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

Gerente de configurações para o ambiente STAR

por

Helena Grazziotin Ribeiro

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

Prof. Lia Goldstein Golendziner
Orientador

Prof. Flávio Rech Wagner
Co-orientador

Porto Alegre, abril de 1993.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Ribeiro, Helena Grazziotin

Gerente de configurações para o ambiente STAR / Helena Grazziotin Ribeiro.—Porto Alegre: CPGCC da UFRGS, 1993.

94 p.: il.

Dissertação (mestrado)—Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, 1993. Orientador: Golendziner, Lia Goldstein; Co-orientador: Wagner, Flávio Rech

Dissertação: Bancos de Dados para Aplicações de Projeto, CAD para Sistemas Digitais, Gerência de Configurações, Versões, Ambientes de Projeto

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Ribeiro, Helena Grazziotin

Gerente de configurações para o ambiente STAR / Helena Grazziotin Ribeiro.—Porto Alegre: CPGCC da UFRGS, 1993.

94 p.: il.

Dissertação (mestrado)—Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, 1993. Orientador: Golendziner, Lia Goldstein; Co-orientador: Wagner, Flávio Rech

Dissertação: Bancos de Dados para Aplicações de Projeto, CAD para Sistemas Digitais, Gerência de Configurações, Versões, Ambientes de Projeto

AGRADECIMENTOS

Gostaria muito de agradecer a dedicação e paciência dos meus orientadores, Lia e Flávio, que realmente foram orientadores com todas as letras maiúsculas, e que me permitiram ver a evolução do trabalho dia a dia.

Um outro agradecimento especial ao Alexandre, que acima de tudo soube compreender, amparar, e ter paciência, muita paciência, e sem o apoio de quem tudo seria bem mais difícil, muitas vezes realmente impossível. A ele, meu grande companheiro, todo o meu amor.

À minha família, em especial aos meus pais, Pedro Jorge e Noelsi, que sempre estiveram ao meu lado, e cuja preseça e apoio foi fundamental em muitos momentos.

A todos os companheiros de trabalho e amigos, Miguel, Ronaldo, Paulo, Dóris, André, Jean, Sandra entre tantos outros pelas saudáveis discussões e brainstorms, pelo carinho e pela amizade.

SUMÁRIO

LISTA DE ABREVIATURAS

DDM	Valid Design Data Management Framework
MVC	Modelo de Versões e Configurações
UC	Unidade de Controle
UCP	Unidade Central de Processamento
ULA	Unidade Lógica e Aritmética

LISTA DE FIGURAS

RESUMO

Este trabalho apresenta os mecanismos de gerência de configurações para o ambiente STAR. STAR é uma plataforma para o desenvolvimento de ambientes para projetos de circuitos e sistemas eletrônicos que está sendo desenvolvido na Universidade Federal do Rio Grande do Sul em cooperação com o Centro Científico da IBM no Rio de Janeiro. Seus objetos de projeto caracterizam-se como sistemas complexos e são representados através de um modelo de dados hierárquico, que tem por base a composição de objetos. Para expressar a evolução dos objetos de projeto no tempo utiliza-se versões, que mantêm as descrições dos objetos num determinado instante de tempo. O mecanismo de gerência de versões é fortemente relacionado à representação dos dados, suportando as diferentes dimensões que essa representação permite: visões, alternativas e revisões.

A utilização de versões associada à composição de objetos faz com que se possa ter diversas possibilidades de descrição para um mesmo sistema complexo, em função da combinação das versões. Para que se possa submeter um objeto de projeto a uma ferramenta, como um simulador, é preciso selecionar versões de modo a obter uma descrição única, que é a sua configuração. A existência de um gerente de configurações dá agilidade a essa tarefa, pois ele oferece recursos para facilitar e tornar mais rápida a construção de configurações, através de manipulação e consultas a informações obtidas junto ao ambiente sobre os objetos de projeto.

As configurações no STAR são determinadas a partir do atributo de referência dos componentes ou através da definição de um objeto-configuração. O gerente de configurações proposto para o ambiente neste trabalho suporta o estabelecimento de configurações estáticas, dinâmicas e abertas. Elas podem ser estabelecidas manualmente, através de escolhas do usuário, automaticamente, através da escolha entre um dos critérios pré-estabelecidos, ou de modo semi-automático, através da definição de uma expressão de configuração. Os critérios pré-estabelecidos têm por base a versão corrente e a versão mais recente. A utilização de expressões de

configuração permite que se selecione versões com mais objetividade, uma vez que sua construção é feita a partir de características dos objetos, dadas por seus atributos, que permitem restringir as versões selecionadas àquelas cujas características são desejadas. A linguagem que permite a definição de expressões de configuração é um dos recursos estabelecidos que facilitam a tarefa do usuário.

Outro recurso provido é a possibilidade de armazenar configurações. Isso torna possível sua reutilização em outros momentos e também por outros objetos, e preserva a flexibilidade de mantê-las como dinâmicas, ou abertas, apesar de já ter-se escolhido versões para complementá-las. Para tanto, tem-se como parte do modelo de dados os objetos-configuração, sobre os quais foram estabelecidas operações de criação, alteração, cópia, remoção, consulta e escolha de objetos. O funcionamento destas operações é a base do processo de configuração.

PALAVRAS-CHAVE: Bancos de Dados para Aplicações de Projeto, CAD para Sistemas Digitais, Gerência de Configurações, Versões, Ambientes de Projeto.

TITLE: "Configuration manager to STAR framework"

ABSTRACT

This work presents the mechanisms for configuration management in the STAR framework. STAR is an electronic design automation framework, under development at the University of Rio Grande do Sul in cooperation with the IBM Rio Scientific Center at Rio de Janeiro, Brazil. The design objects supported are complex systems and they are represented through a hierarchical data model. Versions are used to express the evolution process of design objects. The version management mechanism developed is strongly related with the data representation, and it supports the different dimensions of versions: views, alternatives and revisions.

The use of versions associated with composite objects allows the existence of many possibilities of description for the same complex system, as a consequence of different versions combination. When submitting a design object to a design tool, like a simulator, it is necessary to select versions for components in order to obtain a single object description, called the object configuration. A configuration manager offers resources, as manipulation and query on design objects in the framework, to make version selection in configuration construction easy and fast.

STAR configurations are established through component reference attributes or through a configuration object definition. The configuration management mechanism developed for the STAR framework in this work supports static, dynamic and open configurations. They are established in a manual, automatic or semi-automatic way. In the manual way, the user is responsible for choosing the selected versions. In the automatic way, it is possible to choose between the current version and the most recent version, which are pre-defined criteria. The user can define and use configuration expressions in a semi-automatic way. These expressions make the version selection objective, due to use of objects attributes, representing

objects characteristics in the expression. The use of configuration expressions allows the selection of versions with specific characteristics. A language is available for the definition of configuration expressions.

The possibility to store configurations is provided. Configurations may be reused in another time and by other objects. The possibility to store configurations allows them remain either dynamic or open, even if the choose of versions to complement them had been done. To make this possible, configuration objects are integrated in the data model. Operations on configuration objects are: create, update, copy, delete, query and select . These operations are the basis of the configuration process.

KEYWORDS: Database to Design Applications, Digital Systems CAD, Configuration Manager, Versions, Design Frameworks.

1 INTRODUÇÃO

Ambientes de projeto auxiliado por computador (PAC) procuram facilitar o trabalho dos usuários no projeto de elementos complexos, os quais possuem inúmeros componentes que lhe determinam uma estrutura sofisticada, construída a partir de elementos mais simples. Estruturas mecânicas, como motores, utilizadas nas indústrias automobilística, naval e aeronáutica entre outras, e microprocessadores, memórias e unidades de controle, que fazem parte dos circuitos eletrônicos, são exemplos de elementos complexos. Eles são denominados objetos.

Os modelos de dados orientados a objetos podem ser definidos de forma hierárquica. Deste modo, facilitam a representação dos objetos nos ambientes de projeto, pois a descrição de cada um deles é, de um modo geral, hierárquica e complexa, com vários níveis de definição. A estruturação dos objetos complexos através dos modelos de estrutura hierárquica ocorre por composição de objetos, e tem por base o conceito de agregação [SMI 77]. Por exemplo, no caso dos sistemas digitais, um microprocessador é um objeto complexo, composto pela UCP e barramentos, que também são objetos complexos. A UCP, por sua vez, é composta pela ULA, UC e memória, que também são objetos complexos, e assim por diante. Em função da composição, os objetos complexos são também denominados compostos.

Segundo Ahmed em [AHM 91], um objeto composto é uma agregação recursiva de seus objetos componentes. Kim em [KIM 89] o define como um aninhamento de objetos de profundidade variável. De qualquer forma, cada objeto componente pode também ser um objeto composto ou um objeto primitivo, que não pode mais ser decomposto em outros objetos. A hierarquia de composição de um objeto de projeto é estabelecida em função da recursividade de sua composição determinada pelos componentes compostos.

Num projeto, a descrição de um objeto sofre modificações consecutivas que determinam o aprimoramento de sua forma inicial até chegar a um resultado

considerado satisfatório. Essa evolução é caracterizada, de um modo geral, pela utilização de versões [BAT 85] [KAT 86], que permite o acesso a estados anteriores de descrição dos objetos, uma vez que mantém sua evolução de projeto através de um histórico. Assim, um objeto de projeto é uma entidade independente do tempo, enquanto que uma versão representa o objeto em um instante de tempo, mantendo sua descrição correspondente.

Considerando-se que a cada objeto estão associadas suas versões, tem-se diversas opções de descrição para um mesmo objeto de projeto. A seleção de objetos que estabelece uma destas descrições determina a escolha de uma configuração para o objeto e permite que ele seja submetido a uma ferramenta para testes, como por exemplo um simulador.

Uma configuração para um objeto é obtida a partir do modelo de dados que permite representá-lo num ambiente de projeto. Por exemplo, Katz et al., em [KAT 86], apresentam um modelo de dados genérico, revisado posteriormente em [KAT 87], para um servidor de versões, que organiza os objetos complexos através de uma estrutura hierárquica, na qual há objetos estruturais ou genéricos e objetos representacionais que podem ser primitivos ou compostos. Para as versões, tem-se um objeto estrutural de versão que agrupa as instâncias de versões (objetos representacionais) que lhe correspondem, assim como uma interface generaliza as versões para as quais ela é comum. Versões de objetos compostos são compostas por versões de objetos componentes. Uma configuração associa uma versão de um objeto composto com versões de seus componentes.

Um outro exemplo é a definição de configurações no ambiente PLAYOUT [SIE 89] [SIE 89a], um ambiente para projeto de circuitos VLSI que tem por base um modelo hierárquico, no qual cada célula é representada por uma interface e um conteúdo. Uma interface representa uma abstração das várias alternativas de conteúdo, e um conteúdo se refere às instâncias das células. Descrições de configurações para um objeto são construídas através da utilização hierárquica de descrições de configurações para os sub-objetos associados a seus componentes. Isso

porque há dois níveis de configuração: inicial e final. A configuração inicial refere-se à seleção de um certo conteúdo para uma interface determinada. A configuração final corresponde à seleção de um objeto de configuração para cada interface. Um objeto de configuração é uma descrição de configuração (inicial ou final) associada a um conteúdo. A configuração inicial atinge um nível da hierarquia de composição e a final vários.

Estabelecer configurações para objetos de projeto pode tornar-se complicado, à medida que a complexidade de cada um deles aumenta. Pode haver, por exemplo, muitos componentes em cada nível da hierarquia de composição, ou mesmo a própria hierarquia de composição ter muitos níveis. Se o processo de escolha de versões for manual, podem ocorrer erros com maior facilidade, além de possíveis confusões e demora em demasia nos projetos muito grandes ou com muitos projetistas envolvidos.

Para agilizar este processo, estabelece-se mecanismos de gerência de configurações. Através deles, é possível automatizar esta tarefa, aumentando o rendimento do projeto. Há várias propostas de mecanismos para gerência de configurações, cada uma fortemente relacionada ao ambiente onde se insere, como as apresentadas para os ambientes SDE, ADÈLE, Version Server e DDM.

Kim e Chung em [SKM 91] apresentam o ambiente SDE para projeto de sistemas digitais que utiliza VHDL [LIP 90]. Seu modelo de dados apresenta *class, cell* e *design* como níveis de abstração que definem um objeto genérico, e *conf*, que determina a associação de tecnologias específicas, como cmos e nmos, aos objetos genéricos. A configuração de um objeto no SDE é, portanto, estabelecida através de um objeto *conf*.

O mecanismo de gerência de configurações de SDE é na verdade um mecanismo de ligação de objetos *conf* aos objetos genéricos. Ele é dirigido por restrições referentes ao comportamento estático de um objeto de projeto, que especificam propriedades elétricas e físicas como conectividade, tamanho, delay, latência e consumo

de energia. As restrições podem ser estabelecidas através de atributos sobre componentes específicos, sobre o todo esquemático ou sobre uma sub-estrutura do todo esquemático. Elas são embutidas numa linguagem que é uma extensão do VHDL. Pode-se defini-las individualmente sobre um objeto, ou agrupadas, até mesmo em forma de funções. É oferecida como recurso para estabelecer as restrições uma interface de usuário, na qual as restrições são modeladas como ícones.

O uso das restrições faz com que seja selecionada uma lista de células candidatas, entre as quais é escolhida uma pelo usuário para ser ligada a um determinado componente. Também pode ser utilizado o modo automático de ligação das versões mais recentes aos componentes genéricos de uma célula, a cada vez que há uma modificação nas células envolvidas.

Belkhatir em [BEL 85] e [BEL 87] apresenta o sistem ADÈLE, orientado para sistemas de software e que tem a composição modular dos programas como base para a modelagem de seus objetos. Cada módulo é a associação de uma interface a uma realização. Uma interface especifica o conjunto de recursos fornecidos pelo módulo (por exemplo, chamada de um procedimento). Uma realização implementa os recursos da interface, seja fornecendo-os por conta (similar a versões primitivas), seja utilizando recursos externos (por exemplo, outra chamada de procedimento), que são descritos por outras interfaces e fornecidos por outras realizações. Assim, uma realização consiste de um conjunto de corpos de módulos que é denominado configuração. Uma configuração implementa uma interface.

O estabelecimento de configurações pode ser feito de três formas diferentes: pela nomeação explícita dos componentes numa lista de composição, por um mecanismo default que cria uma configuração onde os componentes correspondem às suas respectivas revisões mais recentes, ou através de um mecanismo de seleção por atributos. Atributos são propriedades que caracterizam as diferenças entre versões e podem ser associadas a cada interface e a cada realização. São estabelecidas expressões lógicas envolvendo condições sobre os valores dos atributos, o que define as restrições para seleção de objetos.

No Version Server de Katz e Chang [KAT 87] [CHA 89], há um mecanismo para estabelecer a ligação de configuração dinâmica com base no conceito de níveis. Existem operações para particionar a base de dados em níveis que correlacionam versões de objetos entre objetos. O nível inicial contém os objetos ditos originais (a primeira instância de cada versão), o segundo nível contém novos objetos adicionados e novas versões de objetos já existentes, e assim por diante. Sob ligação dinâmica, um objeto composto identifica seus componentes pela referência a seus objetos genéricos associados (dos componentes). Isso porque as relações são estabelecidas entre objetos genéricos. As versões são realmente ligadas através da busca entre os níveis pela primeira versão encontrada do objeto de projeto. A ordem de busca nos níveis é determinada pelos contextos definidos pelo usuário.

O sistema DDM [BAN 91], aplicado à CAE, é modelado como um sistema de arquivos genéricos, no qual cada objeto tem um nome e uma localização definidos, e há relacionamentos estabelecidos entre os objetos. Um objeto pode ser um arquivo ou um diretório, e um conjunto de objetos define um projeto. Os objetos são versionáveis, e para capturar ou preservar um estado particular de um projeto como um todo, devem ser identificadas versões específicas de todos os componentes que juntas descrevem um estado do projeto. O conjunto destas versões define uma configuração.

O mecanismo de configurações, que determina a ligação entre os objetos, tem por base as facilidades oferecidas para manipulação de objetos dos tipos "checkpoints" ou "workspaces". Estes objetos contém referências a conjuntos de objetos DDM. As referências dos objetos "checkpoint" são fixas, definindo configurações estáticas, e as do tipo "workspace" podem ser mudadas, definindo configurações dinâmicas. Todas as ferramentas integradas no DDM operam no contexto de um "workspace". Há facilidades para acessar "workspaces" anteriores e para consultar objetos através de buscas em "workspaces".

STAR é uma plataforma para o desenvolvimento de ambientes para projeto de circuitos e sistemas digitais que esta sendo desenvolvido na Universidade

Federal do Rio Grande do Sul. Este trabalho tem por objetivo apresentar a definição de configurações e os mecanismos para gerenciar o seu estabelecimento no ambiente STAR.

Quanto à apresentação do trabalho, no capítulo 2 faz-se uma abordagem genérica de configurações, procurando introduzir características importantes relacionadas a elas e à sua gerência. No capítulo 3 introduz-se o ambiente STAR, através da apresentação de suas características básicas, seu modelo de dados e as características de seu gerente de versões. No capítulo 4, é apresentada a definição das configurações no ambiente STAR através de suas características fundamentais e dos elementos envolvidos. No capítulo 5, é apresentado o gerente de configurações definido: seus mecanismos básicos, os recursos oferecidos e as operações envolvidas, além da descrição do processo de construção de configurações e da dinâmica das próprias operações. No capítulo 6 são apresentadas as conclusões do trabalho desenvolvido.

2 GERÊNCIA DE CONFIGURAÇÕES

Neste capítulo procurou-se caracterizar configurações, numa abordagem genérica, mas direcionada para aplicação em ambientes de PAC. São apresentados seus tipos, modos de seleção de objetos e algumas considerações sobre notificação e propagação de mudanças, armazenamento de configurações e mecanismos de gerência.

2.1 Tipos de Configuração

A definição de configuração estática, dinâmica ou aberta é feita a partir do conceito de ligação. Uma ligação define a vinculação de uma instância de objeto a um determinado componente de um objeto composto, e pode ser estabelecida por uma referência. Ela pode ser estática, se vincular uma instância de um objeto completamente definido, até o nível de objetos primitivos, ou dinâmica, no caso da instância ser de um objeto não definido até o nível de objetos primitivos.

O conjunto de ligações estabelecidas para os componentes de um objeto durante a sua criação permite caracterizar o tipo de configuração estabelecida para ele. Segundo Katz em [KAT 90], a ligação entre objetos compostos e componentes determina configurações estáticas e dinâmicas.

O conceito de configuração aberta origina-se a partir da inexistência de uma ligação entre um componente e a instância de algum objeto. Isso permite que no momento da vinculação de uma instância, qualquer objeto possa ser escolhido. As configurações abertas são utilizadas em VHDL [LIP 90].

O estabelecimento de configurações dinâmicas e abertas, ambas incompletas porque o objeto configurado não fica definido até as versões primitivas, ocasiona apenas a postergação da definição de uma configuração completa para ele. Isso

pode ser desejável em alguns momentos do projeto, quando se quer experimentar configurações alternativas.

2.1.1 Configuração Estática

Uma configuração estática caracteriza-se por ter ligações estabelecidas até o nível de objetos primitivos. Ou seja, para cada componente do objeto configurado que for composto, há referências a outros objetos, ao longo de sua hierarquia de composição, e os últimos objetos referidos são primitivos.

Um objeto de projeto configurado estaticamente está completamente definido, e pode ser submetido a uma ferramenta.

Uma configuração estática é fixa e sua composição não pode ser alterada, a não ser por modificações nos objetos envolvidos, no caso de haver propagação de mudanças, ou na própria configuração se ela também for um objeto.

2.1.2 Configuração Dinâmica

Uma configuração dinâmica caracteriza-se por ter ligações não completamente resolvidas, ou seja, não estabelecidas até os objetos primitivos.

Muitos objetos compostos sem referências estabelecidas podem fazer parte da descrição de um objeto de projeto. Para objetos definidos através de modelos de estrutura hierárquica, compostos por outros objetos, cada objeto componente pode também ser composto e sua definição até os objetos primitivos postergada para um outro momento. Ou pode acontecer que o objeto referido esteja definido até um dos níveis da hierarquia do modelo que seja apenas um nível de abstração, e sua definição até os objetos primitivos também postergada para um outro momento. Essa pos-

tergação da definição completa da configuração faz com que se tenha configurações parciais para o objeto de projeto, se considerarmos sua hierarquia de composição.

A definição parcial de uma configuração pode dar origem a várias configurações diferentes, em função das escolhas de objetos e versões para complementá-la até os objetos primitivos. Essa condição é desejável, pois permite flexibilidade para a definição de configurações [GRA 91]. Pode-se, por exemplo, estabelecer uma referência de um componente para um objeto composto cujos componentes não tenham referências estabelecidas, e assim analisar diversas possibilidades de composição para ele.

Pode-se utilizar ligações dinâmicas para garantir que um objeto seja sempre composto pelas versões mais recentes dos objetos ligados a seus componentes. Assim, quando for necessário estabelecer uma configuração completa para o objeto, ela já estará atualizada pelas versões mais recentes dos objetos componentes.

Muitos sistemas permitem a definição de configurações dinâmicas, como o Version Server [CHA 89], o SDE [SKM 91], o PLAYOUT [SIE 89] e o AMPLO [WAG 88], entre outros.

2.1.3 Configuração Aberta

Uma configuração aberta caracteriza-se por não ter ligações estabelecidas entre os componentes do objeto configurado e outros objetos.

Assim, para um objeto composto é possível definir quantos serão os seus componentes e até mesmo a especificação de suas interfaces, mas apenas dentro da própria definição do objeto composto. Não há ligação entre um componente e um outro objeto, e no momento em que ela for definida, será preciso fazer uma consistência entre cada interface especificada livremente e as interfaces dos objetos ligados, pois é preciso que elas sejam compatíveis.

A possibilidade de utilizar configurações abertas é um recurso importante, principalmente nos projetos "top down". Isto porque muitas vezes sabe-se que um objeto será composto por outros objetos, mas ou eles ainda não foram definidos no ambiente, ou ainda não se decidiu quem serão eles entre os objetos já definidos.

2.2 Modos de Seleção de Objetos

Para criar uma configuração para um objeto de projeto, é preciso escolher uma descrição para ele dentre as várias que podem coexistir devido às inúmeras versões que podem compôr a sua descrição. A tarefa de selecionar componentes no processo de estabelecimento de uma configuração de um objeto de projeto exige uma quantidade de esforço diretamente relacionada à complexidade do objeto.

Os modos que se pode utilizar para seleção de versões de objetos são aperfeiçoados em função da complexidade da estrutura de definição do objeto, que exige recursos mais sofisticados, e em função da própria aplicação, para a qual um modo pode ser mais adequado do que outro.

De uma maneira geral, um modo de seleção de versões pode ser manual, automático ou semi-automático [GRA 91]. O modo manual é o básico e mais simples, com poucas facilidades adicionais. Já o automático e semi-automático dão agilidade ao processo, pois baseiam-se em um ou mais critérios pré-definidos no sistema ou pelo usuário, que permitem automatizar a seleção, ou ao menos uma parte dela.

2.2.1 Modo Manual

A seleção manual é feita através de escolhas do usuário, o qual define, uma a uma, as versões de objeto que farão parte da configuração do objeto de projeto ao longo de sua hierarquia de composição. Os critérios utilizados são os

mais variados possíveis, de responsabilidade de cada usuário. Muitas vezes, ele pode buscar subsídios em informações sobre os objetos obtidas a partir do próprio sistema, que pode ser consultado livremente.

Este é um modo demorado de seleção, principalmente se a composição do objetos for estabelecida através de muitos níveis. Além disso, é trabalhoso e sujeito a muitos erros em função do volume de dados e da própria imperfeição do ser humano.

Em muitos sistemas, como AMPLO [WAG 88] [WAP 91], a seleção manual é facilitada pela presença de um "browser", que permite a navegação sobre a hierarquia de composição, permitindo que o usuário visualize cada objeto envolvido com suas opções de descrição, e selecione as que julgar convenientes.

Quando não se tem um "browser" e todo o processo de escolha precisa ser feito textualmente, ele fica ainda mais demorado e cansativo, pois além de ter que percorrer nível a nível a hierarquia de composição, o usuário tem que imaginar a estrutura que está compondo, a partir dos objetos que vai escolhendo.

2.2.2 Modo Automático

Essa estratégia de seleção pressupõe a existência de critérios pré-definidos, nos quais ela será baseada. Se eles forem pré-definidos no sistema, o usuário terá que optar por um deles. Se tiverem que ser pré-definidos pelo usuário, haverá uma maior flexibilidade, porém os mecanismos de suporte terão que ser mais sofisticados. Um critério pode ser estabelecido a partir de uma combinação de outros critérios.

Em sistemas que utilizam versões, um critério bastante comum e largamente utilizado é a seleção com base na versão mais recente. Como cada versão de um objeto representa uma evolução em relação a uma versão anterior, é interessante que uma configuração descreva a versão mais recente do objeto de projeto, ao menos

em algumas partes. Assim, para cada componente para o qual for necessário selecionar um objeto como referência, será selecionada automaticamente a sua versão mais recente. Esse critério é utilizado, por exemplo, nos sistemas AMPLO [WAG 88], Apollo DSEE, citado por Katz em [KAT 90], DDM [BAN 91], ADELE [BEL 87] e ORION [KIM 90] onde a versão mais recente é chamada de versão *default.* No sistema A_HAND [VIC 89], o modo automático de configuração é implementado através da *definição de configuração default.* Essa definição estabelece que na configuração de um objeto em desenvolvimento será utilizada, para cada objeto envolvido na sua composição, a versão corrente do projetista (versão que tem associado um rótulo CORRENTE) ou, caso não exista uma para aquele objeto, a versão mais recente liberada pelos demais projetistas.

O modo automático de seleção de objetos é bastante comum e largamente utilizado nos processos de configuração.

2.2.3 Modo Semi-Automático

Este modo é definido em função da combinação dos modos anteriores, nos casos em que apenas um deles não é suficiente ou não é adequado para selecionar uma versão para cada posição na descrição do objeto de projeto.

Muitas vezes, a utilização de um critério automático, baseado, por exemplo, em propriedades das versões, faz com que seja selecionada mais de uma versão de objeto para uma mesma posição na descrição do objeto de projeto, o que não é satisfatório para uma configuração, que precisa de uma versão para cada posição. Pode-se então combiná-lo com um modo manual ou com um modo automático baseado em outro critério, que realize a seleção de uma única versão. Em alguns casos esta situação é proposital, como no SDE [SKM 91], no qual o usuário estabelece restrições para que o sistema automaticamente restrinja o universo de seleção a uma

lista de componentes, sobre a qual o usuário deverá selecionar manualmente um objeto, que será utilizado como componente.

Uma outra situação em que esse modo de seleção é útil é quando o usuário muda de idéia com relação a um modo ou critério, ou quando ocorre um erro no meio do processo de configuração, como em AMPLO. Este erro pode ser causado, por exemplo, pela ausência de uma versão de objeto que apresente um determinado valor de atributo. Após a mudança, a seleção continua normalmente a partir do ponto onde tinha parado, utilizando o novo modo ou critério para tentar complementar a configuração.

2.3 Notificação e Propagação de Mudanças

Ao estabelecer uma configuração para um objeto de projeto, são definidas para fazer parte dela versões de objetos envolvidos na sua composição. Muitas vezes são as mais recentes, compondo a versão mais atualizada do objeto de projeto. Cada objeto envolvido nessa composição pode sofrer novas atualizações, que determinam sua evolução ao longo do tempo e desencadeiam a geração de novas versões, e as versões envolvidas na configuração deixam de ser as mais recentes.

Como muitas vezes é importante que as configurações sejam atualizadas constantemente pela incorporação das versões mais recentes dos objetos nela envolvidos, tem-se definidos mecanismos de notificação e propagação de mudanças.

Katz em [KAT 90] define propagação de mudanças como o processo que automaticamente incorpora novas versões às configurações. A utilização de um mecanismo automático, que substitua todas as versões de um objeto em todas as configurações das quais ele fizer parte por sua versão mais recente pode não ser desejável sempre. Pode haver situações em que é importante ter mais de uma versão de um mesmo objeto numa configuração. Como por exemplo no caso em que a versão

mais recente de uma célula de memória é utilizada na cache de instruções de um microprocessador, enquanto que para a memória de dados uma versão mais antiga é suficiente.

Uma outra questão envolvida é a limitação do escopo da mudança. De acordo com o mecanismo de versionamento utilizado, a substituição de uma versão num determinado nível da hierarquia de composição pode determinar o versionamento do nível superior e se propagar pelo resto da hierarquia. Alguns mecanismos utilizados para controlar o processo têm por base a definição de restrições, muitas vezes baseadas em marcos de tempo ("timestamps"), cuja violação encerra a propagação [KAT 90].

Muitos sistemas suportam apenas a notificação de mudanças, situação em que o usuário é avisado que houve geração de uma versão mais recente para um objeto. Pode haver um mecanismo de propagação de mudanças limitado associado que, por exemplo, questiona o usuário sobre cada possível troca de versão ocasionada pela geração de uma versão mais recente.

Batory e Kim em [BAT 85] apresentam um mecanismo de notificação de mudanças que tem por base os selos de tempo. Seu uso permite limitar o número de mensagens de notificação a serem enviadas aos usuários. Segundo Katz em [KAT 90], através desse mecanismo é possível estabelecer distinção entre o momento em que o usuário é notificado de uma mudança e o momento em que ele concorda com ela. Já o Version Server [CHA 89] suporta propagação de mudanças através de um processo de construir uma nova configuração na qual é incorporada a nova versão como parte de sua descrição.

2.4 Armazenamento de Configurações

A utilização de uma configuração definida para um objeto de projeto não precisa ser restrita a um só momento, como uma sessão de projeto, após a qual sua descrição é perdida. Essa é uma restrição muito forte, uma vez que ela não pode ser reaproveitada, o que poderia ser útil e desejável. Além disso, pode não ser interessante que exista apenas uma configuração para um objeto e sim várias, para que possam ser comparadas através dos resultados obtidos na submissão às ferramentas de teste e análise, ou ao menos em relação à sua própria composição. Por isso é importante que elas possam ser armazenadas.

As configurações podem fazer parte do modelo de dados, desta forma sendo objetos definidos e também armazenados, como em VHDL [LIP 90], Version Server [CHA 89] e SDE [SKM 91]. Elas também podem ser armazenadas quando definidas através de estruturas auxiliares, ou mesmo pela combinação de outros objetos, como no sistema SEE [MAH 90].

De acordo com o relacionamento entre configuração e o objeto para o qual ela é definida, o armazenamento dessa descrição pode significar uma configuração completa ou apenas parcial. Por exemplo, na definição de um objeto composto por outros objetos, seus componentes, uma configuração armazenada relacionada a ele poderia conter apenas a descrição das referências estabelecidas para seus componentes, não seguindo a seqüência da hierarquia de composição mesmo que entre os objetos referidos houvesse objetos compostos. Essa configuração seria parcial. Já uma configuração armazenada relacionada a um objeto de projeto, descrevendo todas as referências estabelecidas ao longo da hierarquia de composição, determinaria uma configuração completa.

A reutilização de descrições de configuração por outros objetos também é uma vantagem do seu armazenamento. É um recurso que poupa tempo, pois evita

a realização repetida da definição de configurações, que por sua vez pode ser um procedimento bem desgastante e trabalhoso.

2.5 Mecanismos para Gerência de Configurações

O suporte a configurações dinâmicas e abertas exige mecanismos que permitam estabelecer e modificar configurações a qualquer momento, não somente durante a criação do objeto. A complexidade que estes tipos de configuração podem oferecer deve ser suportada pelos recursos oferecidos pelo gerente de configurações, que deve facilitar o processo de seleção de versões.

Esses recursos para a seleção de componentes possuem uma complexidade crescente de acordo com a sofisticação dos mecanismos envolvidos. O mínimo que se pode exigir como recurso para a seleção de versões é a pré-definição de um ou mais critérios, neste último caso para que o usuário selecione um deles, que permita a seleção automática de versões de objetos. Critérios comumente adotados nestes casos são seleção com base na versão mais recente, ou na versão corrente, muitas vezes a versão *default*.

No sistema A_HAND [VIC 89], a versão *default* é definida para cada objeto e pode ser, para um determinado projetista, a versão corrente ou a mais recente, dependendo da associação ou não do rótulo "CORRENTE". É a versão corrente nos casos em que os objetos referidos têm uma uma versão asociada a um rótulo "CORRENTE", normalmente a última versão liberada pelo proprietário do objeto. É a versão mais recente para os outros objetos referidos, que não têm o rótulo "CORRENTE" associado e normalmente são utilizadas pelo projetista proprietário, que deseja que a configuração utilizada para testes no sistema se constitua das versões mais recentes que ele estiver desenvolvendo.

Critérios mais sofisticados estão ligados à seleção baseada nos atributos e em características dos objetos, como por exemplo velocidade, área e número de componentes, no caso dos sistemas digitais. A seleção de versões de objetos é feita a partir da análise de restrições impostas sobre elas, através da definição de expressões de seleção. Isso torna a seleção mais objetiva e específica, pois é muito mais natural e intuitivo para o usuário lembrar de certas características desejáveis nos objetos que quer selecionar do que lembrar de cada objeto. Por exemplo, Ploedereder em [PLO 89] apresenta um gerenciador que possibilita, a cada seleção de componente, estabelecer restrições por atributos e ver o conjunto das próximas instâncias válidas, sobre o qual haverá outra seleção. SDE, MVC e ADELE também têm por base um mecanismo de restrições a partir de atributos dos objetos.

Há outras propostas para facilitar e tornar esta tarefa mais dirigida, como a utilização de "checkpoints" e "workspaces" por Banks em [BAN 91], ou de "layers" e "contexts" por Katz em [KAT 87]. Ambos baseiam-se na distribuição das versões em níveis ou espaços, de acordo com algum critério de interesse, otimizando a busca das versões para a criação de configurações.

3 O AMBIENTE STAR

O STAR [WAG 91c] [WAG 92] é uma plataforma para o desenvolvimento de ambientes para projetos de circuitos e sistemas eletrônicos que está sendo desenvolvida na Universidade Federal do Rio Grande do Sul. Ambientes para projeto dessa natureza exigem a atuação de diversas ferramentas, que podem ser de fornecedores diferentes e dedicadas a diferentes etapas de projeto [WAG 90]. Além de permitir a integração de ferramentas, é preciso um modelo de dados flexível, que possa se adaptar às diferentes arquiteturas a serem estabelecidas, com possibilidade de definir níveis de abstração adequados a cada elemento a ser projetado.

Para permitir a representação dos seus objetos de projeto, foi definido para o STAR, a partir do modelo de dados do GARDEN [WAG 91b], um modelo de dados hierárquico, baseado em objetos complexos, e que é a base para definições de esquemas de objetos para aplicações específicas. A evolução dos objetos de projeto é controlada pelo uso de versões, através de um mecanismo de gerenciamento fortemente relacionado à representação dos dados, suportando assim as diferentes dimensões que essa representação permite: visões em diferentes níveis de abstração, alternativas de representação para um mesmo nível da abstração e revisões para cada alternativa e visão. Além disso, há também um modelo de gerência de metodologias [WAG 91c], que possibilita definir uma organização específica para as várias representações de cada objeto de projeto, de forma a refletir as descrições criadas por tarefas particulares.

Este capítulo apresenta o modelo de dados e o gerente de versões do STAR.

3.1 O Modelo da Dados do STAR

Os objetos do ambiente STAR são representados através de um modelo de dados de estrutura hierárquica, apresentado na figura 3.1, que tem como diferentes níveis de abstração as classes *Design, ViewGroup* e *View.* Estes nodos determinam o **esquema do objeto**, no qual cada um deles representa um nível de abstração em que mais propriedades são acrescidas ao objeto de projeto. Cada esquema de objeto é um esquema conceitual e representa uma hierarquia de generalização [SMI 77], pois cada nodo possui propriedades que são herdadas por seus descendentes.

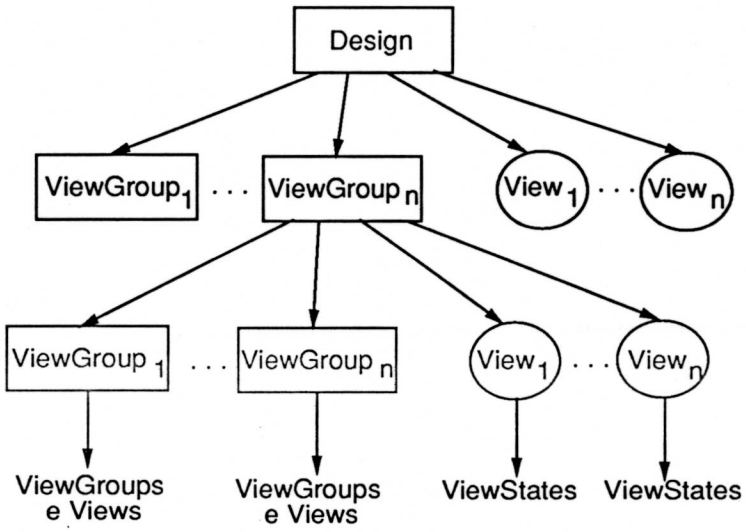

Figura 3.1: Modelo de dados do STAR

Também fazem parte da estrutura hierárquica que define o modelo de dados os *ViewStates*, ligados às *Views*, que mantêm os dados concretos dos objetos de projeto.

A partir do modelo de dados pode-se estruturar cada esquema de objeto de um modo diferente, construindo hierarquias diversas de acordo com critérios definidos pela metodologia ou pelo usuário. Isso porque cada *Design* pode ter um número qualquer de *ViewGroups* e *Views*, o mesmo ocorrendo para cada *ViewGroup*, o que oferece bastante flexibilidade na estruturação dos objetos.

Cada nodo do esquema do objeto possui propriedades, representadas por seus atributos, que são definidas como herdáveis ou não pelos seus descendentes. Além do atributo propriamente dito, o nodo descendente pode herdar também o seu valor, no caso dele estar definido. Além disso, a herança pode ser *default* ou *estrita*. Na herança default, propriedades herdadas são válidas somente se não há uma nova definição destes atributos em seus descendentes. Na herança estrita, todas as propriedades herdadas são válidas para os seus descendentes, não sendo possível redefinir um atributo num nível inferior ao qual ele foi definido. Os atributos podem ser pré-definidos no modelo ou definidos pelo usuário através dos *Userfields*.

A seguir são apresentados os elementos que fazem parte do modelo de dados do STAR, e que estão definidos em [WAG 91b].

3.1.1 Repository, Libraries e Processes

Repository é a coleção de todos os objetos na base de dados. Ele é composto de *Libraries* e *Processes*. As *Libraries* armazenam os objetos de projeto e cada um deles está apenas em uma *Library*. Cada *Library* é uma coleção de objetos de projeto.

Os *Processes* contêm informações de tecnologia e seu conteúdo é manipulado diretamente pelas ferramentas. Se um *Process* não for definido junto ao *Repository* ou à *Library*, isso pode ser feito num *Design*, *ViewGroup* ou *View*. A partir do nodo em que é definido, ele é herdado pelos seus descendentes.

3.1.2 Design

Um *Design* é um objeto de projeto único, como um microprocessador, uma ULA, um registrador ou uma porta lógica. Ele representa o nodo raiz da hierarquia que define um objeto no STAR.

Cada *Design* é uma coleção de *ViewGroups* e *Views*, que herdam alguns de seus atributos, como os *Ports*. *Designs*, assim como *ViewGroups* e *Views* podem ser objetos parametrizados, o que significa que valores reais são atribuídos somente às instâncias, quando elas são criadas. Todos os parâmetros de um *Design* são herdados pelos *ViewGroups* e *Views* descendentes.

3.1.3 ViewGroup

Um *ViewGroup* é uma coleção de representações de um objeto de projeto, as quais têm algumas propriedades em comum que podem ser definidas pelo usuário ou pela metodologia. Essas propriedades são armazenadas em um atributo para fins de documentação.

Cada *ViewGroup* pode ser decomposto em outros *ViewGroups* e *Views*, os quais herdam alguns de seus atributos.

Os *ViewGroups* representam alternativas de projeto, agrupando as representações que correspondem a determinada decisão de projeto. Além disso, a possibilidade de decompor um *ViewGroup* em outros *ViewGroups* peermite inserir cada alternativa de projeto no nível de abstração adequado.

3.1.4 View

Uma *View* é uma representação de um objeto de projeto num certo nível de abstração. Para cada *View*, há um número qualquer de *ViewStates* vinculados.

De acordo com o tipo de descrição a ser utilizada, as *Views* podem ser *HDLView, LayoutView* ou *MHDView*, e a cada uma delas é associado um grafo de *ViewStates*, respectivamente *HDLViewStates, LayoutViewStates* ou *MHDViewStates*. Uma *HDLView* é dedicada a descrições comportamentais em níveis altos de abstração, normalmente feitas através de uma linguagem de descrição de hardware, como VHDL [LIP 90]. Uma *LayoutView* é orientada a descrições geométricas de realizações físicas, como o *layout* de circuitos integrados ou placas de circuito impresso. Já uma *MHDView* (Modular Hierarchical Description View) é utilizada para descrições puramente estruturais.

3.1.5 ViewState

Os *ViewStates* são os nodos da estrutura hierárquica onde são armazenados os dados concretos de cada objeto de projeto. Os *ViewStates* de uma *View* estão organizados na forma de um grafo acíclico, o que será melhor explicado na seção dedicada ao Gerente de Versões, e não há herança de atributos entre eles.

Os *ViewStates* podem ser primitivos ou compostos. *HDLViewStates* e *LayoutViewStates* definem os *ViewStates* primitivos, que contêm descrições comportamentais e geométricas. Os *MHDViewStates* definem os *ViewStates* compostos e são descrições puramente estruturais. Estas descrições definem a composição de objetos, pois é nos *MHDViewStates* que são definidos os sub-objetos ligados a um *Design*. *HDLViewStates* e *LayoutViewStates* até podem possuir sub-objetos, mas as interconexões exatas entre eles só são manipuladas nos *MHDViewStates*, através das *Nets*.

3.1.6 DesignInstances e Components

Os sub-objetos definidos nos *ViewStates* são os *DesignInstances*. Eles e os *Components* permitem estabelecer referências a outros objetos definidos no ambiente. *Components* são generalizações de *DesignInstances* e são declarados localmente nos *ViewStates*. Quando há muitos *DesignInstances* do mesmo tipo num *ViewState*, pode-se declará-los através de um *Component*.

Os *DesignInstances* podem ser utilizados com ou sem vínculo com algum *Component*. Se na especificação de sub-objetos for utilizado um *Component*, ao qual estejam vinculados certos *DesignInstances*, e ele fizer referência a um objeto qualquer, o objeto por ele referido também o será por todos os *DesignInstances* a ele vinculados. Os *DesignInstances*, mesmo estando ligados a um mesmo *Component*, podem fazer referência a objetos diferentes, desde que não tenha sido estabelecida referência no *Component*.

Através de *DesignInstances* e *Components* são estabelecidas ligações entre os objetos, determinando sua composição. Ambos possuem, na sua estrutura:

- um atributo de referência, que permite definir a referência a um outro objeto, estabelecendo a ligação;
- mapeamento de portas, que relaciona portas do objeto composto que faz a referência com o objeto componente referido;
- mapeamento de parâmetros, utilizado caso o *DesignInstance* ou o *Component* seja ligado a um objeto parametrizado.

DesignInstances e *Components* serão chamados, de um modo geral, de componentes.

3.1.7 ConfigurationBodies

Estes objetos permitem o armazenamento de configurações, o que auxilia na complementação de configurações dinâmicas e abertas. Eles serão abordados com mais detalhes no capítulo seguinte.

3.1.8 Ports

Ports são sinais físicos de interface por meio dos quais objetos são interconectados. Eles podem ser fios simples ou um conjunto de fios, quando são definidos por um vetor de bits. Além disso, pode-se defini-los em qualquer nível da estrutura hierárquica de um objeto. Sua existência e valor são herdados pelos descendentes, quando definidos nos nodos *Design, ViewGroup* e *View.* Entre os *ViewStates* não há herança de *Ports.*

3.1.9 Nets

Nets permitem modelar as interconexões entre *Ports* da interface de um objeto de projeto e *Ports* dos *DesignInstances* contidos neste objeto. Elas também podem conectar apenas *Ports* de *DesignInstances* ou apenas *Ports* de objetos de projeto, o que torna possível conexões diretas entre sinais da interface de um objeto de projeto ou entre sinais de seus componentes.

Uma *Net* pode conectar qualquer número de *Ports. Ports* da interface de um objeto ou *Ports* de *DesignInstances* do objeto podem ficar desconectadas.

3.1.10 Auxiliary Objects

Estes objetos são utilizados em aplicações específicas, como simulação, síntese ou geração de testes, durante as quais são criados. Eles não modelam circuitos e sim outros objetos gerados durante as aplicações, como estímulos e resultados de simulações, casos de teste, medidas de testabilidade e restrições de síntese.

3.1.11 Correlations

As *Correlations* possibilitam que se especifique relacionamentos entre objetos, como a equivalência, estabelecidos de acordo com critérios definidos pela metodologia ou pelo usuário.

Uma *Correlation* envolve dois objetos, um à direita e um à esquerda, e um relacionamento entre eles que tem associados uma direção e um modo. Quanto à direção, que representa a dependência entre os objetos envolvidos na correlação, um relacionamento pode ser direcionado (de um objeto à esquerda para outro à direita), não direcionado ou bidirecional.

O modo pode ser protegido ou removível. No modo protegido, caso o relacionamento seja direcionado o objeto à esquerda não pode ser removido, e no caso do relacionamento ser bidirecional tanto o objeto à esquerda quanto o objeto à direita não podem ser removidos. No modo removível, para um relacionamento direcionado a remoção do objeto à esquerda implica a remoção imediata do objeto à direita, e para um relacionamento bidirecional a remoção de qualquer um dos objetos implica a imediata remoção do objeto a ele relacionado. O modo não tem significado para relacionamentos não direcionados.

3.2 O Gerente de Versões

O modelo de dados definido para o ambiente STAR está associado a um modelo de gerência sobre novas representações de objetos criadas no decorrer do desenvolvimento de um projeto bastante abrangente. Esse modelo é bastante flexível e envolve as várias dimensões de evolução de objetos: **visões, alternativas e revisões.**

Através da estruturação de cada esquema do objeto tem-se diferentes visões do objeto.

No esquema do objeto, os níveis *ViewGroup* e *View* permitem que se defina várias alternativas de uma mesma representação, em virtude das decisões de projeto tomadas, determinando diferentes soluções de projeto para um mesmo objeto num mesmo nível. Para cada *View*, há a vinculação de um grafo acíclico de *ViewStates*, que expressa a evolução do objeto de projeto no tempo. Os *ViewStates* representam as revisões criadas em função dessa evolução do objeto, e contêm seus dados efetivos.

Para designar as várias representações criadas para o mesmo módulo de hardware nas diferentes dimensões do processo de evolução do projeto, será utilizado, de um modo geral, o conceito de versão. Assim, versões corresponderão a visões nos vários níveis de abstração, a soluções alternativas de projeto e a revisões para cada alternativa ou visão [WAG 92].

3.2.1 Os Mecanismos de Versionamento

O ambiente STAR possui um mecanismo de gerenciamento de versões fortemente relacionado aos mecanismos de representação de dados e, dessa forma, apropriado aos seus objetos [LAC 92] [WAG 92]. Ele suporta as diferentes dimensões de versionamento decorrentes do seu modelo de dados, representando **visões** através

dos níveis de abstração (*Design, ViewGroup* e *View*), **alternativas** de representação para um mesmo nível de abstração e **revisões** para cada alternativa e visão.

Assim, tem-se o versionamento dos objetos suportado em dois níveis:

- conceitual: a nível de esquema de objeto, para *Design, ViewGroup* e *View*, determinando versões que correspondem a alternativas e visões;
- em nível inferior: para *ViewStates*, determinando novos *ViewStates*, e para os nodos do esquema do objeto, determinando versões para cada um deles.

No versionamento conceitual, novos nodos no esquema do objeto são explicitamente criados sob controle do usuário ou da metodologia de projeto. Já o versionamento inferior é automático. Nesse nível de versionamento, para os nodos do esquema do objeto são criadas novas versões quando atributos definidos são alterados, seja pela criação de novos atributos, modificação de seus valores ou mesmo pela remoção de atributos já existentes, determinando uma estrutura linear. Para os *ViewStates*, a estrutura criada em função do versionamento tem a forma de um grafo acíclico, no qual o novo *ViewState* gerado será considerado sucessor do *ViewState* corrente associado à mesma *View*, a não ser que o usuário defina explicitamente os predecessores.

Na figura 3.2, pode-se observar a representação do versionamento em nível inferior do objeto X. No esquema do objeto, as versões são representadas por círculos e retângulos sobrepostos. Já para os *ViewStates*, tem-se a representação do grafo acíclico que é a estrutura de derivação de versões neste nível. O versionamento conceitual pode ser caracterizado pela existência dos *ViewGroups* X_{VG1} e X_{VG2} ligados ao *Design* X_D, e pelas *Views* X_{V1} e X_{V2} ligadas ao *ViewGroup* X_{VG1}.

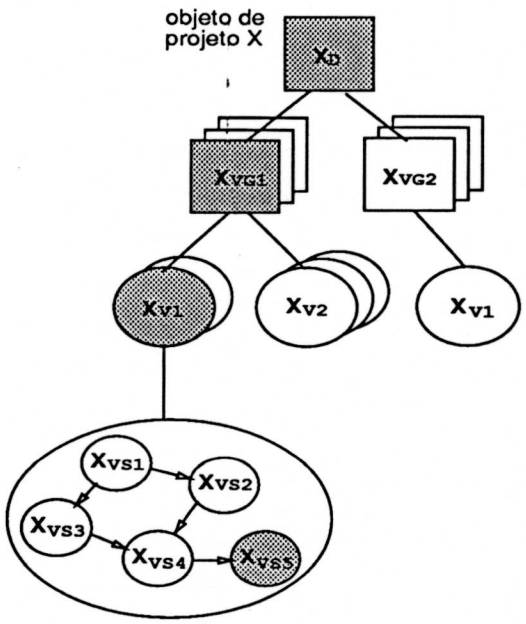

Figura 3.2: Versionamento do objeto de projeto X definido no STAR

3.2.2 Versões Correntes

O gerente de versões do STAR utiliza o conceito de versão corrente, que é a versão sobre a qual serão realizadas todas as operações que afetam o nodo, caso nada seja dito em contrário.

Para cada objeto, é apresentado ao usuário sempre o seu conjunto de versões correntes de *Design, ViewGroups* e *Views*, o que determina uma instância do esquema do objeto válida num determinado instante de projeto. No caso dos *ViewStates*, como eles são os nodos onde os dados de projeto estão realmente armazenados, é permitido o acesso a todos eles pelo usuário. Mas apenas um *ViewState* associado à mesma *View* é corrente.

A nível de esquema do objeto, quando uma nova versão de um nodo é criada, ela se torna automaticamente descendente da versão corrente do nodo de nível superior. A nova versão gerada passa a ser corrente, em substituição à versão corrente anterior. No caso dos *ViewStates*, um novo *ViewState* gerado é descendente

do *ViewState* corrente associado à mesma *View*, a não ser que o usuário estabeleça explicitamente os *ViewStates* predecessores.

Há duas operações estabelecidas para selecionar a versão corrente: seleção parcial, através da qual torna-se corrente apenas a versão selecionada (*Design, ViewGroup* ou *View*), e seleção total, através da qual tornam-se correntes a versão selecionada e seus respectivos ascendentes e descendentes válidos no momento da criação da versão. Para a seleção total pode-se selecionar inclusive *ViewStates*.

3.2.3 Estados de Consistência

No STAR, os objetos, de um modo geral, podem ser compartilhados entre projetistas. Para tanto há necessidade de se adotar mecanismos que garantam a sua consistência. Como critério para permitir e gerenciar esse compartilhamento adotou-se, de acordo com a disponibilidade de uma versão de objeto para os demais projetistas, a associação, a cada uma delas, de um estado de consistência. Esta disponibilidade está relacionada com o grau de desenvolvimento da versão do objeto, e pode ser para o próprio projetista, para seu grupo ou para todos os projetistas. Tem-se o seguinte relacionamento entre os estados de consistência e sua disponibilidade:

- em trabalho: versões disponíveis para o próprio projetista que por ele podem ser alteradas, sem ocasionar a geração de novas versões, e removidas.
- estável: versões que atingiram relativo grau de consistência, disponíveis para consulta para o grupo do projetista. Não podem ser alteradas, se não forem versionáveis. Caso sejam, alterações geram novas versões do objeto, para *Designs, ViewGroups* e *Views*. Para *ViewStates*, geram novos *ViewStates*. Podem ser removidas.

- consolidada: versões que atingiram alto grau de confiabilidade, disponíveis para consulta para todos os projetistas. Não podem ser alteradas nem removidas.

A promoção de um estado de consistência para outro pode ser feita automaticamente, para os *ViewStates* e por questões de integridade, ou de forma explícita pelo usuário, para *Designs, ViewGroups* e *Views* [LAC 92]. Por exemplo, a criação de um *ViewState* determina a promoção automática de seu predecessor (ou predecessores) a estável (ou estáveis).

3.2.4 As Operações sobre os Objetos do STAR

Estão definidas as seguintes operações sobre os objetos de projeto do STAR: criação de objetos, versionamento de objetos, seleção de versões, remoção de objetos, consultas a objetos, alterações de objetos, promoções e manipulação de correlações.

A criação de um objeto consiste na geração de um objeto de projeto inédito, ao qual é atribuída uma identificação única. A ele é atribuído o estado de consistência "em trabalho", e seu nodo pai na hierarquia automaticamente é promovido a estável, não podendo mais sofrer alterações.

O versionamento de objetos corresponde à geração de uma nova versão de um objeto já existente. Essa operação se aplica a nodos do tipo *Design, ViewGroup* e *View*. Ao versionarmos um nodo, a nova versão passará a ser automaticamente a versão corrente desse nodo.

A seleção de versões possibilita a versões antigas tornarem-se correntes novamente. Ela pode ser parcial, utilizada quando o usuário deseja retornar apenas um nodo a uma versão anterior, ou total, utilizada quando o usuário deseja modificar a versão corrente de um nodo, alterando também as versões correntes dos ascendentes

deste nodo. Na seleção total os nodos ancestrais assumirão as versões válidas no momento da geração da versão indicada pelo usuário.

A remoção de objetos possibilita a remoção de um objeto de projeto como um todo ou apenas de seus nodos. Um objeto só pode ser removido se não for utilizado como componente de um outro objeto.

A operação de consultas a objetos permite consulta global, através da qual pode-se visualizar a estrutura completa do objeto, e consulta a nodos específicos, através da qual pode-se visualizar nodos correntes e *ViewStates*. Pode-se ainda consultar atributos de nodos, atributos herdados e informações sobre a composição de um objeto composto e sobre correlações.

A alteração de objetos só pode ser realizada sobre nodos em trabalho. Os nodos estáveis ou consolidados são passíveis de versionamento.

A promoção diz respeito aos estados de consistência dos objetos e pode ser automática ou feita de forma explícita. *Designs, ViewGroups* e *Views* são automaticamente promovidos a estáveis quando for criado o seu primeiro descendente. No caso dos *ViewStates*, ao ser gerado um *ViewState* sucessor, seus *ViewStates* predecessores serão automaticamente promovidos a estáveis. A promoção explícita é de responsabilidade do usuário.

A manipulação de correlações é estabelecida através de três operações: criação, consulta e remoção de correlações.

4 CONFIGURAÇÕES NO STAR

Neste capítulo é apresentada, de um modo geral, a definição conceitual de configurações no ambiente STAR. Para tanto, são caracterizados os tipos de configurações que podem ser definidas e as maneiras de fazê-lo, por definição de atributo ou através de um objeto de configuração. Também são abordados o armazenamento de configurações, problemas de referência e a integração do conceito de configurações no STAR com o modelo de versões definido em [LAC 92], além de ser estabelecida uma classificação a partir das características das configurações.

4.1 Definição Conceitual

Um objeto de projeto do STAR é um objeto complexo, e assim pode ser composto por outros objetos. Essa hierarquia de composição é estabelecida através de referências entre objetos, de um objeto composto para seus objetos componentes.

A partir de cada esquema de objeto definido e de seus respectivos *ViewStates*, tem-se as hierarquias de definição dos objetos do STAR. Uma hierarquia de composição de um objeto de projeto é estabelecida pelas ligações entre as hierarquias de definição dos objetos que fazem parte da sua composição.

Para que um objeto de projeto possa ser testado, ele tem que ter uma definição completa. Num ambiente onde se considera o uso de versões, isso implica a seleção de uma versão a cada nível ao longo da hierarquia de composição do objeto em questão onde houver mais de uma opção de representação [GRA 91]. Assim, estará sendo definida uma configuração possível para o objeto, através da qual ele poderá ser submetido a uma ferramenta para testes, como um simulador.

A escolha das versões (ou opções de representação) que formam uma configuração para um objeto de projeto X é feita a partir da seleção de versões de objetos

$V_1,\ldots,V_n$ para os componentes $C_1,\ldots,C_n$, que fazem parte de um dos *ViewStates* de X. Desta forma, sendo X definido sobre uma estrutura cuja hierarquia de definição é dada pelos níveis *Design, ViewGroup, View* e *ViewState*, e na qual pode-se ter várias versões em cada nível, a especificação de uma configuração para o objeto X significa a seleção de uma versão para cada nível da estrutura de X, e respectivamente para cada nível da estrutura dos objetos referidos pelos componentes de X, e assim sucessivamente, até os níveis finais de representação. Ou seja, são seleções de versões sobre as hierarquias de definição dos objetos existentes ao longo da hierarquia de composição de X.

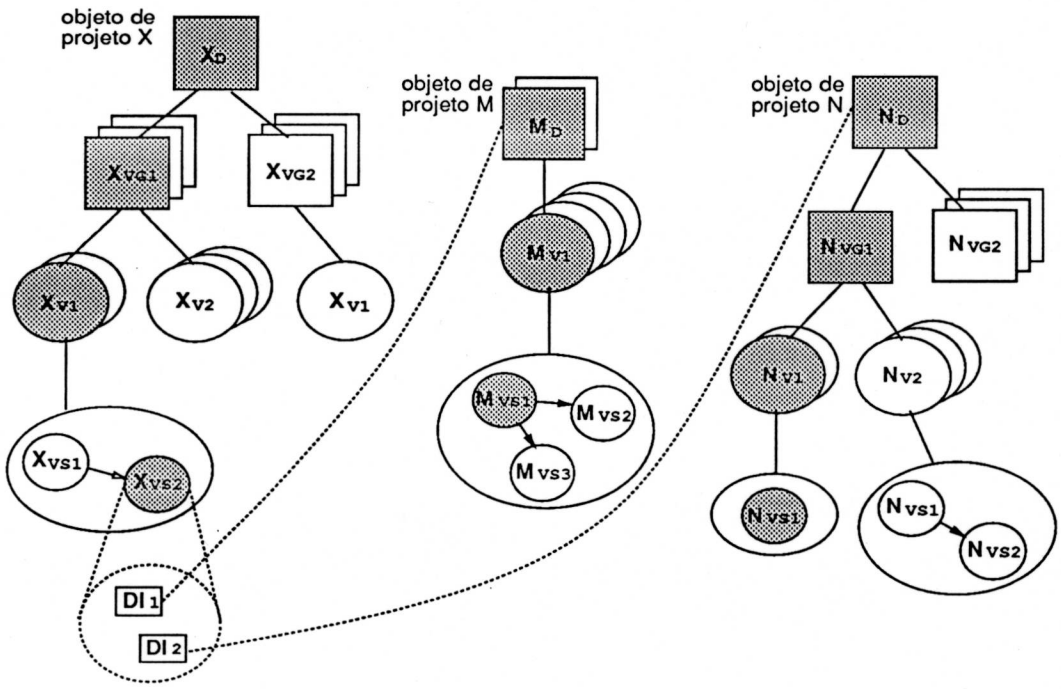

Figura 4.1: Configuração do objeto de projeto X

Por exemplo, na figura 4.1 é apresentada a configuração do objeto de projeto X. Através do *ViewState* escolhido X_{VS2}, composto, que possui dois subobjetos, são referidos os objetos de projeto M e N. Para o objeto M, é escolhido o *ViewState* primitivo M_{VS1}, ligado à *View* M_{V1}. Para o objeto de projeto N, é escolhido o *ViewState* primitivo N_{VS1}, que por sua vez é ligado à *View* N_{V1} e ao *ViewGroup* N_{VG1}. Cabe salientar que para cada nodo escolhido para fazer parte da configuração é selecionada uma de suas versões, que na figura aparece hachurada e

sobreposta em relação às demais versões do mesmo nodo. O conjunto de versões hachuradas determina a configuração do objeto de projeto X.

A diferença de abordagem entre *MHDViewStates* e *HDL/LayoutViewStates* pela presença das interconexões entre os sub-objetos não tem influência nas configurações. O objetivo de uma configuração é definir objetos a serem referidos pelos componentes de um *ViewState*, e isso se aplica a ambos os tipos de *ViewStates*, que podem ser compostos.

Há duas maneiras de definir o objeto referido por cada um dos componentes de um *ViewState*. A primeira delas é através do atributo de referência presente nos componentes, que deve fazer referência a um **objeto-referência**. Um objeto-referência pode ser um nodo do esquema de um objeto, uma versão de um nodo do esquema de um objeto, um *ViewState* ou um objeto-configuração. A criação de objetos-configuração, que serão abordados no decorrer deste capítulo, constitui a segunda maneira de definir objetos referidos para estabelecer ligações de configuração.

4.2 Configurações Estáticas, Dinâmicas e Abertas

No STAR, quando há a seleção de um objeto para ser referido por um componente de um *ViewState*, é estabelecida uma ligação entre eles. Para cada ligação estabelecida, o objeto selecionado pode estar definido até qualquer um dos níveis da hierarquia de definição. O conjunto das ligações estáticas e/ou dinâmicas estabelecidas a partir da definição das referências ao longo da hierarquia de composição do objeto, e o nível até o qual o último objeto vinculado está definido determinam a classificação das configurações em estáticas ou dinâmicas [GRA 91].

No STAR, as configurações podem ser estáticas, dinâmicas ou abertas. Uma configuração é **estática** quando há a seleção de uma versão para cada nível

das hierarquias de definição que fazem parte da sua hierarquia de composição, até o nível final, que é o de *ViewStates* primitivos. O objeto de projeto X (figura 4.1), de acordo com as referências estabelecidas para os objetos M e N, é um exemplo de configuração estática.

Uma configuração é **dinâmica** quando a seleção de versões ao longo da hierarquia de composição é feita até um outro nível que não o de *ViewStates* primitivos, como o de *ViewStates* compostos, *View, ViewGroup* ou *Design.* Nestes casos de configuração dinâmica, considerando-se que os últimos nodos ou *ViewStates* ligados aos componentes até então mais externos na hierarquia de composição não são necessariamente os objetos finais da configuração do objeto de projeto, ainda existirão níveis com várias alternativas de representação e versões, inclusive outros sub-objetos, determinando escolhas a serem feitas num momento posterior. Por exemplo, se os componentes M_{VS1} do objeto M e N_{VS1} do objeto N, ou apenas um deles, fossem compostos (figura 4.2), e a determinação da referência de seus componentes não estabelecida, teríamos uma configuração dinâmica.

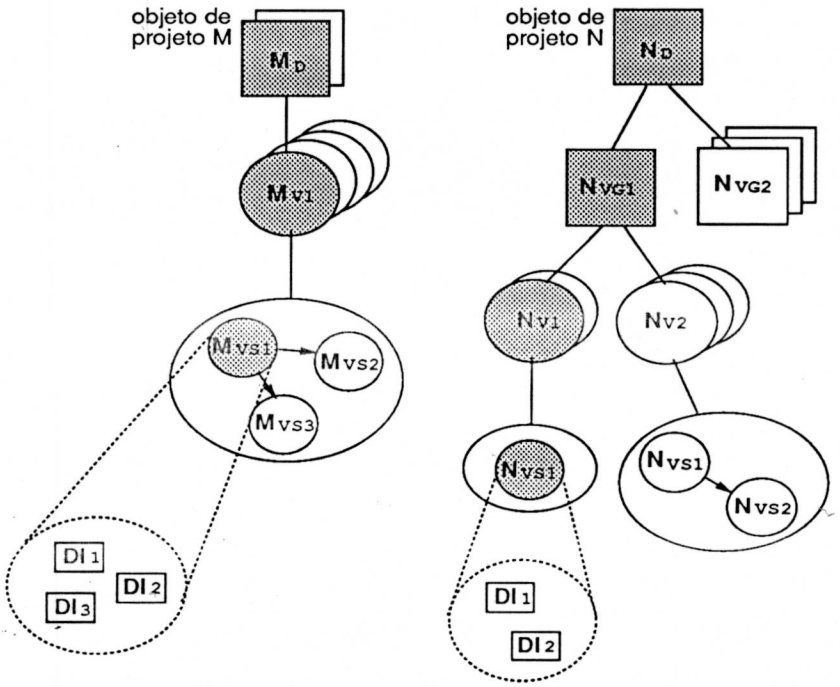

Figura 4.2: Objetos M e N, referidos como componentes de X

É possível, também, que o usuário não estabeleça ligações entre *Components* ou *DesignInstances* e outros objetos, não definindo, assim, os objetos referidos pelos sub-objetos de um *ViewState*. Neste caso, o tipo de configuração a ser estabelecida é **aberta**, e implica a escolha posterior de um objeto a ser referido por cada um dos componentes nos quais isso ainda não ocorreu. As configurações abertas possibilitam que se selecione objetos que não haviam sido definidos ou criados no momento da opção pela não referência a objetos, o que muitas vezes ocorre nos projetos de concepção *top-down*.

4.3 O Objeto Configuração

Para auxiliar no processo de configuração, acrescentou-se ao modelo de dados do STAR um objeto-configuração, denominado *ConfigurationBody*, que não é versionável. Este objeto armazena parte da configuração de um objeto de projeto referente a um ou mais componentes de um *ViewState*. Para o objeto determinado pelo *Design* que dá origem à hierarquia de definição à qual pertence o *ViewState*, o *ConfigurationBody* determina uma **configuração local**. Uma configuração local é sempre restrita a uma hierarquia de definição.

Um *ConfigurationBody* é vinculado ao *ViewState* do *Design* do objeto para o qual se estabelece a configuração local. Deste *ViewState* ele pode herdar atributos, mas pode também ter seus próprios atributos definidos (através dos *UserFields*). Um *ViewState* pode ter vários *ConfigurationBodies* a ele vinculados, e eles podem ser complementares uns em relação aos outros, ou não. Isso possibilita que eles especifiquem as ligações de um ou mais componentes do *ViewState*, sendo que também é possível que um *ConfigurationBody* especifique ligações para todos os componentes do *ViewState*.

Quanto à estrutura do *ConfigurationBody*, ele deve fazer referência a *ViewStates* primitivos ou compostos, ou a *ConfigurationBodies*, que vincularão ou-

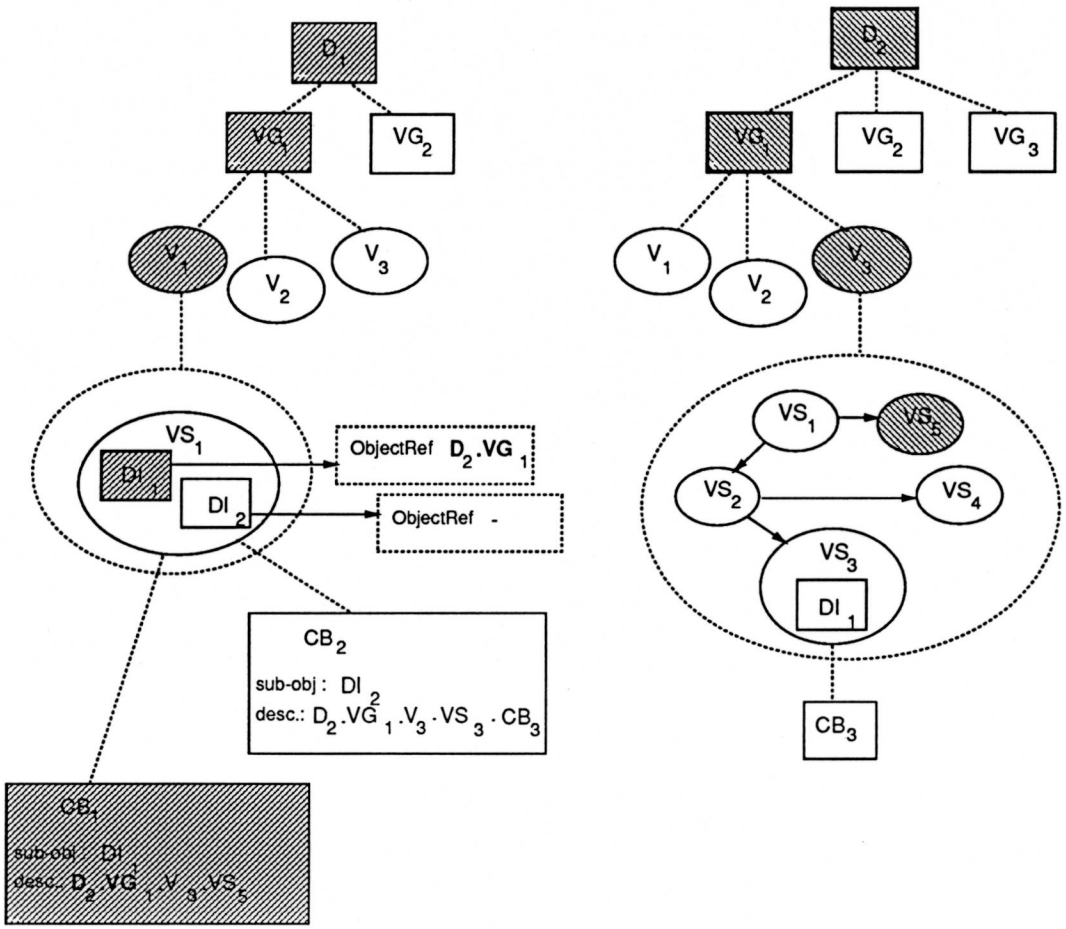

Figura 4.3: Objetos D_1 e D_2 descritos no STAR

tras referências à seqüência da descrição. A única condição é que caso haja alguma referência estabelecida no componente, através de seu atributo de referência, para o qual se está definindo a configuração no *ConfigurationBody*, ela deve ser respeitada. Ou seja, a referência no *ConfigurationBody* deve ser estabelecida a partir do atributo de referência.

Se no *ViewState* houver referência a objetos através de seus componentes sem que se estabeleça uma configuração completa, como no caso das configurações dinâmicas, pode-se utilizar o *ConfigurationBody* para complementar a descrição até um *ViewState*, levando em consideração os objetos já referidos e respeitando a hierarquia de definição do objeto. Por exemplo (figura 4.3), se um *DesignInstance* $D_1.VG_1.V_1.VS_1.DI_1$ fizer referência a um *ViewGroup* $D_2.VG_1$, o *ConfigurationBody*

$D_1.VG_1.V_1.VS_1.CB_1$, hachurado na figura, vinculado ao respectivo *ViewState* complementará a descrição, designando as representações escolhidas em cada nível da estrutura de controle do *Design* D_2, a partir de $D_2.VG_1$ até um de seus *ViewStates*, que poderia ser $D_2.VG_1.V_3.VS_5$.

Se não houver atributo de referência definido, a descrição do *ConfigurationBody* pode indicar qualquer objeto, desde que seja um *ViewState* ou um outro *ConfigurationBody*.

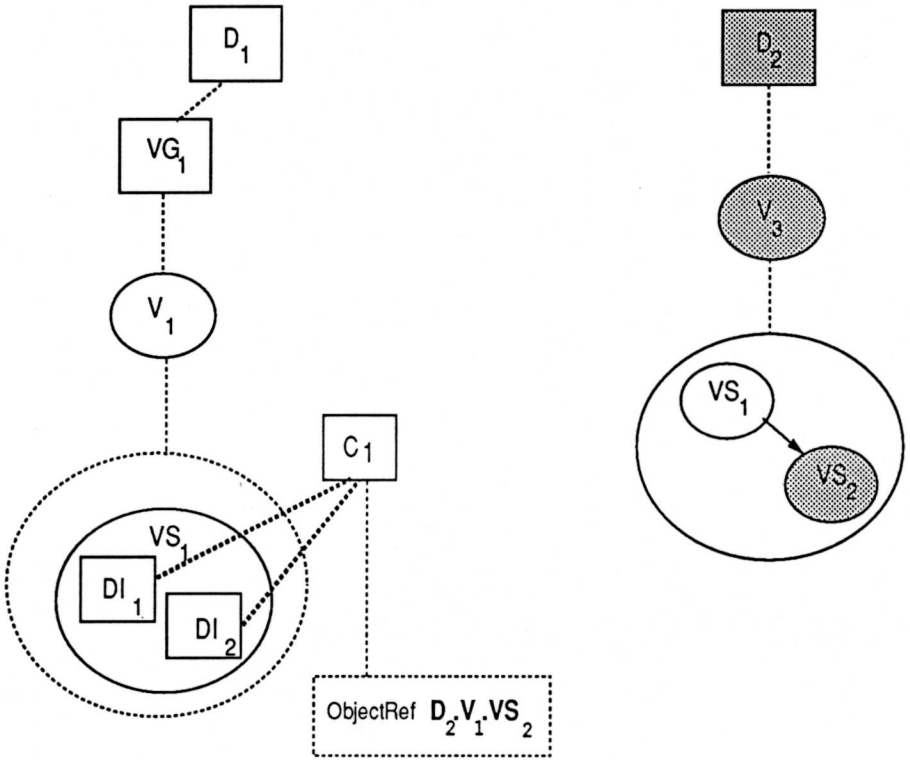

Figura 4.4: Definição de um *Component*

Cabe salientar que no caso da referência a um objeto ser definida num *Component*, todos os *DesignInstances* a ele vinculados farão referência a esse mesmo objeto. Por exemplo, na figura 4.4, os *DesignInstances* DI_1 e DI_2 do *ViewState* $D_1.VG_1.V_1.VS_1$ estão vinculados ao *Component* C_1. A referência ao objeto $D_2.V_1.VS_2$, definida em C_1, faz com que os *DesignInstances* a ele vinculados façam referência a esse mesmo objeto. Assim, é preciso mapear *Ports* do objeto referido

a *Ports* do *Component*, e esse mapeamento é seguido por todos os *DesignInstances* ligados ao *Component*.

Uma referência pode indicar também o *ViewState* corrente, ao invés de um *ViewState* específico. Isso é possível pela presença da expressão **"current-version"** na referência, no local onde se descreveria um *ViewState*. Assim, quando um *ConfigurationBody* que possua uma referência desse tipo for utilizado na configuração de um objeto, a qual será submetida a uma ferramenta, essa referência será substituída pelo *ViewState* corrente naquele momento ligado à *View* especificada na descrição do *ConfigurationBody*.

Além do estabelecimento dos objetos-referência para os componentes, também fazem parte da configuração o mapeamento de portas e parâmetros entre componentes e os respectivos objetos referidos. A referência e os mapeamentos podem ser definidos nos próprios componentes ou nos *ConfigurationBodies*. Só não pode haver duplicação ou redefinição de algum deles, ou seja, não se pode definir a referência, ou algum dos mapeamentos, num componente e também num *ConfigurationBody* vinculado ao mesmo *ViewState* ao qual pertence esse componente.

É possível utilizar aninhamento de configurações, desde que se possa estabelecer relações consistentes nas ligações entre objetos. Assim, uma configuração pode fazer uso de outras configurações para complementar uma descrição de objeto, cada uma delas fazendo referência a outra, se for o caso, e assim por diante. Isso faz com que se possa ter, além de *Designs, ViewGroups, Views* e *ViewStates*, *ConfigurationBodies* na descrição de uma configuração. Por exemplo (figura 4.3), no caso do *ConfigurationBody* CB_2 que define a referência da *DesignInstance* $D_1.VG_1.V_1.VS_1.DI_2$ para $D_2.VG_1.V_3.VS_3.CB_3$. O *ViewState* VS_3 tem um componente, DI_1, e um *ConfigurationBody* vinculado, $D_2.VG_1.V_3.VS_3.CB_3$, onde existe uma descrição de configuração possível para este componente.

4.4 Armazenamento de Configurações

O fato de uma configuração ser estática, dinâmica ou aberta está relacionado com a descrição armazenada do objeto. Se para uma simulação, por exemplo, faz-se a escolha de versões que completam uma configuração dinâmica anteriormente estabelecida, e estas escolhas não são armazenadas, cada vez que se quiser submeter o objeto de projeto a uma simulação, com esses mesmos objetos de escolha, será necessário fazer novamente a seleção. Como muitas vezes poderia haver interesse do projetista em manter este conjunto selecionado, é importante poder armazená-lo.

No STAR há duas formas de armazenar configurações, diretamente relacionadas às maneiras de referir objetos: definindo um objeto-referência no atributo de referência, que é armazenado como parte do objeto, ou definindo um objeto-configuração. O objeto-configuração permite que se armazene, por exemplo, a descrição completa de um objeto-referência que complementa a descrição do atributo de referência, até os *ViewStates* primitivos. Isso determina uma configuração estática, sem que se altere a referência dinâmica já estabelecida no atributo de referência, o que permite manter a flexibilidade de diversas opções de representação a partir de um certo nível de definição.

4.5 Problemas de Referência

As referências a outros objetos numa composição podem gerar **problemas de referência**, como no GARDEN [WAG 91a], nos quais o objeto referido possui mais *Ports* do que o componente que lhe faz referência. Esta é uma situação possível, e para os *LayoutViewStates* e *HDLViewStates* não acarretam problemas, uma vez que não há definição de interconexões entre os sub-objetos. Neste caso, estes *Ports* ficariam, a princípio, sobrando, e assim desligados, o mesmo ocorrendo com os *Ports* que fossem definidos depois do estabelecimento da ligação nos nodos

seguintes da hierarquia de composição do objeto referido. Cabe à aplicação ou ao projetista estabelecer se eles serão utilizados ou não. O que não pode acontecer é a existência de um número menor de *Ports* no objeto referido em relação ao número mínimo necessário definido no componente que faz a referência.

Já para os *MHDViewStates* existe a definição das interconexões, através das *Nets*, o que significa que a relação entre os *Ports* dos componentes, e conseqüentemente entre os *Ports* dos objetos referidos, tem que ser bem definida. A criação de *Ports* nos nodos que seguem o nodo referido na hierarquia de definição após a definição das interconexões, como por exemplo a criação de *Ports* nos *ViewGroups* e *Views* enquanto que a interconexão entre os componentes havia sido definida em função dos *Ports* do *Design*, podem gerar problemas para o usuário. Isto porque eles não estão definidos nas interconexões e poderão ficar perdidos na seqüência da definição do objeto. O controle dessa situação ficará sob responsabilidade do usuário, que tomará cuidado para que isso não ocorra, ou contornará a situação de outra forma, como por exemplo gerando uma nova versão da *Net* que abranja as interconexões com estes novos *Ports*.

4.6 Classificação das Configurações

A partir do estabelecimento dos tipos de configurações a serem considerados, e de acordo com o nível de definição do objeto de projeto na sua composição, elas podem ser **completas** ou **incompletas**. Além disso, considerando-se a existência de dois escopos para estabelecer referências para os componentes de um *ViewState*, que são eles próprios ou um *ConfigurationBody*, a identificação de objetos para uma configuração pode ser **definida** ou **indefinida**. O esquema de classificação pode ser observado na figura 4.5.

Uma configuração estática é completa, pois apresenta uma única definição possível para o objeto. Já uma configuração dinâmica é incompleta, pois é

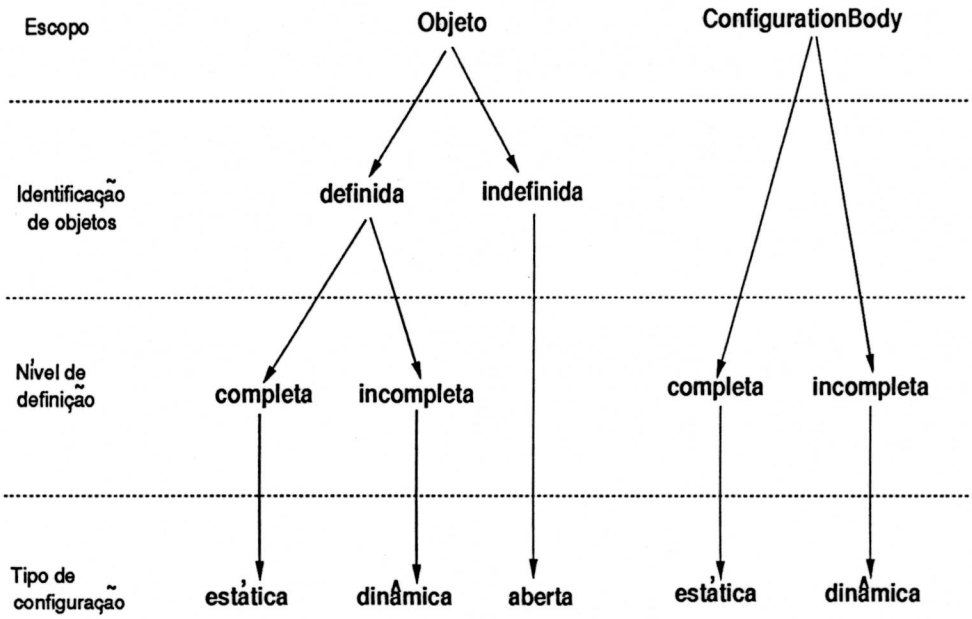

Figura 4.5: Classificação das configurações

preciso fazer escolhas entre os objetos seguintes na hierarquia de composição, até o nível de *ViewStates* primitivos, para que se tenha uma definição única e completa para o objeto considerado.

Sob o ponto de vista do objeto no qual existem sub-objetos para os quais serão estabelecidas referências, a identificação dos objetos-referência para uma configuração pode ser definida ou indefinida. Na identificação definida, os objetos referidos pelos componentes de um *ViewState* são definidos nos *DesignInstances* ou *Components* pela nomeação do seu atributo específico, e dependendo do nível de definição atingido, a configuração será estática ou dinâmica. Já na identificação indefinida não se estabelece referências nos componentes, e os objetos referidos são definidos apenas no *ConfigurationBody*, o que determina uma configuração aberta.

Desse modo, num caso de identificação definida é feita a escolha de um *ViewState, View, ViewGroup, Design* ou mesmo *ConfigurationBody* para ser referido por cada componente existente no *ViewState* a ser configurado. E na identificação indefinida, o estabelecimento da referência no componente é deixado em aberto, sendo feito no *ConfigurationBody*. Na realidade, a escolha de um *ConfigurationBody*

caracteriza a escolha de um ou mais *ViewStates*, como objetos referidos, um para cada componente, dando seqüência a uma composição.

Pode-se dizer, então, que a identificação definida determina **configurações com referências estabelecidas**, enquanto que a indefinida determina **configurações sem referências estabelecidas.** No processo de escolha dos objetos para se chegar a uma configuração completa, pode-se intercalar livremente, ou mesmo utilizar em conjunto, estas duas formas de estabelecer referências.

Sob o ponto de vista do objeto-configuração, o nível de definição que ele atingir determinará configurações completas ou incompletas, e respectivamente estáticas ou dinâmicas.

4.7 Objetos, Versões e Configurações

O mecanismo de gerência de configurações, que está sendo definido neste trabalho trabalha sobre as versões dos objetos do modelo de dados, controladas e mantidas pelo gerente de versões do STAR.

Cada nível do esquema de um objeto de projeto (*Design, ViewGroup* ou *View*) determina um nível de abstração na definição do objeto de projeto, onde existem nodos que contêm suas alternativas de representação nesse nível, com um conjunto de versões para cada nodo. Assim, há nodos do esquema do objeto e versões dos nodos do esquema do objeto em cada nível. No nível inferior de versionamento, tem-se apenas versões de objetos, os *ViewStates*.

As configurações no STAR são consideradas de modo similar às configurações de Katz, pois sua definição completa é estabelecida a partir da seleção de uma versão de um nodo a cada nível do esquema do objeto de projeto seguida da seleção de um *ViewState*, e assim ao longo de sua hierarquia de composição.

Utiliza-se o termo **configuração**, de uma forma geral, como sinônimo de um conjunto de ligações estabelecidas para um objeto de projeto ao longo de sua hierarquia de composição que lhe estabelecem uma descrição única. O *ConfigurationBody*, ou **objeto-configuração** define ligações para uma parte do objeto de projeto determinada por uma de suas hierarquias de definição. Ele estabelece ligações entre componentes do *ViewState* e objetos-referência que são por eles referidos.

Há diferença quanto aos objetos que podem ser envolvidos na determinação de configurações, caso elas sejam estabelecidas no *ConfigurationBody* ou através do atributo de referência de *Components* ou *DesignInstances*. Num *ConfigurationBody*, descreve-se *ViewStates* ou outros *ConfigurationBodies*, enquanto que no atributo de referência de um sub-objeto pode-se descrever qualquer um dos objetos-referência. Em ambos os casos as configurações poderão ficar incompletas e terão que ser complementadas posteriormente, quando for necessária uma configuração completa.

Optou-se pelo não versionamento dos objetos-configuração, pois isso implicaria um grande número de objetos adicionais, em função da vinculação desses objetos aos *ViewStates*, que por si já determinam um número significativo de objetos. Além disso, como é possível a utilização de *ConfigurationBodies* aninhados, a existência de versões de objetos-configuração tornaria o processo de configuração mais complicado, e seria necessário um mecanismo de controle de versões em configurações.

5 O GERENTE DE CONFIGURAÇÕES DO STAR

Um estudo inicial sobre configurações [GRA 91], a necessidade de sua incorporação ao ambiente a as características do modelo de dados levaram ao desenvolvimento de um gerente de configurações para o ambiente STAR. Ele tem como funções a criação e a manipulação de configurações de objetos do ambiente. Esse gerente de configurações integra-se ao gerente de versões estabelecido por [LAC 92], como ferramenta de manipulação de objetos para a definição de descrições individuais de um artefato em projeto. Com isso, é possível testar se determinadas composições de estrutura para o artefato são satisfatórias, estabelecendo novas alternativas de projeto se elas não corresponderem às expectativas, ou consolidando descrições consideradas adequadas.

Neste capítulo, são apresentados os estados de consistência dos objetos-configuração, as operações sobre configurações e a linguagem para seleção de objetos. Em seguida caracteriza-se o processo de construção de configurações através do funcionamento das operações.

5.1 Notificação e Propagação de Mudanças

Quando há alterações em objetos, são geradas novas versões e os objetos que têm versões referidas por outros objetos podem ficar desatualizados. Por isso, é importante poder atualizar as referências dos objetos que utilizam as versões antigas como componentes pelas versões mais recentes. Para tanto, é interessante estabelecer um mecanismo de propagação de mudanças, que faça essa atualização de modo automático. Mesmo assim, muitas vezes a mudança automática de versões pode não ser desejável.

Para o STAR, numa fase inicial, o usuário apenas é notificado da mudança. Qualquer alteração na configuração tem que ser feita explicitamente pelo usuário, em cada local onde foi utilizada uma versão do objeto modificado, uma vez que não foram estabelecidos recursos para automatizar o processo. Isso é válido tanto para o versionamento conceitual quanto para o versionamento em nível inferior.

Em [LIV 92], é apresentado o Gerente de Cooperação do STAR, que coordena o empréstimo de objetos, inclusive providenciando eventuais substituições de versões em empréstimos quando ocorre a geração de versões mais atualizadas para os objetos. Numa fase posterior, está prevista a integração entre o Gerente de Configurações e o Gerente de Cooperação, uma vez que ele pode suprir em parte a necessidade de identificação e propagação de mudanças.

5.2 Estados de Consistência

Versões dos objetos do STAR estão em algum estado de consistência (em trabalho, estável ou consolidada) que reflete o seu estágio de desenvolvimento.

Para as configurações também se considera estados de consistência, seja para os objetos-configuração, aos quais eles são atribuídos, ou mesmo para as referências pelo atributo de referência, onde o usuário tem que fazer o controle a partir dos estados de consistência dos objetos envolvidos, conforme será visto a seguir.

5.2.1 Estados de Consistência dos ConfigurationBodies

O estado de consistência atribuído a um *ConfigurationBody* deve ser coerente e de acordo com o(s) estados(s) de consistência dos objetos envolvidos, o composto e os componentes. Isso faz com que se tenha, por um lado *ViewStates*

com componentes que fazem a referência, e por outro *ViewStates* ou *Configuration-Bodies* referidos. Inicialmente, quando são criados, os *ConfigurationBodies* estão em trabalho e podem ser promovidos no decorrer do processo, conforme as condições estabelecidas no decorrer desta seção. As promoções não são automáticas, e sim determinadas pelo usuário.

estado de consistencia do ViewState ou ConfigurationBody referido	estado de consistencia do ViewState cujos componentes estabelecem referencias: em trabalho	estavel	consolidado
em trabalho	configuracao em trabalho	configuracao em trabalho	configuracao em trabalho
estavel	configuracao em trabalho	configuracao estavel	configuracao estavel
consolidado	configuracao em trabalho	configuracao estavel	configuracao consolidada

Figura 5.1: Estados de consistência dos *ConfigurationBodies*

Na figura 5.1 pode ser observado o estado de consistência máximo que o objeto-configuração pode atingir, de acordo com os estados de consistência dos objetos envolvidos. Ele sempre é ao menos igual ao estado do objeto envolvido que indica o menor estágio de desenvolvimento (o menor estágio de desenvolvimento é característica dos objetos em trabalho, e o maior é dos objetos consolidados). São as seguintes as regras básicas para promoções:

- enquanto houver um *ViewState* ou *ConfigurationBody* em trabalho presente no *ConfigurationBody*, ou o *ViewState* ao qual ele está associado estiver em trabalho, ele permanece em trabalho e só pode ser utilizado pelo projetista proprietário.

- quando todos os *ViewStates* ou *ConfigurationBodies* presentes estiverem ao menos no estado estável, e o *ViewState* ao qual está associado também, o *ConfigurationBody* pode ser promovido a estável e pode ser utilizado pelos projetistas pertencentes ao mesmo grupo do projetista proprietário.

- quando todos os *ViewStates* ou *ConfigurationBodies* presentes estiverem no estado consolidado, e o *ViewState* pai também, o *ConfigurationBody* poderá ser promovido a consolidado e, assim, poderá ser utilizado por todos os projetistas.

Uma configuração ao ser criada na forma de um *ConfigurationBody* pode fazer uso de *ViewStates* ou *ConfigurationBodies* em qualquer estado de consistência, desde que apropriados em relação ao composto e componente envolvidos. As promoções devem ser feitas com base nas regras estabelecidas anteriormente.

Outra regra existente em função dos estados de consistência que diz respeito à manipulação de objetos determina que versões em trabalho de um projetista só podem ser usadas nas configurações que ele especificar. Enquanto os objetos-configuração estão em trabalho, eles podem ser alterados livremente.

5.2.2 Estados de Consistência para Configurações Estabelecidas por Atributo de Referência

Até aqui abordou-se estados de consistência de configurações estabelecidas através da criação de objetos-configuração. Já para configurações estabelecidas nos próprios componentes de um *ViewState* pela instanciação do atributo de referência, é preciso que o projetista controle os estados de consistência do *ViewState* e os dos objetos-referência a ele vinculados no caso de querer fazer promoções. O componente do *ViewState* sempre tem que estar ao menos no mesmo estado de consistência para o qual o objeto-referência vai ser promovido. Poderiam ocorrer problemas, por exemplo, no caso de se querer promover a estável um *ViewState* em

trabalho que tenha um *DesignInstance* que tenha como objeto-referência um objeto em trabalho. A referência do *DesignInstance* ficaria inconsistente no caso do objeto-referência ser removido.

5.3 Operações sobre Configurações

A definição de operações permite regulamentar as ações que determinam a modificação do conjunto de objetos de projeto, ou de cada um em particular.

As configurações são definidas por meio dos *ConfigurationBodies* ou pelas referências estabelecidas no atributo de referência de *Components* e *DesignInstances*, que também funcionam como elementos norteadores para os objetos-configuração, e devem ser respeitados. Alterações no atributo de referência de *ViewStates* estáveis ou consolidados determinam seu versionamento e a alteração no grafo de *ViewStates* que ele acarreta diz respeito à estrutura do objeto, sendo de responsabilidade do gerente de versões. Por isso, as operações aqui definidas estão de um modo geral relacionadas aos *ConfigurationBodies* e atuam, num primeiro momento, sobre uma hierarquia de definição do objeto.

As operações possíveis sobre as configurações, enquanto objetos do modelo de dados, são:

- criação de configurações,
- alteração de configurações,
- consulta a configurações,
- remoção de configurações,
- escolha de objetos para configuração e
- cópia de configuração.

Através destas operações é possível estabelecer uma configuração de um objeto de projeto para ser submetida às ferramentas.

A partir da indicação inicial de um *ViewState* de objeto, seleciona-se para ele uma configuração e, a partir da descrição de objetos por ela estabelecida para os sub-objetos, escolhe-se as próximas, até obter uma configuração completa. Caso existam objetos sem descrição de configuração estabelecida, o usuário poderá fazer escolha de objetos-referência, com ou sem criação de *ConfigurationBodies.* Estes aspectos serão abordados no item de construção de configurações.

Não haverá operação de alteração sobre os objetos-configuração estáveis e consolidados, pois causaria o surgimento de novas versões, e optou-se pelo não versionamento das configurações. Assim, se for necessário fazer alterações, haverá a criação de novos objetos-configuração.

Para um *ConfigurationBody* estabelecido, o versionamento dos nodos aos quais estão relacionados os *ViewStates* referidos na configuração que ele determina, ou o versionamento do *ViewState* pai não provocam nele nenhuma alteração automática. A atualização de versões que fazem parte da configuração, caso desejada e possível, deverá ser feita explicitamente pelo projetista através das operações disponíveis, pois não há um mecanismo de propagação de mudanças.

Quando um novo *ViewState* é criado pelo versionamento do *ViewState* pai, ele não herda automaticamente *ConfigurationBodies* definidos nos antecessores. Caso se queira ter para um *ViewState* qualquer uma mesma descrição de configuração já definida para um outro, isso pode ser feito explicitamente, através da operação de cópia, que permite que um *ConfigurationBody* importe a descrição de um outro já definido, e que será melhor explicada na definição das operações.

5.3.1 Criação de Configurações

É determinada pela definição de uma configuração para um objeto no escopo de sua hierarquia de definição. Pode desencadear a criação de um objeto-configuração ou apenas o estabelecimento de uma descrição temporária de configuração. De qualquer modo, sua descrição é agregada à descrição do objeto de projeto para o qual se estabelece a configuração.

Na criação de um *ConfigurationBody*, lhe é atribuída uma identificação única dentro do espaço de trabalho do projetista proprietário. Como para os demais objetos, sua existência é vinculada à existência de seus ancestrais na hierarquia de definição. Ou seja, é preciso existir *Design, ViewGroup, View* e *ViewState* de um determinado objeto para que se crie para ele um *ConfigurationBody.*

Ao se criar um *ConfigurationBody*, seu estado de consistência atribuído inicialmente é em trabalho, conforme já estabelecido anteriormente.

A criação de configurações é amparada pelas operações de cópia e escolha de objetos para configurações.

5.3.2 Alteração de Configurações

Pode-se alterar apenas *ConfigurationBodies* em trabalho, pois estes objetos não são versionáveis e modificações em objetos em trabalho não ocasionam geração de novas versões. Como não é possível fazer alterações em objetos-configuração estáveis ou consolidados, deve-se criar novos objetos com as mudanças desejadas.

5.3.3 Consulta a Configurações

A partir da definição de um *ViewState*, é possível consultar os *ConfigurationBodies* e ele vinculados. Caso não existam, o usuário ainda assim poderá ver as referências estabelecidas para os componentes do *ViewState*, desde que eles tenham atributos de referência definidos.

5.3.4 Remoção de Configurações

É possível remover um *ConfigurationBody* que se encontra nos estados de consistência "em trabalho" ou "estável", desde que ele não seja utilizado na descrição de outro objeto configuração, pois o gerente de configurações impediria a remoção do *ConfigurationBody* por uma questão de integridade.

Caso não se enquadre na situação anterior, a remoção de um *ConfigurationBody* que complementava a descrição de uma referência estabelecida no sub-objeto não ocasiona nenhum problema ou inconsistência, apenas faz com que a configuração volte a ser incompleta.

5.3.5 Escolha de Objetos para Configuração

A seleção de objetos para definir configurações será realizada a partir de um critério estabelecido, escolhido entre seleção manual, automática pela versão corrente ou mais recente, ou através da definição de expressões de configuração. Esta seleção deverá indicar uma versão de objeto a nível de *ViewState*, ou outro *ConfigurationBody*. Caso o atributo de referência de um componente esteja nomeando um objeto-referência que não seja um *ViewState* ou *ConfigurationBody*, e se queira fazê-lo pela definição de um objeto-configuração, a escolha destes objetos deverá ser feita a partir do objeto-referência estabelecido no atributo de referência.

Para suportar as expressões de configuração, definiu-se uma linguagem textual de seleção de objetos baseada nos seus atributos, que permite escolher um objeto ou restringir o universo de escolha, direcionando e tornando mais objetiva a seleção de objetos-referência para as configurações.

5.3.6 Cópia de Configuração

Muitas vezes num *ViewState* altera-se um atributo qualquer, que no caso do *ViewState* ser estável ou consolidado determina seu versionamento. Neste caso, tem-se um novo *ViewState*, que em relação ao anterior perdeu os *Configurations-Bodies* vinculados, mesmo que a descrição de algum deles fosse desejável para este novo *ViewState*.

Há, também, ocasiões em que simplesmente se desejaria utilizar para esse novo *ViewState* uma descrição de configuração definida para um outro *ViewState*.

Para que não seja necessário definir novamente a descrição de um objeto-configuração quando o usuário já o conhece de algum outro *ViewState*, ele pode utilizar a operação de cópia de configuração. Essa operação consiste na importação de uma descrição de um *ConfigurationBody* para um outro *ConfigurationBody*, sem qualquer ligação ou vínculo existencial ou estrutural entre eles.

5.4 Critérios para Escolha de Objetos

A tarefa de escolher objetos para estabelecer as configurações pode ser manual, automática ou semi-automática [GRA 91]. Um processo totalmente manual é demorado, principalmente se a composição do objeto tiver muitos níveis e envolver muitos objetos, além de trabalhoso e sujeito a erros, muitos decorrentes da própria imperfeição do ser humano. Os critérios são os mais variados possíveis, por conta

do usuário. Já num processo automático, ou na parte automática de um processo semi-automático, os critérios precisam ser bem definidos.

No STAR, a escolha de objetos pode ser feita pelos seguintes modos: manual, semi-automático pela definição de expressões de configuração, e automático pela versão mais recente ou pela versão corrente. O modo *default* é o automático pela versão corrente. Pelo processo semi-automático, o usuário escolhe um dos critérios disponíveis e a partir dele o sistema realiza a busca de *ViewStates*. Se houver mais de um, o usuário tem que optar por um deles.

No esquema do objeto, o conceito de versão corrente confunde-se com o de versão mais recente, mas entre *ViewStates* pode designar objetos distintos.

Assim, tem-se que os critérios disponíveis no gerente de configurações do ambiente STAR são:

- expressões de configuração: construídas com base na linguagem para seleção de objetos para configuração, a ser abordada na seção seguinte.
- versão corrente: o usuário escolhe inicialmente um *Design, ViewGroup* ou *View*, e entre os *ViewStates* vinculados é selecionado o corrente. Caso a escolha inicial designe um nodo no nível de *Design* ou *ViewGroup*, o nodo corrente, ou mais recente, do(s) nível(eis) seguinte(s) (*ViewGroups* ou *Views*) será considerado automaticamente.
- versão mais recente: o usuário estabelece inicialmente um *Design, ViewGroup* ou *View*, e entre os *ViewStates* que esta escolha abranger é selecionado o mais recente. Se for estabelecido inicialmente um *Design* ou *ViewGroup*, no(s) nível(eis) seguinte(s) será escolhido automaticamente o nodo corrente, ou mais recente.

5.5 Linguagem para Seleção de Objetos para Configurações

Os métodos manual e automático, ou a mistura deles, não apresentam muitas opções para que o usuário possa selecionar versões. As características e atributos dos objetos são elementos mais intuitivos ao usuário quando quer-se referir a eles.

Muitos autores já utilizaram este recurso em linguagens para seleção de objetos. Belkhatir em [BEL 87] propõe uma linguagem, com comandos bem definidos, que tem por base restrições sobre atributos, a qual permite selecionar objetos para as configurações do sistema ADELE. Dias em [DIA 91] apresenta uma linguagem para recuperação de objetos, versões e configurações sob a forma de regras de seleção, que pode ser considerada como sua linguagem de consulta. Uma linguagem para seleção de objetos também é utilizada no sistema SDE [SKM 91].

Para aproveitar a associação entre objetos e suas características como facilidade para recuperá-los, estabeleceu-se como um recurso para a seleção de objetos no STAR uma linguagem textual. Através dela, é possível construir **expressões de configuração**, que têm como objetivo restringir o conjunto de objetos candidatos a serem referidos pelos componentes numa composição, tornando mais objetiva a escolha do usuário.

A idéia das expressões de configuração do STAR tem por base uma linguagem de consulta, como a presente em [DIA 91], que permite o acesso às versões de acordo com regras de seleção. Como ainda não houve a definição de uma linguagem de consultas para o STAR, as expressões de seleção são estabelecidas de acordo com uma linguagem de seleção de versões que foi criada especificamente para a busca de objetos para configurações.

A estrutura das expressões pode ser observada na figura 5.2. Cada expressão de configuração se aplica a um *ViewState*, e define, para cada um de seus

componentes, as restrições que determinarão os respectivos *ViewStates* passíveis de seleção, a partir de expressões de seleção. Cada expressão de seleção é específica para um componente e é formada a partir de sub-expressões e operadores lógicos AND, OR e NOT.

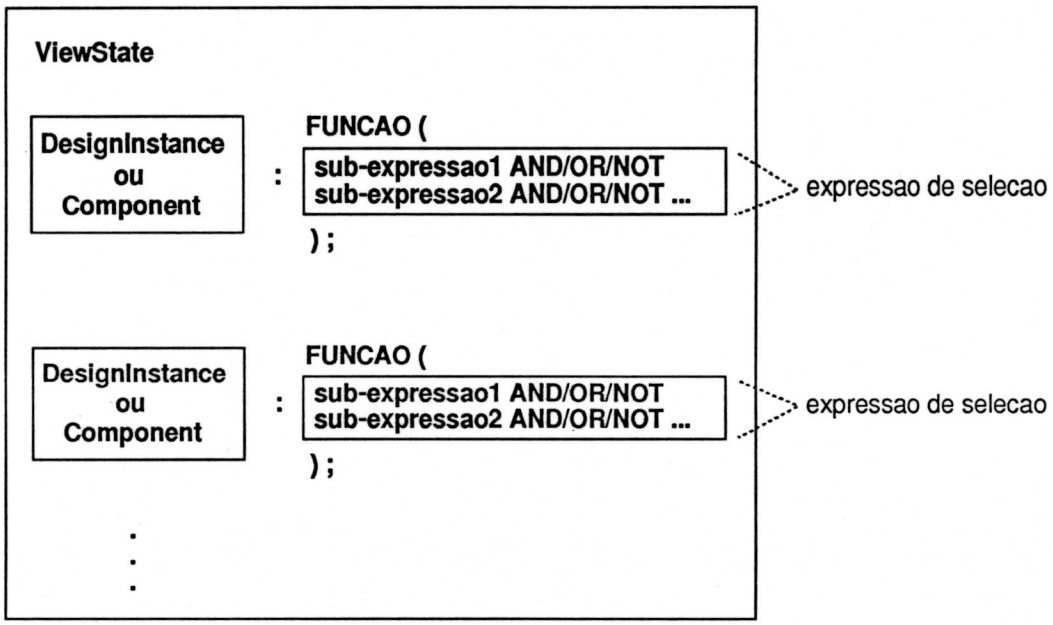

Figura 5.2: Estrutura das expressões de configuração

Uma configuração de objeto do STAR só fica completa com a vinculação de *ViewStates*, sendo que para os objetos-folha da hierarquia de composição eles devem ser primitivos. Cada expressão de seleção tem como resultado um conjunto de *ViewStates*, com nenhum, apenas um ou vários *ViewStates* como elementos.

Mas apenas um *ViewState* deve ser vinculado a cada componente. No caso da expressão de seleção ter como resultado vários *ViewStates*, este é definido por uma função escolhida pelo usuário. Assim, tem-se as seguintes situações possíveis, de acordo com o conjunto resultante a partir das expressões de seleção:

- conjunto vazio: não foi selecionado *ViewState*. O usuário deve construir outra expressão de seleção para este objeto.
- conjunto unitário: foi realizada a seleção de apenas um *ViewState*.

- conjunto com vários elementos: mais de um *ViewState* foi selecionado. Será aplicada uma função sobre este conjunto, a qual determina um novo modo de seleção a ser aplicado sobre ele: seleção manual, por versão corrente, por versão mais recente ou ainda pela redefinição da expressão de configuração. Esta função retorna um *ViewState*.

As funções que definem os modos de seleção que poderão ser aplicados no caso da expressão de seleção determinar um conjunto com mais de um *ViewState* são: MANUAL(), V_CURRENT(), V_MOST_RECENT() e REDEFINITION(). A escolha da função é realizada pelo usuário durante processo de configuração e não faz parte da sintaxe da linguagem.

Após selecionado um *ViewState* único, se ele for composto e tiver *ConfigurationBodies* vinculados, o usuário poderá escolher um deles. Desta forma assegura-se que um *ConfigurationBody* que venha a ser criado terá, para cada componente presente ou uma descrição de objeto até o nível de *ViewStates* ou outro objeto-configuração associado.

Assim como no *ConfigurationBody*, uma expressão de configuração contém apenas os componentes de um *ViewState* para os quais se quer estabelecer referências naquele momento.

As expressões de seleção que determinam as restrições em relação ao conjunto de objetos do STAR, têm como operandos atributos de objetos (*Userfields, Ports, ...*), nodos do esquema de objetos e *ViewStates*, sobre os quais atuam operadores. A utilização de nodos do esquema de um objeto tem por finalidade restringir o sub-conjunto selecionado às suas versões.

Os operadores que podem ser empregados nas sub-expressões são:

- relacionais: estabelecem relações de grandeza e igualdade entre atributos e valores. São eles: $=$, $\neq$, $>$, $<$, $>=$ e $<=$

- existenciais: são aplicados a atributos, e determinam a seleção de *ViewStates* que possuem ou não este atributo definido, mesmo que ele tenha sido herdado. São eles: EXISTS e NOT EXIST.

- outros:

  - MAX e MIN:aplicados a um atributo, determinam a seleção do *ViewState* que tem, respectivamente, o maior ou o menor valor entre os que possuem este atributo.

  - FIRST e LAST: aplicados a um nodo de um esquema de objeto, determinam, respectivamente, a seleção da primeira e da última versão desse nodo.

  - OBJECT: aplicado a um nodo de um esquema de objeto, determina a seleção de todos os *ViewStates* ligados a ele.

  - CURRENT: aplicado a um *Design, ViewGroup* ou *View*, seleciona o(s) *ViewState(s)* correntes. (no caso da *View* há um *ViewState* corrente, mas no caso de *Design* e *ViewGroup* há um *ViewState* corrente para cada *View* vinculada).

A gramática da linguagem para seleção de objetos encontra-se no Apêndice A-1.

5.6 Construção de Configurações

Após a definição conceitual de configurações no ambiente STAR e do estabelecimento de suas características e operações que permitem sua manipulação, é necessário descrever o funcionamento e modo de interação do gerente de configurações com os objetos de projeto e com o usuário. A interação com os objetos ainda é um pouco difícil de ser realizada devido à ausência de uma linguagem de manipulação de objetos.

O gerente de configurações torna possível a construção de configurações e seu funcionamento diz respeito à descrição dos processos de manipulação de objetos-configuração, determinados pelas operações de criação, alteração, consulta, remoção, cópia e escolha de objetos para configurações. O modo de interação deve ser abordado sob dois aspectos: em relação aos objetos de projeto, por serem eles os elementos concretos das configurações, e em relação ao usuário que vai manipulá-los. Através da descrição da interação, serão definidos os elementos envolvidos, as informações que devem ser fornecidas e as que serão retornadas.

Nesta seção, será abordada mais especificamente a interação com os objetos de projeto. A interface com o usuário será realizada através de uma interface gráfica, cujo desenvolvimento está em fase de estudos [MEL 9?]. Esta interface tem como recurso básico um "browser" para manipulação dos objetos de projeto, tornando esta tarefa mais amigável e natural, além de aproveitar os recursos do ambiente onde o STAR está sendo desenvolvido.

5.6.1 Descrição Funcional do Processo de Configuração

O processo de configuração (figura 5.4) é determinado pelas operações definidas para as configurações (figura 5.3), que utilizam-se dos objetos-referência especificados nos atributos de referência e determinam as descrições que são estabelecidas em função de escolhas entre eles. No decorrer do processo, para cada hierarquia de definição que compõe o objeto, há sempre um *ViewState* corrente de configuração, que é o *ViewState* sobre o qual está se executando uma determinada operação de configuração. Num momento inicial e posteriormente, sempre que for necessário mudá-lo, o usuário deve selecionar um *ViewState*, que passa a ser o corrente para as configurações.

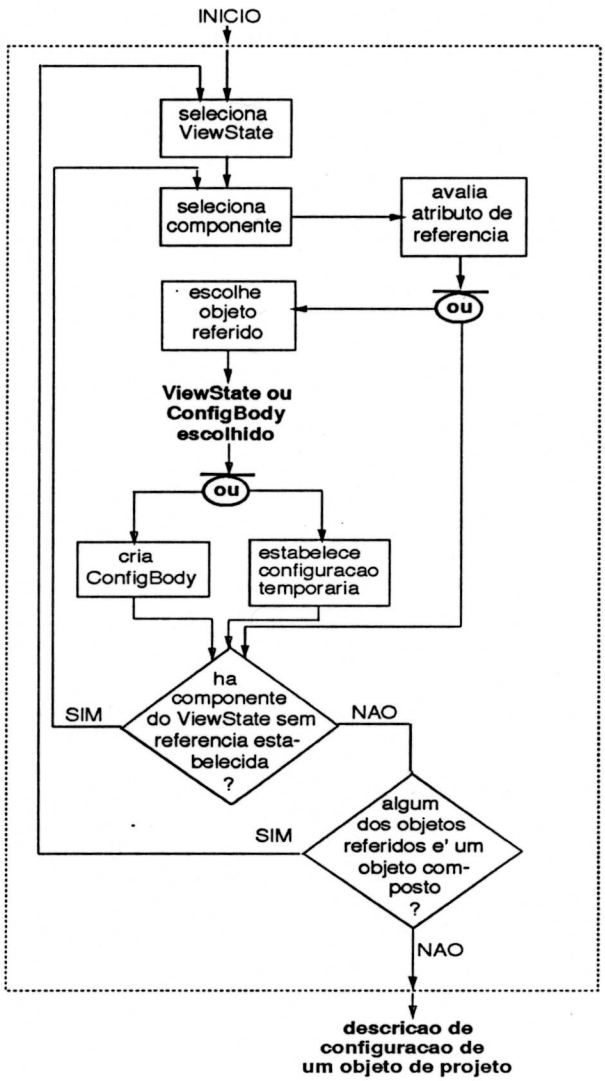

Figura 5.4: O processo de configuração

Este processo é determinado por uma sucessão de operações de criação de configurações. O seu funcionamento é descrito a seguir.

O passo inicial é a definição pelo usuário de um *ViewState* do *Design* mais externo, o objeto de projeto. Para o *ViewState* escolhido, são verificados os atributos de referência de cada um de seus componentes. Se eles determinarem referências a objetos, estes são apresentados ao usuário, que no caso da configuração estar incompleta interage com o gerente de configurações, fazendo as escolhas de objetos-referência necessárias. A escolha de um *ConfigurationBody* como objeto-referência faz com que sejam mostradas ao usuário as descrições ou comple-

mentações de descrição de referência dos componentes que ele possui. Se algum componente do *ViewState* ao qual ele está vinculado ficar sem definição completa ou não tiver definição, o usuário poderá estabelecê-la, e optar pela criação de um objeto-configuração vinculado ao *ViewState*. Só será possível selecionar o próximo *ViewState* quando o *ViewState* atual tiver sua configuração completa, para todos os seus componentes. O próximo *ViewState*, na seqüência da hierarquia de composição, para o qual se selecionará a configuração é escolhido pelo usuário. Assim, o percurso na árvore definida pela hierarquia de composição do objeto de projeto para a seleção de configurações para os *ViewStates* pode ser feito em largura ou em profundidade, com controle do usuário, que será responsável pela garantia da existência de configurações para todos os sub-objetos da composição.

Devido à possibilidade de existir mais de uma descrição de referência num mesmo objeto-configuração, pode ocorrer que mais de um componente de um mesmo *ViewState* faça referência a um mesmo *ConfigurationBody*.

Na escolha de objetos-configuração como referência para componentes de um *ViewState*, é possível que mais de um *ConfigurationBody* faça parte de uma mesma configuração local, desde que eles sejam complementares com relação à definição dos objetos referidos para os componentes. Isso porque não pode haver, numa configuração local que faça parte da configuração completa de um objeto de projeto, mais de uma referência estabelecida para um mesmo componente.

Para os componentes que possuem apenas uma descrição de configuração, completa, pelo atributo de referência ou *ConfigurationBody*, ela será automaticamente apresentada ao usuário como a escolhida. Nos casos em que a descrição de configuração escolhida apresentar a expressão **"current-version"** no lugar de um *ViewState*, automaticamente será apresentada ao usuário a descrição com o *ViewState* corrente. Se este *ViewState* não complementar a configuração, o usuário poderá fazer uso das operações específicas para tal.

O resultado do processo de configuração é uma descrição das ligações estabelecidas ao longo da hierarquia de composição do objeto, numa seqüência **ViewState composto - seus sub-objetos com respectivos ViewStates referidos,** para cada *ViewState* referido que for composto.

Essa descrição das ligações deverá ser convertida para um formato que possa ser entendido pela ferramenta que utilizará a descrição do objeto.

A seguir é descrito o funcionamento das operações no processo de configuração.

5.6.1.1 Criação de Configurações

A criação de uma configuração é realizada a partir da operação de criação de configurações (figura 5.5). Seu **requisito básico** é a definição de um objeto no nível de *ViewStates*, composto, para o qual serão estabelecidos os objetos referidos por cada um de seus componentes.

Para cada componente, será analisado o seu atributo de referência. Se houver alguma descrição de objeto-referência, ela será informada ao usuário e, se for o caso, a partir dela ele deve fazer as escolhas de objetos e versões, necessárias para complementá-la no mínimo até o nível de *ViewStates*. Estas escolhas são realizadas através da operação de escolha de objetos para configuração. Caso o sub-objeto seja um *DesignInstance* vinculado a um *Component* cujo atributo de referência nomeie algum objeto, o objeto-referência escolhido para complementar a descrição será estendido aos demais *DesignInstances* vinculados a esse *Component* automaticamente. Não havendo atributo de referência com descrição de objeto-referência, é disparada a operação de escolha de objetos, sem a restrição de complementar uma descrição já iniciada, e sim executando a escolha a partir do nível inicial de *Design* do objeto a ser referido.

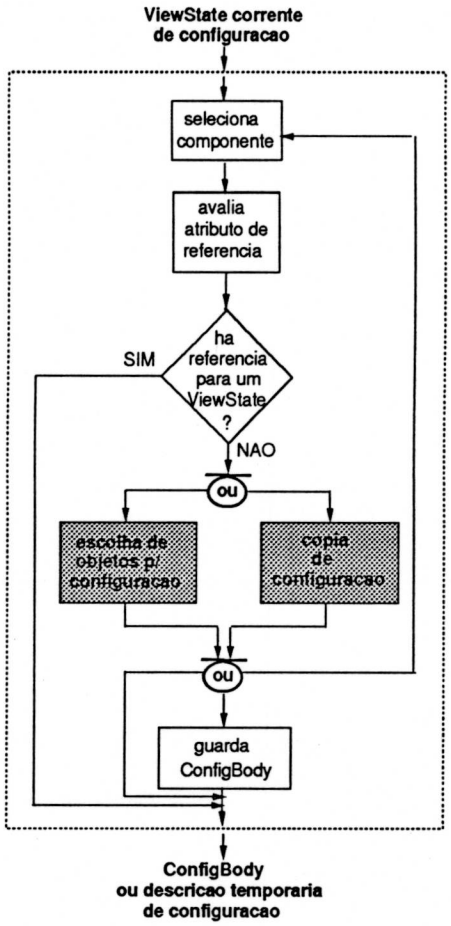

Figura 5.5: Operação de criação de configurações

Ao invés de realizar o processo de escolha para determinar uma configuração, o usuário tem a opção de utilizar uma descrição já existente de um *ConfigurationBody* através da operação de cópia de configuração. Neste caso também é preciso levar em consideração atributos de referência instanciados, caso existam.

O estabelecimento de um *ConfigurationBody* como objeto-referência torna a configuração do objeto de projeto mais direcionada, dando menos flexibilidade no momento da seleção da configuração, uma vez que pré-fixa a referência do componente para o nível seguinte da hierarquia de composição, o que pode ser conveniente ou não. Essa menor flexibilidade é adequada quando o usuário já tem certas opções de projeto bem definidas, pois há algumas decisões com poucas probabilidades de

mudança, o que agiliza o processo. Neste caso, é interessante ter-se uma descrição de configuração que determina outra descrição de configuração.

Após estar completamente definida a referência de um componente, o usuário tem a opção de guardá-la, através de um *ConfigurationBody*. Num *ConfigurationBody* é possível guardar uma ou mais descrições de referência, desde que para componentes diferentes de um mesmo *ViewState*. Para os *DesignInstances* vinculados a um *Component*, basta apenas uma descrição para um deles, ou para o *Component*, a qual será estendida para os demais.

A operação de criação poderá ser realizada repetidamente, caso os objetos referidos pelos componentes sejam também compostos. Seu **escopo** é um *ViewState* composto, seja ele *MHDViewState, HDLViewState ou LayoutViewState*, ao qual serão vinculados os *ConfigurationBodies* que vierem a ser estabelecidos cada vez que ela é disparada. Esta execução repetida da criação permite que se estabeleça e armazene toda a configuração do objeto de projeto, definida ao longo de sua hierarquia de composição.

O **resultado** dessa operação, cada vez que é realizada, é um *ConfigurationBody* que pode ter sido salvo ou não. Caso não seja, ele dá origem a uma descrição temporária de configuração.

5.6.1.2 Alteração de Configurações

Através dessa operação (figura 5.6) é possível alterar descrições de objetos referidos presentes nos *ConfigurationBodies* que estejam em trabalho.

Seu **requisito básico** é a definição do *ConfigurationBody* cujo conteúdo se quer alterar. Se ele não estiver em trabalho, o usuário não poderá alterar sua descrição. Caso ele queira realmente alterá-la, deverá criar um novo *ConfigurationBody*, disparando a operação de criação de configurações.

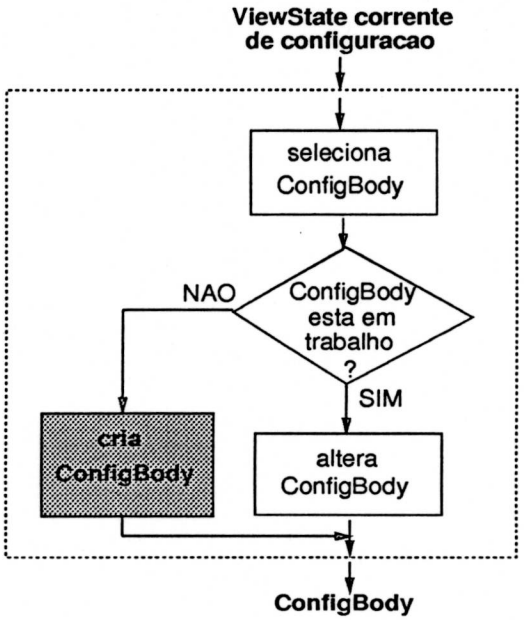

Figura 5.6: Operação de alteração de configurações

O escopo dessa operação é um *ViewState* com *ConfigurationBodies* em trabalho vinculados, e o seu **resultado** é um *ConfigurationBody.*

5.6.1.3 Consulta a Configurações

O requisito básico para consultar uma configuração é a definição de um *ViewState* composto. Se ele tiver *ConfigurationBodies* vinculados, a partir da seleção de um deles é possível ver as referências que ele estabelece para os componentes do *ViewState.* Se não tiver, é possível ver as referências estabelecidas para os componentes através de seus atributos de referência.

A consulta a configurações (figura 5.7) permite que o usuário altere as versões correntes dos objetos somente para fins de informação, e não de forma definitiva. Para fins de versionamento, elas permanecem inalteradas, pois a alteração das versões correntes só pode ser realizada através das operações de seleção parcial e total ligadas ao gerente de versões.

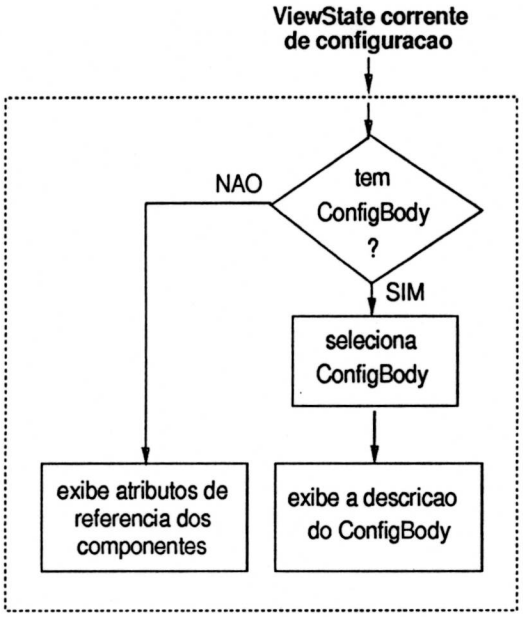

Figura 5.7: Operação de consulta a configurações

O escopo dessa operação é um *ViewState* com componentes, e o **resultado** é a apresentação dos objetos referidos diretamente por cada componente ou através de um *ConfigurationBody*.

5.6.1.4 Remoção de Configurações

O processo de remoção de uma configuração (figura 5.8) tem como **requisito básico** a determinação de um objeto no nível de *ViewStates*, ao qual pertençe o *ConfigurationBody* que se quer remover.

O usuário fará a escolha do *ConfigurationBody*, e ele será removido. Caso ele seja utilizado como referência por outro *ConfigurationBody*, a remoção não poderá ser efetivada, pois causaria uma inconsistência. **O escopo** deste processo é um *ViewState* composto com *ConfigurationBodies* vinculados, e não é esperado nenhum **resultado**.

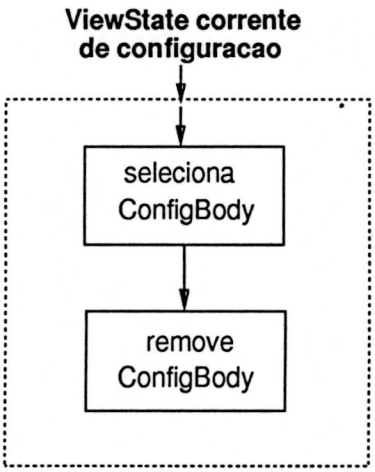

Figura 5.8: Operação de remoção de configurações

5.6.1.5 Escolha de Objetos para Configuração

A escolha de objetos para configuração, conforme pode ser observado na figura 5.9, pode ser realizada de quatro modos diferentes:

- seleção através da definição de uma expressão de configuração,
- seleção com base na versão corrente,
- seleção com base na versão mais recente,
- seleção manual.

O funcionamento de cada um dos modos já é conhecido em função da abordagem dos critérios para escolha de objetos.

O **requisito básico** para a escolha de objetos para uma configuração é a definição do *ViewState* composto para cujos componentes serão selecionados *ViewStates* de referência.

A seguir, utilizando-se dos modos disponíveis, escolhe-se para cada componente o *ViewState* referido. Se o *ViewState* tiver *ConfigurationBodies* vinculados,

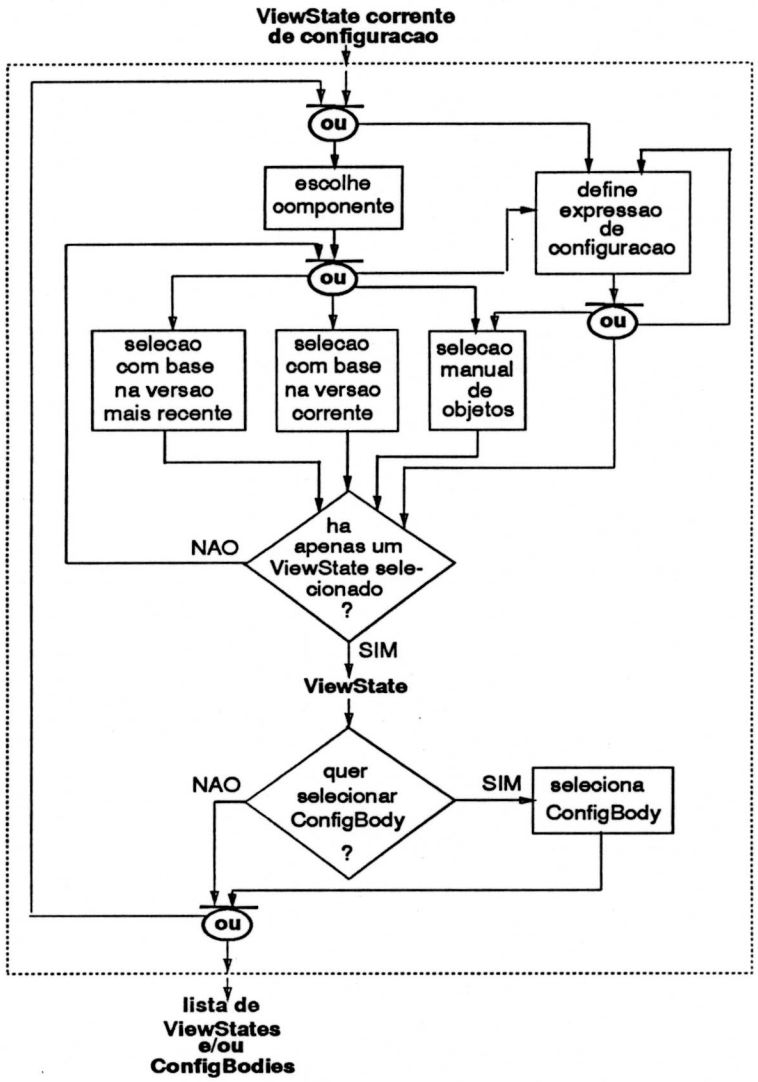

Figura 5.9: Operação de escolha de objetos para configuração

o usuário pode optar por um deles como elemento de configuração de um determinado sub-objeto.

A seleção através de uma expressão de configuração é o único modo que não exige que se escolha antecipadamente um determinado componente do *ViewState*, a cada vez que se for aplicar um dos modos de escolha. Isso porque a sua estrutura é composta de expressões de seleção específicas para cada componente ao qual se queira definir o objeto-referência. Isso permite determinar mais de uma referência explícita de uma só vez. O que pode ocorrer em função dessa seleção em conjunto é que para mais de um componente sejam selecionados vários *ViewStates*.

Neste caso, escolherá um novo modo de seleção que permita selecionar apenas um *ViewState* para cada componente que teve um conjunto de *ViewStates* selecionado. Poderá ser escolhido um critério comum a todos os conjuntos ou um critério para cada conjunto.

O **resultado** da operação de escolha de objetos para configuração é uma lista, cujos elementos são *ViewStates* e/ou *ConfigurationBodies*, cada um deles vinculado ao componente que o referencia.

5.6.1.6 Cópia de Configuração

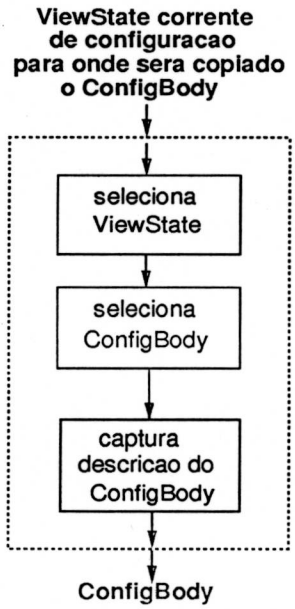

Figura 5.10: Operação de cópia de configuração

Essa operação tem como **requisito básico** a definição do *ViewState* do qual será extraída a descrição do *ConfigurationBody* a ser copiada, além do *ViewState* para onde ela será copiada. Seu **escopo** é um *ViewState* com *ConfigurationBodies* vinculados.

Conforme pode ser observado na figura 5.10, a partir do *ViewState* de onde será copiada a configuração, é selecionado um *ConfigurationBody*, que é capturado para que sua descrição seja utilizada em um outro objeto configuração.

Só é possível copiar a descrição inteira, não se pode particioná-la para copiar apenas algumas descrições de sub-objetos. Alterações podem ser feitas após a cópia, no *ConfigurationBody* que a importou, uma vez que o estado inicial de um objeto é em trabalho, sendo ele sujeito a modificações.

O **resultado** dessa operação é um *ConfigurationBody*.

6 CONCLUSÕES

A partir de alguns elementos relacionados a configurações considerados importantes e do estudo do modelo de dados e do gerente de versões do STAR, foi definido para esse ambiente um gerente de configurações. Ele possui características que dão flexibilidade para o estabelecimento e utilização de configurações no ambiente.

Uma abordagem inicial sobre configurações permitiu selecionar itens que foram posteriormente integrados ao gerente de configurações e que são descritos a seguir.

O ambiente permite estabelecer configurações estáticas, dinâmicas e abertas. O suporte a configurações estáticas e dinâmicas é encontrado na maioria dos sistemas ([CHA 89], [SKM 91], [SIE 89], [WAG 88], [BEL 87]), enquanto que apenas VHDL [LIP 90] e SDE [SKM 91], que tem por base o VHDL, suportam configurações abertas. A estrutura aqui estabelecida para as configurações dinâmicas e abertas não se baseia apenas num objeto ou numa descrição. São utilizados de forma integrada e complementar um atributo que estabelece referências para os componentes e um objeto-configuração. Essa integração permite que configurações continuem dinâmicas ou abertas mesmo após terem sido feitas seleções de objetos para complementá-las até os objetos-folha. Além do mais, as configurações não são perdidas.

Os mecanismos definidos para selecionar objetos abrangem as três alternativas apresentadas em [KEM 91], que procurou generalizar os métodos utilizados para seleção de componentes:

1. seleção de componentes a partir de propriedades que são desejadas para eles;

2. componentes estabelecidos através de um default;

3. a seleção de componentes é guiada somente por informações não presentes na definição do objeto.

A seleção a partir de expressões de configuração se enquadra no primeiro item. Elas permitem selecionar componentes a partir de características dos objetos, que são elementos mais intuitivos para o usuário e cuja combinação permite o estabelecimento dos mais variados tipos de restrições. ADÈLE e SDE possuem mecanismos com base numa linguagem de restrições.

O segundo item é coberto pelo modo de seleção automática com base na versão corrente ou na mais recente. Este é um modo de seleção bastante comum, encontrado em todos os mecanismos de configurações estudados, e é o recurso mínimo que se exige para o estabelecimento de configurações. No terceiro item pode-se enquadrar a seleção manual, onde os critérios para seleção de componentes são estabelecidos a partir do próprio usuário que faz a escolha.

Como pode ser observado, o gerente de configurações do STAR é bastante flexível e abrangente quanto aos mecanismos para seleção de objetos.

As configurações podem ser armazenadas e desta maneira reutilizadas, como ocorre no Version Server [CHA 89] e no SDE [SKM 91].

Não foi estabelecido um mecanismo para propagação de mudanças, como em [CHA 89] e [BAT 85]. Tem-se apenas as notificações aos usuários proprietários de objetos que utilizam objetos para os quais foram geradas novas versões.

O fato de se poder definir objetos-configuração, que inclusive fazem parte do modelo de dados, é um recurso importante e bastante útil para o projeto de sistemas digitais. Há características estabelecidas em função de determinadas configurações e que podem ser mantidas através de atributos dos objetos-configuração.

Está em fase de estudos o desenvolvimento de uma interface gráfica com um browser para manipulação de objetos [MEL 9?] e a ela deverão ser integrados

o gerente de versões e o de configurações. A partir da visualização dos objetos e de sua estrutura, a tarefa de estabelecer configurações será bem mais amigável e os recursos já estabelecidos para o gerente de configurações ficarão mais poderosos, pois a visualização gráfica dos elementos facilita a execução das atividades.

ANEXO A-1 GRAMÁTICA DA LINGUAGEM PARA SELEÇÃO DE OBJETOS

```
<Conf_expression> :: CONFIGURATE <ObjViewState> <Description>

<Description> :: <ConfSubObject> ; <Description> |
                 <ConfSubObject>

<ConfSubObject> :: <SubObject> : <Constraints>

<SubObject> :: DesignInstance |
               Component

<Constraints> :: ( <Constraints> ) |
                  <Expression> <Link> <Constraints> |
                  <Expression>

<Expression> :: ( <Expression> ) |
                <Unary1> <Attribute> |
                <Unary2> <Node> |
                <Attribute> <Binary> <Value>

<Atribute> :: Userfields |
              Ports

<Object> :: <Node> |
            <ObjViewState>
```

```
<Node> :: Design |
          ViewGroup |
          View

<ObjViewState> :: MHDViewState |
                  HDLViewState |
                  LayoutViewState

<Unary1> :: EXISTS |
            NOT EXIST |
            MAX |
            MIN

<Unary2> :: FIRST |
            LAST |
            OBJECT |
            CURRENT

<Binary> :: EQUAL |
            NEQUAL |
            GR |
            LE |
            GREQUAL |
            LEEQUAL

<Link> :: AND | OR | NOT

<Value> :: Constant
```

BIBLIOGRAFIA

[AHM 91] AHMED, R.; NAVATHE, S. Version management of composite objects. **ACM SIGMOD Record**, New York, v.20, n.2, p.218-227, June 1991. Ed. Special: ACM SIGMOD International Conference on Management of Data, May 29-31, 1991, Denver, US.

[BAN 91] BANKS, S. et al. A configuration management system in a data management framework. In: DESIGN AUTOMATION CONFERENCE, 28, June 17-21, 1991, San Francisco, CA. **Proceedings...** Baltimore: ACM, Los Alamitos: IEEE, 1991. 783p. p.699-703.

[BAT 85] BATORY, D.S.; KIM, W. Modeling concepts for VLSI CAD objects. **ACM Transactions on Database Systems**, New York, v.10, n.3, p.322-346, Sept. 1985.

[BEL 85] BELKHATIR,N.; CHEVAL, J.L. Un modele de gestion de gros logiciels en version multiples: ADL2. In: JOURNÉES AFCET-INFORMATIQUE - NOUVEAUX LANGUAGES POUR LE GÉNIE LOGICIEL, Oct. 28-29, 1985, Evry, FR. **Actes...** Grenoble: Bigre+Globule, 1985. p.124-135.

[BEL 87] BELKHATIR,N.; ESTUBLIER, J. Experience with a data base of programs. ACM Sigplan Notices, New York, v.22, n.1, p.84-91, Jan. 1987.

[CHA 89] CHANG, E.E.; GEDYE, D.; KATZ, R.H. The Design and implementation of a Version Server for Computer-Aided Design Data. **Software-Practice and Experience**, Sussex, v.19, n.3, p.199-222. Mar. 1989.

[DIA 91] DIAS, E.Z.V.; MAGALHÃES, G.C. MVC: um modelo para controle de versões e configurações. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 5, 23-25 out. 1991, Ouro Preto, MG. **Anais...** Belo Horizonte: Imprensa Universitária, 1991. 256p. p.93-106.

[GRA 91] GRAZZIOTIN, H. **Estudo sobre gerência de configurações e uma proposta para o sistema AMPLO.** Porto Alegre: CPGCC da UFRGS, 1991. 59p. (Trabalho Individual, 235)

[KAT 86] KATZ, R.H.; CHANG, E.; BHATEJA, R. Version modeling concepts for Computer-Aided Design databases. **ACM SIGMOD Record**, New York, v.15, n.2, p.379-386, June 1986. Ed. Special: ACM SIGMOD International Conference on Management of Data, May 28-30, 1986, Washington, DC.

[KAT 87] KATZ, R.H. et al. Design version management. **IEEE Design & Test of Computers**, New York, v.4, n.1, p.12-22, Feb. 1987.

[KAT 90] KATZ, R.H. Toward a unified framework for version modeling in engineering databases. **ACM Computing Surveys**, New York, v.22, n.4, p.375-408, Dec. 1990.

[KEM 91] KEMPER, F.; WILKES, W.; SCHLAGETER, G. Basic mechanism to support versioning in the database component of a CAD framework. In: IFIP WG 10.2 WORKSHOP ON ELECTRONIC DESIGN AUTOMATION FRAMEWORKS, 2, Nov. 26-28, 1990, Charlotesville, VA. **Proceedings...** Amsterdam: North-Holland, 1991. 397p. p.171-178.

[KIM 89] KIM, W.; BERTINO, E.; GARZA, J.F. Composite objects revisted. **ACM SIGMOD Record**, New York, v.18 , n.2 , p.337-347, June 1989. Ed. Special: ACM SIGMOD International Conference on Management of Data, May 31-June 2, 1989, Portland, OR.

[KIM 90] KIM, W. **Introduction to object-oriented databases.** Cambridge: MIT Press, 1990. 234p. p.147-148.

[LAC 92] LACOMBE, J.C.M. **Modelo de versões para o ambiente Star.** Porto Alegre: CPGCC da UFRGS, 1992. 133p. (Dissertação de Mestrado)

[LIP 90] LIPSETT, R.; SCHAEFER, C.F.; USSERY, C. **VHDL :** Hardware description and design. Boston: Kluwer, 1990. 299p.

[LIV 92] LIVI, M.A.C.; IOCHPE, C. Gerenciando projeto e cooperação através de passos de projeto e características de objetos. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 7, 13-15 maio 1992, Porto Alegre, RS. **Anais...** Porto Alegre: Impa Artes Gráficas, 1992. 500p. p.301-315.

[MAH 90] MAHLER, A.; LAMPEN, A. Integrating configuration management into a generic environment. **Software Engineering Notes,** New York, v.15, n.16, p.229-237, Dec. 1990.

[MEL 9?] MELLO, R. **Linguagem visual de consulta para o ambiente STAR.** Porto Alegre: CPGCC da UFRGS, 199?. (Dissertação em andamento)

[PLO 89] PLOEDEREDER, E.; FERGANI, A. The data model of configuration management assistant. **Software Engineering Notes,** New York, v.14, n.17, p.5-14, Nov. 1989. Ed. Special: International Workshop on Software Configuration Management, 2, Oct. 24, 1989, Princeton, NJ.

[SIE 89] SIEPMANN, E.; ZIMMERMANN, G. An object-oriented datamodel for the VLSI design system PLAYOUT. In: DESIGN AUTOMATION CONFERENCE, 26, June 25-29, 1989, Las Vegas, US. **Proceedings...** Baltimore: ACM, Los Alamitos: IEEE, 1989. 839 p. p.284-289.

[SIE 89a] SIEPMANN, E. A data management interface as part of the framework of an integrated VLSI design system. In: INTERNATIONAL CONFERENCE ON COMPUTER DESIGN, Oct. 2-4,1989, Cambridge, UK. **Proceedings...** Los Alamitos: IEEE, 1989. 587p. p.284-287.

[SKM 91] KIM, S.; CHUNG, M.J. A constraint-driven approach to configuration binding in an object-oriented VHDL CAD system. In: IFIP WG 10.2 INTERNATIONAL SYMPOSIUM ON COMPUTER HARDWARE DESCRIPTION LANGUAGES AND THEIR APPLICATIONS, 10, Apr. 22-24, 1991, Marseille, FR. **Proceedings...** Amsterdam: North-Holland, 1991. 445p. p.359-374.

[SMI 77] SMITH, J.M.; SMITH D.P.C. Database abstractions: aggregation and generalization. **ACM Transactions on Database Systems**, New York, v.2, n.2, p.105-33, June 1977.

[VIC 89] VICTORELLI, E.Z.; MAGALHÃES, G.C.; DRUMMOND, R. Mecanismo de gerenciamento de versões e configurações do A_HAND. **Revista Brasileira de Computação**, Rio de Janeiro, v.5, n.2, p.3-9, out.-dez. 1989.

[WAG 88] WAGNER, F.R.; FREITAS, C.M.D.-S; GOLENDZINER, L.G. The AMPLO system: an integrated environment for digital systems design. In: IFIP WG 10.2 WORKSHOP ON TOOL INTEGRATION AND DESIGN ENVIRONMENTS, Nov. 26-27, 1987, Paderborn, FRG. **Proceedings...** Amsterdam: North-Holland, 1988. 323p. p.221-232.

[WAG 91] WAGNER, F.R. **Modelos de representação e gerência de dados em ambientes de projeto de sistemas digitais.** Rio de Janeiro: IBM Brasil, 1991. 47p. (Relatório Técnico do Centro Científico Rio, 121)

[WAG 91a] WAGNER, F.R.; LIMA, A.H.V. Design version management in the GARDEN framework. In: DESIGN AUTOMATION CONFERENCE, 28, June 17-21, 1991, San Francisco, CA. **Proceedings...** Baltimore: ACM, Los Alamitos: IEEE, 1991. 783p. p.238-245.

[WAG 91b] WAGNER, F.R. **The Data model of the STAR framework.** Porto Alegre: CPGCC da UFRGS, 1991. 27p. (Relatório de Pesquisa, 167)

[WAG 91c] WAGNER, F.R. **Design methodology management in the STAR framework.** Porto Alegre: CPGCC da UFRGS, 1991. 17p. (Relatório de Pesquisa, 170)

[WAG 92] WAGNER, F.R.: GOLENDZINER, L.G.; LACOMBE, J.: LIMA, A.V. Design version management in the STAR framework. In: IFIP WG 10.2/WG 10.5 WORKSHOP ON ELECTRONIC DESIGN AUTOMATION FRAMEWORKS, 3, Mar. 23-25, 1992, Bad Lippspringe, GT. **Proceedings...** Amsterdam: North-Holland, 1992. 288p. p.85-97.

[WAP 91] WAGNER, P.R. **Um Ambiente integrado para simulação de sistemas digitais.** Porto Alegre: CPGCC da UFRGS, 1991. 121p. (Dissertação de Mestrado)

Informática
UFRGS

"Gerente de Configurações para o Ambiente STAR".

Dissertação apresentada aos Srs.:

Profa. Dra. Claudia M. Bauzer Medeiros (UNICAMP)

Prof. Dr. Cirano Iochpe

Prof. Dr. Clesio Saraiva dos Santos

Profa. Lia Goldstein Golendziner

Vista e permitida a impressão.
Porto Alegre, 25 / 08 / 93.

Profa. Lia Goldstein Golendziner,
Orientador.

Prof. Dr. Ricardo A. da L. Reis,
Coordenador do Curso de Pós-Graduação
em Ciência da Computação.