Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Computação
Programa de Pós-Graduação em Ciência da Computação

Reuso de Componentes para Interfaces com Realidade Virtual Apoiado pelo Ambiente GaCIV

José Carlos Lazzari Albertin

São Carlos – SP
Agosto/2003

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

A334rc

Albertin, José Carlos Lazzari.
Reuso de componentes para interfaces com realidade virtual apoiado pelo ambiente GaCIV / José Carlos Lazzari Albertin. -- São Carlos : UFSCar, 2006.
69 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2003.

1. Interação homem-máquina. 2. Realidade virtual. 3. Componentes de software. 4. Reuso. I. Título.

CDD: 004.019 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

REUSO DE COMPONENTES PARA INTERFACES COM REALIDADE VIRTUAL APOIADO PELO AMBIENTE GACIV

JOSÉ CARLOS LAZZARI ALBERTIN

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Profa. Dra. Júnia Coutinho Anacleto Silva
(Orientadora – DC/UFSCar)

Profa. Dra. Rosângela Ap. Dellosso Penteado
(DC/UFSCar)

Profa. Dra. Maria Cecília Calani Baranauskas
(IC/UNICAMP)

São Carlos
Agosto/2003

À minha namorada Márcia pelo
apoio e incentivo nesta nova
conquista.

Agradecimentos

À Deus pela capacitação e bênçãos derramadas em minha vida.

À minha família pelo incentivo e apoio.

À Professora Dra. Júnia Coutinho Anacleto Silva e a co-orientadora Professora Dra. Rosângela Aparecida Dellosso Penteado pela orientação, incondicional apoio e contribuição para meu amadurecimento profissional.

E a todos os professores do Departamento de Computação da Universidade Federal de São Carlos, colegas e demais pessoas que direta ou indiretamente me ajudaram nesta conquista.

Resumo

A Realidade Virtual (RV) permite que a interação humano-computador possa ocorrer de maneira mais natural e intuitiva, já que o usuário pode transmitir o conhecimento que ele possui do mundo real para o mundo virtual. Isso beneficia a usabilidade do sistema pois o usuário, ao sentir-se dentro de um ambiente que se pareça com o mundo real, pode se situar melhor perante as atividades que ele quer realizar, através da navegação no mundo virtual e a interação mais natural com objetos virtuais.

Este trabalho apresenta o Ambiente Computacional GaCIV (Gabaritos Configuráveis para elaboração de Interfaces com realidade Virtual) que propõe a elaboração de interfaces com RV de maneira simples e direta, dando suporte ao reuso de componentes para interfaces com RV, gerando repositórios de componentes para interfaces com RV.

Abstract

This work presents the GaCIV Computational Environment (Configurable Templates for Development of Virtual Reality Interfaces) for building interfaces with non-immersive Virtual Reality for applications in different domains. This new version supports software component reuse, allowing project of friendlier interfaces with virtual reality, with usability and praticity.

Sumário

Sumário i
Índice de Figuras iii
Índice de Tabelas iv
Índice de Abreviaturas v

1. INTRODUÇÃO 1

1.1. OBJETIVOS 1
1.2. MOTIVAÇÃO 2
1.3. ORGANIZAÇÃO DO TRABALHO 3

2. A REALIDADE VIRTUAL COMO TÉCNICA DE DESENVOLVIMENTO DE INTERFACES 4

2.1. CONSIDERAÇÕES INICIAIS 4
2.2. INTERFACES COM REALIDADE VIRTUAL E SUA APLICABILIDADE 6
2.3. PROBLEMAS ENVOLVENDO O DESENVOLVIMENTO DE INTERFACES COM REALIDADE VIRTUAL 10
2.4. CONSIDERAÇÕES FINAIS 14

3. REUSO DE COMPONENTES PARA INTERFACES COM RV 15

3.1. CONSIDERAÇÕES INICIAIS 15
3.2. FATORES RELEVANTES NO DESENVOLVIMENTO DE SOFTWARE BASEADO EM COMPONENTES. 16
3.3. ENGENHARIA DE SOFTWARE BASEADA EM COMPONENTES 17
3.3.1. Engenharia de Domínio 19
3.3.2. Desenvolvimento Baseado em Componentes 20
3.4. DESENVOLVIMENTO DE INTERFACES BASEADO EM REUSO DE COMPONENTES 21
3.5. CONSIDERAÇÕES FINAIS 24

4. O AMBIENTE GACIV PARA APOIO AO PROJETO DE INTERFACES COM REALIDADE VIRTUAL BASEADO EM REUSO DE COMPONENTES 26

4.1. CONSIDERAÇÕES INICIAIS 26
4.2. O AMBIENTE GACIV 26
4.3. O GACIV NO APOIO A ESBC, ED E DBC 32
4.4. ESTRUTURAÇÃO DO AMBIENTE GACIV 34
4.5. FUNCIONALIDADES DO AMBIENTE GACIV 36
4.6. DESENVOLVIMENTO DE UM NOVO COMPORTAMENTO PARA COMPONENTES 3D NO GACIV 46
4.6.1.Desenvolvimento do Comportamento 46
4.7. CONSIDERAÇÕES FINAIS 48

5. ANÁLISE DOS RESULTADOS OBTIDOS 49

5.1. CONSIDERAÇÕES INICIAIS 49
5.2. PROPOSTA DE SOLUÇÃO PARA OS PROBLEMAS TÍPICOS NOS PROJETOS DE SISTEMAS COM RV 50
5.2.1. Proposta de solução para o problema "Projetar comportamentos não semelhantes a objetos do mundo real" 50
5.2.2. Proposta de Solução para o problema "Embutir imagem 3D no código da aplicação" 51
5.2.3. Proposta de Solução para o problema "Utilizar ferramentas de prototipação não-extensíveis" 52
5.2.4. Proposta de Solução para o problema "Não definir ou não utilizar arquiteturas 3D para construção de aplicações com RV" 52
5.3. COMPONENTES 3D, SUA DOCUMENTAÇÃO E SUAS INTERFACES NO GACIV 54

5.4. GABARITOS X REPOSITÓRIOS DE COMPONENTES 3D E MECANISMOS DE BUSCA EM REPOSITÓRIOS 56
5.5. CONSIDERAÇÕES FINAIS 59
6. CONCLUSÕES 60
6.1. CONSIDERAÇÕES INICIAIS 60
6.2. PRINCIPAIS CONTRIBUIÇÕES 60
6.3. DIFICULDADES A SEREM SUPERADAS 61
6.3.1. Dificuldades Relacionadas à Implementação 61
6.3.2. Dificuldades Relacionadas à Usabilidade 62
6.4. TRABALHOS FUTUROS 63
7. BIBLIOGRAFIA 65

Índice de Figuras

FIGURA 2.1 – AS TRÊS CARACTERÍSTICAS DA RV (BURDEA & COIFFET, 1994).....4
FIGURA 2.2 – CARACTERÍSTICAS DE AMBIENTES VIRTUAIS EM RELAÇÃO A OUTROS TIPOS DE INTERFACES (KAUR, 1998).....6
FIGURA 3.1 – ENGENHARIA DE SOFTWARE BASEADA EM COMPONENTES (BRAGA, 2000) 18
FIGURA 4.1 – UMA TELA DA PRIMEIRA VERSÃO GACIV.....27
FIGURA 4.2 – OPÇÕES DA BARRA DE FERRAMENTAS DO INTERBUILDER 29
FIGURA 4.3 – TELA DO ISTORE.....30
FIGURA 4.4 – OPÇÕES DO IBUILDER.....30
FIGURA 4.5 – OPÇÕES DO IPACKAGE.....31
FIGURA 4.6 – MODOS DE ACESSO AO AMBIENTE GACIV 31
FIGURA 4.7 – ESTRUTURA DA TERCEIRA VERSÃO DO GACIV.....36
FIGURA 4.8 – CADASTRANDO UM AMBIENTE.....37
FIGURA 4.9 – INSERINDO UM COMPORTAMENTO. A) DEFININDO A DOCUMENTAÇÃO, B)ESCOLHENDO COMPORTAMENTO E SEU NOME.....38
FIGURA 4.10 – INSERINDO O COMPONENTE ESTANTE. A) ESCOLHENDO A CLASSE PRINCIPAL E ATRIBUINDO UM NOME, B) ESCOLHENDO UM COMPORTAMENTO, C) TELA DE CONFIGURAÇÃO DO COMPORTAMENTO, D) INCLUINDO INFORMAÇÕES DA DOCUMENTAÇÃO. 39
FIGURA 4.11 – INSERINDO UM *LINK* 40
FIGURA 4.12 – ETAPAS DA CONSTRUÇÃO DE UM GABARITO. A) ESCOLHA DE UM AMBIENTE, B) ESCOLHA DE COMPONENTES 3D E C) ATRIBUINDO UM NOME AO GABARITO. 41
FIGURA 4.13 – O GABARITO DESENVOLVIDO PARA O DOMÍNIO DE VÍDEO LOCADORA 42
FIGURA 4.14 – CRIAÇÃO DE UMA INTERFACE DE VÍDEO LOCADORA. A) SELEÇÃO DE GABARITO, B) O AMBIENTE DO GABARITO JÁ ABERTO, C) INCLUINDO COMPONENTES 3D DO GABARITO, D) ATRIBUINDO UM NOME À INTERFACE.....43
FIGURA 4.15 - ATRIBUIÇÃO DE *LINKS* À INTERFACE LOCADORA.....44
FIGURA 4.16 – A INTERFACE DA VÍDEO LOCADORA 44
FIGURA 4.17 - INTERFACE DA LOCADORA DE VÍDEO NO INTERVIEWER.....45
FIGURA 4.18 – PROGRAMA DE CADASTRO DE CLIENTES.....46
FIGURA 5.1 – REALIZANDO UMA BUSCA DE COMPONENTES 3D.....57
FIGURA 5.2 – CONJUNTO DE RESULTADOS DA BUSCA REALIZADA.....58

Índice de Tabelas

TABELA 2.1 – TECNOLOGIAS UTILIZADAS PARA O DESENVOLVIMENTO DE APLICAÇÕES COM RV 12
TABELA 3.1 – TIPOS DE DESENVOLVIMENTO DE INTERFACES COM RV (ADAPTADA DE DACHSELT (2001) E CONTIGRA (2003)) 23
TABELA 4.1 – ALGUNS ATRIBUTOS E EVENTOS DO COMPONENTE JBUTTON 32

Índice de Abreviaturas

DBC	Desenvolvimento Baseado em Componentes
ED	Engenharia de Domínio
ESBC	Engenharia de Software Baseada em Componentes
IHC	Interação Humano-Computador
RV	Realidade Virtual

1. Introdução

1.1. Objetivos

A evolução dos computadores tem provocado significativas alterações no relacionamento entre humanos e computadores, uma vez que eles têm causado um impacto social, por tornar-se cada vez mais comum seu uso. Dessa forma, é cada vez maior o interesse na forma como as pessoas utilizam os sistemas de softwares. Isso eleva o interesse, tanto na indústria como no meio acadêmico, pela área de Interação Humano-Computador (IHC). O principal objetivo de IHC é o desenvolvimento de sistemas que atendam cada vez melhor às necessidades dos usuários em relação a sua funcionalidade e também em relação à usabilidade, ou seja, a maneira como os usuários devem utilizar um sistema de maneira satisfatória (Nielsen, 1993).

A Realidade Virtual (RV) permite que a interação humano-computador possa ocorrer de maneira mais natural e intuitiva, já que o usuário pode transmitir o conhecimento que ele possui do mundo real para o mundo virtual (Turoff, 1997). Isso beneficia a usabilidade do sistema já que o usuário, ao sentir-se dentro de um ambiente que se pareça com o mundo real, pode se situar melhor perante as atividades que ele quer realizar, através da navegação no mundo virtual e a interação com objetos virtuais, fato realizado através de analogias, que podem ser entendidas como poderosos mecanismos cognitivos usados pelas pessoas para construir novos conhecimentos a partir de outros conhecimentos adquiridos (Barbosa, 1999).

Mas a programação de interfaces com RV não é um processo trivial (Kim et al, 1998). Ela requer muito conhecimento de computação gráfica, já que é realizada por elementos da interface, que são objetos 3D. A programação é realizada através de coordenadas 3D do mundo real que são vistas através de dispositivos de visualização, como monitores de vídeo. Assim, para passar realismo, é preciso programar os diversos fatores existentes em um mundo virtual, tais como transformações geométricas em objetos 3D, animações, colisão, tratamento de dispositivos de entrada e saída, comportamentos de objetos 3D, entre outros.

Nesse sentido, torna-se evidente a necessidade de métodos, técnicas e ferramentas que apóiem e facilitem o desenvolvimento de interfaces com RV.

Também com o objetivo de prover maior usabilidade aos sistemas de software, tem-se a tecnologia de desenvolvimento de software baseado em componentes que, por terem sido previamente usados/testados, agilizam o desenvolvimento e garantem qualidade ao projeto. Neste trabalho, é proposta uma forma de desenvolver sistemas com interfaces baseadas em RV não

imersiva de forma simples e direta, promovendo a integração entre engenheiros de software, projetistas de interfaces e usuário final. Na proposta, é utilizada a tecnologia de componentes, com o objetivo de permitir experimentação e validação de tais tecnologias no apoio ao desenvolvimento de interfaces com RV.

Este trabalho se baseia no Ambiente de Apoio GaCIV (Gabaritos Configuráveis para elaboração de Interfaces com realidade Virtual) (Silva, 1999), projeto desenvolvido pelo DC-UFSCar, que propõem a elaboração de interfaces com RV de maneira simples e prática, através de gabaritos configuráveis, entendidos como modelos semi-prontos que contém um conjunto de objetos 3D, representando um dado domínio de aplicação. Pelo fato de poderem ser modificados, são chamados gabaritos configuráveis. O trabalho consiste na modificação dos gabaritos configuráveis para aceitar um conjunto de componentes para interfaces com RV (e não mais objetos) que são incluídos aos gabaritos conforme um domínio de software, e juntamente com uma ferramenta de busca, passam assim a ser repositórios de componentes. Portanto, o GaCIV passa a ser um gerador de repositório de componentes para construção de interfaces com RV.

1.2. Motivação

A dificuldade em programar interfaces com RV, aliada à necessidade da utilização de ferramentas que apóiem seu desenvolvimento, já que essas se propõem a criar mundos virtuais em diferentes domínios de aplicação, com rapidez e facilidade, são fatores motivantes para seu estudo e desenvolvimento.

Tem-se percebido alguns problemas que prejudicam a capacidade dessas ferramentas (Dachselt, 1999), (Dorner et al, 2001), (Döllner & Hinrichs, 1998). Os dois problemas mais complexos são embutir objetos 3D no código da aplicação e o fato das ferramentas de construção de interfaces com RV não aceitarem o acoplamento de novas funcionalidades, já que é necessária sua recompilação.

A proposta deste trabalho é o estudo de reuso de componentes para apoiar o desenvolvimento de interfaces com RV no intuito de elevar a usabilidade dessas interfaces, a partir do reuso de componentes de software para interfaces desse tipo. Dessa forma, componentes de software, ao serem reutilizados, têm todo seu conjunto de processos utilizados em seu desenvolvimento também reutilizados, tais como, análise de comportamentos, *use cases*, modelagem de interação e partes implementacionais.

Sendo assim, a motivação deste trabalho não inclui somente a dificuldade de programação de interfaces com RV, mas também a capacidade de projetar um ambiente de apoio

flexível, que permite a geração de interfaces com RV, com o suporte a adicionar novas funcionalidades e permitir a separação entre objetos 3D e o código, de maneira que componentes para interfaces com RV possam ser utilizados para resolver esses problemas e ajudar na elaboração dessas interfaces, de forma rápida e prática, maximizando sua capacidade de reuso.

O projeto do GaCIV, como gerador de repositórios, efetivamente, se propõe a fazer o reuso de componentes de software para interfaces com RV armazenados nos repositórios de componentes.

1.3. Organização do Trabalho

No capítulo 2, é apresentada a conceitualização de RV e discutida sua aplicabilidade como interface humano-computador. Também são apresentados os problemas no desenvolvimento de interfaces com RV.

O capítulo 3 apresenta o reuso de componentes para interfaces com RV. Nele são apresentadas as formas como o reuso pode ser realizado e a definição de componentes de software utilizada neste trabalho. São apresentados fatores relevantes para o desenvolvimento de software baseado em reuso de componentes e o processo de engenharia de software baseada em componentes (ESBC), juntamente com a engenharia de domínio (ED) e suas fases e o desenvolvimento baseado em componentes (DBC). Por fim, é discutido como o reuso de componentes é realizado em interfaces com RV.

No capítulo 4 apresenta-se o Ambiente de Apoio GaCIV, levando em consideração os dois capítulos anteriores. É discutido como o GaCIV apóia o desenvolvimento de interfaces com RV através ESBC, ED e DBC e fatores relevantes. Também é apresentada a estruturação do GaCIV e suas funcionalidades.

No capítulo 5 são apresentados os resultados obtidos por este trabalho, bem como a realização dos objetivos apresentados.

No capítulo 6 são apresentadas as conclusões deste trabalho, as principais contribuições para Engenharia de Software e IHC e são discutidas propostas de trabalhos futuros.

2. A Realidade Virtual como Técnica de Desenvolvimento de Interfaces

2.1. Considerações iniciais

Conforme Burdea & Coiffet (1994), Realidade Virtual (RV) pode ser entendida como sendo uma tecnologia avançada de interface, que envolve simulações em tempo real e interações, através de canais multisensoriais pertencentes aos diversos sentidos do homem, tais como: visão, audição, tato, cheiro, gosto, etc.

Essa definição pressupõe que a RV utiliza componentes de hardware e software avançados, no sentido de que as simulações e interações em tempo real sejam possíveis. O usuário entra no ambiente virtual e visualiza, manipula e explora objetos virtuais em tempo real, através dos seus sentidos.

Com base nisso, o advento de novas tecnologias como dispositivos de visualização coloridos tais como monitores e capacetes, bem como outros dispositivos de entrada e saída como luvas, permitiu que se formasse uma base tecnológica para que o desenvolvimento de interfaces com realidade virtual fosse possível.

RV pode ser considerada como uma combinação das três características (Burdea & Coiffet, 1994), como demonstrado na Figura 2.1.

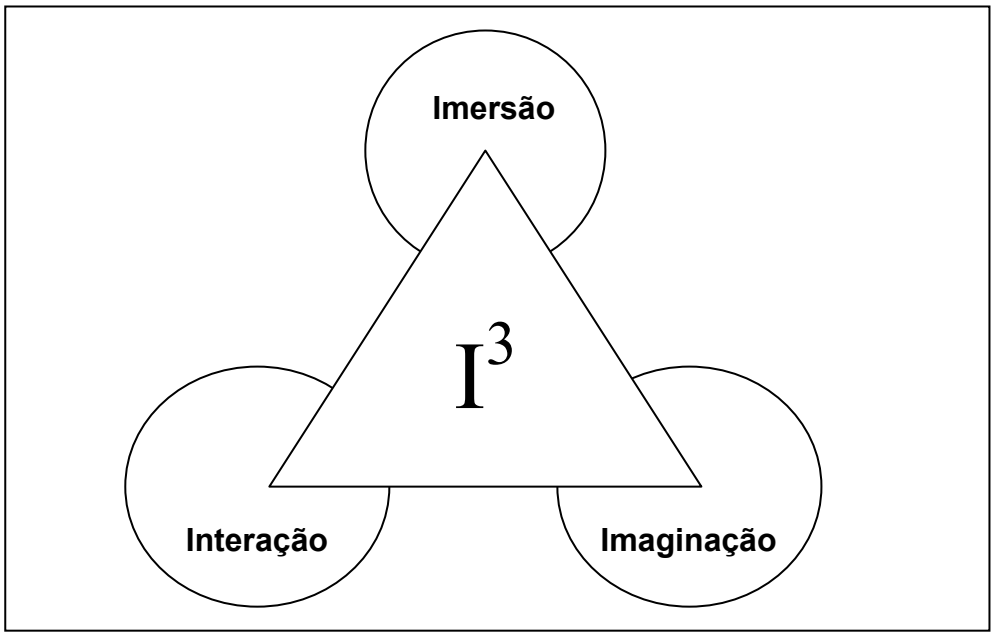

Figura 2.1 – As três características da RV (Burdea & Coiffet, 1994)

Na Figura 2.1, a **interação** é a qualidade do sistema em agir conforme as ações realizadas pelo usuário, modificando o ambiente virtual e as suas ações. A **imersão** está relacionada

à sensação que o usuário tem de estar dentro da interface com a qual interage. O **envolvimento** *(imagination)* está ligado com o grau de motivação para o engajamento de uma pessoa com uma determinada atividade, podendo ser do tipo passivo ou ativo, dependendo da atividade a ser executada.

Em relação à característica de imersão, pode-se perceber quatro tipos de sistemas de RV, que são: os imersivos, os não-imersivos, os semi-imersivos e os híbridos (Vince, 1995) e (Bowman et al, 2001).

Um sistema de RV é imersivo quando é baseado no uso de capacetes ou salas de projeção. Nesse caso, o ambiente virtual reage ao estímulo do usuário através da geração de imagens de computador que vão adaptar o ambiente virtual à ação do usuário. Nesses ambientes, toda a interação do usuário é realizada através dos equipamentos que ele usa, tais como capacete e luvas. Capacetes são mais baratos e requerem menos espaço físico do que as salas de projeções, porém essa última é aconselhável quando o ambiente virtual é freqüentado por vários usuários ao mesmo tempo, dando liberdade de movimento.

Sistemas não-imersivos, também chamados sistemas *desktop*, utilizam recursos tradicionais do computador, tais como monitores de vídeo, teclado e mouse, como forma de navegação em ambientes virtuais. Embora não haja uma imersão de fato, a sensação de imersão pode ser realizada através de transformações perspectivas, dando a idéia de profundidade, sem a necessidade do usuário desligar-se do mundo real. Essa técnica é bastante explorada em jogos computacionais em primeira pessoa. Vantagens da utilização de RV não-imersiva é o baixo custo de implementação, a facilidade de uso e a não necessidade de adquirir dispositivos específicos para realidade virtual.

Sistemas semi-imersivos são aqueles que não exigem dispositivos imersivos para as aplicações, porém permitem alguma sensação de imersão ao usuário. Eles permitem que o usuário possa ver entre o ambiente virtual e o físico. Alguns dispositivos que caracterizam esse tipo de sistemas são as salas de projeções e dispositivos *surround*.

Sistemas híbridos são constituídos por imagens virtuais sobrepondo as imagens reais, caracterizando o que se chama de realidade aumentada. Geralmente, em sistemas de realidade aumentada, o usuário utiliza capacete com visor semitransparente que permite a sobreposição das imagens reais nas virtuais. Uma desvantagem desse sistema é a necessidade de combinar as duas imagens, sabendo a posição exata entre os objetos do mundo virtual e do mundo real.

Este capítulo estrutura-se da seguinte maneira: a seção 2.2 trata das interfaces com RV e sua aplicabilidade; a seção 2.3 discute os problemas do desenvolvimento de interfaces com RV; finalmente, na seção 2.4 são apresentadas as considerações finais deste capítulo.

2.2. Interfaces com Realidade Virtual e sua Aplicabilidade

Interfaces com RV podem ser consideradas como um tipo de interfaces pós-WIMP. Entende-se por interfaces pós-WIMP aquelas onde não é necessário botões e menus para que a interação possa ser realizada (Van Dam, 1997), usando dispositivos não convencionais nessa interação.

Em sistemas de RV, usuários podem transferir o conhecimento intuitivo do mundo real para o mundo virtual, o que pode tornar a interação humano-computador mais natural e aumentar a usabilidade da aplicação, já que o usuário, ao encontrar um ambiente que reflete a realidade, pode interagir melhor com o sistema através de analogias que, de acordo com Turroff (Turroff, 1997), são mecanismos cognitivos usados pelas pessoas para construir novos conhecimentos a partir de outros conhecimentos adquiridos.

Nesse sentido, interface com RV possui diversas características distintas em relação a outros tipos de interface, que potencializam sua utilização como interface humano-computador, como por exemplo à naturalidade e a intuitividade. Ela pode ser comparada às interfaces com manipulação direta, interfaces hipertexto e a do tipo *walk-up-and-use* (Kaur, 1998), como mostra a Figura 2.2.

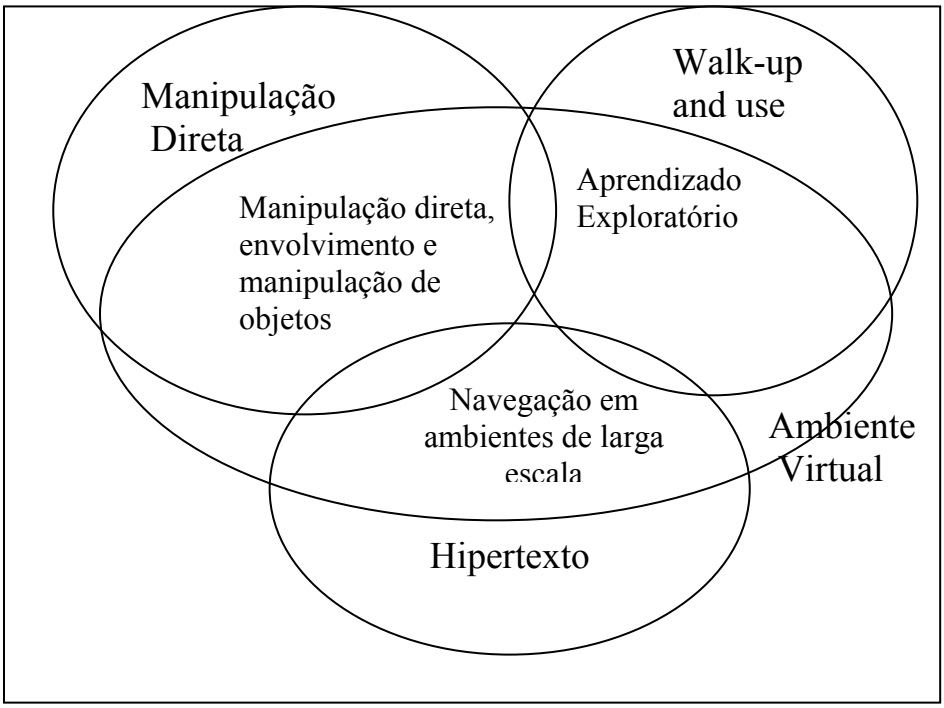

Figura 2.2 – Características de ambientes virtuais em relação a outros tipos de interfaces (Kaur, 1998)

Interfaces de **manipulação direta** são aquelas cuja interação é realizada através da manipulação de objetos computacionais (Kaur, 1998). As interfaces WIMP são o paradigma de interação das interfaces de manipulação direta. Os princípios da manipulação direta podem ser utilizados por interfaces com RV, devido à manipulação de objetos 3D. Além disso, o ambiente virtual representa o domínio em que o usuário se situa e esse domínio é visualizado através de uma representação tridimensional que forma o ambiente 3D, contendo uma série de objetos que podem ser manipulados diretamente pelo usuário nesse espaço determinado. No caso das interfaces WIMP, o espaço é representado por objetos de interesse que realizam uma função pré-determinada, por exemplo, um botão ou um ícone.

Interfaces com RV têm a vantagem de poder reduzir o processamento cognitivo, dependendo do domínio de aplicação, pois permitem uma interação mais realista do que as interfaces de manipulação direta, permitindo realizar tarefas da maneira natural e explorar os diversos sentidos do homem. A percepção do usuário acontece somente em relação ao ambiente gerado, isto é, o usuário tem a possibilidade de interagir com o ambiente como se ele estivesse presente nesse ambiente; em interfaces WIMP, o usuário precisa perceber a funcionalidade de objetos representados de forma 2D na tela do computador e saber quais as funções estão associadas com aqueles objetos (Kaur, 1998).

Outra característica usada por interfaces com RV derivadas das interfaces WIMP é a manipulação de objetos de interface. No caso das interfaces com RV as manipulações envolvem coordenadas 3D para qualquer ação do usuário, seja modificar a escala de um objeto ou rotacioná-lo. Nas interfaces WIMP a manipulação está relacionada somente à capacidade do usuário de clicar ou arrastar um objeto de interface para outro local.

Cabe mencionar, no entanto, que os objetos de interfaces WIMP podem ser aplicados às interfaces com RV como forma de permitir a manipulação de objetos do mundo virtual de maneira mais simples. As funções de seleção, rotação, escala, etc., podem ser selecionadas por usuários através de botões ou menus na interface da aplicação, permitindo que as ações do usuário sejam aplicadas aos objetos de interfaces 3D (Van Dam, 1997).

Para Bowman et al (2001), a interação 2D possui algumas vantagens em relação à interação 3D para alguns tipos de tarefas. Quando dispositivos táticos não estão presentes, os objetos de interface 2D proporcionam um senso de *feedback* maior para as tarefas de criação de objetos, escrita e anotação. As técnicas de seleção mais eficientes são as do tipo 2D, porém a manipulação pode requerer técnicas de interação 3D. Juntas, interações 2D e 3D podem criar interfaces de aplicações 3D mais fáceis de serem usadas e mais intuitivas aos usuários.

Já em relação às interfaces **hipertexto**, pode-se considerar que interfaces com RV possuem uma característica desse tipo de interface, que é o caso da navegação em espaços de larga escala (Kaur, 1998). Nessas interfaces, informações estão ligadas através de fragmentos de texto que servem como *links* ou botões entre páginas com informações relevantes ao domínio tratado, como também a outras informações referentes a domínios de informação diferentes.

Nas interfaces com RV, um objeto 3D pode ser utilizado para apresentar informações ou acessar dados referentes à função daquele objeto de interface, como também levar o usuário a outros domínios de aplicações, através da utilização de metáforas de interfaces como portas ou portais de tele-transporte. Além disso, o sistema pode alertar o usuário para um dado evento que exige atenção, como soar alarmes quando o usuário entra em um ambiente que ele não possui permissão (Kaur, 1998).

Em relação às interfaces ***walk-up-and-use*** a semelhança com interfaces com RV está relacionada ao fato dos dois tipos de interfaces permitirem um fácil aprendizado do usuário quando navega pelo ambiente virtual (Kaur, 1998). Essas interfaces são aquelas cuja proposta é aquelas que não necessitem de tempo de aprendizado para que o usuário possa manipular. Exemplos de sistema com interfaces *walk-up-and-use* são os quiosques de informações ou caixas-eletrônicos (Nielsen, 1993). Esses dois sistemas permitem que a utilização do sistema ocorra de maneira exploratória, realizada de maneira natural. Tanto Bowman et al (2001) quanto Kaur (1998) afirmam que as interfaces com RV são exploratórias por natureza, permitindo que o tempo exigido de aprendizado do iniciante seja bem curto, pois o conhecimento do ambiente onde o usuário vive pode ajudá-lo a interpretar o ambiente virtual.

É válido lembrar que em interfaces com RV têm a proposta de fazer a interação humano-computador ocorre de forma mais natural e intuitiva. A causa disso é uma série de fatores que viabilizam a utilização da RV como interface humano-computador, como aparência e comportamento semelhante aos objetos do mundo real, formas de apoio à interação e navegação, sensação de imersão e *feedback* (Bowman et al, 2001), (Kaur, 1998), (Turoff, 1997) e (Dachselt ,1999).

A aparência e comportamentos semelhantes aos objetos encontrados no mundo real são importantes, pois o homem utiliza modelos mentais como forma de entender o mundo virtual (Dachselt, 2000) (Turoff, 1997). Modelos mentais podem ser entendidos como um modelo imaginativo que o usuário faz a respeito das coisas que ele interage (Rocha & Baranauskas, 2000). Portanto, a similaridade dos objetos virtuais com objetos reais deve ser utilizada como metáfora para a funcionalidade que se deseja atribuir a cada objeto virtual. Dessa forma, a fácil interação

acontece porque os usuários mapeiam seus objetivos em relação à sua intenção, utilizando objetos do mundo virtual, devido à familiaridade que eles têm em relação a esses objetos no mundo real.

As formas de apoio à interação são importantes em ambientes virtuais pois elas servem de intercâmbio entre os dispositivos de entrada e saída e a ação do usuário na interação com o sistema (Bowman et al, 2001) (Pinho, 2000). Essas técnicas de apoio servem para a seleção, manipulação e navegação em ambientes virtuais. O realismo e a facilidade com que o usuário navega num ambiente virtual depende da capacidade de mesclar essas técnicas enquanto o usuário está interagindo com um determinado ambiente. A alteração entre uma técnica de apoio e outra é de responsabilidade do controle do sistema, que tem por objetivo modificar o estado do sistema, facilitar a navegabilidade e a interação nesses ambientes (Bowman et al, 2001).

A característica da imersão tem importância devido à necessidade dos ambientes virtuais em dar a sensação de presença que um usuário tem ao interagir com esse tipo de ambiente (Kaur, 1998). Em ambientes imersivos, a sensação de presença é mais real do que em ambientes não imersivos, onde se utilizam dispositivos convencionais para a realização de uma tarefa. A sensação de presença pode ser alcançada dependendo das técnicas de interação utilizadas e dos diversos tipos de *hardware*.

Por fim, o *feedback* é o responsável por dar a resposta do sistema referente a um estímulo causado pelo usuário ou pelo próprio sistema e deve ser em tempo real (Kaur, 1998). Em interfaces não-imersivas, o *feedback* é realizado através de mudanças comportamentais e geométricas do ambiente virtual, além de formas de sonorização 3D. Em relação aos ambientes imersivos, o *feedback* é realizado através dos dispositivos de entrada e saída do ambiente virtual, tais como os dispositivos táticos. Nesse caso, o usuário pode sentir a resposta do sistema através de canais multi-sensoriais (Bowman et al, 2001). Isso eleva a sensação de imersão, presença, além da interação parecer como se fosse realizada no mundo real, contribuindo para que a interação humano-computador seja realizada de forma mais natural e intuitiva.

Cabe mencionar que Bowman et al (2001) e Fencott & Isdale (2001) concordam que o desenvolvimento de interfaces com RV estão em uma fase de pouca maturidade. Para Bowman et al (2001), o sucesso do projeto de interfaces 3D leva em conta primeiramente a existência de novas pesquisas referentes aos fatores humanos em sistemas computacionais. Em segundo lugar, consideram a necessidade de utilizar técnicas de interação já desenvolvidas e novas propostas de interação. Em terceiro, considera a necessidade de utilizar criatividade e novos enfoques que podem ajudar na invenção de interfaces e novas técnicas de apoio à interação. E, finalmente, devem existir modelos de projeto de estratégias para o projeto de interfaces 3D.

Os dois trabalhos discutem a filosofia do projeto de interação 3D como sendo de dois tipos: a artística e a sistêmica. Eles concordam em dizer que esses dois tipos são problemáticos no projeto de interfaces com RV, cabendo mais pesquisas nessa área. A visão artística só se preocupa com a estética, sendo que a funcionalidade da aplicação pode ser prejudicada em função dessa prioridade. A visão sistêmica só se preocupa em pensar na funcionalidade, sem se preocupar com a questão estética.

Neste trabalho é dado o enfoque sistêmico no desenvolvimento de interfaces virtuais, por considerar a necessidade de analisar e projetar o domínio conforme as necessidades do usuário e possibilitar o reuso de componentes para interface dentro e fora do domínio de aplicação. Entretanto, a questão estética é tratada, embora de maneira indireta, por se considerar tratar-se de uma questão de usabilidade importante para o sucesso do projeto de uma aplicação com RV.

2.3. Problemas envolvendo o Desenvolvimento de Interfaces com Realidade Virtual

Ao projetar sistemas com RV, deve-se implementá-lo de forma que esse possa ser utilizado para representar um ambiente virtual em diversas circunstâncias, sem que afete a usabilidade e o projeto desse tipo de sistema. Porém, diferente do que acontece com outros tipos de interfaces, objetos 3D devem transmitir significado aos usuários a partir do seu conhecimento do mundo real, como também ambientar o usuário em um contexto, seja relativo ao passado, presente ou futuro, seja em relação à sua cultura ou a necessidade de simular uma situação real em um ambiente virtual. Para isso, o desenvolvimento de interfaces com RV deve ser ancorado por fatores que dão flexibilidade à criação e programação desse tipo de interface, através da solução dos seguintes problemas (falhas de projeto):

1. Projetar comportamentos não semelhantes a objetos do mundo real;
2. Embutir imagem 3D no código da aplicação;
3. Utilizar ferramentas de prototipação não-extensíveis, caso sejam utilizadas;
4. Não definir ou não utilizar arquiteturas 3D para construção de aplicações.

Para solucionar o problema 1 supõe-se que um objeto 3D deva ser criado baseado em características encontradas em um objeto do mundo real e que seu comportamento deva ser parecido com esse objeto. É importante ter aparência semelhante aos objetos do mundo real, devido à necessidade de um usuário perceber a funcionalidade do objeto no momento em que o visualiza. Comportamentos semelhantes aos encontrados nos objetos do mundo real são importantes devido à necessidade de resposta (*feedback*) do ambiente virtual ao usuário para que este entenda que a ação

está sendo realizada de maneira correta. Um objeto 3D que não tem o comportamento esperado, e conhecido, pode trazer problemas de usabilidade na interface e desorientação do usuário. Porém, a definição de comportamentos é um problema bastante difícil quando considerado o contexto de interfaces com RV. Os comportamentos do mundo real são ricos em detalhes e formas de interação com os objetos, que são difíceis de implementar em ambientes virtuais. Para isso, o desenvolvedor deve simplificar a interação com o objeto 3D, programando formas de interação similares porém devem ser mais fáceis e intuitivas.

O problema 2 se refere a embutir o objeto 3D no código da aplicação. Como destacam Döllner & Hinrichs (1998), muitos sistemas de RV utilizam objetos 3D que foram transformados em um conjunto de pontos e vértices e colocados dentro do código da aplicação. Embora, haja um maior desempenho das aplicações em carregar na memória um objeto 3D embutido, a necessidade de modificar o objeto ou o domínio de aplicação leva a realização de modificação e recompilação dos seus módulos a cada modificação ou inserção de objeto 3D na interface com RV. Nesse sentido, é importante a separação entre o objeto 3D e o código da aplicação. Em seu trabalho, Döllner & Hinrichs (1998), propõem a criação de uma biblioteca de comportamentos. Assim, os objetos 3D reutilizam o(s) comportamento(s) programado(s) para compor a interface em desenvolvimento. Outra vantagem nessa separação é que o objeto 3D pode ser modificado sem se preocupar com o comportamento. Isso facilitaria o reuso tanto do objeto 3D quanto do comportamento, até por pessoas que não conhecem programação, mas utilizam ferramentas de prototipação para criar interfaces com RV.

O problema 3 diz respeito à flexibilidade das ferramentas de prototipação de interfaces com RV. Embora não seja necessário seu uso, já que interfaces com RV podem ser geradas através de código, a dificuldade de programar aplicações de RV, o tempo maior de desenvolvimento e a grande quantidade de linhas de código, torna o uso de ferramentas de prototipação uma opção a ser considerada (Dachselt, 2001). No caso do trabalho de Döllner & Hinrichs (1998), um comportamento era reutilizado para um objeto 3D. Porém, os comportamentos definidos no código dão à ferramenta um número limitado de recursos a serem atribuídos aos objetos 3D. É possível perceber que quando esses comportamentos não são suficientes para o reuso em alguma situação, a ferramenta deve ser modificada para acrescentar um novo comportamento. Esse problema exemplifica a necessidade de que as ferramentas sejam flexíveis e extensíveis, pois devem ser projetadas para gerar interfaces com RV de diversos domínios. Conforme (Dachselt, 2001), as ferramentas devem ter capacidade de agregar novos objetos 3D e comportamentos sem que elas passem por modificações. Dessa forma, não têm funcionalidades limitadas a certos comportamentos e, por tanto, são capazes de gerar interfaces com RV para diversos domínios de aplicação, com

objetos 3D respondendo às ações do usuário adequadamente sem que a usabilidade da interface seja afetada. A capacidade de agregar novos comportamentos leva à necessidade de serem reunidos em bibliotecas para ser reutilizados no futuro.

A necessidade de definir formas de comunicação com comportamentos e com a interface com RV leva à necessidade de projetar ou utilizar arquiteturas de software, relacionada ao problema 4. Entende-se por arquitetura de software um conjunto de classes, objetos e relacionamentos agrupados para construir aplicações específicas (Coad, 1992). No caso de aplicações 3D, são necessárias arquiteturas que definem relacionamento de classes, desde o controle de primitivas gráficas até recursos específicos de RV, tais como controle de dispositivos de interação como, por exemplo, mouse, luvas e capacetes de RV. Essas arquiteturas são chamadas por Dörner & Grimm (2001) de arquiteturas 3D.

Devido à complexidade de ambientes virtuais, é comum a utilização de grafos de cenário. Entende-se por grafo uma árvore direcionada formada por nós de diferentes funcionalidades. Grafos de cenário otimizam o desempenho de aplicações com RV e o uso efetivo do hardware (VrJuggler, 2003). Eles são utilizados para representar cenários virtuais complexos e com muitos detalhes que podem ser formados por transformações e interações em cada nó do grafo.

Existem diversas implementações de grafos de cenários, bem como de arquiteturas. Na Tabela 2.1, são apresentadas algumas tecnologias no apoio ao desenvolvimento de aplicações com RV.

Tabela 2.1 – Tecnologias utilizadas para o desenvolvimento de aplicações com RV

Tecnologia	Classificação	Descrição	Tipo de Licença
OpenSG (Open, 2003)	Grafo de cenário	Implementado em C++, tem suporte a vários sistemas operacionais. Sua maior vantagem é que o grupo desenvolvedor é formado por várias empresas e universidades interessadas que auxiliam seu desenvolvimento. Necessita de uma arquitetura para a construção de interfaces com RV, além do grafo de cenário.	OpenSource
OpenInventor® (SUN, 2003a)	Arquitetura 3D e grafo de cenário	Implementado em C++, tem suporte a vários sistemas operacionais. Desenvolvido pela SUN Microsystems (www.sun.com), é destinado a aplicações 3D não muito complexas.	Comercial

OpenPerformer® (SUN, 2003b)	Grafo de Cenário	Implementado em C, tem suporte a vários sistemas operacionais. Desenvolvido pela SUN Microsystems (www.sun.com), é destinado a criação de aplicações 3D.	Comercial
WorldToolKit® (Sense8, 2003)	Arquitetura 3D e grafo de cenário	É uma arquitetura para aplicações de RV com suporte a grafo de cenário em suas versões mais recentes. É bastante utilizada no mercado.	Comercial
Java3D® (SUN, 2003c)	Arquitetura 3D e grafo de cenário	É uma arquitetura para aplicações 3D em geral, capaz de criar aplicações de RV com suporte a grafo de cenário. Implementado em Java, pela SUN Microsystems, têm uma especificação que foi formado por um aglomerado de empresas e universidades. Também é destinada a criação de aplicações 3D na Web.	Freeware
VrJuggler (VrJuggler, 2003)	Arquitetura 3D	Implementada em C, é um projeto desenvolvido por trabalhos de pesquisa, destinado a criar aplicações com RV imersiva e não-imersiva. Uma vantagem é seu suporte para um grande número de dispositivos de interação. Porém, pode necessitar de grafos de cenários para representar ambientes virtuais complexos.	OpenSource

Para resolver os problemas do desenvolvimento de interfaces com RV existem duas alternativas.

Uma delas é utilizar linguagens de *scripts* (Contigra, 2003). *Scripts* são interpretados pela aplicação de RV executados em tempo de execução. Uma vantagem dessa alternativa é a facilidade de criar código e novas funcionalidades em tempo de execução, através de ferramentas de edição de código ou construtores de código. A desvantagem é que essas linguagens devem ser planejadas para criar *scripts* flexíveis e incorporar não somente manipulações gráficas e interações, mas permitir criar novas funcionalidades como, por exemplo, suporte a acesso a banco de dados dentro de aplicações com RV.

Uma outra alternativa que resolve a desvantagem das linguagens de *scripts* é utilizar componentes de software carregados dinamicamente. Nesse caso, os componentes utilizam os recursos da linguagem de programação da ferramenta, podendo comunicar-se com outros componentes ou com a ferramenta, além de serem planejados através de um processo de desenvolvimento de componentes de software com ênfase na capacidade de reutilização. Aliado à capacidade de algumas linguagens, como Java, de permitir criar e recompilar classes e reconhecer métodos e atributos em tempo de execução, propriedade conhecida como Introspecção (SUN, 2003), aplicações que utilizam esse recurso são capazes de gerar código e novas funcionalidades. Outra vantagem é a criação de COTS *(Commercial off to Shelf)*, embora não seja comum em interfaces com RV, é explorado por (Dachselt, 2003), como forma de permitir que desenvolvedores de componentes possam desenvolver, distribuir e/ou comercializar componentes reutilizáveis que serão usados por outros desenvolvedores em seus projetos com RV.

2.4. Considerações Finais

Este capítulo apresentou uma introdução à RV, discutiu os problemas do desenvolvimento de aplicações com RV e possíveis alternativas para resolver os problemas apresentados. Pode-se pensar que desenvolvimento baseado em componentes é interessante devido ao processo de criação favorecer sua reutilização e ser planejado conforme os requisitos da aplicação em desenvolvimento, podendo criar interfaces com mais usabilidade, pois incorporam componentes que foram desenvolvidos e testados em outras ocasiões. Entretanto, esta questão deve ser estudada para verificar sua veracidade.

Como este trabalho explora o desenvolvimento baseado em componentes para interfaces com RV, o capítulo 3 apresenta o desenvolvimento de aplicações baseado em componentes de software e fatores relevantes, bem como, o desenvolvimento de interfaces com o usuário através de reuso de componentes.

3. Reuso de Componentes para Interfaces com RV

3.1. Considerações iniciais

Reuso, no contexto de Engenharia de Software, pode ser entendido como a reutilização de partes pré-existentes do processo de desenvolvimento de um software. Cada parte pré-existente representa um produto ou artefato de software, tais como: requisitos, documentação, dados, arquitetura, estruturas, ou qualquer informação necessária ao projeto de um software.

A maneira como isso é realizado e a forma com que esses artefatos são utilizados obedece a uma classificação proposta por Prieto-Diaz (1993). Essa classificação divide as formas de reuso em seis tipos:

- Por substância: quando a essência do item do software a ser reutilizado é claramente definida. Essa essência pode ser vista como idéias e conceitos, através de reuso de conceitos formais; ou artefatos e componentes (que correspondem ao reuso de código de fato); ou procedimentos e habilidades, que definem o reuso de processos de desenvolvimento de software.
- Por modo define como o reuso é conduzido: planejado e sistemático ou ad-hoc e oportunista (prática informal, conduzida individualmente).
- Conforme escopo: define a forma e extensão do reuso. Esse tipo sugere que o reuso pode ser vertical, relacionado à reutilização dentro de um domínio de aplicação, ou horizontal, que sugere o reuso quando artefatos genéricos podem ser reutilizados em diferentes aplicações em diferentes domínios.
- Conforme técnica: estabelece a abordagem utilizada para implementar o reuso. Pode ser do tipo composição ou geração. No primeiro caso, os artefatos implementacionais para reuso formam blocos a ser encaixados em novos sistemas. No outro caso, o reuso é feito no nível de especificação, através de uma aplicação ou gerador de código.
- Conforme intenção: define como os artefatos são reutilizados. Pode ser o reuso caixa-preta em que esses artefatos são reutilizados sem modificação ou o reuso caixa-branca, em que são modificados para atender um requisito específico do projeto de um software.
- Pelo produto: define quais produtos, gerados ao fim de cada fase do processo de desenvolvimento de software, são reutilizados. Esses produtos podem ser o código fonte, o projeto, a especificação, objetos, texto ou arquitetura.

Essa classificação mostra as diversas formas de se aplicar o reuso. Em relação às partes implementacionais de software, chamados de componentes de software, não existe um consenso

sobre a sua melhor definição. No entanto, vários pesquisadores concordam que existem algumas características que um componente deve prover. Essas características são as seguintes: realizar uma função específica e por completo (Kozaczynski, 1999) (Sametinger, 1997), (Yacoub et al, 1999), prover um conjunto de interfaces do componente bem definidas (CBSE, 1998, 1999, 2000) (Sametinger, 1997), (Szyperski, 1998), estar contido em uma arquitetura de software (Presman, 2001) e disponibilizar uma documentação adequada (Yacoub et al, 1999b), (Sametinger, 1997). Portanto, levando em consideração esse conjunto de autores pode-se definir que componentes de software são: "a*rtefatos autocontidos, claramente identificáveis, que descrevem ou realizam uma função específica e têm interfaces claras em conformidade com um dado modelo de arquitetura de software, documentação apropriada e um grau de reutilização definido."* (Werner & Braga, 2000). Essa definição é que será defendida neste trabalho.

Este capítulo está dividido da seguinte maneira: a seção 3.2 analisa os fatores relevantes no desenvolvimento de aplicações baseado em componentes; a seção 3.3 discute a Engenharia de Software baseado em Componentes, a Engenharia de Domínio e o Desenvolvimento baseado em Componentes; na seção 3.4 é apresentado o reuso de componentes em interfaces com o usuário; por fim, na seção 3.5, apresentam as considerações finais deste capítulo.

3.2. Fatores Relevantes no Desenvolvimento de Software Baseado em Componentes

Analisando Pressman (2001) e os trabalhos de Silva (2001), Werner & Braga (2000), Crnkovic & Larsson (2000) e Karlsson (1995) é possível afirmar a existência dos seguintes fatores que facilitam o desenvolvimento de software baseado em componentes:

- Generalidade e eficiência de um componente;
- Utilização de um modelo arquitetural bem definido;
- Planejamento e Automação;
- Portabilidade e compatibilidade;
- Apoio da organização;
- Testes e revisões técnicas;
- Gerenciamento de configuração;

Generalidade significa que o desenvolvedor deve reconhecer características comuns e específicas nos componentes a serem desenvolvidos, permitindo criar componentes mais genéricos que podem ser adaptados a situações específicas.

Já a utilização de um modelo arquitetural bem definido visa a importância de especificar camadas dentro dessa arquitetura, para que componentes mais específicos possam reutilizar componentes mais genéricos como base para a implementação de novos componentes reutilizáveis.

Por planejamento entende-se que uma atividade planejada facilita o cumprimento das metas a serem alcançadas na atividade de mudança de domínio.

Ser portável e compatível com diversas plataformas e sistemas operacionais é importante devido à necessidade de se comunicar com outros componentes e estes poderem ser acessados por diversas linguagens de programação.

Em relação à automação, o uso de ferramentas computacionais ajuda o planejamento e permitem uma maior eficiência no projeto de componentes reusáveis.

O apoio da gerência é importante na prática de reuso no desenvolvimento de software e conseqüentemente na mudança de domínio de software. Como citado anteriormente, Poulin (1999) diz que um dos problemas que inibem o reuso é a falta de apoio da organização frente ao reuso de componentes de software.

Testes e revisões técnicas ajudam a adquirir maior conhecimento relativo ao componente a ser desenvolvido e em relação aos problemas de funcionalidades e adaptações que um componente pode sofrer.

Gerenciamento de configuração é um meio de controlar versões do desenvolvimento de um componente.

3.3. Engenharia de Software Baseada em Componentes

A Engenharia de Software baseada em Componentes (ESBC) é um processo de desenvolvimento de software realizado concorrentemente a partir de duas atividades: o desenvolvimento para reuso e o desenvolvimento com reuso.

O desenvolvimento para reuso objetiva criar componentes reutilizáveis em um ou mais domínios de aplicação. Os componentes desenvolvidos integram uma arquitetura de software existente para aquele domínio e são armazenados em repositórios de componentes para serem utilizados futuramente.

Um repositório de componente é um lugar utilizado para armazenar componentes implementacionais. Seu objetivo é possibilitar uma forma fácil e rápida de localizar componentes aptos ao reuso. Eles são compostos por mecanismos de busca e seleção de componente e normalmente armazenam informações referentes à documentação do componente, domínios nos

quais podem ser empregados e palavras-chaves do domínio empregado, utilizadas para localizar componentes aptos ao reuso.

O desenvolvimento com reuso utiliza componentes implementacionais já desenvolvidos que vão integrar o desenvolvimento da aplicação através da composição de um ou mais componentes, formando uma aplicação no domínio de software que foi escolhido, como mostra a Figura 3.1.

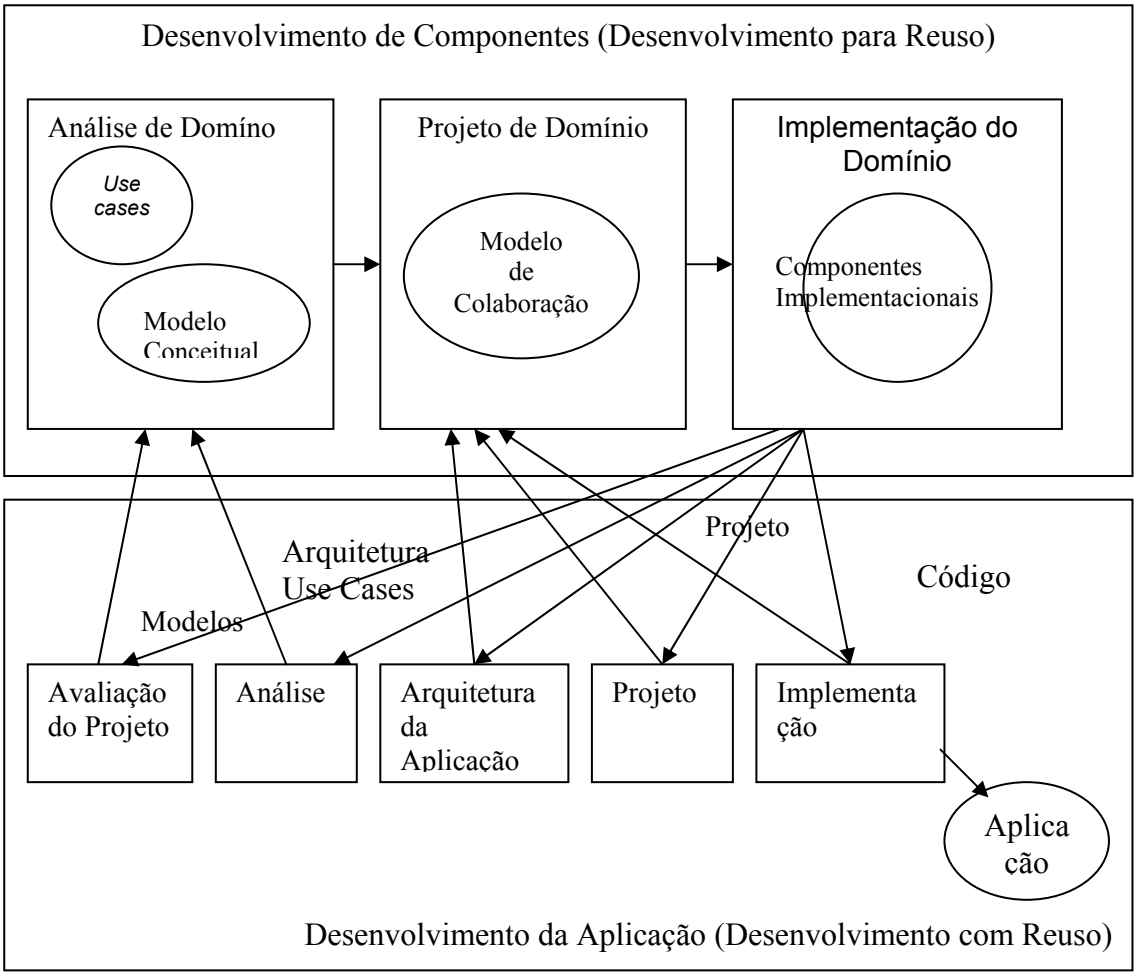

Figura 3.1 – Engenharia de Software baseada em Componentes (Braga, 2000)

O processo de desenvolvimento com reuso começa a partir da **avaliação do projeto**, que determina a necessidade de especificar componentes para o domínio de aplicação que está sendo desenvolvido. Dessa forma, requisitos de software são levantados e analisados para uma família de aplicação a que esse componente se destina, analisando seus requisitos funcionais a serem implementados para reutilização. Ao mesmo tempo são pesquisados **componentes implementacionais** já implementados que possam ser reutilizados para compor a aplicação em desenvolvimento. Nesse caso, todo o conjunto de análise, projeto e implementação vão ser incorporados ao desenvolvimento da aplicação buscando indicar possíveis modificações e

integrações do componente à aplicação em desenvolvimento, reutilizando todo o processo de desenvolvimento do componente escolhido, que será integrado à aplicação.

No processo de **desenvolvimento de componentes para reuso,** depois de levantados os requisitos funcionais e desenvolvida a análise, na fase de **projeto do domínio**, os componentes são projetados conforme uma **arquitetura de aplicação**, como comentado na seção 3.2 deste capítulo, que indica como é realizada a interoperação da aplicação com os demais componentes reutilizados e que são desenvolvidos conforme os requisitos de projeto e implementação para compor a aplicação em desenvolvimento. Quando um **componente implementacional** é reutilizado em uma aplicação, todo o conjunto do processo de desenvolvimento do componente é reaproveitado (use cases, modelo conceitual, modelo de colaboração e código), sendo que esse componente integra a aplicação com suas devidas funcionalidades já testadas, reduzindo o tempo e custo gasto com a análise, projeto e implementação da aplicação desenvolvida e, portanto, aumentado a qualidade de software.

As fases de análise e projeto de domínio são realizadas através da Engenharia do Domínio e a implementação de **componentes implementacionais** se dá através do Desenvolvimento Baseado em Componentes que serão explicados em seguida.

3.3.1. Engenharia de Domínio

A Engenharia de Domínio (ED) é uma atividade que tem por objetivo estudar uma aplicação de forma planejada e sistemática, caracterizando um processo completo para especificação de componentes reutilizáveis (Werner & Braga, 2000). Nesse sentido, a ED identifica componentes de software em todas as fases do desenvolvimento de uma aplicação.

A ED, conforme estudo de Werner & Braga (2000), é dividida em três fases:

1. Análise de domínio;
2. Projeto do domínio;
3. Implementação do domínio;

Na primeira fase, a análise de domínio, são estudados os requisitos comuns de um conjunto de aplicações, que pode ser entendido como uma família de aplicações, nas quais essa análise tem o papel de identificar possíveis casos de reutilização de componente. Possibilita reconhecer componentes de software, maximizando a reutilização desses componentes, graças ao trabalho planejado e sistemático que a análise de domínio propicia (Werner & Braga, 2000).

Dessa forma, a análise de domínio ajuda o engenheiro de software a criar componentes para reuso, tanto em nível vertical quanto em horizontal, como também reconhecer componentes já implementados, suas características e adaptações possíveis para que o desenvolvimento com reuso

de componentes também seja possível. Todo conjunto de componentes, juntamente com a coleção de aplicações existentes, bem como aplicações necessárias futuramente, é objeto de estudo da análise de domínio. Esse conjunto é chamado de domínio do problema (Werner & Braga, 2000).

A segunda fase da ED corresponde ao projeto de domínio, que consiste em identificar e projetar componentes que possam ser desenvolvidos de maneira generalizada, para expressar requisitos comuns presentes nas aplicações do domínio estudado. Para isso, são necessárias as informações identificadas na fase de análise de domínio.

A terceira e última fase da ED é a implementação do domínio, com o objetivo de transformar as informações obtidas na fase de projeto de domínio em um conjunto de componentes implementacionais, ou em outros serviços, como a manutenção de componentes. A fase de implementação do domínio é realizada através das atividades do Desenvolvimento Baseado em Componentes (DBC).

3.3.2. Desenvolvimento Baseado em Componentes

DBC é composto por três atividades: qualificação, adaptação e composição.

A qualificação ocorre quando um componente já existente é candidato ao reuso. A procura por componentes existentes ocorre através da seleção de componentes presentes nos repositórios de componentes e resulta numa coleção de componentes candidatos a reuso.

Cada componente dessa coleção é qualificado no intuito de analisar se o reuso desse componente é mesmo vantajoso. Se o reuso for vantajoso, o componente apto ao reuso deve passar por um processo que consiste em analisar sua funcionalidade em relação aos requisitos funcionais do software. Se o componente não necessitar de modificações e atender os requisitos do software, ele pode ser reutilizado sem modificações, o que caracteriza o reuso do tipo caixa-preta (Prieto-Diaz, 1993).

Se, porém, o componente necessitar ser modificado, o que caracteriza o reuso do tipo caixa-branca, esse componente é adaptado a um requisito específico do software em desenvolvimento e composto com essas novas características. O novo componente, resultante da adaptação, é acrescentado ao repositório como um novo componente.

O processo de adaptação de um componente é a segunda atividade do DBC. Nesse sentido, a adaptação de componente objetiva modificar uma característica que não se encaixa nos requisitos funcionais do sistema (ou que o componente não possui) ou na arquitetura da aplicação. As adaptações podem ocorrer em nível de código, quando esse está disponível, ou através da interface do componente (Pressman, 2001). Nesse último caso, o componente deve implementar o

conjunto de funções ou métodos definidos pela interface, que é chamada pela aplicação, quando o componente é utilizado.

Diferente do processo de adaptação, a composição de componentes (que corresponde à terceira atividade do DBC) utiliza o processo de desenvolvimento do componente ou sua qualificação e adaptação (se esse componente não for novo) para a composição de componentes. Nesse caso, é preciso definir uma arquitetura da aplicação que os componentes implementacionais devem seguir. Segundo Szyperski (1998), uma arquitetura de software para atender ao DBC consiste de um conjunto de decisões relacionadas à plataforma operacional, ao modelo de composição dos componentes e ao projeto de interoperação para o modelo de composição. Nesse sentido, uma arquitetura de aplicação define três aspectos principais:

1. A infra-estrutura para interconexão dos componentes (serviços básicos, restrições e ligações semânticas) (Shaw & Garlan, 1996);
2. A composição dos componentes em termos funcionais;
3. A interoperação de componentes, ou seja, a habilidade dos componentes de se comunicar e cooperar sem levar em consideração diferenças de linguagem, plataforma de execução e interfaces (Sametinger, 1997).

A implementação de componentes deve levar em consideração os produtos levantados durante a realização da análise e projeto do domínio e a arquitetura de aplicação, como forma de garantir que esse componente atenda às necessidades do domínio em questão.

A composição também pode ser realizada conforme um padrão de desenvolvimento de componentes, caso seja utilizado. Nesse caso, o componente deve seguir as regras definidas pelo padrão utilizado, pois essas regras definem como é feita a comunicação entre os diversos componentes dentro desse padrão. Os padrões existentes no mercado, apontados por Pressman (2001) são: OMG/Corba®, Sun JavaBeans® e Microsoft COM®.

3.4. Desenvolvimento de Interfaces baseado em Reuso de Componentes

O reuso de componentes de interfaces WIMP baseia-se no uso de *widgets*, que encapsulam geometria e comportamento (inclusive tratamento de eventos) e permitem que um componente possa ser reutilizado em novos contextos através de pequenas modificações (Cooper & Gray, 2002).

Essas modificações consistem em adaptar atributos e eventos para as necessidades do projeto de interface.

Ao projetar uma interface, é necessário definir como cada componente presente nesse projeto vai reagir quando um evento é acionado. Nesse caso, é preciso definir os comportamentos do componente para interface WIMP, configurando-o para realizar uma tarefa específica. Quando for necessário criar um novo tipo de componente de interface para uma aplicação, é importante salientar que o comportamento de um componente de interface deve ser reconhecido na fase de análise do componente, só assim é possível saber quais os comportamentos precisam ser implementados para facilitar o reuso.

Um aspecto interessante dos componentes para interfaces WIMP é sua generalidade (Grundy & Hosking, 2000). Eles são passíveis de ser reutilizados em um número muito grande de projetos de interfaces independente da arquitetura da aplicação. Isso possibilita seu reuso em diversas interfaces de domínios variados.

Em interfaces com RV, sistemas podem ser desenvolvidos sem o uso de componentes de software. Assim, o reuso de funções é proporcionado através da chamada de funções para tarefas específicas dentro de uma biblioteca de funções que o desenvolvedor cria para a construção de aplicações desse tipo. Já em interfaces com RV desenvolvidas a partir de reuso de componentes, o desenvolvedor projeta componentes para cada tipo de funcionalidade. É comum o desenvolvimento de arquiteturas 3D, com suas devidas correlações entre classes e suas interoperações entre componentes. Isso ajuda a reusabilidade de um componente e facilita a criação de interfaces. Nesse caso, os componentes são executados através da sua interface.

Um exemplo de arquitetura 3D para o desenvolvimento de aplicações com RV utilizando componentes de software é o VrJuggler (VrJuggler, 2003). O VrJuggler, é uma arquitetura 3D gratuita, formada por um aglomerado de classes e componentes que formam sua arquitetura 3D e controlam desde a renderização de objetos como controle de dispositivos de entrada/saída.

Geralmente, a utilização de componentes de software para a construção de interfaces com RV produz interfaces onde objetos presentes na interface têm uma única forma de interação, controlada pela aplicação. Conner et al (1992) introduzem o conceito de componentes para interfaces 3D como sendo uma encapsulamento de imagem e comportamento. Essa definição é similar à definição de componentes para interfaces WIMP, permitindo que cada componente de interface tenha seus próprios comportamentos e formas de interação. O uso de componentes para interfaces 3D tornou-se comum em pesquisas realizadas por (Dachselt, 1999),(Dörner & Grimm, 2001) entre outros, por permitir criar objetos 3D que respondem adequadamente à ação do usuário, aumentado a usabilidade da interface.

É importante salientar que os componentes para interfaces 3D, conhecidos como componentes 3D (Dachselt, 2001), não possuem as três características de uma interface com RV (Burdea & Coiffet, 1994) que são: imersão, interação e envolvimento. Eles somente são os objetos com os quais o usuário interage. O componente 3D é utilizado em aplicações com RV, entretanto a aplicação deve garantir as características de imersão, interação e envolvimento, para compor essas interfaces.

Assim, componentes 3D são uma aglomeração de imagem(s) e comportamento(s) que definem sua funcionalidade. Entende-se por comportamento todo conjunto de transformações lineares, renderização, interação com o usuário, controle de dispositivos de entrada/saída, tratamento de colisão, animação, etc. Assim, cada componente 3D tem seu próprio comportamento, semelhante aos componentes para interfaces WIMP. Porém, esses comportamentos e geometrias são mais complexos no componente 3D.

Pode-se classificar o desenvolvimento de interfaces com RV como mostra a tabela 3.1.

Tabela 3.1 – Tipos de desenvolvimento de interfaces com RV (adaptada de Dachselt (2001) e Contigra (2003))

<table>
<tr><td rowspan="2">Não orientado a componentes</td><td>Normais</td><td></td></tr>
<tr><td>Baseados em Scripts</td><td></td></tr>
<tr><td rowspan="4">Orientado a componentes</td><td>Não orientado a componentes 3D</td><td></td></tr>
<tr><td rowspan="3">Orientado a componentes 3D</td><td>Arquiteturas Proprietárias/Scripts</td></tr>
<tr><td>Arquitetura Padrão de Componentes</td></tr>
<tr><td>Documentos Declarativos</td></tr>
</table>

Os tipos de desenvolvimento que se baseiam em **conceitos de componentes 3D** são aqueles que contêm comportamentos associados a uma determinada imagem, e não necessariamente utilizam a definição de composição de geometria e comportamento. O tipo de desenvolvimento baseado em documentos declarativo é aqueles que especificam componentes 3D a partir de linguagens declarativas, como por exemplo, XML® *(eXtended Markup Language)* (W3C, 2003c). A grande vantagem desse tipo de desenvolvimento é que as descrições de componentes podem ser implementadas em qualquer linguagem e plataforma de software, não congestionam a largura de banda quando utilizadas na Web e em sistemas distribuídos, pois não são necessários arquivos binários. Os pesquisadores que implementam esse tipo de desenvolvimento defendem uma

padronização para a descrição de componentes 3D. Eles afirmam que a especificação de objetos 3D do formato X3D (Rudolf, 1999) não é suficiente para especificar comportamentos de objetos 3D. Por isso, existe uma série de pesquisadores que estão desenvolvendo especificações para a descrição de componentes 3D baseados na especificação do X3D, algumas dessas especificações são: comportamentos de objetos 3D (Behavior3D (Dachselt, 2003)), descrição de interação com os objetos (InML (Figueroa et al, 1999)), sonorização 3D (Hoffmann et al, 2002), acesso a banco de dados (Walczak & Cellary, 2002), entre outros.

O tipo de desenvolvimento através do uso de **arquiteturas padrões de componentes** são aquelas que utilizam arquiteturas conhecidas pelo mercado, como por exemplo, Sun JavaBeans® e Microsoft COM®. Embora sejam implementações binárias, utilizam os recursos oferecidos pelas arquiteturas padrões. Isso facilita o desenvolvimento de aplicações baseado em componentes em uma plataforma de software escolhida e possibilitam integrar componentes com aplicações que utilizam uma determinada arquitetura de componentes.

Arquiteturas proprietárias são aquelas que utilizam arquiteturas e conjunto de componentes criados especificamente para uso próprio. Nesse caso, um componente 3D que utiliza essa arquitetura não pode ser portado a outras arquiteturas e linguagens de programação.

As implementações que utilizam **componentes de software** são aquelas que não são providas de comportamentos próprios para cada objeto 3D da interface. Nesse caso, é comum que todos os objetos tenham uma única forma de interação, embora possam apresentar características comuns de componentes 3D, as interações e comportamentos não estão fisicamente agregados a um objeto 3D.

O último tipo se refere às implementações de RV que **não são orientadas a componentes.** Esse tipo subdivide-se em dois grupos: geração de interfaces com RV sem a utilização de componentes ou através de *scripts* (pequenos programas interpretados em tempo de execução e que realizam animações e mudanças comportamentais em objetos 3D). As implementações não orientadas a componente são ainda as mais exploradas, embora não permitam o reuso.

3.5. Considerações Finais

Este capítulo apresentou a ESBC, a ED e suas fases e o DBC. Foi apresentado também, o conceito de componentes 3D e a elaboração de interfaces com RV através do apoio de reuso de componentes.

Neste trabalho é explorado o uso orientado a componentes 3D para interfaces com RV numa arquitetura padrão de componentes, conforme elementos em destaque da Tabela 3.1.

No próximo capítulo é apresentado o ambiente de apoio GaCIV, objeto de estudo deste trabalho, que foi modificado para permitir reuso de componentes 3D no desenvolvimento de interfaces com RV, suas funcionalidades e sua estrutura.

4. O Ambiente GaCIV para Apoio ao Projeto de Interfaces com Realidade Virtual baseado em Reuso de Componentes

4.1. Considerações iniciais

Discutida a problemática da elaboração de interfaces com RV, o desenvolvimento de componentes e a aplicação destes no que refere à criação de interfaces, este capítulo apresenta o ambiente de apoio GaCIV, para elaboração de interfaces com RV através do reuso de componentes 3D, considerando o que foi descrito anteriormente.

A seção 4.2 apresenta o Ambiente GaCIV (Gabaritos Configuráveis para a elaboração de Interfaces com realidade Virtual), sua proposta, funcionalidade e filosofia. A seção 4.3 diz respeito a ESBC e fatores relevantes, como o GaCIV pode gerar interfaces conforme esses fatores e quais proveitos os componentes apresentam quando implementados conforme a visão conceitual do desenvolvimento de interfaces com RV. A seção 4.4 mostra a estruturação do GaCIV, implementado considerando o conceito de reuso de componentes. A seção 4.5 apresenta as novas funcionalidades do Ambiente GaCIV em relação às versões anteriores. Por fim, na seção 4.6 são apresentadas as considerações finais deste capítulo.

4.2. O Ambiente GaCIV

Tendo por definição que um ambiente de apoio é um software que guia, orienta ou apóia o desenvolvedor em suas tarefas do processo de desenvolvimento de software e ferramenta, na engenharia de software, é aquele que automatiza uma ou mais tarefas do desenvolvimento de software, o GaCIV, projeto desenvolvido pelo grupo de Interação Humano-Computador (IHC) do Departamento de Computação da Universidade Federal de São Carlos, é um ambiente que auxilia usuários, projetistas de interfaces e engenheiros de software na elaboração de sistemas com RV não-imersiva.

Como foi discutido no capítulo 2, nos sistemas não-imersivos o mundo real não é ocultado e o usuário não perde a noção do ambiente onde se encontra, já que utiliza dispositivos convencionais de visualização como monitores de vídeo. Essa característica é devida não somente à restrição de equipamentos mas tornar a RV acessível aos usuários de sistemas computacionais, sem que eles tenham que adquirir equipamentos sofisticados para a interação com o sistema (Assis, 2001).

O GaCIV utiliza templates para a construção de interfaces com RV, denominado gabaritos configuráveis. Um gabarito é formado por um ambiente virtual e um conjunto finito de objetos que representam um domínio de software. Eles são utilizados como modelos semi-prontos para a geração de interfaces com RV. Como permitem modificar a posição e rotação dos objetos, retirar e colocar inúmeras cópias dos objetos na interface com o usuário, conforme os requisitos da aplicação em desenvolvimento, são chamados gabaritos configuráveis.

Na primeira versão do sistema, os gabaritos continham objetos 3D em posições pré-definidas, sendo que os objetos do gabarito eram implementados no próprio código do GaCIV, não permitindo a escolha de objetos e suas posições, nem adição e alteração de gabaritos, o que limitava seu uso. Quando um gabarito era usado, o desenvolvedor atribuía *links* de aplicação para cada objeto 3D, e o arquivo produzido era visualizado por um *browser*, chamado InterViewer.

A Figura 4.1 mostra uma tela da primeira versão do GaCIV (Appel et al, 1999).

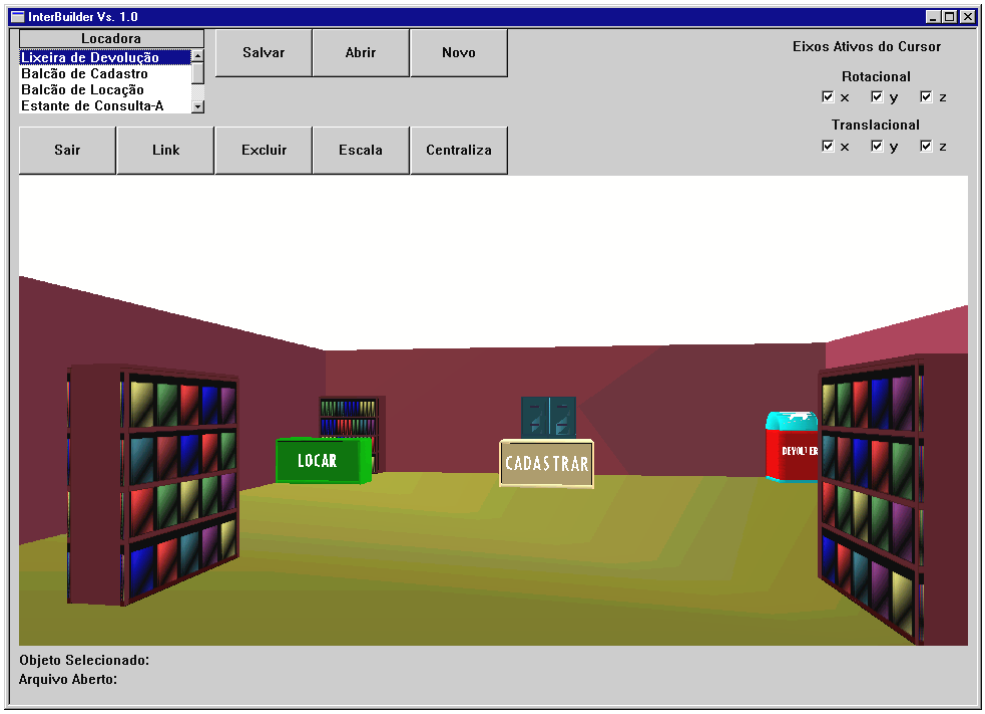

Figura 4.1 – Uma tela da primeira versão GaCIV

A segunda versão tinha significativas mudanças em relação à anterior. A mudança para o paradigma de orientação a objetos adicionou recursos como herança, conexão de mensagens, abstração e encapsulamento. A partir dessa versão, gabaritos podiam ser acrescentados e editados, bem como as interfaces geradas com o apoio destes. Ambientes e objetos eram armazenados em banco de dados com possibilidade de adicionar uma descrição para eles. Nessa versão, um pacote

de aplicação era gerado contendo ambiente, objetos e aplicações, sendo instalado pelos usuários através de um programa instalador (Assis, 2001).

Essas duas versões foram feitas com o uso dos softwares Microsoft Visual C++® e WorldToolkit® e banco de Dados Microsoft Access®, sendo que o GaCIV lia imagens 3D de extensão **.nff* própria da biblioteca do WorldToolkit®. O uso excessivo de ponteiros dessa biblioteca e falhas de algumas de suas funções, a necessidade de otimizar a manipulação dos dados armazenados no banco de dados relacional escolhido, a inexistência em C++ de um gerenciador de memória capaz de eliminar referências esquecidas (Assis, 2001), bem como a necessidade de portabilidade e utilização de softwares livres, levou ao processo de reengenharia da segunda versão para a linguagem Java, utilizando Java 3D como arquitetura 3D. A terceira versão baseia-se nesse processo para compor interfaces com RV com apoio de componentes de software.

Essa versão utiliza componentes 3D para permitir criar componentes para interfaces com RV que possuem comportamentos próprios e usa JavaBeans como padrão de desenvolvimento de componentes, transformando o GaCIV em um gerador de repositórios de componentes que são usados para gerar interfaces baseadas em componentes, como proposta de trabalho apresentada por Albertin (2002). Os gabaritos passaram a ser formados por um ambiente e um conjunto de componentes 3D que caracterizam um domínio de software, sendo os repositórios de componentes. Dessa forma, o GaCIV armazena quatro elementos: ambientes, comportamentos, componentes 3D e *links* de aplicação.

O GaCIV gera gabaritos e esses são repositórios de componentes 3D. Assim, ele gerencia o uso desses gabaritos na geração de interfaces para sistemas com RV. O GaCIV armazena ambientes que são usados para montar gabaritos e vão conter os componentes 3D representantes de um determinado domínio de software. Armazena comportamentos, que são um conjunto de funcionalidades relacionadas a um objeto 3D e determinam uma forma específica de interação e manipulação de um objeto 3D num cenário de RV. Armazena componentes 3D que são formados por uma imagem 3D e um comportamento que melhor expressa a interação do componente 3D em um cenário virtual. Por fim, armazena *links* de aplicação que são utilizados como vínculo entre um componente 3D e uma aplicação. Dessa forma, a utilização de um comportamento associado a um objeto 3D produz um componente 3D que possui interação e manipulação própria em uma interface com RV.

Como as demais, a terceira versão do GaCIV é formada por duas ferramentas: **InterBuilder e InterViewer**. A diferença entre a terceira versão das demais é o fato do InterBuilder apresentar cinco opções em sua barra de ferramentas, entre elas, três opções principais: **iStore**, **iBuilder** e **iPackage**, como mostra a Figura 4.2.

Figura 4.2 – Opções da Barra de Ferramentas do InterBuilder

O InterBuilder oferece recursos para a criação de interfaces com RV de maneira simples e prática. Ele se baseia em repositórios de componentes 3D, chamados gabaritos configuráveis, que podem ser entendidos como um ambiente que tem objetos de contextos semelhantes e que são oferecidos ao desenvolvedor como forma de reuso na elaboração da interface (Assis, 2001). Dessa forma, é possível criar um gabarito para cada domínio de aplicação, como por exemplo, locadora de vídeo, oficina mecânica, escritório, entre outros.

Um aspecto importante do InterBuilder é o modo como está estruturado, pois possibilita que o desenvolvimento da interface seja completamente dissociado do desenvolvimento da aplicação, garantindo, dessa forma, a independência de diálogo. Com essa característica, a ferramenta GaCIV pode ser usada tanto para a construção de uma interface para novos sistemas como para apoio à reengenharia de interfaces de sistemas legados (Soares, 2002). O Interbuilder também pode ser visto como uma ferramenta de prototipação, pois permite um refinamento iterativo a partir da inclusão e multiplicação de componentes 3D oferecidos durante o projeto de interfaces com RV.

A opção iStore do InterBuilder é responsável por oferecer recursos para armazenamento dos quatro elementos que o GaCIV manipula: ambientes, comportamentos, componentes 3D e *links* de aplicação, como mostra a Figura 4.3. É através dessa opção que o desenvolvedor acrescenta mais elementos a serem oferecidos pelo GaCIV na elaboração de gabaritos e interfaces.

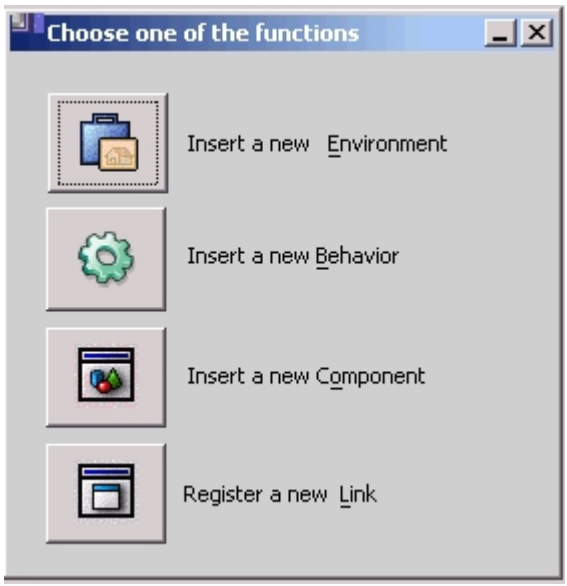

Figura 4.3 – Tela do iStore

A opção iBuilder do InterBuilder é o próprio editor de gabaritos e interfaces. Através dessa opção os elementos cadastrados estão disponíveis para a criação e edição de gabaritos configuráveis e interfaces com RV, como mostra a Figura 4.4.

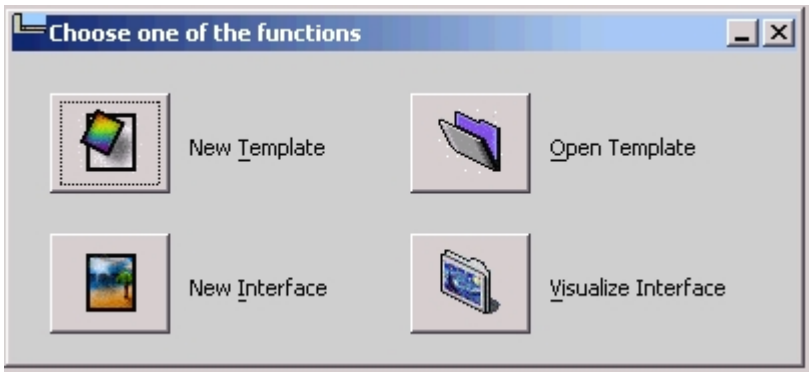

Figura 4.4 – Opções do iBuilder

A opção iPackage do InterBuilder é responsável por cadastrar banco de dados e gerar pacotes de aplicação a partir de interfaces desenvolvidas. A Figura 4.5 mostra as opções pertencentes ao iPackage.

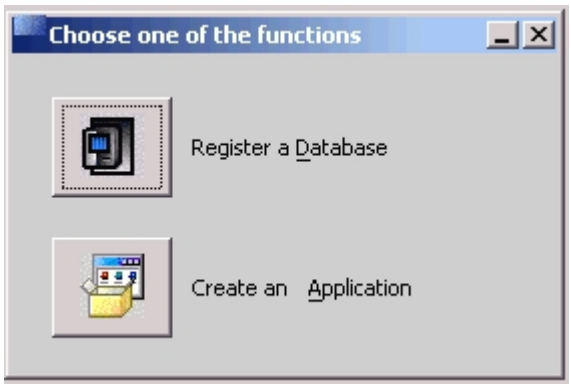

Figura 4.5 – Opções do iPackage

As funcionalidades de cada opção do InterBuilder são apresentadas com mais detalhes na seção 4.5 deste capítulo.

O InterViewer é o visualizador de interfaces produzidas pelo InterBuilder. Assemelha-se com um *browser* no qual usuários interagem com o ambiente virtual, navegando no ambiente e acionando os componentes 3D de sua escolha, que podem estar associados com *links* de aplicação e outras interfaces com RV.

Além das funcionalidades das versões anteriores, essa tem a preocupação de melhorar a usabilidade do ambiente e proporcionar mais independência com relação aos softwares de criação de imagens 3D. Essa versão permite utilizar mais formatos de imagens 3D, tais como: VRML, 3D Studio®, Autocad®, WaveFront Maya® além de serem extensíveis, já que componentes 3D podem ler novos formatos de imagens. Além disso, essa versão é direcionada a dois tipos de usuários: Engenheiros de Software e projetistas de interfaces e usuários finais, como mostra a Figura 4.6.

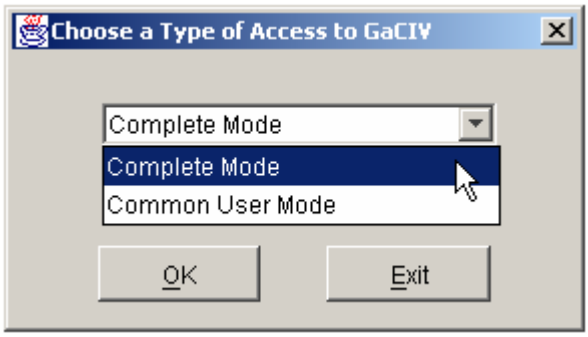

Figura 4.6 – Modos de acesso ao Ambiente GaCIV

O modo *Complete Mode* é dedicado aos dois primeiros tipos de usuário que têm total controle do GaCIV, o que permite acrescentar aplicações, comportamentos, componentes 3D, *links* de aplicação e criar e editar gabaritos e interfaces. Os últimos usuários têm acesso limitado ao

GaCIV através do modo *Common User Mode*, podendo criar gabaritos a partir da escolha de um ambiente e componentes 3D e/ou interfaces com a utilização de gabaritos já definidos.

Uma característica importante da terceira versão do GaCIV é a capacidade de cada componente apresentar um comportamento diferente, ao contrário das versões anteriores, em que um objeto 3D tinha um único comportamento (relacionado à manipulação e interação) e esse comportamento era reutilizado para todos os objetos 3D de um ambiente. Essa última versão permite criar componentes cujos comportamentos são similares aos encontrados em objetos do mundo real, facilitando a interação do usuário com o sistema, pois a torna mais natural e intuitiva.

4.3. O GaCIV no apoio a ESBC, ED e DBC

Como afirmado anteriormente, o GaCIV apóia o desenvolvimento sistêmico de interfaces com RV, já que para desenvolver interfaces com RV de qualidade, as etapas de levantamento de requisitos, especificação e projeto de interação com componentes 3D devem ser cuidadosamente implementadas para aumentar a usabilidade da interface com o usuário, embora leve em consideração aspectos de usabilidade. Nesse sentido, o desenvolvimento de componentes 3D deve apoiar a ESBC e fatores relevantes já apontados.

Uma característica dos componentes para interfaces WIMP é a capacidade que esses têm de serem reutilizados em vários projetos de interfaces, através de parametrização de seus atributos ou a personalização dos eventos do componente para a interface em desenvolvimento.

A Tabela 4.1 mostra alguns dos seus atributos e eventos do componente **Jbutton**, pertencente ao pacote de componentes SWING, responsável por prover componentes para construções de interfaces no Java. Através da configuração dos atributos **name** e **text** e escrevendo o evento **btn1_mouseclicked(MouseEvent e)**, função criada pelo **JBuilder**, dentro do evento **actionPerformed**, é possível o reuso desse componente em novos projetos de interfaces WIMP.

Tabela 4.1 – Alguns Atributos e Eventos do Componente JButton

Algumas Propriedades	
Propriedades	**Descrição**
Name	Nome do Componente
Text	Texto a ser inserido
border	Tipo de Borda
enabled	Ativa/Desativa componente
Alguns Eventos	
Eventos	**Descrição**
actionPerformed	Executa uma ação quando um evento do sistema ocorre
stateChanged	Executado quando um estado do componente foi modificado

Semelhantes aos componentes para interfaces WIMP, componentes 3D devem ser desenvolvidos pensando-se na capacidade de serem reutilizados em outras circunstâncias, mesmo em domínio diferente do projetado. Isso envolve a capacidade do engenheiro de software em analisar domínios de aplicação e encontrar atributos e eventos comuns para componentes 3D, que através de configuração ou personalização, permitam o reuso em outros domínios de software.

Para tanto, o processo de ESBC deve ser considerado como forma de maximizar a capacidade de reuso dos componentes 3D implementados. Assim, é possível especificar os seguintes conjuntos de passos no projeto de interfaces com RV através do apoio do ambiente GaCIV:

a. Durante o levantamento e análise dos requisitos do sistema, o usuário se reúne com a equipe de desenvolvimento para definição dos principais objetos virtuais que devem compor a interface, de acordo com as funcionalidades definidas para o sistema (Assis, 2000);

b. São levantados as funcionalidades e comportamentos dos objetos 3D necessários para o desenvolvimento de interfaces. Para cada objeto 3D os desenvolvedores de software vão procurar nos comportamentos cadastrados quais são aptos ao reuso. Se houver, os desenvolvedores analisam o comportamento com base na documentação anexada e avaliam se o comportamento passará por um processo de adaptação aos requisitos do sistema. Caso contrário, os desenvolvedores poderão reutilizá-los sem modificações, o que resulta no reuso do tipo caixa-preta (Prieto-Diaz, 1993).

c. Se não houver comportamentos adequados para reuso na interface com RV do sistema, os desenvolvedores realizam a análise de domínio, procurando determinar aspectos funcionais e interativos do componente 3D proposto, criando um comportamento a ser utilizado para formar o componente 3D. É necessário salientar aspectos de generalidade dos componentes de interface, que favorecerá seu reuso em novos projetos. Realizado o processo de análise, o comportamento passa para o projeto de domínio onde será integrado à arquitetura do GaCIV e implementado conforme o conjunto de interface do ambiente. O novo comportamento é cadastrado para ser atribuído a um objeto 3D, formando um componente 3D.

d. Se o ambiente e os objetos tridimensionais necessários ainda não estão disponíveis na ferramenta, são então criados por um designer gráfico e cadastrados no banco de dados do GaCIV. Os comportamentos criados para o sistema e aqueles que foram qualificados e/ou adaptados são atribuídos a um objeto 3D formando os componentes 3D;

e. A partir daí, o InterBuilder pode ser usado para criação do repositório de componentes 3D, o gabarito configurável. Isso é feito através da escolha do ambiente, seleção e posicionamento dos componentes 3D. Considerando que um gabarito ainda não existe para o domínio da aplicação em desenvolvimento, os componentes 3D são selecionados e posicionados no ambiente para compor o gabarito;

f. Após a composição do gabarito configurável, que se torna disponível para reuso em novos projetos cujo sistema tenha características semelhantes (Assis, 2001), a interface pode ser criada com base em um gabarito configurável, que representa um domínio de software. Para isso, o desenvolvedor seleciona o gabarito a ser utilizado e escolhe os componentes presentes neste, para compor a interface com RV. Os componentes 3D são inseridos na posição em que foram configurados no gabarito, mas o desenvolvedor pode modificar sua posição e orientação conforme desejado como também colocar quantas cópias do componente forem necessárias na interface com RV em desenvolvimento.

g. Feito isso, é possível atribuir uma aplicação a cada componente 3D da interface. A interface desenvolvida quando salva, pode ser reutilizada como modelo para outras interfaces no mesmo domínio de software, com a vantagem de ser modificada e reconfigurada utilizando os componentes 3D do gabarito que foi usado.

É válido salientar que o GaCIV permite a construção gradativa de interfaces com RV. Sendo assim, os componentes 3D podem ser acrescentados à interface e ao banco de dados conforme vão sendo desenvolvidos e as aplicações atribuídas aos *links* não precisam ser associadas no momento da criação da interface. A flexibilidade de construção de interfaces com RV proporcionado pelo GaCIV permite, através de prototipação, o desenvolvimento iterativo da interface em criação com um contínuo processo de avaliação da usabilidade da interface e dos componentes 3D pertencentes a ela.

4.4. Estruturação do Ambiente GaCIV

Como foi afirmado anteriormente, o ambiente GaCIV armazena quatro tipos de elementos: ambientes, comportamentos, componentes 3D e *links* de aplicação. *Links* de aplicação podem ser entendidos como vínculos entre uma aplicação a ser executada e componentes 3D. Os quatro elementos são armazenados em banco de dados através da opção **iStore** presente na ferramenta **InterBuilder**, como mostrado na Figura 4.3.

A construção e edição de gabaritos são feitas através da opção **iBuilder** do **InterBuilder**. O InterBuilder lê o conjunto de ambientes e componentes cadastrados e apresenta-os para o desenvolvedor. Quando um ambiente é escolhido, o InterBuilder permite a escolha de componentes 3D que farão parte do gabarito. Da mesma maneira, o InterBuilder permite a criação e edição de interfaces. Neste último caso, quando um gabarito é escolhido, o InterBuilder apresenta o ambiente e permite a escolha dos componentes 3D armazenados no gabarito. Quando um gabarito ou uma interface é gravado, o InterBuilder armazena no banco de dados do GaCIV, sendo possível seu reuso em outros projetos de interfaces.

O processo de criação de uma aplicação é realizado através da opção **iPackage** do **InterBuilder**, como representado na Figura 4.5. A criação da aplicação copia todos os arquivos necessários para um diretório cujo nome é o mesmo da interface. Nesse diretório são gravados: ambientes, componentes 3D com seus arquivos de configuração (arquivo declarativo com a configuração dos atributos do comportamento configurados conforme sua utilização), imagens 3D, aplicações, além dos seguintes arquivos relacionados:

- Arquivo *.gcv***:** arquivo com a descrição do cenário virtual;
- Arquivos *.class* que são parte do InterViwer, o visualizador de interfaces produzida pelos InterBuilder;
- Arquivo *viewer.bat* que contém a chamada para o InterViewer.

Assim constituídos, o diretório de aplicação pode ser transferido para a máquina do usuário com todos os itens necessários para executar o sistema.

Quando o usuário executa o arquivo *viewer.bat* o InterViewer é chamado. Ele lê as informações do cenário e coordena o carregamento dos componentes 3D e suas funcionalidades, apresentando a interface com RV. Dessa forma, o usuário navega sobre o ambiente e interage com os componentes 3D representados na interface que agem conforme os comportamentos programados. Quando o usuário clica sobre um componente 3D, o InterViewer verifica se esse tem um *link* associado, executando a aplicação vinculada.

A Figura 4.7 apresenta a estrutura do GaCIV descrita nesta seção.

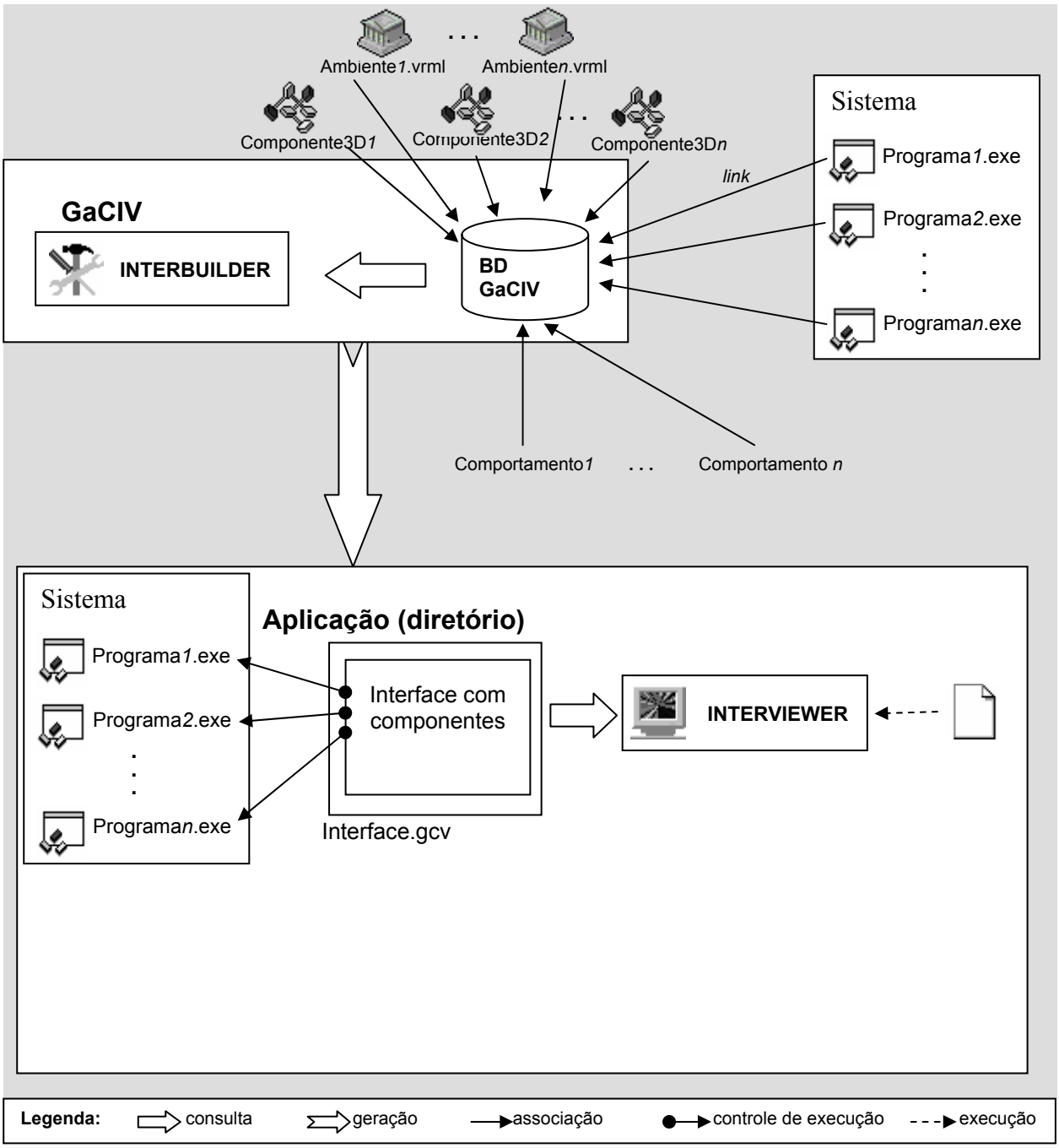

Figura 4.7 – Estrutura da terceira versão do GaCIV

4.5. Funcionalidades do Ambiente GaCIV

Esta seção apresenta os recursos e funcionalidades incluídas ao GaCIV para dar suporte ao reuso de componentes 3D. É utilizado o domínio de uma **locadora de vídeo** para exemplificar os recursos oferecidos pelo ambiente.

Como foi dito na seção 4.5, o ambiente GaCIV divide-se em 2 ferramentas: **InterBuider e InterViewer**.

Na barra de ferramentas do InterBuilder contém três opções principais: iStore, iBuilder e IPackage, conforme apresentado na Figura 4.2. Para gerar aplicações com RV são necessários cadastrar ambientes, comportamentos, componentes 3D e *links* de aplicação, realizado através da opção iStore do InterBuilder, conforme especificado na análise de requisitos do sistema e desenvolvido através da ESBC.

O cadastro de ambientes é realizado através da opção *Insert a new Environment* do iStore. Para efetivar o cadastro deve ser escolhido o arquivo de imagem 3D e inserido o nome do ambiente, que será incluído no banco de dados do GaCIV e disponibilizado através do seu nome, conforme a Figura 4.8.

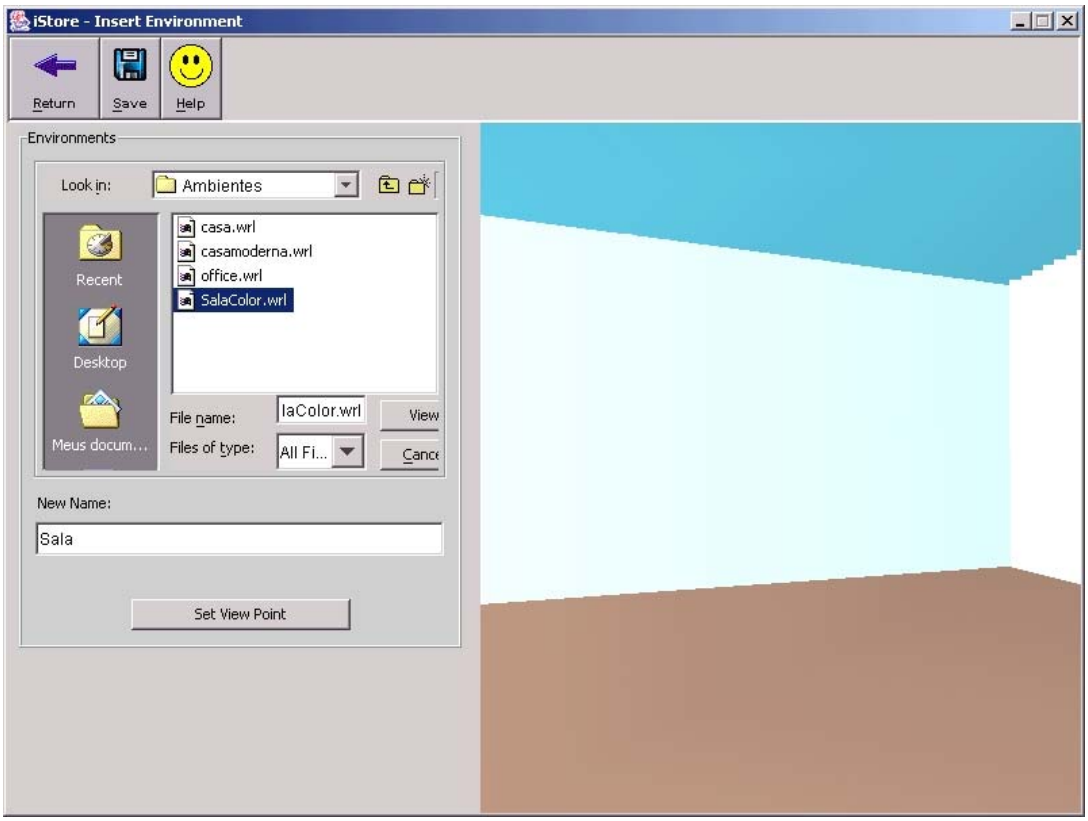

Figura 4.8 – Cadastrando um Ambiente

Realizado o cadastramento do ambiente, que representa o domínio da **locadora de vídeo**, o segundo passo é cadastrar o(s) comportamento(s) do(s) componente(s) desenvolvido(s), conforme a necessidade através da implementação do domínio de software, que corresponde ao DBC. Assim, os comportamentos ficam disponíveis para a atribuição de componentes 3D e ao reuso quando houver necessidade. A Figura 4.9 mostra partes da tela de cadastro de comportamentos no GaCIV, exibindo assim todos os passos necessários ao seu armazenamento. Esse processo é feito através da seleção da opção *Insert a new Behavior* do iStore. Para o cadastro é necessário escolher

a(s) classe(s) do comportamento, indicando em primeiro a classe principal, a qual o GaCIV carregará em tempo de execução para executar o comportamento. Ao escolher, o GaCIV mostrará a classe principal do comportamento sendo necessário incluir seu nome (Etapa b) e informações referentes a sua documentação (Etapa a), que incluem: **nome do comportamento**, **versão**, **domínios de softwares empregados**, **palavras-chaves**, **descrição** e **arquivo de documentação**. Depois basta pressionar o botão *Save* da barra de ferramentas para que o comportamento seja gravado no banco de dados.

É válido salientar que comportamentos podem ser reutilizados para um domínio de aplicação desde que estejam cadastrado no GaCIV e novos comportamentos podem ser cadastrados conforme vão sendo desenvolvidos.

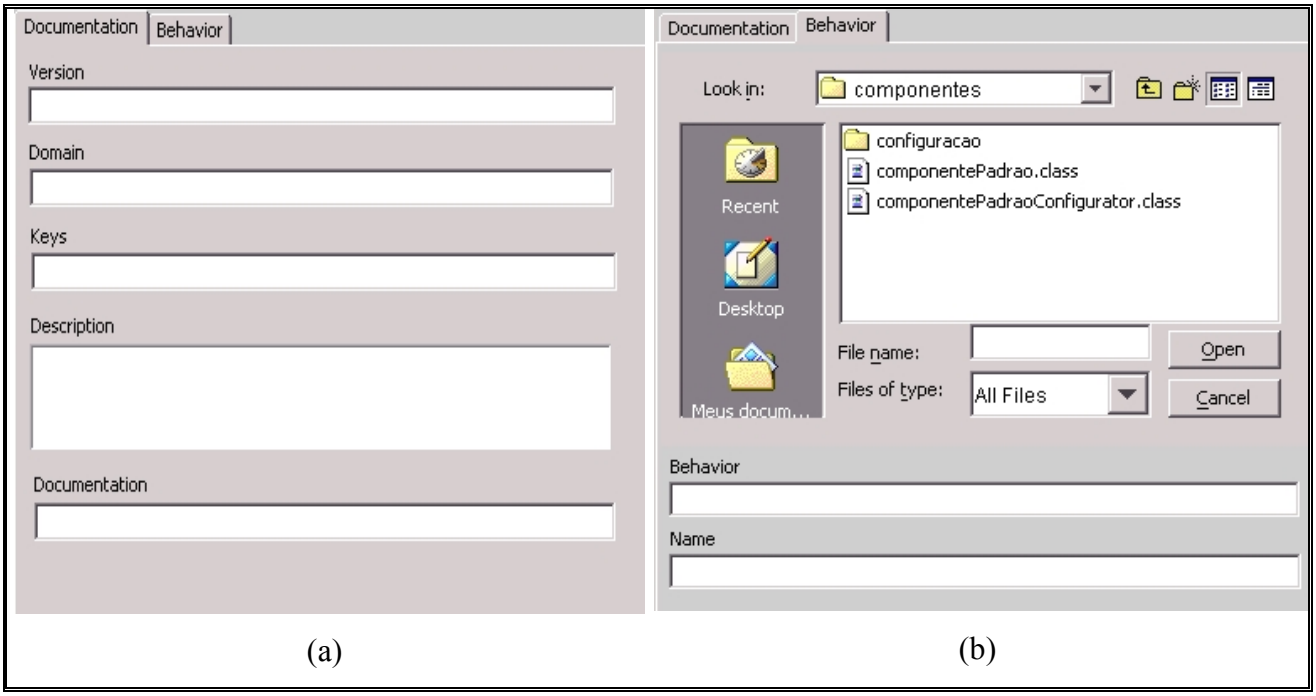

Figura 4.9 – Inserindo um Comportamento. a) Definindo a documentação, b)Escolhendo comportamento e seu Nome.

O cadastro de componentes é realizado através da opção *Insert a new Component* do iStore. Para efetivar seu cadastro é necessário que o comportamento a ser utilizado esteja disponível, sendo realizado através do conjunto de informações que devem ser incluídas, conforme a Figura 4.10. Para incluir componentes é necessário selecionar a **imagem 3D** e incluir o **nome do componente**, conforme mostra a etapa (a). Assim que o botão *Select* for pressionado a imagem 3D é visualizada na tela. É necessário escolher o comportamento do componente através da seleção dos comportamentos disponíveis e apresentados na tela, como mostra a etapa (b). Ao pressionar sobre um comportamento são exibidas sua versão e descrição, sendo possível acessar sua documentação

através do botão *View Documentation*. A etapa (c) da Figura 4.10 apresenta a tela de configuração do componente. Essa tela é feita pelo desenvolvedor do componente para configurar seus atributos e é carregada dinamicamente pelo Java quando um comportamento é selecionado. Esses atributos são variáveis para cada componente, mas devem ser preenchidos para o correto funcionamento do componente 3D. Além disso, são necessárias informações referentes à sua documentação, tais como: **domínio**, **palavras-chaves** e **descrição**, pois focalizam o componente 3D para um domínio de aplicação e junto à documentação do comportamento formam a documentação do componente 3D. Caso o componente 3D não seja associado a um comportamento, o GaCIV atribui um comportamento padrão ao componente 3D adicionado, que provê funcionalidades compatíveis com as versões anteriores do ambiente. Quando pressionado o botão *Save* da barra de ferramentas da tela de cadastro de componentes, o componente é gravado no banco de dados.

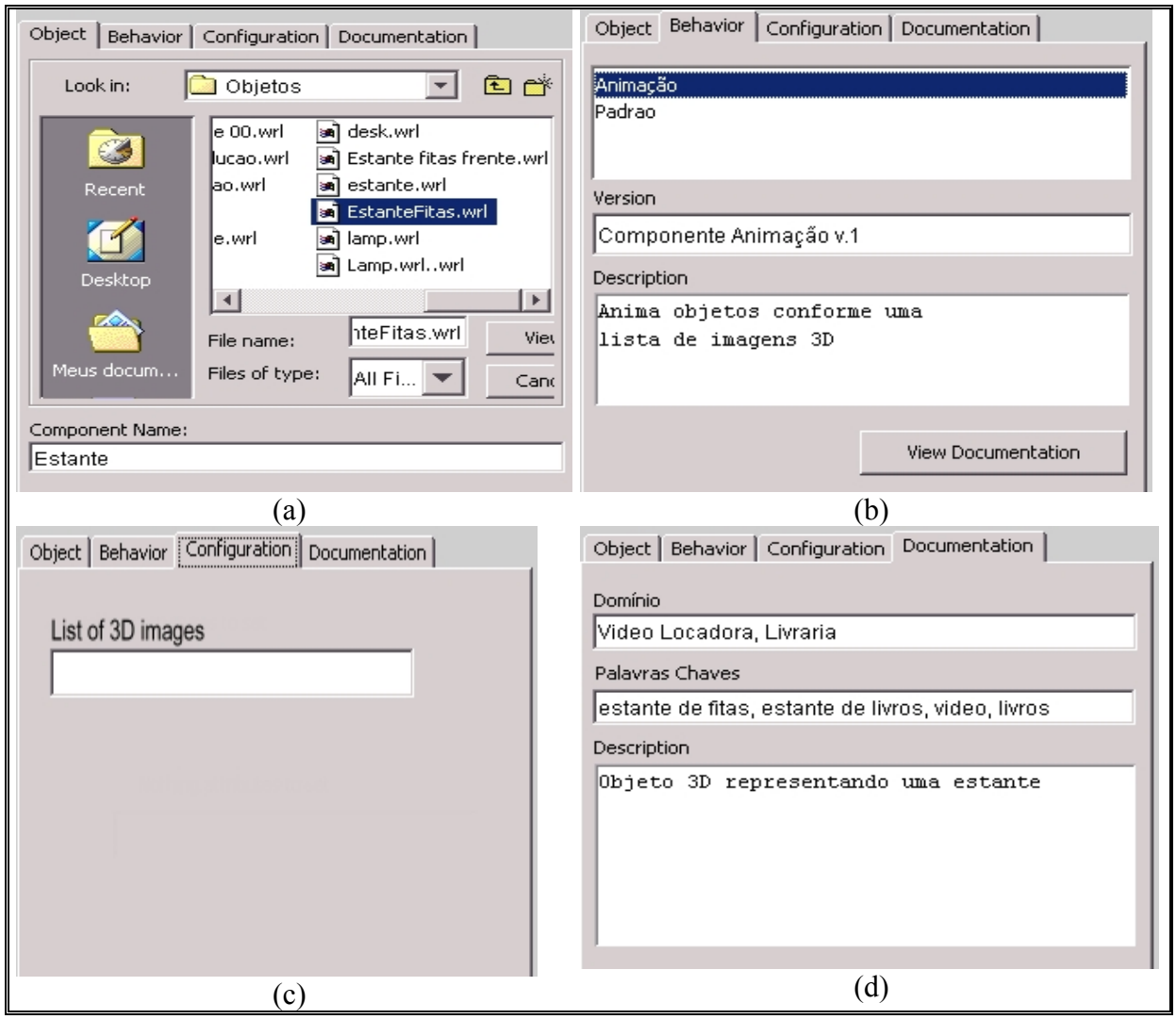

Figura 4.10 – Inserindo o Componente Estante. a) escolhendo a classe principal e atribuindo um nome, b) escolhendo um comportamento, c) tela de configuração do comportamento, d) incluindo informações da documentação.

Links de aplicação são adicionados através da Opção *Insert a new link* do iStore. *Links* de aplicação são associados aos componentes 3D e servem para executar aplicações quando o usuário clica sobre um componente 3D. Para cadastrar novas aplicações o GaCIV pede as seguintes informações: **arquivo de aplicação**, **nome da aplicação** a ser cadastrada e **descrição**. A aplicação e o tipo de arquivo selecionado (exemplo, *.exe, .class*) são apresentados na tela. Ao pressionar o botão *Insert,* o *link* de aplicação é cadastrado no GaCIV. A tela de cadastro de links de aplicação é apresentada na Figura 4.11.

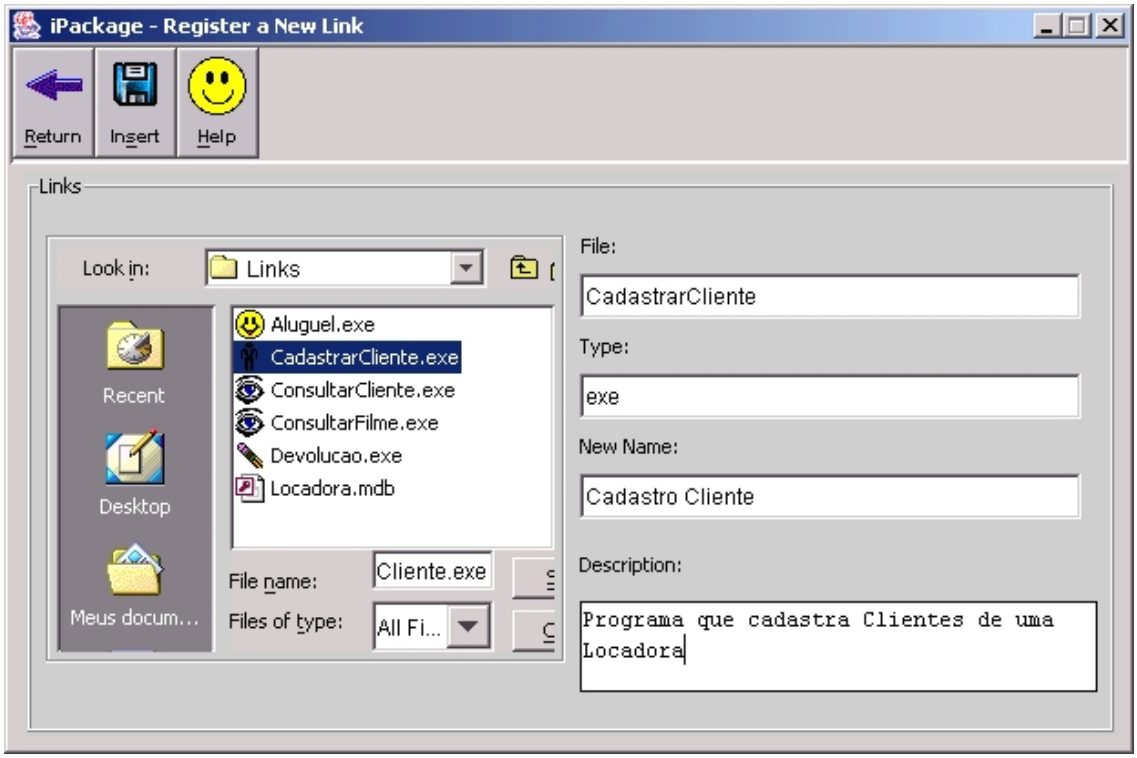

Figura 4.11 – Inserindo um *link*

Ao fim desse processo, é possível criar repositórios de componentes 3D para o domínio de aplicação em desenvolvimento, os gabaritos configuráveis. Tanto no caso de gabaritos e interfaces, a forma como o GaCIV carrega de componentes 3D e seus comportamentos são transparentes aos usuários e desenvolvedores, bastando sua seleção através das interfaces do GaCIV.

A Figura 4.12 contém as telas responsáveis pela construção de gabaritos, exibindo as etapas para sua criação. Na etapa (a), é escolhido um ambiente que o represente, nesse caso, a sala de uma locadora de vídeo. A escolha é feita entre os diversos ambientes cadastrados e apresentados para seleção. Na etapa (b), são escolhidos todos os componentes 3D que compõem o ambiente. A lista à esquerda exibe todos os componentes 3D cadastrados. Ao inserir um componente 3D

selecionado e apertar o botão *Insert*, o componente 3D é apresentado conforme a configuração do comportamento feito quando o componente 3D foi adicionado ao GaCIV. Os componentes 3D adicionados ao gabarito são exibidos na lista à direita e podem ser visualizados no *browser* da ferramenta. Conforme vão sendo incluídos, pode-se alterar a sua posição e orientação. Essa configuração é feita através da manipulação direta dos componentes 3D, sendo gravados conforme essa configuração. Caso o usuário ou o desenvolvedor decida retirar algum componente 3D do gabarito, é só selecioná-lo na lista à direita e pressionar o botão *Remove*.

Para efetivar o cadastro basta inserir seu nome, como mostra a etapa (c) da Figura 4.12. A Figura 4.13 mostra o gabarito construído. Conforme comentado na seção 4.3 deste capítulo, o gabarito pode ser alterado conforme novas necessidades e requisitos da aplicação.

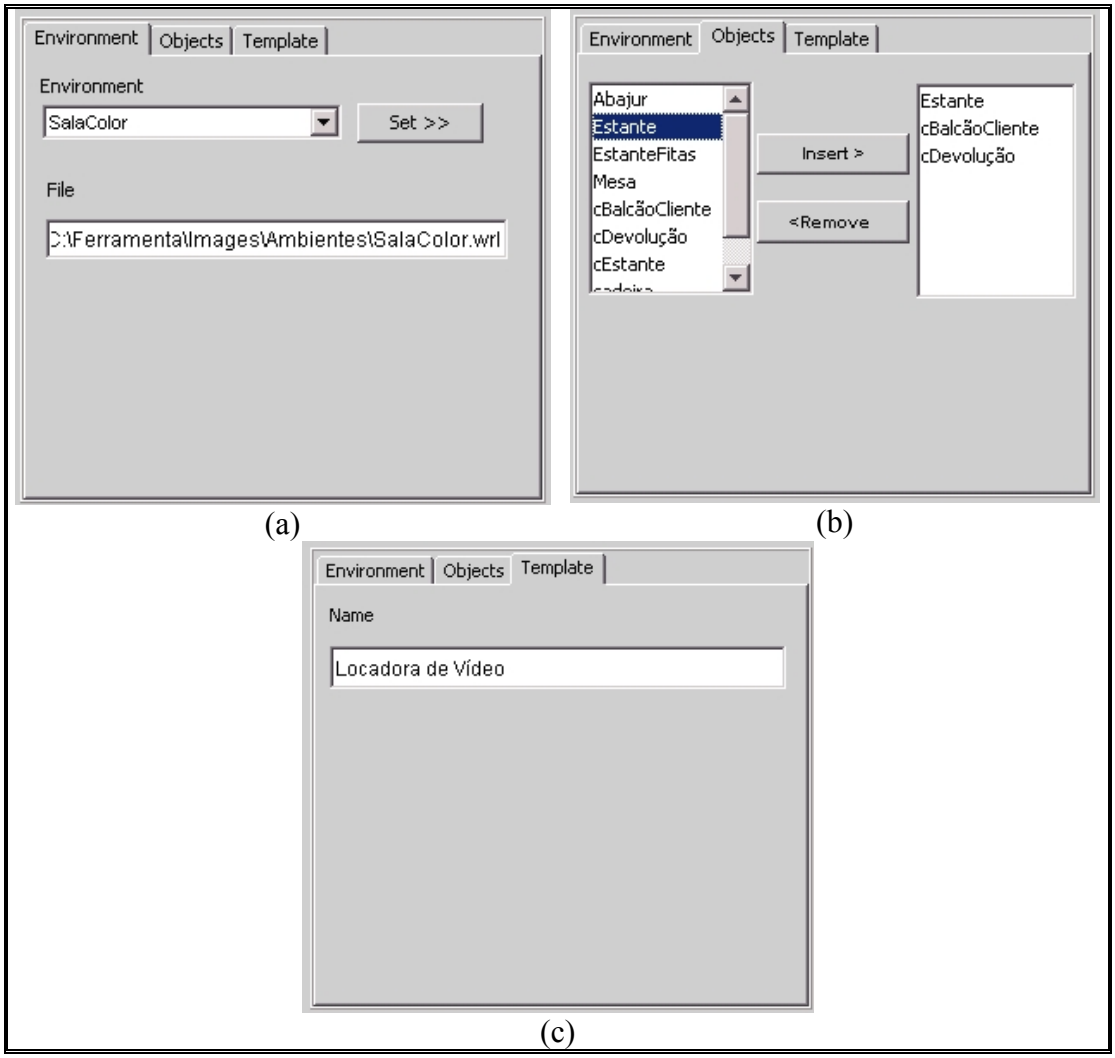

Figura 4.12 – Etapas da construção de um Gabarito. a) Escolha de um Ambiente, b) escolha de componentes 3D e c) atribuindo um nome ao gabarito.

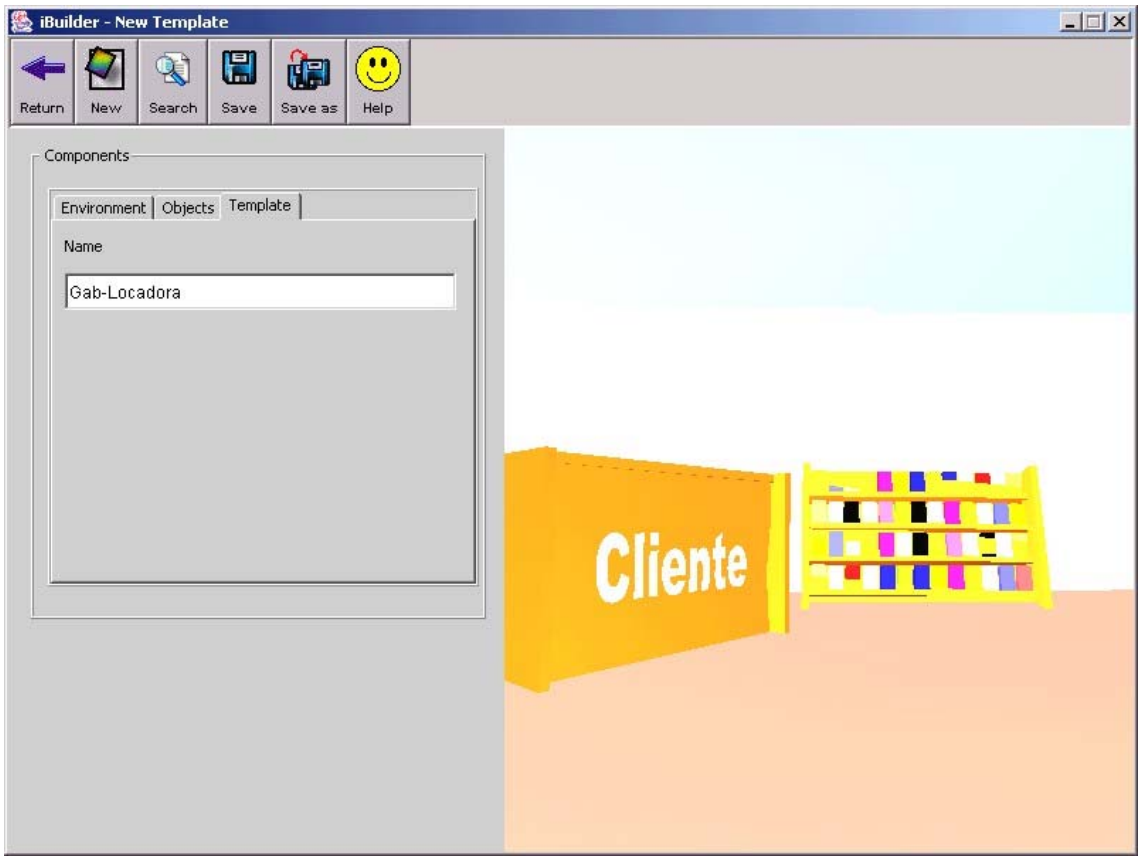

Figura 4.13 – O gabarito desenvolvido para o domínio de Vídeo Locadora

A partir de um gabarito é possível desenvolver uma interface. A Figura 4.14 mostra as telas responsáveis por essa criação, mostrando todas as etapas necessárias. Na etapa (a), é escolhido o gabarito utilizado para a construção de interfaces com RV. Após selecionar um gabarito e clicar sobre o botão *Open*, o gabarito é aberto e o ambiente aparece na tela do editor de interfaces, como mostra a Figura 4.14(b). Na etapa (c), são apresentados na lista à esquerda, componentes 3D pertencentes ao gabarito. Ao selecionar um componente 3D e apertar o botão *Insert*, o componente é inserido na lista à direita e apresentado na tela, onde foi previamente configurado no gabarito. Através de manipulação direta é possível modificar sua posição e rotação na tela como for necessário. Pode-se omitir componentes 3D do gabarito na interface como também colocar inúmeras cópias do componente 3D, conforme forem necessárias aos requisitos da aplicação. Se o desenvolvedor ou usuário quiser retirar o componente 3D da tela, basta selecioná-lo da lista à direita e pressionar o botão *Remove*.

Feito isso, a interface pode ser armazenada no GaCIV, adicionando um nome (etapa d) e pressionando o botão *Save* da tela de construção de interfaces. Assim, a interface é gravada e poderá ser reutilizada e modificada conforme as necessidades e novos requisitos da aplicação, como comentado na seção 4.3.

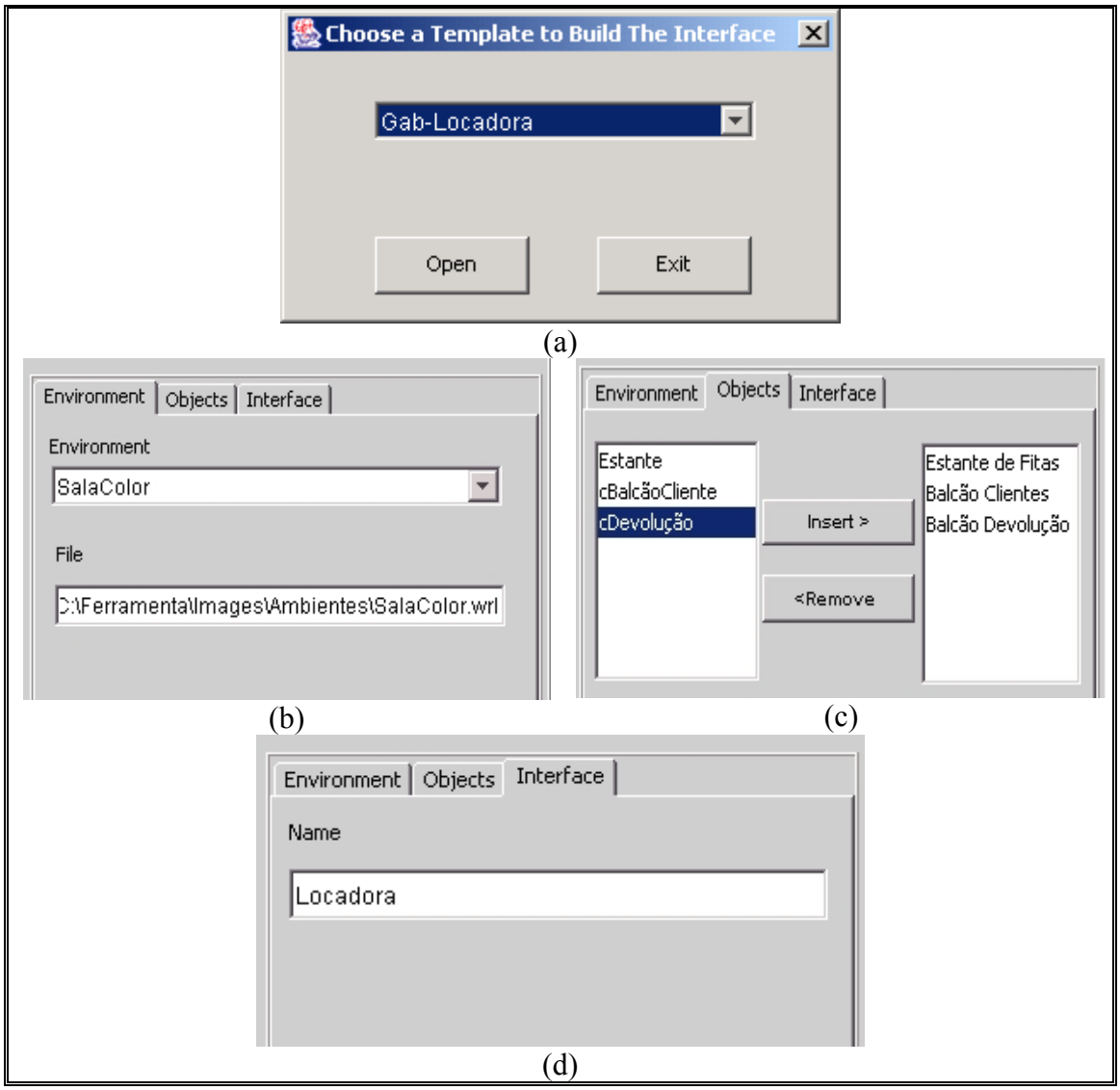

Figura 4.14 – Criação de uma Interface de Vídeo Locadora. a) Seleção de Gabarito, b) O ambiente do Gabarito já aberto, c) incluindo componentes 3D do gabarito, d) atribuindo um nome à interface.

Ao acrescentar um componente na interface ou tendo-se uma interface já armazenada, é possível associá-lo a um *link*, que é um vínculo entre um componente 3D e uma aplicação, quando o usuário estiver navegando na interface com RV através do InterViewer e pressionar o botão do mouse sobre um componente 3D, a aplicação ou interface vinculada é aberta na tela. A Figura 4.15 mostra a atribuição de *links* a um componente 3D. Para isso, basta selecionar o componente 3D da lista à direita e escolher o tipo de *link* que será associado: aplicação ou interface. Marcando uma dessas opções, o InterBuilder apresenta uma lista de aplicações ou interfaces que ao ser selecionada e apertar o botão *Set* é vinculada ao componente 3D selecionado. A Figura 4.16 mostra a tela da interface da locadora construída no InterBuilder.

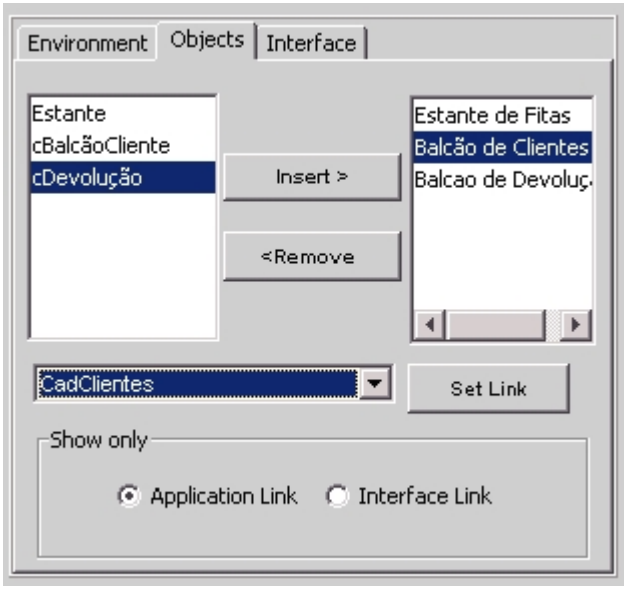

Figura 4.15 - Atribuição de *links* à Interface Locadora.

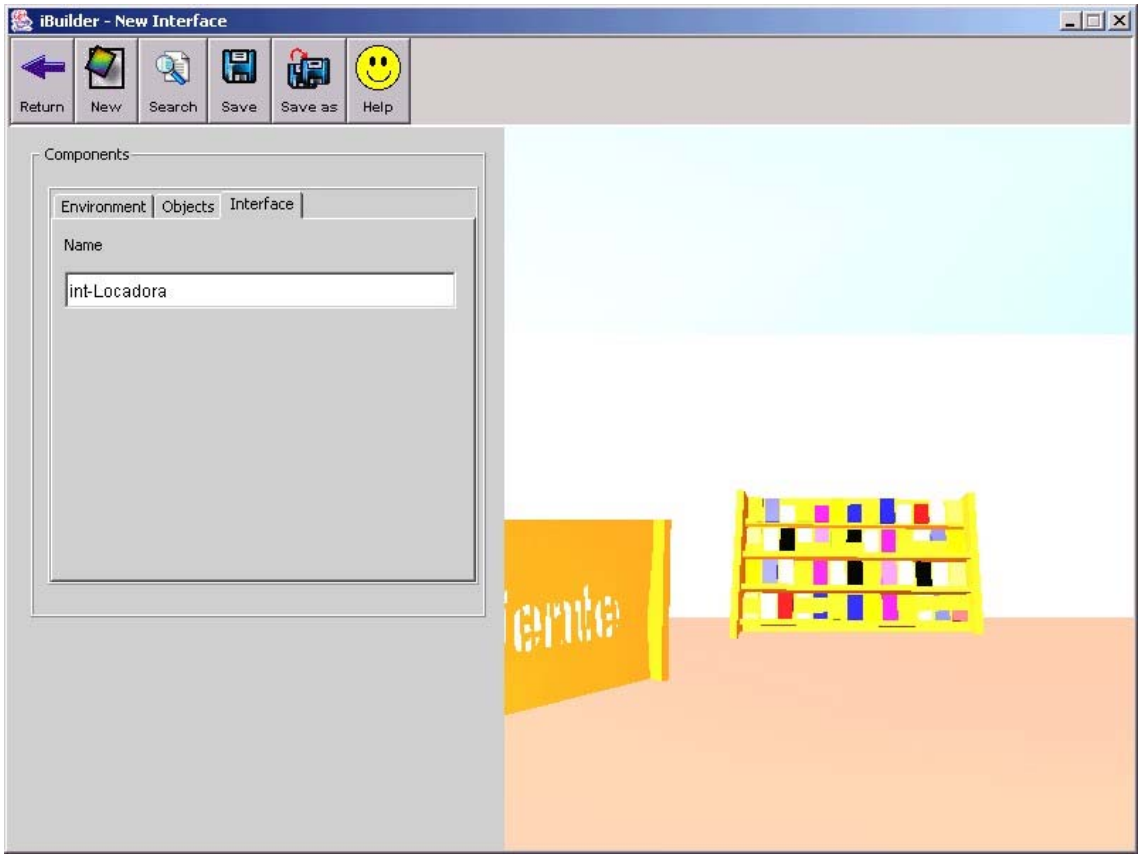

Figura 4.16 – A Interface da Vídeo Locadora

Para gerar a interface com as aplicações é necessário gerar o pacote de aplicação, através da opção **iPackage** do InterBuilder. Para isso, basta selecionar a interface e pressionar o botão Select. O pacote de aplicação é gerado em um diretório cujo nome é igual ao da interface, contendo todos os itens necessários a sua execução, como comentado na seção 4.3 deste capítulo.

Depois, é possível transferir o diretório para o computador do usuário e executar a aplicação com RV através do **InterViewer**, bastando executar o arquivo *iviewer.bat* no diretório do pacote de aplicação. A Figura 4.17 apresenta a tela de uma **locadora de vídeo** sendo executada no **InterViewer**. O usuário interage com a interface com RV através dos componentes 3D presentes nela.

Quando o usuário pressiona o botão do mouse sobre um componente 3D, o InterViewer verifica se o componente está vinculado a uma interface ou aplicação. Se estiver, o InterViewer abre outra janela com a aplicação ou interface vinculada. A Figura 4.18 apresenta a tela de **cadastro de clientes** que foi vinculado ao componente **balcão de clientes** da interface da **locadora de vídeo**.

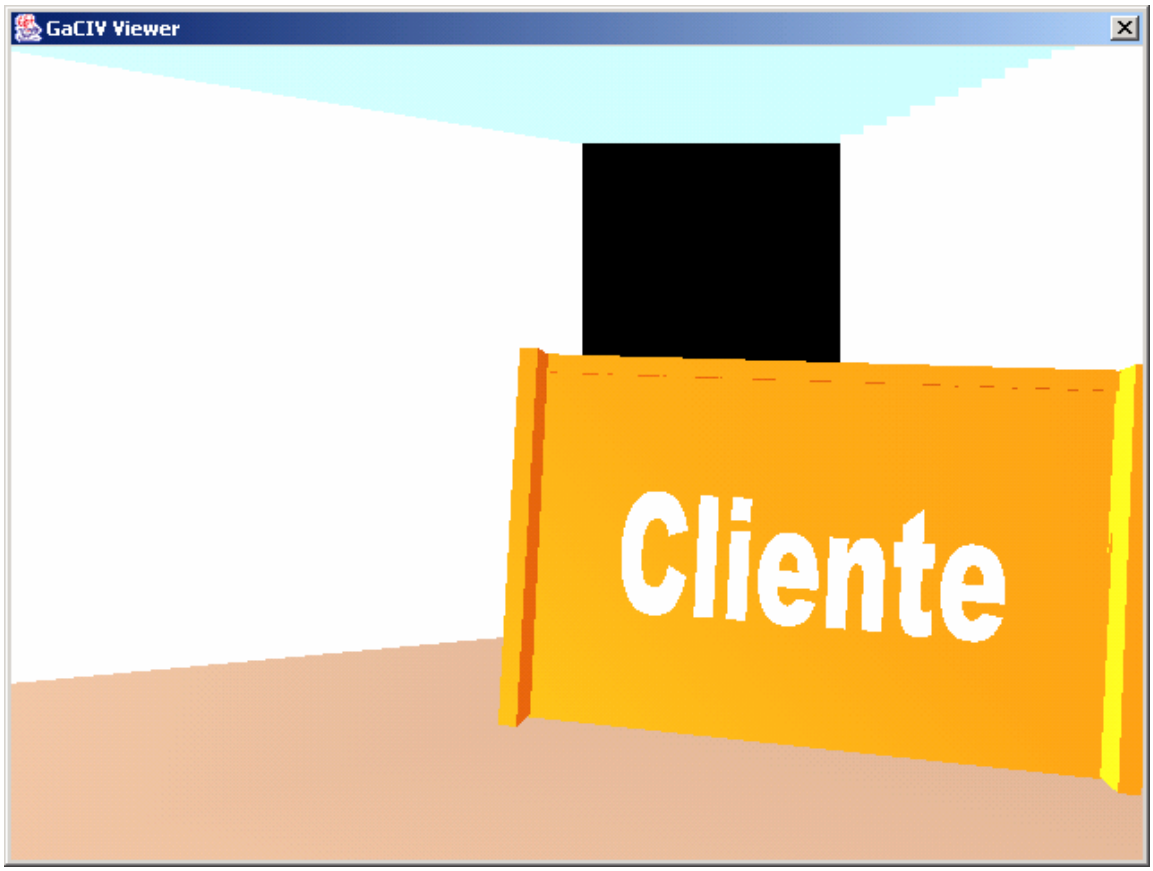

Figura 4.17 - Interface da locadora de vídeo no InterViewer

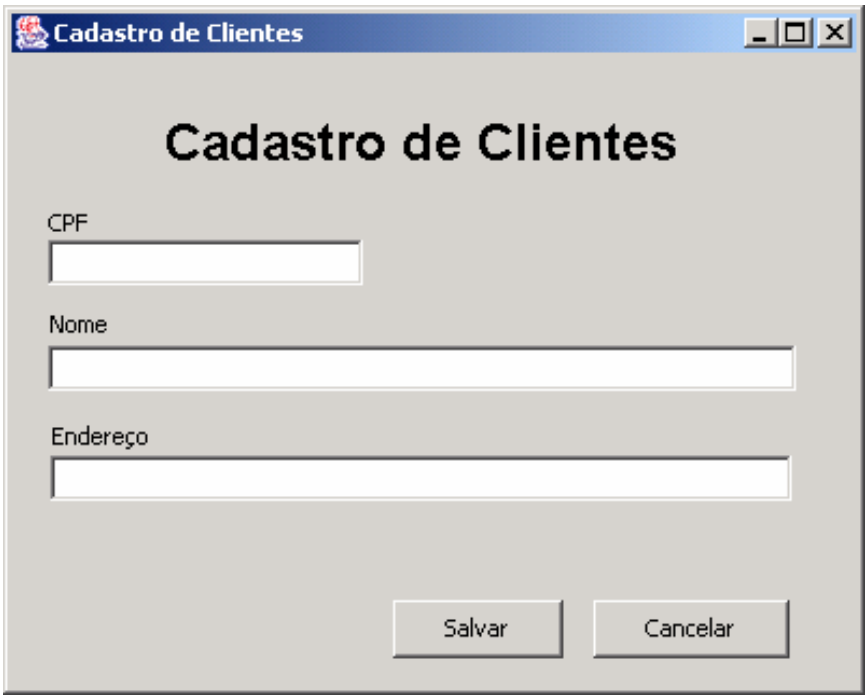

Figura 4.18 – Programa de Cadastro de Clientes

4.6. Desenvolvimento de um novo Comportamento para Componentes 3D no GaCIV

Para que seja possível o desenvolvimento de componentes 3D com novas funcionalidades, é necessário a implementação de novos comportamentos. Esta seção mostra o desenvolvimento de um comportamento de manipulação de componentes 3D no InterViewer. O intuito de desenvolvê-lo é adicionar a funcionalidade de manipulação direta de componentes 3D no InterViewer, visto que os componentes 3D tem sua posição e orientação definidas no InterBuilder quando uma interface é criada, e não é possível modifica-las quando visualizadas pelo *browser*. Portanto, este comportamento adiciona a capacidade de mover e rotacionar componentes 3D e pode ser utilizado para prover funcionalidade de "arrastar-soltar", típica de interfaces de manipulação direta.

Esse comportamento mostra também a capacidade do GaCIV implementar novos comportamentos reutilizando comportamentos implementados. Neste caso, através de herança.

4.6.1. Desenvolvimento do Comportamento

O comportamento desenvolvido é baseado no comportamento padrão, cujo objetivo é prover funcionalidade compatível com as versões anteriores do GaCIV e foi utilizado no exemplo da locadora de vídeo, apresentado na seção 4.5.

Para que um componente 3D que utiliza esse comportamento funcione corretamente no GaCIV, o comportamento deve implementar a interface do componente. Só assim, o GaCIV o reconhecerá como um componente 3D. O processo de desenvolvimento de componentes para o GaCIV deve seguir os passos determinados na seção 4.3, conforme o processo de ESBC.

Para isso, o comportamento em desenvolvimento precisa implementar os métodos correspondentes à interface do componente. Esses métodos são os seguintes: **isIBuilder**, **loadProperties**, **loadComponent** e **onMouseClick**. Além disso, são necessários os métodos da classe **<componente>Configurator.class** que são responsáveis por gravar atributos do componente, atribuídos durante o processo de cadastro do componente 3D. Os métodos dessa classe são: **saveConfiguration** e **saveBehavior**. Todos esses métodos são melhor discutidos no capítulo 5.

No caso dos métodos do componente, as funções **isIBuilder**, **loadProperties** e **loadComponent** devem passar por modificações. Em especial o método **isIBuilder**, cuja função é indicar onde o componente está sendo executado, ou seja, no InterBuilder ou no InterViewer, pois a funcionalidade de manipulação do componente 3D deve ser inserida somente no InterViewer, já que o InterBuilder tem suas próprias funções de manipulação e não devem ser sobrecarregadas. No caso dos métodos da classe de configuração, suas funcionalidades não diferem do comportamento padrão. Assim, a classe de configuração do componente reusa os métodos da classe herdada.

Além desses métodos correspondentes à interface do componente 3D e da classe de configuração, o componente de manipulação adiciona outros métodos. Esses métodos devem ser chamados entre si ou dentro dos métodos da interface do componente, pois o GaCIV executa somente os métodos definidos na interface do componente 3D e sua classe de configuração. Os métodos acrescentados são:

manipula	Adiciona o comportamento de manipulação de objetos 3D. É um nó *Behavior* na representação do grafo de cenário do Java3D.
salvaConfiguracao	Salva a posição e rotação do componente 3D manipulado, para persistir as modificações realizadas na interface.

Implementado o comportamento, ele é cadastrado no GaCIV como mostrado na Figura 4.9. Componentes 3D que utilizam esse comportamento podem ser adicionados, como foi apresentado na Figura 4.10 e inseridos em um gabarito para gerar interfaces com RV.

O suporte de Java e Java 3D permite ao GaCIV criar comportamentos referentes à: comportamentos dinâmicos ativos (cujas ações são realizadas independente da ação do usuário) e passivos (cujo comportamentos são realizados quando o usuário clica sobre o componente, por

exemplo), animações, efeitos mágicos, sonorização 3D, transparência, técnicas de *morph*, etc. Além disso, é possível desenvolver componentes de acesso a base de dados e comunicação com aplicações que utilizam interfaces WIMP.

4.7. Considerações Finais

Neste capítulo foram apresentados os recursos e funcionalidades do GaCIV. Foi discutida a construção desse tipo de interface levando em consideração os processos da Engenharia de Software e da ESBC e a capacidade do desenvolvedor de componentes de interface gerar componentes 3D que possam ser reutilizados em novos projetos. Uma característica interessante é a habilidade que o GaCIV tem de permitir não apenas o reuso de componentes para interfaces, mas de todo o conjunto de interfaces e gabaritos produzidos. Isso torna o GaCIV um ambiente de construção de interfaces com RV potencialmente prático e simples para os seus usuários.

A seguir são apresentados os resultados obtidos por esse trabalho e como foi desenvolvida a proposta de trabalho apresentada por Albertin (2002).

5. Análise dos Resultados Obtidos

5.1. Considerações iniciais

Este capítulo descreve as modificações realizadas no GaCIV para a realização da proposta de trabalho apresentada por Albertin (2002).

Recapitulando, a proposta de trabalho consistia em dois objetivos:

1. Definir os componentes 3D, suas interfaces e sua documentação, da seguinte maneira:
   a) Analisar partes do código do GaCIV a ponto de identificar as partes relacionadas à leitura de objetos 3D que são responsáveis pela parte geométrica de um componente 3D e as partes responsáveis por sua funcionalidade, manipulação e interação, parte essa correspondente ao comportamento de um componente 3D.
   b) Identificadas essas duas partes, elas deviam ser reunidas para formar um componente 3D;
   c) Realizados os dois primeiros itens, deveria ser definida uma interface dos componentes 3D para ser chamada pelo InterBuilder e pelo InterViewer.
   d) Definir informações referentes à documentação do componente;
2. Transformar gabaritos configuráveis em repositórios de componentes, através da inclusão de componentes 3D nos gabaritos e da definição de mecanismo de busca no repositório, tornando o GaCIV um gerador de repositórios.

Além desses dois objetivos, este trabalho se propõe a apresentar uma solução para os problemas identificados no capítulo 2.

Dessa forma, na seção 5.2 deste capítulo são apresentadas as propostas de solução para os problemas apresentada no capítulo 2; a subseção 5.2.1 apresenta a proposta de solução ao problema “Projetar comportamentos não semelhantes a objetos do mundo real”; a subseção 5.2.2 apresenta a proposta de solução ao problema “Embutir imagem 3D no código da aplicação”; a subseção 5.2.3 apresenta a proposta de solução ao problema “Utilizar ferramentas de prototipação não-extensíveis”; a subseção 5.2.4 apresenta a proposta de solução ao problema “Não definir ou não utilizar arquiteturas 3D para construção de aplicações com RV”; a seção 5.3 apresenta uma solução ao objetivo 1 apresentado acima; a seção 5.4 apresenta uma solução ao objetivo 2 também apresentado acima; por fim, a seção 5.5 apresenta as considerações finais.

5.2. Proposta de Solução para os Problemas Típicos nos Projetos de Sistemas com RV

Esta seção discute as propostas de solução para os problemas apontados no capítulo 2, que são:

1. Projetar comportamentos não semelhantes a objetos do mundo real;
2. Embutir imagem 3D no código da aplicação;
3. Utilizar ferramentas de prototipação não-extensíveis;
4. Não definir ou não utilizar arquiteturas 3D para construção de aplicações com RV.

Embora alguns desses problemas já tenham sido comentados nos capítulos anteriores, esta seção procura concentrar as soluções apresentadas em uma única seção, fornecendo informações mais precisas.

5.2.1. Proposta de solução para o problema "Projetar comportamentos não semelhantes a objetos do mundo real"

Como foi afirmado no capitulo 2, o GaCIV utiliza componentes 3D, que são uma agregação entre imagem(s) 3D e métodos cujas funcionalidades expressam a maneira como é feita a leitura de geometria (imagem 3D), manipulação e transformações geométricas, ou seja, definição do seu comportamento dentro de uma interface com RV. A utilização desse tipo de componente, permite ao GaCIV ter formas diferentes de interação para cada objeto 3D num cenário virtual.

É necessário o desenvolvimento de comportamentos semelhantes aos dos objetos do mundo real, pois usuários devem saber manipular os objetos 3D levando em consideração o conhecimento do funcionamento de objetos semelhantes no mundo real. Assim, é possível criar interfaces mais funcionais e intuitivas.

Com o uso de componente 3D, o GaCIV permite aos desenvolvedores criar novos comportamentos específicos de cada componente 3D, conforme os requisitos da aplicação. Esses componentes são armazenados no GaCIV e estão podem ser reutilizados para outros domínios de aplicação. O fato do GaCIV permitir adicionar componentes 3D com comportamentos próprios ajuda os desenvolvedores desenvolverem seus componentes a medida em que são necessários, sempre satisfazendo princípios de usabilidade.

5.2.2. Proposta de Solução para o problema "Embutir imagem 3D no código da aplicação"

No que diz respeito à separação entre geometria e parte comportamental do componente 3D, ou seja, embutir imagem 3D no código da aplicação, o GaCIV permite que haja separação entre a imagem 3D e o código da aplicação, possibilitando que um comportamento desenvolvido em JavaBeans seja associado a uma imagem 3D, formando assim um componente 3D. Assim, o GaCIV contém comportamentos armazenados em seu banco de dados, permitindo a busca e qualificação de comportamentos que melhor se adaptam às características da imagem 3D, que terá mecanismos de interação próprios, conforme os requisitos da aplicação.

Dessa forma, comportamentos mais genéricos podem ser reutilizados para várias imagens 3D e domínios diferentes. Já o componente 3D, como tem um objeto 3D associado, sua reutilização é aconselhável em domínios de software em que esse componente 3D pode ser utilizado, satisfazendo princípios de usabilidade e estética, para ambientar o usuário em um certo domínio de aplicação.

No entanto, cabe ao desenvolvedor avaliar as funcionalidades desse comportamento e determinar se ele está apto a formar um novo componente 3D, que satisfaz os requisitos funcionais de um domínio de aplicação. Caso contrário, é necessário adaptá-lo à nova necessidade, modificando seus métodos ou criando um novo componente.

É válido salientar que a separação de geometria e comportamento é difícil de ser realizada por completo, como destaca Dachselt (1999). Isso porque comportamentos complexos podem conter métodos que alteram uma parte de uma geometria, sendo que modificações nessa parte podem não alterar um objeto 3D como um todo.

As imagens 3D do formato VRML, por exemplo, podem conter nós para cada parte da imagem 3D. Nesse caso, através de um *parser* VRML é possível manipular só a imagem associado a esse nó, possibilitando criar comportamentos mais fiéis a objetos 3D.

O GaCIV permite desenvolver componentes 3D que possuem comportamentos complexos utilizando tanto imagens 3D separadas como *parsers*. Para isso, a documentação de um comportamento complexo deve incluir detalhes dos arquivos de imagens 3D, ajudando os projetistas gráficos na elaboração de imagens para esses comportamentos.

5.2.3. Proposta de Solução para o problema "Utilizar ferramentas de prototipação não-extensíveis"

Como citado anteriormente, ferramentas extensíveis são aquelas que permitem acrescentar novos comportamentos de componentes 3D sem a necessidade de recompilar o ambiente de criação e visualização de cenários virtuais. O GaCIV, sendo um ambiente que contém esse tipo de ferramenta para gerar cenários virtuais, supera esse problema através do carregamento dinâmico de componentes de software.

Os componentes são carregados e compilados no GaCIV através de carregamento dinâmico.

Como o Java considera sendo diferentes arquivos.**class** com nomes iguais, já que uma classe Java de mesmo nome pode pertencer a programas diferentes, é necessário que haja no GaCIV um gerenciador de componentes, com o objetivo de identificar classes já compiladas, otimizando o desempenho da ferramenta. A maneira pela qual o gerenciador de componentes do GaCIV trabalha é através de um texto referente à versão do componente e nome da classe, incluída na documentação do componente 3D.

5.2.4. Proposta de Solução para o problema "Não definir ou não utilizar arquiteturas 3D para construção de aplicações com RV"

O GaCIV utiliza uma arquitetura 3D para representar um cenário virtual, Java3D, que se baseia em grafo de cenário (SUN, 2003c) para representar uma interface 3D.. Grafos de cenários são uma espécie de árvore direcionada que contém nós onde cada nó possui uma funcionalidade diferente.

No caso do grafo de cenários do Java3D, um nó pode ser de quatro tipos:

BranchGroups: Nós básicos que podem conter a representação de um objeto 3D no contexto do Java3D ou organizar o grafo de cenário em sub-grafos.

TransformGroups: Nós que podem fazer transformações lineares em matrizes de pontos dos objetos 3D, modificando sua forma, rotação, escala, entre outros. Subdividem-se em 2 grupos: *OrderedGroup* (objetos são renderizados conforme uma ordem definida) e *SwithGroup* (objetos renderizados de acordo com um valor booleano).

Behaviors: Nós de comportamento associados a *TransformGroups,* armazenam comportamentos relativos a um nó do grafo de cenário, tais como manipulação, animação (*morph*), tratamento de colisão, controle de dispositivos de entrada/saída.

SceneGroups: Associados a um *BranchGroup*, contêm a representação em Java3D dos objetos em terceira dimensão, seus pontos e vértices, materiais (luzes, cor, etc) e texturas.

Dessa forma, um cenário virtual é construído conforme esses tipos de nós e é associado a um ponto de visão, que representa um cenário 3D. A capacidade de interação, imersão e envolvimento permite criar uma representação de interface com RV no conceito do Java3D. Assim, a partir de grafos de cenários é possível criar estruturas complexas de cenário virtual com comportamentos próprios para cada nó do grafo.

Pelo motivo do GaCIV utilizar componentes de software, o JavaBeans foi escolhido, pois é a arquitetura padrão de desenvolvimento de componentes para a plataforma cliente do Java. A união Java3D e JavaBeans, dá ao GaCIV as seguintes características:

- Descobrir classes e interfaces em tempo de execução;
- Não necessidade de recompilar o GaCIV cada vez que adiciona um componente;
- Suporte a eventos e troca de mensagens com o GaCIV e outros componentes;
- Permissão para customizar propriedades do componente durante a fase de projeto de interfaces com RV, através de editores de propriedades;

Como o GaCIV é implementado através do uso de uma arquitetura padrão de desenvolvimento de software, os componentes 3D são implementados tendo uma representação binária, que é o próprio componente *JavaBean*. Embora o GaCIV tenha as descrições da interface com RV representadas da forma de arquivos XML e a configuração do componente também ser através de documentos declarativos, o GaCIV não pode ser considerado como baseado em documentos declarativos (Dachselt, 2003) sendo, então, baseado em uma arquitetura padrão. O fato do GaCIV não ser implementado totalmente com base em documentos declarativos é devido à dificuldade de encontrar objetos 3D no formato X3D (Rudolf, 1999), pois são raros os editores de imagens 3D que dão suporte à geração de arquivos para esse formato. As especificações de documentos declarativos, como a definição de interação (Figueroa et al, 1999) e comportamento 3D (Contigra, 2003), entre outras, consideram a utilização desse formato, além de serem pesquisas em desenvolvimento cuja maioria das especificações não é liberada para implementação. Mas o GaCIV tem suporte a vários tipos de formatos de imagens 3D que facilitam encontra-las ou desenvolvê-las através de editores de imagens 3D, isso torna o GaCIV mais independente de editores de imagens, bem como a capacidade de reuso proporcionada pelo JavaBeans e o suporte a essa arquitetura permite trabalhar com ferramentas CASE e gerar componentes JavaBeans.

5.3. Componentes 3D, sua Documentação e suas Interfaces no GaCIV

Os componentes 3D do GaCIV são compostos a partir da separação do objeto 3D e do seu comportamento, como foi apresentado na seção 4.2. Retomando a definição de componentes de software apresentada na seção 3.1, cada componente 3D do GaCIV tem uma função específica sendo realizada por esse em uma interface com RV, sendo que esse componente é identificado pelo GaCIV de maneira única, já que o GaCIV possui um gerenciador de componentes cuja proposta é gerenciar os componentes carregados dinamicamente, executando suas funcionalidades. A documentação do componente 3D é formada em conjunto com a documentação do comportamento. A primeira consiste em informações sobre: **imagem 3D, comportamento, domínio, palavras-chaves, descrição do componente e configuração** (arquivo XML gerado através da tela de configuração de um componente, como apresentado na Figura 4.10(c), que contém atributos do comportamento a serem configurados pelo desenvolvedor quando utiliza o componente). Essa tela, como foi comentado na seção 4.5, é carregada dinamicamente pelo GaCIV, quando um comportamento é selecionado. Assim, cada componente tem sua própria tela de configuração, o que lhe dá flexibilidade, pois pode-se acrescentar informações adicionais sobre: arquivos, banco de dados, etc, utilizadas pelo comportamento e desenvolvidas conforme requisitos da aplicação. A segunda documentação consiste nas informações sobre: **classe principal do componente *JavaBean***, **nome do comportamento**, **versão**, **descrição do componente**, **domínios de software onde podem ser empregados**, **palavras-chaves dos domínios e documentação do componente**.

O fato do componente 3D também ter informações referentes ao domínio, bem como palavras-chaves e descrição do componente, é devido ao componente 3D ser personalizado para domínios de software mais restritos ao comportamento, já que tem um objeto 3D associado e está inserido dentro do domínio para o qual foi designado. Já o comportamento pode ser genérico, sendo utilizado para vários domínios de aplicação.

Para que seja possível carregar e executar métodos, é necessário que o componente implemente a sua interface, que é chamada dentro do ambiente GaCIV. Essa interface contém métodos bem definidos quanto às suas funções em um componente 3D. Componentes 3D do GaCIV devem implementar os seguintes métodos:

loadProperties Responsável por ler e atribuir os valores dos atributos referentes aos itens de cada informação do arquivo de configuração do componente, que é um arquivo*.xml*. O método pelo qual um componente lê o arquivo XML não está definido pelo GaCIV. Dessa forma, o desenvolvedor tem

a liberdade de escolher como o componente 3D lerá o seu arquivo de configuração, podendo-se utilizar qualquer *parser* XML para isso, por exemplo, SAX *(Simple API for Xml)* (W3C, 2003a), DOM *(Document Object Model)* (W3C, 2003b), entre outros. O arquivo de configuração do componente é criado pela classe de configuração do componente, carregada em tempo de execução e que tem como nome:

<nome_do_componente>Configurator.class

O fato de não utilizar formas padrão de configuração de atributos do JavaBeans se explica pela necessidade que o GaCIV tem de gravar a configuração do componente para posteriormente ser lida por este método dentro do InterViewer, carregando os atributos com os valores configurados.

loadComponent Método cuja função é criar a representação geométrica e comportamental de um componente 3D em um formato aceitável pelo Java3D. É através dele que o desenvolvedor deve criar métodos que lêem e manipulam imagens 3D e definem a interação com o componente 3D na interface com RV. Nesse método também devem ser especificados controle de dispositivos de entrada/saída, tratamento de colisão, luzes e sombras, etc, caso haja necessidade.

onMouseClick Método chamado exclusivamente no InterViewer, responsável por executar uma ação quando o usuário pressiona botão do mouse sobre o componente 3D.

isIBuilder Método que retorna onde o componente está sendo executado: InterBuilder ou InterViewer. É utilizado pelo componente para executar métodos dentro das ferramentas apropriadas.

A classe de configuração do componente é um componente *Swing* (conjunto de classe responsável por fazer interfaces gráficas no Java) do tipo *JPanel*, que auxilia o desenvolvedor na configuração dos atributos do componente, utilizando a maneira declarativa de desenvolvimento de componentes 3D (Dachselt, 2001). Ela faz a persistência dos atributos do componente utilizando uma notação XML para serem, posteriormente, lidos pelo iViewer, através do método **loadProperties** do componente. Essa classe deve conter os seguintes métodos:

saveConfiguration	Responsável por gravar os atributos de um componente em um arquivo XML.
saveBehavior	Responsável por gravar outros arquivos, atribuídos através da configuração do componente, para gerar todos os itens necessários ao seu correto funcionamento, quando uma aplicação é gerada no GaCIV.

Além disso, o GaCIV permite ao desenvolvedor saber em quantos e em quais projetos um comportamento e/ou componente 3D foi reutilizado, como especificado por Sametinger (Sametinger, 1997). Esse controle é realizado internamente pelo GaCIV, sendo transparente aos desenvolvedores. Entretanto, a informação é disponibilizada através da etapa de qualificação de componentes 3D e comportamentos aptos ao reuso, que o GaCIV disponibiliza através de busca e seleção em repositórios. Isso ajuda desenvolvedores a avaliar o reuso de um componente 3D e/ou comportamento em uma aplicação, pois é provável que esses tenham passado por avaliações na usabilidade nos gabaritos e nas interfaces e, portanto, podem ser reutilizados em domínios semelhantes.

5.4. Gabaritos x Repositórios de Componentes 3D e mecanismos de busca em repositórios

Como comentado na seção 4.4, a proposta deste trabalho é transformar o GaCIV em um gerador de repositório de componentes 3D. Assim, um gabarito configurável passa a ser formado por um conjunto de componentes 3D com suas devidas funcionalidades, além de um ambiente que representa o cenário virtual da aplicação.

Tanto no cadastro de comportamentos, cadastro de componentes, criação de gabaritos e interfaces, o GaCIV dispõe de uma opção de busca de componentes 3D e/ou comportamentos. Seu propósito é ajudar desenvolvedores na busca e qualificação de comportamentos e componentes 3D aptos ao reuso.

Na Figura 5.1 é apresentada a tela de busca de componentes 3D. A maneira como está organizado permite encontrar componentes 3D com seus devidos comportamentos através de duas opções: domínio(s) e palavra(s)-chave(s) de um domínio de aplicação. Se a busca for realizada durante a construção de gabaritos, ela é feita para todos os componentes cadastrados, no caso do desenvolvimento de uma interface, a procura é realizada dentro do gabarito utilizado.

Assim, é apresentada uma lista de comportamentos/componentes encontrados através do parâmetro de busca escolhido e, ao ser selecionado, permite acessar as informações referentes a sua documentação, como mostra a Figura 5.2.

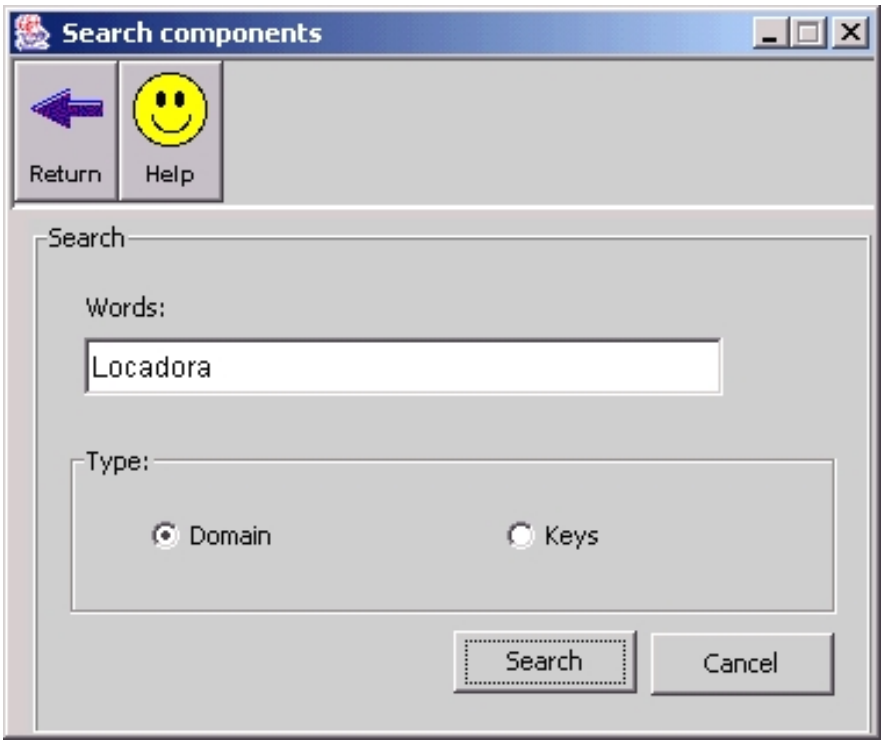

Figura 5.1 – Realizando uma busca de componentes 3D

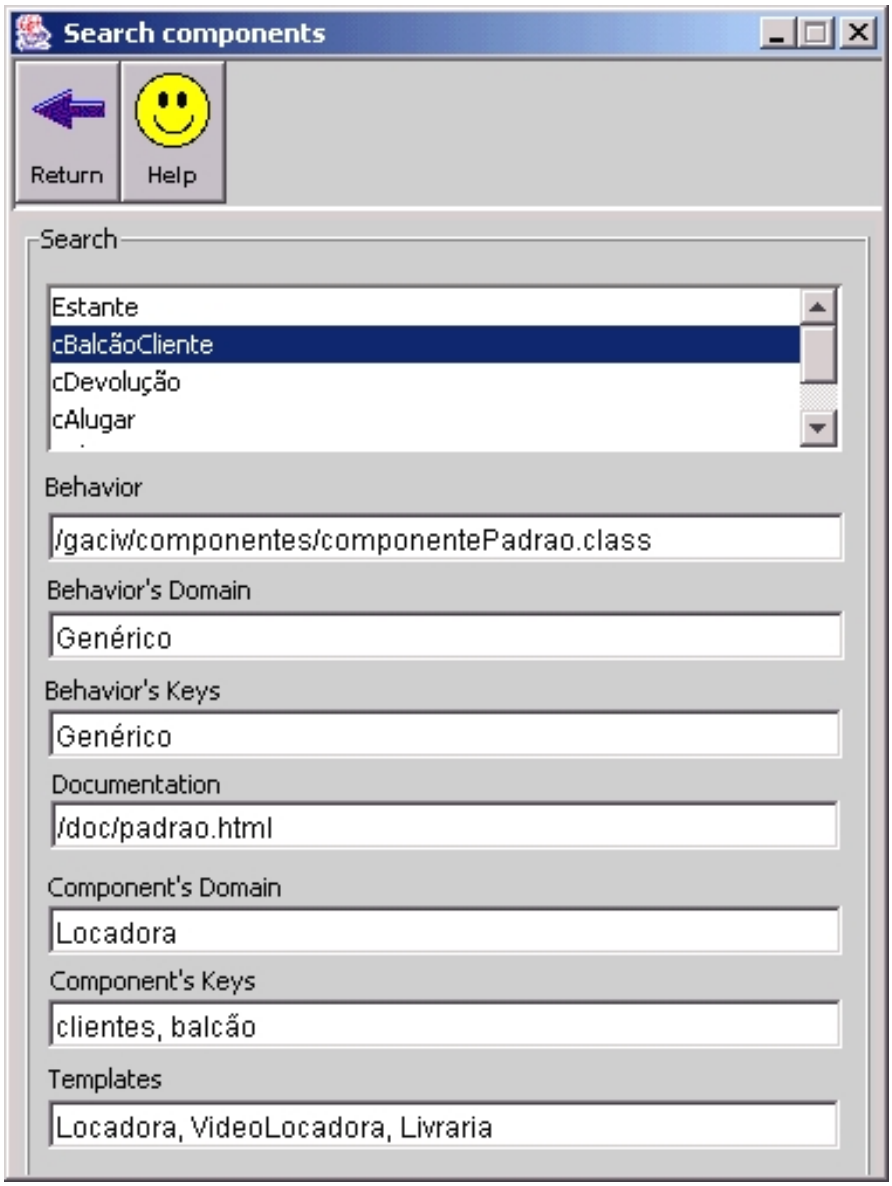

Figura 5.2 – Conjunto de resultados da busca realizada

Nesse caso, as informações contêm dados do comportamento utilizado e do componente 3D e, ainda, informação referente sobre quais gabaritos esse componente está sendo utilizado, que ajudam o desenvolvedor saber que tipos de domínio os componentes podem ser reutilizados, já que supõem-se que o reuso de um componente em um dado domínio tenha passado por avaliações de usabilidade.

O componente qualificado para reuso pode então ser selecionado para avaliação de suas funcionalidades ou inserido em um gabarito ou uma interface, como foi descrito no capítulo 4.

5.5. Considerações Finais

Neste capítulo foram apresentadas as propostas de solução para os problemas no desenvolvimento de interfaces com RV e discutido como a modificação foi realizada para permitir o reuso de componentes 3D em interfaces com RV produzida pelo GaCIV.

No próximo capítulo são apresentadas as conclusões deste trabalho.

6. Conclusões

6.1. Considerações iniciais

Este capítulo sintetiza os resultados obtidos por este trabalho, suas contribuições para o estudo e desenvolvimento de interfaces com RV através de reuso de componentes, dando margem à continuação em trabalhos futuros, solidificando os conceitos apresentados e incrementando o GaCIV para novas funcionalidades.

Considera-se concluído com sucesso o trabalho proposto, tendo definido e implementado todas as modificações levantadas na proposta de trabalho apresentada por Albertin (2002), considerando os problemas típicos do desenvolvimento de aplicações com RV mencionados neste trabalho.

Assim, a seção 6.2 traz as principais contribuições deste trabalho; a seção 6.3 apresenta as dificuldades a serem superadas; a subseção 6.3.1 discute as dificuldades em relação à implementação; a subseção 6.3.2 discute as dificuldades relacionadas à usabilidade; Por fim, a seção 6.4 discute os trabalhos a serem desenvolvidos, baseado no que foi apresentado durante o desenvolvimento deste trabalho.

6.2. Principais Contribuições

Interfaces com RV permitem superar diversos desafios, relacionados a uma maior facilidade dos usuários em interagir com o sistema, pois podem ser mais naturais e intuitivas. A capacidade dos usuários utilizarem modelos mentais para transferir conhecimento do mundo real para o mundo virtual os ajuda na interação com o sistema e sua utilização com um menor tempo de aprendizado. Tendo alcançado os objetivos propostos, ou seja, a modificação no GaCIV para o apoio ao reuso de componente para interfaces com RV e uma solução para os problemas típicos do projeto de sistemas com RV, o GaCIV colabora com a área de Interação Humano-Computador (IHC) fornecendo um ambiente propício para a criação de sistemas com RV fáceis de serem construídos, utilizando reuso de componentes 3D.

Dessa forma, este trabalho busca através do esforço acadêmico ajudar a pesquisa e desenvolvimento de interfaces com RV através de ferramentas de criação de ambientes virtuais, que escondem a complexidade de programação desse tipo de sistema, mas permitem seu projeto e

implementação de forma prática e acessível aos diversos usuários e desenvolvedores. Assim, os principais resultados obtidos por este trabalho são:

- A modificação da estrutura do GaCIV para suporte a componentes 3D;
- A transformação do GaCIV em um gerador de repositórios de componentes 3D para diferentes domínios de aplicação;
- O desenvolvimento de um ambiente de fácil manuseio, que permite a elaboração de interfaces com RV, escondendo sua complexidade de desenvolvimento;
- A proposta de criação de uma ferramenta extensível, que aceita novos componentes 3D com suas funcionalidades e adicionados ao ambiente sem a necessidade de recompilação;
- A proposta de solução aos problemas encontrados no desenvolvimento de aplicações com RV, tais como: a separação do objeto 3D do código do componente 3D, a possibilidade de criar novos componentes cujos comportamento são similares aos encontrados no mundo real e a utilização de uma arquitetura 3D baseada em grafo de cenário, para representar mundos virtuais complexos e detalhados;
- A seguinte submissão:
    - Albertin, J. C. L., Balbino; F. C.; Silva, J. C. A.; Penteado, R. A. D.; *Desenvolvimento de Sistemas com Realidade Virtual Baseadas em Reuso de Componentes*, Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas, Manaus, Amazônia, Brasil, 2003.

6.3. Dificuldades a serem Superadas

O desenvolvimento deste trabalho acumulou experiências em seu uso para determinar situações em que o GaCIV precisa de melhorias. Elas estão relacionadas a dificuldade de implementação e aos problemas de usabilidade.

6.3.1. Dificuldades Relacionadas à Implementação

Uma dificuldade de implementação foi o processo de reengenharia da segunda versão para a versão implementada com Java e Java3D. A interface gráfica 2D implementada é formada por componentes para interface do tipo SWING, classificados pela Sun (2003d) de componentes leves por estarem totalmente implementados em Java, não sendo utilizados recursos de

implementação de interface com usuário do sistema operacional. Porém, os recursos do Java3D implementam componentes para interfaces do tipo AWT, que por utilizarem recursos próprios do sistema operacional, são designados como componentes pesados. Boa parte dos problemas relacionados à implementação no GaCIV é relacionada à integração AWT-Swing, cujos problemas mais comuns são sobreposição de componentes Swing com os do tipo AWT, pois deixam rastros na tela, devido a velocidade de renderização ser diferente.

Assim, telas carregadas dinamicamente, como a tela responsável por configurar o componente, devem ser cuidadosamente implementadas para uma melhor integração entre os dois tipos componentes para interface, através do uso de *layouts* (forma de organização de componentes 3D utilizadas pelo Java) compatíveis entre esses dois tipos.

Outro ponto a considerar é que o GaCIV configura os componentes no ato do seu cadastramento. É aconselhável que o componente 3D possa mudar sua configuração quando utilizado, através do processo de criação de gabaritos e interfaces. Assim, devem existir meios que facilitem a comunicação entre componentes a partir do InterBuilder, visto que a comunicação entre componentes pode ser programada a partir do comportamento, mas reduz sua reusabilidade por causa dos objetos 3D controlados pelo comportamento terem que ser pré-configurados, via código. Essa mudança requer o re-estudo das funcionalidades permitidas aos desenvolvedores e aos usuários finais, já que o GaCIV permite que o desenvolvimento de interfaces com RV seja realizado por esses tipos de usuários, mas a configuração e, possivelmente, escrita de código relativos aos eventos do componente, podem dificultar seu uso por usuários finais.

Além disso, o GaCIV pode ser melhor estudado para fornecer um conjunto de métodos aos componentes 3D, facilitando a comunicação entre componentes e entre o InterBuilder e/ou InterViewer. Dessa forma, um componente 3D pode ter controle maior em relação às suas ferramentas, modificando seu funcionamento.

Por fim, a utilização de apenas um comportamento associado a uma imagem 3D levou a percepção da necessidade de um objeto ser formado por um conjunto de comportamentos. Assim, um componente 3D poderia ser formado por comportamentos que definem uma simples funcionalidade mas que associado a outros, formariam um comportamento mais complexo.

6.3.2. Dificuldades Relacionadas à Usabilidade no uso da ferramenta GaCIV

Em relação à usabilidade do GaCIV pode-se considerar as seguintes dificuldades, listadas abaixo.

A não avaliação do que é digitado, levando ao travamento inesperado do GaCIV.

A necessidade de programar métodos de controle de colisão mais efetivos, já que as funções de controle de colisão do Java3D são básicas (J3d, 2003).

A necessidade de criar uma caixa de seleção nos componentes 3D selecionados a partir do InterBuilder, para que o usuário saiba qual componente 3D está selecionado e permitir uma melhor resposta à seleção de componentes 3D.

Definir uma restrição à capacidade de manipulação de componentes 3D no cenário, já que o GaCIV permite uma livre rotação, o que torna difícil a manipulação de componentes 3D na interface com RV. Essa dificuldade é relacionada ao fato do GaCIV utilizar os métodos de manipulação já desenvolvidos no Java3D. Pela não disponibilização de métodos que controlem restrições a livre manipulação de objetos 3D presentes no cenário, é necessário desenvolver ou modificar esses métodos para esse objetivo, a partir do seu código.

Por fim, deve ser desenvolvido um controle dinâmico das coordenadas 3D, para facilitar a manipulação e orientação dos componentes 3D na interface com RV.

6.4. Trabalhos Futuros

O GaCIV tem muito a evoluir em relação às suas funcionalidades e ao estudo de usabilidade para interfaces com RV, para que se possa avaliar formalmente se interfaces com RV produzidas tornam a interação mais natural e intuitiva. Ao longo deste trabalho, a experiência acumulada por sua realização permite afirmar alguns direcionamentos para trabalhos futuros relacionados ao GaCIV, como também em RV no contexto de IHC. As propostas são as seguintes:

- Re-estudo das funcionalidades do GaCIV para usuários finais *(End User Programming)*, com objetivo de permitir criar novas funcionalidade por esses usuários;
- Estudo da usabilidade da ferramenta GaCIV
- Estudo de usabilidade de interfaces com RV, criadas a partir do ambiente GaCIV, para saber se componentes 3D permitem criar funcionalidades que aumentam a usabilidade e a naturalidade de interfaces com RV;
- Exploração das potencialidades do GaCIV na reengenharia de sistemas utilizando componentes 3D;
- Criação de editor de objetos 3D que facilite o reuso composicional, a partir da criação visual de componentes 3D compostos por outros componentes 3D;
- Projeto de interfaces com RV através da Web (Web3D);

- Projeto de interfaces com RV baseado em aspectos;

7. Bibliografia

(Albertin, 2002) Albertin, J, C. L. *Reuso de componentes para a construção de Interfaces com Realidade Virtual.* Monografia de Qualificação. Departamento de Computação, UFSCar, São Carlos, SP, Brazil, 2002.

(ALICE, 2003) ALICE web site. URL: www.alice.com, acessado em maio/2003.

(Appel et al, 1999) APPEL, A. P. et al. GACIV - A Realidade Virtual Apoiando o Desenvolvimento de Interfaces com a Participação Efetiva do Usuário. In: Simpósio Brasileiro de Engenharia de Software, *Sessão de Ferramentas*. Santa Catarina, Outubro/1999.

(Assis, 2000) Assis, A. S. F. R., *Um ambiente computacional para desenvolvimento de interfaces utilizando realidade virtual*, Monografia de Qualificação, DC-UFSCar, 2001.

(Assis, 2001) Assis, A.F.S.R. *Um Ambiente Computacional para Desenvolvimento de Interfaces Utilizando Realidade Virtual*. Dissertação de Mestrado. Departamento de Computação, UFSCar, São Carlos, SP, Brazil, 2001.

(Barbosa, 1999) Barbosa, S.D.J. *Programação Via Interface*. Tese de Doutorado. Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ, 1999.

(Bowman et al, 2001) Bowman, D., Kruijff, E., LaViola, J., and Poupyrev, I. An Introduction to 3D User Interface Design, *Presence: Teleoperators and Virtual Environments*, vol. 10, no. 1, 2001, p96-108.

(Braga, 2000) Braga, R. M. M., Desenvolvimento Baseado em Componentes. Disponível em http://www.cos.ufrj.br/~odissey, visitado em outubro/2002.

(Burdea & Coiffet, 1994) Burdea, G. Coiffet, P. *Virtual Reality Technology*. Estados Unidos: John Wiley & Sons, 1994.

(CBSE, 1998, 1999, 2000) Component-Based Software Engineering Workshops Series, URL: www.sei.cmu.edu/cbs, vistado em Janeiro/2001.

(Coad, 1992) Coad, P. Object-Oriented Patterns. *Communications of the ACM*, V. 35, nº9, p. 152-159, setembro 1992.

(Conner et al, 1992) Conner, B., Snibbe, S., Herndon, K., Robbins, D., Zelesnik, R., van Dam, A., Three-Dimensional Widgets, Computer Graphics, *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, 25, 2, pp. 183-188, 1992.

(Contigra, 2003) Contigra web site. URL: www.contigra.com, acessado em maio/2003.

(Cooper & Gray, 2001) Cooper, R., Gray, P., *Characterising User Interface Componentes for Reuse*, http://www.dcs.gla.ac.uk/~rich [visitado em 06 de dezembro de 2001].

(Crnkovic & Larsson, 2000) Crnkovic, I., Larsson, M.,A Case Study: Demands on Component-based Development, *22th International Conference of Software Engineering, Limerick*, Ireland, June 2000.

(Dachselt, 1999) Dachselt, R., The Challenge to Build Flexible User Interface Components for Non-Immersive 3D Environments; *Proceedings of HCI International '99*, Munich, Germany, 1999.

(Dachselt, 2000) Dachselt, R., Action Spaces - A metaphorical concept to support navigation and interaction in 3D interfaces; *Proceedings of the Workshop Usability Centered Design and Evaluation of Virtual 3D Environments*, Paderborn, Germany, 2000.

(Dachselt, 2001) Dachselt, R., Contigra - Towards a Document-based Approach to 3D Components; *Proceedings of the Workshop Structured Design of Virtual Environments and 3D-Components at the ACM Web3D 2001 Symposium*, Paderborn, 2001.

(Dachselt, 2003) R. Dachselt: CONTIGRA: An XML-Based Architecture.for Component-Oriented 3D Applications, disponível em http://www-mmt.inf.tu-dresden.de/web3d2003/, visitado em junho/2003.

(Döllner & Hinrichs, 1998) Döllner, J.; Hinrichs, K.; Interactive, Animated 3D Widgets. *In IEEE Proceedings of CGI '98*, p.278-286, 1998.

(Dörner & Grimm, 2001) Dörner, R., Grimm, P., Building 3D Applications with 3D Components and 3D Frameworks, *International Workshop on Structured Design of Virtual Environments and 3D-Components at the WEB3D 2001 Conference*, Paderborn, Germany, Feb. 19th, 2001.

(Fencott & Isdale, 2001) Fencott, C., Isdale, J., Design Issues for Virtual Environments**,** *International Workshop on Structured Design of Virtual Environments and 3D-Components at the WEB3D 2001 Conference*, Paderborn, Germany, Feb. 19th, 2001.

(Figueroa et al, 2002) Figueroa, P.; Green, M.; Hoover, H. J.; InTml: a description language for VR applications, *Proceeding of the seventh international conference on 3D Web technology*, Tempe, Arizona, USA, p. 53 – 58, 2002.

(Grundy & Hosking, 2000) Grundy, J.C. and Hosking, J.G. Developing Adaptable User Interfaces for Component-based Systems, *In Proceedings of the 1st Australasian User Interface Conference*, Canberra, Australia Jan 30-Feb 2 2000, IEEE CS Press, pp. 17-25.

(Hoffmann et al, 2002) Hoffmann, H.; Dachselt, R.; Meißner, K.:An Independent Declarative 3D Audio Format on the Basis of XML; *Proceedings of the 2003 International Conference on Auditory Display*, Boston, MA, USA, 6-9 July 2003

(J3d, 2003) J3D Web Site. URL: http://www.j3d.org, visitado em janeiro/2003.

(Karlsson, 1995) Karlsson, E**.,** ***Software Reuse: A Holistic Approach***, Wiley Series in Software Based Systems, England, 1995.

(Kaur, 1998) Kaur K. *Designing virtual environments for usability*. PhD thesis. Centre for HCI Design, City University, London, June 1998.

(Kim et al, 1998) Kim, G. J., Kang K. C., Kim H., Lee J., Software engineering of virtual worlds, *Proceedings of the ACM Symposium on Virtual reality software and technology*, p131 – 138, Taipei, Taiwan, 1998.

(Kozaczynski, 1999) Kozaczynski, W., Composite Nature of Component, International *Workshop on Component-Based Software Engineering*, Los Angeles, EUA, Maio, 1999.

(Nielsen, 1993) Nielsen, J., *Usability Engineering*. Estados Unidos: AP Professional, 1993.

(OpenSG, 2003) Open Scene Graph web site URL: www.opensg.org, acessado em maio/2003.

(Pinho, 2000) Pinho, M. S., Interação em Ambientes Tridimensionais, Workshop de Realidade Virtual, Gramado, RS, *Anais*... Porto Alegre, RS:Sociedade Brasileira de Computação, 2000.

(Poulin, 1999) Poulin, J. S., Reuse: Been There, Done That. *Communications of the ACM*,v.43,n.5, 1999.

(Pressman, 2001) Pressman,S.R., *Software Engineering: A Practitioner's Approach*, Fifth Edition, McGraw-Hill Higher Education, New York, 2001, 860p.

(Prieto-Diaz, 1993) Prieto-Diaz, R. Status report: software reusabillity**.** ***IEEE Software***, v.10, n.3, p61-66, 1993.

(Rocha & Baranauskas, 2000) Rocha, H. V. D., Baranauskas, M. C. C., *Design e avaliação de interfaces humano-computador*, Escola de Computação,São Paulo, Julho/2000, 242p.

(Rudolf, 1999) X3d Components Specification. URL: http://www.web3d.org/
TaskGroups/x3d/lucidActual/X3DComponents/X3DComponents.html, 1999, acessado em abril/2003.

(Sametinger, 1997) Sametinger, J., *Software Engineering with Reusable Components*, Springer, 1997.

(Sense8, 2003) Sense 8 web site. URL: www.sense8.com, acessado em maio/2003.

(Shaw & Garlan, 1996) Shaw, M.; Garlan, D.; *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996

(Silva, 1999) Silva, J.C.A. Development of Virtual Interfaces Using Configurable Templates. In *3rd International Conference on Computational Intelligence and Multimedia Applications*, New

Delhi, India, September 1999, p. 354-358.

(Silva, 2001) Silva, W. F. da, *Sistemas Hipermídia de Apoio a Documentação de Software – Uma investigação sobre a mudança de Domínio de Software*, ICMC, USP, Dissertação de Mestrado, 2001.

(Soares, 2002) Soares, C. L. *Reengenharia de Interfaces com Realidade Virtual Considerando o Paradigma de Orientação a Objetos*, Dissertação de Mestrado, DC-UFSCar, 2002.

(SUN, 2003) JavaBeans web Site. URL: http://java.sun.com/javabeans, visitado em janeiro/2003.

(SUN, 2003a) OpenInventor Home Page. URL: http://www.sgi.com/software/inventor/, visitado em maio/2003.

(SUN, 2003b) OpenGL Performer. URL: http://www.sgi.com/software/performer/, visitado em maio/2003.

(SUN, 2003c) Sun. Microsystems. Java 3D home page. URL: http://java.sun.com/products/ java-media/3D/index.html, acessado em abril/2003.

(SUN, 2003d) Sun. Microsystems. Java home page. URL: http://java.sun.com, acessado em abril/2003.

(Szyperski, 1998) Szyperski, C.; *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley, 1998.

(Turoff, 1997) Turoff, M., Virtuality. *Communications of the ACM*, v.40, Issue 9,p38 – 43, 1997.

(Van Dam, 1997) Van Dam, Adrian. Post-WIMP User Interfaces. *Comunications of the ACM*, v.40,n.2,p.63-67, Fevereiro/1997.

(Vince, 1995) VINCE, John. *Virtual Reality Systems*. Estados Unidos: Addison-Wesley, 1995.

(VrJuggler, 2003) VrJuggler web site. URL: www.vrjuggler.org, acessado em maio/2003.

(W3C, 2003a) SAX Specification Project Web Site. URL://www.saxproject.org, visitado em abril/2003.

(W3C, 2003b) DOM Specification Web Site. URL: http://www.w3.org/DOM, visitado em abril/ 2003.

(W3C, 2003c) XML Specification Web Site. URL: http://www.w3.org/XML, visitado em abril/2003.

(Walczak & Cellary, 2002) Walczak, K.; Cellary, W.; Building database applications of virtual reality with X-VRML, *Proceeding of the seventh international conference on 3D Web technology*, Tempe, Arizona, USA, p. 111 – 120, 2002.

(Werner & Braga, 2000) Werner, C.M.L., Braga, R.M.M., Desenvolvimento Baseado em Componentes, Universiadade Federal do Rio de Janeiro, *Mini Cursos*, SBES, 2000;

(Yacoub et al, 1999a) Yacoub , S., Ammar, H., Mili, A., Characterizing a Software Component, *International Workshop on Component-Based Software Engineering*, Los Angeles, EUA, Maio 1999.

(Yacoub et al, 1999b) Yacoub , S., Ammar, H., Mili, A., A Model for Classifying Component interfaces, *International Workshop on Component-Based Software Engineering*, Los Angeles, EUA, Maio 1999.