ISSN 1414-4727

UM ESTUDO SOBRE ASSISTENTES DE ESPECIFICAÇÃO

Sílvia Maria Fonseca Silveira Massruhá

ISSN 1414-4727

UM ESTUDO SOBRE ASSISTENTES DE ESPECIFICAÇÃO

Sílvia Maria Fonseca Silveira Massruhá

REPÚBLICA FEDERATIVA DO BRASIL

Presidente
Fernando Henrique Cardoso

Ministério da Agricultura e do Abastecimento

Ministro
Arlindo Porto Neto

Empresa Brasileira de Pesquisa Agropecuária - Embrapa

Presidente
Alberto Duque Portugal

Diretores
Elza Angela Battaggia Brito da Cunha
Dante Daniel Giacomelli Scolari
José Roberto Rodrigues Peres

Centro Nacional de Pesquisa Tecnológica em Informática para a Agricultura - CNPTIA

Chefe Geral
Moacir Pedroso Júnior

Chefe Adjunto de Pesquisa e Desenvolvimento
João Camargo Neto

Chefe Adjunto Administrativo
Leila Maria Lenk

ISSN 1414-4727

Empresa Brasileira de Pesquisa Agropecuária - Embrapa
Centro Nacional de Pesquisa Tecnológica em Informática para a Agricultura - CNPTIA
Ministério da Agricultura e do Abastecimento - MA

UM ESTUDO SOBRE ASSISTENTES DE ESPECIFICAÇÃO

Sílvia Maria Fonseca Silveira Massruhá

Campinas, SP
1997

EMBRAPA-CNPTIA. Relatório Técnico, 2.

Exemplares desta publicação podem ser solicitados à:
Empresa Brasileira de Pesquisa Agropecuária - EMBRAPA
Centro Nacional de Pesquisa Tecnológica em Informática para a Agricultura - CNPTIA
Av. Dr. André Tosello s/nº.
Cidade Universitária "Zeferino Vaz" - Barão Geraldo
Caixa Postal 6041
13083-970 - Campinas, SP
Telefone: (019) 239-9800
Fax: (019) 239-9594
E-mail: cnptia@cnptia.embrapa.br
URL: http://www.cnptia.embrapa.br

Editoração: Ivanilde Dispato

Tiragem: 100 exemplares

MASSRUHÁ, S.M.F.S. *Um estudo sobre assistentes de especificação.* Campinas: EMBRAPA-CNPTIA, 1997. 27p. (EMBRAPA-CNPTIA. Relatório Técnico, 2).

Assistentes de especificação; análise de domínio; Ferramentas CASE, Reusabilidade, Prototipação; Specification assistant; Domain analysis; Reusability; Prototype.

SUMÁRIO

Resumo..... 1
Abstract..... 1
1. Introdução..... 2
2. Assistentes de Especificação..... 5
2.1 KRET..... 5
2.2 KBRA..... 7
2.3. ASPIS..... 9
2.4. Requirements Apprentice (RA)..... 11
2.5. Assistente Especialista para Especificação de Requisitos (AR)..... 13
2.6. AEsp..... 14
3. Comparação entre Assistentes..... 18
4. Conclusão..... 23
5. Referências Bibliográficas..... 24

LISTA DE FIGURAS

FIG. 1. Escopo do AEsp sob modelo PW..... 15
FIG. 2. Arquitetura geral do AEsp..... 16
FIG. 3. Paradigma de desenvolvimento automatizado de software..... 18
FIG. 4. Mapeamento das etapas do ciclo de vida no paradigma de Desenvolvimento automatizado de software..... 19

UM ESTUDO SOBRE ASSISTENTES DE ESPECIFICAÇÃO[1]

Silvia Maria Fonseca Silveira Massruhá[2]

RESUMO - O AEsp é uma ferramenta para auxiliar a captura das especificações informais das aplicações do domínio de Administração Rural e transformá-las, incrementalmente, em uma representação que pode ser traduzida de forma automática para um programa executável. Esta ferramenta está sendo desenvolvida como parte do ambiente FMS ("Farm Management System"). O FMS é um ambiente para a geração automatizada de aplicativos no domínio de Administração Rural. O objetivo deste trabalho é fazer uma revisão bibliográfica dos assistentes de especificação existentes para auxiliar na especificação e dimensionamento do AEsp. Com este intuito foram descritos um conjunto de assistentes e um detalhamento do AEsp. Na seqüência, um conjunto de atributos, que serviram como crivo de comparação, foi proposto. Estes critérios foram aplicados sistematicamente a todos os assistentes.

Palavras-chave: assistentes de especificação; análise de domínio; ferramentas CASE; reusabilidade; prototipação.

AN OVERVIEW OF SPECIFICATION ASSISTANTS

ABSTRACT - The AEsp is a tool for capturing informal specifications of the Farm Management domain aplications and incrementally transform them into a representation that can be automatically translated into an operational program. This program is a component of the FMS (Farm Management System), a prototype software engineering environment for farm management systems. This report compares and evaluates the specification assistants that are described in the literature in order to help AEsp's design and dimensioning. Aiming at this, a set of assistants is described and the AEsp is thoroughly described. Following in this, a set of attributes is proposed to support a comparison sieve. These criterias were systematically applied to all assistants.

Key words: specification assistant; domain analysis; CASE tools; reusability; prototype.

---

[1] Analisado pelo Comitê de Publicações da Embrapa-CNPTIA em novembro de 1996.

[2] Mestre em Engenharia de Software, Embrapa-CNPTIA, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP.

UM ESTUDO SOBRE ASSISTENTES DE ESPECIFICAÇÃO

1. Introdução

Existem, na literatura, propostas de ferramentas de software, denominadas genericamente CASE, com objetivo de automatizar o processo de produção de software. Algumas destas ferramentas combinam métodos e técnicas, de engenharia de software e inteligência artificial, em uma infra-estrutura comum para suportar as atividades de produção de software visando melhorar sua qualidade e produtividade (Gomaa et al., 1992a; Karimi & Konsynski, 1988; Premkumar et al., 1991; Pressman, 1992).

Uma das etapas mais importantes e problemáticas do ciclo de vida do software para ser automatizada é a etapa de especificação dos requisitos (Borgida et al., 1986; Greenspan & Mylopoulos, 1984). Erros ocorridos nesta etapa, não detectados, tornam-se extremamente caros para serem corrigidos nas etapas posteriores do ciclo de vida (Boehm, 1981).

Segundo Aslett (1991), a comunicação entre o usuário e o desenvolvedor de sistema é um fator limitante na etapa de especificação de requisitos que compromete a confiabilidade do software produzido. O usuário é um especialista do domínio da aplicação e não conhece a tecnologia e a terminologia utilizada pelo desenvolvedor do sistema e vice-versa.

Além disso, as etapas do ciclo de vida (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 ambígüa.

Waters (1991) fala sobre a transformação da especificação informal de requisitos em especificação formal e aponta seis tipos de informalidades típicas na comunicação entre o usuário e o analista: abreviação, ambigüidade, ordenação, contradição, incompleteza e inconfiabilidade.

A transformação dos requisitos dos usuários em especificações formais tem sido considerada um problema em aberto. O trabalho de Balzer et al. (1986) sobre este tópico mostra que processamento de especificações escritas em linguagem natural está longe de ser resolvido.

Assistentes de especificação são ferramentas que fazem parte de ambientes de software automatizados (CASE) para auxiliar na etapa de captura das especificações.

Uma das abordagens para estas ferramentas utiliza técnicas de elicitação de requisitos (entrevistas) (Gause & Weinberg, 1989; Linster, 1988; Zucconi, 1989), técnicas para decomposição de problemas (DFD, MER, SADT) (Pressman, 1992) e estratégias de reuso (Biggerstaf & Richter, 1986; Prieto Díaz & Arango, 1991; Neighbors, 1986, 1989) para minimizar as informalidades típicas na comunicação entre usuário e analista.

Estes sistemas, enquanto eliminam o trabalho de anotações das informações do usuário, não eliminam a ambigüidade na comunicação do usuário com o analista. A inexistência de uma teoria formal para atacar este problema tem levado muitos ambientes de software a adotarem a metodologia de prototipação (Jordan et al. 1989; Luqi, 1988, 1989; Tanik & Yeh, 1989) como uma alternativa, pois através desta técnica o usuário e o analista podem, juntos, desenvolver e validar a especificação do sistema.

O modelo destes ambientes de software interativos tem as seguintes características (Pressman, 1992):

- possibilitar a um analista criar interativamente uma especificação baseada em uma linguagem;

- invocar ferramentas automatizadas que traduzem especificações baseadas nesta linguagem para sistemas executáveis; e

- permitir que o usuário use protótipos para refinar requisitos anotados.

O modelo de geradores de aplicação, para algumas classes de aplicações, baseia-se neste processo de desenvolvimento de software, onde é possível obter-se um sistema executável (aplicativo) gerado a partir de uma especificação refinada ao longo do processo de interação.

A abordagem de geradores de aplicação é bastante conhecida na literatura (Cleaveland, 1988; Masiero & Meira, 1993). Associados a estes geradores, estão as linguagens que são usadas para exprimir as aplicações e essas linguagens normalmente são conhecidas como linguagens de quarta geração - 4GL (Martin, 1985a). O processo de tradução dessas linguagens pode ser diretamente para código executável ou para uma linguagem alvo, para o qual já existe um compilador disponível ("Source to source translation") (Aho et al. 1986).

Nos trabalhos de Prieto Diaz (1990) e de Leite & Franco (1990), sobre Linguagens de Domínio e Linguagens de Aplicação, existem propostas de como sintetizar linguagens desse tipo, porém com escopo mais limitado.

A especificação de uma aplicação nessas linguagens necessita de uma tradução do nível informal (nível do usuário) para um nível formal (nível da linguagem da aplicação). Assistentes de Especificação são ferramentas de apoio a esta atividade.

Alguns assistentes de especificação operam sob a perspectiva de um domínio de aplicações, permitindo o desenvolvimento de família de sistemas com uma infra-estrutura comum.

Um processo, denominado análise de domínio, é usado para capturar as características comuns e as variações entre uma família de sistemas de um domínio para definir essa infra-estrutura comum (Arango, 1989; Prieto Díaz, 1990; Freeman, 1987).

Segundo Prieto Díaz (1990), a análise de domínio visa encontrar técnicas para capturar, estruturar e armazenar informações de um domínio de aplicações de modo que se possa reusá-las na especificação de novas aplicações do domínio.

Assistentes de especificação que se utilizam destes conceitos e oferecem uma infra-estrutura de reuso são usados para desenvolver várias aplicações do domínio. Estes assistentes permitem que a especificação de uma nova aplicação do domínio possa usar as informações já armazenadas no desenvolvimento de aplicações anteriores (Prieto Díaz et. al. 1991). Sob esta ótica, assistentes de especificação tornam-se ferramentas de apoio à atividade de análise de domínio.

Nos próximos itens deste trabalho, alguns assistentes citados na literatura serão descritos e comparados para auxiliar na especificação do Assistente de Especificação do FMS - AEsp, considerando-se os seguintes critérios:

- os assistentes, aqui tratados, são aqueles que auxiliam nas primeiras etapas do ciclo de vida da produção do software (assistente de requisitos, assistente de especificação, assistente de projeto);

- modelo de ciclo de vida do software adequado a estas ferramentas;

- os métodos, técnicas e ferramentas utilizadas por assistentes para a formalização das etapas do ciclo de vida do software;

- as ferramentas de software que compõem a arquitetura de um assistente;

- a plataforma de suporte para auxiliar na implementação de um assistente;
- as metodologias utilizadas no desenvolvimento de um assistente, tais como análise de domínio, reuso, componentização e prototipação.

2. Assistentes de Especificação

2.1. KBRET

a) Descrição

O KBRET (Knowledge-Based Requirements Elicitation Tool) é uma ferramenta baseada em conhecimento que visa gerar especificações do sistema alvo a partir da modelagem do domínio. A ferramenta KBRET é parte do protótipo de um ambiente, desenvolvido pela George Mason University, para demonstrar os conceitos de modelagem de domínio e reuso. Este ambiente, independente de um domínio de aplicações, é usado para suportar modelos de domínios e gerar a especificação do sistema alvo a partir destes modelos (Gomaa, 1992a, 1992b).

b) Modelo de Ciclo de Vida

O modelo de ciclo de vida proposto para este ambiente é ciclo de vida baseado em reuso, denominado ciclo de vida evolucionário e orientado a domínio. Este ciclo de vida é altamente interativo e opera sob a perspectiva de um domínio de aplicações permitindo o desenvolvimento de família de sistemas.

c) Metodologia

O processo de gerar especificações do sistema alvo utilizado no KBRET consiste em coletar os requisitos em termos das características do domínio, recuperar os componentes correspondentes do modelo do domínio para suportar aquelas características e garantir a consistência entre elas.

d) Arquitetura Geral

A arquitetura do KBRET possui duas fontes de conhecimento:

Conhecimento Independente do Domínio: fornece conhecimento de controle para as várias funções suportadas pelo KBRET. Estas funções incluem ferramentas tais como:

- "Dialog manager": é responsável por conduzir o diálogo com o engenheiro do sistema alvo e elicitar os requisitos para o sistema alvo.

- "Domain browser": fornece regras para consultar as fontes de conhecimento do domínio para o engenheiro de sistema poder modelar o sistema alvo.

- "Feature and object selection/deletion": mantém a forma de seleção e remoção das características para o sistema alvo e os tipos de objetos correspondentes.

- "Dependency checker": consiste as inter e intra-dependências dos objetos.

- "Target system generator": responsável por montar o sistema alvo depois que as especificações do sistema estiverem completas.

Conhecimento Dependente do Domínio: representa as visões múltiplas de um modelo de domínio da aplicação. Ferramentas que fazem parte desta fonte são:

- "Features and object types knowledge source": contêm uma lista de todos os tipos de objetos e suas características incorporados ao modelo do domínio.

- "Inter-feature and feature-object dependencies knowledge source": captura os vários relacionamentos e dependências entre as características e entre os objetos.

- "Multiple views": contêm visões diferentes do modelo do domínio tais como hierarquia de agregação, generalização e especialização. Este conhecimento é derivado pelo "Domain Dependent Knowledge Base Extractor Tool", que estrutura o conhecimento como fatos em CLIPS, do Repositório de Objetos (onde ficam armazenados os tipos de objetos do domínio, suas características e seus relacionamentos para poder modelar um domínio).

e) Plataforma de Suporte

O KBRET foi implementado em CLIPS (C Language Integrated Production System) - Shell para desenvolvimento de sistemas especialistas desenvolvido pela NASA. Outro software utilizado no ambiente onde está inserido o KBRET é o STP (IDE's Software Through Pictures CASE) para representar as visões múltiplas do

modelo do domínio. O STP armazena os dados em uma base de dados relacional denominada TROLL.

f) Metodologias de Análise de Domínio

O KBRET pode ser caracterizado como uma ferramenta de apoio à atividade de análise de domínio, pois permite gerar especificações de aplicações a partir de modelos de domínios. O ambiente, no qual o KBRET está inserido, é independente de um domínio de aplicações, isto é, ele não é voltado para um domínio específico mas funciona na perspectiva de domínios de aplicações. A especificação e a implementação da infra-estrutura deste ambiente é construído de modo a permitir modelagem de domínios de aplicações e reuso das informações destes domínios.

2.2. KBRA

a) Descrição

O KBRA (Knowledge-Based Requirements Assistant) é uma ferramenta para auxiliar a etapa de análise de requisitos do ambiente KBSA (Knowledge-Based Software Assistant). O KBSA é proposta de um ambiente que tem como objetivo formalizar todo o processo de produção de software e fornecer suporte, baseado em conhecimento, para todos aspectos do ciclo de vida da produção de software (Green et al., 1986; Czuchry & Harris, 1988).

b) Modelo de Ciclo de Vida

O modelo de ciclo de vida do KBSA é baseado no paradigma de desenvolvimento incremental de software. Divide o ciclo de vida em três etapas principais: desenvolvimento, validação da especificação e gerenciamento de projeto. Segundo o modelo de ciclo de vida proposto neste ambiente, a atividade de evolução torna-se a atividade central no processo de desenvolvimento de software e adequa-se ao modelo de ciclo de vida evolucionário.

c) Metodologia

A base metodológica do KBSA é capturar e formalizar todo o processo de desenvolvimento de software (a identificação de requisitos, o projeto da especificação, a implementação daquela especificação, e sua manutenção). O KBRA é uma ferramenta que auxilia a etapa de captura de requisitos cobrindo três

fases principais: aquisição, análise e comunicação. Para tal, o KBRA apresenta as seguintes propriedades: uso do computador desde o início do projeto; capacidade de fornecer versões múltiplas do sistema que está sendo especificado; suporte para captura e auxílio em decisões; oportunidades para manusear reuso na etapa de requisitos; capacidade de consistir e completar alguns requisitos automaticamente.

d) Arquitetura Geral

O protótipo do KBRA auxilia a etapa de análise e aquisição de requisitos desde a fase inicial até a preparação do documento final. Para suportar esta funcionalidade, o KBRA possui uma biblioteca de componentes reusáveis e formas de apresentação das informações. As formas de apresentação suportadas pelo KBRA são:

- "Notepad" inteligente: um editor de texto estruturado para o usuário rascunhar as idéias iniciais do sistema.

- Diagrama de contexto: editor de diagramas utilizado para descrever a interface entre o sistema e o hardware, software e operadores do ambiente externo.

- Diagrama de estados: editor gráfico utilizado para descrever os estados do sistema e eventos que provocam uma mudança no estado.

- Planilha de cálculos: é um objeto que armazena a descrição da evolução do sistema mostrado na forma de planilha.

- Documento de requisitos: KBRA gera automaticamente um documento de requisitos para o sistema, na mesma maneira apresentada no "notepad".

A partir das informações obtidas por estas formas de apresentação, o KBRA propaga decisões para um repositório central de requisitos e verifica a consistência dos resultados.

e) Plataforma de Suporte

O KBRA utiliza técnicas de representação de conhecimento associadas com a etapa de análise e aquisição de requisitos: apresentação, texto estruturado e descrição do sistema envolvido. A plataforma de suporte do KBRA também é baseada em técnicas de inteligência artificial, tais como: herança de propriedades de tipos de objetos genéricos, classificação automática baseada nos

discriminadores que indicam como especializar instâncias, e propagação de restrições para processamento de ramificações de decisões de requisitos e para suporte a mudanças.

f) Metodologias de Análise de Domínio

O KBRA é uma ferramenta de software independente de um domínio de aplicações. O KBSA, do qual o KBRA faz parte, é um assistente especialista nas metodologias envolvidas nas etapas do ciclo de vida do software.

2.3. ASPIS

a) Descrição

O ASPIS (A Knowledge Based CASE Environment) é um ambiente que visa melhorar a qualidade e produtividade das primeiras fases do ciclo de vida do software (Análise de Requisitos e Projeto), combinando técnicas de Inteligência Artificial e Engenharia de Software (Aslett, 1991; Puncello et al., 1988).

b) Modelo de Ciclo de Vida

Modelo de ciclo de vida evolucionário é adotado pelo ASPIS para fazer a ligação entre as fases de Análise e Projeto. O ciclo de vida do ASPIS suporta a interação entre o projetista e o analista com um processo chamado "Síntese". Neste processo a informação é coletada da fase de análise e colocada de forma disponível na fase de projeto, permitindo a comunicação entre analista e projetista. É um processo contínuo onde as inconsistências são checadas e, então, modificadas para atender as necessidades dos usuários.

c) Metodologia

O ASPIS tem como objetivo melhorar a qualidade do software suavizando a transição entre as necessidades dos usuários, análise e projeto através de um conjunto de ferramentas integradas. Na fase de Análise de Requisitos é utilizada SAL (Structural Analysis Language) para representar funções e esquemas entidade-relacionamento para representar dados. O RSL (Reasoning Support Logic) é utilizado para otimizar os diagramas SA e para checar a consistência e a corretude das especificações. O RSL permite especificar propriedades de um sistema com um conjunto de axiomas em uma linguagem baseada em lógica. Usando o RSL, o analista pode escrever axiomas em termos de funções, predicados e posições em um diagrama SA. Estes axiomas são transformados em

código PROLOG para serem executados. A fase de Projeto é dividida em duas etapas:

- projeto do sistema: descreve o sistema nos termos globais de hardware e software; e
- projeto do software: descreve as funções do software em termos de processos e transições.

d) Arquitetura Geral

O ASPIS possui um conjunto de ferramentas baseadas em conhecimento, denominadas assistentes, e a definição de um formalismo baseado em lógica para especificações. O ASPIS possui quatro assistentes: assistente de análise, de projeto, de protótipo e de reuso.

- Assistente de análise e assistente de projeto são caracterizados como assistentes baseados em conhecimento, pois incorporam o conhecimento sobre o método e sobre o domínio da aplicação. Eles são utilizados por desenvolvedores de uma aplicação particular vinculada a uma metodologia particular. O assistente de análise possui um conjunto de ferramentas que permite ao usuário e ao sistema gerenciar documentos de análise.
- Assistente de protótipo e assistente de reuso são caracterizados como assistentes de suporte, pois o primeiro apóia a verificação das propriedades do sistema e o segundo apóia os desenvolvedores a reusar especificações e projetos.

e) Plataforma de Suporte

Estes assistentes do ASPIS foram implementados utilizando a linguagem PROLOG.

f) Metodologias de Análise de Domínio

O ASPIS é um ambiente independente de um domínio de aplicações, mas ele é utilizado na perspectiva de domínios de aplicações permitindo reusar informações de um domínio específico na especificação de novas aplicações deste domínio. O ASPIS incorpora conhecimento sobre o método e sobre os domínios de aplicações particulares.

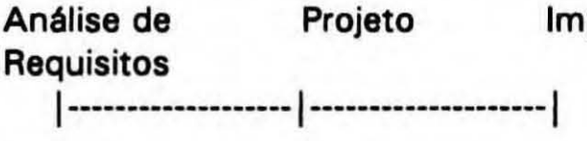
2.4. Requirements Apprentice (RA)

a) Descrição

O Requirements Apprentice (RA) é um assistente automatizado para aquisição de requisitos. Ele ajuda o analista na criação e modificação dos requisitos de software. Focaliza a transição entre especificações informais e formais. O RA está sendo desenvolvido no contexto do projeto Programmer's Apprentice, cujo objetivo é a criação de um assistente inteligente para todos os aspectos do desenvolvimento de software (Waters, 1991).

b) Modelo de Ciclo de Vida

O modelo de ciclo de vida adotado no RA é o modelo evolucionário que se ajusta bem à abordagem de desenvolvimento incremental utilizada neste tipo de assistente. O ciclo de vida utilizado é divido em três partes principais:



A ferramenta RA é usada na etapa de Análise de Requisitos.

c) Metodologia

O Requirements Apprentice atua na fase de aquisição de requisitos. Esta fase pode ser divida em três etapas:

- elicitação: que tem como produto final requisitos informais;
- formalização: etapa que constrói a especificação formal a partir dos requisitos informais; e
- validação: etapa para validar a especificação formal.

O RA atua mais especificamente na etapa de formalização que faz a ponte entre a especificação informal e a especificação formal. O alvo do RA são seis tipos de informalidades típicas da comunicação entre o usuário e o analista: abreviação, ambigüidade, ordenação, contradição, incompleteza e inconfiabilidade.

O RA tem três tipos de saídas usadas para interagir com o analista:

- saída interativa: notifica o analista de conclusões e inconsistências detectadas enquanto a informação de requisitos está sendo fornecida;

- uma base de conhecimento de requisitos; e

- documentos contendo o resumo da base de conhecimento de requisitos.

d) Arquitetura Geral

A arquitetura do RA é composta por três módulos:

- CAKE: é um sistema para inferência e representação do conhecimento, que suporta as habilidades de inferência do RA.

- EXECUTIVE: este sistema manuseia a interação com o analista e fornece controle em alto nível do inferência executado pelo CAKE.

- Biblioteca de "Clichés": é um repositório declarativo de informações relevantes para requisitos gerais e requisitos de domínios particulares. Esta biblioteca possibilita inferir informação a partir dos enunciados do analista.

A interface do RA é dividida em três janelas:

- janela principal: é usada para mostrar informações sobre os requisitos;

- janela de diálogo: é usada para digitar os comandos para o RA e para mostrar imediatamente as respostas correspondentes; e

- janela de saídas pendentes: é usada para mostrar uma lista de ações pendentes que necessitam ser resolvidas.

e) Plataforma de Suporte

O Requirements Apprentice (RA) foi implementado em Commom LISP. O diálogo entre o analista e o RA é feito através de expressões descritas em LISP.

f) Metodologias de Análise de Domínio

O RA pode ser utilizado como uma ferramenta de suporte a atividade de Análise de Domínio. Este não é voltado para um domínio específico mas ele permite

desenvolver sistemas sob a perspectiva de domínios de aplicações. O RA permite reusar informações de domínios particulares através de uma estrutura de clichés.

2.5. Assistente Especialista para Especificação de Requisitos (AR)

a) Descrição

O AR, sigla utilizada para fins de simplificação, é um assistente especialista para especificação de requisitos. Este assistente possibilita a especificação dos requisitos através de respostas dadas pelo usuário, em linguagem natural, a um questionário proposto pelo assistente. A partir destas respostas um modelo semi-formal, baseado no conhecimento sobre o modelo entidade-relacionamento (MER) é gerado (Arias, 1992).

b) Modelo de Ciclo de Vida

O modelo de ciclo de vida adotado neste trabalho é baseado no paradigma de programação automática proposto por Balzer. O conceito utilizado é que a partir dos conceitos informais obtém-se incrementalmente os requisitos do sistema até conseguir uma especificação formal, de onde pode-se derivar um protótipo sobre o qual é feita a validação.

c) Metodologia

Este assistente é classificável como um sistema especialista em metodologias, cuja abordagem consiste em observar como os peritos em metodologias atuam na aplicação das mesmas. O AR exibe as características básicas de um assistente especialista em metodologias:

- são ferramentas baseadas em conhecimento;
- servem para comparar ou mesmo compelir a utilização da metodologia; e
- proporcionam uma interface amigável com o usuário.

A metodologia é expressa através de regras que definem seus princípios e regras que restrigem práticas contrárias ao enfoque dado.

Este assistente de requisitos utiliza o MER como método de especificação. O MER é gerado a partir de um questionário que é aplicado ao usuário.

d) Arquitetura Geral

A arquitetura deste assistente é particionada em:

- metodologia (neste caso é o MER);
- base de dados: onde é armazenado o conhecimento obtido através do diálogo com o usuário, e de onde se obterá, posteriormente, o modelo textual;
- máquina de inferência: é a parte do assistente que consulta a base de conhecimentos para validar a especificação segundo o modelo entidade-relacionamento; e
- interface em linguagem natural: é um ambiente no qual o usuário fornece os conceitos informais da aplicação, através de um diálogo em um sub-conjunto da linguagem natural, guiado pelo assistente.

e) Plataforma de Suporte

Este assistente foi implementado utilizando o ambiente PROLOG.

f) Metodologias de Análise de Domínio

O AR se enquadra na classe de assistentes especialistas em metodologias. É uma ferramenta independente de um domínio de aplicações.

2.6. AEsp

a) Descrição

O AEsp é uma ferramenta para auxiliar na captura das especificações das aplicações do domínio de administração rural. Esta ferramenta está sendo desenvolvida no contexto do projeto FMS ("Farm Management Systems"). Este projeto visa construir um ambiente para a geração automatizada de aplicativos no domínio de administração rural. Este ambiente consiste de um conjunto de ferramentas integradas (Ferraretto, 1992, 1993, 1994).

b) Modelo de Ciclo de Vida

Modelo de ciclo de vida adotado é o modelo de ciclo evolucionário. O modelo básico usado é o PW (Lehman, 1984) que permite separar as etapas de abstração (coleta de especificação, incluindo reuso e prototipação) da etapa de reificação (geração automatizada do código). Este modelo foi adaptado para amparar tanto versões incrementais de uma mesma aplicação quanto aplicações semelhantes de um mesmo domínio. A Fig. 1 mostra o escopo do AEsp sob o modelo PW.

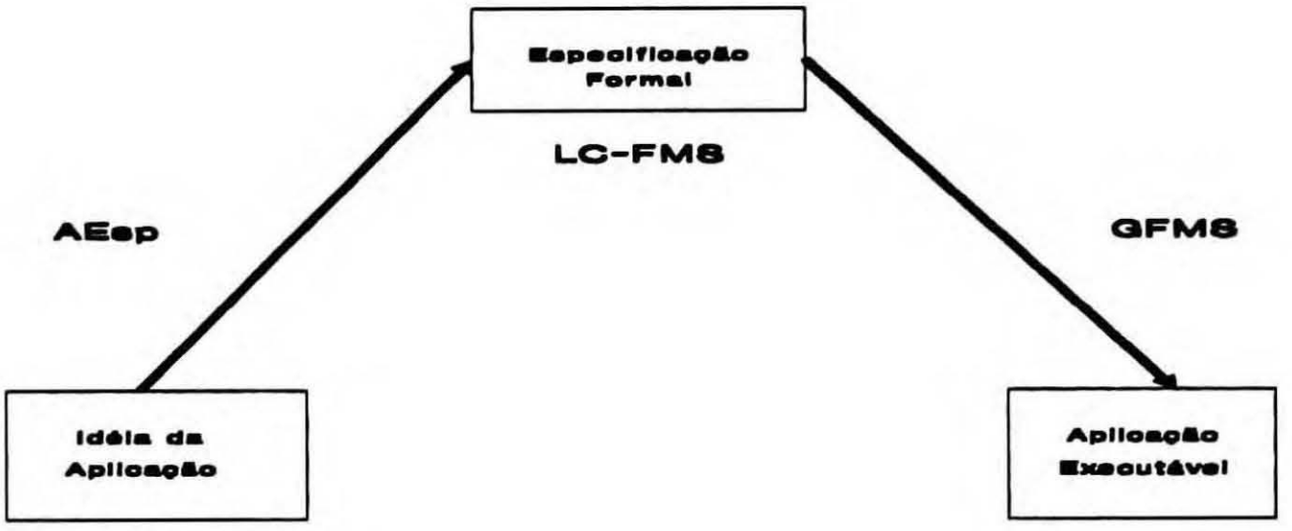


FIG. 1. Escopo do AEsp sob modelo PW.

c) Metodologia

A metodologia utilizada no FMS é a MEDRA (Ferraretto, 1994). Esta metodologia prevê que durante a fase de especificação do processo de desenvolvimento de software, 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. No nível de representação, uma aplicação do domínio agropecuário é vista como um conjunto de especificações expressas na Linguagem de Composição do FMS - LC-FMS através de suas sublinguagens. Para suportar esta abordagem o FMS é baseado no modelo de níveis para os quais há um mecanismo de tradução convencional. Este modelo propõe uma separação para os níveis do processo de transformação dos requisitos do usuário: nível conceitual, nível não formalizado e o nível operacional, onde já há uma descrição formal da aplicação. O AEsp auxilia o processo de tradução dos requisitos informais do usuário para uma representação formal (nível conceitual). Para suportar este processo, o AEsp foi construído com as seguintes características:

- estrutura de controle de entrevista com o usuário, permitindo capturar as especificações das aplicações;
- estrutura de prototipação para auxiliar na entrevista e validar a especificação;
- esquema de catalogação de componentes (operandos e operadores) de modo a incrementar permanentemente o acervo do domínio;
- esquema de reuso das informações do domínio; e
- apresentação da decomposição da aplicação, sob várias formas, para facilitar "Backtracking".

d) Arquitetura Geral

O AEsp suportará a etapa de elicitação das aplicações através de um conjunto de ferramentas integradas. As ferramentas que compõem o AEsp (Fig. 2) são: Entrevistador, Prototipador, Catalogador, Reutilizador e Montador.

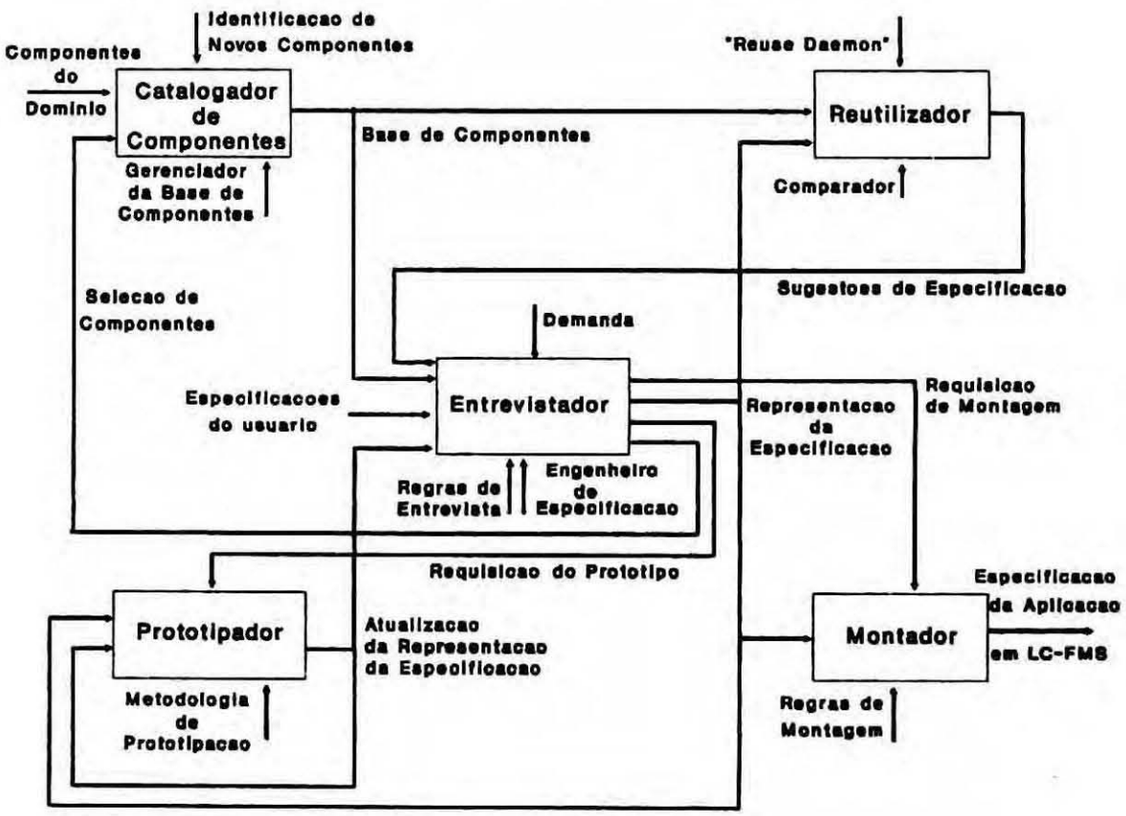


FIG. 2. Arquitetura geral do AEsp.

O ENTREVISTADOR é uma ferramenta que possibilita a decomposição do problema segundo uma metodologia (Martin, 1985b; Hamilton, 1976a, 1976b), obtendo os componentes da aplicação que está sendo especificada a partir das especificações do usuário. Estes componentes são capturados segundo um modelo de entrevista e são armazenados em uma base de conhecimento através de uma ferramenta denominada CATALOGADOR DE COMPONENTES. Durante o processo de entrevista o entrevistador pode utilizar sugestões de especificação vindas do reutilizador.

O REUTILIZADOR é uma ferramenta que permite reusar componentes armazenados na base de conhecimento na especificação de uma nova aplicação do domínio. O modelo de domínio do FMS é compatível com este método de reuso.

O Engenheiro de Especificação pode oferecer o protótipo da aplicação em qualquer momento da entrevista enviando uma requisição ao prototipador.

O PROTOTIPADOR é uma ferramenta para auxiliar na validação da especificação durante a entrevista. Esta ferramenta gera uma primeira versão do protótipo incompleta mostrando a interface da aplicação. O protótipo completo da aplicação será obtido depois que o prototipador enviar uma mensagem de montagem ao Montador.

O MONTADOR gera a especificação da aplicação em LC-FMS a partir da varredura da árvore gerada pelo Entrevistador e bases de conhecimento correlatas. Essa especificação descrita em LC-FMS é automaticamente traduzível pelo GFMS.

e) Plataforma de Suporte

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

- Nexpert: é um ferramenta baseada em conhecimento com um gerenciador de objetos e classes com estrutura adequada para armazenar componentes do domínio, e tem interface com Linguagem C.
- Silk:gerador de interfaces gráficas para openwindows com recursos de "Link-edição dinâmica".

- Linguagem C.

A plataforma de hardware para desenvolvimento do AEsp é PC-DOS e UNIX. Os sistemas gerados são usados em PC.

f) Metodologias de Análise de Domínio

O AEsp pode ser caracterizado como uma ferramenta de apoio a atividade de análise de domínio. O AEsp é um assistente de especificação voltado para um domínio específico. O ambiente FMS, no qual o AEsp está inserido, é visto como uma infra-estrutura de reuso para aplicações no domínio de administração rural.

3. Comparação entre Assistentes

No item anterior pôde-se constatar que, em linhas gerais, os assistentes analisados são baseados no modelo de ciclo de vida evolucionário (Gomaa, 1992a, 1992b; Luqi, 1988, 1989) e suportam o desenvolvimento de software de forma automática como descrito na Fig. 3 (Pressman, 1992).

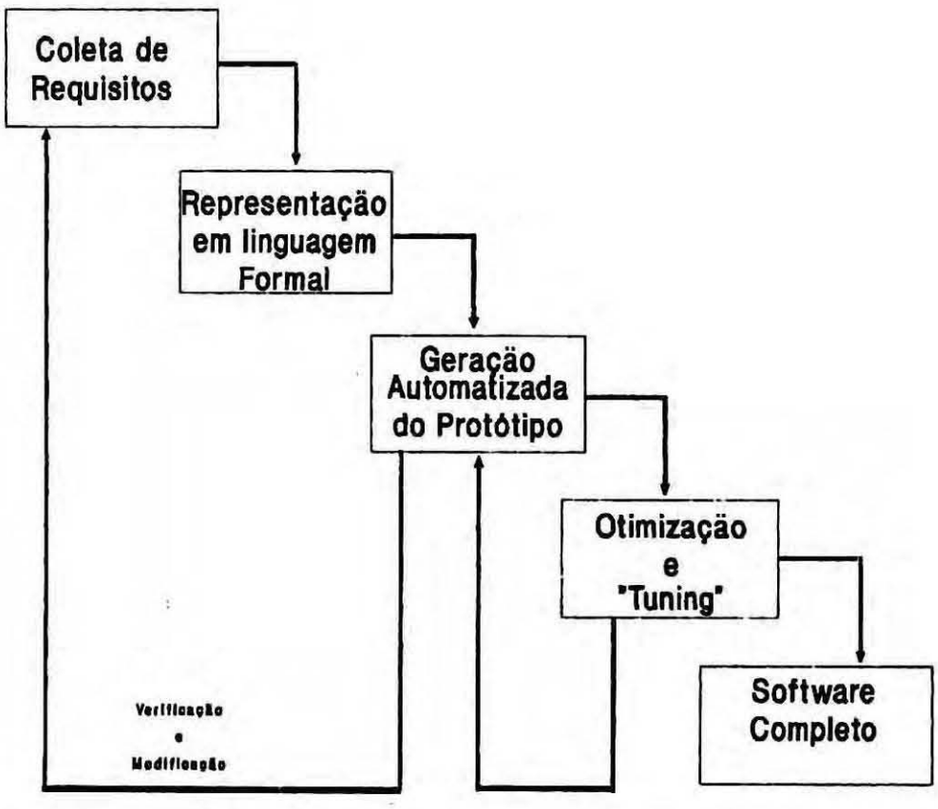


FIG. 3. Paradigma de desenvolvimento automatizado de software.

O modelo evolucionário prevê que a fase de desenvolvimento englobe a fase de evolução e manutenção, de modo que o desenvolvimento seja incremental e haja uma validação da especificação antes da geração do código. Desta forma, o modelo evolucionário se encaixa no paradigma de desenvolvimento de software automatizado.

Dividiu-se o ciclo de vida no qual se baseiam estes Assistentes de Especificação em 3 etapas principais: análise de requisitos/especificação, validação da especificação e desenvolvimento.

O mapeamento destas etapas do ciclo de vida no paradigma de desenvolvimento automatizado de software pode ser representado como na Fig. 4.

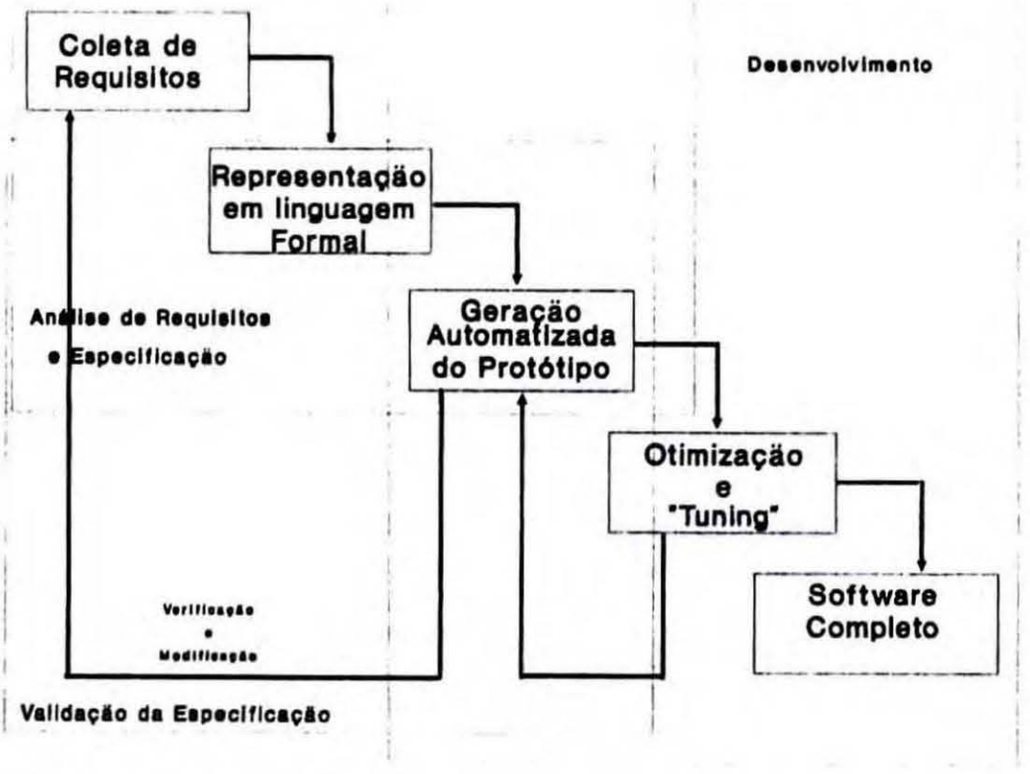


FIG. 4. Mapeamento das etapas do ciclo de vida no paradigma de desenvolvimento automatizado de software.

A análise de requisitos e especificação é a etapa responsável por capturar as informações do usuário, decompor e refinar o problema até se obter uma especificação.

A etapa de validação da especificação é responsável por transformar a especificação em um protótipo executável para que o usuário possa validá-la.

A etapa de desenvolvimento engloba a implementação e manutenção: a implementação sendo realizada como resultado de várias transformações das especificações e a manutenção sendo executada pela alteração da especificação.

Neste trabalho é mostrado que Assistentes de Especificação auxiliam na primeira e segunda etapas diretamente, e através da especificação auxiliam a fase de desenvolvimento, permitindo que ferramentas automatizadas traduzam as especificações baseadas em linguagem para sistema executável.

Além das atividades, descritas anteriormente, referentes às etapas do ciclo de vida, existem outras, tais como desempenho e gerenciamento, que não serão abordadas neste trabalho, embora alguns assistentes as suportem.

Neste trabalho, de modo a tornar mais objetiva a comparação entre assistentes, selecionou-se algumas características relativas às três etapas do ciclo de vida. Estas características foram citadas em (Green, 1986) e adaptadas para as necessidades deste trabalho.

a) Análise de Requisitos e Especificação

- Forma de aquisição das especificações: meio pelo qual o assistente se comunica com seu usuário alvo.

- Técnica de aquisição de conhecimento: técnicas utilizadas para extrair as informações do usuário.

- Linguagem de manipulação dos requisitos: são linguagens específicas suportadas pelo assistente para especificar os requisitos.

- Metodologia de refinamento e decomposição do problema: metodologias para refinar o problema até obter a especificação.

- Estratégia de reuso dos requisitos: um sistema de reuso que permita reutilizar especificações de aplicações anteriores.

- Domínio de aplicações: domínio específico suportado pelo assistente, se este for orientado à dominio.

- Classes de usuário: o usuário alvo do assistente.

b) Validação da Especificação

- Modelo de representação da especificação: modo pelo qual são representadas as especificações dos requisitos, formalmente ou informalmente.

- Linguagem de especificação: forma pela qual a especificação é representada - em uma linguagem que pode ser executável ou interpretável.

- Consistência da especificação: mecanismos e processos que permitam garantir a consistência interna da especificação.

- Teste da especificação: mecanismos e processos que permitam que a especificação possa ser testada para ser validada pelo usuário antes de gerar o código da aplicação.

- Parafraseador de especificação: capacidade para parafrasear uma especificação formal em linguagem natural com vocabulário restrito.

- Prototipação rápida: capacidade para automaticamente compilar uma especificação para um nível que permita gerar um protótipo.

- "Tracer" de requisitos: mecanismos e processos que permitam garantir o acompanhamento dos requisitos do usuário em qualquer ponto do ciclo de vida.

- Descritor de comportamento: mecanismos com capacidade de explicitar em linguagem natural o comportamento de uma especificação.

c) Desenvolvimento

- Modelo de implementação: o processo subjacente de desenvolvimento dos sistemas - incremental ou convencional.

- Linguagem de amplo espectro: existência de uma linguagem capaz de representar o projeto de um sistema em todos os estágios do ciclo de vida, da especificação formal até a implementação.

- Linguagem de transformações: se há disponibilidade de uma linguagem capaz de descrever as transformações da forma abstrata para concreta.

- Linguagem de propriedades: disponibilidade de uma linguagem capaz de descrever as propriedades de segmentos dos programas (variáveis, módulos, relações etc.).

- Documentação interativa: se há disponibilidade de um sistema capaz de documentar passos selecionados pelo usuário e as decisões que o levaram a tal.

- Prova automática de propriedades: mecanismo de inferência para automaticamente provar propriedades de partes do sistema em desenvolvimento.

- Desenvolvimento automatizado: disponibilidade de um sistema capaz de gerar a aplicação automaticamente a partir de uma especificação.

- Reuso automatizado: capacidade de adaptar um desenvolvimento prévio para uma especificação alterada.

Comparam-se, a seguir, os assistentes descritos no item 2, segundo as características específicas de cada etapa do ciclo de vida descritas anteriormente. O AEsp será inserido nos quadros comparativos para oferecer uma comparação com os outros assistentes citados.

a) Análise de requisitos e especificação

Assistentes/ Caractetísticas[3]	KBRET	KBRA	ASPIS	RA	AR	AEsp
Forma de aquisição das especificações	Diálogo	Diálogo	Diálogo	Diálogo	Diálogo	Diálogo
Técnicas de aquisição do conhecimento	Entrevista	Entrevista	Entrevista	Entrevista	Entrevista	Entrevista
Linguagem de manipulação de requisitos	Gráfica	Textual/ gráfica	Textual	Textual/ gráfica	Textual/ gráfica	Textual/ gráfica
Metodologia de refinamento do problema	Refinamento através de 5 visões múltiplas	Prevê uma metodologia	SAL	Estrurura de clichés	MER	HOS
Estratégia de reuso	Reuso dos objetos do repositório de objetos	Biblioteca de componentes reusáveis	Reuso de análise/ projeto	Reuso de clichés		Reuso das especifi-cações
Orientado a domínio	Perspectiva domínio	Independên-cia domínio	Perspectiva domínio	Perspectiva domínio	Independên-cia domínio	Domínio adminis-tração rural
Classes de usuário	Engenheiro requisitos do sistema	Usuário final	Desenvol-vedor da aplicação	Analista de sistemas	Engenheiro de sistemas	Engenheiro de especi-ficação

[3] As colunas preenchidas com (*) asteriscos, indicam que esses assistentes não possuem as características assinaladas ou que estas características não puderam ser identificadas nos artigos referenciados.

b) Validação da especificação

Assistentes/ Caractetísticas[4]	KBRET	KBRA	ASPIS	RA	AR	AEsp
Modelo de representação	Formal	Formal	Formal	Formal	Formal	Formal
Linguagem de especificação	Interpretável	Executável	Executável	Executável	Interpretável	Interpretável
Consistência da especificação	Disponível	Disponível	Disponível	Disponível	Disponível	Disponível
Teste de especificação	*	Protótipo executávek	Protótipo executável	Protótipo executável	Validação do MER	Protótipo incompleto
Parafraseador de especificação	*	Disponível	*	Disponível	*	*
Prototipação rápida	*	Disponível	Disponível	Disponível	*	Disponível
"Tracer" de requisitos	Disponível	Disponível	*	Disponível	*	*
Descritor de comportamento	*	Disponível	*	Disponível	*	*

c) Desenvolvimento

Assistentes/ Caractetísticas[4]	KBRET	KBRA	ASPIS	RA	AR	AEsp
Modelo de implementação	Incremental	Incremental	Incremental	Incremental	Incremental	Incremental
Linguagem de amplo aspectro	*	Disponível	*	Disponível	*	Disponível
Linguagem de transformações	*	Disponível	*	Disponível	*	Disponível
Linguagem de propriedades	*	Disponível	*	*	*	Disponível
Documentação interativa	Disponível	Disponível	Disponível	Disponível	Disponível	Disponível
Prova automática de propriedades	*	Disponível	*	Disponível	*	*
Desenvolvimento automatizado	*	Disponível	*	Disponível	*	Disponível
Reuso automatizado	Disponível	Disponível	Disponível	Disponível	*	Disponível

4. Conclusão

Os trabalhos citados não foram escritos tendo em mente os critérios de comparação aqui estabelecidos. A comparação aqui sugerida visa auxiliar no dimensionamento e especificação do AEsp.

É notável que os modelos de ciclo de vida para ambientes desta classe são muito parecidos. Isto é evidenciado pelas tabelas do item 3 quando se examina atributos ligados a esse modelo - forma de aquisição do conhecimento, orientação a domínio, modelo de representação, testes e linguagens usadas. Esta semelhança se deve, ao que parece, mais ao tipo de ferramentas e métodos usados (Análise de Domínio e Engenharia de Conhecimento) que às propriedades específicas do objeto alvo do sistema.

[4] As colunas preenchidas com (*) asteriscos, indicam que esses assistentes não possuem as características assinaladas ou que estas características não puderam ser identificadas nos artigos referenciados.

A homogeneidade da arquitetura dos assistentes é evidenciada pelas ferramentas que os compõe: ferramentas para elicitação de requisitos; especificação formal; decomposição de problemas; prototipação rápida e geração automática de código.

As plataformas de suporte utilizadas para implementação dos assistentes também são similares. Utilizam-se de técnicas de análise de sistemas (orientação a objetos), linguagens de quarta geração, processos de inteligência artificial e técnicas de engenharia de software.

Outro ponto de observação é que alguns destes assistentes são utilizados na perspectiva de domínios de aplicações, permitindo o desenvolvimento de família de sistemas com uma infra-estrutura comum. Estes assistentes tornam-se ferramentas de apoio à atividade de Análise de Domínio.

Independente da ambição dos assistentes anteriormente citados, é possível verificar que a proposta do AEsp, no espectro de suas aplicações, corresponde às características de um assistente.

5. Referências Bibliográficas

AHO, A.V.; SETHI, R.; ULLMAN, J.D. *Compilers*: principles, techniques, and tools. Reading: Addison Wesley, 1986. 796p.

ARANGO, G. Domain analysis - from art form to engineering discipline. *ACM SigSoft Software Engineering Notes*, v.14, n.3, May 1989.

ARIAS, C.I.S. *Um assistente especialista para especificação de requisitos.* Campinas: UNICAMP-IMECC, 1992. 115p. Tese Mestrado.

ASLETT, M.J., ed. *A knowledge based approach to software development*: ESPRIT Project ASPIS. Amsterdam: North Holland, 1991. 249p.

BIGGERSTAF, F.T.J.; RICHTER, C. *Reusability framework, assessment, and directions*. Austin: MCC, 1986. 18p. (MCC. Technical Report, STP-345-86).

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: Morgan Kaufmannk, 1986. p.223-232.

BOEHM, B.W. *Software engineering economics.* Englewood Cliffs: Prentice Hall, 1981. 767p.

BORGIDA, A.; GREENSPAN, S.; MYLOPOULOS, J. *Knowledge representation as the basis for requirements specifications*. In: RICH, C.; WATERS, R.C. Readings in artificial intelligence and software engineering. Los Altos: Morgan Kaufmannk, 1986. p.561-570.

CLEAVELAND, J.C. Building application generators. *IEEE Transactions on Software Engineering*, v.5, n.4, p.25-33, July, 1988.

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

FREEMAN, P.A. Conceptual analysis of the Draco approach to constructing software systems. *IEEE Transactions on Software Engineering*, v.13, n.7, p.830-844, July, 1987.

FERRARETTO, M.D. *Projeto FMS*. Campinas: EMBRAPA-CNPTIA, 1992. (Documento reservado).

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

FERRARETTO, M.D.; MASSRUHÁ, S.M.F.S. *Projeto ambiente de desenvolvimento de software para domínio de administração rural - FMS*. Campinas: EMBRAPA-CNPTIA, 1993. (Documento interno apresentado ao Sistema EMBRAPA de Planejamento - SEP)

GAUSE, D.C.; WEINBERG, G.M. *Exploring requirements:* quality before design. Dorset House, 1989.

GOMAA, H.; KERSCHBERG L.; BOSCH, C.; SUGURUMAN, V.; TAVAKOLI, I. *A prototype software engineering environment for domain modeling and reuse.* Fairfax: George Mason University, 1992a. (Technical report)

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

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: Morgan Kaufmannk, 1986. p.525-535.

GREENSPAN, S.L.; MYLOPOULOS, J.; BORGIDA, A. Capturing more world knowledge in the requirements specification. In: FREEMAN, P.; WASSERMAN, A.I. *Tutorial software design techniques*. 4.ed. Los Angeles: IEEE Computer Society, 1984. p.231-240.

HAMILTON, M.; ZELDIN, S. Higher order software - a methodology for defining software. *IEEE Transactions Software Engineering*, v.2, n.1, p.9-32, Mar. 1976a.

HAMILTON, M.; ZELDIN, S. *Integrated software development system/Higher Order Software conceptual description, version 1.* Fort Monmouth: ECOM, 1976b. (Research and Development Technical Report, ECOM-76-0329-F).

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.

KARIMI, J.; KONSYNSKI, B.R. An automated software design assistant. *IEEE Transactions on Software Engineering*, v.14, n.2, p.194-210, Feb. 1988.

LEHMAN, M.M. *A further model of coherent programming processes.* Software Process Workshop, p.27-35, 1984.

LEITE, J.C.S. do P.; FRANCO, A.P.M. O uso de hipertexto na elicitação de linguagens de aplicação. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 4., Águas de São Pedro, 1990. *Anais.* São Paulo: USP-CCS, 1990. p.134-149.

LINSTER, M. *A critical look at Kriton. In: AAAI Knowledge Acquisition for Knowledge-based Systems Workshop*, 3., 1988. Banff, Canadá. *Proceedings.*

LUQI. Knowledge-based support for rapid software prototyping. *IEEE Expert*, v.3, n.4, p.9-15, Winter, 1988.

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

MARTIN, J. *Fourth generation languages*. New Jersey: Prentice-Hall, 1985. v.1, p.319-335a.

MARTIN, J. *System design from provably correct constructs.* Englewood Cliffs: Prentice-Hall, 1985b.

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.

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: Morgan Kaufmannk, 1986. p.525-535.

NEIGHBORS, J.M. *Draco:* a method for engineering reusable software systems, software reusability, concepts and models. 1989. (ACM Press Frontier Series, 1)

PREMKUMAR D.; RONALD J.; BRACHMAN; SELFRIDGE, P.G.; BALLARD, B. Lassie: a knowledge-based software information system. *Communications of the ACM*, v.34, n.5, p.34-49, May, 1991.

PRESSMAN, R. *Software engineering:* a practitioner's approach. 3.ed. New York: McGraw-Hill, 1992. 793p.

PRIETO DÍAZ, R. Domain analysis: an introduction. *ACM Sigsoft Software Engineering Notes*, v.15 n.2, p.47-54, Apr. 1990.

PRIETO DÍAZ, R.; ARANGO, G. *Domain analysis and software systems modeling*. Los Alamitos: IEEE Computer Society, 1991. 299p.

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.

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.

WATERS, R. The requirements apprentice: automated assistance for requirements acquisition. *IEEE Transactions on Software Engineering*, v.17, n.3, p.226-240, Mar. 1991.

ZUCCONI, L. Techniques and experiences capturing requirements for several real-time aplications. *ACM SIGSOFT Software Engineering Notes*, v.14, n.6, p.51-55, Oct. 1989.

Empresa Brasileira de Pesquisa Agropecuária
Centro Nacional de Pesquisa Tecnológica em Informática para a Agricultura
Ministério da Agricultura e do Abastecimento
Av. Dr. André Tosello s/nº. - Cidade Universitária "Zeferino Vaz"
Barão Geraldo - Caixa Postal 6041
13083-970 - Campinas, SP
Telefone (019) 239-9800 - Fax (019) 239-9495
cnptia@cnptia.embrapa.br