122488-6

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Sistema Gerenciador de Documentação de Projeto

por

Sandro Neves Soares

Dissertação submetida como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Flávio Rech Wagner
Orientador

Porto Alegre, maio de 1996.

SABi
UFRGS
05222170

UFRGS
INSTITUTO DE INFORMÁTICA
BIBLIOTECA

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Soares, Sandro Neves

Sistema Gerenciador de Documentação de Projeto / Sandro Neves Soares.—Porto Alegre: CPGCC da UFRGS, 1996.
89 p.: il.

Dissertação (mestrado)—Universidade Federal do Rio Grande do Sul. Instituto de Informática. Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, BR-RS, 1996.Wagner, Flávio Rech, orient.

1.Sistemas Digitais. 2.Frameworks. 3. Ambientes de Projeto. 4. Gerência de Projetos. 5. Integração de Ferramentas. 6. Controle Autático de Fluxos de Tarefas. I.Wagner,Flávio Rech, orient. II.Título

UFRGS
INSTITUTO DE INFORMÁTICA
BIBLIOTECA

N.º CHAMADA	N.º REG:
681.325.65(043) S676S	32692
	DATA: 19,11,96
ORIGEM: D / DATA: 21/10/96	PREÇO: R$ 29,00
FUNDO: II	FORN.: II

Sistemas digitais - SBU/II
Ambiente: Projeto
Gerência: Projeto
Documentação: Sistemas
Integração: Ferramentas
Electronic CAD Frameworks
CNPq 3.04 05 00-9

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor:Prof. Hélgio Trindade

Pró-Reitor de Pesquisa e pós-Graduação: Prof. Cláudio Scherer

Diretor do Instituto de Informática: Prof. Roberto Tom Price

Coordenador do CPGCC: Prof. Flávio Rech Wagner

Bibliotecária-Chefe do Instituto de Informática: Zita Prates de Oliveira

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus pelo vencimento de mais esta etapa.

Agradeço ao meu orientador, Flávio Rech Wagner, pelo conhecimento e experiência que me proporcionou nestes anos de trabalho em seus projetos. E ao colega Gleiber pelas dicas sempre importantes.

Agradeço ao Banco do Brasil pelo tempo que me ofereceu para a elaboração desta dissertação e ao CURV (Campus Universitário da Região dos Vinhedos, ligado à Universidade de Caxias do Sul - UCS) pelo constante incentivo.

E, por último, agradeço aos meus pais, Leo e Eunice, à vó Prenda, à minha namorada Juliana, aos companheiros do mundo cinófilo, Bill e Buena, e a todos os amigos, parentes e conhecidos, que, através do seu convívio, sempre possibilitaram a renovação de forças, indispensável para o atingimento deste objetivo.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Sistema de Biblioteca da UFRGS

32692

681.325.65(043)
S676S

INF
1996/122488-6
1996/11/19

MOD. 2.3.2

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A complexidade do projeto de sistemas eletrônicos, devido ao número de ferramentas envolvidas, ao grande volume de dados gerado e à natureza complicada destes dados, foi a causa principal do aparecimento, no final da década de 80, dos frameworks.

Frameworks são plataformas que suportam o desenvolvimento de ambientes de projeto e que têm, como objetivo principal, liberar os projetistas das tarefas acessórias dentro do processo de projeto (como, por exemplo, a gerência dos dados criados), possibilitando-lhes direcionar os esforços, exclusivamente, para a obtenção de melhores resultados, em menor tempo e a baixo custo. Para a realização deste objetivo, diversas técnicas são utilizadas na construção dos frameworks. Uma delas é conhecida como documentação dos passos de projeto.

A documentação dos passos de projeto é um recurso utilizado para manter a história do projeto (usualmente, ferramentas executadas e dados gerados). Ela tem sido amplamente utilizada em trabalhos relacionados a frameworks. Porém, nenhum destes trabalhos aproveita toda a potencialidade do recurso. Alguns utilizam-no apenas nos serviços relacionados à gerência de dados. Outros, utilizam-no apenas nos serviços relacionados à gerência de projeto.

A proposta deste trabalho, então, é a criação de um sistema que explore toda a potencialidade da documentação dos passos de projeto, disponibilizando, a partir daí, informações e serviços a outros sub-sistemas do framework, de forma a complementar a funcionalidade destes, tornando-os mais abrangentes e poderosos.

PALAVRAS-CHAVE: Frameworks, Gerência de Projeto, Gerência de Dados, Documentação dos Passos de Projeto.

TITLE:"A DESIGN TRACE MANAGEMENT SYSTEM"

ABSTRACT

The VLSI design complexity, due to the number of involved tools, the enormous generated data volume and the complex nature of the data, was the main cause of the appearance of the frameworks in the end of the 80's.

Frameworks are platforms that support the development of design environments and, as their main purpose, liberate the VLSI designers from the supplementary tasks in the design process, as the data management. It makes possible to direct efforts exclusively to obtaining better results, in shorter time and with lower costs. To this purpose, many techniques have been used in the implementation of frameworks. One of these techniques is known as design steps documentation.

The design steps documentation is a resource used to keep the design history (usually, executed tools and generated data). It has been widely used in various frameworks. But none of them take full advantage of this resource. Some of them use the design steps documentation only in the data management services. Others, use it only in the design management services.

So, the proposal of this work is to create a system that takes full advantage of the design steps documentation, providing information and services to other sub-systems of the framework to complement their functionality, making them more powerful.

KEYWORDS: Frameworks, Design Management, Data Management, Design Steps Documentation.

1 INTRODUÇÃO

Os frameworks começaram a surgir no final da década de 80, devido à necessidade de se obter, nos projetos de sistemas eletrônicos, resultados de maior qualidade, a baixo custo e em tempo reduzido. São plataformas que provêem um ambiente operacional comum a ferramentas de projeto (os programas utilizados), disponibilizando facilidades para a manutenção e o gerenciamento dos dados criados (ou objetos de projeto), e conhecimento a respeito do processo de projeto, sendo capazes, por exemplo, de conduzir o projetista na seqüência correta de execução de ferramentas dados os seus objetivos.

Um framework não é estático, ou seja, não se restringe a um conjunto de ferramentas particular. Novos programas para o desenvolvimento de sistemas eletrônicos podem ser incorporados a qualquer momento. As vantagens do uso de frameworks vão desde o aumento da produtividade global do projeto até a preservação de investimentos.

A gerência de dados e a gerência de projeto (ou gerência do processo de projeto) são as áreas mais desafiadoras e complexas quando da definição e implementação de um framework. A maioria dos serviços, disponibilizados por estas plataformas, ou fazem parte da gerência de dados, ou fazem parte da gerência de projeto. Assim, as políticas de controle de acesso aos dados e o gerenciamento de versões e configurações constituem serviços da gerência de dados (no capítulo 2, detalhamos os conceitos de versão e configuração). Enquanto que o controle de acesso às ferramentas e do fluxo de execução destas, bem como a administração do trabalho em equipe são serviços da gerência de projeto.

Chama-nos a atenção o fato de uma técnica, utilizada em diversos trabalhos relacionados a frameworks, e que chamaremos aqui de *documentação dos passos de projeto*, apresentar soluções satisfatórias a problemas tanto de gerência de dados quanto de gerência de projeto.

A documentação dos passos de projeto é um recurso usado para a manutenção de um histórico de tudo o que acontece no desenrolar do processo de projeto: quais as ferramentas utilizadas, quando, por quem, em que ordem, as opções usadas, quais os dados de entrada, quais os dados de saída, entre outros. Esta documentação é aproveitada para:

- que os projetistas possam acessar versões e configurações de objetos (navegando em browsers gráficos que se utilizam das informações na documentação, por exemplo);
- o controle do fluxo de execução de tarefas. Projetistas inexperientes poderiam fazer uso de seqüências que já foram usadas com sucesso por outros mais experientes;
- a exploração do espaço de projeto, que vem a ser um retorno a um estado anterior e, a partir daí, o uso de um conjunto alternativo de recursos (ferramentas, dados e opções) para a obtenção de resultados, quem sabe, mais próximos dos desejados;
- o estudo, por parte dos construtores de ambientes de projeto, para efetuar melhorias no sistema. Por exemplo, uma ferramenta que nunca aparece na documentação, pode ser interessante retirá-la do ambiente;
- fornecer dados sobre execução de programas a algoritmos de distribuição de carga, que escolhem o melhor ponto (naquele dado instante) de uma rede para a execução de uma determinada ferramenta.

No entanto, o que observamos é que os diversos trabalhos, que incorporam a documentação dos passos de projeto, utilizam-se apenas de forma parcial das informações, potencialmente, disponíveis. Alguns utilizam-se destas informações apenas nos serviços relacionados à gerência de dados. Outros, nos serviços relacionados à gerência de projeto. Outros, ainda, numa mistura destes serviços. Nenhum deles, porém, tira proveito total das potencialidades do recurso.

Sendo assim, propomos um sistema gerenciador da documentação dos passos de projeto, independente dentro de um framework e fornecedor de informações e serviços para outros sistemas, que podem ser os diferentes serviços do framework, ou sistemas independentes que propõem soluções para a gerência de dados ou gerência de projeto e que não são parte integrante de um framework. Como exemplos destes outros sistemas, podemos listar (os três primeiros constituem serviços de um framework):

- gerenciadores de versões de objetos de projeto;
- gerenciadores de fluxo de execução de tarefas;
- distribuidores de carga de processamento numa rede;
- o Gerenciador Automático para Projetos de Sistemas Digitais [ROY 95], também desenvolvido no CPGCC da UFRGS;
- os sistemas ADAM Planning Engine e Cadweld, que, entre outros, serão descritos e analisados no capítulo 3.

O objetivo deste sistema é a complementação da funcionalidade destes outros sistemas, no sentido de torná-los mais abrangentes e poderosos. Por exemplo, um sistema específico para controle do fluxo de execução de tarefas poderia utilizá-lo para o desempenho dos serviços de gerência de dados.

Como forma de demonstrar, na prática, o funcionamento deste sistema gerenciador da documentação dos passos de projeto (que chamaremos, a partir de agora, de SGDP, Sistema Gerenciador de Documentação de Projeto), vamos utilizá-lo:

- para consultar as relações entre os dados gerados no decorrer do processo de projeto, através de um interface gráfica, onde o usuário poderá navegar à procura de um determinado objeto ou daqueles que o originaram;
- para acessar versões anteriores de objetos de projeto. O Gerenciador Automático para Projetos de Sistemas Digitais [ROY 95] fará uso deste serviço fornecido pelo SGDP;

- para registrar fluxos de execução de ferramentas utilizados para a obtenção exitosa de um determinado resultado, e possibilitar a gravação, em arquivo, destes fluxos. Tais arquivos vão constituir uma biblioteca para futuras consultas;
- para prover, a sistemas de controle do fluxo de execução de tarefas, novas seqüências de execução comprovadamente capazes de apresentar os resultados esperados, tendo como base a biblioteca de arquivos mencionada no item anterior;
- para a manutenção do estado do projeto, possibilitando o retorno a pontos anteriores para a reexecução de ferramentas.

Com relação ao algoritmo de distribuição de carga e às informações para estudo pelo construtor de ambiente (que mencionamos, anteriormente, na conceituação da documentação dos passos de projeto), vamos apenas manter os dados necessários na documentação, sem, num primeiro momento, utilizá-las na prática.

O presente trabalho está dividido da seguinte forma: no segundo capítulo, detalhamos frameworks, gerência de dados e gerência de projeto, buscando inserir os objetivos do trabalho neste contexto.

No terceiro capítulo, descrevemos e estudamos detidamente diversos sistemas que fazem uso da documentação dos passos de projeto, com o intuito de encontrar os benefícios da sua utilização e possíveis deficiências a serem corrigidas.

No quarto capítulo, apresentamos toda a funcionalidade do sistema proposto e os avanços com relação àqueles sistemas descritos no terceiro capítulo.

No quinto capítulo, descrevemos as aplicações que foram construídas para, em conjunto com o SGDP, constituírem um ambiente de projeto completo com serviços consistentes de gerência de dados e gerência de projeto, e a utilização do SGDP pelo Gerenciador Automático para Projetos de Sistemas Digitais.

No sexto capítulo, abrangemos alguns detalhes de implementação.

E no sétimo e último capítulo, apresentamos as conclusões e discutimos algumas sugestões para possíveis implementações futuras.

Em anexo, listamos os protótipos das funções, a definição das estruturas e o formato dos arquivos, que são necessários para a utilização do SGDP.

2 FRAMEWORKS

O objetivo deste capítulo é proporcionar uma visão abrangente da área de projeto automatizado de sistemas eletrônicos, mais especificamente da parte relacionada a frameworks.

Iniciamos apresentando o contexto no qual surgiram os frameworks. A seguir, definimos o que são e que serviços disponibilizam. Por último, descrevemos as duas áreas mais destacadas dentro deste assunto: a gerência de dados e a gerência de projeto.

2.1 Projeto de sistemas eletrônicos

O projeto de um sistema eletrônico [WAG 94] deve resultar em informações de duas naturezas que são essencias para o processo de manufatura. Informações estruturais descrevem quais componentes (ou células, em um circuito integrado) serão utilizados e quais interconexões devem existir entre eles. Informações geométricas, no caso de uma implementação com componentes discretos, descrevem o posicionamento dos componentes sobre uma placa de circuito impresso e o roteamento das interconexões entre os seus pinos. No caso de um circuito integrado, as informações geométricas descrevem o posicionamento das células na superfície de silício, o layout interno de cada célula e o roteamento das interconexões entre os terminais das células.

A obtenção destas informações não é uma tarefa trivial quando se projeta um sistema complexo. Numa perspectiva geral, trata-se o problema da complexidade baseando-se em abstrações de duas naturezas. Em primeiro lugar, o projetista deve considerar múltiplos níveis de projeto. Um nível de projeto mais abstrato permite ao projetista lidar com um número de componentes menor do que aquele realmente existente no sistema físico. Uma simplificação adicional importante pode ser obtida se as informações estru-

turais e geométricas forem abstraídas, de modo que o projeto seja realizado em termos do comportamento do sistema, que pode ser descrito por exemplo de forma algorítmica.

O projeto de um sistema eletrônico complexo é realizado portanto ao longo de três eixos de projeto, considerados ortogonais entre si: os eixos estrutural, geométrico e comportamental. Cada eixo contém vários níveis de projeto, representando diferentes graus de abstração em relação ao sistema físico. Esta situação pode ser apreciada através do diagrama Y de Gajski e Kuhn [GAJ 92] na Figura 2.1. A origem dos eixos representa o sistema físico, enquanto círculos concêntricos identificam pontos de projeto em diferentes eixos que correspondem a um mesmo nível de projeto. Pontos de projeto mais distantes da origem correspondem a representações mais abstratas.

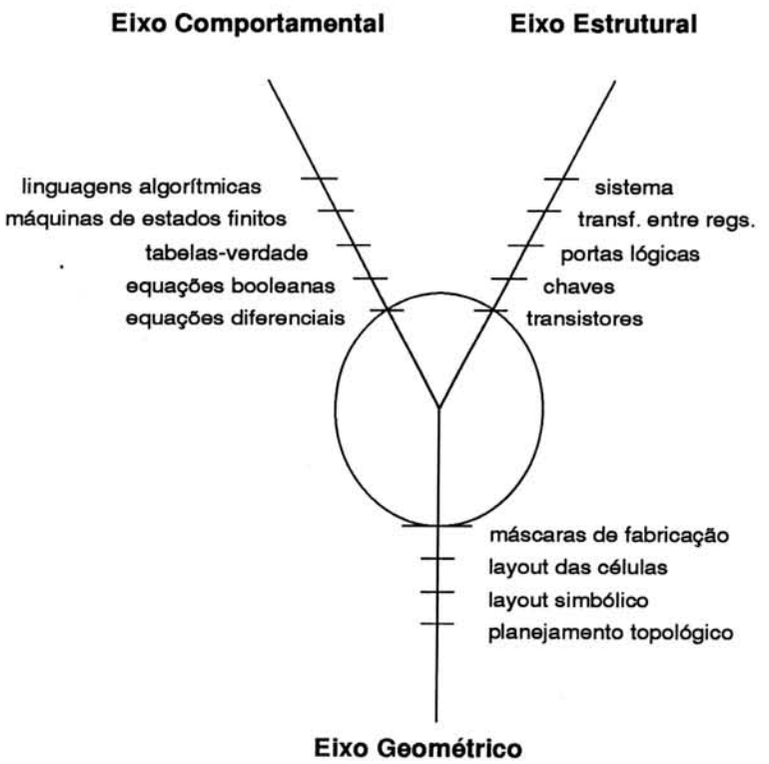


FIGURA 2.1- Diagrama Y de Gajski e Kuhn

2.2 Processo de Projeto de sistemas eletrônicos

O processo de projeto de sistemas eletrônicos [WAG 94] pode ser pensado como uma seqüência de transformações envolvendo os eixos e níveis de projeto. Numa metodologia típica, o projeto procede numa combinação das abordagens *top-down* e *bottom-up*. A partir de uma especificação comportamental abstrata inicial, o projetista progressivamente agrega informações estruturais e geométricas (abordagem *top-down*), até chegar a uma descrição do sistema que contém componentes que podem ser encontrados numa biblioteca, cujos elementos foram projetados numa abordagem *bottom-up*.

2.3 Características do processo de projeto de sistemas eletrônicos

O processo de projeto de sistemas eletrônicos possui algumas características marcantes, que devem ser conhecidas para que se possa entender os esforços empreendidos no sentido de torná-lo cada vez mais eficiente.

O processo de projeto trata, simultaneamente, inúmeros objetos de projeto (preliminarmente, podemos afirmar que objetos de projeto são os arquivos de dados manipulados pelas ferramentas) e cria múltiplas representações para cada objeto, uma por nível de projeto sob o qual este foi tratado. Sendo assim, um mesmo objeto pode estar representado, entre outras, por uma linguagem algorítmica, por uma descrição com portas lógicas e através de uma descrição com transistores. Para cada uma destas representações, existe um conjunto de ferramentas (os programas utilizados) específicas que a manipulam.

Adiante, quando se procede à exploração do espaço de projeto, ou seja, quando, a partir de uma dada representação existente, se quer chegar a um resultado diferente, com características bem específicas (com restrições de área ou de consumo, por exemplo), deve-se utilizar ferramentas que contemplem outras tecnologias ou arquiteturas e vai se estar criando mais representações para o objeto (agora, várias representações num mesmo

nível de projeto. Por exemplo, duas descrições comportamentais alternativas para um mesmo objeto). Isto implica a geração de grandes volumes de dados e uma quantidade significativa de ferramentas envolvidas.

É interessante, então, que se divida o trabalho em equipes. Cada equipe, por exemplo, ficará responsável por um módulo, que poderá, ainda, ser dividido em sub-módulos, que ficarão a cargo dos diversos projetistas da equipe. No decorrer dos seus trabalhos, os projetistas e as equipes deverão cooperar, requerendo e disponibilizando resultados, bem como integrando soluções.

As características principais do processo de projeto de sistemas eletrônicos são, portanto:

- envolve até milhares de passos, onde um passo constitui-se na execução de uma ferramenta;
- cria enormes volumes de dados;
- deve permitir a exploração do espaço de projeto, o que contribui para os dois primeiros ítens;
- o trabalho pode ser dividido em equipes, com a introdução da modularização e da hierarquização como forma de se dominar a complexidade.

2.4 Ambientes usuais

Descrevemos, aqui, os ambientes usuais de projeto para enfocar aspectos que impactam, negativamente, o processo de projeto.

Em ambientes usuais de projeto, as ferramentas são executadas na linha de comandos do sistema operacional, sendo os seus dados de entrada e de saída arquivos gerenciados por este mesmo sistema.

O construtor do ambiente de projeto monta-o instalando as ferramentas (normalmente de procedências diversas: próprias ou de outros fornecedores) em diferentes diretórios e em diferentes nodos de uma rede. Ele, então, disponibiliza *scripts*, que preparam o ambiente para a sua utilização pelos projetistas.

Inicia-se o projeto, e o primeiro problema com o qual um projetista vai se deparar é o fato de que cada ferramenta propõe uma interface diferente: linha de comando, janelas gráficas, janelas texto, etc. Uma opção *Gerar*, disponível em ferramentas diferentes, apesar da mesma sintaxe, pode ter ações totalmente opostas, apenas para exemplificar. Mas, mesmo com uma **interface não homogênea**, após um certo tempo (não muito curto), o projetista se familiariza com as diversas formas de interação disponíveis.

E agora? Qual se vai utilizar primeiro? E na seqüência? Qual vai lhe dar a melhor resposta dados os objetivos e as restrições propostas? O próprio projetista deve ter em mente o conjunto de ferramentas que vai utilizar e em qual seqüência, correndo o risco de inconsistências. Este tipo de conhecimento, chamamos gerência de projeto. Os ambientes usuais, então, **não disponibilizam recursos de gerência de projeto**.

Dominando a seqüência de ferramentas a ser utilizada, o projetista vai ter que desempenhar ainda uma tarefa redundante, que não acrescenta nada ao projeto em si. No momento que o resultado de uma ferramenta deve ser tratado por outra, provavelmente será necessário um conversor de formatos, uma vez que cada ferramenta, pelo fato de possuírem procedências diversas, exige um formato próprio dos dados de entrada, bem como gera suas saídas num formato específico. **Não há, portanto, um modelo uniforme de representação de dados**.

Contornados os problemas das diferentes interfaces, da gerência de projeto e da conversão dos dados, o projetista começa o trabalho propriamente dito. Gera resultados intermediários e finais (cada um constituindo um arquivo no sistema de arquivos). Testa alternativas e gera novos resultados. Logo, o volume de dados com o qual está tratando cresce a níveis impossíveis de serem gerenciados. Por exemplo, ficará difícil saber quais arquivos constituem os sub-módulos de um sistema. Ou qual é a versão dos sub-módulos

que deve ser utilizada? A última? A que melhor responde a um conjunto de restrições? Fora isso, surge também o problema de inconsistência ou duplicação das informações. Normalmente, existe o armazenamento dos mesmos dados ou dados correlatos em arquivos diferentes. Este é o caso, por exemplo, da representação de sinais de interface de um certo módulo. A alteração de atributos destes sinais, ou mesmo sua remoção ou inclusão, deveria ser feita de forma consistente em todos os níveis de abstração nos quais o módulo está representado.

Se pensarmos, ainda, que o projeto está sendo executado por uma equipe, estes **problemas de gerenciamento dos dados** assumem proporções gigantescas, sendo praticamente impossível o seu tratamento.

E, para fechar, o ambiente não é estático. Novas tecnologias e facilidades surgem a cada dia. O construtor de ambiente deve ficar atento a estes detalhes e incorporar novas ferramentas ao ambiente quando julgar necessário. Para o projetista, todos os problemas anteriores repetir-se-ão a cada incremento no ambiente de projeto.

Nos dias atuais, não se pode imaginar um ambiente com tais problemas. O projeto deve ser implementado da melhor forma possível, no menor tempo possível e com os menores gastos. Assim, atualmente, os ambientes usuais de projeto estão totalmente ultrapassados.

2.5 O que se espera de um ambiente de projeto?

As funcionalidades esperadas de um ambiente de projeto são exatamente aquelas que possam eliminar os problemas verificados nos ambientes usuais.

Um ambiente de projeto deve:

- disponibilizar uma interface homogênea aos usuários;

- prover uma estrutura interna única para a representação dos objetos, dispensando o uso de conversores;
- prover recursos para a gerência de projeto, ou seja, guiar o projetista no seu trabalho informando-lhe a melhor seqüência de ferramentas dados os seus objetivos e restrições;
- tratar a possibilidade de inconsistência de dados armazenados em diferentes representações do mesmo objeto;
- gerenciar o volume de dados gerado, controlar o acesso a estes dados e tratar as relações entre eles;
- gerenciar a cooperação entre as diversas equipes e projetistas;
- possibilitar o trabalho em rede: distribuição dos dados e acesso transparente às ferramentas.

Um ambiente de projeto que disponibiliza os serviços acima listados aos seus usuários é dito um ambiente integrado. Mas, além de integrado, o ambiente deve ser, também, aberto. Ambiente aberto é aquele:

- que permite a integração de novas ferramentas;
- que provê recursos para a comunicação entre ferramentas, para a construção/extensão da interface com o usuário;
- que provê uma interface procedural de acesso aos recursos de gerência de dados e de projeto, bem como à base de dados;
- usa padrões de rede, sistemas operacionais, interface, transferência de dados.

2.6 Frameworks

Dentro deste contexto de se buscar desenvolver sistemas cada vez melhores, em tempo abreviado e com baixo custo, substituindo os problemas dos ambientes usuais pelos benefícios de ambientes integrados e abertos, começaram a surgir, no final da década de 80, os denominados frameworks. Frameworks são plataformas para o desenvolvimento de ambientes integrados e abertos de projeto de sistemas eletrônicos [WAG 94]. Segundo o CFI (CAD Framework Initiative, que é um consórcio de empresas e instituições de pesquisa para a criação de padrões na área de frameworks), é uma estrutura de software que provê um ambiente operacional comum a ferramentas de projeto. Um framework não é um ambiente particular para uma determinada aplicação. Ele deve compreender uma coleção de serviços de propósitos gerais que podem ser configurados de maneira ótima para um largo espectro de aplicações, devendo poder adequar-se a qualquer conjunto de ferramentas de projeto.

Um framework permite aos construtores de ambientes de projeto:

- integrar novas ferramentas ao ambiente;
- definir processos e metodologias de projeto; e
- definir políticas de uso de ferramentas e dados.

E, aos projetistas, usuários finais dos ambientes, permite:

- disparar e gerenciar a execução de ferramentas;
- criar, organizar e gerenciar dados;
- visualizar (graficamente) todo o processo de projeto; e
- realizar tarefas diversas de gerência de projeto, inclusive relativas ao trabalho em equipe.

Nos frameworks, a gerência de dados e a gerência de projeto são, com certeza, os temas mais complexos e fascinantes para os pesquisadores. Isto justifica o grande número de soluções apresentadas em trabalhos submetidos a congressos da área. Vamos, a seguir, detalhar um pouco mais os aspectos envolvidos na gerência de dados e na gerência de projeto.

2.7 Gerência de dados

Já sabemos que, num processo de projeto de sistemas eletrônicos, o volume de dados gerado é enorme. Torna-se necessário, então, que o framework disponibilize facilidades para o tratamento destes dados. Ao conjunto destas facilidades, atribuímos o nome de gerência de dados.

O controle de versões, o controle de configurações e o controle de acesso aos dados são os serviços mais importantes da parte de gerência de dados em frameworks. Detalharemos cada um a seguir, dando maior importância ao controle de versões por constituir um dos objetivos deste trabalho.

2.7.1 Controle de versões

No desenrolar de um projeto, são criadas múltiplas representações para um mesmo objeto de projeto. Denominamos vistas aquelas representações em diferentes níveis de projeto [WAG 94]. E, alternativas aquelas num mesmo nível de projeto. Por exemplo, uma descrição estrutural VHDL (VHDL é uma linguagem de descrição de hardware amplamente difundida [IEE 88]) derivada de uma descrição comportamental VHDL constituem, ambas, vistas de um objeto, enquanto que duas descrições estruturais VHDL distintas, derivadas de uma única descrição comportamental VHDL, constituem alternativas de um objeto.

O termo versão é utilizado para designar uma generalização dos conceitos de vista e alternativa, enquanto que objeto de projeto é uma generalização do conceito de versão.

O controle de versões disponibiliza aos projetistas:

- consulta a todas as versões de um dado objeto de projeto;
- acesso a qualquer versão de um objeto para utilização num próximo passo de projeto;
- seleção de versões (para consulta e acesso) que atendam a determinadas restrições de projeto;
- classificação de versões segundo uma determinada política imposta pelo projetista;
- visualização das relações de uma dada versão de um objeto de projeto com versões de outros objetos.

2.7.2 Controle de configurações

Segundo [WAG 94], uma representação composta Y^k de um objeto Y contém sub-objetos C_i,...,C_j, que são instâncias de X_i,...,X_j. Como um objeto X_n pode possuir múltiplas versões, deve-se selecionar, para sua instância em Y^k, uma destas, para que a representação Y^k possa ser processada por uma ferramenta de análise ou síntese. Uma configuração de Y^k é uma seleção de uma versão para cada objeto X_n, instanciado por cada sub-objeto C_n. A este processo de seleção, chamamos resolução da configuração.

A resolução de configurações pode ser implementada de diversas formas, conforme o mecanismo de controle de configurações empregado. Uma forma de resolução exige que se defina, quando da criação da representação composta do objeto, quais as versões que a constituem (configurações estáticas). Outra retarda esta definição para

o momento do uso da representação por alguma ferramenta (configurações dinâmicas) [WAG 94].

2.7.3 Controle de acesso

Um mecanismo de controle de acesso [WAG 94] aos dados é um recurso importante num framework pelo aspecto de segurança dos dados. O controle de acesso pode ser feito com dois níveis de granularidade possíveis. Um mecanismo de controle de larga granularidade define direitos de acesso para grandes grupos de objetos, como por exemplo bibliotecas. Um mecanismo de granularidade fina, por outro lado, pode definir direitos de acesso específicos para cada objeto ou até versão de objeto.

2.8 Gerência de projeto

Na seção 2.4, definimos genericamente a gerência de projeto como sendo o conhecimento, por parte do projetista, do conjunto de ferramentas que vai utilizar (e em qual seqüência) para alcançar os resultados planejados. Agora, vamos ampliar este conceito.

Fundamentalmente, a gerência de projeto deve ser um serviço do ambiente e não do projetista. Assim, chama-se gerência de projeto ou gerência do processo de projeto à forma como o sistema controla o fluxo de execução de tarefas. Tal controle objetiva automatizar seqüências tediosas de tarefas, forçar disciplina de projeto, para evitar inconsistências, bem como ajudar projetistas inexperientes. Mas, e o que vem a ser tarefas? Tarefas são abstrações de ferramentas. São o encapsulamento de uma ou mais ferramentas numa unidade de trabalho com um objetivo específico.

Na realidade, além do controle do fluxo de tarefas, a gerência de projeto trata também do controle das equipes envolvidas no processo, administrando o acesso às fer-

ramentas e a cooperação entre equipes e entre projetistas, devido à modularização do sistema em desenvolvimento, como se estudou anteriormente. No entanto, não detalharemos tais tópicos por não estarem ligados, num primeiro momento, aos objetivos do presente trabalho.

O fluxo de tarefas pode ser definido através de descrições textuais ou gráficas, que englobam seqüências de execução de ferramentas (ou mesmo de tarefas previamente definidas) com uma função específica. Ou, em ambientes mais poderosos, o fluxo pode ser derivado pelo sistema, em tempo de execução, levando-se em conta objetivos e restrições de projeto definidos previamente. Em ambos os casos, distinguem-se duas entidades que, em conjunto, implementam o controle do fluxo de tarefas. Uma é constituída pelas próprias descrições de fluxos de tarefas e a outra é o mecanismo que, a partir destas descrições e, na maioria dos casos, da interação com os projetistas, procede à execução das ferramentas que constituem a tarefa.

A especificação de objetivos e restrições, nestes sistemas que geram o fluxo de tarefas em tempo de execução, pode ser feita em momentos distintos. Ela pode ser feita no desenrolar do projeto, para cada objeto sendo criado, ou, o que é mais comum, num momento inicial, sendo o sistema responsável pelo uso destas informações nos momentos apropriados.

Durante o processo de projeto, é fundamental que o ambiente possibilite o retorno a pontos anteriores (entre tarefas, ou mesmo, dentro de uma tarefa, isto é, entre as ferramentas que a compõem) , para que, a partir daí, o projetista possa testar alternativas (como, por exemplo, uma ferramenta que contemple uma tecnologia diferente ou que possa maximizar ou minimizar um dado aspecto de um determinado objeto). A esta possibilidade denominamos Exploração do Espaço de Projeto, que pode ser feita, exclusivamente, pelo sistema, levando em conta os objetivos e as restrições, exclusivamente pelo projetista, ou pelo projetista com a supervisão do sistema.

A integração de novas ferramentas, requisito para sistemas abertos, pode disponibilizá-las, imediatamente ou não, para o mecanismo de controle do fluxo de tarefas

existente no ambiente. A execução de uma ferramenta, num próximo passo de projeto, é determinada pela sua referência numa descrição de fluxo de tarefas. Se esta referência for feita através da codificação da linha de comando de chamada da ferramenta, será necessária a criação de uma descrição para a utilização da nova ferramenta. Enquanto que, se a referência for feita através da especificação da função a ser desempenhada naquele passo de projeto (simulação, por exemplo), ficando a cargo do mecanismo a identificação das ferramentas que desempenham aquela função, a ferramenta estará automaticamente disponível (seguindo no exemplo, se for um simulador), sem depender da criação de uma nova descrição. Obviamente, este último caso é o mais conveniente, pois não há necessidade de qualquer modificação para a utilização da nova ferramenta.

2.9 Resumo

Os serviços da gerência de dados e da gerência de projeto estão resumidos na Tabela 2.1.

TABELA 2.1- Resumo de serviços

Gerência de Dados	**Gerência de Projeto**
- *controle de versões*	- *controle do fluxo de execução de tarefas*
- controle de configurações	- gerenciamento das equipes de projeto
- controle de acesso aos dados	- controle de acesso às ferramentas

O Sistema Gerenciador de Documentação de Projeto, SGDP, utiliza o recurso de documentação dos passos de projeto para apresentar soluções a problemas de gerência de dados e de gerência de projeto.

Num primeiro momento, o sistema tratará, particularmente, o controle de versões e o controle do fluxo de execução de tarefas. A este último item, chamaremos, durante toda a apresentação do Sistema a partir do capítulo 4, de controle do fluxo de execução de ferramentas, uma vez que são estas (e não as tarefas) que constituem a unidade de execução no Sistema.

3 ESTUDO DE CASOS

Neste terceiro capítulo, descrevemos e analisamos alguns ambientes sob a ótica do aproveitamento da documentação dos passos de projeto, evidenciando as vantagens da sua utilização e buscando aspectos que poderiam ser aperfeiçoados. Os sistemas a serem estudados são os seguintes:

- ADAM Planning Engine - da University of Southern California. Utiliza técnicas de inteligência artificial para a criação de planos de execução de tarefas;
- CADEC (CArlsruhe DEsign Consultant) - do Forschungszentrum fuer Informatik - Centro de Pesquisas em Informática, ligado à Universidade de Karlsruhe, Alemanha. Também usa técnicas de inteligência artificial e distingue quatro fases obrigatórias para a conclusão do processo de projeto;
- Cadweld - da Carnegie Mellon University. Provê uma visão orientada a objeto para o encapsulamento de ferramentas. O mecanismo de controle do fluxo de tarefas busca, neste encapsulamento, informações para uma possível execução da ferramenta num próximo passo de projeto;
- um ambiente que propõe uma estrutura de dados que registra toda a história do projeto. Esta estrutura é organizada em diversos níveis. Este sistema foi desenvolvido na University of California, em Berkeley, e chamaremos de Sistema de Modelo Histórico;
- VOV - também da University of California em Berkeley. É um Gerenciador de Projeto baseado numa estrutura de grafo denominada *design trace*, que registra todos os passos dos projetistas.

3.1 Um Gerenciador de Projeto baseado em técnicas de Inteligência Artificial: o ADAM Planning Engine

3.1.1 Descrição

O ADAM Planning Engine [KNA 86] é um sistema especialista que constrói um plano de execução de tarefas em resposta a necessidades de um conjunto particular de especificações e restrições, que são informadas pelo projetista. Em seguida, este plano é executado e, no caso de sucesso, um projeto correto é produzido. Caso contrário, um novo plano deve ser construído. O planejamento, aqui, é usado para simular a operação do sistema e, assim, economizar pesquisa e evitar inconsistências.

Há dois espaços distintos a serem considerados neste ambiente: o espaço de execução constituído de ferramentas e os dados para estas, e o espaço de planejamento que é formado por representações abstratas de ferramentas (denominadas operadores) e de dados (chamadas de estados). Os estados do espaço de planejamento são coleções de asserções acerca do projeto e os operadores são caracterizados por pré e pós-condições. As pré-condições expressam as condições para a aplicação do operador e as pós, condições que existirão após a utilização daquele operador. Na Figura 3.1, a estrutura Lisp **exist eqs-bool it** (que verifica se as equações booleanas de um circuito **it** existem) constitui a pré-condição para a aplicação dos operadores **build-Std-cell-logic** ou **build-PLA-logic**, que, uma vez aplicados, adicionarão, ao estado do projeto, suas pós-condições.

Em ADAM, então, um plano é construído por encadeamento sucessivo a partir do estado inicial. Ou seja, as asserções do estado inicial devem coincidir com as pré-condições de um dado operador, que, aplicado, vai gerar um novo estado. O processo se repete até a obtenção do estado objetivo. E, por último, procede-se à execução do plano.

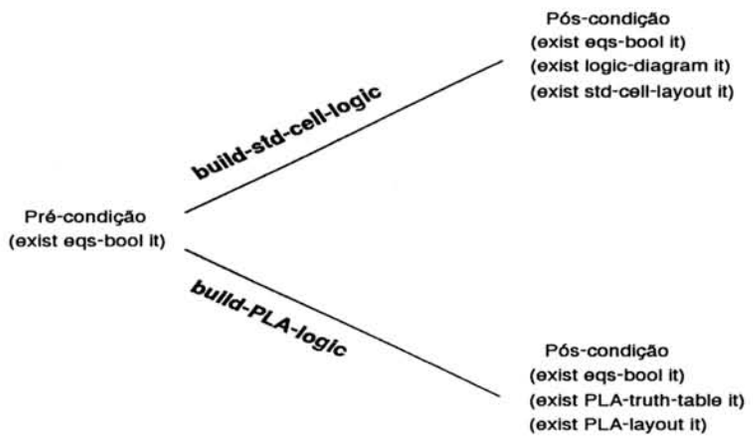


FIGURA 3.1- Ilustração do planejamento em ADAM

3.1.2 Análise

Intencionalmente, iniciamos este estudo com um sistema que não menciona a documentação dos passos de projeto para mostrar a desvantagem da sua não utilização. Imagine que um projetista vai reutilizar o sistema ADAM, declarando as mesmas especificações e restrições de uma rodada anterior. Ele terá que esperar um tempo, no mínimo razoável, para que o sistema derive um plano de execução de tarefas utilizando as técnicas de inteligência artificial disponíveis. Agora, se este plano tivesse sido documentado e armazenado na primeira vez que foi criado, o tempo gasto seria apenas o da execução das tarefas.

3.2 CADEC - um Consultor de Projeto VLSI

3.2.1 Descrição

CADEC (CArlsruhe DEsign Consultant) [KWE 92] é um consultor de projeto VLSI (ou projeto de sistemas eletrônicos) que distingue quatro fases obrigatórias para a conclusão do processo de projeto, conforme a Figura 3.2.

FIGURA 3.2- Fases de projeto segundo CADEC

Na fase inicial, de **especificação de projeto**, ocorre o particionamento em subprojetos, a escolha de formatos (descrições informais, descrições HDL ou netlists, por exemplo) e tarefas (síntese, simulação, verificação, teste, etc) e a determinação de restrições de tecnologia, tempo, custo e potência.

Na fase seguinte, a de **métodos recomendados de projeto**, determina-se o fluxo de ferramentas a ser usado a partir dos dados da fase inicial. O ambiente apresenta conjuntos de ferramentas alternativos para que se selecione um. A partir desta escolha, o fluxo é derivado por um sistema especialista e assume a forma de um grafo com nodos ferramentas e nodos dados de projeto (ver Figura 3.3).

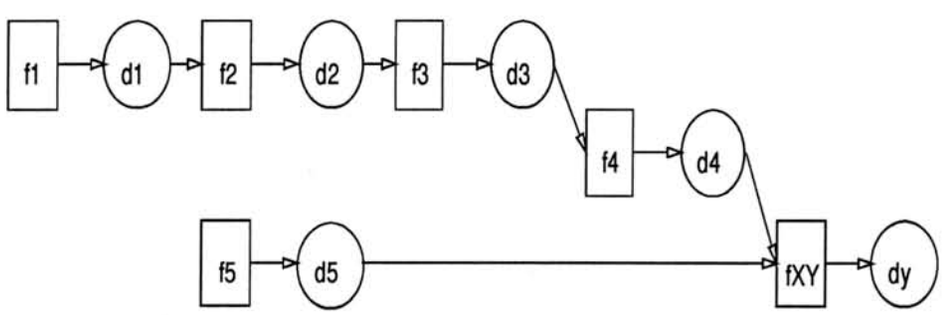


FIGURA 3.3- Exemplo de um grafo derivado em CADEC

Seguindo, na fase de **controle de execução de ferramentas**, todas as ferramentas passíveis de execução já foram determinadas e a decisão sobre qual executar a seguir (na seqüência do grafo) é deixada para o projetista. Dependendo das diferenças entre expectativas e resultados, há a possibilidade de um retorno para novas decisões de prosseguimento. Num ambiente distribuído, o sistema oferece, também, a possibilidade de escolha do recurso onde executar a próxima ferramenta.

E, na última fase, a de **compilação de resultados**, é gerada uma documentação com base nas informações acumuladas nas fases anteriores. A análise de tal documentação, por parte do construtor do ambiente, pode revelar pontos importantes para o aperfeiçoamento do sistema.

3.2.2 Análise

Em CADEC, há duas situações distintas nas quais podemos analisar a utilização da documentação dos passos de projeto. Uma está, explicitamente, apresentada na fase de compilação de resultados, na qual é gerado um arquivo de log com as ações tomadas nas fases anteriores. Porém, tal arquivo só presta serviço ao sistema em conjunto com a intervenção do construtor do ambiente. Por si só, não possui funcionalidade alguma, é um mero arquivo texto, quando poderia ser utilizado como uma fonte de informações para que o próprio sistema escolhesse o melhor recurso para a execução da próxima ferramenta (no estágio atual, ele apenas oferece a possibilidade de escolha), apenas para dar um exemplo.

E a outra situação de utilização da documentação dos passos de projeto é a possibilidade do retorno a pontos anteriores de projeto para a utilização de ferramentas alternativas. O fluxo possível de ferramentas fica documentado no grafo, bem como os dados de entrada e de saída. Porém, parece não haver lugar, neste mesmo grafo, para se manter e gerenciar novos dados que seriam criados quando do retorno a pontos anteriores (as execuções das ferramentas trabalham sempre sobre os mesmos dados, não havendo possibilidade de geração de alternativas). Sendo assim, nesta segunda situação,

podemos concluir que a documentação é utilizada para auxiliar apenas a gerência de projeto, mantendo o fluxo de execução, e não é utilizada para os problemas de gerência de dados.

3.3 Cadweld - uma visão orientada a objeto do controle de ferramentas de CAD

3.3.1 Descrição

Em Cadweld [DAN 89], as ferramentas de CAD são encapsuladas por um modelo representativo, que, entre outras coisas, apresenta as características e potencialidades das ferramentas, e gerencia detalhes de baixo nível (normalmente detalhes relacionados à execução, ver Figura 3.4). Esta ferramenta encapsulada constitui um CTKO (Cad Tool Knowledge Object).

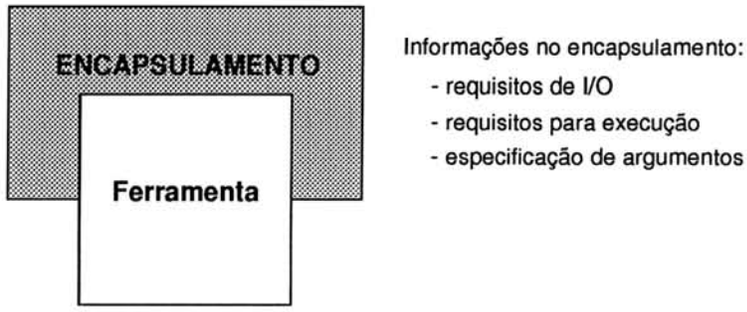


FIGURA 3.4- Um Cad Tool Knowledge Object

O administrador do sistema força políticas de projeto, relacionando passos de projeto nas chamadas *CAD Tasks*, que englobam seqüências de execução de CTKOs ou outras *CAD Tasks*. O mecanismo de controle do fluxo de tarefas, baseado nas *CAD Tasks*, envia informações a um centralizador sobre o próximo passo de projeto a ser executado, para identificar um grupo de CTKOs, ou outras *CAD Tasks*, voluntários para esta execução. A escolha de um CTKO particular, para dar prosseguimento ao projeto, dentre aqueles que se voluntariam, pode ser automática ou feita pelo projetista.

O sistema mantém a história das decisões e dados para possíveis *backtracks*, usando um mecanismo de log que salva o contexto, bem como um servidor de versões de arquivos.

3.3.2 Análise

Em Cadweld, a gerência de dados parece ficar bem atendida pelo histórico mantido no servidor de versões de arquivos. E o controle do fluxo de execução de tarefas, no que se refere ao registro dos fluxos utilizados (bem como de dados associados) igualmente é atendido, pois o sistema possibilita retornos a pontos anteriores (*backtracks*) para se tentar alternativas com outros CTKOs ou *CAD Tasks*. Há, então, a manutenção do estado do projeto tanto em termos de dados como de fluxos de execução utilizados (em CADEC, havia apenas esta último).

Mas, a documentação dos passos de projeto seria ainda melhor aproveitada, no que se refere à gerência de projeto, se registrasse a atividade do administrador do sistema quando da construção de novas *CAD Tasks*. Se um dado fluxo de ferramentas chegar aos objetivos, ele poderia ser, automaticamente, capturado na documentação e transformado numa *CAD Task* (no estágio atual, o administrador do sistema codifica as *CAD Task*). A documentação, em Cadweld, então, apenas registra um fluxo de tarefas já pré-definido pelo administrador do sistema, quando poderia ser utilizada para disponibilizar novas *CAD Tasks* para o mecanismo de controle, ou seja, poderia ser utilizada para criar novas descrições de fluxos de tarefas.

3.4 Um Modelo Histórico para o gerenciamento do processo de projeto VLSI

3.4.1 Descrição

O Sistema de Modelo Histórico [CHI 91] é baseado numa estrutura de dados da história do projeto, que registra todos os passos dos projetistas, incluindo tarefas executadas e objetos associados. A história possui uma forma hierárquica, cujos níveis são: processo, atividade, tarefa e ferramenta, conforme Figura 3.5. Os conceitos destas duas últimas são idênticos ao que foi apresentado na seção 2.8. Já atividades constituem uma unidade coerente de tarefas e dados associados, com um objetivo específico. E processos registram a história do projetista dentro de um dado projeto, englobando diversas atividades. Detalharemos, a seguir, o funcionamento de tarefas e atividades, pois resumem bem o propósito deste ambiente.

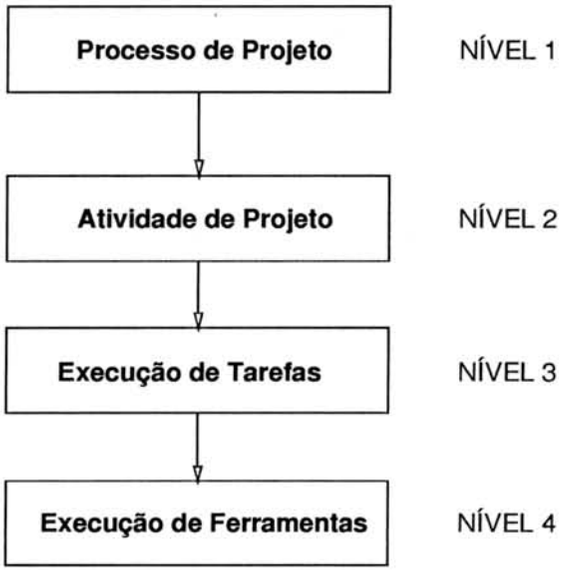


FIGURA 3.5- Níveis do Sistema de Modelo Histórico

As tarefas são a base da gerência de projeto neste sistema. Sua manutenção consiste de três etapas: um compilador que traduz as especificações (de seqüências de execução de ferramentas) em *task templates*, que são armazenadas na base de dados, um

gerenciador que interage com os projetistas para a execução das tarefas, e uma ferramenta de depuração para as especificações.

Uma seqüência de tarefas com um objetivo específico constitui uma atividade.

Um projetista encara três espaços distintos no desenrolar do processo de projeto: o espaço de dados (criados ou referenciados), o espaço de ferramentas (todas as ferramentas do ambiente), e o espaço de operações (modificações nos dados pela execução de ferramentas). O conteúdo físico da história de uma atividade combina subconjuntos destes espaços:

- *activity workspace*: contém os dados utilizados e produzidos pelas tarefas da atividade. Podem ser organizados em versões e configurações;
- *activity history*: seqüência de registros de história, cada um correspondendo a uma invocação de tarefa. Pode englobar outros registros hierarquicamente (subtarefas). Registra entradas, saídas e parâmetros. Entre dois registros, existe um *design point*, que identifica o estado do *activity workspace*, chamado *data scope*.

Uma atividade, então, provê uma forma de organização que vai além da simples organização temporal da execução de tarefas. Dentro dela, há um cursor atual, que aponta para o *design point* em que será inserido o registro de história da tarefa subseqüente. Após a inserção, o cursor é avançado. Há, também, a possibilidade de retornar o cursor para um instante anterior: o *design point* e o *data scope* acompanham. E pode-se ainda manipular a história através de um browser gráfico, que simplifica a identificação e seleção de versões.

3.4.2 Análise

O Sistema de Modelo Histórico é bem completo com relação à utilização da documentação dos passos de projeto, pelo menos, no que tange aos aspectos de gerência

de dados e gerência de projeto (manutenção do estado: versões de objetos e fluxos de execução). Mas, com relação a este último, novamente, a documentação não é utilizada para auxiliar na criação de novas descrições de fluxos de tarefas, o que poderia ser feito disponibilizando, automaticamente, novas seqüências de execução de ferramentas, que lograram êxito nos seus objetivos, para que constituíssem tarefas do mecanismo de controle do fluxo de tarefas.

O ponto mais interessante deste ambiente é a sua estruturação em níveis, o que possibilita a visualização de como foram desenvolvidas partes específicas do projeto. Por exemplo, pode-se visualizar como procedeu um projetista para efetuar uma dada atividade, dentre as muitas que ele possui dentro do projeto.

3.5 VOV - um Gerenciador de Projeto baseado em design traces

3.5.1 Descrição

VOV [CAS 90] é um gerenciador automático para projetos VLSI que, a partir das próprias ferramentas, registra toda a atividade dos projetistas numa estrutura de grafo chamada *design trace* - ver Figura 3.6. Os arcos do grafo expressam relações de entrada e saída entre os nodos, que podem ser de dois tipos: lugares e transições. Genericamente, lugares correspondem aos dados e transições, às ferramentas executadas.

VOV utiliza uma arquitetura cliente-servidor. Cada projeto possui um *design trace* associado, gerenciado pelo servidor. Vários clientes podem se conectar a este servidor, requisitando consultas ou modificações no grafo.

Os projetistas utilizam as ferramentas normalmente, como se o gerenciador VOV não existisse. Quando a ferramenta é executada, estabelece uma conexão com o servidor, informando dados de entrada, dados de saída e o momento da sua finalização,

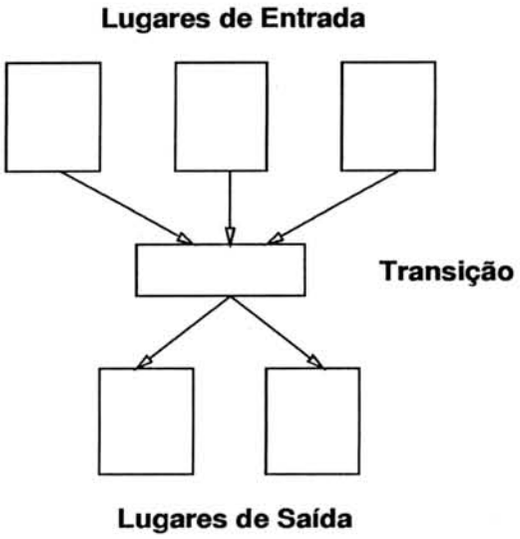


FIGURA 3.6- Design Trace

para que este construa o grafo. Os fontes das ferramentas não precisam ser modificados: elas são encapsuladas por *shell-scripts*.

O ambiente disponibiliza os seguintes serviços:

- uma interface gráfica para visualização e consulta ao *design trace*;
- aproveitamento de *design traces* prontos para guiar usuários novatos na seqüência correta de execução de ferramentas;
- criação de refinamentos (mesmo lugar no *trace*) e alternativas (lugares distintos) de objetos de projeto;
- gerenciamento de equipes de projeto, notificando entradas não válidas ou saídas já utilizadas. Se uma mudança é detectada em algum lugar do grafo, todos os outros lugares dependentes tornam-se inválidos, devendo ser refeitos;
- medições nos dados de projeto. Para isso, cada medida é caracterizada por um nome, uma unidade, um domínio (conjunto de lugares o qual a medida se aplica), um método para calculá-la e um objetivo opcional, que descreve se é para minimizar a medida, maximizar, etc.

3.5.2 Análise

Este último sistema faz uso praticamente completo das funcionalidades da documentação dos passos de projeto. Utiliza-a na gerência de dados, possibilitando a criação e manutenção de refinamentos e alternativas no *design trace*. E, também, na gerência de projeto, gerando os fluxos de tarefas a serem utilizados por usuários novatos (ou reutilizado pelos já experientes), o que não acontecia nos sistemas anteriores, como já foi mencionado. Utiliza-a até no controle de acesso aos dados pelas equipes de projeto. A documentação apenas não é utilizada para tratar a distribuição do processamento.

Em VOV, a documentação dos passos de projeto é o próprio sistema. Todos os serviços disponíveis são provenientes dela. Agora, se pensássemos VOV como um sistema independente, provedor de informações e serviços para outros sistemas, com certeza, poderíamos utilizá-lo em todos os anteriores. Por exemplo, em ADAM, registraria e manteria planos de execução válidos; e em Cadec, executaria as funções de um gerenciador de versões, inexistente naquele ambiente.

E, com isso, teríamos inúmeras vantagens, pois os construtores de sistemas poderiam concentrar suas atenções apenas no algoritmo de gerência de projeto (ou de dados) que estivessem propondo, deixando outros detalhes para serem mantidos por este sistema provedor independente.

3.6 Resumo

A funcionalidade apresentada pela documentação dos passos de projeto foi evoluindo nos sistemas estudados. Em ADAM, a documentação não foi utilizada e mostramos as desvantagens. Em CADEC, constituía uma fonte de informações que poderia levar o construtor do ambiente a propor melhorias no sistema e, também, registrava o fluxo de ferramentas para possíveis *backtracks*. Em Cadweld, atendia à gerência de dados (controle de versões) e mantinha o fluxo de execução para retornos a pontos anteriores de

projeto. No Sistema de Modelo Histórico, foi adicionada uma estruturação em níveis, que possibilitava o aproveitamento de partes específicas da documentação. E, finalmente, em VOV, vimos a documentação implementar o próprio mecanismo de controle do fluxo de tarefas.

TABELA 3.1- Uso da documentação nos sistemas estudados

Áreas	**Serviços**	**CADEC**	**Cadweld**	**SMH**	**VOV**
Gerência de Dados	controle de versões	não	sim	sim	sim
	controle de configurações	não	não	sim	não
	controle de acesso aos dados	não	não	não	sim
	manutenção do estado - dados (p/*backtracks*)	não	sim	sim	sim
Gerência de Projeto	manutenção do estado - fluxo de tarefas (p/*backtracks*)	sim	sim	sim	sim
	controle do fluxo de tarefas	não	não	não	sim
	disponibilização de novos fluxos de tarefas	não	não	não	sim
	gerenciamento das equipes de projeto	não	não	não	sim
	controle de acesso às ferramentas	não	não	não	não
Outros	provedor de informações e serviços a usuários finais	sim	sim	sim	sim
	provedor de informações e serviços a outros sistemas	não	não	não	não

Resumimos o estudo dos sistemas na Tabela 3.1 (note que o sistema ADAM não consta da tabela, um vez que não utiliza a documentação dos passos de projeto).

4 FUNCIONALIDADE DO SISTEMA GERENCIADOR DE DOCUMENTAÇÃO DE PROJETO

Neste capítulo, vamos apresentar o Sistema Gerenciador de Documentação de Projeto, delineando os serviços que são oferecidos por ele. Iniciamos listando a funcionalidade possível da documentação dos passos de projeto, segundo o estudo proposto no capítulo anterior:

- disponibilização de dados relevantes sobre o projeto, que podem ser utilizados, por exemplo, pelo construtor do ambiente para propor melhorias no sistema;
- manutenção e controle dos objetos de projeto, armazenando relações entre os dados e gerenciando versões;
- registro do fluxo de tarefas utilizado, bem como de dados associados, para possíveis retornos a pontos anteriores de projeto;
- disponibilização de novas descrições de seqüências válidas de execução de ferramentas ao mecanismo de controle do fluxo de tarefas existente;
- enfoque e tratamento de partes específicas do projeto (uma dada atividade, por exemplo, dentre as muitas que compõem o processo de projeto) quando esta estiver organizada em níveis hierárquicos.

Todos os ítens listados serão tratados pelo SGDP e, além destes, o provimento de informações que permitam a escolha do melhor recurso para executar uma ferramenta num ambiente distribuído

4.1 Ambiente de Projeto

O Sistema Gerenciador de Documentação de Projeto captura e mantém a história de um projeto (resumidamente, fluxos de ferramentas utilizados e dados associados), disponibilizando, através de uma API (Application Programming Interface), informações e serviços para outros sistemas.

No presente trabalho, como forma de demonstrar sua funcionalidade, ele constitui-se no sistema principal de um ambiente de projeto composto por duas aplicações (ver Figura 4.1), a saber:

FIGURA 4.1- Ambiente de Projeto

- aplicação A, que utiliza os serviços de gerência de dados e as informações sobre a seqüência de execução de ferramentas armazenadas no Sistema Gerenciador. Estas informações poderão ser utilizadas, posteriormente, por um mecanismo de controle de fluxo de ferramentas, disponibilizado pela aplicação B. De que maneira? A aplicação A permite a seleção, a partir das informações disponíveis

no SGDP, de uma seqüência de ferramentas que tenha apresentado resultados satisfatórios, e o seu correspondente armazenamento em arquivo. Novas seqüências poderão ser selecionadas e armazenadas, de forma que teremos um conjunto de arquivos com seqüências validadas de execução de ferramentas para reaproveitamento posterior. Denominaremos este conjunto de arquivos de *biblioteca de tarefas*;

- a aplicação B, que utiliza os serviços de gerência de dados e os serviços de gerência de projeto (constituídos pela biblioteca de tarefas) do Sistema Gerenciador. O SGDP dará acesso à biblioteca de tarefas à aplicação B, que, por sua vez, possibilitará que o projetista escolha a tarefa que pretende executar. Depois da seleção, a aplicação B guiará o usuário na execução desta tarefa.

A diferença fundamental entre as aplicações é o fato de que, na aplicação A, o projetista tem liberdade total para executar as ferramentas na seqüência que melhor lhe convier, enquanto que, na aplicação B, o projetista fica amarrado a um mecanismo de controle do fluxo de ferramentas, baseado na biblioteca de tarefas do Sistema Gerenciador.

Como foi colocado na introdução deste trabalho, o objetivo do mesmo é a construção de um sistema que, através do gerenciamento da documentação dos passos de projeto, constitua-se num provedor de informações e serviços para outros sistemas. E, como podemos perceber pela descrição das três partes que compõem o ambiente, este objetivo está materializado no sistema principal, isto é, no SGDP. Então, para que as outras duas aplicações? Pelos seguintes motivos:

- para demonstrar que a interação do SGDP é feita com outros sistemas, e não diretamente com o usuário final (projetista);

- para ilustrar as potencialidades do SGDP, que serão, com certeza, exploradas por outros sistemas "não proprietários";

- para, através de um protótipo, disponibilizar aos usuários projetistas um sistema completo, com serviços consistentes de gerência de dados e gerência de projeto.

A seguir, descrevemos o Sistema Gerenciador de Documentação de Projeto. A descrição das aplicações será feita no capítulo seguinte.

4.2 Sistema Gerenciador de Documentação de Projeto

O SGDP é o sistema central do ambiente e objetivo principal deste trabalho. Ele permite a criação e a manutenção de uma estrutura de dados que registra todos os passos dos projetistas, isto é, de uma estrutura que guarda a história do projeto.

O único pressuposto para a utilização do SGDP é o cadastramento, por parte daqueles sistemas que não contemplam a execução propriamente dita das ferramentas, das ferramentas que serão utilizadas.

A partir de então, o sistema permite às aplicações (e a outros sistemas igualmente):

- a execução das ferramentas cadastradas;
- o registro dos fluxos de execução de ferramentas;
- a visualização dos fluxos de execução de ferramentas utilizados;
- a seleção de determinados fluxos para armazenamento em arquivo;
- a reexecução de fluxos;
- a disponibilização de fluxos específicos armazenados em arquivo;
- a consulta aos dados gerados e às suas relações;

- a obtenção de uma dada versão de um objeto para a sua utilização num próximo passo de projeto;
- a manutenção de estatísticas sobre a execução de ferramentas para aproveitamento por um possível algoritmo de distribuição de carga;
- a manutenção de dados estatísticos sobre a utilização do sistema como um todo, que poderão ser utilizados para a proposição de melhorias no ambiente de projeto.

A comunicação entre SGDP e aplicações se dá através de mensagens denominadas *requisições de serviço*, funcionando da seguinte forma: conforme a informação ou serviço que a aplicação espera receber, ela monta e envia uma determinada requisição de serviço. O SGDP, ao receber esta requisição, decodifica-a e fornece a informação ou serviço.

4.2.1 Cadastro e Execução de ferramentas

O SGDP procede à execução de ferramentas desde que estas tenham sido previamente cadastradas no sistema, como já foi mencionado. As informações necessárias para o cadastro de uma ferramenta são (ver Figura 4.2):

- nome da ferramenta;
- diretório de execução;
- diretório de trabalho;
- argumentos.

Para cada argumento, deve-se informar (ver Figura 4.3):

- mensagem a ser exibida quando da entrada de um valor para o argumento;

UFRGS
INSTITUTO DE INFORMÁTICA
BIBLIOTECA

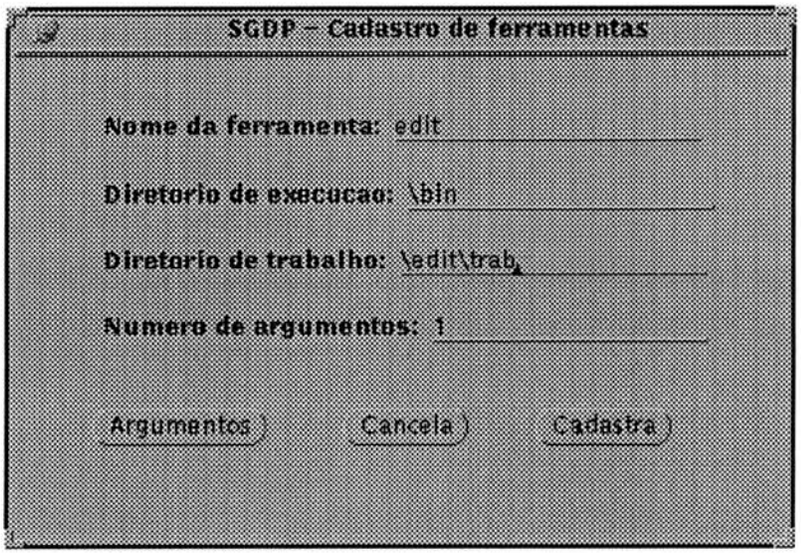


FIGURA 4.2- Tela de Cadastro de Ferramentas

- delimitador do argumento, que vem a ser um literal que identifica o argumento. Por exemplo, na chamada *tool -f45*, *-f* seria o delimitador do argumento inteiro 45. O delimitador pode ser nulo;
- tipo: inteiro, real, nome de arquivo, string (cadeia de caracteres);
- se é opcional ou obrigatório.

Se o argumento for um nome de arquivo, deve-se informar adicionalmente:

- a classe do arquivo, que deve ser definida pelo usuário para facilitar, posteriormente, a consulta aos objetos de projeto (neste trabalho, os objetos de projeto são arquivos). Para cada classe, existirão um ou mais objetos de projeto; e, para cada objeto de projeto, existirão uma ou mais versões (conforme Figura 4.4);
- se é um dado de entrada (in) ou de saída (out), ou ambos (in e out);
- o caractere separador, no caso de vários nomes de arquivo.

Estas informações, em conjunto com os valores dos argumentos a serem informados no momento da execução propriamente dita, serão utilizadas para a montagem da

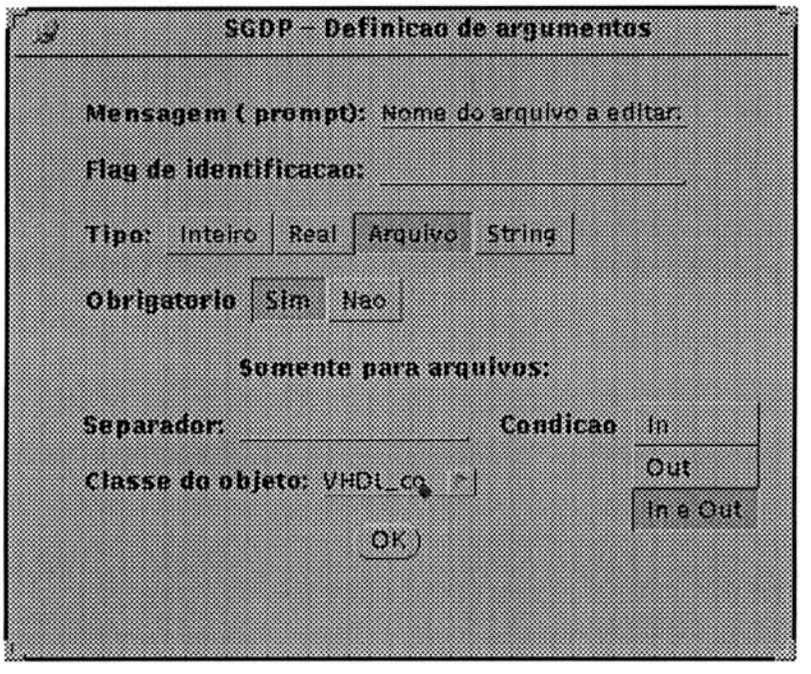


FIGURA 4.3- Tela de Definição de Argumentos

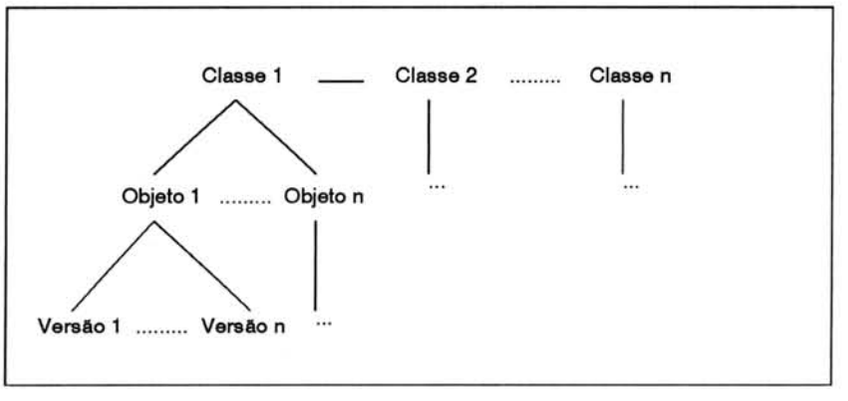


FIGURA 4.4- Classes, objetos e versões em SGDP

linha de comando de chamada da ferramenta, bem como serão utilizadas para o **registro do fluxo de execução** na estrutura de dados da história do projeto.

O SGDP permite o cadastro de ferramentas quando recebe, das aplicações ou outros sistemas, uma *requisição de cadastro de ferramenta.* Este cadastro, então, poderá ser feito interativamente, por meio de janelas de diálogo, ou através de um arquivo texto que deve seguir um formato pré-determinado [1] e cujo nome vem informado na requisição.

[1] o formato deste arquivo é detalhado no anexo A-2

O SGDP executa uma ferramenta quando recebe uma *requisição de execução de ferramenta*. Se, na requisição, for informado o nome da ferramenta a ser executada, o SGDP solicitará diretamente os argumentos para a execução (ver Figura 4.6). Para cada argumento que se constituir de um arquivo de dados, será permitida a seleção, através de um browser gráfico, de uma dada versão de um determinado objeto de projeto daquela classe requerida pela ferramenta. Se não for informado o nome da ferramenta na requisição, o Sistema apresentará um browser gráfico com as ferramentas cadastradas para seleção (ver Figura 4.5).

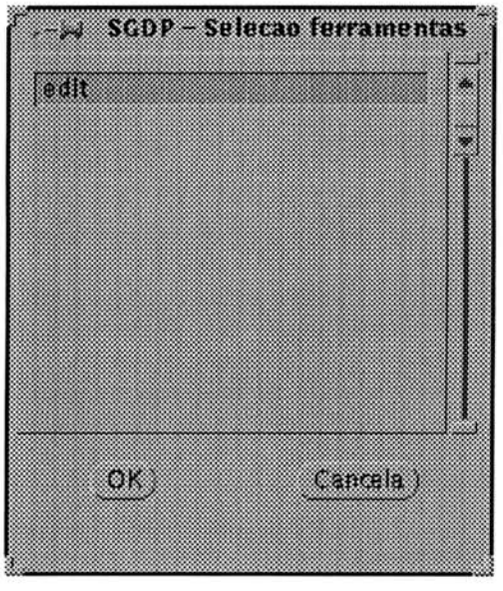


FIGURA 4.5- Tela de Seleção de Ferramentas para Execução

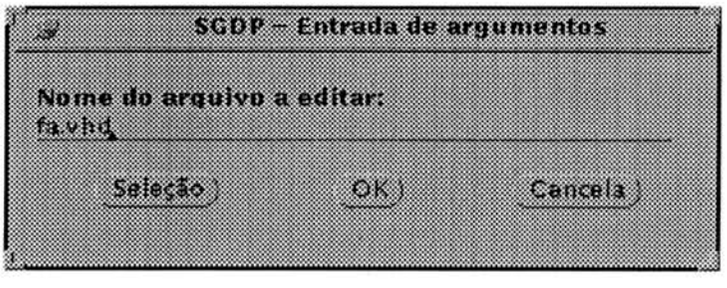


FIGURA 4.6- Tela de Entrada de Argumentos

Aquelas aplicações ou sistemas que fazem, diretamente, a execução das ferramentas, e não através do SGDP, poderão utilizar, ainda assim, o Sistema para o registro dos fluxos utilizados. Para isso, elas enviarão mensagens informando os arquivos de entrada, a ferramenta executada e os arquivos de saída.

4.2.2 Visualização de fluxos de ferramentas

O SGDP possibilita a visualização de um fluxo de execução de ferramentas através de uma *requisição de visualização de fluxo* enviada pelas aplicações ou outros sistemas.

Ele solicitará, através de um browser gráfico, a seleção de uma versão de um objeto de projeto de uma determinada classe. Com esta informação, o Sistema derivará e mostrará graficamente o fluxo de ferramentas que gerou aquela versão, juntamente com as versões dos objetos usados ao longo do fluxo (na Figura 4.7, é apresentada uma segunda execução da ferramenta *edit*, além daquela exibida nas Figuras 4.5 e 4.6).

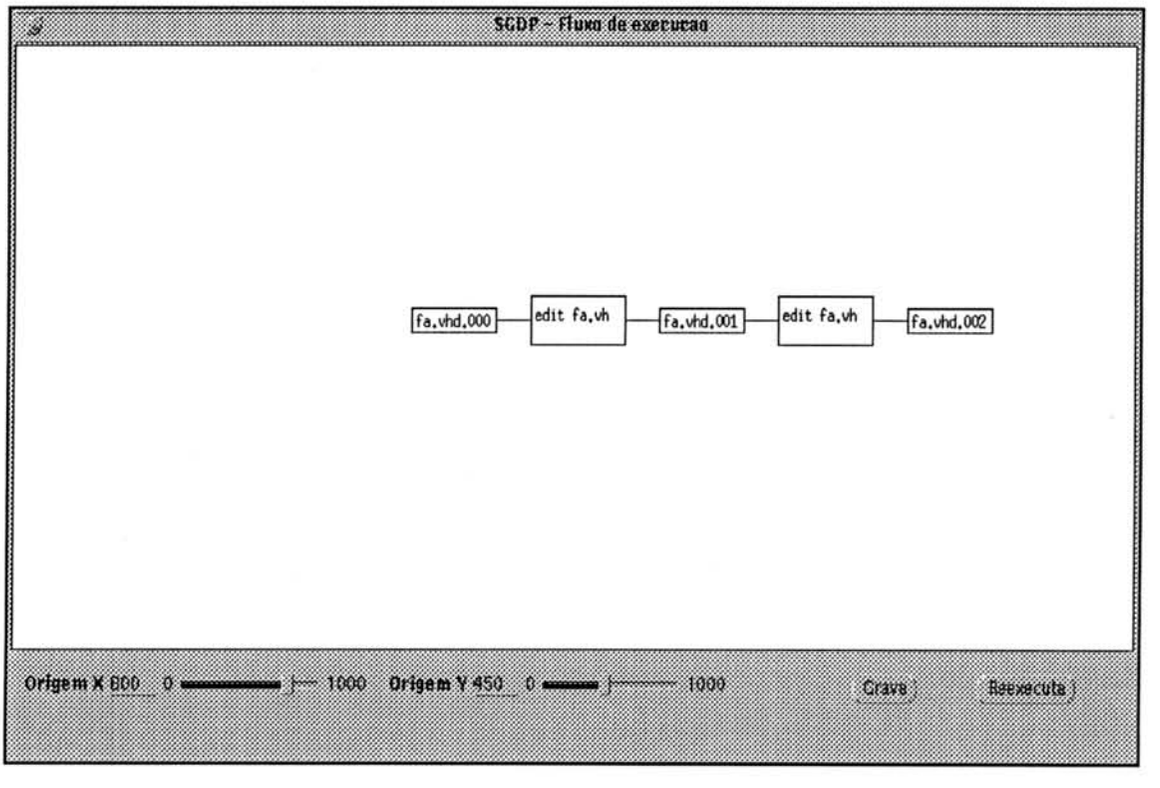


FIGURA 4.7- Tela de Visualização de Fluxo

4.2.3 Seleção de fluxos para armazenamento em arquivo

O SGDP permite a seleção de um fluxo de ferramentas para armazenamento em arquivo por meio de uma *requisição de seleção de fluxo*.

Ele, então, solicitará, através de um browser gráfico, a seleção de uma versão de um objeto de projeto de uma determinada classe. Com esta informação, o Sistema derivará e mostrará graficamente, através de uma janela de diálogo, o fluxo de ferramentas que gerou aquela versão. Nesta interface, será permitido, ao usuário da aplicação ou sistema que fez a requisição, indicar a ferramenta que inicia o fluxo e aquela que o encerra (ver Figura 4.8). Por último, se não foi informado na requisição, deve-se entrar com o nome do arquivo onde guardar o fluxo. As informações são gravadas neste arquivo num formato pré-estabelecido [2].

4.2.4 Reexecução de fluxos

O SGDP possibilita a reexecução de um fluxo de ferramentas através de uma *requisição de reexecução de fluxo*.

O Sistema pedirá, através de um browser gráfico, a seleção de uma versão de um objeto de projeto de uma determinada classe. A partir deste ponto, ele derivará e mostrará graficamente, através de uma janela de diálogo, o fluxo de ferramentas que gerou aquela versão. Nesta interface, será permitido, ao usuário da aplicação que fez a requisição, indicar a ferramenta que inicia e aquela que encerra o fluxo a ser reexecutado (ver Figura 4.8). Por último, o Sistema solicita a informação dos argumentos da primeira ferramenta no fluxo e o reexecuta. Para cada argumento que constituir-se de um arquivo de dados, poder-se-á selecionar, através de um browser gráfico. uma dada versão de um determinado objeto de projeto daquela classe requerida pela ferramenta. Quando não for

[2] o formato deste arquivo é detalhado no anexo A-2

informado um destes argumentos, é utilizada a versão do objeto que está registrada na estrutura de dados para aquele ponto de projeto.

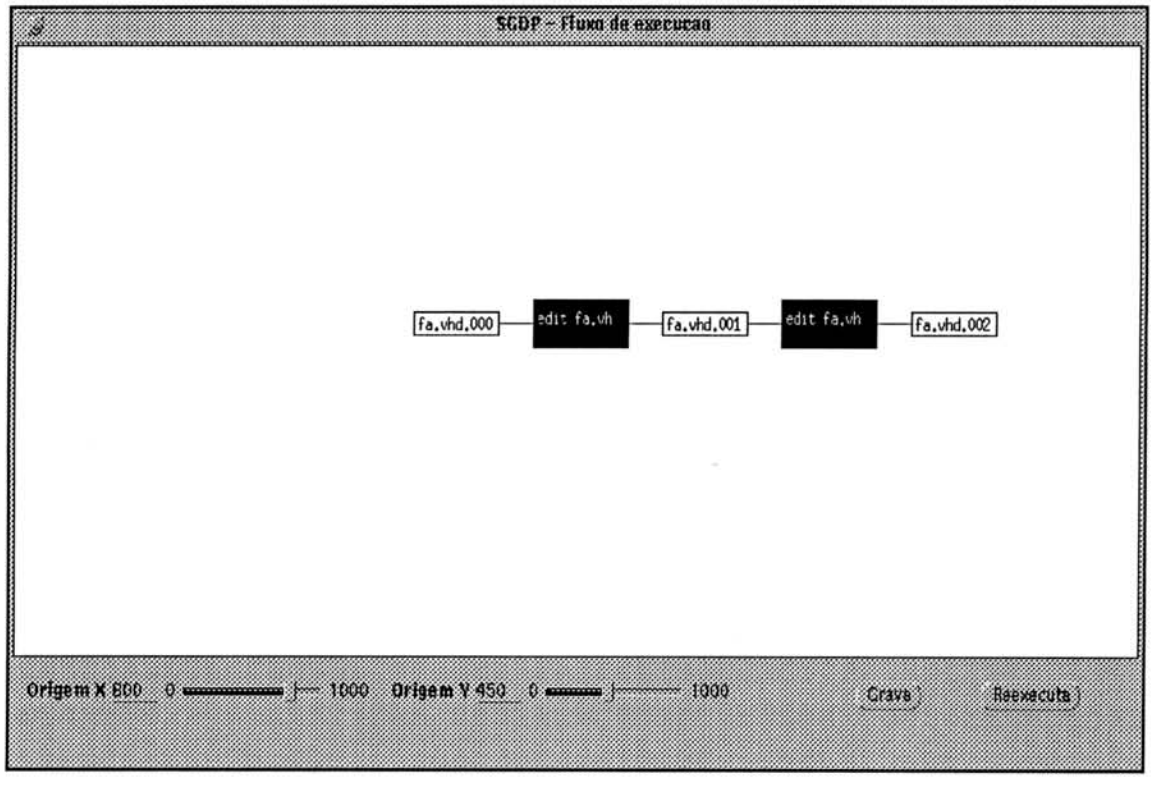


FIGURA 4.8- Tela de Seleção de Fluxo para Armazenamento ou Reexecução

4.2.5 Disponibilização de fluxos armazenados em arquivo

O SGDP permite a disponibilização de um fluxo, previamente armazenado em arquivo, para as aplicações ou outros sistemas, via uma *requisição de disponibilização de fluxo armazenado*.

Se não for informado, na requisição, o nome do arquivo, o SGDP apresenta uma janela de diálogo com um browser que lista e permite a seleção dos arquivos disponíveis.

4.2.6 Consulta aos dados gerados

A prestação deste serviço pelo SGDP se dá através da recepção de uma *requisição de consulta de dados.*

Ele apresentará, inicialmente, uma janela de diálogo com um browser que permite a seleção da classe de objeto a ser consultada (ver Figura 4.9). Num segundo momento, haverá um browser para seleção de um objeto daquela classe (ver Figura 4.10). E, por último, existirá um browser para seleção de uma versão daquele objeto de projeto (ver Figura 4.11). Na requisição, poder-se-á informar a classe e/ou objeto de projeto a ser consultado. Neste caso, o primeiro e/ou segundo browsers não serão mostrados.

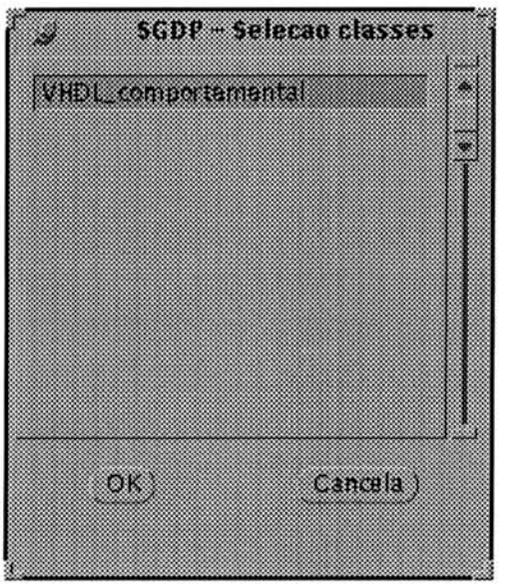


FIGURA 4.9- Tela de Seleção de Classes

4.2.7 Consulta às relações entre os dados

A prestação deste serviço se dá através de uma *requisição de consulta às relações.*

O SGDP solicitará, através de um browser gráfico, a seleção de uma versão de um objeto de projeto de uma dada classe. Com esta informação, o Sistema derivará e

mostrará graficamente o fluxo de ferramentas que gerou aquela versão, juntamente com os dados de entrada e saída para cada ferramenta, permitindo a visualização de maiores detalhes sobre estes dados, tais como data e hora de criação, tamanho, entre outros.

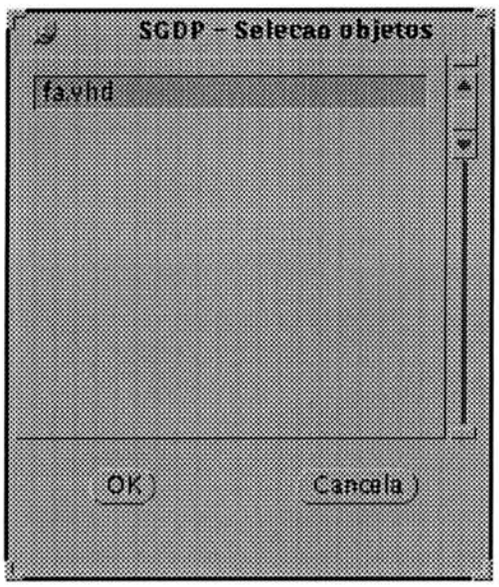


FIGURA 4.10- Tela de Seleção de Objetos

4.2.8 Obtenção de versão de objeto de projeto

Uma *requisição de obtenção de versão*, quando recebida pelo SGDP, permite a obtenção de uma dada versão de um objeto de projeto por parte de aplicações ou outros sistemas.

O Sistema mostrará uma janela de diálogo com um browser que permite a seleção da classe do objeto. Depois da seleção da classe, surgirá um browser para seleção de objetos daquela classe. E, uma vez selecionado um objeto, aparecerá um browser para seleção de uma dada versão daquele objeto de projeto. Na requisição, poder-se-á informar a classe e/ou objeto de projeto a ser consultado. Neste caso, o primeiro e/ou segundo browsers não serão mostrados. Deve-se informar na requisição o nome do arquivo onde disponibilizar a versão; caso contrário, o SGDP abrirá uma janela de diálogo para a entrada desta informação.

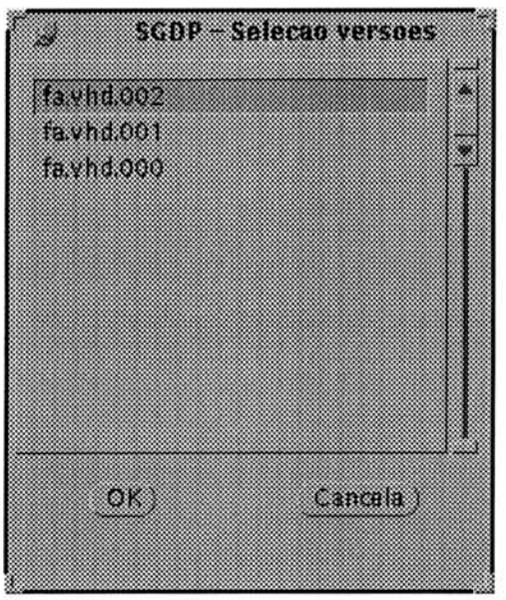


FIGURA 4.11- Tela de Seleção de Versões

4.2.9 Manutenção de estatísticas sobre a execução de ferramentas

O SGDP, para cada ferramenta, manterá informações sobre o tempo médio de CPU utilizado, bem como o tempo médio utilizado para operações de I/O. Tais informações poderão ser utilizadas por um sistema de distribuição de carga de processamento.

Como já mencionamos, as ferramentas, a serem utilizadas, devem estar catalogadas no sistema. Desta forma, haverá um registro por ferramenta. E, nestes registros, serão armazenados tais dados, que vão ser atualizados a cada nova execução da ferramenta.

4.2.10 Manutenção de dados estatísticos sobre a utilização do sistema

O SGDP manterá informações sobre a utilização do sistema como um todo, tais como:

- número de execuções com sucesso de cada ferramenta;
- número de execuções que não obtiveram sucesso;

- número de objetos de projeto de cada classe;
- número de versões de cada objeto;
- valores médios dos números acima;
- espaço em disco sendo utilizado pelas versões.

4.3 Utilização do SGDP

Outros sistemas não proprietários poderão utilizar o SGDP, no todo ou parcialmente, para complementar os serviços que oferecem aos seus usuários. Eles terão acesso às informações e serviços do SGDP através das requisições de serviço. Na prática, esta utilização se dará a partir de uma API (Application Programming Interface) que será disponibilizada. Tal API conterá três conjuntos de funções, que enviarão as requisições de serviço ao SGDP [3]:

- um conjunto para o cadastramento de ferramentas;
- um conjunto para o uso dos serviços de gerência de dados, como, por exemplo, consulta aos dados e obtenção de versões; e
- um conjunto para o uso dos serviços de gerência de projeto, como visualização de fluxos de ferramentas e reexecução.

Portanto, as aplicações ou outros sistemas deverão efetuar modificações nos seus fontes para a utilização do SGDP.

Listamos, a seguir, situações nas quais outros sistemas poderiam se beneficiar da utilização do Sistema Gerenciador de Documentação de Projeto:

[3] a descrição das funções e estruturas necessárias para o envio das requisições de serviço é feita no anexo A-2

- em primeiro lugar, podem utilizá-lo para criar e testar novas descrições de fluxos de execução de ferramentas a serem incorporados pelos seus mecanismos de gerência de projeto;

- para que seja implementada a gerência de dados, mantendo e organizando em versões todos os objetos criados, por um sistema que possua apenas o mecanismo de gerência de projeto, ou para um sistema que está sendo desenvolvido e que, ainda, não possui um mecanismo de gerência de dados. Neste ponto, mostraremos, na prática, o aproveitamento do Sistema pelo Gerenciador Automático para Projetos de Sistemas Digitais [ROY 95];

- para manter dados de contexto que possibilitem o retorno a pontos anteriores de projeto, em sistemas que não contam com esta possibilidade.

4.4 Conclusão

O Sistema Gerenciador de Documentação de Projeto usa por completo as funcionalidades disponíveis no recurso de documentação dos passos de projeto e, além disso, é extremamente versátil e flexível, podendo ser utilizado por outros sistemas para complementar os seus serviços.

E, da forma como está implementado, em conjunto com as duas aplicações apresentadas, constitui um ambiente de projeto completo, disponibilizando aos usuários projetistas serviços consistentes de gerência de dados e gerência de projeto.

Retomamos à tabela do capítulo anterior, acrescentando as características do SGDP, para efeitos de comparação.

TABELA 4.1- Uso da documentação: **SGDP x Outros Sistemas**

Áreas	**Serviços**	**CADEC**	**Cadweld**	**SMH**	**VOV**	**SGDP**
Gerência de Dados	controle de versões	não	sim	sim	sim	sim
	controle de configurações	não	não	sim	não	não
	controle de acesso aos dados	não	não	não	sim	não
	manutenção do estado - dados (p/*backtracks*)	não	sim	sim	sim	sim
Gerência de Projeto	manutenção do estado - fluxo de tarefas (p/*backtracks*)	sim	sim	sim	sim	sim
	controle do fluxo de tarefas	não	não	não	sim	sim
	disponibilização de novos fluxos de tarefas	não	não	não	sim	sim
	gerenciamento das equipes de projeto	não	não	não	sim	não
	controle de acesso às ferramentas	não	não	não	não	não
Outros	provedor de informações e serviços a usuários finais	sim	sim	sim	sim	não
	provedor de informações e serviços a outros sistemas	não	não	não	não	sim

Podemos notar, nesta tabela, que há três serviços disponíveis em outros sistemas que não são tratados pelo SGDP: o controle de configurações, o controle de acesso aos dados e o gerenciamento das equipes de projeto. Isto ocorre porque tais serviços necessitam de mecanismos de controle próprios, que vão além da utilização das informações necessárias disponíveis na documentação dos passos de projeto e, por este motivo, não foram objeto de estudo deste trabalho. De qualquer forma, no último capítulo discutimos a possível disponibilização de alguns destes serviços pelo SGDP.

5 UTILIZAÇÃO PRÁTICA DO SGDP

Neste capítulo, descrevemos as aplicações que compõem, junto com o SGDP, o ambiente de projeto apresentado no capítulo anterior. Também mostramos a utilização do SGDP pelo Gerenciador Automático para Projetos de Sistemas Digitais [ROY 95].

5.1 Aplicação A

Ao contrário do Sistema Gerenciador, a aplicação A (ver Figura 5.1) possui interface direta com o projetista. Ela é uma aplicação na qual o usuário seleciona e executa as ferramentas necessárias para a conclusão da sua atividade, e, conseqüentemente, cria e altera objetos de projeto.

O fluxo de execução de ferramentas e os dados associados vão sendo capturados pelo SGDP para a construção da estrutura de dados que implementa a documentação dos passos de projeto.

A aplicação A possibilita aos projetistas:

- execução de ferramentas, através do envio de uma *requisição de execução de ferramenta* ao SGDP. As ferramentas, é claro, já deverão estar cadastradas no Sistema. Para isso, a aplicação enviará uma *requisição de cadastro de ferramenta* ao SGDP com o nome do arquivo contendo as ferramentas a serem cadastradas;
- seleção e armazenamento em arquivo de fluxos de execução de ferramentas que obtiveram sucesso na obtenção de resultados, através de uma *requisição de seleção de fluxo*;
- consulta aos dados gerados, através de uma *requisição de consulta de dados*;

- consulta às relações entre os objetos de projeto, através de uma *requisição de consulta às relações*;
- acesso a versões anteriores de objetos de projeto, através de uma *requisição de obtenção de versão*;
- retorno a pontos anteriores do projeto para a reexecução de fluxos de ferramentas com novos (ou anteriores) objetos de projeto, através de uma *requisição de reexecução de fluxo*.

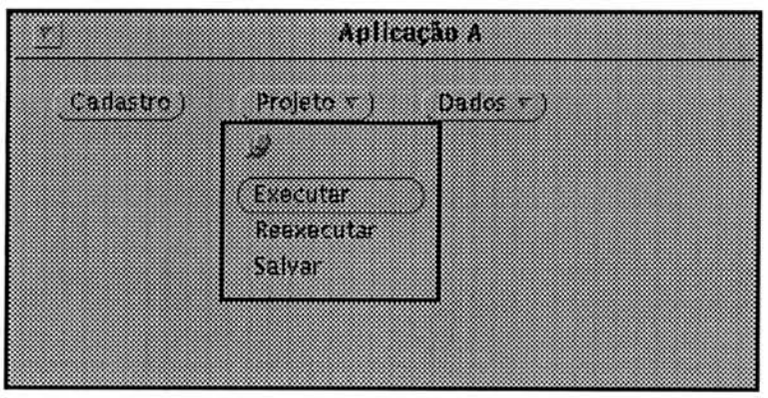


FIGURA 5.1- Tela Principal da Aplicação A

5.2 Aplicação B

A aplicação B (ver Figura 5.2) possui, igualmente, interface direta com o projetista. Distingue-se da anterior por estar amarrada a um mecanismo de controle de fluxo de ferramentas, alimentado pelas informações da biblioteca de tarefas, mantida pelo SGDP.

Como já foi visto, fluxos de execução de ferramentas que obtiveram sucesso nos seus propósitos já foram gravados, previamente, na biblioteca de tarefas pela aplicação A.

A aplicação B possibilita aos seus usuários:

- carga de arquivos da biblioteca de tarefas;
- execução de ferramentas segundo a política da tarefa carregada;
- consulta aos dados gerados, através do envio de uma *requisição de consulta de dados* ao SGDP;
- consulta às relações entre objetos de projeto, através de uma *requisição de consulta às relações*;
- acesso a versões anteriores de objetos de projeto, através de uma *requisição de obtenção de versão*.

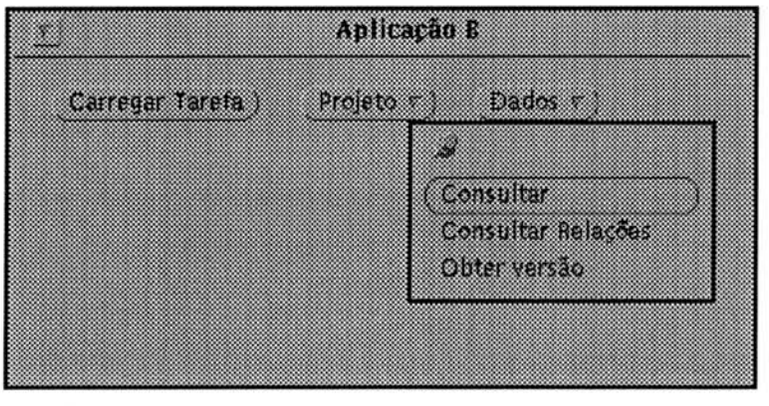


FIGURA 5.2- Tela Principal da Aplicação B

5.2.1 Carga de arquivos da biblioteca de tarefas

A aplicação B possibilita, aos seus usuários, a seleção de arquivos contendo fluxos de ferramentas previamente testados e consolidados através de uma *requisição de disponibilização de fluxo armazenado*.

Este fluxo (ou tarefa) será a base do mecanismo de controle de fluxo de ferramentas existente nesta aplicação.

5.2.2 Execução de ferramentas segundo a política da tarefa

A aplicação B possui um mecanismo de controle de fluxo de execução de ferramentas que se baseia na tarefa disponibilizada pelo SGDP após uma *requisição de disponibilização de fluxo armazenado*.

A aplicação mostra graficamente um fluxo de execução e permite, ao usuário, a entrada de argumentos para a primeira ferramenta na seqüência. Com estes dados informados, ela dispara a execução do fluxo.

No momento da entrada dos argumentos, a aplicação permite a obtenção de versões já existentes de objetos de projeto através de *requisições de obtenção de versão*.

Mesmo que a aplicação B faça, ela própria, a execução das ferramentas, ela pode, ainda, utilizar o SGDP para o registro do fluxo de execução conforme explicamos no final do item 4.2.1. Isto é fundamental para a consulta aos dados gerados e suas relações, bem como o acesso a versões de objetos de projeto.

5.3 Gerenciador Automático para Projetos de Sistemas Digitais

O Gerenciador Automático para Projetos de Sistemas Digitais (GAFT) [ROY 95] é um sistema destinado ao controle das tarefas que compõem os passos de projeto a serem realizados para a implementação de um projeto. Este controle das tarefas é feito através da análise de expressões (pré-condições e objetivos), levando-se em conta o estado atual do projeto, traduzido por valores de atributos de objetos de projeto, o que leva à formação de seqüências não fixas de passos de projeto.

As expressões de pré-condições são usadas para a análise de quais tarefas podem estar aptas para execução num dado instante, enquanto que as expressões de objetivos definem se a tarefa foi executada com sucesso ou não. O teste da obtenção

de um objetivo final (meta de projeto) é feito a cada passo e serve para identificar o encerramento do projeto.

Há duas formas possíveis para o controle das tarefas em GAFT: a forma semi-automática e a forma automática. No controle semi-automático, quando duas ou mais tarefas estão aptas para execução, a decisão sobre qual executar é repassada ao projetista. No controle automático, GAFT toma esta decisão utilizando critérios específicos para a comparação entre tarefas alternativas. O fluxo de tarefas construído pode ser visualizado graficamente.

O sistema GAFT não contempla o controle dos objetos de projeto que vão sendo produzidos, nem tampouco permite capturar uma seqüência de tarefas realizada com sucesso. Para disponibilizar tais serviços aos seus usuários, GAFT fará uso do SGDP.

5.3.1 Utilização do SGDP por GAFT

GAFT fará uso dos serviços do Sistema Gerenciador de Documentação de Projeto e, para isso, informará, a cada passo de projeto, a tarefa executada, seus dados de entrada e de saída. O SGDP, por sua vez, montará a estrutura de dados da história do projeto para, a partir dela, disponibilizar as informações e serviços requisitados por GAFT.

Quando da execução de uma tarefa, GAFT solicita ao projetista o nome do objeto a ser usado. Com a utilização do SGDP, ele dará aos seus usuários, também, a possibilidade de selecionar uma dada versão de um determinado objeto de projeto daquela classe requerida pela tarefa, criada numa rodada anterior, através do envio de uma *requisição de obtenção de versão*. A consulta aos dados gerados e suas relações são outros serviços que GAFT disponibilizará aos seus usuários através do envio das respectivas requisições de serviço ao SGDP.

GAFT não permite a reexecução de fluxos, a não ser que as expressões sejam todas reavaliadas. Isto será evitado com o armazenamento daqueles fluxos, que obtiveram sucesso nos seus propósitos e que são comumente executados, na biblioteca de tarefas do SGDP através de *requisições de seleção de fluxo*. Posteriormente, GAFT poderá acessar estas descrições, através de *requisições de disponibilização de fluxo armazenado* e executá-las sem a necessidade de avaliar pré-condições e objetivos.

6 DETALHES DE IMPLEMENTAÇÃO

Além de apresentar detalhes da implementação do SGDP e suas aplicações, o objetivo deste capítulo é descrever módulos de software que, apesar de terem sido desenvolvidos para este ambiente, poderão ser usados, devido a sua generalidade, por outros programas sem quaisquer alterações [1].

Os tópicos que abordaremos são os seguintes:

- a interface do ambiente;
- a estrutura de dados;
- a visualização do grafo;
- a comunicação entre processos.

6.1 A interface do ambiente

O ambiente de projeto, SGDP e aplicações, foi desenvolvido em linguagem C em estações de trabalho SUN sobre o sistema operacional UNIX.

Todos os elementos de interface (janelas, popups, menus, botões, ...) foram construídos através do utilitário **guide**, que é um construtor de interfaces no padrão OPEN LOOK. Ele permite a definição, de forma interativa, da interface de um sistema, gerando, para a realização desta definição, o correspondente código fonte em linguagem C, ao qual o programador deverá anexar o seu código específico para a implementação da funcionalidade daquele sistema.

---

[1] o anexo A-1 dá a indicação da localização destes módulos na rede das estações de trabalho do Instituto de Informática

Como todo utilitário de propósitos gerais, o **guide** dificulta implementações de certas peculiaridades que um sistema, normalmente, necessita. Nestes casos, recomenda-se a consulta ao XView Programming Manual [HEL 90]. Se a resposta não for encontrada, o que é comum, sugere-se, então, a consulta ao Xview Version 2 Reference Manual [SUN 90]. Apenas para exemplificar, a definição de um canvas (área de trabalho) com dimensões superiores à da janela base que o comporta não pode ser feita apenas com o **guide**.

6.2 A estrutura de dados

A base da estrutura de dados do sistema é aquela que possibilita a criação de grafos. Em termos de programação, os nós de um grafo estarão dispostos numa estrutura de lista, onde cada nó será representado por uma estrutura da linguagem C. Nesta estrutura, um nó possui ponteiros para os seus antecessores, para os seus sucessores, e para as informações que ele mantém, no caso informações sobre dados (versões de objetos de projeto) ou ferramentas. Isto pode ser visualizado na Figura 6.1 que mostra a estrutura de dados para o grafo da Figura 6.2

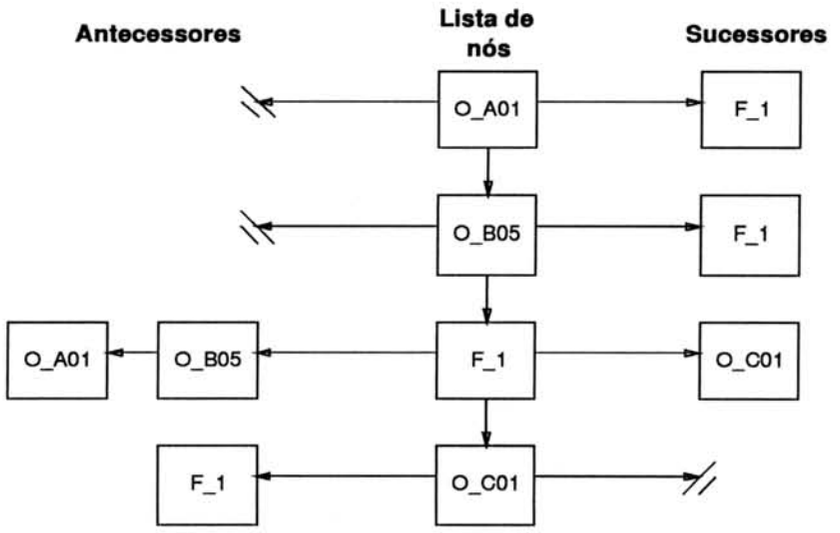


FIGURA 6.1- Estrutura de dados representando um grafo

As ocorrências de F_1 (que significa Ferramenta 1, enquanto O_A01 significa Objeto A versão 01), na Figura 6.1, que são antecessores ou sucessores de outros nós, mantêm apenas a referência (endereço) para o respectivo nó na lista de nós, ou seja, não são ocorrências distintas daquela na lista de nós. As informações, mantidas pelos nós, não estão representadas na figura.

Pelo fato de as informações específicas da aplicação (sobre dados ou ferramentas, neste caso) não serem armazenadas diretamente na estrutura da linguagem C que representa o nó (há apenas uma referência a estas informações através de um ponteiro, ou seja, através do seu endereço), a estrutura de dados para a criação de grafos torna-se totalmemente genérica.

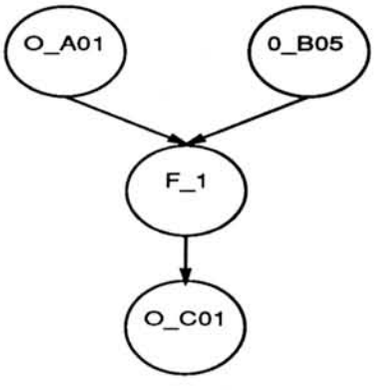
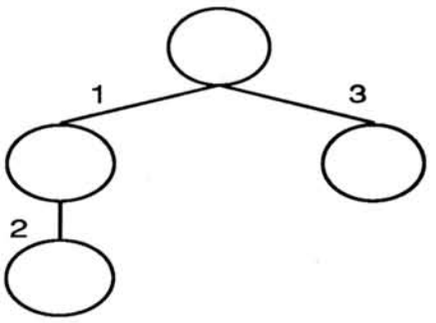


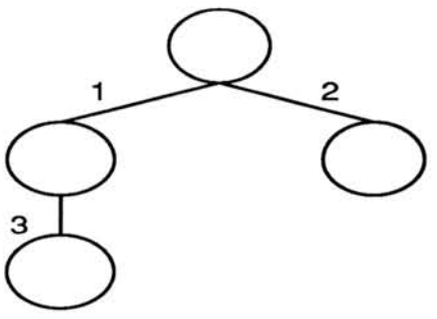
FIGURA 6.2- Grafo representando a execução de uma ferramenta

Há a possibilidade de se percorrer o grafo tanto em largura quanto em profundidade (ver Figura 6.3 - os números indicam a seqüência de nodos percorrida), para a frente ou para trás, e de se definir o número de níveis que devem ser percorridos. No caso de percurso em profundidade, pode-se optar pela continuação ou não do percurso (retorno) quando se encontra um nó folha (ou seja, sem sucessores).

Sobre a estrutura de grafo, foram desenvolvidas outras três estruturas de listas para a representação de classes, objetos e versões, organizadas da seguinte forma: no topo da hierarquia, existe a lista de classes, na qual cada elemento, isto é, cada classe, aponta para uma lista de objetos daquela classe, e, por sua vez, cada objeto desta lista aponta para uma lista de versões daquele objeto. Neste último nível, dá-se a conexão com o grafo:



Percurso em Profundidade



Percurso em Largura

FIGURA 6.3- Percursos

cada versão aponta para o nó que a representa. Percorrendo, então, o grafo, a partir deste nó para trás, teremos o fluxo de ferramentas, bem como as versões de outros objetos que originaram esta versão, conforme Figura 6.4.

6.3 A visualização do grafo

A visualização do grafo pressupõe uma varredura na respectiva estrutura de dados. Como o grafo, que se deseja visualizar no SGDP, é aquele que originou uma dada versão de um objeto de projeto, a varredura se dá através de um percurso em largura, a partir do nó que representa esta versão, para trás. Durante o percurso, os nós visitados são armazenados numa estrutura de fila, onde são mantidas, também, as seguintes informações: o número do nó (que é atribuído seqüencialmente), o nível do nó (nível 0 corresponde ao

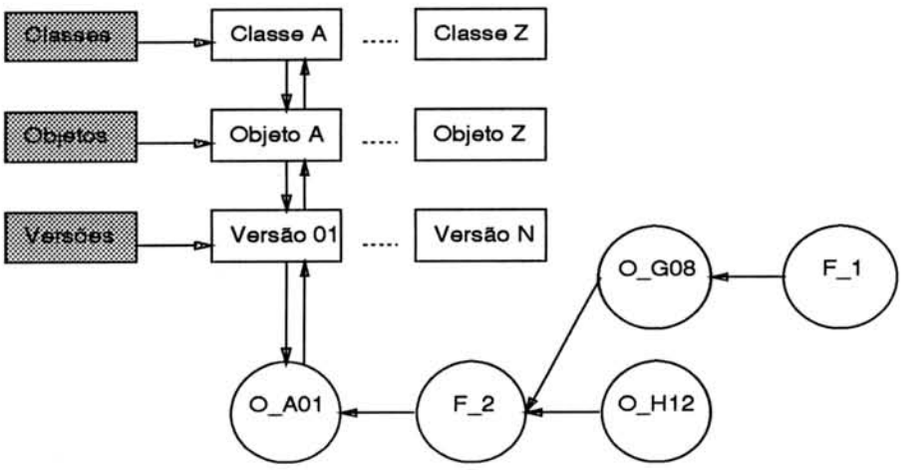


FIGURA 6.4- Conexão com o grafo

primeiro nível percorrido) e o número do nó antecessor imediato (para a interligação do grafo).

Para o desenho do grafo, utilizam-se técnicas de geração de imagens em duas dimensões [HEA 86]. Inicialmente, definimos um Universo de 1000 x 1000 dividido em inúmeras células, e uma janela de visualização, conforme Figura 6.5. Criamos, a seguir, os objetos a serem visualizados a partir da fila de nós gerada durante o percurso no grafo, de forma que existirá um objeto para cada nó. Além das coordenadas de Universo, cada objeto detém as informações de número do nó e número do nó antecessor imediato.

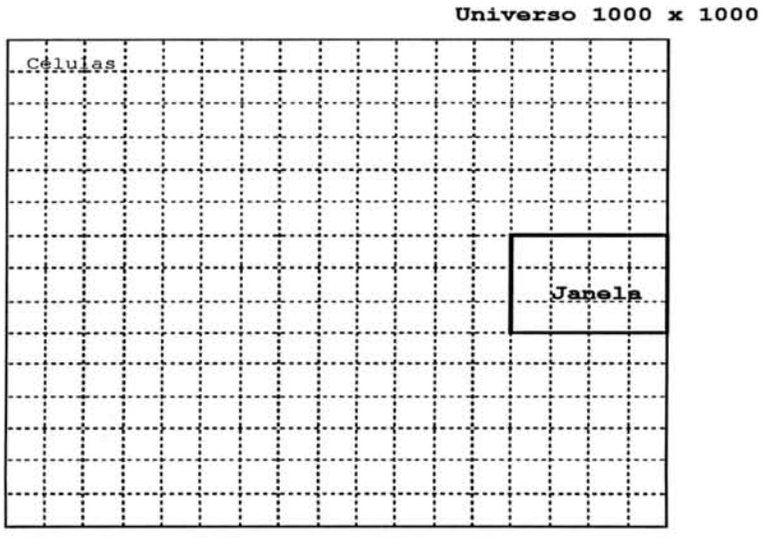


FIGURA 6.5- Universo, células, janela de visualização

As coordenadas do objeto são calculadas a partir da definição da célula em que será posicionado, conforme as regras a seguir (ver Figura 6.6 que apresenta a disposição do grafo da Figura 6.2 na janela de visualização):

- todos os nós de um mesmo nível ficam numa mesma coluna de células;
- quanto menor o nível do nó, mais à direita está a coluna que o contém;
- os nós de uma mesma coluna são dispostos de forma a ocupar as células centrais da respectiva coluna.

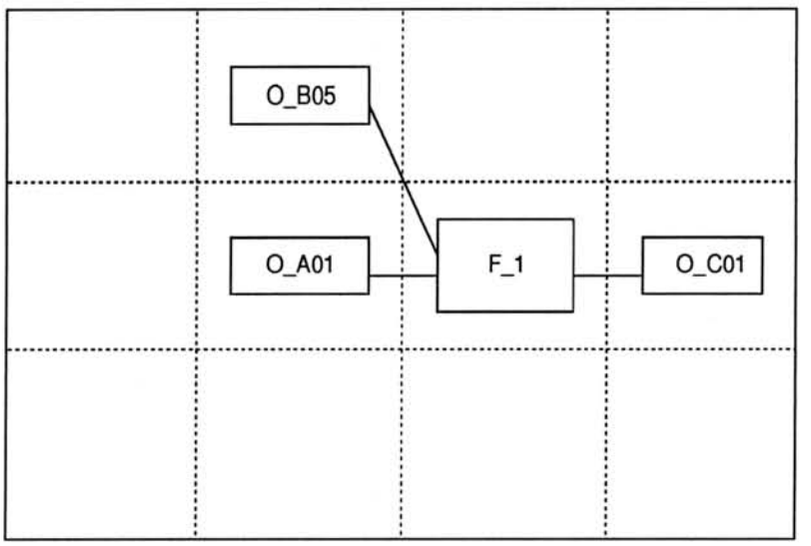


FIGURA 6.6- Definição de células para o desenho do grafo

Já as informações de número do nó e número do nó antecessor imediato, armazenadas em cada objeto, são usadas para a geração dos arcos que interligam os nós.

Por último, as coordenadas de Universo dos objetos são transformadas em coordenadas de tela, tendo como base a janela de visualização (mapeamento para Viewport, onde a Viewport é representada pelo canvas, criado na definição da interface), e estes são desenhados juntamente com os arcos.

Apesar de algumas particularidades, as funções de mapeamento (Universo x Viewport) e desenho poderão ser aproveitadas por outros sistemas.

6.4 A comunicação entre processos

O SGDP e suas duas aplicações constituem três processos distintos executando sobre o UNIX. Como estes processos precisam se comunicar (o SGDP recebe e atende as requisições de serviço provenientes das aplicações), houve a necessidade de escolher uma das formas de comunicação entre processos disponíveis neste sistema operacional. Devido a sua maior flexibilidade, a escolha recaiu sobre o mecanismo de memória compartilhada.

O mecanismo de memória compartilhada permite mapear o mesmo bloco de endereçamento real em dois ou mais espaços de endereçamento de processo. Para administrar o acesso a esta memória compartilhada, utilizamos semáforos, que constituem um método popular para controlar o acesso a um determinado recurso. Normalmente, um semáforo é um valor binário, que indica se o recurso está disponível ou não.

Nesta implementação do ambiente de projeto, há um semáforo que permite o acesso à região de memória compartilhada a um único processo de cada vez. A memória compartilhada, por sua vez, é dividida em duas áreas: uma área de controle e uma área de dados. A área de controle mantém informações do tipo: quantas mensagens existem para serem lidas, há ou não espaço para a inserção de uma mensagem, endereço onde deverá ser posicionada uma nova mensagem, entre outras. E a área de dados armazena as mensagens propriamente ditas, ou seja, as requisições de serviço.

O SGDP, quando da sua execução, cria a região de memória compartilhada e o semáforo e, constantemente, acessa-a para ver se há alguma requisição para ser atendida. As aplicações, bem como outros sistemas que utilizarem o SGDP, associam-se a esta memória compartilhada e, nela, depositam as requisições de serviço. Na finalização do SGDP, ele destrói a memória compartilhada e o semáforo.

7 CONCLUSÃO

A documentação dos passos de projeto é um recurso fundamental dentro de um framework. A exploração do espaço de projeto, por exemplo, que vem a ser o retorno a um ponto anterior do projeto para o teste de alternativas e que é de extrema importância para a obtenção dos melhores resultados, não poderia ser feita sem a manutenção do estado do projeto (ferramentas utilizadas e dados criados) na documentação dos passos de projeto.

O SGDP é um sistema que explora a potencialidade da documentação dos passos de projeto, transformando-a em informações e serviços aos seus clientes. Um ponto importante é que estas informações e serviços estão organizados de uma forma totalmente modular, o que possibilita aos clientes o aproveitamento apenas daqueles que lhe são necessários. Por exemplo, um cliente pode utilizar somente os serviços relacionados com a gerência de dados.

Agora, o grande avanço deste trabalho, com relação aos demais, advém do fato de que ele está implementado de forma a interagir com outros sistemas, e não diretamente com os usuários finais. Estes outros sistemas podem, seletivamente, fazer uso das informações e serviços do SGDP para complementar sua funcionalidade, tornando-se mais abrangentes e poderosos, sem deixarem de ser "enxutos", no sentido de que não precisam comportar o código para o desempenho daquelas funções. Este código está no SGDP e os sistemas precisam apenas ter disponível a API que comunica com o SGDP, enviando-lhe as requisições de serviço.

Apesar desta característica única de interação com outros sistemas, tivemos a preocupação de atender também aos usuários finais, os projetistas. O SGDP, em conjunto com as duas aplicações que foram construídas, constitui um ambiente completo que permite:

- a execução de ferramentas, com ou sem uma política de controle de fluxo;
- a visualização dos fluxos de ferramentas utilizados;

- a disponibilização de descrições de fluxos de ferramentas validados para a incorporação por mecanismos de controle do fluxo de tarefas. A aplicação A, conjuntamente com o SGDP, pode ser utilizada com esta função de criação e teste de novos fluxos por construtores de ambiente de outros sistemas;
- a reexecução de fluxos;
- a consulta aos dados gerados e às suas relações;
- a obtenção de versões anteriores de objetos de projeto.

7.1 Propostas de implementações futuras

O SGDP explora praticamente toda a potencialidade da documentação dos passos de projeto, de forma que novos serviços podem, ainda, ser disponibilizados. Em VOV (ver capítulo 3), por exemplo, a documentação foi utilizada para o controle de acesso aos dados.

Nas seções que seguem, listamos alguns novos serviços que poderiam estar disponíveis no SGDP. Podemos, inclusive, adiantar que a inclusão destes serviços não será uma tarefa complexa, devido à forma como o Sistema foi organizado e implementado.

7.1.1 Compactação de versões

Os objetos de projeto, em SGDP, são arquivos. Conseqüentemente, cada versão de um objeto de projeto constitui um arquivo em disco. Isto faz com que um grande espaço em disco seja utilizado para o armazenamento das versões. Um aperfeiçoamento importante no sistema, que diminuiria, consideravelmente, este espaço utilizado, seria a utilização de um algoritmo de compactação de dados. Uma versão de um objeto,

ao ser criada, seria compactada para armazenamento. Quando fosse utilizada por uma ferramenta, seria descompactada.

7.1.2 Classificação de versões

Os objetos de projeto de uma classe, assim como as versões destes objetos, são classificados em ordem temporal quando são apresentados ao projetista numa consulta. Seria muito interessante que outras políticas de classificação estivessem disponíveis para escolha por parte do projetista. Isto poderia ser facilmente implementado associando-se, às classes de objetos, ferramentas que fizessem avaliações nos objetos desta classe e devolvessem valores que permitiriam a classificação (valores para a área consumida, por exemplo).

7.1.3 Controle de configurações

Um mecanismo de controle de configurações poderia ser criado para o tratamento de representações compostas de objetos de projeto.

As versões dos sub-objetos seriam capturadas na documentação, que, no caso de configurações dinâmicas, apresentaria a possibilidade de se escolher versões com características bem específicas, conforme a política de classificação adotada (com restrições de área, por exemplo), como visto no tópico anterior.

7.1.4 Conceito de projeto e projetistas

O SGDP gerencia uma única estrutura de dados da história do projeto por vez. Esta limitação seria evitada com a criação do conceito de projeto, de forma que cada projeto teria uma estrutura de dados gerenciada pelo sistema. Sob este conceito de projeto,

poderíamos criar, também, o conceito de projetista para a implementação de políticas de controle de acesso, como veremos a seguir.

7.1.5 Controle de acesso aos dados e às ferramentas

O conceito de projetista, proposto no item anterior, poderia ser utilizado para controlar o acesso aos dados e às ferramentas. Assim, para cada classe de objeto e para cada ferramenta, poder-se-ia determinar quais projetistas estariam habilitados a utilizá-los. Um aperfeiçoamento ainda maior seria o estabelecimento de classes de projetistas.

7.1.6 Fluxos de ferramentas que geram uma classe de objetos

Os fluxos de execução de ferramentas, que podem ser visualizados em SGDP, são aqueles que foram utilizados para a criação de uma determinada versão de um objeto de projeto, ou seja, existe um fluxo para cada versão armazenada no sistema. Com certeza, estes fluxos se repetem na criação das versões de objetos de uma dada classe. Uma informação adicional importante, então, seria a disponibilização daqueles fluxos de ferramentas que geram os objetos de uma classe específica.

Tal característica poderia ser implementada através de uma varredura nos fluxos que geraram as versões dos objetos daquela classe de interesse e a manutenção de um exemplar de cada fluxo distinto que tenha sido utilizado.

ANEXO A1 LOCALIZAÇÃO DOS CÓDIGOS FONTE DE SGDP

O ambiente de projeto, SGDP e aplicações, foi desenvolvido em linguagem C nas estações SUN da rede das estrelas do Laboratório de Informática, sobre o sistema operacional UNIX.

O código fonte com as funções para a criação e manutenção de grafos está no diretório /home/sirius/sandrons/sgdp com o nome de grafo.c.

O código fonte com as funções para o tratamento e desenho de objetos bidimensionais está no diretório /home/sirius/sandrons/sgdp com o nome de des2d.c.

O código fonte com as funções para a comunicação entre processos está no diretório /home/sirius/sandrons/sgdp com o nome de com_proc.c.

Os módulos executáveis encontram-se no diretório /home/sirius/sandrons/bin com os seguintes nomes: **sgdp**, **aplic_a** e **aplic_b**.

ANEXO A2 DESCRIÇÃO DE FUNÇÕES, ESTRUTURAS E FORMATOS DE ARQUIVO PARA UTILIZAÇÃO DO SGDP

Neste anexo, apresentamos os protótipos das funções necessárias para a utilização do SGDP e a definição das estruturas que devem ser preenchidas para cada requisição de serviço. Listamos, também, o formato dos arquivos para cadastro de ferramentas e para disponibilização de fluxos de execução (arquivos da biblioteca de tarefas).

As funções estão disponíveis na biblioteca sgdplib.a e o arquivo de cabeçalho, com a definição das estruturas, no arquivo sgdp.h, ambos encontrados no diretório /home/sirius/sandrons/sgdp.

A2.1 Estrutura e protótipo de função para cadastro de ferramenta

```
struct mesg_sgdp_cadastro {
        char nome_arq [ 255];
};
```

FIGURA A2.1- Estrutura utilizada numa *requisição de cadastro de ferramenta*

```
void
req_sgdp_cadastro ( struct mesg_sgdp_cadastro *) ;
```

FIGURA A2.2- Protótipo da função para *requisição de cadastro de ferramenta*

A2.2 Estruturas e protótipos de funções para uso dos serviços de gerência de projeto

```
struct mesg_dado {
        char classe [ 255];
        char objeto [ 255];          /* Nome do arquivo */
        char obj_dir [ 255];         /* Diretorio do arquivo */
};

struct mesg_ferramenta {
        char ferramenta [ 255];
};

struct mesg_sgdp_gp_exec {
        char ferramenta [ 255];
};

struct mesg_versao {
        char classe [ 255];
        char objeto [ 255];
};

struct mesg_sgdp_gp_sel {
        char classe [ 255];
        char objeto [ 255];
        char nome_arq [ 255];
};

struct mesg_sgdp_gp_biblioteca {
        char nome_arq [ 255];
};
```

FIGURA A2.3- Estruturas utilizadas para uso dos serviços de gerência de projeto

```
/* Funcao utilizada para informar dados de entrada
* Chamada por sistemas que executam suas ferramentas
*/
void
msg_dado_ent ( struct mesg_dado *);

/* Funcao utilizada para informar dados de saida
* Chamada por sistemas que executam suas ferramentas
*/
void
msg_dado_sai ( struct mesg_dado *) ;

/* Funcao utilizada para informar ferramenta executada
* Chamada por sistemas que executam suas ferramentas
*/
void
msg_ferramenta ( struct mesg_ferramenta *) ;

/* Funcao utilizada para requisitar a execucao de
* uma ferramenta
*/
void
req_sgdp_gp_exec ( struct mesg_sgdp_gp_exec *);

/* Funcao utilizada para requisitar a visualizacao de um
* fluxo
*/
void
req_sgdp_gp_vis ( struct mesg_versao *);

/* Funcao utilizada para requisitar a selecao de um fluxo
*/
void
req_sgdp_gp_sel  ( struct mesg_sgdp_gp_sel *);

/* Funcao utilizada para requisitar a reexecucao de um
* fluxo
*/
void
req_sgdp_gp_reexec ( struct mesg_versao *);

/* Funcao utilizada para requisitar a disponibilizacao de
* um arquivo da biblioteca de tarefas
*/
void
req_sgdp_gp_biblioteca ( struct mesg_sgdp_gp_biblioteca *);
```

FIGURA A2.4- Protótipo das funções para uso dos serviços de gerência de projeto

A2.3 Estruturas e protótipos de funções para uso dos serviços de gerência de dados

```
struct mesg_sgdp_gd_get {
        char classe [ 255];
        char objeto [ 255];
        char nome_arq [ 255];
};
```

FIGURA A2.5- Estruturas utilizadas para uso dos serviços de gerência de dados

```
/* Funcao utilizada para requisitar consulta aos dados
*/
void
req_sgdp_gd_consulta ( struct mesg_versao *);

/* Funcao utilizada para requisitar consulta as relacoes
*  entre os dados
*/
void
req_sgdp_gd_relacoes ( struct mesg_versao *);

/* Funcao utilizada para requisitar a obtencao de uma
*  versao
*/
void
req_sgdp_gd_get ( struct mesg_sgdp_gd_get *);
```

FIGURA A2.6- Protótipo das funções para uso dos serviços de gerência de dados

A2.4 Formato do arquivo de cadastro

O cadastro de ferramentas, conforme visto no capítulo 4, também pode ser efetuado através de um arquivo texto, cujo nome vem informado numa *requisição de cadastro de ferramenta*. Neste arquivo, devem constar as mesmas informações que são digitadas no cadastro interativo. Cada linha descreve uma ferramenta ou um argumento para a ferramenta mais próxima (linha anterior mais próxima que descreve uma ferramenta).

A primeira linha do arquivo deve conter as seguintes informações, separadas por vírgula: nome da ferramenta, diretório de execução, diretório de trabalho e número de argumentos. As linhas seguintes (uma ou mais, de forma que há uma linha para cada argumento) definem os argumentos para esta ferramenta, contendo as seguintes informações, separadas por vírgula: delimitador, tipo, se é opcional ou obrigatório, classe, se é dado de entrada, saída ou ambos e, por último, caracter separador. No caso de se cadastrar várias ferramentas, esta seqüência se repete.

Qualquer informação nula deve ser identificada pelo literal NULL. A seguir, listamos outros literais que devem ser utilizados:

- INTEIRO, REAL, ARQUIVO, STRING - identificam, respectivamente, os tipos de argumento inteiro, real, nome de arquivo e string (cadeia de caracteres);
- OPCIONAL, OBRIGATORIO - expressam a obrigatoriedade ou não do argumento;
- IN, OUT, INOUT - identicam a condição do dado: de entrada, saída ou ambos.

A seguir, mostramos o exemplo de um arquivo de cadastro para a ferramenta *edit*, cujo argumento é o nome de um arquivo texto que será editado:

```
edit, \bin, \edit\trab, 1
NULL, ARQUIVO, OBRIGATORIO, TEXTO, INOUT, NULL
```

FIGURA A2.7- Exemplo de arquivo de cadastro

A2.5 Formato do arquivo para disponibilização de fluxos de execução

Um arquivo da biblioteca de tarefas, contendo a descrição de um fluxo de execução de ferramentas validado, também possui um formato texto e suas linhas identificam, nesta ordem, dados de entrada, ferramenta executada e dados de saída, sendo que estes dados de saída constituirão os dados de entrada da ferramenta seguinte, e assim sucessivamente até o encerramento do fluxo.

As linhas que identificam dados iniciam com o caracter 'D', englobando as seguintes informações: classe do objeto, nome do objeto, versão e diretório onde está armazenado. As linhas que identificam ferramentas executadas iniciam com o caracter 'F' e contêm a linha de comando para execução das ferramentas. Linhas que identificam dados e antecedem uma linha identificando ferramenta executada representam os dados de entrada para esta ferramenta. Linhas que identificam dados e precedem uma linha identificando ferramenta executada representam os dados de saída para esta ferramenta.

Adicionalmente à disponibilização deste arquivo, quando a execução das ferramentas for efetuada pelo SGDP, será disponibilizado outro arquivo de mesmo nome, mas com extensão "cad", contendo o cadastro das ferramentas utilizadas no fluxo com formato idêntico ao de um arquivo de cadastro, conforme descrito na seção anterior.

Segue o arquivo representando duas execuções da ferramenta *edit* utilizando um mesmo objeto de projeto (arquivo contendo código fonte em linguagem C de nome fa.c):

```
D TEXTO, fa.c, fa.c.000, /sgdp/versoes/fa.c
F edit fa.c
D TEXTO, fa.c, fa.c.001, /sgdp/versoes/fa.c
F edit fa.c
D TEXTO, fa.c, fa.c.002. /sgdp/versoes/fa.c
```

FIGURA A2.8- Exemplo de arquivo para disponibilização de fluxo

BIBLIOGRAFIA

[BOS 91] BOSCH, K.O.ten; BINGLEY, P.; WOLF, P.van der. Design Flow Management in the NELSIS CAD Framework. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 28., 1991, San Francisco, USA. **Proceedings...** [S.l.]: IEEE, 1991. p.711-716.

[BRE 90] BRETSCHNEIDER, F. et al. Knowledge Based Design Flow Management. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, Nov., 1990, Santa Clara, USA. **Proceedings...** [S.l.]: IEEE, 1990. p.350-353.

[BRO 92] BROCKMAN, J.B.; DIRECTOR, S.W. A Schema_Based Approach to CAD Task Management. In: IFIP WORKSHOP ON ELETRONIC DESIGN AUTOMATION FRAMEWORKS, 1992, [S.l.]. **Proceedings...** Amsterdam: North-Holland, 1992. p.71-83.

[CAS 90] CASOTTO, A. et al.. Design Management Based on Design Traces. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 27., Jun., 1990, Orlando, USA. **Proceedings...** [S.l.]: IEEE, 1990. p.136-141.

[CHI 91] CHIUEH, T.; KATZ, R.H.; KING, V. A History Framework for Managing VLSI Design Process. In: IFIP WORKSHOP ON ELETRONIC DESIGN AUTOMATION FRAMEWORKS, 1990, [S.l.]. **Proceedings...** Amsterdam: North-Holland, 1991. p.155-170.

[CHU 90] CHUNG, M.J.; KIM, S. An Object-Oriented VHDL Design Environment. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 27., Jun., 1990, Orlando, USA. **Proceedings...** [S.l.]: IEEE, 1990. p.431-436.

[DAN 89] DANIEL, J.;Director,S.W. An Object Oriented Approach to CAD Tool Control Within a Design Framework. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 26., 1989. **Proceedings...** [S.l.]: IEEE, 1989. p.197-202.

[GAJ 92] GAJSKI,D.D; KUHN,R.H. Guest editor's introduction: New VLSI tools. **Computer**, New York, v.16, n. 12, p.11-14, Out. 1983.

[HAR 90] HARRISON, D.S. et al. Eletronic CAD Frameworks. **Proceedings of the IEEE**, New York, v. 78, n. 2, p.393-417, Feb. 1990.

[HEA 86] HEARN, Donald; BAKER, M. Pauline. **Computer Graphics**. New York: Prentice-Hall, 1986.

[HEL 90] HELLER, D. **XView Programming Manual**. Sebastopol: O'Reilly Associates, 1990. 642p.

[IEE 88] IEEE. **IEEE Standard VHDL Language Reference Manual.** New York: IEEE, 1988.

[KEM 92] KEMPER, F. et al. A Knowledge-Based Design Consultant: Model and Architecture. In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEMS SCIENCE, 1992, Honolulu, USA. **Proceedings...** [S.l.:s.n.],1992.

[KNA 86] KNAPP, D.W.; PARKER, A.C. A Design Utility Manager: the ADAM Planning Engine. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 23., 1986, Las Vegas, USA. **Proceedings...** [S.l.]: IEEE, 1986. p.48-54.

[KWE 92] KWEE-CHRISTOPH E. et al. The CADEC VLSI Design Support Methodology. In: International COMP EURO'92 Conference on Computer Systems and Software Engineering, The Hague, Netherlands **Proceedings...** [S.l.]: IEEE, 1992.

[ROY 94] ROYES G.F. **Uma Análise Comparativa de Frameworks para Projeto de Sistemas Digitais.** Porto Alegre, Brasil: CPGCC da UFRGS, 1994. (TI-378].

[ROY 95] ROYES G.F. **Um Gerenciador Automático para Projetos de Sistemas Digitais.** Porto ALegre, Brasil: CPGCC da UFRGS, 1995. Dissertação de Mestrado.

[SOA 94] SOARES S.N.. **Um Estudo sobre Gerência de Projeto em Frameworks:** Problemas e Soluções. Porto Alegre: CPGCC da UFRGS, 1994. (TI-380).

[SUN 90] SUN MICROSYSTEMS. **XView Version 2 Reference Manual:** Converting Sunview Applications. Mountain View: Sun Microsystems, 1990. 157p.

[VEL 83] VELOSO, Paulo et al. **Estruturas de Dados**. Rio de Janeiro: Campus, 1983.

[WAG 92] WAGNER, F.R. et al. Design Version Management in the STAR Framework. In: IFIP WORKSHOP ON ELETRONIC DESIGN AUTOMATION FRAMEWORKS, Mar., 1992, Bad Lippspringe, Germany. **Proceedings...** Amsterdam: North-Holland, 1992. p.85-97.

[WAG 94] WAGNER, F.R. Ambientes de Projeto de Sistemas Eletrônicos. In: ESCOLA DE COMPUTAÇÃO, 9., 1994, Recife, Brasil. **Anais...** Recife: UFPE, 1994. 155p.

[WOL 94] WOLF, P. van der. **CAD Frameworks:** Principles and Architecture. The Netherlands: Kluwer Academic Publishers, 1994. p.1-34.

Informática  UFRGS

CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Sistema Gerenciador de Documentação de Projeto

por

Sandro Neves Soares

Dissertação apresentada aos Senhores:

Profa. Dra. Lia Goldstein Golendziner

Profa. Dra. Taisy Silva Weber

Prof. Dr. Ney Laert Vilar Calazans (PUCRS)

Vista e permitida a impressão.
Porto Alegre, 05 / 07 / 96 .

Prof. Dr. Flávio Rech Wagner,
Orientador.

Prof. Flávio Rech Wagner
Coordenador do Curso de Pós Graduação
em Ciência da Computação - CPGCC
Instituto de Informática - UFRGS