A Construção de um Gerador de Programas Aplicativos segundo Conceitos de Análise de Domínios

RENATO FILETO
fileto@cnptia.embrapa.br

CARLOS ALBERTO ALVES MEIRA
CLEVAN RICARDO COSTA
SILVIA MARIA FONSECA SILVEIRA MASSHURÁ

Empresa Brasileira de Pesquisa Agropecuária - EMBRAPA
Centro Nacional de Pesquisa Tecnológica em Informática para a Agricultura - CNPTIA
Cidade Universitária Zeferino Vaz - Campus da UNICAMP
Caixa Postal 6041, CEP 13083-970, Campinas-SP

Abstract

This paper is related with the experience acquired in the development of the GFMS, a source code generator intended to aid the construction of application programs, for a specific application field, from analysis and project information. The GFMS is part of a greater effort called FMS, whose goal is the development of an environment directed to the semi-automatic generation of software for agricultural activities management, through the use of domain analysis concepts and techniques.

We present the basic principles employed, the general architecture of the code generator, as well as some examples and characteristics of the application programs produced by the FMS and indications of the methods to be used for the development and documentation of them. We expect that the experience described here be helpful to check the domain analysis concepts in a practical situation and to encourage and help resembling projects in other application fields.

KEY WORDS: Domain Analysis, Application Generators, Software Reuse.

1. Introdução

As ferramentas para auxiliar a construção de software mais utilizadas atualmente são aquelas que visam auxiliar o tratamento de aspectos específicos de programação, tais como bancos de dados, linguagens de consulta a bases de dados, geradores de relatórios e geradores de interfaces gráficas. Há ainda ferramentas CASE [Gim95] para auxiliar a especificação e o projeto de software, em geral cobrindo também aspectos específicos como o projeto da estrutura do código ou do modelo de dados. Algumas dessas ferramentas permitem a geração automática de código, com limitações especialmente no que se refere à funcionalidade relativa à área de aplicação à qual o software se destina.

Outra alternativa para se conseguir uma certa automação do processo de desenvolvimento de software é a construção de ferramentas para auxiliar a produção de software para áreas

específicas. Em muitas áreas de aplicação é necessário produzir um amplo conjunto de programas, para atender diversas peculiaridades de clientes, atividades ou métodos de trabalho. Geralmente, esses programas aplicativos podem ter diversos aspectos de seu funcionamento padronizado e, muitas vezes, possuem mais funcionalidade em comum do que específica. Isso favorece o reuso de componentes de software, seja na forma de especificação e projeto, seja na forma de código fonte.

A motivação para esta abordagem é obter uma redução no custo de produção de um elenco de programas aplicativos, dentro de uma área de aplicação delimitada, através do reuso de componentes. O custo de produção dos componentes reutilizados fica diluído na grande quantidade de programas aplicativos produzidos e instalados. Contudo, para se conseguir este efeito são necessárias técnicas adequadas para o desenvolvimento e manipulação desses componentes reusáveis, que permitam utilizá-los com facilidade na elaboração de programas executáveis.

Com a automação do processo de composição de programas aplicativos empregando informação reusável, além do aumento de produtividade, pode-se também obter ganhos na qualidade do software produzido, pela sua padronização, aderência à área de aplicação e eliminação de erros de programação.

A análise de domínios trata justamente das questões relativas à produção de software dentro de áreas de aplicação específicas. Segundo Prieto-Díaz e Arango [DA91], a análise de domínios é o processo de identificar e organizar conhecimento a respeito de alguma classe de problemas – o domínio do problema – a fim de suportar a descrição e a solução desses problemas. Diversas técnicas têm sido propostas para identificação, aquisição, representação, desenvolvimento, verificação e validação de componentes reusáveis, além da própria coordenação do processo de produção de software com informação reusável. Diversos ambientes de suporte têm sido propostos e alguns experimentos de implementação foram realizados e relatados na literatura [Neig84,Free87,DA91], com experiências bem sucedidas e promissoras.

1.1 Objetivos Gerais

Este trabalho descreve um gerador de código fonte, integrante de um ambiente de desenvolvimento de programas aplicativos para o domínio de administração rural. Considera-se a administração de uma propriedade rural dividida em dois grandes segmentos: a gerência das atividades do sistema produtivo e o planejamento dessas atividades. A modelagem e o acompanhamento das atividades são pré-requisitos para viabilizar o planejamento das mesmas. Portanto, a gerência é o domínio onde se pretende atuar inicialmente. O objetivo é obter sistemas de informação para facilitar o acesso e a manutenção de informações sobre as atividades que ocorrem em uma propriedade.

A demanda por software nesta área é imensa. A construção de um elenco completo de programas suficientemente detalhados para apoiar cada atividade do setor agropecuário em toda a diversidade de ecossistemas, categorias de propriedades e técnicas de cultivo e manejo existentes em nosso país é economicamente inviável, utilizando técnicas convencionais de produção de sistemas de informação. Entretanto, a análise das necessidades do setor agropecuário tem sugerido que as mesmas são suficientemente simples para possibilitar a construção de um ambiente de desenvolvimento, com o propósito de automatizar o processo de produção de programas aplicativos [FM94].

Ao contrário do problema da construção de geradores de aplicações genéricos ou de recursos para reuso de software em geral, onde diversos projetos têm fracassado e outros têm respondido apenas timidamente às expectativas inicialmente levantadas, a construção de geradores de aplicações para domínios restritos tem alcançado bons resultados, contando inclusive com resultados concretos. O projeto FMS[1] é mais um esforço neste sentido. A pretensão inicial é obter uma ferramenta para auxiliar a confecção de software destinado à gerência de propriedades agrárias, embora as técnicas utilizadas sejam suficientemente genéricas e expressivas para a cobertura de outros domínios, inclusive mais amplos.

Os programas aplicativos produzidos no ambiente FMS são padronizados e freqüentemente singelos, em termos da aparência e dos recursos empregados no produto final, para possibilitar o próprio processo de geração (e, eventualmente, por questões financeiras que inviabilizam a inclusão dos melhores produtos do mercado para a manipulação de bases de dados e a confecção das interfaces, por exemplo). Contudo, as primeiras experiências têm demonstrado que estes programas podem perfeitamente atender às necessidades e expectativas de grande parte dos usuários, respondendo a uma porção considerável da demanda.

Embora o objetivo do FMS possa ser considerado modesto – a construção de um gerador de aplicações restrito ao domínio de administração rural – este não constitui uma tarefa trivial e não se observa muitos relatos de experiências semelhantes na literatura. A construção do ambiente FMS, o qual já se encontra razoavelmente sofisticado e parcialmente operacional, envolveu o emprego de diversas técnicas e um projeto minucioso de alguns aspectos da sua arquitetura. A boa utilização deste ambiente requer ainda o emprego de métodos de desenvolvimento e documentação adequados, questão apenas esboçada neste trabalho.

2. Fundamentos e Técnicas Empregados

2.1 A Abordagem DRACO

A abordagem DRACO para a construção de software a partir de componentes reusáveis, proposta por Neighbors [Neig84,Neig89], provê grande parte da base conceitual do ambiente FMS. Os experimentos com os protótipos do sistema Draco, relatados por Neighbors, contribuíram muito para o aperfeiçoamento das técnicas propostas e para a definição de uma arquitetura geral para geradores de aplicações em domínios restritos. Freeman [Free87] efetuou uma análise conceitual da abordagem Draco, formalizando diversos dos conceitos e técnicas propostos e avaliando a sua contribuição, de acordo com princípios fundamentais da engenharia de software.

Dentre os conceitos e técnicas da ciência da computação e da engenharia de software que possibilitaram o sistema Draco e são utilizados também no FMS, pode-se destacar o seguinte.

- Técnicas de meta-compiladores, para a construção de tradutores fonte a fonte, os quais permitem o refinamento gradativo das especificações, escritas em linguagens de alto nível, até chegar ao código fonte, escrito em linguagens de programação convencionais, que possa ser compilado para se obter programas executáveis. Um conjunto de tradutores permite a estruturação do processo de desenvolvimento em níveis de abstração bem

[1] *Farm Management Systems*: nome utilizado para apresentação em *workshop* internacional e adotado como sigla do projeto.

definidos e auxilia o processo de reuso das especificações. Eles efetuam a montagem do código fonte de forma automatizada.

- Técnicas de modelagem de domínios, para a concepção de múltiplas linguagens de alto nível, especializadas na especificação dos vários domínios ou aspectos das aplicações, resultando em redes de domínios. A decomposição da linguagem de especificação em múltiplas sublinguagens contribui para a organização das informações, promovendo também a abstração e permitindo a representação direta dos objetos e das operações a serem modelados, de forma bem próxima ao modo como estes são percebidos no mundo real.

2.2 Processamento de Eventos

Uma característica importante do ambiente FMS é a presença de recursos para embutir no software gerado a capacidade de programação de atividades. Na ocorrência de determinados eventos (notificações de realização de atividades ou de quaisquer ocorrências relevantes do mundo real), outros eventos podem ser programados ou cancelados, sendo os registros das atividades pendentes armazenados em bases de dados, para serem acessados posteriormente via relatórios ou processamento de eventos. O processamento de um evento pode envolver também a modificação dos valores dos atributos de uma ou mais entidades da aplicação.

Através deste mecanismo, originário do conceito de sistemas reativos, é possível embutir no software aplicativo conhecimentos detalhados sobre o funcionamento de um determinado negócio, de modo que o próprio sistema vai se alimentando com os dados referentes às atividades que podem ser previstas. Com as informações obtidas pelo uso deste mecanismo, pode-se alertar o usuário sobre os procedimentos corretos, facilitar a programação de atividades e eliminar grande parte do trabalho de digitação.

3. A Estrutura e o Funcionamento do GFMS

3.1 A Técnica de Tradução

Para facilitar a implementação dos diversos tradutores necessários ao GFMS foi utilizado o GEDAI [Mei91,MM91,MM93], uma ferramenta que provê funções básicas para a construção da árvore sintática relativa à especificação e recursos para a composição dos fontes a serem gerados, na forma de um interpretador de uma linguagem denominada LDP (Linguagem de Definição de Produtos). O interpretador LDP recebe como entrada a árvore sintática, construída com o uso de rotinas do GEDAI, e um ou mais *templates*, isto é, definições de como montar cada arquivo a ser gerado, escritas na LDP, e devolve como saída um arquivo de produto para cada *template* fornecido. Os comandos da LDP permitem formatar as partes fixas do código do produto e utilizam rotinas que percorrem a árvore sintática de modo a obter as informações dependentes da especificação de cada programa aplicativo em particular, para completar a composição dos produtos.

Para confeccionar os analisadores léxicos e sintáticos necessários ao GFMS foram utilizados o LEX e o YACC, respectivamente [LMB92]. As ações relativas às regras de produção da gramática realizam as chamadas às rotinas do GEDAI que permitem construir a árvore sintática no formato reconhecido pelos comandos da LDP.

O processo de tradução efetuado com o uso do GEDAI permite estabelecer dois níveis distintos de implementação dos tradutores. O primeiro nível, denominado nível de *parser*, diz respeito à definição das sublinguagens de especificação, utilizadas pelo desenvolvedor de software aplicativo no FMS e à implementação dos *parsers* das mesmas. É o nível no qual as alterações apresentam os efeitos mais dramáticos para o usuário do GFMS. Portanto, é fundamental que as gramáticas das sublinguagens de especificação sejam cuidadosamente projetadas, para evitar alterações constantes.

O segundo nível é denominado nível de *templates*. Este é o nível em que são definidos os formatos dos fontes (ou arquivos intermediários) a serem gerados pelos tradutores. Alterações neste nível podem ser efetuadas sem a necessidade de modificar as gramáticas das sublinguagens de especificação. Por exemplo, é possível modificar a linguagem alvo dos programas aplicativos, a estrutura do código gerado ou até mesmo o banco de dados utilizado, sem a necessidade de alterar o formato das especificações de entrada. Assim, as mesmas especificações podem gerar o programa para diferentes plataformas ou com padrões de interface distintos, por exemplo.

O conteúdo das especificações caracteriza o nível de desenvolvimento dos programas aplicativos, utilizando a abstração provida pelas sublinguagens de especificação. Estas especificações são mais simples de tratar que código fonte em linguagens de programação convencionais, tanto em termos de seu levantamento, quanto para efeito de manutenção e identificação das possibilidades de reuso, pois esconde muitos dos detalhes de implementação do software, realçando as questões relativas à área de aplicação.

As gramáticas das sublinguagens de especificação constituem o elo de comunicação entre o GFMS e o AEsp, o assistente de especificação do FMS [Mass94,Mass95]. Este último tem a função de auxiliar na captura das especificações e montar um banco de especificações, devidamente catalogadas para reuso, possibilitando maior produtividade na construção de programas aplicativos a medida que se for acumulando mais conhecimentos sobre o domínio de aplicação.

A Figura 1 ilustra a estrutura do GFMS acomodando os níveis de implementação dos tradutores e os subdiretórios para o desenvolvimento dos diversos programas aplicativos. A organização de todos os arquivos fonte, objeto e executáveis do GFMS, bem como dos arquivos com código dos programas aplicativos em LC-FMS, dados intermediários e código gerado está implementada numa estrutura de subdiretórios exatamente como ilustrada na Figura 1.

Cada nível desta estrutura compreende uma ou mais instâncias da estrutura subjacente a ele. Novas instâncias podem ser criadas a qualquer momento, replicando a subárvore da estrutura original a partir do nível desejado. As evoluções e manutenções podem então ser realizadas na réplica. As instâncias, que evoluem independentemente, podem ser relativas a:

- versões do GFMS ou dos programas aplicativos, como ocorre, por exemplo, no nível de parser ilustrado na Figura 1, onde há atualmente uma única versão;
- alternativas de implementação, como ocorre no nível de templates ilustrado, no qual são oferecidas as opções de geração de código dos programas aplicativos para utilização de bases de dados Cbase [Stev87] ou Codebase [CB93], sem modificar os parsers da LC-FMS e utilizando as mesmas especificações de entrada;
- diferentes programas aplicativos em desenvolvimento no nível de especificação, onde é ilustrado o desenvolvimento paralelo dos programas aplicativos *SisCoReb* (Sistema de Controle de Rebanho Leiteiro) e *ModFaz* (Modelo de Fazenda).

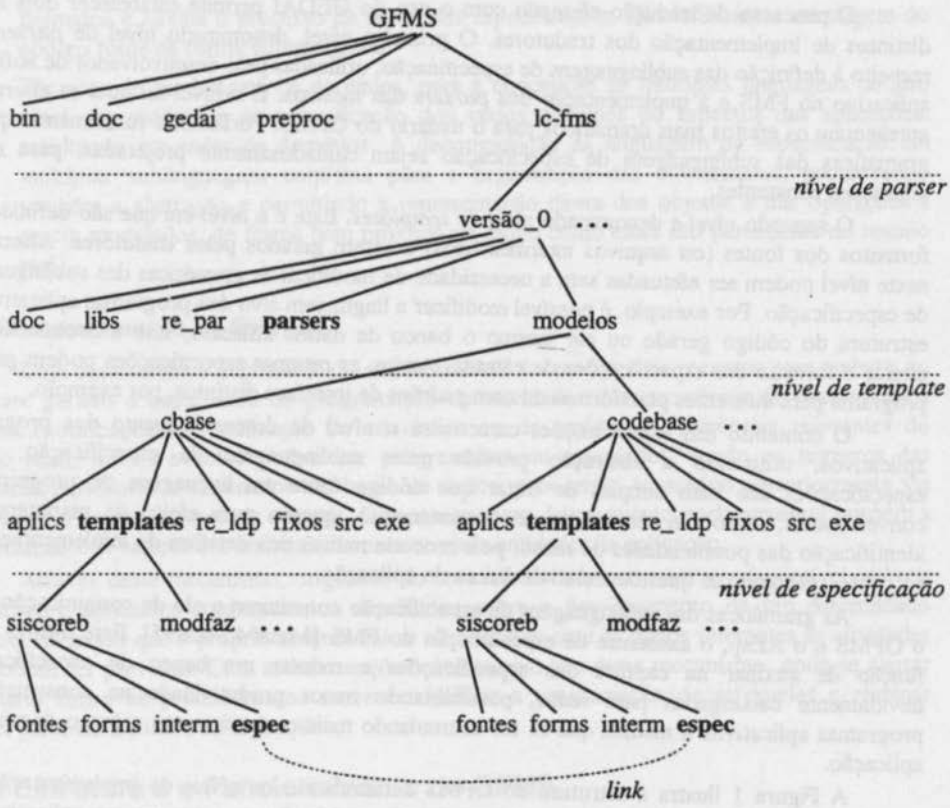

Figura 1: A estrutura de níveis de implementação do GFMS

Observe que, embora se tenha as opções de gerar os programas aplicativos para bases de dados Cbase ou Codebase, há um único repositório de especificações para cada aplicação, sendo a conexão realizada através de um *link*.

Esta estrutura permite uma boa flexibilidade para o controle de versões, experimentos de alternativas de implementação e o desenvolvimento de diversos programas aplicativos em paralelo. Além disso, há uma definição clara dos níveis de atuação nas manutenções do gerador e das aplicações nele desenvolvidas:

- nível de parser - alteração da LC-FMS,
- nível de templates - alteração do código gerado (por exemplo, mudanças de estilo ou estrutura, da linguagem de programação ou das ferramentas utilizadas para compor os programas aplicativos),
- nível de especificação - alteração das funções do programa aplicativo.

Os efeitos colaterais ocasionados por uma modificação em um determinado nível são facilmente classificados e analisados, pois se propagam através da estrutura hierárquica e de *links*, de uma forma bem definida.

3.2 As Sublinguagens de Composição

A linguagem de composição de programas aplicativos do FMS (LC-FMS) é fragmentada em diversas sublinguagens, seguindo as recomendações da abordagem Draco, sobre as vantagens de múltiplas linguagens de especificação. Essas sublinguagens são descritas a seguir.

- **Formulários (LC-FORM)** é a sublinguagem de especificação das telas de entrada e saída de informações e da estrutura de menus de cada programa aplicativo. Através dela, é definida toda a interface com o usuário, sendo que as definições dos campos de edição de dados são utilizadas também para inferir o modelo de dados da aplicação. As especificações de formulários são o ponto de partida para o processo de geração de código, sendo utilizadas inclusive na composição do dicionário de dados.
- **Ajuda (LC-HELP)** permite a especificação de mensagens de ajuda associadas aos campos de edição e aos formulários, que podem ser acessadas durante a execução do software aplicativo. A especificação de mensagens de ajuda é útil também na composição do manual de referência do software aplicativo, automatizando parte da sua redação.
- **Consistência (LC-CONSIST)** é utilizada para especificar as regras de consistência a serem satisfeitas pelas informações inseridas nos campos de edição dos formulários de entrada de dados. Através da sublinguagem de consistência podem ser especificadas as condições de erro a serem testadas nas entradas de dados e as mensagens a serem apresentadas ao usuário do software aplicativo na ocorrência dessas condições de erro.
- **Síntese de Campos (LC-APPLY)** permite efetuar o preenchimento de campos de formulários com valores obtidos ou calculados a partir dos valores preenchidos para outros campos e valores oriundos das bases de dados. Ela é utilizada nas situações em que se faz necessário o fornecimento de *feedback* para o usuário.
- **Eventos (LC-EVENT)** é a sublinguagem onde são especificadas as operações a serem efetuadas na ocorrência de cada evento. A ocorrência de um evento quase sempre modifica o conjunto de eventos pendentes, armazenados na forma de registros de bases de dados e, muitas vezes, requer atualizações nos atributos de diversas entidades. As especificações de manipulação de eventos embutem muito conhecimento especializado a respeito da área de aplicação, sob a forma de operações de remoção, inserção e atualização de registros das bases de dados, com as situações em que estas operações devem ser efetuadas.
- **Ações (LC-ACTION)** possui recursos básicos da álgebra relacional para possibilitar a manipulação de bases de dados, de modo a permitir a construção de bases intermediárias para o tratamento dos eventos pendentes e para a emissão de relatórios.
- **Relatórios (LC-REPORT)** serve para efetuar a formatação dos relatórios (e não o processamento dos dados para a obtenção das informações necessárias). Os relatórios sempre são emitidos a partir de bases de dados intermediárias, produzidas com a utilização da sublinguagem de ações.

Além dos tradutores para as especificações nas sublinguagens mencionadas acima, o GFMS tem alguns outros tradutores intermediários, essenciais ao processo incremental de geração do código fonte do programa aplicativo. Entre as especificações intermediárias utilizadas internamente pelo GFMS, destaca-se o dicionários de dados. As informações do dicionário de

dados são necessárias ao longo de todo o processo de desenvolvimento das aplicações, pois os tradutores precisam sempre conferir as características de tipo e tamanho dos atributos referenciados e a pertinência dos mesmos aos formulários e bases de dados utilizados.

3.3 A Rede de Domínios

Observando-se os requisitos de expressividade das sublinguagens de composição do FMS, optou-se pelo estabelecimento de construções gramaticais análogas em várias dessas sublinguagens. Através de procedimentos de análise, estruturação e compatibilização das gramáticas das sublinguagens do GFMS já foram identificados e formalizados quatro subdomínios intermediários na implementação dos tradutores, relativos às construções gramaticais necessárias às sublinguagens e à montagem do código fonte a ser gerado, os quais são descritos a seguir.

- **Seqüências de Comandos** diz respeito ao tratamento de listas de comandos, tais como atribuições de valores de dados, mensagens a serem apresentadas na tela e chamadas de funções, possivelmente entremeadas com testes para alterar a execução seqüencial, executando alguns comandos somente quando determinadas condições forem satisfeitas. As necessidades desse domínio são supridas adotando-se seqüências de comandos com eventuais aninhamentos de IF análogas às das linguagens procedurais, como C e Pascal.
- **Buscas em Bases de Dados** lida com especificações de busca de registros nas bases de dados, de modo a esconder do desenvolvedor de programas aplicativos na LC-FMS os detalhes de programação e das rotinas de acesso ao banco de dados.
- **Expressões Aritméticas e Lógicas** envolve o tratamento de expressões algébricas e lógicas, com valores de dados de diversos tipos (INTEIRO, REAL, CADEIA DE CARACTERES, DATA e LÓGICO). No processamento dessas expressões freqüentemente é necessário acessar o dicionários de dados, para obter o tipo e o tamanho dos operandos, de modo a determinar se os operadores utilizados se aplicam e inferir a sua semântica em cada caso, tratando eventuais incompatibilidades.
- **Formatação e Documentação de Código** diz respeito aos padrões e procedimentos empregados na formatação e documentação do código gerado pelo GFMS. Há um conjunto de rotinas em LDP e rotinas em C utilizadas pelos tradutores para auxiliar na tarefa de geração de código.

A Figura 2 ilustra a rede de domínios atualmente implementada no GFMS. Cada domínio embute os recursos de representação dos domínios acessíveis a partir dele por um caminho descendente no grafo e provê abstrações adicionais, que são traduzidas para os recursos existentes nos níveis inferiores. A especificação submetida a um domínio pode utilizar, na sua notação, tanto abstrações daquele domínio quanto recursos de representação dos níveis inferiores no grafo. Os recursos de tradução do domínio convertem as abstrações em representações de níveis inferiores, enquanto o que já estiver escrito em notações de domínios inferiores é deixado intacto.

A LC-FMS vista pelo usuário do GFMS compreende as sete sublinguagens apresentadas na seção 3.2. O subdomínio de seqüências de comandos é utilizado pelas sublinguagens de ações, eventos e consistência e o subdomínio de buscas em bases de dados, pelas sublinguagens de eventos, consistência e síntese de campos (*apply*). Esses dois subdomínios recorrem ao subdomínio de tratamento de expressões algébricas e lógicas. Todas as sublinguagens utilizam o

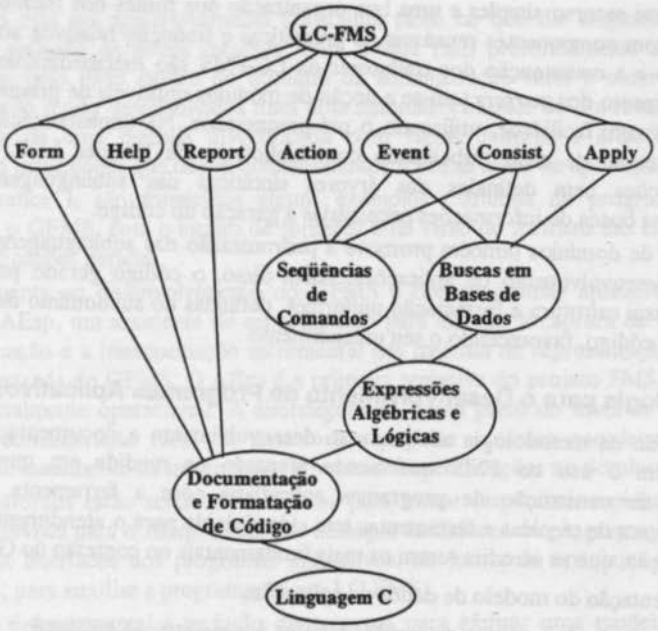

Figura 2: A Rede de Domínios do GFMS

subdomínio de formatação e documentação de código, para a elaboração do código fonte em linguagem C.

Cada domínio é caracterizado por sua linguagem de representação de conhecimento. Contudo, na implementação física do GFMS, existem tradutores executáveis somente para as sublinguagens integrantes da LC-FMS (segundo nível de abstração da Figura 2, de cima para baixo) e para a linguagem C (nível de abstração mais baixo). As gramáticas e recursos de tradução dos domínios intermediários são incluídos textualmente no código fonte dos tradutores das sublinguagens de especificação, através de um pré-processador construído especialmente para este fim.

Tal pré-processador, simplesmente inclui arquivos texto dos repositórios de componentes reusáveis de parsers e *templates* (subdiretórios *re-par* e *re-ldp* da Figura 1), nos arquivos fonte correspondentes dos tradutores das sublinguagens (código Lex/Yacc e LDP, respectivamente). Ele fornece um recurso similar à diretiva *#include* do pré-processador da linguagem C, com alguns recursos adicionais para o tratamento de diversos repositórios de arquivos para inclusão.

No código fonte dos tradutores, denota-se a inclusão de um arquivo relativo a um subdomínio por:

#modulo{*<nome_arquivo>*}

O pré-processamento de um fonte de tradutor é realizado através do comando:

pre_gfms *<nome_fonte>* *<diretório_dos_módulos_a_incluir>*

Com esse recurso simples e uma boa organização dos fontes dos tradutores do GFMS e dos arquivos com componentes reusáveis de gramáticas e tradução relativos aos subdomínios, a implementação e a manutenção dos tradutores da LC-FMS são extraordinariamente facilitadas. No desenvolvimento dos *parsers* tem-se a noção de módulos reusáveis de gramáticas, que podem ser conectados com facilidade, utilizando o pré-processador. Na implementação dos *templates*, tem-se a impressão de estar trabalhando com bibliotecas de funções LDP, as quais podem percorrer porções bem definidas das árvores sintáticas das sublinguagens, relativas aos subdomínios, na busca de informações necessárias à geração do código.

A rede de domínios também promove a padronização das sublinguagens, simplificando o processo de desenvolvimento de aplicações. Além disso, o código gerado para os programas aplicativos possui estrutura e formatação uniformes, definidas no subdomínio de documentação e formatação de código, favorecendo o seu entendimento.

3.4 Metodologia para o Desenvolvimento de Programas Aplicativos com o GFMS

O estudo da metodologia adequada ao desenvolvimento e documentação de programas aplicativos com o uso do FMS está sendo efetuado na medida em que se realizam os experimentos de construção de programas aplicativos com a ferramenta. Até o presente momento, a busca de técnicas e ferramentas tem sido dirigida para o atendimento a dois aspectos da documentação, que se acredita serem os mais fundamentais no contexto do GFMS:

- representação do modelo de dados da aplicação;
- representação da máquina de estados relativa aos eventos da aplicação.

A representação do modelo de dados constitui o aspecto estático da documentação. Ela deve determinar a estrutura sobre a qual são armazenados os dados da aplicação, relativos às suas entidades básicas e às atividades (eventos) realizadas e programadas. O modelo de entidade-relacionamento tem sido utilizado com bons resultados para o tratamento deste aspecto.

A representação das possíveis seqüências de ocorrência de eventos, por outro lado, envolve o aspecto dinâmico de alimentação das bases de dados onde os eventos são armazenados. É imprescindível uma representação gráfica geral das possíveis seqüências de eventos da aplicação, das condições para a ocorrência de cada evento e das operações a serem efetuadas no processamento de cada um. Acredita-se que seja viável algum tipo de diagrama de estados para atender a essas necessidades, o que ainda está se pesquisando.

4. Situação Atual e Extensões Futuras

Os primeiros programas aplicativos do FMS têm sido desenvolvidos programando-se diretamente na LC-FMS. O GFMS tem sido executado em plataforma UNIX, embora seja construído de forma a ser portável para PC. O código atualmente gerado é linguagem C para plataforma DOS, com chamadas a rotinas para acesso ao banco de dados (*CBase* [Stev87] ou *Codebase* [CB93]) e um *toolkit* para o tratamento da interface (*Turbo C Utilities* [KK92]).

O primeiro programa aplicativo desenvolvido com o GFMS é um sistema para auxiliar o controle de um rebanho leiteiro. Uma versão preliminar deste sistema encontra-se instalada em alguns centros de pesquisa sobre criação de gado e em algumas fazendas, para avaliação. Os resultados obtidos têm sido muito satisfatórios. Com as críticas e sugestões dos pesquisadores e produtores vem sendo elaborada uma nova versão do programa.

O GFMS está em aperfeiçoamento constante, tanto no que diz respeito à correção de falhas quanto à inclusão de novos recursos. Para avaliar mais profundamente a adequação do modelo proposto pelo FMS dentro do domínio de administração rural e obter subsídios para aperfeiçoa-lo, estão sendo desenvolvidos mais dois sistemas: controle de um rebanho de gado de corte e controle de uma fazenda diversificada com várias culturas e criações, sempre em colaboração com instituições que detêm conhecimentos sobre as áreas de aplicação.

No Apêndice I, são fornecidos alguns exemplos extraídos de programas aplicativos produzidos com o GFMS, com o intuito de fornecer uma visão do formato das especificações de entrada e das aplicações geradas.

Paralelamente ao desenvolvimento do GFMS e dos programas aplicativos, vem sendo desenvolvido o AEsp, um assistente de especificação para auxiliar a captura de informações do domínio de aplicação e a transformação incremental das mesmas na representação em LC-FMS, utilizada como entrada do GFMS. O AEsp é a primeira tentativa do projeto FMS neste sentido e encontra-se parcialmente operacional. A abordagem do AEsp parte do nível de abstração mais alto possível – administração rural, de uma forma genérica – e visa coordenar as atividades relacionadas ao levantamento de informações e reuso de especificações no domínio proposto.

Alguns esforços estão sendo direcionados para tornar disponível, rapidamente, parte da funcionalidade prevista para o AEsp, como por exemplo facilidades de programação visual para a especificação das interfaces dos programas aplicativos, na forma de um *front-end* executando acima do GFMS, para auxiliar a programação em LC-FMS.

Também é fundamental a inclusão de recursos para efetuar uma modelagem de dados consistente, robusta e versátil. É necessário um método adequado para aperfeiçoar o modelo de dados preliminar com as entidades básicas e seus atributos, derivado da especificação das interfaces, de modo a garantir a sua consistência segundo os conceitos de bancos de dados, antes da sua implementação física. Isso deverá ser solucionado com o acoplamento de uma ferramenta CASE ao ambiente FMS.

Outro problema do GFMS atualmente é a ausência de um dicionário de dados com primitivas de acesso eficientes que atendam às necessidades de manipulação de atributos de dados, testes de compatibilidade e manutenção da integridade dos acessos e atualizações a esses dados, durante os processos de captura das especificações e tradução das mesmas em código fonte. Sem um dicionário de dados adequado não é possível efetuar as conferências necessárias e fornecer informações a respeito de inconsistências no ponto do processo de desenvolvimento onde elas são provocadas. Assim, erros que poderiam ser reportados pelo assistente de especificação ou pelo GFMS, somente são detectados pelo compilador do código fonte gerado. Isso ocasiona problemas para o desenvolvedor, que precisa mapear o problema reportado em termos do código gerado para as representações de nível mais alto (LC-FMS ou representações do assistente de especificação), para poder solucionar o problema. Isso compromete o processo de desenvolvimento em níveis de abstração. Espera-se solucionar este problema com a implantação de um dicionário de dados, com uma *API* para efetuar acesso ao mesmo ao longo de todo o processo de desenvolvimento, desde o levantamento das especificações até a geração de código fonte.

A Figura 3 ilustra o fluxo de informações proposto para a evolução do ambiente FMS, contemplando a inserção da ferramenta para auxiliar a modelagem de dados e do dicionário de dados a ser utilizado ao longo de todo o processo de desenvolvimento de aplicações.

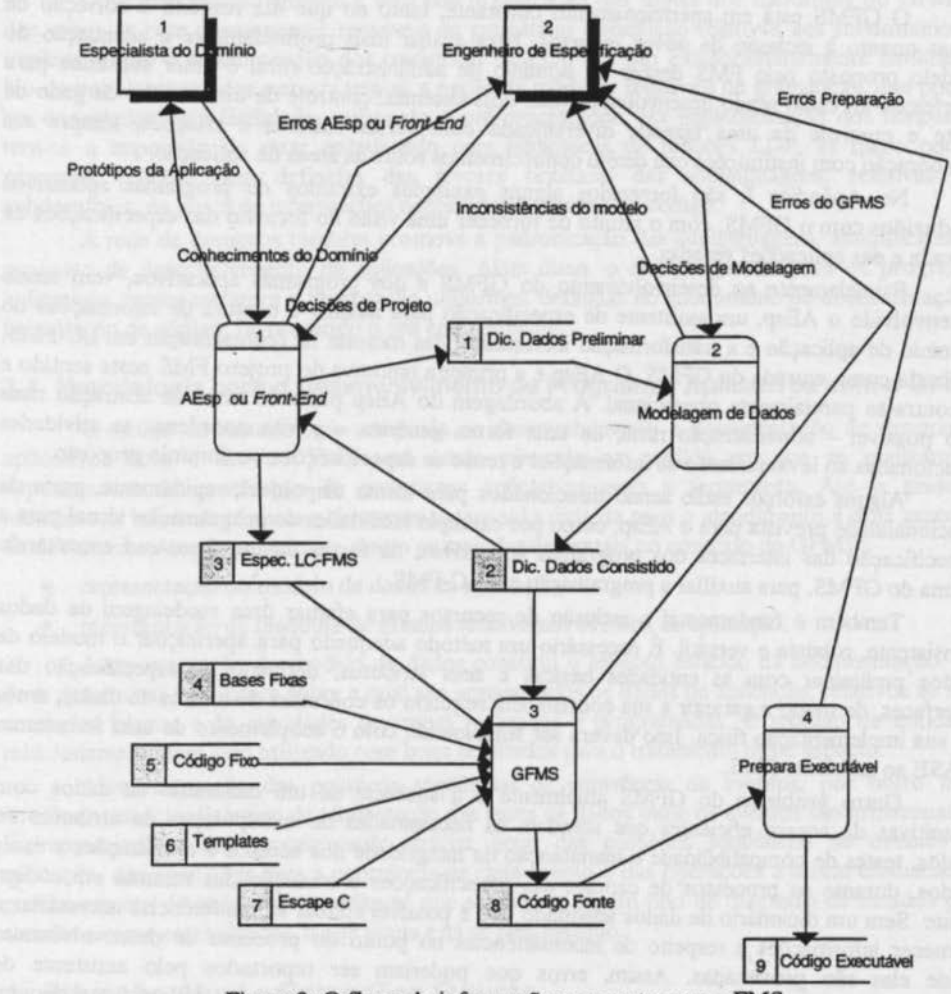

Figura 3: O fluxo de informações proposto para o FMS

Estão previstas ainda diversas outras adequações e aperfeiçoamentos no ambiente FMS, com o intuito de acompanhar a evolução tecnológica. Sempre que possível, serão agregadas ferramentas de mercado ao ambiente, de modo a facilitar e até viabilizar a sua evolução com os recursos técnicos e humanos de que se dispõe. Entre as ferramentas estudadas para acoplamento ao FMS estão geradores de interfaces gráficas por programação visual, bancos de dados, linguagens de consulta a bases de dados e geradores de relatórios, além de ferramentas CASE para auxiliar na especificação e no projeto das aplicações, fornecendo métodos adequados de desenvolvimento, documentação e gerência de configuração.

5. Conclusões

A adoção de técnicas e ferramentas para auxiliar a construção de software voltado para áreas de aplicação específicas é um caminho promissor para o atendimento à enorme demanda de software em determinadas áreas. Lançando mão de técnicas de levantamento de especificações, geração de código e reuso de componentes de software (informações de especificação, projeto e eventualmente, código em linguagem de programação de computador), em áreas de aplicação delimitadas, pode-se obter programas aplicativos de boa qualidade, através de um processo de produção semi-automatizado.

As técnicas de captura e reuso de especificações, integradas segundo conceitos de análise de domínios, são utilizadas no levantamento das funcionalidades específicas ao domínio de aplicação, isto é, dos conhecimentos sobre o domínio, a partir dos quais são definidos o modelo de dados, o fluxo e a forma de processamento das informações para a confecção do software aplicativo. Essas técnicas são úteis na captura do conhecimento especializado e refinamento do mesmo até chegar a um nível de formalização e detalhamento suficiente para permitir a geração automática de código em linguagem de programação de computador.

Contudo, a proposta de redes de domínios não dispensa o uso de ferramentas como bancos de dados, geradores de interfaces gráficas ou as chamadas linguagens de quarta geração para o desenvolvimento de sistemas de informação. As técnicas de levantamento de especificações facilitam a manipulação do conhecimento relativo à área de aplicação a que o software se destina, ao passo que as ferramentas de quarta geração auxiliam o tratamento de questões específicas de programação, como o gerenciamento de bases de dados e a construção de interfaces gráficas. Portanto, as duas abordagens são complementares, quando se trata de aumentar a produtividade do processo de desenvolvimento e a qualidade do produto.

A proposta do projeto FMS consiste em agregar técnicas e ferramentas da ciência da computação e da engenharia de software, na construção e evolução do ambiente de desenvolvimento de programas aplicativos voltados para a gerência de propriedades rurais, embora não se descarte a utilização do ambiente construído em outros domínios. A agregação de todas as facetas tecnológicas empregadas requer alguma criatividade na definição da estrutura geral do ambiente e, principalmente, organização e metodologia para integração de sistemas.

Referências

[CB93] **CODEBASE; version 5.0; user's guide**. Sequiter Software Inc., 1992.

[DA91] DÍAZ, R.P.; ARANGO, G. **Domain analysis and software systems modeling**. Los Alamitos/CA: IEEE Computer Society, 1991.

[FM94] FERRARETTO, M.D.; MASSHURÁ, S.M.F.S. **Projeto: ambiente de desenvolvimento de software para o domínio de administração rural - FMS**. Campinas/SP: EMBRAPA-CNPTIA, 1994. (Documento interno apresentado ao Sistema Embrapa de Planejamento - SEP)

[Free87] FREEMAN, P. A conceptual analysis of the Draco approach to constructing software systems. **IEEE transactions on software engineering**, v.se-13, n.7, p.830-844, Jul. 1987.

[Gim95] GIMENES, I.M.S. **Ferramentas CASE**. Recife: Simpósio Brasileiro de Engenharia de Software, 9., 1995 (Tutorial)

[KK92] KEYTE, K. **Turbo C Utilities, version 3.3 - reference manual**. Darmstadt/Germany: ESOC, 1992.

[LMB92] LEVINE, J.R.; MASON, T.; BROWN, D. **Lex & Yacc**. Sebastopol/CA: O'Reilly & Associates, 1992.

[Mass94] MASSRUHÁ, S.M.F.S.; FERRARETO, M.D.; MAXIMO, F.A.; MEIRA, C.A.A.; PASSOS, S.L.Z.; VISOLI, M.C. Aesp: um assistente de especificação. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 8., 1994, Curitiba. **Anais**. Curitiba: PUC-PR, 1994. p.311-324.

[Mass95] MASSRUHÁ, S.M.F.S. **Aesp: um assistente de especificação para administração rural**. Campinas/SP: UNICAMP-FEE, 1995. Dissertação Mestrado.

[Mei91] MEIRA, C.A.A. **Sobre geradores de aplicação**. São Carlos: USP-ICMSC, 1991. Dissertação Mestrado.

[MM91] MEIRA, C.A.A; MASIERO, P.C. Um gerador de aplicações para sistemas reativos. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 5., 1991, Ouro Preto. **Anais**. Ouro Preto: UFMG, 1991. p.45-59.

[MM93] MASIERO, P.C.; MEIRA, C.A.A. Development and instantiation of a generic application generator. **The journal of systems and software**, v.23, n.1, p.27-38, Oct. 1993

[Neig84] NEIGHBORS, J.M. The Draco approach to constructing software from reusable components. **IEEE transactions on software engineering**, v.se-10, n.5, p.564-574, Sep. 1984.

[Neig89] NEIGHBORS, J.M. Draco: a method for engineering reusable software systems. **Software reusability, concepts and models**, v.1, p.295-320, 1989. ACM Press.

[Pre92] PRESSMAN, R.S. **Software engineering - a practitioner's approach**. 3.ed., McGraw-Hill, 1992.

[Stev87] STEVENS, A. **C data base development**. Portland: Management Information Source, Inc., 1987.

Apêndice I: Exemplos de Aplicações

Programação de Eventos

```
%%EVENT ABORTO

REMOVE_SECTION
      PRECONDITION NONE;
      REMOVE SCHEDULED SECAG
      FOR :
            FIELD BRINCO == ABORTO.BRINCO;
      PRECONDITION NONE;
      REMOVE SCHEDULED PARTO
      FOR :
            FIELD BRINCO == ABORTO.BRINCO;

SCHEDULE_SECTION
      PRECONDITION
            REBANHO.UTIL_ANI == "Matriz" &&
            VDGRA.EX_AB == "Requerido" ||
            REBANHO.TIPO_ANI == "Vaca" && REBANHO.EST_REPR == "Seca";
      SCHEDULE EXA_VET
      SET :
            EV_ID          = NEW_VAL(EV_ID);
            BRINCO         = ABORTO.BRINCO;
            DATA_EV        = ABORTO.DATA_EV;
            TIPO_EXA       = "Pos-Aborto";
      ENDSET;

ACTION_SECTION
      UPDATE REBANHO
      FOR :
            FIELD BRINCO == ABORTO.BRINCO;
      SET :
            ESTADO = "Vazia";
            DTA_PCIO = ABORTO.DATA_EV + VDGRA.VDGR2;
            IF ABORTO.DATA_EV >= DTA_INSM + VDGRA.VDGR14 THEN
                  LACTACAO = LACTACAO + 1;
                  DTA_LACT = ABORTO.DATA_EV;
            ENDIF;
      ENDSET;

(FETCH
      RECORD REBANHO
      LOAD FROM REBANHO
      FOR :
            FIELD BRINCO == ABORTO.BRINCO;
      ACCESS_ERROR;
)
```

Funções de Consistência

```
FIELD CONSISTENCY AnimalNaoCadastrado (consisted_field)
      IF ( EXISTS REBANHO
           FOR :
             FIELD BRINCO == consisted_field; )
      THEN
           ERROR_MSG "BRINCO REFERENTE A ANIMAL JA' CADASTRADO";
      ENDIF;
ENDFUNC;


FORM CONSISTENCY CadastConsist ()
IF (TIPO_ANI == "Novilha") THEN
      IF ( ! (ESTADO IN ("Virgem", "Inseminada",
                         "Prenha", "Vazia")) )
       THEN
           ERROR_MSG "ESTADO DO ANIMAL E' INCOMPATIVEL COM O TIPO";
      ENDIF;
ELSE
      IF (TIPO_ANI == "Vaca") THEN
           IF ( ! (ESTADO IN ("Inseminada", "Prenha",
                              "Parida", "Vazia", "Seca")) ) THEN
                ERROR_MSG "ESTADO E' INCOMPATIVEL COM O TIPO";
           ENDIF;
      ELSE
           IF (ESTADO != "Nao usado") THEN
                ERROR_MSG "ESTADO E' INCOMPATIVEL COM O TIPO";
           ENDIF;
      ENDIF;
ENDIF;
```

Síntese de Campos

```
APPLY SintDiasLeite:
      PRECONDITION
            REBANHO.ESTA_LAC == S;
      SYNTHESIS
            DIAS_LEI = DATA_EV - REBANHO.DTA_LACT;
      DEFAULT
            0;
(FETCH
      RECORD REBANHO
      LOAD FROM REBANHO
      FOR :
            FIELD BRINCO == BRINCO; ACCESS_ERROR;
)
```

Uma Tela de um Programa Aplicativo Gerado

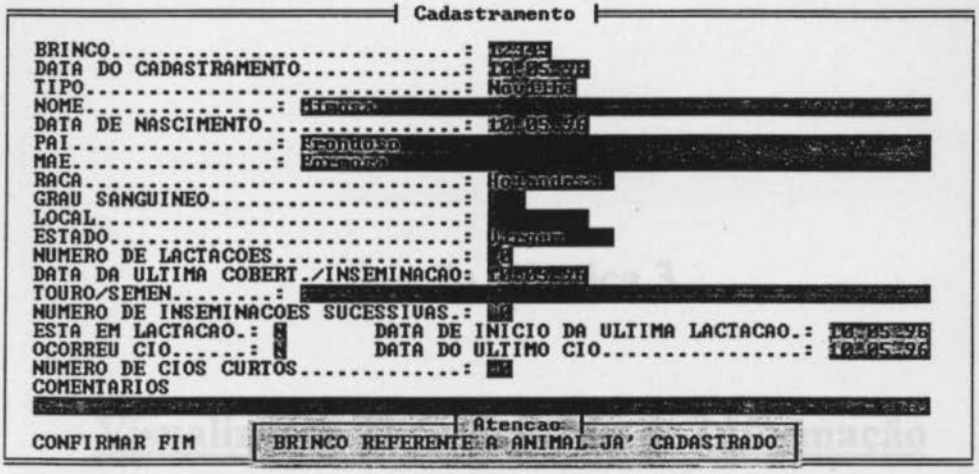

Figura 4: Tela de cadastramento de animais do sistema de controle de rebanho leiteiro

Um Relatório com as Atividades Previstas

```
Data: 10/5/1996      SISTEMA P/ CONTROLE DE REBANHO LEITEIRO      Pag.1
Relatório de atividades previstas após o cadastramento de um animal
De 01/01/1950 a 31/12/2049
================================================================================
BRINCO  DATA       EVENTO          PARAMETROS
=====   ========   =============   ====================================
12345   25.05.96   VACINACAO       Pneumoenterite
12345   08.08.96   VACINACAO       Aftosa
12345   08.08.96   VACINACAO       Brucelose
12345   07.09.96   VACINACAO       Carbunculo
12345   07.09.96   VACINACAO       Leptospirose
12345   07.09.96   VACINACAO       Raiva
12345   03.08.97   CIO
7 registro(s) processado(s)
```