.; VAN LAAR, H.H. (Ed.). **La productivite des paturages saheliens:** une etude des sols, des vegetations et de l'exploitation de cette ressource naturelle. Wageningen: PUDOC, 1982.

[10] FARIA, R. T.; BOWEN, W. T. Evaluation of DSSAT soil-water balance module under cropped and bare soil conditions. **Brazilian Archives of Biology and Technology**, v. 46, n. 4, p. 489-498, 2003.

[11] HILLSIDE. Patterns library. Disponível em: <http://hillside.net/patterns>. Acesso em: 21 mar. 2007.

[12] HUNT, L. A.; PARARAJASINGHAM, S. Cropsim-wheat: A model describing the growth and development of wheat. **Canadian Journal Plant Science**, v. 75, p. 612-632, 1995.

[13] HUNT, L.A. Modelo de simulação cropsim. Comunicação pessoal. 2007. Disponível em: <thunt@uoguelph.ca>. Acesso em: 23 jan. 2007.

[14] ICASA. International Consortium for Agricultural Systems Applications. Disponível em: <http://icasa.net>. Acesso em: 14 abr. 2007.

[15] JONES, J. W. et al. The dssat cropping system model. **European Journal of Agronomy: Modelling Cropping Systems: Science, Software and Applications**, v. 18, n. 3/4, p. 235-265, 2003.

[16] JONES, J. W.; RITCHIE, J. T. **Crop Growth Models**. American Society of Agricultural Engineers, 1990.

[17] KEATING, B. A. et al. An overview of apsim, a model designed for farming systems simulation. **European Journal of Agronomy**, v. 18, n. 22, p. 267-288, 2003.

[18] KEEN, R. E.; SPAIN, J. D. **Computer simulation in biology:** a basic introduction. New York: Wiley-Liss, 1992.

[19] KOGAN, M. Integrated pest management:historical perspectives and contemporary developments. **Annual Review of Entomology**, v. 43, n. 1, p. 243-270, 1998.

[20] LEAL, S. Modelação matemática uma proposta metodológica para o curso de economia. Dissertação de Mestrado. 1999. Disponível em: <http://www.eps.ufsc.br/disserta99/leal>. Acesso em: 30 nov. 2006.

[21] MAVROMATIS, T. et al. Developing Genetic Coefficients for Crop Simulation Models with Data from Crop Performance Trials. **Crop Sci**, v. 41, n. 1, p. 40-51, 2001.

[22] NORWOOD, F. B.; ROBERTS, M. C.; LUSK, J. L. Reply: Ranking crop yield models. **American Journal of Agricultural Economics**, v. 88, n. 4, p. 1111-1112, 2006.

[23] PAPAJORGJI, P. J.; PARDALOS, P. M. **Software Engineering Techniques Applied to Agricultural System - An Object-Oriented and UML Approach**. Springer, 2006.

[24] PORTER, C. H.; BRAGA, R. P.; JONES, J. W. An approach for modular crop. model development. 1999. Disponível em: <http://www.icasa.net/modular/pdf/modular.pdf>. Acesso em: 13 abr. 2007.

[25] R Development Core Team. **R: A Language and Environment for Statistical Computing**. Vienna: R Foundation for Statistical Computing, 2006.

[26] REYNOLDS, J. F.; ACOCK, B. Modularity and genericness in plant and ecosystem models. **Ecological Modelling**, v. 94, n. 1, p. 7-16, 1997.

[27] RIBEIRO JÚNIOR, P. J. Introdução ao R. 2001. Disponível em: <http://www.leg.ufpr.br/Rtutorial>. Acesso em: 10 abr. 2007.

[28] ROBERTS, C. A.; DESSOUKY, Y. M. An Overview of Object-Oriented Simulation. **Simulation**, v. 70, n. 6, p. 359-368, 1998.

[29] RSERVE. Rserve – Binary R server. 2007. Disponível em: <http://www.rforge.net/Rserve/>. Acesso em: 15 abr 2007.

[30] SHANNON, R. E. **Systems simulation:** the art and science. New Jersey: Prentice-Hall, 1975.

[31] SILVA JÚNIOR, V. V. Perspectivas para uso da linguagem R. 2005. Disponível em: <http://mov.void.cc/articles/r-perspectivas/r-perspectivas.pdf>. Acesso em: 10 abr. 2007.

[32] SILVEIRA, I. F. Linguagem Java. 2007. Disponível em: <http://www.infowester.com/lingjava.php>. Acesso em: 13 abr. 2007.

[33] SUN Microsystems. Designing enterprise applications with the j2ee platform (second edition). 2002. Disponível em: <http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e>. Acesso em: 15 abr. 2007.

[34] SUN Microsystems. The java technology phenomenon. 2007. Disponível em: <http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html>. Acesso em 13 abr. 2007.

[35] SWANEY, D. P. et al. Using crop models for management: Impact of weather characteristics on irrigation decisions in soybeans. **Transactions of the American Society of Agricultural Engineers**, v. 26, n. 6, p. 1808-1814, 1983.

[36] URBANEK, S. Rserve a fast way to provide R functionality to applications. 2003. Disponível em: <http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Drafts/Urbanek.pdf>. Acesso em: 15 abr. 2007.

[37] VERHAGEN, A.; CONIJN, S.; SCHAPENDONK, A. H. C. M. Quickscan of simulations models. 2001. Disponível em: <http://library.wur.nl/wasp/bestanden/LUWPUBRD_00121624_A502_001.pdf>. Acesso em: 01 dez. 2006.