AEsp: UM ASSISTENTE DE ESPECIFICAÇÃO

Massruhá, Silvia M.F.S.
e-mail: silvia@cnptia.embrapa.br

Ferraretto, Mário D. (*)
Máximo, Fernando A.
Meira, Carlos A. A.
Passos, Sérgio L. Z.
Visoli, Marcos C.

Centro Nacional de Pesquisa Tecnológica
em Informática para Agricultura - CNPTIA/EMBRAPA
(*) Universidade de São Paulo - USP

RESUMO:

Este trabalho descreve uma ferramenta, denominada AEsp, que visa auxiliar a captura das especificações das aplicações do domínio de administração rural e transformá-las, incrementalmente, em uma representação que pode ser traduzida para um programa operacional. Esta ferramenta é parte do ambiente FMS ("Farm Management System") que está sendo desenvolvido para a EMBRAPA. O FMS é um ambiente para a geração automatizada de aplicativos do domínio de administração rural.

Palavras-Chaves: assistente de especificação, análise de domínio, gerador de aplicação, reuso.

ABSTRACT:

This paper describes a tool, called AEsp, for capturing informal specifications of farm management domain applications and incrementally transforms them into a representation that can be translated into an operational program. This tool is part of the FMS (Farm Management System), a prototype software engineering environment for EMBRAPA, aimed at automatic generation of small farm management applications.

Keywords: specification assistant, domain analysis, application generator, reuse.

1) INTRODUÇÃO

O FMS é um ambiente de software para a geração automatizada de aplicativos do domínio de administração rural.

A abordagem utilizada pelo FMS prevê que durante a fase de especificação as informações de uma aplicação sejam capturadas, decompostas e armazenadas para posterior reutilização. As informações devem ser decompostas até um nível de representação para o qual estará disponível um gerador de código fonte, simplificando a fase de implementação [6,7]. Esta abordagem foi definida durante uma atividade denominada pré-análise do domínio.

Segundo Neighboors [18], cada problema tem sua própria linguagem que é denominada Linguagem do Domínio (LD), e a identificação desta linguagem é resultado de um processo denominado Análise de Domínio.

A análise de domínio, para Neighboors, produz os operandos e operadores de uma classe de aplicações, e, com base nesta análise, uma linguagem é construída e usada para especificar outros problemas do domínio em questão.

A análise de domínio, no escopo do Projeto FMS, foi realizada em duas etapas. A primeira etapa identificou o elenco de funções básicas e o modelo de fluxo de controle e dados típicos das aplicações do domínio de administração rural. A segunda etapa agregou esse conjunto de funções na forma de uma Linguagem de Composição do FMS (LC-FMS) e modelou as aplicações na forma de programas da classe reativa [7].

Sob o modelo do FMS, os requisitos do usuário são incrementalmente transformados em uma especificação descrita em LC-FMS, e a aplicação, gerada automaticamente a partir da especificação em LC-FMS, comporta-se como uma máquina de estados estendida [22]. Para suportar esta abordagem, a metodologia subjacente ao FMS é baseada em um modelo de níveis.

2) MODELO DE NÍVEIS DO FMS

O modelo de níveis do FMS propõe uma hierarquização entre os níveis do processo de transformação dos requisitos do usuário, basicamente devido aos métodos disponíveis para essa tarefa.

O modelo utilizado é o modelo PW de Lehman [11], no qual o nível não formalizado é descrito como nível conceitual e o nível onde há uma descrição formal da aplicação, ainda que incompleta, é denominado, neste trabalho, nível operacional. Para este nível há um mecanismo de reificação (concretização) bem determinado (Figura 1).

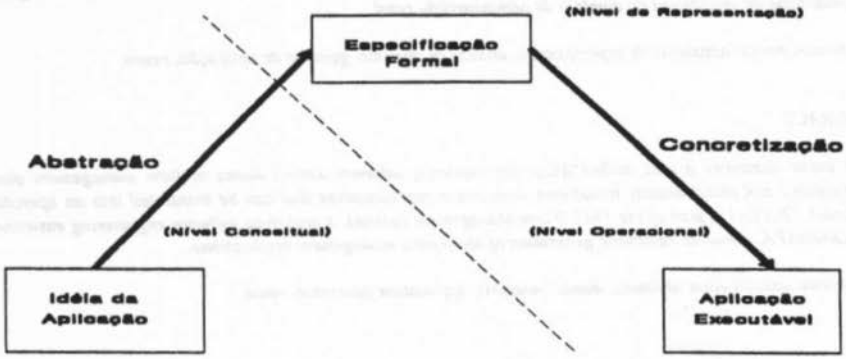

Figura 1. Modelo do FMS sob o modelo PW

2.1) Nível Conceitual

O processo de captura dos requisitos do usuário (informal) e sua tradução para uma representação formal é um problema em engenharia de software [19]. Os trabalhos de Balzer [3] sobre este tópico mostram que o processamento de especificações em linguagem natural é um problema em aberto. Os problemas que ocorrem na etapa de especificação de requisitos a tornam uma das etapas mais críticas do processo de desenvolvimento de software [16].

Segundo Aslett [2], a comunicação entre o usuário e o desenvolvedor de sistemas é um fator limitante nesta etapa e que compromete a confiabilidade do software produzido. O usuário é

um especialista do domínio e não conhece a tecnologia e a terminologia utilizada pelo desenvolvedor de sistemas e vice-versa.

Além disso, as etapas do ciclo de vida do software (conversão de requisitos em especificação, especificação em implementação) são mal documentadas, fazendo com que as informações que estão por trás de cada passo não estejam disponíveis nas etapas seguintes. Estes problemas de comunicação e documentação tornam a especificação de requisitos errônea, incompleta e ambígua.

Assistentes de Especificação são ferramentas para auxiliar nesta etapa de captura das especificações. Para apoiar o processo de abstração, essencial na captura de requisitos do usuário e posterior transformação para o nível de representação, o FMS prevê a existência do AEsp - Assistente de Especificação, de forma semelhante a outros sistemas [1,2,4,8,9,20,23].

O AEsp suporta o processo de entrevista e documentação dos requisitos dos usuários , sua decomposição sistematizada sob uma metodologia formal e a coleta do jargão específico através da instanciação de componentes pré-existentes. O resultado de uma sessão sob o AEsp, quando bem sucedida, é a aplicação do usuário e sua Linguagem da Aplicação(LA) subjacente [12].

2.2) Nível Operacional

O nível operacional incorpora o processo de transformação dos requisitos do usuário para um programa executável. Este nível foi reduzido ao processo de tradução entre os níveis da aplicação, representação e implementação. A Figura 2 mostra essa hierarquização.

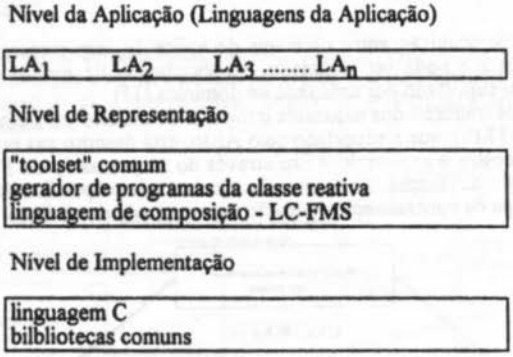

Figura 2. Hierarquização do nível operacional

2.2.1) Nível da Aplicação

Usando a terminologia de Leite [12], uma aplicação, neste nível, é descrita em uma LA. Segundo Leite, as linguagens da aplicação (LAs) são linguagens que têm todas as características das linguagens do domínio, só que tratam de uma ou duas instâncias daquele domínio. As LAs, no modelo do FMS, são expressas em LC-FMS. Esta metáfora suporta tanto:

a) versões diferentes da mesma aplicação, o que representa o seu desenvolvimento incremental;

b) aplicações diferentes para problemas semelhantes, que é amparado pelo modelo de Linguagens de Domínio [17].

2.2.2) Nível de Representação

O nível de representação é descrito pela sintaxe e semântica da LC-FMS, que é composta por várias sublinguagens: linguagem de especificação de formulários (telas), linguagem de especificação de transições entre formulários, linguagem de especificação de base de dados, linguagem de especificação de ações (operações relacionais, transformações e relatórios), linguagem de especificação de consistências e helps. Através dessas linguagens, a aplicação incorpora os vários componentes do domínio:

a) componentes de alta especificidade que aparecem como operandos e operadores dessas linguagens;
b) componentes de alto índice de reutilização, que são "templates" ou "clichés" [23] cuja expansão sintática materializa funções específicas da aplicação em desenvolvimento.

2.2.3) Nível de Implementação

Do nível de representação, a aplicação descrita em LC-FMS é traduzida automaticamente para uma implementação em linguagem C através de um gerador de código fonte (GFMS - Gerador do FMS), que usa as técnicas convencionais de compilação e ferramentas de geração de código [15].

O processo de tradução entre os níveis de aplicação, representação e implementação é conhecido na literatura e pode ser modelado, na abordagem de análise de domínio, como um processo de tradução suportado por uma rede de domínios [17].

O processo de tradução dos requisitos informais do usuário para uma representação formal (LA descrita em LC-FMS), que é suportado pelo AEsp, está descrito nas próximas seções.

A Figura 3 mostra o escopo do AEsp através do mapeamento do FMS sob o modelo PW que permite separar as etapas de abstração (coleta de especificação, incluindo reuso e prototipação) da etapa de concretização (geração automatizada do código).

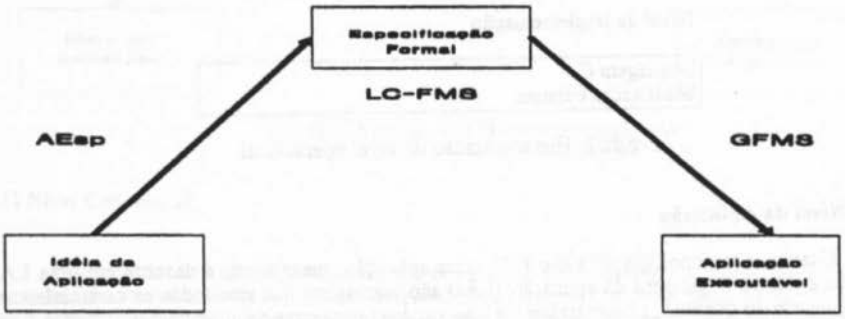

Figura 3. O escopo do AEsp sob o modelo PW

3) CARACTERÍSTICAS DO AEsp

O enfoque utilizado, pelo AEsp, para síntese da aplicação descrita em LC-FMS é:

a) considerar o usuário como o especialista que fornece a informação sobre a aplicação;
b) elicitar incrementalmente, através de uma entrevista, os conceitos da aplicação sob uma metodologia de decomposição;
c) não tentar automatizar a etapa de especificação, isto é, evitar o processamento automático da linguagem natural, mas utilizar um Engenheiro de Especificação para conduzir a entrevista e reduzir o vocabulário envolvido;
d) oferecer ao usuário uma infraestrutura para entrevista similar a aplicação final;
e) catalogar as abstrações de dados e ações de forma a possibilitar seu reuso em outras aplicações e, portanto, potencializar a transformação de uma LA em LD.

Para suportar este processo, o AEsp foi construído com as seguintes características:

a) estrutura de controle de entrevista com o usuário, permitindo capturar as especificações das aplicações;
b) estrutura de prototipação para apoiar na entrevista e validar a especificação a qualquer momento;
c) esquema de catalogação de componentes (operandos e operadores) de modo a incrementar permanentemente o acervo do domínio. A transformação de componentes específicos em componentes instanciáveis é feita "off-line";
d) esquema de reuso das informações do domínio, através de "reuse daemons" (seção 4.3);
e) apresentação da decomposição da aplicação, sob várias formas, para facilitar o "backtracking".

O protótipo do AEsp suporta essas funcionalidades através de um conjunto de ferramentas integradas, cuja conceituação está apresentada na Figura 4. Estas ferramentas são: Entrevistador, Prototipador, Catalogador de Componentes, Montador e Reutilizador. A descrição dessas ferramentas está na seção 4.

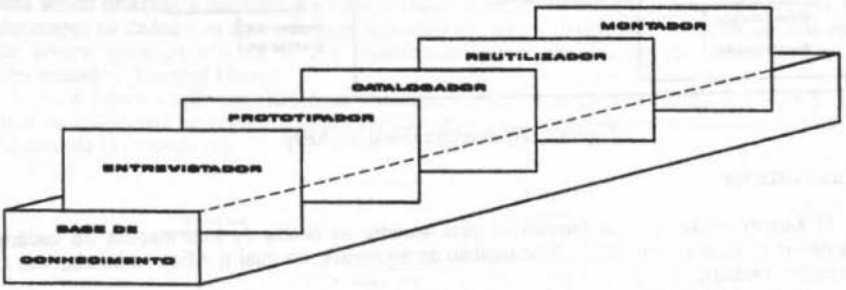

Figura 4. Ferramentas do AEsp

3.1) O Engenheiro de Especificação

O **Engenheiro de Especificação** é um técnico treinado a operar o AEsp com o objetivo de conduzir a entrevista para extrair a informação do Especialista do Domínio (usuário). Suas atividades são:

a) auxiliar no mapeamento da linguagem natural para o vocabulário controlado do ambiente AEsp;
b) oferecer o protótipo da aplicação durante a entrevista;
c) apoiar a modelagem dos dados e decomposição das ações;
d) apoiar o "backtrack" quando necessário.

4) ARQUITETURA DO AEsp

Nesta seção, está descrito detalhadamente cada ferramenta que compõe a arquitetura do AEsp (Figura 5).

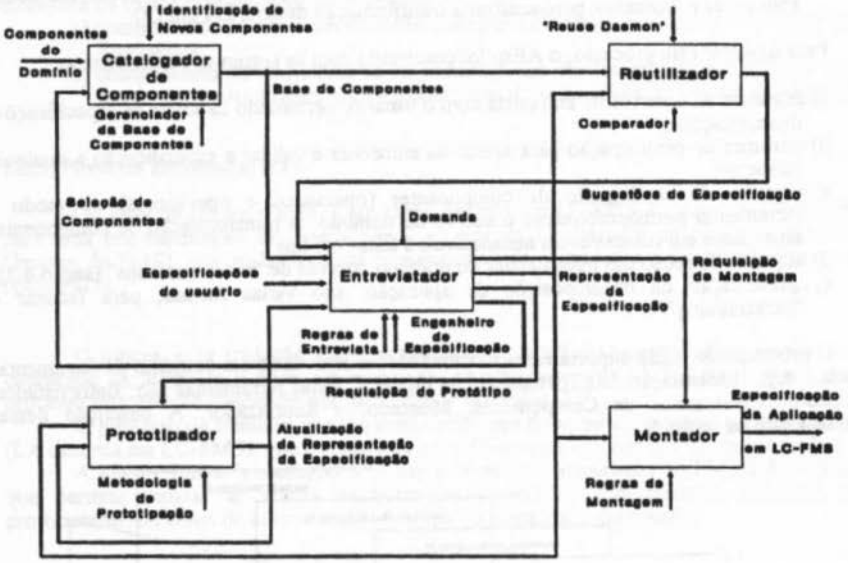

Figura 5. Arquitetura Geral do AEsp

4.1) Entrevistador

O **Entrevistador** é uma ferramenta para auxiliar na coleta de informações do usuário através de um modelo de entrevista. Este modelo de entrevista, no qual o AEsp se baseia, tem as seguintes propriedades:
a) roteiro de entrevista para refinamento do problema;
b) decomposição do problema segundo uma metodologia;
c) aplicação de regras da metodologia para garantir sua corretude;
d) invocação dos "reuse daemons";
e) mudanças de estratégia na decomposição quando necessário;
f) suporte a visão global do processo de decomposição.
O entrevistador executa várias funções para suportar este modelo de entrevista, como descrito na Figura 6.

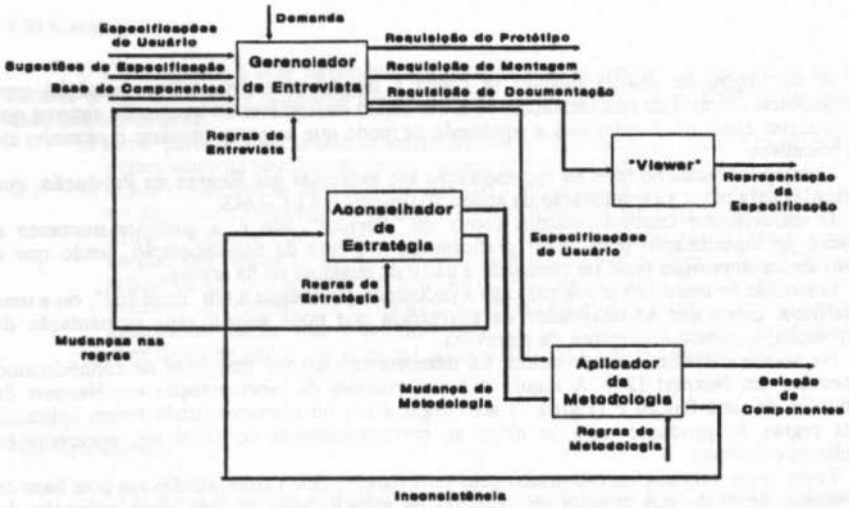

Figura 6. Funções do Entrevistador

O entrevistador funciona como um **Gerenciador da Entrevista**, que especifica as necessidades do especialista do domínio seguindo um roteiro de entrevista. O roteiro de entrevista foi definido a partir de uma análise local das aplicações do domínio. Este roteiro é baseado em perguntas direcionadas ao usuário e permite especificar o problema segundo uma metodologia de decomposição.

A estrutura do entrevistador depende da metodologia de decomposição e, neste trabalho, está sendo utilizada a metodologia HOS ("High-Order Software")[14]. Esta metodologia permite decompor os dados e as ações de uma aplicação em uma especificação na forma de uma estrutura de árvore onde os nós da árvore representam níveis hierárquicos da decomposição, e são denominados "Control Maps".

A especificação em HOS é representada, como é mostrado na Figura 7, por uma estrutura arbórea onde cada nó representa uma Função (F) e cada função tem um ou mais objetos (dados) de entrada (x) e saída (y).

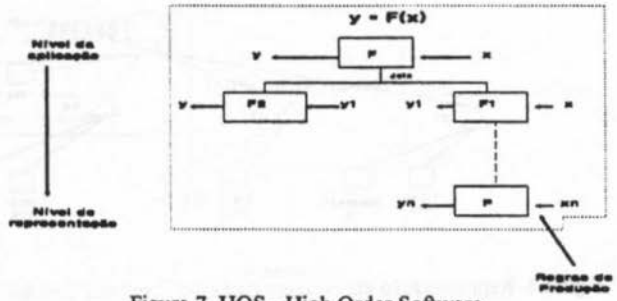

Figura 7. HOS - High Order Software

A decomposição finaliza quando se chega a um nó primitivo, para o qual há uma correspondência no nível de representação, ou a um objeto instanciável no acervo do sistema que faça o processo recorrer. A entrevista é conduzida de modo que se possa otimizar o caminho até os nós primitivos.

Correspondências no nível de representação são expressas por **Regras de Produção**, que são aplicadas para obter a especificação da aplicação descrita em LC-FMS.

O entrevistador também funciona como um **"Viewer"**, isto é, a qualquer momento o engenheiro de especificação pode exibir graficamente a árvore de decomposição, sendo que o processo de decomposição pode ser reiniciado a partir de qualquer nó da árvore.

O modelo de entrevista prevê que caso a decomposição chegue a um "dead end", ou a uma inconsistência, existe um **Aconselhador de Estratégia** que pode sugerir uma reorientação da decomposição, e, consequentemente, da entrevista.

Na versão corrente, o entrevistador foi desenvolvido usando uma base de conhecimento implementada em Nexpert [24]. A Figura 8 é um exemplo da representação em Nexpert da especificação de uma função F (Figura 7) até chegar a um nó primitivo, onde foram aplicadas algumas regras de produção para se obter as correspondências no nível de representação (Formulários e Campos).

Todas essas informações capturadas pelo entrevistador ficam armazenadas em uma base de conhecimento de modo que possam ser reusadas na especificação de uma nova aplicação do domínio.

Exemplo:

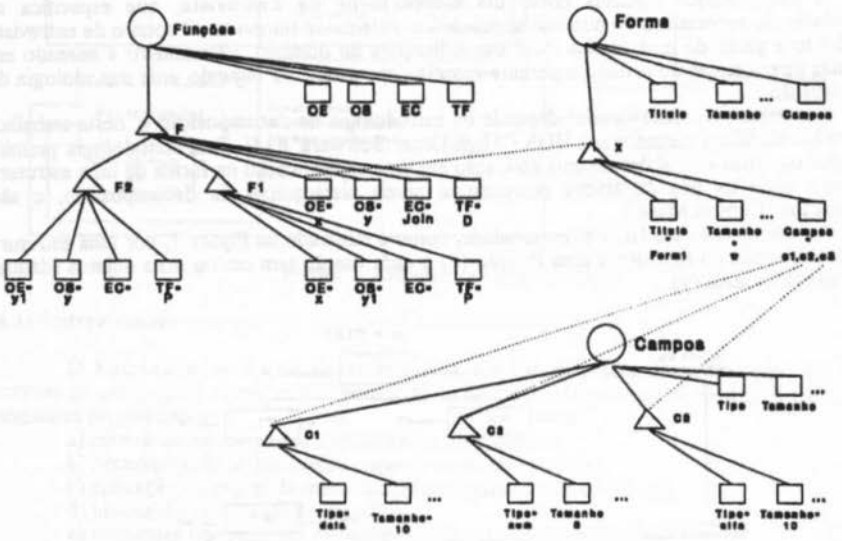

Figura 8. Representação em Nexpert da especificação da função F

4.2) Catalogador

O **Catalogador** é uma ferramenta para auxiliar na captura dos componentes do domínio durante a especificação e armazená-los na base de conhecimento. Esta ferramenta implementa duas funções básicas:

a) salvar parte de uma sessão da entrevista que o Engenheiro de Especificação perceba ser potencialmente reutilizável - normalmente são "Control Maps";

b) incluir componentes instanciáveis, que são a forma básica de reutilização de informações do domínio. Componentes instanciáveis são descritos em uma linguagem denominada CMO - "Control Maps Operators"[7]. A entrevista é, em última análise, o conjunto de perguntas e respostas que permite instanciar esses componentes para o "Control Map" específico da aplicação do usuário. As técnicas de reanálise do domínio que permitem examinar um conjunto de componentes específicos e transformá-los em um componente instanciável ainda não estão formalizadas. Componentes instanciáveis podem ser identificados tanto no nível da aplicação quanto no nível de representação.

4.3) Reutilizador

O AEsp é um assistente de especificação voltado para um domínio específico, permitindo a especificação de família de sistemas sob uma infraestrutura comum.

Sob esta visão, uma infraestrutura de reuso pode ser construída, de modo que a especificação de uma nova aplicação do domínio possa usar as informações já armazenadas no desenvolvimento de aplicações anteriores [5].

A base de conhecimento mantida pelo catalogador é continuamente varrida durante o transcorrer da entrevista pelo que se convencionou chamar de "reuse daemons". Estes processos são disparados sempre que:

a) o usuário fornece uma ação expressa por um verbo que, colocado na forma de infinitivo pelo engenheiro de especificação, fornece a chave de busca na base de componentes;

b) o usuário fornece um objeto da ação que conste em uma das classes existentes.

Nestas ocasiões, o **Reutilizador**, mostrado na Figura 9, dispara a instanciação dos componentes envolvidos (a maior parte das vezes pela troca dos identificadores formais) e oferece as possíveis sugestões de especificações previamente armazenadas.

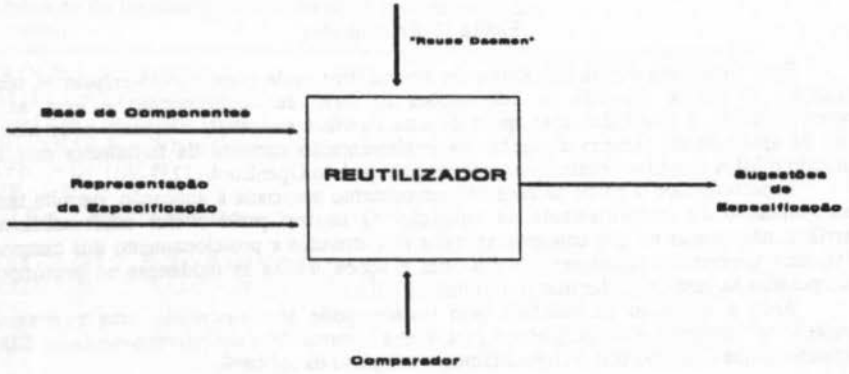

Figura 9. Reutilizador

Conjectura-se que esta abordagem de reuso é aderente ao domínio agropecuário baseado nas seguintes premissas:

a) propriedades com a mesma atividade de produção têm necessidade de informações muito semelhantes, mas não iguais;
b) processos gerenciais são particulares do proprietário e devem ser incorporados na aplicação.

4.4) Prototipador

O **Prototipador** é uma ferramenta para auxiliar na especificação da aplicação permitindo a validação incremental da aplicação durante a entrevista.

Um sistema de apoio à captura das especificações, enquanto diminui o trabalho de anotações das informações do usuário não elimina a ambiguidade que existe na produção de software e o entendimento errado pelo engenheiro de especificação das necessidades do usuário. Prototipação [10,13,21] é uma técnica que está sendo adotada para que, em qualquer momento da entrevista, o usuário tenha uma visão do que já foi capturado. Desta forma, o especialista do domínio e o engenheiro de especificação juntos podem rever e validar a especificação do sistema.

O prototipador do FMS, mostrado na Figura 10, tem como funções básicas:

a) suportar a metodologia de prototipação incremental;
b) "tunning" da interface com o usuário;
c) interface com o montador.

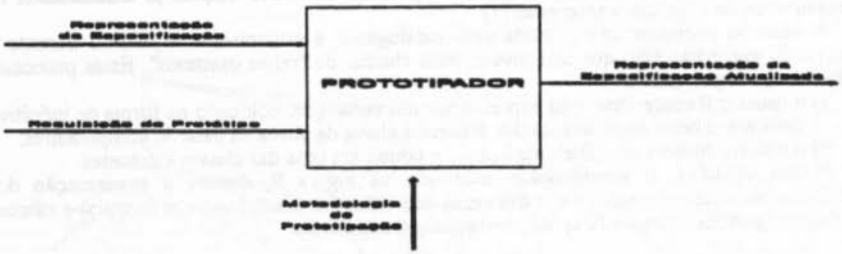

Figura 10. Prototipador

Esta ferramenta possui um Editor de Formulários onde pode ser desenhado as telas da aplicação. O Editor recupera as informações da base de conhecimento e gera as telas correspondentes. A partir daí, com apoio de uma interface amigável, o usuário pode alterar as telas da aplicação da maneira desejada. Na implementação corrente da ferramenta está sendo utilizado o SILK (gerador de interfaces gráficas para padrão OpenLook) [25].

O prototipador, a partir da base de conhecimento associada à aplicação, permite também uma simulação do comportamento da aplicação. O usuário pode alterar substancialmente a interface, não apenas no que concerne ao valor dos atributos e posicionamento dos campos das telas, mas também no encadeamento das telas e ações. Todas as mudanças no protótipo são incorporadas na base de conhecimento original.

Após a validação da interface pelo usuário pode ser requisitado uma montagem da aplicação ao Montador para completar o protótipo e transmiti-lo a um microcomputador IBM PC compatível, que pode mostrar o comportamento completo da aplicação.

4.5) Montador

O **Montador**, a partir da varredura da árvore gerada pelo entrevistador e base de conhecimento correlata, gera o programa descrito em LC-FMS que será a entrada para o GFMS gerar a aplicação do usuário, como é mostrado na Figura 11.

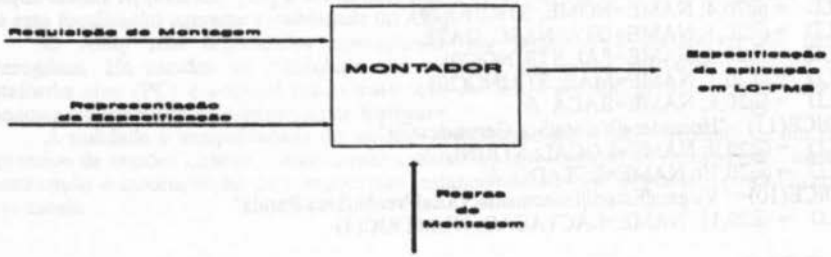

Figura 11. Montador

A forma LC-FMS, textual na atual implementação, permite tradução automática para outras especificações híbridas ("resources" do Windows ou "widgets" do XView).

Como um exemplo da LC-FMS, abaixo é dada a especificação de uma tela de um Sistema de Controle de Rebanho Leiteiro (SISCOREB) que está sendo desenvolvido para validar os conceitos do FMS.

```
%%EXT_FORM
ATTRIBUTE = Edit
CONTROL = BUTTON

%%FORM
!=============================================
! REBANHO.CUF
!=============================================
! Definicao do formulario para a edicao da base de dados do
! rebanho
!=============================================
COLOUR     = White /Blue
INPUT      = Yellow/Cyan
TCOLOUR    = Yellow/Blue
EDITCOLOUR = White/Cyan
TYPE   = DOUBLE
SIZE   = 80,25
TITLE = "Cadastro do Rebanho"

TEXT   = @3,2;  "NRO BRINCO (ID):"
FIELD  = @20,2; NAME=BRINCO; STRING(5)
TEXT   = @3,3;  "TIPO.....:"
TEXT   = @3,4;  "NOME.....:"
TEXT   = @3,5;  "DATA NASC.....:"
TEXT   = @3,6;  "PAI.....:"
TEXT   = @3,7;  "MAE.....:"
```

```
TEXT  = @3,8;  "RACA.....:"
TEXT  = @3,9;  "LOCAL.....:"
TEXT  = @3,10; "ESTADO.....:"
TEXT  = @3,11; "LACTACAO.....:"
FIELD = @20,3; NAME=TIPO; CHOICE(7) = "Novilha|Vaca|Touro"; &
VALUE=2
FIELD  = @20,4; NAME=NOME; STRING(20)
FIELD  = @20,5; NAME=DTA_NASC; DATE
FIELD  = @20,6; NAME=PAI; STRING(20)
FIELD  = @20,7; NAME=MAE; STRING(20)
FIELD  = @20,8; NAME=RACA; &
CHOICE(13) ="Holandesa|Nelore|Sta-Gertrudes|Gir"
FIELD  = @20,9; NAME=LOCAL; STRING(2)
FIELD  = @20,10; NAME=ESTADO; &
CHOICE(10)= "Virgem|Esteril|Inseminada|Vazia|Prenha|Seca|Parida"
FIELD  = @20,11; NAME=LACTACAO; NUMERIC(4)

%%SCHEMA
#FILE REBANHO
#KEY BRINCO
```

5) PROTÓTIPO DO AEsp

A implementação do AEsp exigiu a integração de geradores de interfaces, bases de conhecimento, gerenciadores de bases de objetos, técnicas de análise de domínio e reuso. A plataforma de software escolhida para implementar o AEsp foi:

a) Nexpert: gerenciador de bases de objetos e máquina de inferência para implementar as regras da metodologia e disparo dos "reuse daemons";
b) Silk: gerador de interfaces gráficas para padrão OpenLook com recursos de "link dinâmico".

A plataforma de hardware para desenvolvimento do AEsp, na versão corrente, é uma estação de trabalho SUN. Entretanto, a aplicação final é gerada para o ambiente PC que é a plataforma padrão para os usuários do domínio em questão.

6) ESTÁGIO DE DESENVOLVIMENTO

O Entrevistador do AEsp está com a sua arquitetura razoalvemente estabilizada. A implementação em Nexpert já inclui um avaliador genérico de "Control Maps" e sua instanciação. O Prototipador e Montador estão implementados e geram, respectivamente, o protótipo da interface da aplicação e parte substancial das especificações em LC-FMS. O reutilizador não está implementado e o catalogador dispõe apenas parte de sua funcionalidade.

Os compiladores das sublinguagens da LC-FMS estão operacionais, exceto a sublinguagem de eventos e relacional.

O ambiente FMS (AEsp, GFMS e ferramentas auxiliares) também está sendo usado para a definição e geração de um sistema de Controle de Rebanho Leiteiro (SISCOREB).

O componente "Controlar", essencial para essa aplicação, está bem definido e estudado, e já há uma descrição desse objeto com a correspondente linguagem de operadores para sua catalogação. O SISCOREB já incorpora o conceitos de eventos e decomposição de estados suportados pela metodologia de entrevista usada no AEsp. A liberação da versão alfa do sistema está prevista para o ano de 1994.

7) CONCLUSÃO

As características do AEsp, como implementado, são compatíveis com os outros projetos desse tipo [16]. O escopo é mais limitado que ASPIS [2], KBRA [4], KBRET [8] e "Requirements Apprentice" [23] e cobre uma classe de aplicações mais simples, porém, supõe-se que essa focalização aumente a viabilidade do AEsp.

O AEsp tem dificuldades operacionais por exigir uma plataforma de hardware heterogênea. Há estudos de viabilidade que mostram que a sua migração completa para a plataforma alvo (PC) é possível pela substituição do SILK por um gerador de interfaces para Windows e a execução do Nexpert neste hardware.

A utilidade e exequibilidade do modelo preconizado pelo AEsp depende de um número expressivo de sessões quando o acervo realmente puder corresponder às demandas do usuário. A identificação e incorporação de componentes instanciáveis também depende dessa utilização em larga escala.

8) BIBLIOGRAFIA

[1] ARIAS, C. **Um assistente especialista para especificação de requisitos**, Campinas: UNICAMP/IMECC, 1992. (Tese de Mestrado)

[2] ASLETT, M. J. **A knowledge based approach to software development**: ESPRIT Project ASPIS. Amsterdam: North Holland, 1991.

[3] BALZER, R.; GOLDMAN, N.; WILE, D. Informality in Program Specifications. In: RICH, C.; WATERS R. C. **Readings in artificial intelligence and software engineering**. Los Altos, CA: Morgan Kaufmannk, 1986. p. 223-232.

[4] CZUCHRY, A. J.; HARRIS D. R. KBRA: A new paradigm for requirements engineering, **IEEE Expert**, v. 3, n. 4, p. 21-35, winter, 1988.

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

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

[7] FERRARETTO, M.D. **Metodologia para desenvolvimento rápido de aplicações - MEDRA**. Campinas/SP: EMBRAPA-CNPTIA, 1994. (Documento interno)

[8] GOMAA, H.; KERSCHBERG L.; SUGURUMAN V. A knowledge-based approach to generating target system specifications from domain model. In: **IFIP Congress**, Madrid, Spain, 1992.

[9] GREEN, C.; LUCKMAN,D.; BALZER, R.; CHEATHAM, T.; RICH, C. Report on a Knowledge-Based Software Assistant. In: RICH, C.;WATERS R. C. **Readings in artificial intelligence and software engineering**. Los Altos, CA: Morgan Kaufmannk, 1986. p. 525-535.

[10] JORDAN, P.W.; KELLER, K.; TUCKER; VOGEL, D. Software storming - combining rapid prototyping and knowledge engineering. **IEEE Computer**, v.22, n.5, p. 39-48, may 1989.

[11] LEHMAN, M.M. A futher model of coherent programming processes. **Software Process Workshop**, p. 27-35, 1984.

[12] LEITE, J.C.S.P. O uso de hipertexto na elicitação de linguagens de aplicação. Simpósio Brasileiro de Engenharia de Software, 4, Águas de São Pedro, 24-26 de outubro de 1990. **Anais**. São Paulo: USP/CCS, 1990. p. 134-149.

[13] LUQI. Software evolution through rapid prototyping. **IEEE Computer**, v.22, n.5, p. 13-25, may 1989.

[14] MARTIN, J. **System design from provably correct constructs**. Englewood Cliffs: Prentice-Hall, 1985.
[15] 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.
[16] MASSRUHÁ, S.M.F.S. **Um estudo sobre assistentes de especificação**. Campinas/SP: EMBRAPA-CNPTIA, 1994. (Relatório Técnico, submetido a publicação)
[17] NEIGHBORS, J.M. The Draco approach to constructing software from reusable components. In: RICH, C.; WATERS R. C. **Readings in artificial intelligence and software engineering**. Los Altos, CA: Morgan Kaufmannk, 1986. p.525-535.
[18] NEIGHBORS, J.M. **Draco: a method for engineering reusable software systems, software reusability, concepts and models**. 1989 (ACM Press Frontier Series, 1)
[19] PRESSMAN, R. **Software engineering: a practitioner's approach**. 3.ed. New York: McGraw-Hill, 1992.
[20] PUNCELLO, P.; TORRIGIANI, P.; PIETRI, F.; BURLON, R.; CARDILE, B.; CONTI, M. ASPIS: A knowledge-based CASE environment. **IEEE Software**, v.5, n.2, p. 58-65, mar. 1988.
[21] TANIK, M.M.; YEH, R.T. Guest editor's introduction: Rapid prototyping in software development. **IEEE Computer**, v. 22, n.5, p.9-12, may 1989.
[22] WASSERMAN, A. I. Extending state transition diagrams for the specification of human-computer interaction. **IEEE Transactions on Software Engineering**, v. 11, n. 8, ago. 1985.
[23] WATERS, R. The requirements apprentice: automated assistance for requirements acquisition. **IEEE Transactions on Software Engineering**, v. 17 n. 3, p. 226-240, mar. 1991.
[24] **NEXPERT OBJECT; version 2.0; introduction manual**. Palo Alto/CA: Neuron Data, 1991.
[25] **SILK - Encap: user's manual**, release 1.0. Austin/TX: ISSI, 1993.