UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ESTUDO COMPARATIVO E TAXONOMIA DE FERRAMENTAS DE SUPORTE À CONSTRUÇÃO AUTOMÁTICA DE SISTEMAS

por

HUBERT AHLERT

Exame de Qualificação III

Prof. Dr. Carlos Alberto Heuser
Prof. Dr. José Palazzo M. de Oliveira

Avaliadores

Porto Alegre, janeiro de 1991.

SUMÁRIO

LISTA DE FIGURAS 02
RESUMO 03
ABSTRACT 04
1. INTRODUÇÃO 05
2. O CICLO DE VIDA DO SOFTWARE E O PARADIGMA DE PROGRAMAÇÃO AUTOMÁTICA 08
2.1 Ciclo tradicional (cascata) 08
2.2 Prototipação 11
2.3 Programação automática 15
3. AS CARACTERÍSTICAS DE FERRAMENTAS DE SUPORTE À CONSTRUÇÃO AUTOMÁTICA DE SISTEMAS 21
3.1 O perfil da ferramenta fixando o grau de automação.. 21
3.2 Critérios de classificação de ferramentas..... 23
3.2.1 Critérios no âmbito da construção automática de sistemas 24
3.2.2 Critérios adicionais 34
4. ESTUDO DE CASO: COMPARAÇÃO DE FERRAMENTAS SEGUNDO OS CRITÉRIOS ESTABELECIDOS 42
4.1 LINC..... 42
4.1.1 Considerações preliminares 42
4.1.2 Características da ferramenta 46
4.2 SADS..... 57
4.2.1 Considerações preliminares 57
4.2.2 Características da ferramenta 62
4.3 Quadro sintético de comparação das características . 74
5. CONCLUSÕES..... 77
REFERÊNCIAS BIBLIOGRÁFICAS..... 79

LISTA DE FIGURAS

Fig. 2.1 - Ciclo de vida tradicional 09
Fig. 2.2 - Ciclo de vida no paradigma da prototipação .. 14
Fig. 2.3 - Ciclo de vida no paradigma da programação automática 19
Fig. 3.1 - O ciclo de vida e as intervenções humanas ... 22
Fig. 3.2 - Tabelas de consistência fixando a semântica dos programas genéricos de consistência 27
Fig. 4.1 - A construção de sistemas através do LINC 44
Fig. 4.2 - Tela de "menu" de atividades do LINC 45
Fig. 4.3 - A construção de sistemas através do SADS 60
Fig. 4.4 - Tela de "menu" de atividades do SADS 61
Fig. 4.5a- Quadro sintético de comparação (critérios no âmbito da construção automática de sistemas). 75
Fig. 4.5b- Quadro sintético de comparação (critérios adicionais)..... 76

RESUMO

Este trabalho apresenta um estudo comparativo das características de ferramentas de suporte à construção automática de sistemas, buscando estabelecer uma taxonomia com base em um conjunto de critérios de classificação, fundamentalmente naqueles critérios relacionados com os aspectos da construção automática.

Visando mostrar a influência que o uso de ferramentas tem sobre o processo de desenvolvimento de sistemas, esta monografia procura inicialmente apresentar uma restrospectiva sobre o ciclo de vida do software dentro dos paradigmas de construção de sistemas: tradicional, prototipação e programação automática.

Uma avaliação das ferramentas LINC e SADS foi elaborada segundo os critérios de classificação aqui estabelecidos.

ABSTRACT

This work presents a comparative study about the features of tools that support the automatic construction of systems, trying to establish a taxonomy based upon a set of classification criteria, essentially those related to the automatic construction aspects.

With the aim of exhibiting the influency that the use of tools has over the system development process, this monography initially looks for presenting a retrospective about the software lifecycle into the system construction paradigms: traditional, prototyping and automatic programming.

It was elaborated an appraisement of the LINC and SADS tools according to the classification criteria previously defined.

1. INTRODUÇÃO

No início da computação comercial o trabalho de um analista de sistemas era considerado uma tarefa essencialmente artesanal que dependia, em grande parte, da experiência e senso crítico do profissional responsável por modelar e desenvolver o novo sistema aplicativo.

Com a evolução do hardware e da correspondente mudança na proporção de custo hardware e software, o perfil do analista necessariamente teve que acompanhar as mudanças provocadas pelo avanço tecnológico. Um modelo de linha de produção, para aumentos de produtividade, tinha que ser introduzido também na construção de software. O custo de pessoal no desenvolvimento de sistemas representava uma fatia muito elevada no custo final do produto (software).

A partir deste quadro socio-tecnológico começam a surgir as investigações em metodologias que auxiliem os analista e programadores em suas tarefas rotineiras visando uma padronização e consequente aumento de produtividade. Os conceitos da engenharia de software começam a ser sedimentados e as práticas como estruturação, modularização e reusabilidade são exploradas por estas metodologias.

Passam a ser usados métodos e técnicas para melhor estruturar as atividades de análise e projeto sempre enfatizando a diagramação dos componentes do sistema para que o próprio usuário final possa dar uma contribuição mais efetiva na concepção deste sistema. Estas metodologias, no entanto, por si só, ainda exigiam um grande esforço em tarefas manuais de modelagem do sistema,atribuindo-lhes uma conotação de "metodologias muito trabalhosas" em vista dos inúmeros detalhes presentes numa especificação e das constantes validações de coerência e completeza necessárias para evitar eventuais erros de definição. Por que então não usar o computador para automatizar algumas tarefas de analistas e programadores?

As ferramentas de suporte às metodologias de análise e projeto aparecem, assim, como mais uma área de interesse dentro da engenharia de software.

Fases específicas do ciclo de vida de um sistema vão sendo supridas por metodologias e estas recebendo um suporte automatizado de ferramentas projetadas para atender a tarefas específicas.

Atualmente o elenco de ferramentas disponíveis no mercado já é significativo, principalmente nas fases de modelagem de sistemas onde aparecem as ferramentas tipo CASE como suporte gráfico para as tarefas de desenho do sistema e os geradores de aplicações e linguagens de quarta geração como um conjunto de ferramentas destinadas a atender à fase de desenvolvimento e manutenção dos sistemas.

Fala-se muito hoje em integração de ferramentas e em ambiente unificado de desenvolvimento de sistemas onde estudos procuram determinar como interligar as ferramentas das diversas fases do ciclo de vida e como padronizar o diálogo homem-máquina tornando o ambiente de construção dos sistemas mais homogêneo.

Outra linha de pensamento procura direcionar esforços na procura de um ambiente de programação automática onde, através de sucessivos refinamentos de especificação, se consiga obter, como produto, o código executável do programa, com a mínima intervenção humana.

Dentro do panorama das ferramentas de suporte à construção automática de sistemas, este trabalho visa determinar as características destas ferramentas com o intuito de estabelecer uma taxonomia segundo alguns critérios de classificação.

No capítulo 2 é feita uma retrospectiva sobre a evolução do processo de desenvolvimento de sistemas focalizando 3 grandes grupos de ciclos de vida: o tradicional, a prototipação e a programação automática.

No capítulo 3 são avaliadas as características de ferramentas de suporte à construção de sistemas e estabelecidos os critérios de classificação destas ferramentas.

Para validar os critérios, o capítulo 4 apresenta um estudo de caso comparando duas ferramentas, o LINC e o SADS, dentro destes critérios.

2. O CICLO DE VIDA DO SOFTWARE E O PARADIGMA DE PROGRAMAÇÃO AUTOMÁTICA

O processo de desenvolvimento de sistemas vem acompanhando os constantes avanços tecnológicos que estão sendo registrados na área de software. Metodologias novas e ferramentas de apoio para suporte a estas metodologias são propostas e, associadas a elas, surgem novos procedimentos operacionais (administrativos e computacionais) exigindo eventuais mudanças no ciclo de desenvolvimento do sistema. O ciclo de vida do software passa, portanto, por reformulações e evolui. Neste sentido, no presente capítulo, é apresentada a trajetória de evolução do processo de desenvolvimento de sistemas até atingir a proposta de programação automática onde se persegue o objetivo de mínima intervenção humana em todo o processo.

2.1 Ciclo tradicional (cascata)

O ciclo de vida do sistema pressupõe um conjunto de fases pelas quais este sistema passa desde sua concepção até a sua eventual desativação. No ciclo tradicional, o processo de desenvolvimento de um sistema é visto como uma sequência de tarefas em cascata (ciclo de vida em cascata). A medida que uma maior compreensão do sistema é obtida, validações são realizadas sobre cada fase e eventuais alterações são produzidas para que erros não sejam sedimentados de uma fase para outra (hipóteses geradas sobre erros no projeto). O ciclo de vida, portanto, é decomposto em fases ou subciclos e listas de verificação.

Para este ciclo tradicional, não existe um consenso entre os pesquisadores da área de engenharia de software em relação a uma padronização na composição das fases e no estabelecimento das fronteiras entre cada fase. Na literatura, Martin [MAR 83] e Boehm [BOE 88] apresentam pequenas variações na especificação das fases dentro do ciclo de vida. Em linhas gerais, a estrutura do modelo

clássico de desenvolvimento pode ser ilustrado como mostra a figura 2.1.

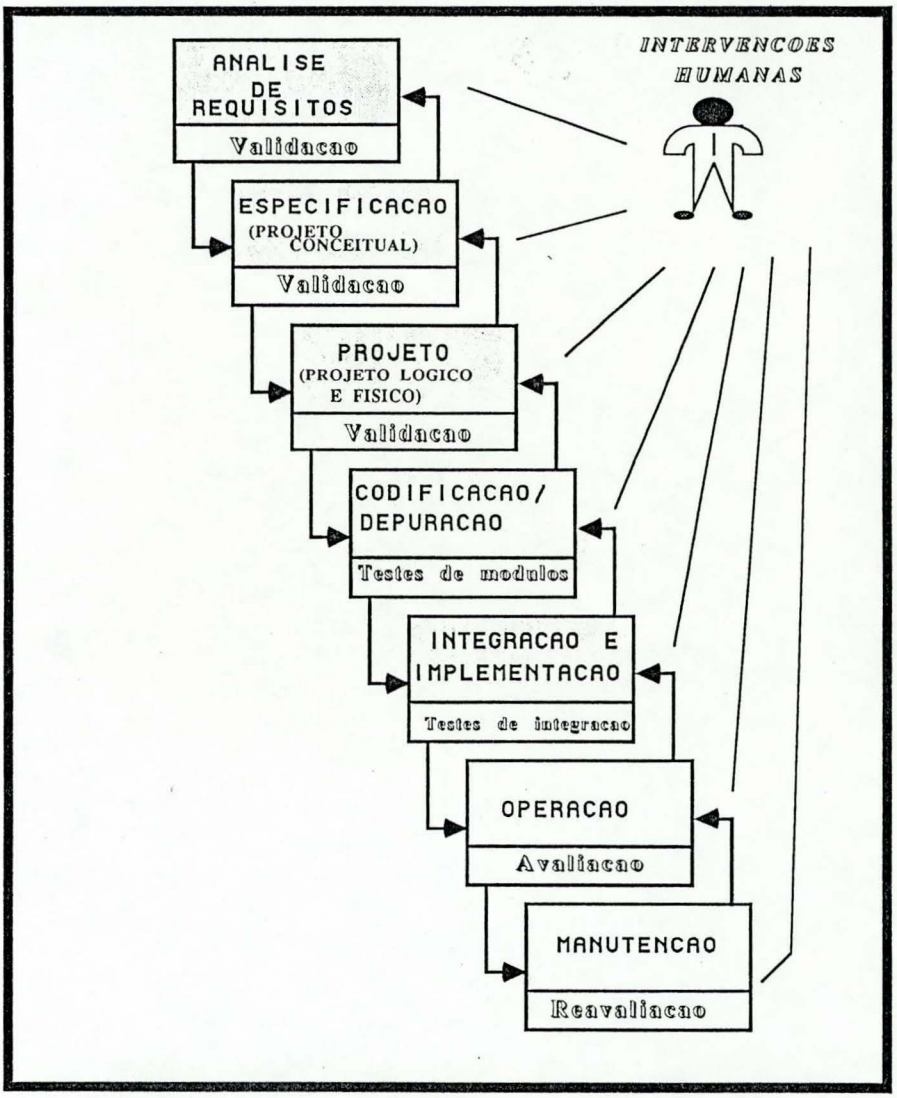


Fig. 2.1 - Ciclo de vida tradicional

Os defensores deste modelo de desenvolvimento salientam que sua utilização permite um maior planejamento do percurso a ser seguido pelo sistema durante todo o ciclo, proporcionando um guia e controle nas diferentes fases.

Gutierrez e De Souza [GUT 89] consideram que este modelo clássico peca no sentido de ignorar alguns aspectos

altamente relevantes na atual visão de desenvolvimento de sistemas. Por exemplo:

- a possibilidade cada vez maior de desenvolvimento dos sistemas pelos usuários finais, apesar da limitação imposta pela disponibilidade de ferramentas adequadas. Martin [MAR 83] faz referências ao "projeto de uma só pessoa".

- a pesada participação dos usuários em todas as fases do processo de desenvolvimento.

- as técnicas de desenvolvimento devem considerar as mudanças nas necessidades do usuário exigindo uma rápida adaptação.

Este modelo clássico de desenvolvimento foi, por um bom tempo, aceito e difundido pelos profissionais da área de informática. Toda uma tecnologia foi construída baseada na estrutura imposta por este modelo. Metodologias e ferramentas foram construídas e aperfeiçoadas para atender a fases específicas do ciclo de desenvolvimento. Estas, no entanto, apresentavam algumas deficiências envolvendo, principalmente, os aspectos de documentação e manutenção dos sistemas.

Martin [MAR 83] prega a automação em todos os sentidos e salienta que os analistas de sistemas procuram automatizar todos os trabalhos, exceto o seu próprio. Neste sentido, Martin expõe uma série de quesitos que considera conveniente incorporar nas metodologias de PD. Dentre estes quesitos, cabe relacionar:

- automação como ponto chave;
- minimização da programação manual;
- flexibilização das modificações e controle automatizado das mudanças;

-verificação de correção;
-técnicas que facilitem comunicação com usuário;
-computação dirigida pelo usuário;
-projeto estável de banco de dados e linguagens poderosas de consulta;
-auxílio à eliminação de redundâncias;
-modularidade;
-controle de bibliotecas;
-poder evolutivo (versões, desenvolvimento incremental);
-dialetos alternativos para interfaces (conceituação, desenho e projeto de sistemas);
-conjunto integrado de ferramentas.

A engenharia de software tem investido muito na produção de ferramentas que sirvam de suporte as mais diversas fases do ciclo de vida. Neste elenco figuram as ferramentas CASE que procuram, através de recursos gráficos, facilitar o uso de determinadas metodologias e técnicas de análise e projeto de sistemas que, até então, dependiam de tarefas manuais realizadas por analistas e programadores.

Outra área que vem sendo explorada pela engenharia de software é a da aplicação de sistemas especialistas e técnicas de inteligência artificial na construção de ferramentas que apoiem as tarefas dos projetistas de sistemas. Este assunto é amplamente discutido em [AHL 90c].

2.2 Prototipação

A prototipação é uma abordagem de desenvolvimento de sistemas que considera a construção de uma versão preliminar, experimental ou evolutiva, do sistema aplicativo, ou parte dele, a fim de testar os princípios de seu funcionamento e equacionar o seu comportamento em confronto com as reais necessidades impostas pelo ambiente

do usuário. Em outras palavras, a intenção é construir um modelo operacional do aplicativo que imite o comportamento do futuro sistema a fim de se obter uma avaliação antecipada do projeto e permitindo, assim, os devidos ajustes antes que grandes investimentos sejam feitos.

Segundo Nogueira [NOG 87], como pre-requisitos tecnológicos para esta abordagem, devem ser considerados:

- -sistema de gerência de banco de dados;
- -linguagens interativas;
- -ferramentas para criação de telas, relatórios e diálogos.
- -ferramentas para criação, resolução e apresentação de modelos;
- -ferramentas que possibilitem a integração de dados e modelos.

No processo de concepção do protótipo é enfatizada a participação constante e ativa do usuário ao longo de todo o ciclo. O usuário acompanha o desenvolvimento, passo a passo, a medida em que os requisitos do sistema vão sendo definidos. Isto porque ninguém melhor do que o próprio usuário para dizer se o produto que está sendo gerado está de acordo com suas expectativas.

Este método de desenvolvimento pressupõe uma rápida prototipação a fim de permitir uma adequada realimentação na aprendizagem e no esclarecimento dos requisitos, fundamentada em observações feitas no comportamento de interfaces.

Riddle e Williams [RID 86] classificam as metodologias de prototipação em três classes:

- -Prototipação exploratória: protótipo usado com propósitos de formular problemas ou soluções.

-Prototipação experimental: protótipo usado para investigar estratégias de especificação e alternativas de soluções. Normalmente não chegam a ser implantados.

-Prototipação evolutiva: refinamentos sucessivos no protótipo até a obtenção de uma versão definitiva que originará o sistema final a ser implantado.

Como ambiente de desenvolvimento de protótipos, é recomendado que se pense em uma integração de ferramentas que dão suporte a diversas tarefas da prototipação. Segundo Gutierrez e De Souza [GUT 89], as linguagens de quarta geração, como Focus e Linc II, atendem aos requisitos impostos por estes ambientes. Como características dessas linguagens citam:

-compilador incremental;
-linguagem não procedural;
-S.G.B.D.;
-geradores de relatórios;
-facilidades de consultas "Ad-Hoc";
-facilidades gráficas;
-formatador de telas;
-dicionário de dados integrado;
-facilidades para processamento distribuído;
-facilidades de "restart" e "recovery";
-facilidades para análise estatística e financeira.

Horowitz e colegas [HOR 85] salientam que os geradores de aplicações e linguagens de quarta geração (L4G), atendendo basicamente à fase de projeto lógico ou de implementação, permitem que uma especificação seja, literalmente, transformada em código executável, simplificando consideravelmente o ciclo de vida do sistema e permitindo que usuários não qualificados possam desenvolver seus próprios aplicativos. Segundo Martin [MAR

83], as L4G fazem presunções inteligentes sobre o que considera que o usuário precisa (Ex.: seleção automática de formatos de relatórios, seleção de tipos para gráficos, etc.).

Com a utilização da metodologia de prototipação, o ciclo de desenvolvimento de sistemas fica reduzido a quatro passos básicos, conforme mostra a figura 2.2.

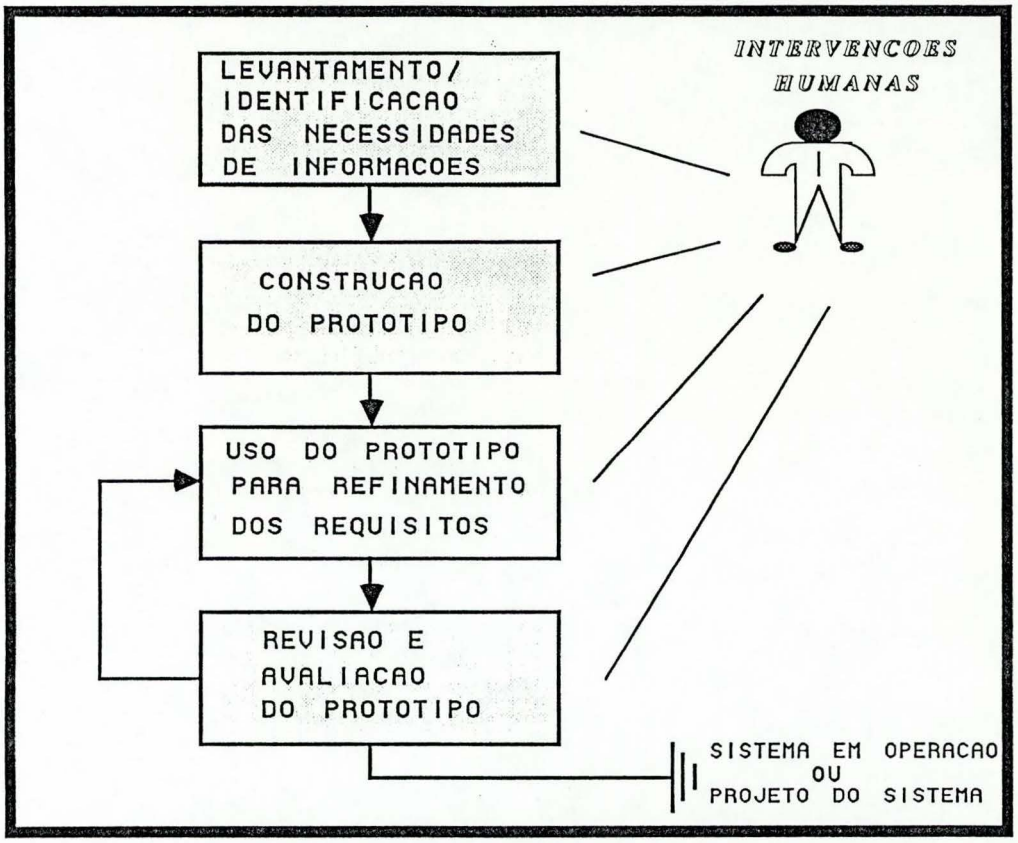


Fig. 2.2 - Ciclo de vida no paradigma da prototipação (Adaptado de [WEI 89])

A prototipação pode ser vista também como uma técnica para teste e/ou implementação de especificações. Uma vez obtida a especificação formal, torna-se necessário investigar as implicações comportamentais para avaliar se refletem as reais intenções de implementação. Nesta investigação aparecem duas correntes de pensamento:

a)usar um protótipo como meio de testar uma especificação através de sua execução simbólica [COH 82] [CHE 84];

b)converter a especificação em uma alternativa de implementação usando uma linguagem própria para construção de protótipos [FEA 82a].

O paradigma de prototipação pode também ser estendido para ambientes de sistemas especialistas. Com a crescente necessidade de construção de sistemas especialistas, assegurando uma rápida prototipação, aparece o conceito de "Shell" (embrião, gerador de sistema). Trata-se de um software gerador de sistemas especialistas que recebe os fatos e regras e, a partir deles, elabora toda a especificação, gerência e manipulação dos conhecimentos necessários à execução do sistema sendo concebido [AHL 90c].

2.3 Programação automática

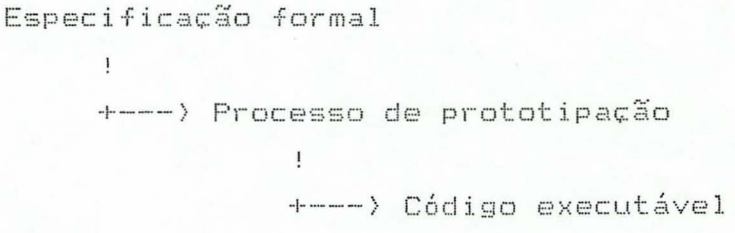
A programação automática pode ser vista como o processo de construção de programas onde, a partir de uma especificação informal ou semiformal submetida a sistemas transformacionais (mecanismos de suporte à transformação de especificações onde existe um mapeamento de uma especificação para outra), é obtido automaticamente o código executável, com a mínima intervenção humana. Existe um processo de sucessivos refinamentos, controlados por computador, mediante aplicação de regras de transformação, até a obtenção de código executável.

Embora a programação automática vise obter, como produto e de forma automática, um código executável, assemelhança dos pressupostos do paradigma de prototipação, estas duas formas de desenvolvimento de sistemas aplicativos apresentam, cada qual, as suas peculiaridades. Os detalhes que, basicamente, diferenciam os dois processos podem ser resumidos nas colocações de Barstow [BAR 82].

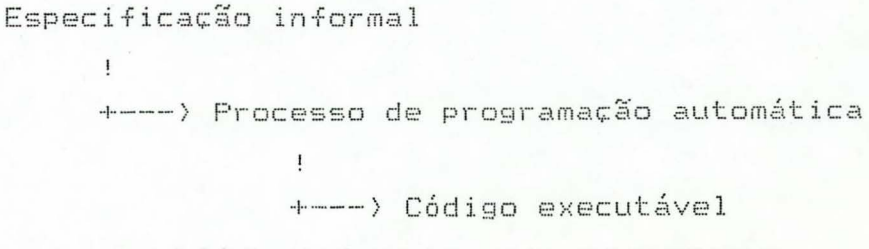
Em [BAR 82], o autor salienta que a programação automática pode ser vista como estratégia de desenvolvimento de protótipos. Barstow comenta que uma estratégia de especificações executáveis (prototipação) envolve a descrição do software em alguma linguagem de especificação (relativamente formal) que possui uma semântica operacional capaz de ser diretamente executada. Já no caso de usar-se uma estratégia de programação automática é possível iniciar a definição do sistema a partir de uma linguagem mais informal, principalmente se as ferramentas de apoio à programação automática forem interativas nas fases de aquisição da especificação e de implementação. Donde se conclui que a programação automática pode vir a ser uma boa alternativa para uma rápida prototipação visto que a especificação do protótipo pode estar mais próxima de uma linguagem natural (especificação informal: gráfica ou textual).

As colocações de Barstow podem ser sintetizadas pelo seguinte diagrama comparativo:

Paradigma de prototipação:



Paradigma de programação automática:



A prática de programação, por si só, é um processo de transformação. Utiliza uma especificação informal ou semiformal e, a partir de um agente representado pelas técnicas de programação sendo manipuladas pelo programador (num processo altamente dependente do fator humano), produz uma especificação formal (programa fonte). Na escalada em direção à programação automática, a idéia básica é a de minimizar a intervenção humana no processo de construção dos programas/sistemas e, assim, automatizando o desenvolvimento transformacional de software.

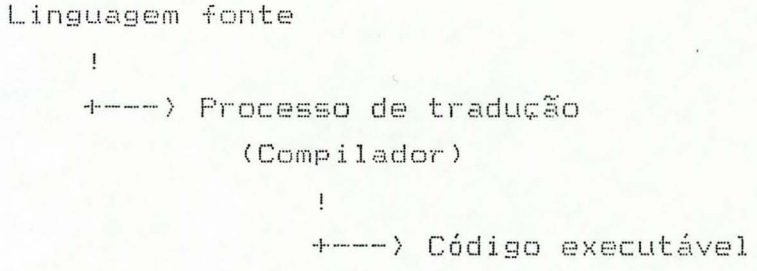
As ferramentas de programação automática devem incorporar conhecimento suficiente sobre técnicas de programação para servir de ponte na lacuna existente entre o escopo de linguagens de especificação e o escopo de uma linguagem convencional de programação.

Este tema e assuntos correlatos à programação automática foram amplamente discutidos e analisados, anteriormente, em [AHL 90a], onde a intenção foi a de consolidar as diversas idéias que incorporam e fundamentam o paradigma de construção automática de sistemas. Deste trabalho foi possível concluir que a abordagem de programação automática, baseada em sistemas transformacionais, nada mais é do que um processo de compilação/tradução só que em nível mais elevado onde a linguagem de especificação que alimentará o processo está mais próxima da linguagem natural (humana) e existem mais estágios de refinamentos necessários até obter código executável. A máquina começa a prestar suporte desde o início do ciclo de desenvolvimento e o destino final permanece sendo a obtenção do código executável. A idéia é entregar uma especificação, o mais próxima de uma linguagem natural ou informal, a um sistema transformacional e receber, como produto, um programa executável. Toda a tarefa de desenvolvimento, ou a maior parte dela, passa a

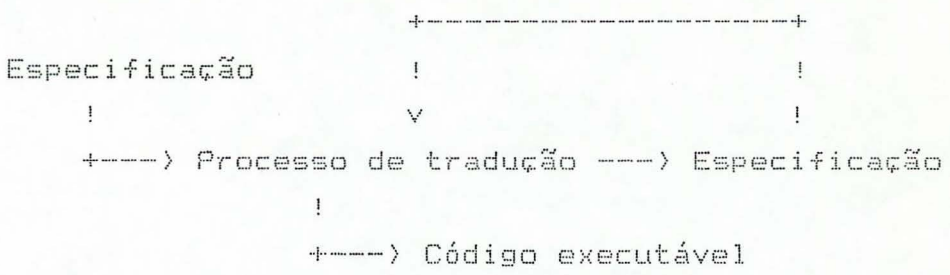
ser realizada, de forma automática, pelo computador.

No confronto dos estágios da programação convencional com os da programação automática, quanto ao suporte que o computador oferece, chega-se ao seguinte diagrama representativo das transformações que ocorrem no processo:

Programação convencional:



Programação automática:



No ciclo da programação automática, um dos estágios críticos, e que deve receber uma atenção especial por parte das futuras pesquisas, é o da aquisição da especificação onde é necessário obter uma descrição formal que reflita as reais intenções do sistema, representada na especificação informal original através de algum mecanismo linguístico ou gráfico. Esta especificação formal será o ponto de partida de todo processo transformacional.

Um dos principais trabalhos no contexto do paradigma de programação automática situa-se nas pesquisas

de Balzer e sua equipe ([BAL 81], [BAL 83a], [BAL 85a]). Como resultado de seus trabalhos, Balzer propõe um novo ciclo de desenvolvimento de sistemas fundamentado no princípio da implementação transformacional [BAL 81]. A arquitetura proposta por Balzer [BAL 85a] está ilustrada na figura 2.3.

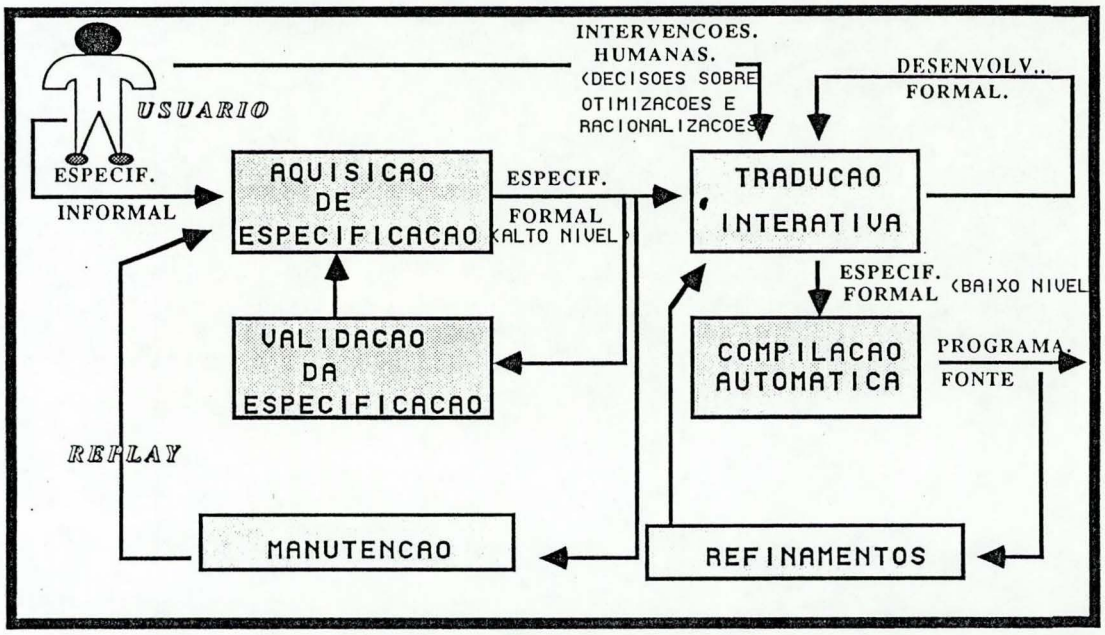


Fig. 2.3 - Ciclo de vida no paradigma da programação automática (Adapatado de [BAL 85a])

Nesta arquitetura tem-se intervenção humana somente na fase de aquisição da especificação, onde o usuário fornece a descrição do problema em uma linguagem informal, e durante a tradução interativa, alimentando o sistema com decisões sobre otimizações e racionalizações do processo de transformação para aqueles detalhes que os algoritmos de refinamento não possam ser autosuficiente. O restante do processo é automaticamente executado sob

controle e gerência do sistema transformacional. Neste modelo salienta-se também a fase de manutenção sendo realizada sobre a própria especificação original e num procedimento de "Replay" (reimplementação).

Na estrutura de desenvolvimento formal, associada à tradução interativa, é registrada a história de derivações a partir da sequência de transformações aplicadas. Com isso permite-se que uma nova implementação seja rederivada de uma especificação modificada e somente os procedimentos envolvidos na alteração sejam reeditados, mantendo os demais inalterados.

Nesta arquitetura todos os conhecimentos sobre a prática de programação são mantidos em uma base de conhecimentos, configurada a partir de fatos e regras, e gerenciado por sistemas especialistas, utilizando técnicas de I.A.

3. AS CARACTERÍSTICAS DE FERRAMENTAS DE SUPORTE A CONSTRUÇÃO AUTOMÁTICA DE SISTEMAS

3.1 O perfil da ferramenta fixando o grau de automação

Observando o ciclo de vida em cada um dos três paradigmas de construção de sistemas, discutidos no capítulo 2, é possível identificar diferentes fases associadas a cada modelo em particular.

Cada fase pode ser alvo da abertura de um leque de perspectivas quanto à aplicação de ferramentas para automatizar ou dar suporte a tarefas específicas.

O número de fases tende a se reduzir a medida que aumenta o grau de automação do processo de desenvolvimento de software. Fases como especificação, projeto e codificação/depuração do modelo tradicional em cascata são agrupadas em fase única de especificação e construção do protótipo no paradigma de prototipação.

De um modo geral, pode-se afirmar que o número de intervenções humanas no processo de construção do software é diretamente proporcional ao número de fases que compõem o ciclo de desenvolvimento aliado ao perfil das ferramentas que prestam suporte a tarefas dentro de cada fase. A figura 3.1 ilustra o ciclo de vida nos paradigmas de desenvolvimento de sistemas, anteriormente discutidos, mostrando os locais onde são previstas as intervenções de usuários finais, analistas e programadores.

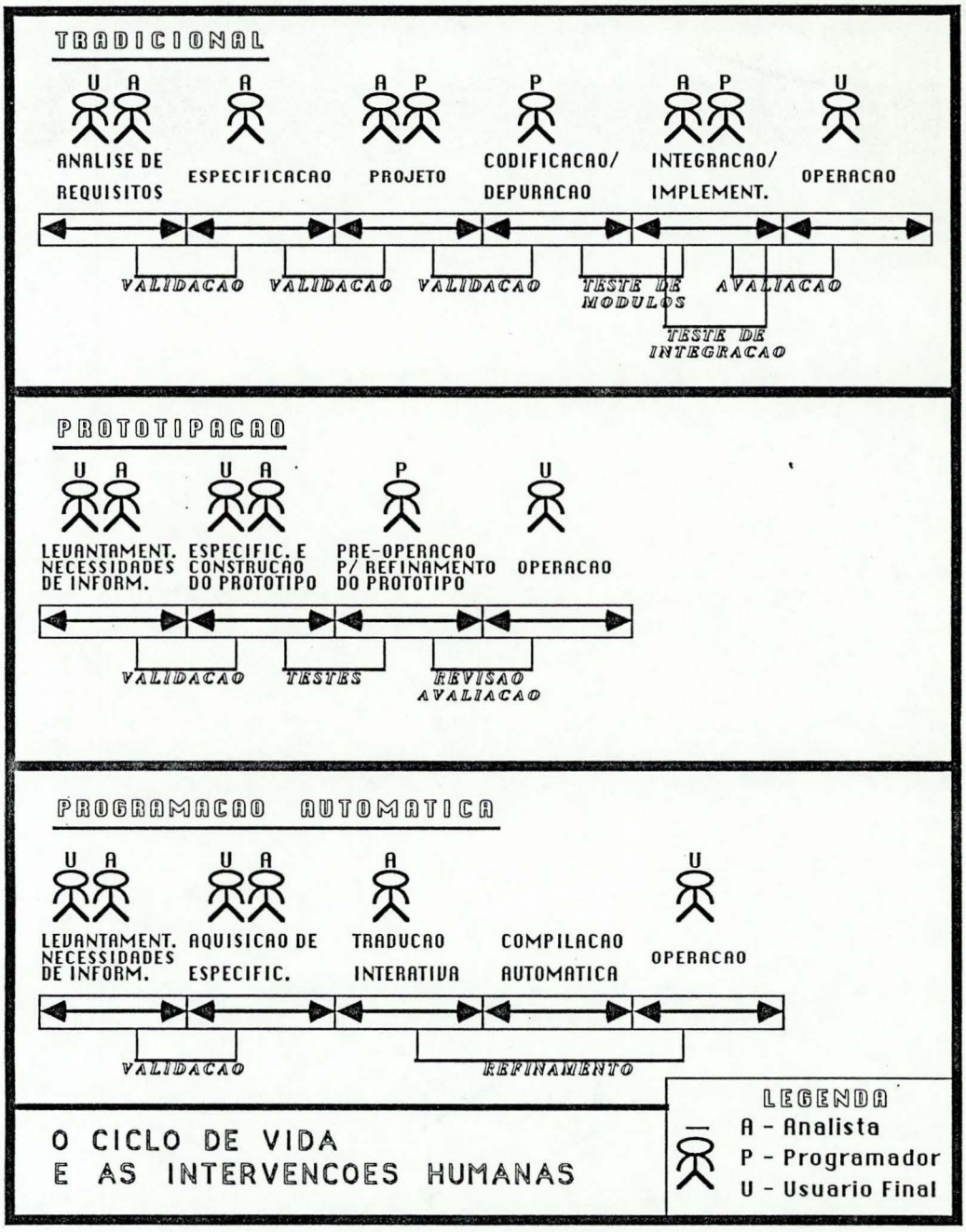


Fig.3.1 - O ciclo de vida e as intervenções humanas

As características da ferramenta de suporte estabelecem o grau de automação do processo de construção de sistemas. Algumas ferramentas simplesmente reduzem o trabalho braçal na especificação e projeto do software, como o caso das ferramentas CASE que permitem, através dos recursos gráficos, substituir as tarefas manuais, normalmente muito trabalhosas, de desenho dos componentes

(estáticos e dinâmicos) de um sistema de informações. Estas ferramentas muitas vezes viabilizam uma determinada metodologia que, sem a ferramenta, seria impraticável em vista dos inúmeros controles e detalhamentos a serem observados.

Outras ferramentas, como os geradores de aplicações, incorporam um maior grau de conhecimento sobre técnicas de análise e de programação, permitindo assim aumentar o grau de automação e, consequentemente, reduzir as intervenções humanas. Estas ferramentas passam a assumir um maior nível de decisões próprias sobre as tarefas que implementam. Sob este aspecto, as pesquisas na área de inteligência artificial podem trazer grandes contribuições [AHL 90c], pois a forma de representação e manipulação do conhecimento (regras, frames, redes semânticas) dão maior flexibilidade à especificação e, com isso, as ferramentas de suporte à construção de sistemas podem atingir um espectro maior de tipos de aplicações. Neste sentido, a tendência é gerar ferramentas cada vez mais inteligentes onde se procura incorporar mecanismos de aprendizagem para que possam inferir novos conhecimentos a partir do conhecimento armazenado.

A intenção destas pesquisas vem sendo a diminuição progressiva de intervenções humanas no processo de concepção do software, objetivando a busca de uma automatização das tarefas de analistas e programadores.

3.2 Critérios de classificação de ferramentas

Considerando o elenco de ferramentas de suporte à construção automática de sistemas, a presente seção visa avaliar as características destas ferramentas estabelecendo alguns critérios de classificação.

Em virtude do tema básico desta monografia situar-se no estudo das características diretamente ligadas

aos aspectos da construção automática de sistemas e, por outro lado, não desprezando aquelas que se relacionam com os demais aspectos operacionais de manuseio e funcionamento das ferramentas, os critérios de classificação foram divididos em 2 grandes grupos:

.critérios no âmbito da construção automática de sistemas;
.critérios adicionais.

Ambos os critérios foram subdivididos focalizando os diversos aspectos operacionais que ditam o perfil de uma ferramenta e, para cada aspecto, listadas as alternativas de enquadramento da ferramenta, seguidas de um breve comentário.

3.2.1 Critérios no âmbito da construção automática de sistemas

Estes critérios agrupam as características ligadas aos aspectos da construção do sistema propriamente dito e estão relacionados a:

1)Perfil da especificação:

.linguagem natural;
.suporte gráfico à especificação (diagramação da especificação);
.linguagem formal de especificação;
.parametrização (assinalamento de opções);
.asserções matemáticas.

Para o perfil da especificação foram enumeradas todas as formas usuais da ferramenta receber e tratar as especificações do sistema. Muitas destas ferramentas admitem formas mistas conforme o tipo de detalhe do sistema que estiver sendo modelado. Por exemplo, suporte gráfico na

especificação do fluxo de documentos ou processos de um sistema e linguagem formal na especificação de lógica dos procedimentos. Ou então, a lógica especificada através de uma linguagem formal complementada por tabelas de decisões representadas graficamente.

2)Especificação de semântica:

+---+
| .associado aos dados no dicionário de dados; |
| .tabelas de consistência/validação; |
| .referência a macros ou rotinas pre-codificadas; |
| .programado. |
+---+

Sob este critério foram enumeradas as alternativas de definir a semântica de manipulação dos dados para o sistema aplicativo a ser gerado pela ferramenta.

Antes do surgimento da tecnologia de banco de dados (modelos hierárquicos e em rede), toda semântica para tratamento de dados se baseavam na codificação de procedimentos em alguma linguagem. O programa era o centro de gravitação de um sistema.

A tecnologia de banco de dados trouxe novos conceitos e técnicas e, entre estes, aparecem os mecanismos incorporados nos SGBD onde figuram, por exemplo, as rotinas padrões de acesso (primitivas de manipulação do banco de dados) pre-programadas. A simples referência a estas rotinas nos programas, através de uma linguagem de manipulação (DML), permite efetivar a busca dos dados no banco de dados de forma padronizada e independente do programa.

A necessidade de inovar a tecnologia de BD, para atender a aplicações mais complexas, fez surgir a idéia de buscar soluções na experiência obtida a partir de pesquisas

em outras áreas. Um exemplo é o aproveitamento dos resultados obtidos da investigação de orientação a objetos na área de linguagens de programação [AHL 90b]. Para este novo enfoque, onde figuram os BD semânticos e de orientação a objetos, a preocupação passa a ser a associação de significativa parcela de semântica aos dados.

Todas estas idéias vem sendo aproveitadas pela engenharia de software na fixação de características das ferramentas de construção de sistemas.

Outra forma alternativa de estabelecer semântica tem origem nos programas genéricos de consistência que utilizam tabelas de consistência/validação para armazenar informações semânticas de controle dos algoritmos de validação dos dados (tipo de campo, faixa de valores válidos, teste de dígito de controle, teste de datas, rotinas especiais para tratamento do campo, etc). Este mecanismo está ilustrado na figura 3.2.

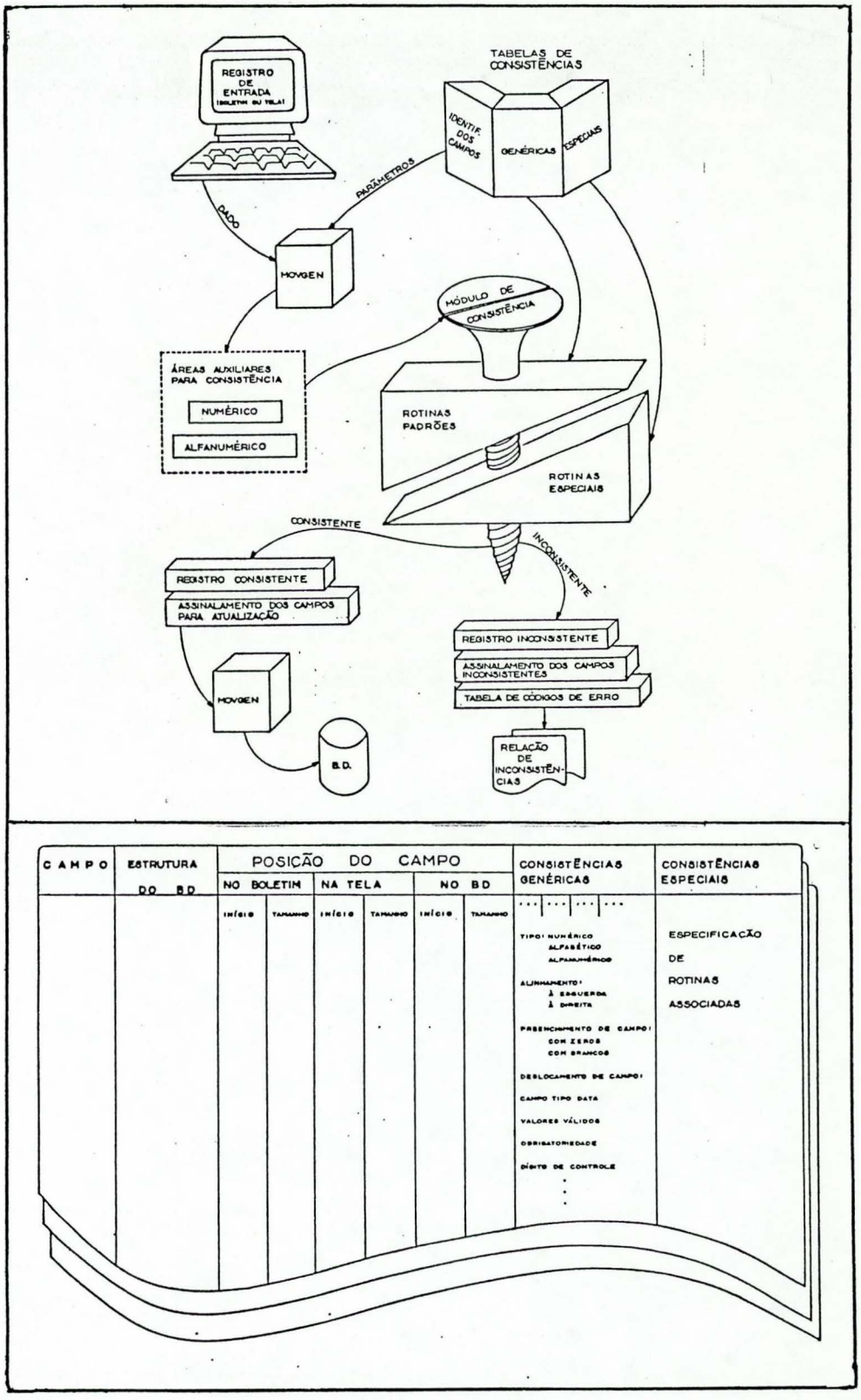


Fig.3.2 - Tabelas de consistência fixando a semântica dos programas genéricos de consistência (Extraído de [KOR 87])

3)Construção do banco de dados:

+--+
| .manual a partir de linguagem específica; |
| .automática a partir do projeto das entradas do sistema. |
+--+

Algumas ferramentas simplesmente mantém uma interface de definição do banco de dados com auxílio de telas de parametrização ou um editor de especificações que considera a DDL (Data Definition Language) do SGBD utilizado ou uma variante dela.

Por outro lado, uma tendência atual é automatizar todo este processo de construção do BD de forma que o projetista só se preocupe com o projeto das entradas (telas), deixando que o mapeamento do modelo externo para o modelo interno (BD e transações básicas de atualização do BD) seja gerado automaticamente pela ferramenta.

4)Construção dos programas (geração de código):

+--+
| .manual (codificação & compilação); |
| .semi-automática (esqueletos de rotinas/programas pre-programados e complementados pela especificação e com intervenções manuais no processo de geração); |
| .automática a partir de uma especificação. |
+--+

Para a construção dos programas, o desejável é que todo este processo de concepção e desenvolvimento seja transparente para quem utiliza a ferramenta de construção automática de sistema e que estes programas sejam gerados automaticamente a partir da especificação,sem a necessidade de indicações ou definições (quantos e quais programas) por parte do analista/programador.

Algumas ferramentas, porém, exigem uma maior intervenção do programador em atividades como codificação de trechos de programas, compilações, etc, tornando o processo manual ou semi-automático.

5)Validação da especificação e dos programas gerados:

a)Quanto à verificação sintática:

.imediata, durante a especificação;
.postergada para a fase de compilação.

b)Quanto à verificação semântica:

.mecanismos de geração de testes a partir da especificação;
.facilidades de execução simbólica;
.facilidades de mensuração dos efeitos (simulações);
.testes manuais.

A ferramenta pode dispor de mecanismos mais ou menos sofisticados para validação da especificação e dos programas gerados. Pode, por exemplo, validar sintaticamente uma especificação durante o processo de definição evitando que gere fonte de programas com erros de sintaxe, detectáveis somente na fase de compilação.

Em termos de validação semântica dos programas e especificações, a ferramenta pode ter recursos de simulações prévias ou então executar simbolicamente uma especificação para avaliar o seu comportamento. Também pode ter rotinas para geração de arquivos de testes a partir da própria especificação.

6)Armazenamento das especificações:

+--+
| .em um banco de especificações; |
| .em dispositivos independentes, isoladas por fases. |
+--+

Uma vez gerada a especificação, ela pode ser armazenada e mantida pela própria ferramenta em um banco de dados para esta finalidade ou então ela pode transferir ,para o analista, a responsabilidade de controlar e salvar as especificações. Este controle manual é inevitável quando a especificação não é auto-suficiente para gerar código executável, exigindo codificações manuais de programas ou trechos de programas.

7)Suporte ao projeto das E/S:

a)Quanto à especificação:

+--+
| .codificação manual de telas/relatórios: |
| -com linguagem convencional de programação; |
| -com linguagem específica; |
| .facilidades de desenho de telas/relatórios("Paint"). |
+--+

b)Quanto à validação:

+--+
| .mecanismos de apresentação e simulação da execução |
| no próprio ambiente da ferramenta; |
| .teste via programa aplicativo. |
+--+

A definição de telas e relatórios é uma tarefa básica na construção de sistemas. Para gerar uma tela, ou então um relatório, a ferramenta de suporte pode ter características que lhe permitem o desenho do layout diretamente no vídeo, interpretar este layout e gerar código correspondente para materializá-la.

Mecanismos de simulação da apropriação de dados via tela projetada, avaliando os efeitos de rotinas de validação de dados, podem estar incorporados no ambiente da ferramenta. Isto permite testes preliminares de entrada de dados antes mesmo de gerar os programas aplicativos.

8)Suporte à implantação:

!.testes de validação a cada etapa;
!.prototipação.

A ferramenta pode ser um instrumento de prototipação, permitindo uma rápida demonstração do funcionamento do futuro sistema num processo de validação simultânea com o desenvolvimento. Em alguns casos, no entanto, a ferramenta não dispõe desta potencialidade, exigindo testes individualizados de validação a cada etapa concluída.

9)Procedimentos de manutenção:

!.sobre a especificação;
!.sobre o programa fonte.

Considerando que a manutenção é uma fase bastante onerosa no ciclo de vida de um software, o ideal é que todo e qualquer procedimento de manutenção seja sempre feito sobre a especificação e jamais seja necessário realizar uma alteração diretamente sobre o fonte do programa. Isto evita complexos controles para implementação e gerenciamento das modificações em um sistema.

10)Aspectos ligados ao contexto da transformação de especificações (sistemas transformacionais): (segundo [PAR 83])

a)Quanto à organização e tipos de transformação:

+--+
| .uso de catálogos de regras (sistemas baseados em |
| conhecimentos) onde as regras de transformação são |
| predefinidas para algum aspecto particular como: |
| -regras sobre conhecimento de programação; |
| -regras de otimização com base em facilidades de |
| linguagens; |
| -regras sobre conhecimento do domínio dos dados; |
| -informações de eficiência; |
| -regras para seleção automática de estrutura de |
| dados. |
| .uso de um conjunto gerador, ou seja, a partir de um |
| conjunto básico de regras para transformações |
| elementares, construir novas regras. |
+--+

b)Quanto à forma das regras de transformação:

+--+
| .na forma de algoritmos de transformação (regras |
| procedurais), normalmente empregadas para regras |
| globais (regras semânticas) como: |
| -regras de verificação de consistência; |
| -regras de representação de técnicas de |
| programação; |
| -regras de semântica das operações. |
| |
| .na forma de regras de troca de padrões (regras |
| esquemáticas), normalmente usadas em conexão com |
| regras locais (regras de refinamento) como: |
| -regras sintáticas de correlação de linguagens; |
| Ex.: (L:if B then S; goto L fi) equivale à |
| (while B do S od) |
| -regras de descrição de propriedades algébricas; |
| Ex.: (i + if B then x else y fi) equivale à |
| (if B then i+x else i+y fi) |

```
|     -regras de domínio para propriedades dos  tipos |
|      de dados;                                       |
|      Ex.: (b ^ b) equivale à (b)                     |
|                                                      |
|.na forma híbrida.                                    |
+------------------------------------------------------+
```

c)Quanto a características do processo de transformação:

```
+------------------------------------------------------+
|.automação:                                           |
|     -transformação sob controle do usuário;          |
|     -sistema de transformação semi-automático;       |
|     -sistema de transformação automático.            |
|                                                      |
|.casamento de representações:                         |
|     -de primeira ordem:  somente instâncias  de      |
|      variáveis objetos;                              |
|     -de segunda ordem: variáveis objetos e variáveis |
|      de funções.                                     |
|                                                      |
|.forma de avaliação das regras(ativação a partir de): |
|     -condições de habilitação da regra;              |
|     -condições  estratégicas  (seleção baseada  no   |
|      objetivo a ser alcançado);                      |
|     -asserções  (com  base  nas  propriedades  dos   |
|      objetos);                                       |
|     -informações  sobre  o  escopo  de aplicação da  |
|      regra (em que ponto aplicar outra regra).       |
+------------------------------------------------------+
```

O mecanismo de transformação de especificações é o núcleo básico de uma ferramenta de construção automática de sistemas. Esta transformação pode ser realizada considerando diferentes níveis de especificação (informal para formal, formal para formal) e eventualmente em um processo de sucessivos refinamentos.

Um estudo bastante abrangente sobre sistemas

transformacionais pode ser encontrado em [PAR 83]. Um levantamento bibliográfico comentado sobre o assunto foi também, anteriormente, realizado em [AHL 90a].

Os critérios acima relacionados foram sintetizados a partir das considerações feitas no artigo do Partsch e Steinbruggen [PAR 83] e sumariza os diversos aspectos ligados ao processo de transformação de especificações.

Alguns itens são característicos de ferramentas desenvolvidas a partir da tecnologia de inteligência artificial. Outros itens, porém, aparecem também no elenco de ferramentas que utilizam a tecnologia tradicional de programação de sistemas.

3.2.2 Critérios adicionais

Nestes critérios foram inseridas as características que afetam o ambiente operacional de manuseio e funcionamento da ferramenta e que, no entanto, não tem influência direta sobre o processo de construção automática dos sistemas.

Como critérios adicionais, foram considerados:

1)Escopo de aplicação:

.propósitos específicos (aplicações altamente especializadas como, por exemplo, CAD);
.propósitos gerais (aplicações tradicionais).

É o nível de conhecimento que a ferramenta tem sobre a aplicação. Ferramentas voltadas à construção de sistemas aplicativos de propósitos específicos, tem uma afinidade maior com a tecnologia de inteligência artificial onde aparecem os "Shells" como geradores de sistemas

especialistas [AHL 90c]. Enquadrá-los na área da inteligência artificial se deve ao fato de que, nesta área, existe uma maior versatilidade na representação e manipulação do conhecimento da aplicação, através da utilização de regras de produção, frames, redes semânticas, etc, o que permite maior flexibilidade na especificação (mais naturalmente).

As aplicações de propósitos gerais, por sua vez, podem ser vistas como produtos obtidos de ferramentas do tipo geradores de aplicações convencionais e linguagens de quarta geração.

2)Interfaces de banco de dados:

!.programáveis com comandos navegacionais (SGBD hierárquicos e em rede);
!.não procedurais (SGBD relacionais);
!.gráficos específicos (Ex.: QBE);
!.especializados no tratamento de textos;
!.de manipulação especializados;
!.inteligentes (Auto-adaptativas).

Dizem respeito a como a ferramenta irá manter o intercâmbio com o banco de dados para o acesso e manipulação dos dados. Pesquisas recentes procuram direcionar esforços na busca de interfaces auto-adaptativas ou inteligentes. Para Oliveira e Hoppen [OLI 87], interfaces inteligentes apresentam características como:

- -oferecem uma forma de representação lógica de objetos coerente com a manipulação rotineira feita por parte do usuário;
- -integram os métodos de manipulação de dados (homogeneidade);
- -o resultado de consultas podem ser armazenadas

como tipo de dados no banco de dados.

3)Diálogo homem-máquina:

+--+
| .dirigido pelo computador (assistido;do tipo pergunta |
| e resposta); |
| .navegacional (menus e janelas); |
| .manipulação direta com suporte gráfico (desenhos com |
| o auxílio de mouse); |
| .comandos de especificação: |
| .em linguagem natural (informal); |
| .em linguagem de especificação (formal); |
| .em linguagem de programação. |
+--+

É através do diálogo homem-máquina que o projetista do sistema se comunica com a ferramenta e é um fator que influencia o grau de satisfação e motivação no manuseio da ferramenta. Sendo assim, é tema de estudo ligado às pesquisas nos aspectos humanos associados ao projeto e desenvolvimento de peças de software (ergonomia na construção de software).

As pesquisas na área de comunicação do homem com o computador tem como premissa básica a minimização da dificuldade das pessoas em aprender o uso dos sistemas.

Sob este contexto, foram enumeradas algumas formas como a ferramenta de construção de sistemas pode se comunicar com o projetista.

4)Perfil do dicionário de dados:

+--+
| .integrado; |
| .isolado para cada fase do projeto e com eventuais |
| mecanismos de mapeamento entre fases; |
| .ativo (interação do sistema em tempo de execução). |
+--+

O dicionário de dados, como repositório de informações sobre os dados manipulados pelo sistema, é um dos principais módulos e "peça chave" de uma ferramenta de construção de sistemas.

Quando o ambiente de desenvolvimento de sistemas for composto por um conjunto de ferramentas de suporte, atendendo fases distintas e, cada qual com um dicionário próprio, é importante que haja algum mecanismo de mapeamento entre eles.

Algumas ferramentas mantém um dicionário de dados ativo que troca informações com o sistema aplicativo gerado também em tempo de execução. Isto, no entanto, faz com que o sistema permaneça constantemente vinculado ao ambiente operacional da ferramenta ou, que ambos, ferramenta e sistema aplicativo, compartilhem um dicionário de dados integrado.

5)Suporte à documentação do sistema:

!.construção automática a partir da especificação;
!.construção semi-automática (usuário complementa as especificações);
!.construção dirigida pelo computador;
!.manual.

A documentação do sistema é um item que vem sendo negligenciado por analistas e programadores. Isto se deve ao fato de ser uma tarefa bastante trabalhosa, quando feita manualmente,e em vista de ser, na maioria dos casos, postergada para após o processo de desenvolvimento quando os responsáveis já não possuem a mesma motivação acerca do sistema.

Todavia, a elaboração de uma boa documentação é fundamental para os profissionais de informática em vista de futuras manutenções no sistema e fundamental para o usuário final porque possibilita um correto manuseio deste sistema.

Por conseguinte, é interessante que a ferramenta ofereça máximas condições para favorecer a elaboração desta documentação.

6)Gerenciamento do ambiente de desenvolvimento:

.manual, por fase;
.integração do ambiente (ferramentas integradas).

Quando, durante o ciclo de desenvolvimento do sistema, forem necessárias diversas ferramentas para tarefas específicas (ferramentas CASE, geradores de telas/relatórios, geradores de programas, etc) nas diferentes fases, o gerenciamento de todo ambiente operacional (salva de arquivos, mapeamento entre ferramentas, etc) provavelmente é executado de forma manual.

Por outro lado, uma tendência atual surge em direção a busca de um ambiente unificado de desenvolvimento com uma integração total entre as diferentes ferramentas envolvidas no processo. Esta integração se materializa através do compartilhamento de bancos de especificações, de dicionário de dados e de rotinas operacionais (segurança, recuperação, backup, etc) e através de uma padronização nos principais diálogos com/entre as ferramentas.

7)Perfil da metodologia de análise e projeto:

+---+
| .estática e predefinida; |
| .configurada dinamicamente segundo necessidades |
| específicas: |
| -com adição de regras pelo usuário; |
| -com derivação de novas regras (aprendizado). |
+---+

As ferramentas de suporte à construção automática de sistemas visam automatizar as atividades de análise e programação e, geralmente, são fundamentadas em uma determinada metodologia que, normalmente, é estática e pre-definida.

Contudo, tem surgido algumas pesquisas que enfatizam a necessidade de construir ferramentas que generalizam o uso de metodologias, ou seja, são flexíveis o suficiente para permitir uma configuração dinâmica da metodologia conforme necessidades específicas de determinada instalação. Neste caso, aparece associado à ferramenta um módulo de configuração de métodos e técnicas para compor a nova metodologia. Este é também um tema relacionado com o uso da inteligência artificial na engenharia de software [AHL 90c].

8)Definição de mecanismos de controle de acesso ao sistema (autorizações):

a)Quanto à especificação:

+---+
| .externo ao sistema; |
| .no dicionário de dados; |
| .associado a cada estrutura/componente do sistema; |
| .em tabelas/estruturas de controle; |
| .programado. |
+---+

b)Quanto à abrangência:

.a nível de conteúdo de atributos;
.a nível de atributos;
.a nível de registros;
.a nível de estruturas;
.a nível de transações (operações combinando níveis anteriores).

Dizem respeito às formas de associar procedimentos de segurança e controle de acesso aos sistemas gerados. Estes mecanismos se baseiam na indicação do que o usuário pode acessar no sistema (estruturas, registros , atributos, etc) e de que forma pode realizar este acesso (consultas, atualizações).

Estes mecanismos, uma vez incorporados ao sistema através da ferramenta, buscam os indicadores de autorizações (para confronto usuário vs componentes do sistema) a partir da especificação mantida em algum dispositivo como tabela ou estrutura de controle, associado a um dicionário de dados, associado às estruturas ou componentes do sistema ou, então, representada através de alguma lógica programada.

Pode ocorrer também que a ferramenta não ofereça estes recursos, devendo então serem providenciados, caso necessário, por algum mecanismo externo ao sistema.

O nível de abrangência se refere à granularidade da especificação do controle de acesso e se aplica basicamente nos casos em que os indicadores de autorizações forem definidos de forma parametrizada (no dicionário de dados, em tabelas/estruturas de controle, associados a componentes do sistema ou em alguma interface externa ao sistema), já que, para a especificação programada, todos níveis são válidos desde que devidamente codificados.

9)Suporte à recuperação/reconstrução do sistema:

!.externo ao sistema;
!.mecanismos incorporados ao SGBD;
!.mecanismos próprios (arquivos tipo LOG, Journal,etc)

Em geral a ferramenta lança mão dos recursos de recuperação/reconstrução embutidos no SGBD sobre o qual baseou sua arquitetura. Tem aquelas, no entanto, que mantém algum mecanismo próprio incorporando-o no sistema a ser gerado. Em muitos casos, este mecanismo próprio é acrescido aos recursos do SGBD objetivando um aumento no nível de segurança.

10)Suporte à reorganização do sistema:

!.controle e execução manual;
!.transparente ao usuário/projetista.

Uma modificação eventual na especificação do sistema pode exigir a realização de procedimentos de reorganização das estruturas do banco de dados. A tarefa de controle e execução destes procedimentos normalmente é realizada pelo setor de suporte de forma manual. Uma alternativa interessante, no entanto, é a realização automática destes procedimentos pela própria ferramenta a medida que constatar uma modificação que exija uma reorganização.

4. ESTUDO DE CASO: COMPARAÇÃO DE FERRAMENTAS SEGUNDO OS CRITÉRIOS ESTABELECIDOS

Baseado nos critérios de classificação de ferramentas estabelecidos na seção 3.2, o presente capítulo apresenta um estudo comparativo entre duas ferramentas de suporte à construção de sistemas, o LINC e o SADS, no intuito de enquadrá-las na taxonomia de características obtida a partir destes critérios.

4.1 LINC

4.1.1 Considerações preliminares

O LINC (Logic and Information Network Compiler) é uma ferramenta para construção de sistemas aplicativos, produzida e comercializada pela UNISYS Corporation ([UNI 88a], [UNI 88b], [UNI 88c]) e originalmente desenvolvido na Nova Zelândia.

Segundo o fornecedor deste software, o LINC pode ser visto como:

- .uma ferramenta para aumento de produtividade no desenvolvimento de sistemas;
- .um método eficiente de gerar sistemas on-line e banco de dados completos e prontos para execução;
- .uma linguagem interativa de quarta geração;
- .um gerador de sistemas dentro do contexto da prototipação e com efetiva participação do usuário final no processo;
- .uma ferramenta para automatizar o projeto de banco de dados.

Em sua versão original (LINC I, versão 10), toda especificação do sistema aplicativo (telas, estruturas e

lógica) era feita, de forma não interativa, através de uma linguagem própria, o LDL (Linc Definition Language), ao estilo de uma "Data Division" do Cobol. Esta especificação era, então, submetida a um compilador que produzia as definições (DASDL - Data And Structure Definition Language) do banco de dados DMSII e fontes Cobol para a implementação das aplicações (transações) sobre este banco de dados.

Atualmente, já com a denominação de LINC II, a ferramenta possui um compilador interativo composto por módulos de desenho de telas, de especificação de lógica, de desenho de relatórios, de manutenção de dicionário de dados e de geração do sistema, totalmente integrados em um único ambiente de desenvolvimento do sistema aplicativo, como mostra a figura 4.1. O projetista realiza a especificação do sistema de forma totalmente interativa e as validações sobre esta especificação são feitas instantaneamente, no momento da transmissão da linha de definição, a fim de que o produto gerado a partir desta especificação seja um programa Cobol compilado sem erros de sintaxe. Toda manutenção também é realizada sempre sobre a própria especificação, não necessitando, jamais, realizar alterações sobre o fonte do programa.

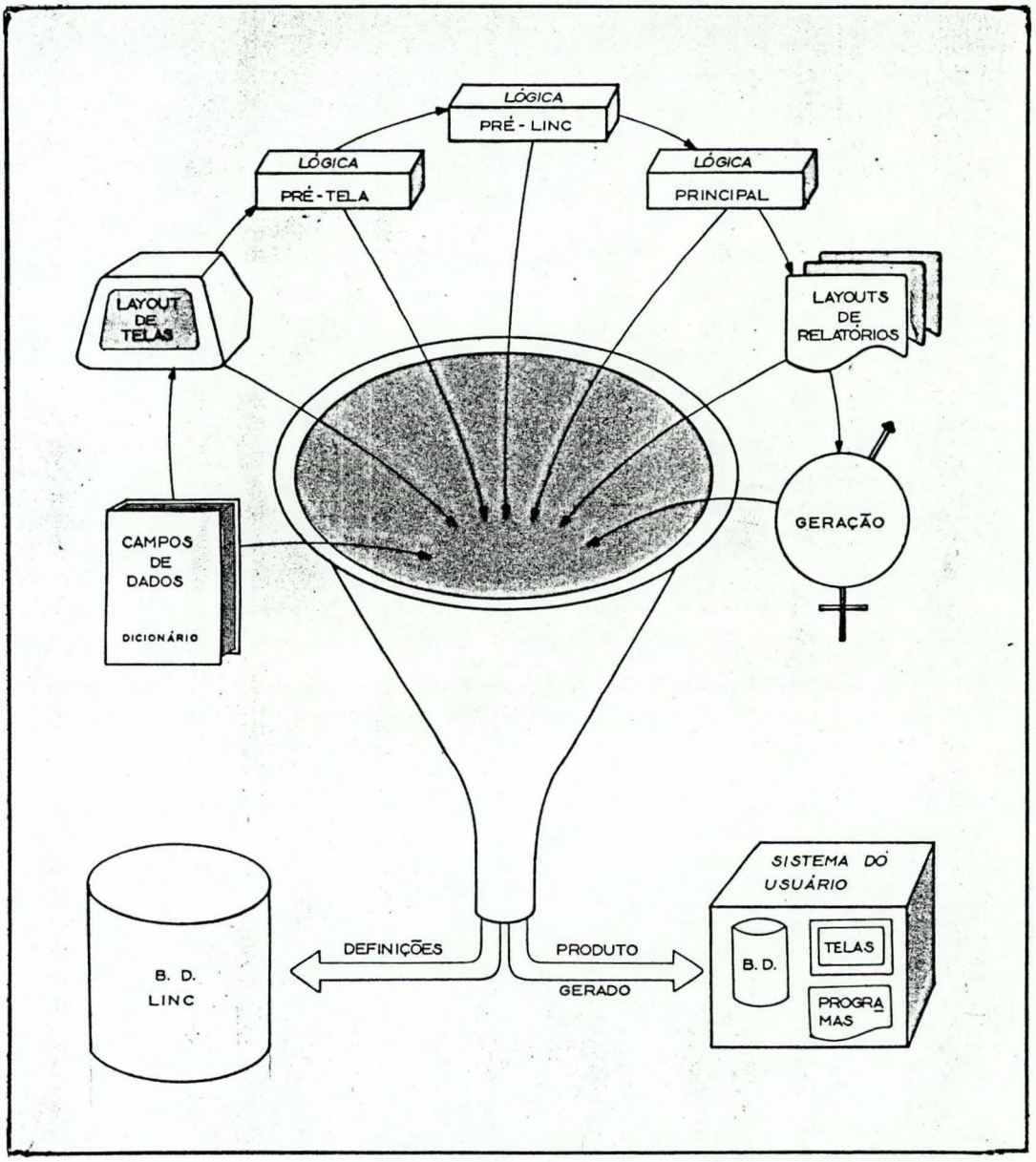


Fig. 4.1 - A construção de sistemas através do LINC (Extraído de [AHL 88])

O compilador interativo LINC II é constituído de telas voltadas a "Menu" onde o projetista realiza uma navegação sobre as diversas telas para a definição completa do sistema aplicativo. A tela que enumera e aciona as principais funções do LINC está ilustrada na Figura 4.2.

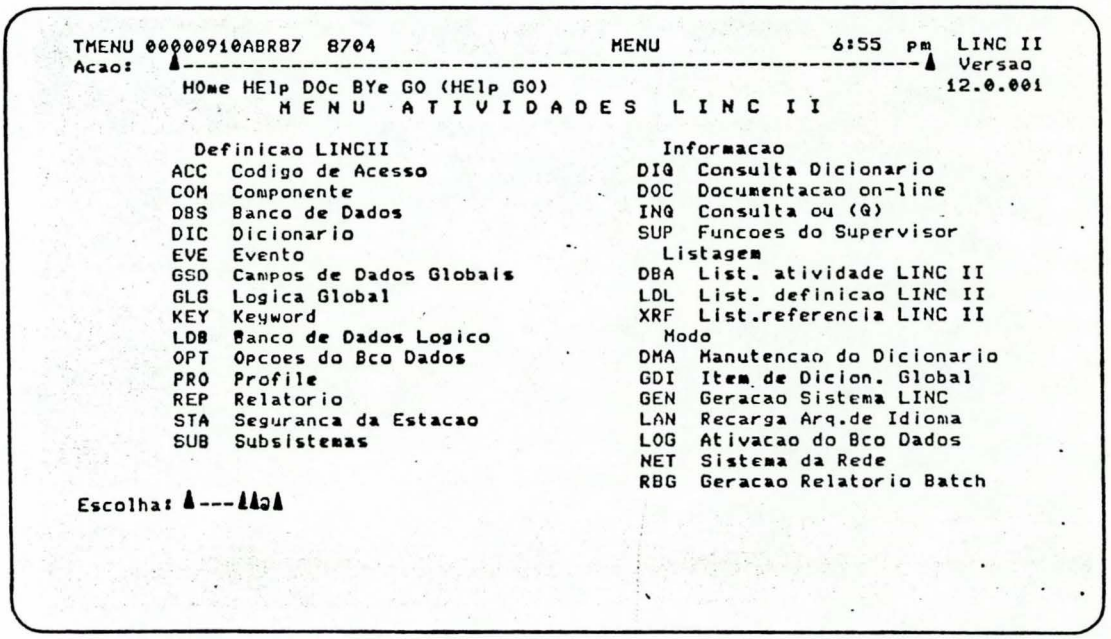


Fig. 4.2 - Tela de "menu" de atividades do LINC

Como principais facilidades incorporadas ao LINC, podem ser salientadas:

.desenho de telas/relatórios diretamente no vídeo do terminal;

.integração entre todos os módulos da ferramenta, ou seja, toda construção do sistema sendo realizada em um ambiente único de desenvolvimento;

.definição do banco de dados a partir da definição das telas;

.verificação sintática da especificação de forma on-line;

.programas fontes Cobol gerados de forma transparente ao usuário;

.manutenção só sobre a especificação;
.reorganizações transparentes ao usuário;
.facilidades de documentação on-line (para a ferramenta e associadas ao sistema aplicativo);
.mecanismos de controle de acesso à especificação e incorporáveis ao sistema aplicativo.

Para a presente avaliação, foi considerada a versão i3.0 do LINC.

4.1.2 Características da ferramenta

Como características do LINC, segundo os critérios estabelecidos na seção 3.2, podem ser relacionadas:

a) **Quanto aos critérios no âmbito da construção automática de sistemas:**

1) Perfil da especificação:

A especificação do sistema em LINC é feita através de parametrização nas diversas telas de seleção de opções e através da linguagem LDL. O suporte gráfico à especificação é muito limitado, restringindo-se basicamente a fase de modelagem do sistema com o CASE-LINC.

2) Especificação de semântica:

No dicionário de dados do LINC é possível associar-se uma pequena parcela de semântica aos dados como limites e lista de valores válidos e consistência de datas.

O LINC gera lógica automática para cruzamento de atributos entre estruturas (Lock.up automáticos) além de lógica automática para inclusão (ADD), alteração (CHG) e exclusão (DEL) de registro e de consultas a registro específico (REC), ao último (LAS), ao anterior (BAC), ao

próximo (NEX) e ao primeiro (FIR).

A semântica associada a cada estrutura do banco de dados (via tela de cadastro) e transação (via tela de evento) pode ser programada usando a linguagem LDL em três níveis de lógicas distintas:

.Pre-tela: qualquer lógica antes do LINC enviar a tela;

.Pre-Linc: lógica, como críticas e montagem de chaves, a ser executada antes da lógica automática do LINC;

.Principal: lógica a ser executada após a lógica automática do LINC, como seleção de itens a mostrar em consultas e atualizações manuais.

3)Construção do banco de dados:

A construção do banco de dados DMSII é totalmente transparente ao projetista do sistema sendo automaticamente gerado a partir da especificação das telas de cadastro (componente) do sistema e do volume de registros esperados em cada estrutura do banco de dados, especificado na própria interface da tela de cadastro.

4)Construção dos programas:

O LINC gera automaticamente um programa Cobol (Manager) para gerenciamento do tráfego multi-usuário de mensagens de terminais, um programa Cobol (Update) de manipulação das transações (consultas e atualizações) para cada subsistema especificado e um programa Cobol para cada relatório do sistema. A geração do fonte Cobol de todos estes programas é transparente para o usuário.

5)Validação da especificação e dos programas gerados:

As validações sintáticas sobre a especificação LINC são feitas imediatamente após a transmissão da linha/tela de definição. Opcionalmente pode ser indicado uma postergação da validação até a fase de geração do sistema. De qualquer forma, todos os erros sintáticos são eliminados antes da geração do fonte Cobol evitando, assim, correções sobre programas fonte em fase de compilação.

As validações semânticas, por sua vez, devem ser realizadas através de testes manuais sobre o sistema gerado.

6)Armazenamento das especificações:

Todas as especificações LINC são mantidas em um único banco de dados DMSII,próprio para esta finalidade, e associado ao compilador interativo.

7)Suporte ao projeto das E/S:

O LINC apresenta facilidades de desenho de telas/relatórios a partir do desenho direto no vídeo do terminal. O projetista desenha o layout da tela/relatório no vídeo indicando com pontos as posições a serem ocupadas pelos itens de dados. Após a transmissão da tela construída, o LINC vai solicitando o detalhamento de cada item de dado, um a um, colocando a área da tela prevista para o item em vídeo reverso e aguardando a intervenção do projetista para indicar o nome do item e, opcionalmente, as suas características de edição. No caso do desenho de relatórios, para aqueles que ocupam mais de 80 colunas, o LINC utiliza recursos de visualização com "janelas" sobre o layout para a completa especificação.

Os testes das rotinas de validação dos dados recebidos como entrada nos diversos atributos da tela devem

ser realizados após a geração do sistema mediante a sua execução.

8)Suporte à implantação:

O LINC é uma ferramenta própria para a abordagem de prototipação evolutiva onde a implantação do sistema é o resultado dos sucessivos refinamentos nos protótipos.

Uma prototipação para fins ilustrativos pode ser obtida de imediato pelo simples desenho das telas e ativação da geração do sistema. O LINC gera com isto, automaticamente, o banco de dados e os programas para manipulá-lo. As lógicas automáticas do LINC permitem uma rápida demonstração do funcionamento do futuro sistema.

9)Procedimentos de manutenção:

Toda e qualquer manutenção no sistema aplicativo gerado pelo LINC é feita sobre a especificação e jamais sobre o fonte Cobol gerado.

10)Aspectos ligados ao contexto da transformação de especificações:

O processo de transformação de especificação, convertendo definições LDL em fonte Cobol, é totalmente automático e transparente para o usuário (analista/programador).

O LINC mantém rotinas e esqueletos de rotinas pre-codificadas que são complementadas com a especificação LDL convertida para Cobol.

Na visão do analista/programador, o LINC possui procedimentos automáticos e outros que devem ser programados via LDL:

O LINC faz automaticamente (rotinas pre-codificadas):

- -validação das autorizações de acesso a nível de estrutura;
- -busca do registro pela chave primária (item Ordinate);
- -consistências básicas definidas no dicionário de dados (edição, lista de valores válidos, preenchimento obrigatório, etc);
- -consistências básicas de atualização (inclusão para já existente, alteração para inexistente, etc);
- -movimentações automáticas da tela para o BD (em inclusões e alterações) e do BD para a tela (em consultas) de todos os itens de dados associados ao registro em questão;
- -verificação da presença de registros, em outras estruturas do BD, que se vinculam ao registro corrente mediante uma identidade de nomes entre os itens de dados e onde o item se configura como um atributo chave na outra estrutura (Lock.up automático);
- -armazenamento do registro no BD em operações de inclusão, alteração e exclusão, neste último caso em vista da exclusão ser lógica.

Programado via LDL (especificação complementar):

- -consistências específicas do usuário com mensagens programadas;
- -pesquisas via sets do BD (Profiles) que não envolvam a chave primária na ordenação básica (comando DETERMINE) ou que envolvam chaves primárias em outras estruturas(comando LOCK.UP);
- -movimentações bi-direcionais (tela <--> BD) de itens de dados isolados (por exemplo, o caso de consultas que envolvam critérios de autorização para visualização de campo);
- -cálculos aritméticos;
- -atualizações cruzadas de itens e registros em

outras estruturas (comando FLAG, comando AUTO);
-troca de informações entre telas;
-ativação de Job's e programas emissores de relatórios;
-lógica de navegação entre telas;
-validações das autorizações de acesso a nível de registro e atributo;

Para a especificação complementar, programada via LDL, o LINC mantém algoritmos de transformação do LDL para comandos Cobol correspondentes. Existem esqueletos de programas/rotinas pre-codificadas que são complementadas com estes comandos oriundos da transformação.

Comandos SETUP.DATA, que criam variáveis locais de trabalho no LDL, são convertidos em variáveis de trabalho do Cobol. Demais comandos da lógica LDL também são transformados em correspondentes trechos de programa Cobol.

Exemplo:

Em LDL:

```
SETUP.DATA; USER-AL        ED;N LE;9
SETUP.DATA; NOME-AUX       ED;A LE;30
        .
        .
        .
MOVE; ESTRUT1.ITEM-USER   USER-AL
LOCK.UP; FROM USER-AL (ESTRUT2)
   DO.WHEN; ESTRUT2.MAINT NOT = (D)
      BREAK;
   END;
END;
```

Em Cobol:

```
01 GLB-TPHILIGHT.
      .
      .
   05 FIRST-TIME          PIC 99 COMP VALUE 0.
   88 FIRST-THRU          VALUE 0.
      .
      .
01 CONSTANT-TABLE.
      .
   05 REL1USER-AL         PIC 9(0009).
   05 REL1NOME-AUX        PIC X(0030).
      .
      .
SIST00001.
      MOVE ESTRUT1-ITEM-USER      TO REL1USER-AL.
      MOVE ZERO                   TO FIRST-TIME.
      ADD 1                       TO GLB-DMS-READS.
      MOVE SPACES                 TO GLB-STATUS.
SIST00002.
      IF FIRST-THRU
         FIND KEY OF ESTRUT2 VIA SET-ESTRUT2
                  AT CHAVE-ESTRUT2 = REL1USER-AL
              ON EXCEPTION
                 IF NOT DMSTATUS(NOTFOUND)
                    PERFORM DMSERROR
                 ELSE
                    NEXT SENTENCE.

      IF FIRST-THRU
         IF DMSTATUS(NOTFOUND)
            ADD 1    TO GLB-DMS-READS
            FIND KEY OF ESTRUT2 VIA NEXT SET-ESTRUT2
                 ON EXCEPTION
                    MOVE "*****" TO GLB-STATUS.

      IF NOT FIRST-THRU
```

```
            ADD 1    TO GLB-DMS-READS
            FIND KEY OF ESTRUT2 VIA NEXT SET-ESTRUT2
                 ON EXCEPTION
                    MOVE SPACES  TO GLB-STATUS.

         IF DMSTATUS(DMERROR)
            IF NOT DMSTATUS(NOTFOUND)
               PERFORM DMSERROR.

         IF DMSTATUS (NOTFOUND)
            IF NOT FIRST-THRU
               MOVE "X" TO GLB-ESTRUT2-PRIOR.

         MOVE 1 TO FIRST-TIME.
         IF DMSTATUS(NOTFOUND)
            GO TO SIST00003.
         MOVE SPACES TO GLB-ESTRUT2-PRIOR.
         MOVE "SET-ESTRUT2" TO GLB-ESTRUT2-PROFILE.
         MOVE 1 TO ESTRUT299.
         IF ESTRUT2-MAINT = "D"
            GO TO SIST00004.
         GO TO SIST00003.
      SIST00004.
            GO TO SIST00002.
      SIST00003.
```

b)Quanto aos critérios adicionais:

1)Escopo de aplicação:

O LINC permite desenvolver sistemas aplicativos tradicionais, para as mais variadas finalidades (folha de pagamento, controle de estoques, automação bancária, automação comercial, etc), essencialmente nas aplicações comerciais transacionais.

2)Interfaces de banco de dados:

A arquitetura do LINC está embasada em um SGBD DMSII mantendo interface navegacional a partir dos comandos de manipulação dos dados do banco de dados. Esta interface se materializa através da linguagem hospedeira Cobol com comandos DML (Data Manipulation Language) embutidos (As especificações LINC de acesso ao banco de dados são transformados em comandos DML do Cobol).

O LINC possui também interface SQL, não procedural, para consultas "AD HOC" em banco de dados gerados por ele.

3)Diálogo Homem-máquina:

O diálogo que o LINC mantém com o projetista é essencialmente através do uso de telas de "menu" onde as opções de especificação são selecionadas. Para o projeto de telas/relatórios, o analista e/ou programador tem a sua disposição um módulo de desenho onde a definição é feita através do desenho da tela diretamente no vídeo.

Com o uso da metodologia "Business Design", própria para a modelagem de sistemas a serem desenvolvidos em LINC, a ferramenta CASE-LINC permite, com o auxílio de um "mouse", uma manipulação direta no vídeo de diagramas representativos de componentes do sistema.

Para uma eventual especificação de lógica do sistema, o LINC oferece um editor próprio que manipula a especificação feita na linguagem LDL (Linc Definition Language).

4)Perfil do dicionário de dados:

Dentro do escopo do compilador interativo, o LINC apresenta um dicionário de dados centralizado para

definição e documentação descritiva dos itens de dados a serem manipulados pelos sistemas aplicativos. Quanto à abrangência deste dicionário de dados pode ser estabelecido um dicionário Global, compartilhado por todas as aplicações LINC, ou um dicionário Local, usado por apenas uma aplicação específica. Os itens de dados definidos no dicionário podem, a qualquer momento, ser solicitados pelo nome durante a elaboração de uma tela ou relatório.

Uma eventual alteração na definição dos itens de dados (dimensionamento, edição) tem reflexos diretos sobre toda a especificação, ativando um módulo de manutenção que faz os ajustes automáticos em layout de telas e relatórios.

A partir da versão 14.0, o LINC mantém interface com o ADDS (Advanced Data Dictionary System). Os sistemas podem utilizar diretamente itens de dados previamente definidos no ADDS sem necessitar redefinir tais itens no dicionário de dados do LINC II.

Para a fase de modelagem do sistema, a ferramenta CASE-LINC (disponível para equipamentos da família IBM-PC) mantém um dicionário de dados próprio, ainda não conectável ao compilador interativo LINC e ao seu dicionário.

5)Suporte à documentação do sistema:

A documentação do sistema aplicativo no LINC é feita de forma semi-automática. O projetista deverá complementar detalhes de documentação através de intervenções durante a especificação.

Em cada tela de especificação de estruturas e componentes do sistema aparece uma opção TX (Texto de comentário) onde o projetista pode associar uma documentação a ser direcionada para dois tipos de manuais:

.manual do operador
.manual do programador

Além disso, durante a especificação do sistema, pode ser definida uma documentação on-line de ajuda ao usuário associada a cada tela do sistema aplicativo (tela de ajuda acionada por um comando TEACH).

6)Gerenciamento do ambiente de desenvolvimento:

Abstraindo a fase de modelagem do sitema (ferramenta CASE-LINC), que ainda tende a ser incorporado ao ambiente LINC, as demais ferramentas (de projeto de BD, de projeto de E/S, de projeto de rede, etc) estão totalmente integradas em um ambiente único representado pelo compilador interativo LINC II.

7)Perfil da metodologia de análise e projeto:

Para a metodologia de análise e projeto associada a construção de sistemas via LINC, a UNISYS comercializa o "Business Design" apresentando, para suporte a esta metodologia, a ferramenta CASE-LINC. Esta ferramenta é utilizada na fase de modelagem do sistema e atualmente ainda não está completamente integrada à fase de desenvolvimento suprida pelo compilador interativo.

8)Definição de mecanismos de controle de acesso ao sistema:

No LINC, o controle de acesso ao sistema aplicativo se baseia em níveis de privilégio associados ao usuário e ao seu terminal em confronto com o especificado em cada tela do sistema. Conforme o nível de privilégio do usuário/terminal, o acesso é inibido automaticamente a determinada estrutura do banco de dados.

A autorização de acesso a nível de atributo e a nível de registro deve ser programado via LDL.

9)Suporte à recuperação/reconstrução do sistema:

Além dos mecanismos incorporados ao SGBD DMSII, o LINC oferece recursos próprios para administrar a recuperação e reconstrução em caso de falhas. Este mecanismo se baseia no registro e recuperação de informações de controle no arquivo LINCLOG.

Após a recuperação do DMSII, o LINC resubmete as transações pendentes registrados no arquivo LINCLOG.

Este procedimento é adotado para a ferramenta (compilador LINC) e para o sistema gerado.

10)Suporte à reorganização do sistema:

Todo e qualquer procedimento de reorganização do banco de dados é transparente aos usuários do LINC. Caso o compilador interativo detectar a necessidade de uma reorganização em virtude de alguma modificação expressiva na especificação, os procedimentos necessários para efetivar a reorganização serão automaticamente acionados.

4.2 SADS

4.2.1 Considerações preliminares

O SADS (Sistema de Automação do Desenvolvimento de Sistemas) é um software para construção de sistemas comercializado pela empresa mineira MSA INFOR [MSA 89].

Esta ferramenta procura automatizar e dar suporte às atividades de projeto de sistemas em suas fases de desenvolvimento e implantação. As funções do SADS são

executadas de forma on-line e em tempo real.

Os principais módulos do SADS são:

a)Dicionário de dados (SADS/DD):

O SADS utiliza o dicionário de dados como infra-estrutura principal onde os analistas armazenam todos os elementos lógicos (entidades, funções, atributos, documentos) e físicos (arquivos, telas, tabelas, programas, relatórios, estruturas de dados), de forma gradual.

b)Gerador de programas (SADS/GP):

Esta ferramenta gera programas fonte Cobol a partir de especificações cadastradas no dicionário de dados utilizando módulos pré-codificados, agregados em tempo de compilação, que executam funções como: comunicação com os monitores de TP, recuperação de BD e tratamento de exceções do DMSII. Utiliza lógicas padronizadas gerando comandos Cobol dentro de um padão único de codificação.

c)Gerenciador de tabelas (SADS/ST):

É um módulo de gerenciamento de tabelas, projetado para criar, manter e consultar tabelas de dados evitando que estas sejam definidas internamente nos programas. A definição e manutenção é feita de forma simples e interativa através do terminal.

d)Gerador de estruturas de dados (SADS/GD):

É um gerador de "libraries" em Cobol, para qualquer tipo de estrutura de dados e de comandos da linguagem DASDL para estruturas de banco de dados DMSII. A geração é executada a partir das definições lógicas das estruturas previamente registradas no dicionário de dados.

e)Gerador de telas (SADS/GT):

É uma ferramenta para geração automática de telas que tenciona tornar mais produtivas as atividades de desenho, teste e manutenção de telas para programas on-line. Permite criação e manutenção das telas de forma interativa via terminal eliminando a necessidade de codificá-las manualmente no fonte dos programas.

f)Consistência genérica de dados (SADS/CG):

Consistências de diferentes naturezas (formato, conteúdo, fechamentos, relacionamentos, testes lógicos, testes padrões, testes contra tabelas, etc) são efetuados a partir de parâmetros previamente cadastrados, elminando a necessidade de suas definições e codificação nos programas aplicativos.

Para a construção dos sistemas, o analista segue um roteiro de execução de tarefas, como mostra a figura 4.3, onde todos os componentes deste sistema são definidos mediante a ativação de módulos específicos para cada tarefa.

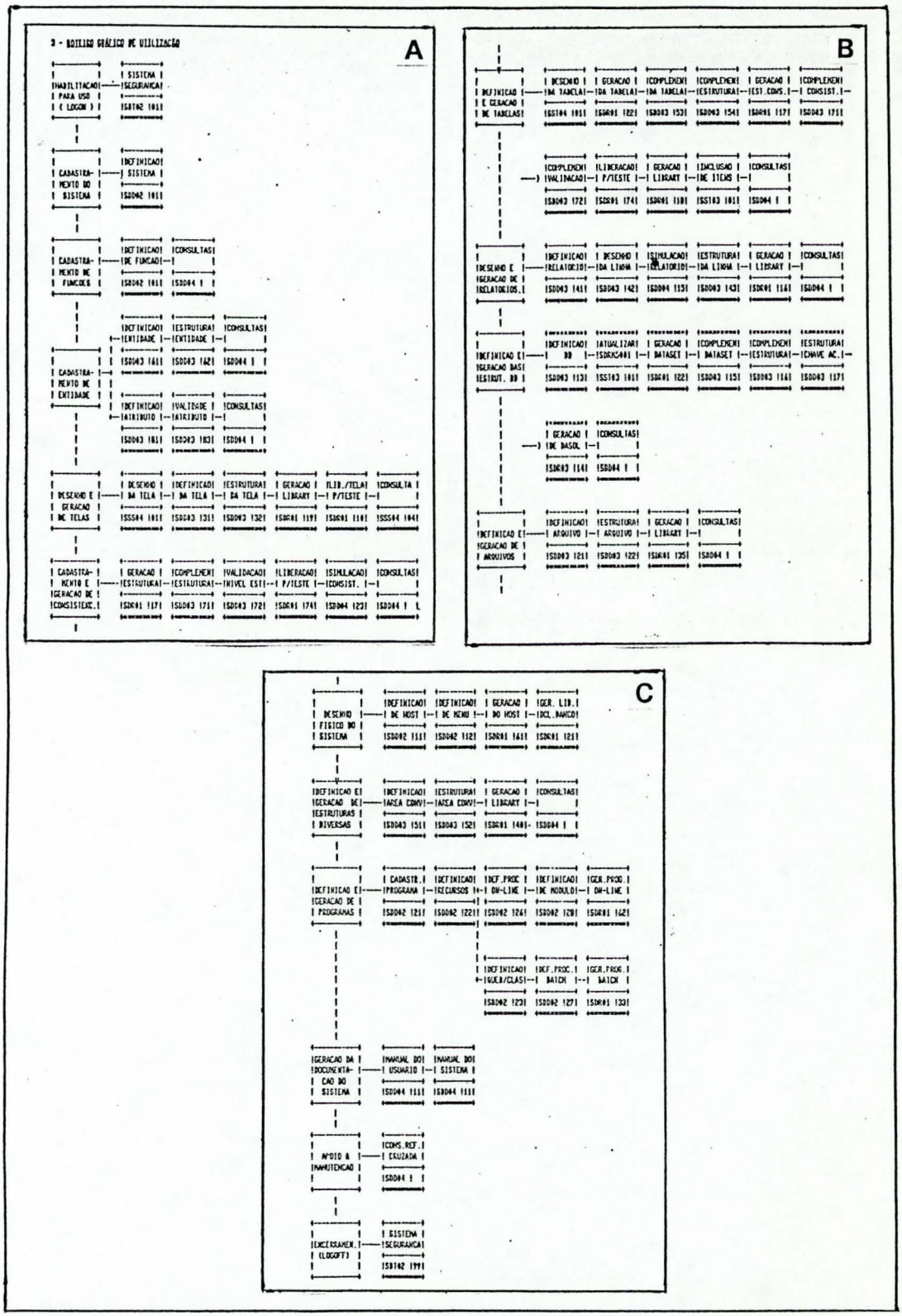


Fig. 4.3 - A construção de sistemas através do SADS (Extraído de [MSA 89])

A construção do sistema é feita de forma interativa a partir de um conjunto de telas de especificação sobre as quais o projetista realiza uma navegação, selecionando os itens de especificação entre as opções de definição existentes em cada uma delas. A figura 4.4 ilustra a tela de "menu" de atividades básicas do SADS.

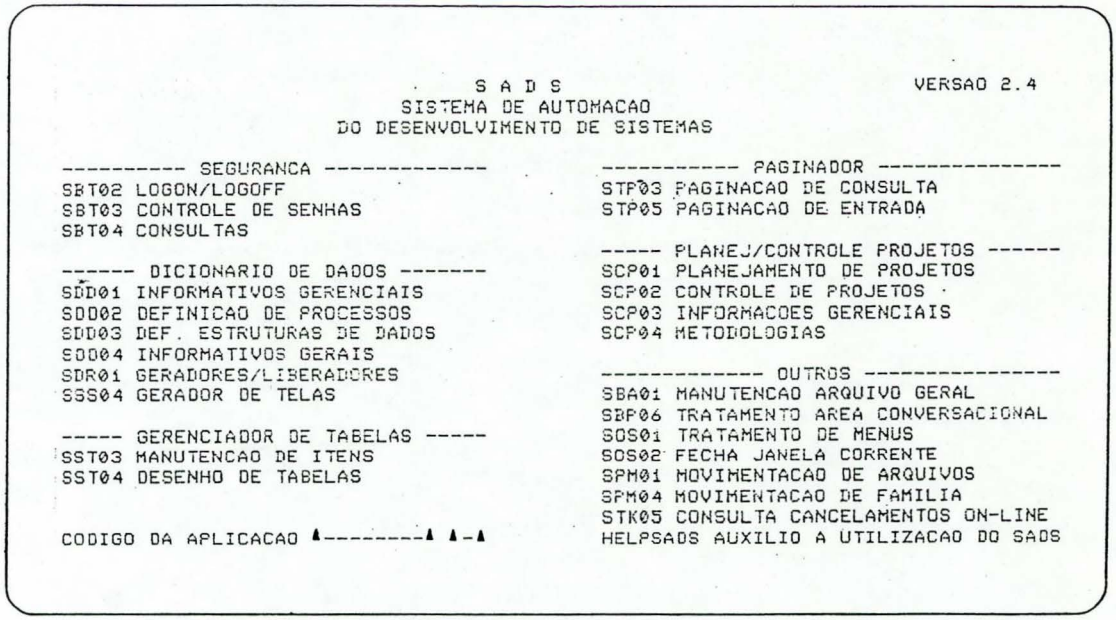


Fig. 4.4 - Tela de "menu" de atividades do SADS

Cabe ressaltar que um sistema aplicativo gerado através do SADS fica permanentemente sob a tutela desta ferramenta pois, para a sua execução, utiliza peças de software (consistências especificadas no dicionário, tabelas, etc) que compõem o ambiente de produção SADS.

Os detalhes constantes na presente análise se baseiam na versão 2.4 do SADS.

4.2.2 Características da ferramenta

Como características do SADS, segundo os critérios estabelecidos na seção 3.2, podem ser relacionadas:

a) Quanto aos critérios no âmbito da construção automática de sistemas:

1) Perfil da especificação:

A especificação do sistema em SADS é realizada principalmente com recursos de parametrização de definições nas diversas telas de seleção de opções. Não possui uma linguagem própria de especificação e não tem incorporado, em seu ambiente, nenhum editor de especificações. Em algumas telas de definições de lógica de programas permite colocar comandos escritos diretamente na linguagem Cobol. Fora do ambiente SADS, podem ser escritas rotinas (procedures) em fonte Cobol referenciáveis nas telas do SADS para definição de recursos de programas. Estas rotinas são posteriormente incorporadas no fonte Cobol gerado.

Suporte gráfico à especificação somente aparece na ferramenta CASE, ET-SADS, responsável pela fase de análise.

2) Especificação de semântica:

O dicionário de dados do SADS permite associar semântica aos dados. Neste dicionário são definidas variáveis (itens de dados) e os valores que podem assumir ,ou seja, atributos com suas tabelas de validade. Permite associar, também, subrotinas aos itens de dados para que estas possam fazer algum tratamento específico sobre o item.

No dicionário de dados é possível especificar validações automáticas de hora, data, dígito de controle

(módulo i0 e i1), nome da estação, etc. No âmbito dos itens da tela, permite especificar consistências parametrizadas a nível de tela e de sua estrutura (Ex.: SE item EQ var) com mensagens de erros correspondentes. Tem a possibilidade de estabelecer também, a nível do dicionário de dados, um confronto direto com tabelas internas do sistema.

O SADS gera lógicas padrões de inclusão, alteração, exclusão, consulta e impressão (geração de relatórios), construídas automaticamente a partir da especificação de parâmetros de cada programa. Estas lógicas podem ser complementadas através de módulos de definição de lógica do usuário (parametrizadas ou definidas fora do ambiente SADS como procedures Cobol).

3)Construção do banco de dados:

A definição do banco de dados é feita através de telas onde o DASDL (Data And Structure Definition Language) do DMSII é especificado, indicando os data-sets, a estrutura de cada data-set e os sets associados. É uma especificação do DASDL via telas de opções.

A partir desta definição é gerada uma "library" DASDL que deve ser compilada manualmente através de um "job" específico. A geração da "library" DASDL deve ser feita para cada data-set do banco de dados.

4)Construção dos programas:

Existe um controle manual de criação dos programas (quantos e quais).

Para cada tela de cadastramento, deve ser especificado um programa de inclusão, outro de alteração, um terceiro de exclusão e um quarto de consulta. Existem lógicas automaticamente geradas pelo SADS para estes programas mas, de qualquer forma, a indicação de quais

programas devem ser gerados é de responsabilidade do analista.

A definição do programa é feita nas seguintes etapas:

.cadastramento do programa com os dados de identificação.
.definição dos recursos usados pelo programa (data-sets, tabelas internas, telas, layout de relatórios, subrotinas, procedures e áreas de trabalho).
.definição de parâmetros referentes a execução on-line (transações por exemplo).
.lógica programada: tela com especificação direta de comandos Cobol parametrizados e sem uso de um editor e algumas funções de macros Cobol fixas que são referenciáveis durante a definição (rotinas de movimentação de dados, busca de registro no banco de dados, etc).

Opcionalmente, o analista pode gerar subrotinas externas em alguma linguagem de programação (Cobol, Algol, etc) e referenciá-las durante a definição do programa (recursos do programa). Pode também gerar rotinas (procedures) em fonte Cobol, codificadas fora do SADS, que irão fazer parte do programa e referenciá-las, igualmente, como recursos deste programa.

Deverá ser especificado, também, um programa Host que gerencia as atividades do sistema aplicativo onde é indicada a forma de navegar sobre as telas e os programas chamados.

5)Validação da especificação e dos programas gerados:

As validações sintáticas sobre a especificação SADS são feitas somente durante a compilação dos programas

Cobol gerados.

A semântica de telas pode ser validada através de simulações em tempo de especificação. Todos os demais testes, no entanto, devem ser realizados manualmente em tempo de execução.

6)Armazenamento das especificações:

Tudo que estiver no ambiente SADS, como dicionário, telas e parâmetros de programas, é mantido em um banco de especificações mantido pelo DMSII.

Procedures, subrotinas e alterações em fonte Cobol, no entanto, não são controladas pelo SADS e o armazenamento fica sob responsabilidade do analista.

7)Suporte ao projeto das E/S:

No SADS, as telas podem ser desenhadas diretamente no vídeo do terminal. O seu layout é desenhado por completo no terminal colocando uma máscara de edição, semelhante a "picture" do Cobol, na posição destinada a cada item de dado. Após o desenho, deve ser invocada uma outra tela do SADS para a definição da estrutura da tela sendo projetada, indicando todos os itens de dados que irão compô-la, numa correspondência com as máscaras de edição definidas durante o desenho do layout.

Uma vez definidos o layout (desenho) da tela e a sua estrutura, deve ser ativada a geração de uma "library" Cobol correspondente.

No projeto do relatório, por sua vez, a especificação é feita a nível de linhas com a definição sendo realizada linha após linha. A cada linha sendo definida, o SADS mostra a linha anterior. Não existe, durante a definição, uma visão completa do layout. Este

inconveniente é superado mediante o uso de um módulo de simulação do relatório que emite uma imagem de seu layout na impressora.

Os itens de dados que compõem as linhas do relatório também devem ser indicados em uma tela específica para definição da estrutura do relatório.

Para o projeto de uma tela, após a sua definição, pode ser ativado também um módulo de simulação. Nesta simulação podem ser testadas todas as consistências definidas no dicionário de dados para os itens de dados associados à tela e as consistências parametrizadas a nível da estrutura da tela.

8)Suporte à implantação:

A implantação de sistemas gerados pelo SADS depende de:

-especificação do DASDL do DMSII;
-desenho e simulação de execução das telas;
-construção dos programas;
-compilações manuais de banco de dados e programas;
-testes manuais.

Na atual versão do SADS não existem maiores recursos para uma prototipação que ilustre o manuseio do conjunto de telas e facilite as demonstrações de funcionamento do futuro sistema já nas fases preliminares do projeto. O SADS é uma ferramenta que continua muito dependente dos programas aplicativos. Existe somente a possibilidade de simular a execução de cada tela em particular onde é apresentado o seu layout e efetuadas as consistências definidas no dicionário de dados.

9)Procedimentos de manutenção:

Em determinadas situações torna-se necessário realizar manutenção sobre programas fonte Cobol em vista da impossibilidade de obter, a partir dos recursos do SADS, uma completa especificação de procedimentos exigidos pelo sistema em condições peculiares. Eventualmente, algum procedimento específico a ser incorporado em um programa aplicativo somente pode ser acrescentado mediante a codificação direta no fonte Cobol. Isto traz inconvenientes de segurança pois o controle de alterações e armazenamento destas alterações fogem da responsabilidade do SADS.

10)Aspectos ligados ao contexto da transformação de especificações:

No SADS, uma vez indicado o tipo de programa que deve ser gerado (INC-inclusão,ALT-alteração,EXC-exclusão, CON-consulta,IMP-emissão remota) e outros parâmetros (consistência geral, confirmação de transação, paginação, etc), o processo de transformação de especificação utiliza rotinas pre-codificadas correspondentes a parametrização fornecida.

Para cada tipo de programa, há uma série de rotinas prontas,ou montadas automaticamente a partir dos parâmetros, que são incorporadas no fonte Cobol gerado.

Exemplo: Programa de inclusão simples

Este programa utiliza como rotinas pre-codificadas ou de montagem automática:

-lê registro;
-lê tela;
-compõe cabecalho;
-consistência de área conversacional;
-consistência contra cadastro (verificação

da existência (ou não) do registro com a correspondente emissão de mensagens);

-atualização (cria, compõe e inclui registro);

-trata próxima transação.

Para complementar estas rotinas automáticas do SADS, o analista pode definir lógicas programadas em módulos pre-determinados para cada tipo de programa. Estas lógicas são especificadas através de uma parametrização de comandos escritos diretamente em Cobol ou de algumas funções macros do Cobol (também pre-codificadas).

Exemplo:

Na tela "Definição de módulo de processamento"

PROGRAMA {) MODULO {)

				ACESSOS			
SEQ	COMANDO	ORIGEM	DESTINO	SET/ULT	ATR	IND	N.V
01{0010)	{ALT)	{CPJT0103)	{CPJB01)	{A) {)		{)	{)
		{					)
02{0020)	{MVE)	{CPJB04)	{CPJT0110)	{) {)		{)	{)
		{					)
03{0030)	{MOV)	{NOMEFUN)	{FUNNOM)	{) {)		{)	{)
		{					)
04{0040)	{COB)	{)	{)	{) {)		{)	{)
		{IF SSC-QTDERRO) O					)
05{0050)	{COB)	{)	{)	{) {)		{)	{)
		{ MOVE GBL-TELAERROS TO WSS-MENSAGEM					)
06{0060)	{COB)	{)	{)	{) {)		{)	{)
		{ GO TO 99000-99-EXIT-PROGRAM.					)

Em Cobol:

"SEQ 010 - Lê registro do dataset CPJB01 , através do set CPJB01A, composto a partir dos campos da tela CPJT0103 de mesmo mnemônico. Move os campos correspondentes desta tela para o registro do dataset e, a seguir, o grava através do camando STORE."

```
MOVE JT03I-PR03I-PRJCOD  TO J01A-PRJCOD.

LOCK CPJB01A AT
     J0100-PRJCOD        = J01A-PRJCOD
   ON EXCEPTION
     GO TO 9800-00-EXCECAO-FATAL.

MOVE GBL-DATA            TO J0100-HDRDATA
MOVE GBL-HORA            TO J0100-HDRHORA
MOVE H-ESTACAO           TO J0100-HDRSTATGER

MOVE JT03I-PRJCOD        TO J0100-PRJCOD.

STORE CPJB01
   ON EXCEPTION
     GO TO 98000-00-EXCECAO-FATAL.
```

"SEQ 020 - Faz a movimentação de campos de mesmo nome do dataset CPJB04 para a tela CPJT0110."

"SEQ 030 - Faz a movimentação do campo NOMEFUN do dataset CPJB04 para o campo FUNNOM da tela CPJT0110."

```
MOVE J0400-FUNCOD        TO JT100-FUNCOD-RED.
MOVE J0400-UNDCOD        TO JT100-UNDCOD.
MOVE J0400-UNDNOM        TO JT100-UNDNOM.

MOVE J0400-NOMEFUN       TO JT100-FUNNOM.
```

"SEQ 040/060 - Transposicão direta dos comandos Cobol."

```
IF SSC-QTDERRO > 0
   MOVE GBL-TELAERROS TO WSS-MENSAGEM
   GO TO 99000-99-EXIT-PROGRAM.
```

Esta especificação parametrizada, em vista dos módulos pre-determinados para a sua definição, eventualmente não é auto-suficiente para a completa especificação do programa, exigindo codificação de trechos de programas externamente ao ambiente do SADS. Isto torna semi-automático o processo de transformação de especificação SADS-Cobol, condicionando-as a intervenções humanas em determinadas situações.

Outros recursos de programa, como subrotinas externas, procedures, layout de telas e relatórios, "libraries" de BD, etc, também são necessários para realizar o processo de transformação SADS-Cobol.

b)Quanto aos critérios adicionais:

1)Escopo de aplicação:

O SADS é um gerador de sistemas comerciais tradicionais caracterizando-se como uma ferramenta para propósitos gerais.

2)Interfaces de banco de dados:

O SADS, utilizando um SGBD DMSII para manter os dados e gerando programas em linguagem Cobol para a manipulação destes dados, estabelece uma interface navegacional sobre as estruturas do banco de dados através dos comandos DML (Data Manipulation Language) embutidos no

fonte dos programas Cobol.

3)Diálogo Homem-máquina:

O diálogo básico do SADS com o projetista do sistema é do tipo navegacional via telas de "menu" onde as opções de especificação, para os diversos componentes do sistema, são selecionadas.

Considerando o ET-SADS, como ferramenta CASE integrada ao ambiente de desenvolvimento SADS, é possível identificar também um diálogo homem-máquina do tipo manipulação direta com suporte gráfico onde a modelagem do sistema é feita, com o auxílio de um mouse, a partir de uma diagramação dos componentes do sistema segundo os conceitos da Análise Estruturada (AE) e da representação de dados via entidade/relacionamento (Diagramas E/R).

4)Perfil do dicionário de dados:

O SADS tem um dicionário próprio bastante poderoso em termos de associar semântica.

Abstraindo-se o dicionário do ET-SADS, que pode ser importado para dentro do ambiente SADS, o dicionário de dados se caracteriza por ser centralizado e integrado. Todas as aplicações desenvolvidas através do SADS utilizam um único dicionário.

Todas alterações em parâmetros de consistências para as rotinas de validações, incorporadas no dicionário de dados, tem reflexos imediatos no aplicativo, mesmo em tempo de execução. Sob este aspecto, o dicionário de dados do SADS pode ser visto como ativo, apesar de que quaisquer mudanças em termos de dimensionamento e edição dos itens de dados não tem repercussão automática sobre o restante da especificação (telas e relatórios, por exemplo) devendo ser feita uma revisão manual.

5)Suporte à documentação do sistema:

A documentação gerada pelo SADS é composta de:

.manual do sistema;
.manual do usuário.

Esta documentação é automaticamente gerada a partir da especificação (dicionário de dados, telas relatórios, programas, etc).

6)Gerenciamento do ambiente de desenvolvimento:

Para o projeto de um sistema, o SADS oferece um conjunto de ferramentas de especificação como dicionário de dados, DASDL do banco de dados DMSII, telas, tabelas, relatórios e programas. O SADS gera fonte Cobol para os programas, eventualmente incrementados manualmente, e fonte DASDL para o DMSII. Estes fontes, no entanto, devem ser compilados manualmente com os recursos de um ambiente extra SADS. Todo gerenciamento é manual e de responsabilidade do próprio analista.

7)Perfil da metodologia de análise e projeto:

Como metodologia de análise e projeto, o SADS considera os conceitos da Análise Estruturada para modelagem e representação da dinâmica do sistema e os diagramas entidade/relacionamento para representar a estrutura de dados associada a este sistema.

Como ferramenta CASE que suporta esta metodologia, a MSA-INFOR comercializa, um software para uso em microcomputadores da família IBM-PC, denominado ET-SADS. Este oferece todos os recursos gráficos de diagramação dos componentes do sistema através dos DFD's (diagramas de fluxo de dados) da Análise Estruturada e

diagramas E/R. O ET-SADS mantém também um dicionário de dados próprio que pode ser, posteriormente, exportado para o ambiente SADS.

8)Definição de mecanismos de controle de acesso ao sistema:

O SADS oferece um mecanismo de controle de acesso bastante completo. Permite definir, por exemplo:

.transações permitidas por aplicativo;
.associação usuário com aplicativos, terminais e horários que este usuário pode manipular;
.elementos do sistema autorizados para o usuário.

A abrangência do acesso depende do tipo de autorização fornecido ao usuário. Esta autorização é concedida através de:

.transações (programas) para a abrangência a nível de estrutura do banco de dados;
.transações e conteúdo de itens de dados para a abrangência a nível de registro do banco de dados;
.lógica programada ou transações para a abrangência a nível de atributo do banco de dados.

9)Suporte à recuperação/reconstrução do sistema:

Todo suporte à recuperação/reconstrução fica ao encargo dos mecanismos do SGBD DMSII. Opcionalmente, permite a possibilidade de manter uma recuperação sincronizada com o software COMS (Communications Management System) da UNISYS, caracterizando um procedimento de recuperação externo ao sistema.

10)Suporte à reorganização do sistema:

Todo o controle e execução dos procedimentos de reorganização (inclusive especificação do DASDL) são realizados manualmente sob a responsabilidade do próprio usuário ou setor de suporte.

4.3 Quadro sintético de comparação das características

Uma vez realizada a classificação de características das ferramentas de suporte à construção automática de sistemas e realizada uma avaliação das ferramentas LINC e SADS, dentro do contexto dos critérios estabelecidos na seção 3.2, a forma ideal de ilustrar um sumário de todas as conclusões estabelecidas desta análise é a de mostrar as características num "quadro sintético de comparação". As figuras 4.5a e 4.5b ilustram esta síntese.

CRITÉRIOS				ILINC	SADS
Perfil da especificação	Linguagem natural				
	Suporte gráfico à especificação			[illegible]	[illegible]
	Linguagem formal			X	
	Parametrização			X	X
	Asserções matemáticas				
Especificação de semântica	Associado aos dados no dicionário			X	X
	Tabelas de consistências				X
	Referência a macros/rotinas				X
	Programado			X	X
Construção do banco de dados	Manual com linguagem específica				X
	Automática via projeto das entradas			X	
Construção dos programas	Manual				
	Semi-automática				X
	Automática			X	
Validação da especificação e Programas	Verificação sintática	Imediata na especificação		X	
		Postergada para compilação			X
	Verificação semântica	Mecanismo de geração de testes			
		Execução simbólica			
		Mensuração de efeitos (simulação)			X
		Testes manuais		X	X
Armazenamento das especificações	Em banco de especificações			X	X
	Em dispositivos independentes por fase				X
Suporte ao projeto das entradas/saídas	Quanto à especificação	Codificação manual	Com ling. conv.de prog.		
			Com ling. específica		
		Facilidades de desenho		X	X
	Quanto à validação	Apresentação/simulação de execução			X
		Via programa aplicativo		X	X
Suporte à implantação	Testes de validação a cada etapa				X
	Prototipação			X	
Procedimentos de manutenção	Sobre a especificação			X	X
	Sobre o programa fonte				X
Transformação de especificação	Quanto à organiz./tipo de transform.	Uso de catálogos de regras			
		Uso de um conjunto gerador			
	Quanto à forma das regras de transformação	Algoritmos de transformação		X	X
		Regras de troca de padrões			
		Híbrido			
	Quanto à caracteris-ticas do processo	Automação	Manual		
			Semi-automático		X
			Automático	X	
		Casamento de representação	Primeira ordem		
			Segunda ordem		
		Avaliação das regras	Habilitação da regra		
			Estratégica		
			Asserções		
			Inform.escopo apl.regra		

Fig. 4.5a - Quadro sintético de comparação (critérios no âmbito da construção automática de sistemas)

CRITÉRIOS			ILINC	SADS
Escopo de aplicação	Propósitos específicos			
	Propósitos gerais		X	X
Interfaces com banco de dados	Programáveis		X	X
	Não procedurais		X	
	Gráficos			
	Especializados no tratamento de textos			
	Manipulação especializada			
	Auto-adaptativas			
Diálogo homem-máquina	Dirigido pelo computador			
	Navegacional		X	X
	Suporte gráfico		LIMITADO	LIMITADO
	Comandos de especificação	Linguagem natural		
		Linguagem formal	X	
		Linguagem de programação		X
Perfil do dicionário de dados	Integrado		X	X
	Isolado para cada fase do projeto			
	Ativo			LIMITADO
Suporte à documentação do sistema	Construção automática			X
	Construção semi-automática		X	
	Construção dirigida			
	Manual			
Gerenciamento do ambiente de desenvolvimento	Manual			X
	Integração do ambiente		X	
Perfil da metodologia	Estática e predefinida		X	X
	Configurada dinamicamente	Por adição de regras pelo usuário		
		Por derivação de novas regras		
Definição de mecanismos de controle de acesso	Quanto à especificação	Externo ao sistema		
		No dicionário de dados		
		Associado às estrut./comp. do sistema	X	
		Em tabelas/estruturas de controle		X
		Programado	X	X
	Quanto à abrangência (para especificações parametrizadas)	Nível de conteúdo de atributos		LIMITADO
		Nível de atributos		
		Nível de registros		X
		Nível de estruturas	X	
		Nível de transações		X
Suporte à recuperação/ reconstrução	Externo ao sistema			X
	Mecanismos do S.G.B.D		X	X
	Mecanismos próprios		X	
Suporte à reorganização	Controle e execução manual			X
	Transparente ao usuário		X	

Fig. 4.5b – Quadro sintético de comparação (critérios adicionais)

5. CONCLUSÕES

Conforme visto na seção 3.2, uma ferramenta pode ser qualificada submetendo-a a uma série de critérios. As alternativas para atender a determinado critério podem vir a ser bastante variadas e, eventualmente, a ferramenta adota uma solução híbrida que engloba mais de uma alternativa. Ocasionalmente uma possibilidade nova surge, em vista dos avanços tecnológicos na área de engenharia de software e áreas correlatas, e que não aparece enumerada entre as opções ligadas ao critério em análise.

Esta monografia não procurou esgotar totalmente o assunto, mas sim pre-estabelecer alguns critérios importantes na classificação de ferramentas visando, principalmente, o aspecto de construção automatizada dos sistemas.

Para situar o uso de ferramentas dentro do contexto da construção de sistemas de informações, foi realizado um rastreamento panorâmico sobre o ciclo de vida do software para os modelos: tradicional, prototipação e programação automática. A partir deste estudo foi possível determinar um elenco de características incorporadas ou incorporáveis nestas ferramentas.

Cabe relembrar que, quando se fala em ferramentas de suporte à construção automática de sistemas, a intenção é procurar os mecanismos que automatizem as tarefas tradicionais de analistas e programadores e, se possível, que sejam alimentados por uma especificação que se aproxime ao máximo de uma linguagem natural (de preferência aquela utilizada na comunicação com o usuário final). As formas de implementar estes mecanismos são as mais variadas e o fator humano (motivação e satisfação do projetista e facilidades de uso) é um dos aspectos que deve ser considerado na caracterização das ferramentas.

Uma peculiaridade destas ferramentas que está sendo muito enfatizada pelos profissionais de informática é o suporte gráfico com auxílio de "mouse" aplicável em diversas etapas do projeto de um sistema (modelagem dos componentes do sistema, projeto das E/S, alternativas de especificação de lógica como tabelas e árvores de decisão, etc).

Constatou-se também que, dentro do elenco de características desejáveis para uma ferramenta, figuram: 1) transparência em relação aos detalhes operacionais (construção dos programas, construção do banco de dados, armazenamento da especificação, etc); 2) manutenção exclusiva sobre a especificação; 3) mecanismos de mensuração dos efeitos (comportamento) da especificação ainda no ambiente da ferramenta; 4) integração do ambiente de desenvolvimento de sistemas; 5) especificação de semântica extra-lógica e 6) processo de transformação de especificação com a mínima intervenção humana.

Na intenção de gerar um amplo material como fonte de referência e modelo de avaliação dos resultados de futuras pesquisas no âmbito da programação automática e eventual implementação de novas ferramentas que dêem suporte a esta atividade, procurou-se mostrar, aqui, o horizonte de possibilidades que giram em torno das características que estas ferramentas podem incorporar. Apesar da avaliação, nos moldes dos padrões estabelecidos neste estudo, de duas ferramentas comercialmente disponíveis no mercado, a intenção não foi a de fixar critérios de escolha de ferramentas, mas sim a de validar os critérios de classificação definidos.

REFERÊNCIAS BIBLIOGRÁFICAS

[AHL 88] AHLERT, H., KORNDORFER, S.A.: Curso de LINC II, CPD-UFRGS, 1988.

[AHL 90a] AHLERT, H.: Estudo de metodologias e ferramentas que envolvam suporte à construção automática de sistemas, TI. n. 188 - CPGCC/UFRGS, mai 1990.

[AHL 90b] AHLERT, H.: Estudo sobre banco de dados e linguagens orientados a objetos, Exame de Qualificação I - CPGCC/UFRGS, ago 1990.

[AHL 90c] AHLERT, H.: Sistemas especialistas para a engenharia de software, Exame de Qualificação II - CPGCC/UFRGS, out 1990.

[BAL 81] BALZER, R., Transformational implementation: An example, IEEE Trans. Software Eng., vol. SE-7, pp. 3-14, Jan. 1981.

[BAL 82] BALZER, R., Goldman, N., Wile, D.: Operational specification as the basis for rapid prototyping, ACM Sigsoft Software Engineering Notes, vol. 7, no. 5, pp. 3-16, Dec 1982.

[BAL 83a] BALZER, R., CHEATHAM, T.E. Jr. & GREEN, C.: Software technology in the 1990,s using a new paradigm. Computer, vol 16, no. 11 pp. 39-45 (1983)

[BAL 85a] BALZER, R.: A 15 years perspective in automatic programming, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. vol.11, no.11, pp 1257-1267 (1985)

[BAR 82] BARSTOW, D.: Rapid prototyping, automatic programming and experimental sciences. ACM Sigsoft Software Engineering Notes, pp. 33-34, Dec. 1982

[BOE 88] BOEHM, B.W.: A spiral model of software development and enhancement. Computer, Los Alamitos, 21(5): 61-72, may 1988.

[CHE 84] CHEATHAM, T.E.Jr.: Reusability Through Program Transformation, IEEE Trans. on Soft. Engineering vol. SE-10, no. 5,Sept. 1984, pp. 589-594.

[COH 82] COHEN, D., SWARTOUT, W., BALZER, R.: Using symbolic execution to characterize behavior. ACM Sigsoft Software Engineering Notes, pp. 25-32, Dec 1982.

[FEA 82a] FEATHER, M. : Mappings for rapid prototyping, ACM Sigsoft Software Engineering Notes, pp. 17-24, Dec 1982.

[GUT 89] GUTIERREZ, M.R.G., DE SOUZA, O.J.C.: Prototipação em software e sua relação com outros modelos de desenvolvimento, CPGCC 89 (Trabalho submetido a SUCESU 89)

[HAR 89] HARTSON, H.R., HIX, D.: Human-computer interface development: concepts and systems for its management, ACM Computing Surveys, vol. 21, nro. 1, may 1989.

[HOR 85] HOROWITZ, E., KEMPER, A., NARASIMHAN, B.: A survey of aplication generators. IEEE Software, Los Alamitos, 2(1): 40-54, Jan 1985.

[KOR 87] KORNDORFER, S.A.: SGBDS com mais semântica no modelo de dados, Anais CIIS = Três Dias com DMSII, Angra dos Reis, Set 1987.

[LIA 87] LIANG, T.P.: User interface design for decision support systems: a self-adaptative approach, Information & management, vol. 12, nro. 4, apr 1987.

[MAR 83] MARTIN, J.: Manifesto: Presente e futuro da informática, Compucenter Ltda., 1983.

[MSA 89] MSA INFOR: SADS: Sistema de Automação do Desenvolvimento de sistemas, manual do usuário, ago 1989.

[NOG 87] NOGUEIRA, R., GARCIA, J.: Protótipos de sistemas de informação, I Simpósio sobre Produtividade no Desenvolvimento de Sistemas, 8-11 jun 1987, Brasília, pp. 1-18.

[OLI 87] OLIVEIRA, J.P.M., HOPPEN, N.: Sistemas de apoio à decisão para grupos: banco de dados generalizados e automação de escritórios, RP 74 - CPGCC , Ago 1987.

[PAR 83] PARTSCH, H., STEINBRUGGEN, R.: Program transformation systems, ACM Computing Survey, vol. 15, nro. 3, sep 1983, pp. 199-236

[RID 86] RIDDLE, W.E., WILLIAMS, L.G.: Software environments workshop report, ACM SIGSOEI = Software Engineering Notes, vol 11, no. 1, jan. 1986.

[UNI 88a] UNISYS Corporation, LINC II: Language Implementation Reference Manual, Feb. 1988.

[UNI 88b] UNISYS Corporation, LINC II: Operations Reference Manual, Feb. 1988.

[UNI 88c] UNISYS Corporation, LINC II: Administration and Operations Guide, Dec. 1988.

[WEI 89] WEITZEL, J.R., KERSCHBERG,L.: Developing Knowledge-based systems: reorganizing the system development life cicle, Communications of the ACM, apr 1989, vol. 32, nro. 4, pp. 482-488.