Uma Abordagem 3D para a Visualização de Padrões de Projeto

Marcelo Campo[1] Roberto Tom Price Alfredo Teyseyre[1]

UFRGS - Instituto de Informática
Caixa Postal 15064 - Porto Alegre-RS, Brasil

[1]UNICEN - FCEx - Instituto de Sistemas
Grupo de Objetos y Visualización
San Martín 57, (7000) Tandil, Bs. As., Argentina

email: {mcampo, tomprice}@inf.ufrgs.br

Resumo

Neste trabalho apresenta-se uma abordagem para a visualização da informação de padrões de projeto existentes em um framework utilizando uma representação visual tridimensional. Sob esta abordagem, uma classe é visualizada como um volume composto pelos três eixos do espaço tridimensional. Cada eixo representa uma das três categorias nas quais os padrões são classificados. Cada padrão existente numa classe é representado por um poliedro de forma característica e a cor do poliedro relaciona as diferentes classes associadas pelo padrão. Através desta representação é possível visualizar de forma integrada a informação de classes, seus relacionamentos e a informação relativa aos padrões que definem a estrutura do framework, sem a necessidade de múltiplas visualizações bidimensionais. Esta abordagem foi implementada como uma extensão da ferramenta MetaExplorer, utilizando-se de uma extensão 3D do sub-framework de visualização do framework Luthier, desenvolvido para a construção de ferramentas de visualização de software.

Palavras Chave: Visualização de Software, Framework Orientados a Objetos, Padrões de Projeto, Compreensão de Software

Abstract

In this work, a 3D visualization approach to the visualization of design patterns existing in a given framework is presented. In this approach, a class is visualized as volume composed of three semi-axes that represents the coordinate system. One concrete class is visualized as a sphere and an abstract class is visualized as a pyramid. Each axis represents a pattern category, that is, behavioral, creational and structural. Each pattern is represented by a distinctive polyhedral shape defined by specific mapping components. Through this representation, it is possible to simultaneously visualize the classes that compose the application and the way they are related by those design patterns that define the framework essential structure, without the need of several two dimensional representations. The approach was implemented as an extension of the MetaExplorer tool, using a 3D extension of the visualization sub-framework of the Luthier framework, specially developed for the construction of software visualization tools.

Keywords: Software Visualization, Object-Oriented Frameworks, Design Patterns, Software Comprehension

1. Introdução

A *compreensão de programas* é um dos problemas mais críticos dentro do ciclo de vida de software. O sucesso de atividades tais como manutenção, depuração e reutilização dependem, em grande parte, da facilidade com a qual um programador possa *compreender* um dado programa. Este problema é ainda maior para o caso de programas

orientados a objetos e, particularmente, frameworks. A dicotomia entre o modelo de execução e o modelo estático de hierarquias de classes [BUH 92], a distribuição de funcionalidade entre múltiplas classes [WIL 92], o acoplamento dinâmico [DEP 93][LAN 95] e sua combinação com o polimorfismo [TAE 89], tornam os programas orientados a objetos difíceis de compreender.

Neste contexto, ferramentas capazes de analisar o comportamento de um programa e visualizar como suas classes se organizam e relacionam através da troca de mensagens, tornam-se um complemento muito importante para facilitar a compreensão desse programa, bem como de um framework que o suporte. Particularmente, ferramentas capazes reconhecer e visualizar abstrações de maior nível que classes e mensagens, como subsistemas e padrões de projeto (*design patterns*) [GAM 94], oferecem um excelente veículo para compreender o projeto de um programa, ou framework, em um nível de abstração muito maior que o nível provido por simples visualizações baseadas em classes e mensagens.

Os padrões de projeto representam abstrações de projeto que não são suportadas pelas linguagens orientadas a objetos atuais, mas que são de grande importância para compreender como os objetos de um sistema são organizados e colaboram para satisfazer a funcionalidade global. Um padrão de projeto nomeia uma dada combinação de classes e métodos que solucionam um problema de projeto que aparece repetidamente em diferentes aplicações. Se um programador conhece qual o problema que um dado padrão resolve, e quais classes e métodos o padrão prescreve para a solução genérica, então esse programador pode compreender mais rapidamente a natureza do relacionamento existente entre duas classes sem necessidade de uma análise detalhada.

Assim, a *visualização* dos padrões existentes em um dado programa surge como um elemento importante para facilitar a compreensão de como determinadas partes desse programa foram projetadas e a função que determinados métodos tem dentro de uma dada classe. Sem dúvida, a utilização de técnicas visuais é um componente essencial para facilitar o processo de compreensão de sistemas complexos, tais como os frameworks. Entretanto, é necessário tomar em consideração que a compreensão de sistemas complexos envolve uma grande quantidade de informação que implica na necessidade de mecanismos para visualizar e ressaltar aspectos relevantes dessa informação. Grandes quantidades de dados podem facilmente inibir a capacidade humana para assimilá-los e construir um modelo mental adequado [MIL 90]. Por esta razão, as técnicas de visualização providas são um fator chave para uma ferramenta apoiar eficazmente à compreensão de programas. Deste modo, o objetivo principal de uma visualização é condensar a maior quantidade possível de informação em uma mesma apresentação. Uma visualização compacta facilita o processamento imediato da informação transmitida.

No caso da visualização de padrões de projeto, é necessário levar em conta que estes padrões representam uma abstração de projeto que é descrita em termos de classes e métodos. Assim, uma visualização adequada, em termos de síntese de informação, deve tornar visualmente evidente tanto o padrão quanto as classes e métodos envolvidos. A ferramenta

MetaExplorer [CAM 97], por exemplo, fornece visualizações alternativas, baseadas na notação OMT, utilizando cores diferentes para identificar os métodos e relacionamentos correspondentes com cada padrão reconhecido em um framework analisado. A ferramenta ProgramExplorer [LAN 95], por sua vez, utiliza diagramas de interação para mostrar as classes e o fluxo de mensagens correspondente com um dado padrão. Ambas visualizações são úteis, sem duvida, mas apresentam limitações quando é necessário visualizar globalmente a estrutura de relacionamentos entre classes e os padrões envolvidos em um dado framework. Assim, um mecanismo alternativo para a visualização integrada de classes, relacionamentos, métodos e a informação de padrões de projeto é ainda necessário.

Neste trabalho apresenta-se uma abordagem para a visualização da informação de padrões de projeto existentes em um framework utilizando uma representação visual tridimensional. Sob esta abordagem, uma classe é visualizada como um volume composto pelos três eixos do espaço tridimensional. Cada eixo representa uma das três categorias nas quais os padrões são classificados. Cada padrão existente numa classe é representado por um poliedro de forma característica e a cor do poliedro relaciona as diferentes classes associadas pelo padrão. Esta abordagem foi implementada como uma extensão da ferramenta MetaExplorer, utilizando-se de uma extensão 3D do sub-framework de visualização do framework Luthier [CAM 96][CAM 97], desenvolvido para a construção de ferramentas de visualização de software.

Este artigo é organizado da maneira seguinte. A seguir, na seção 2, descrevem-se sinteticamente os padrões de projeto e sua classificação. A seção 3 apresenta uma breve descrição da ferramenta MetaExplorer e a abordagem adotada para a visualização tridimensional de padrões de projeto. A seção 4 descreve os trabalhos relacionados, enquanto a seção 5 apresenta algumas conclusões relevantes.

2. Padrões de Projeto

Os padrões de projeto são padrões de organização de hierarquias de classes, protocolos e distribuição de responsabilidades entre classes, que caraterizam *construções elementares* de projeto orientado a objetos [GAM 94]. Um *padrão de projeto* é uma estrutura que aparece repetidamente nos projetos orientados a objetos, a qual é utilizada para resolver um problema determinado de forma flexível e dinamicamente adaptável. Por exemplo, a forma em que uma aplicação pode ser independente do tipo de interface gráfica na qual tem que rodar ou como separar a representação dos dados da sua apresentação na tela.

Os padrões de projeto representam um avanço importante na área de orientação a objetos, pois oferecem um catálogo de *planos de projeto* que permitem a reutilização de *soluções* de projeto que provaram ser efetivas para resolver problemas semelhantes. Eles dão um nome e uma descrição abstrata para estas estruturas, permitindo assim comunicar um projeto em termos de uma linguagem de mais alto nível que as notações básicas. A sua descrição num nível abstrato permite sua reutilização para projetar novas aplicações ou

melhorar aplicações existentes, de forma independente da implementação. Também, este tipo de informação pode ser de muita utilidade para documentar o projeto de um framework em um alto nível de abstração.

Um exemplo conhecido é o denominado padrão denominado *Composite*. Este padrão descreve a forma de compor objetos em estruturas de árvore para representar hierarquias de partes ou de conteúdo. O padrão define a forma de tratar múltiplos objetos compostos recursivamente como um objeto simples, simplificando assim o código dos clientes. A Fig. 1 apresenta a estrutura de classes típica deste padrão, para um exemplo particular de objetos representando visões gráficas. Tipicamente o objeto composto distribui a aplicação de operações entre seus filhos, obtendo uniformidade através do polimorfismo. Este é o caso do método *displayOn:*, cuja implementação no objeto composto consiste de propagar a mensagem aos componentes visuais contidos pela instância de *CompositePart*.

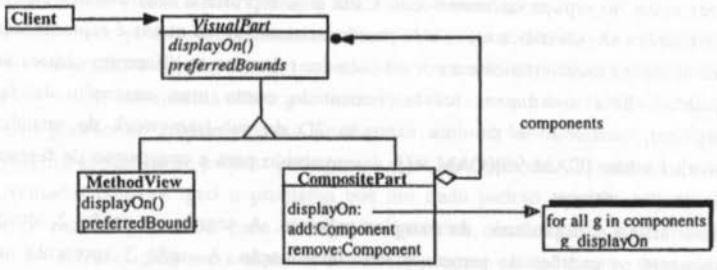

Fig. 1- Estrutura abstrata de classes do padrão *Composite*

Os padrões de projeto identificados até agora variam na sua granulosidade e nível de abstração, mas mesmo assim, possuem caraterísticas comuns e em alguns casos estão fortemente relacionados. Os padrões são classificados de acordo a um sistema que agrupa os diferentes padrões em categorias que facilitam sua compreensão e utilização. Este sistema divide o conjunto de padrões em função de dois critérios ortogonais: escopo e caracterização. A Tabela 1 mostra o espaço de padrões de projeto atualmente identificado, particionado por estes critérios. Cada padrão é descrito informalmente através de uma ficha que lista todas as caraterísticas relevantes para sua utilização.

•**Escopo** é o domínio sobre o qual o padrão pode ser aplicado. Os padrões categorizados dentro da jurisdição de classes tratam acerca de relacionamentos entre classes e as suas subclasses. A jurisdição de classes cobre a semântica estática dos relacionamentos. A jurisdição de objetos envolve relacionamentos entre objetos individuais, enquanto a jurisdição de compostos abrange as estruturas de objetos recursivas.

•**Caracterização** identifica a função do padrão. Os padrões *Criacionais* envolvem o processo de criação de objetos, os *Estruturais* tratam acerca da forma em que classes e objetos são compostos para formar estruturas de nível superior. Os padrões *Comportamentais*, por sua vez, caraterizam a forma em que classes e objetos interagem e distribuem responsabilidades.

		Caracterização		
		Criacional	**Estrutural**	**Comportamental**
Escopo	**Classe**	*Factory Method*	*Adapter*	*Template Method*
	Objeto	*Abstract Factory* *Prototype* *Singleton*	*Adapter* *Bridge* *Facade* *Proxy* *Flyweight*	*Chain of Responsibility* *Command* *Iterator* *Mediator* *Memento* Observer *State* *Strategy* *Visitor*
	Compostos	*Builder*	*Composite* *Decorator*	*Iterartor* *Visitor*

Tabela 1- Espaço de Padrões de Projeto

3. Visualizando Padrões de Projeto com MetaExplorer

MetaExplorer é uma ferramenta para análise e visualização de programas orientados a objetos desenvolvida utilizando o framework Luthier [CAM 96]. MetaExplorer carateriza-se pela utilização de técnicas de reflexão computacional baseadas em meta-objetos para analisar a estrutura estática e dinâmica de aplicações construídas com um dado framework. Meta-objetos [MAE 88] fornecem um excelente veiculo para capturar informações tanto estáticas quanto dinâmicas, permitindo construir ferramentas dinamicamente configuráveis para análise de programas dentro de um modelo orientado a objetos uniforme.

O mecanismo básico para a análise de uma aplicação com MetaExplorer, envolve três passos, habitualmente iterativos. Em primeiro lugar, é necessário refletir as classes da aplicação a ser analisada sobre um conjunto predefinido de meta-objetos. Após este processo, executa-se a aplicação e os meta-objetos associados no processo anterior, coletam a informação da execução, criando a representação da estrutura do framework utilizando um gerenciador de hipertexto. A partir desta representação diferentes abstrações, como subsistemas e padrões de projeto são automaticamente recuperadas, e diferentes visualizações da estrutura estática dinâmica do framework são produzidas usando notações convencionais, tais como OMT e diagramas de interação, etc. (Fig. 2). Também, MetaExplorer fornece visualizações abstratas alternativas, como grafos abstratos de hierarquias e grafos de fluxos de mensagens, as quais realçam aspectos característicos dos frameworks, como por exemplo categorias de métodos (abstratos, templates, hooks e base).

A navegação e as interfaces de manipulação direta são dois mecanismos complementares para facilitar a compreensão de dados complexos. MetaExplorer fornece uma interface com o usuário poderosa, a qual permite, por exemplo, a seleção e *zooming* de porções de diagramas específicas utilizando notações alternativas, bem como a animação do fluxo de mensagens do framework sob controle interativo do usuário [CAM 96]. MetaExplorer também implementa um mecanismo inovativo de zoom semântico [MUT 95]

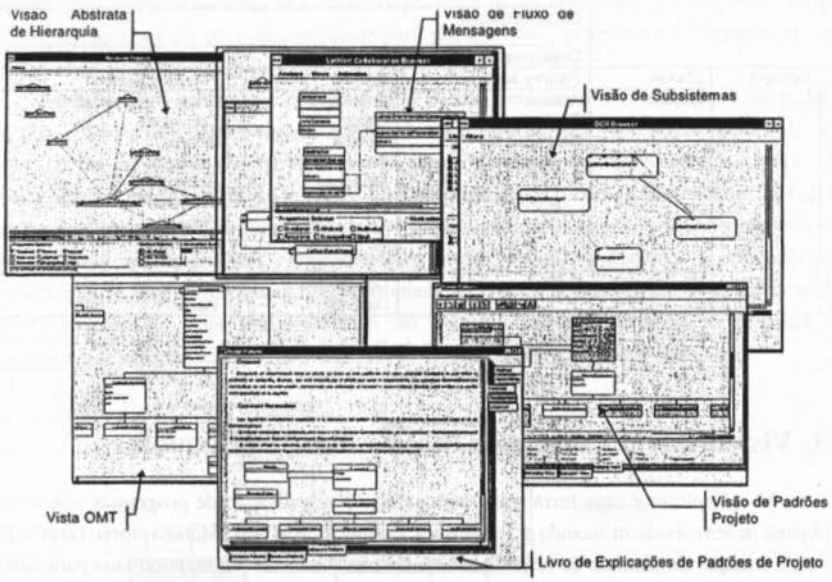

Fig. 2- Diferentes visualizações bidimensionais produzidas por MetaExplorer

baseado em escalas de abstração. Este mecanismo habilita o zoom contínuo dos diagramas, mostrando ou ocultando informação pertinente a cada nível de detalhe. Este capacidade contribui para reduzir a proliferação de múltiplas janelas, contribuindo assim para diminuir o bem conhecido problema da desorientação nos sistemas de hipertexto.

A filtragem interativa de informação através das capacidades de consulta providas é um mecanismo adicional para ajudar a reduzir a complexidade das visualizações. Consultas textuais, baseadas em propriedades abstratas de mensagens, permite ao usuário filtrar as visualizações para mostrar só aqueles componentes que satisfazem um conjunto dado de propriedades, como por exemplo, classes relacionadas por mensagens que ativam a redefinição de métodos abstratos.

3.1.1 Visualização Bidimensional de Padrões de Projeto

A Fig. 3 apresenta uma imagem do browser de padrões provido pela ferramenta. Neste browser, utiliza-se da visualização OMT, substituindo a visualização dos relacionamentos pelo fluxo das mensagens trocadas entre os métodos que correspondem com cada padrão. O painel inferior apresenta a lista completa de padrões, ressaltando com cores diferentes os padrões que foram reconhecidos durante a análise. A seleção de um padrão nesta lista, produz que as classes envolvidas mostradas na visualização sejam coloridas com a cor que representa

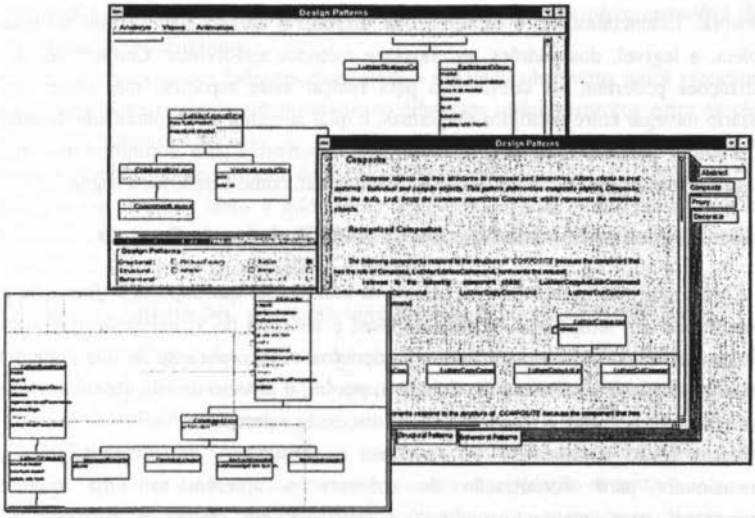

Fig. 3 - Visualizações bidimensionais alternativas de padrões de projeto e o livro de documentação

o padrão. Isto permite analisar cada padrão separadamente, e realizar o zoom para a visualização de fluxo de mensagens para analisar o comportamento dinâmico do padrão.

Alternativamente pode ser possível visualizar com cores os métodos e os relacionamentos que definem cada padrão. A primeira visualização é útil para centrar a atenção em padrões particulares, enquanto a segunda é útil para visualizar quais os padrões envolvidos no projeto de cada classe.

O browser de padrões permite a geração de um livro que contém os padrões reconhecidos no framework analisado, com uma explicação sintética do objetivo do padrão, e para cada instância de padrão reconhecida, porque as classes é métodos envolvidos são parte desse padrão, bem como o diagrama OMT que descreve a localização do padrão na estrutura de classes. O livro é dividido em três capítulos, um para cada categoria de padrão, isto é criacionais, estruturais e comportamentais. Cada capítulo é dividido em seções, as quais contém a explicação correspondente a cada padrão reconhecido. Este tipo de informação fornece ao usuário informação adicional que facilita a compreensão da funcionalidade implementada pelas classes envolvidas. A combinação de representações textuais é gráficas permite analisar cada padrão de acordo com a estrutura de classes e a funcionalidade dos métodos envolvidos do ponto de vista da intenção de projeto representada pelo padrão.

3.1.2 Limitações

Este tipo de representações, no entanto, apresentam o problema da dificuldade de visualizar em termos globais como os padrões intervém na arquitetura geral de um

framework. Essencialmente, o espaço físico disponível da tela não facilita a visualização completa, e legível, dos padrões, as classes e métodos envolvidos. Certamente, diferentes visualizações poderiam ser construídas para realçar estes aspectos, mas neste caso seria necessário navegar entre múltiplos diagramas, o qual aumenta a complexidade da interação e pode produzir problemas de desorientação. Uma alternativa para diminuir estes afeitos é a utilização de um espaço de visualização tridimensional, como é descrito a seguir.

3.2 Um Esquema de Visualização 3D para Padrões de Projeto

Com o surgimento de hardware gráfico de baixo custo que suporta a síntese de gráficos tridimensionais em tempo real, novas interfaces e sistemas de visualização que aproveitam estas capacidades estão atualmente sendo explorados. A incorporação de una nova dimensão na visualização apresenta múltiplas vantagens, porém, o aspecto de sua utilidade efetiva para a visualização de software é ainda matéria de discussão e pesquisa.

Nesta seção discutem-se as vantagens e limitações da utilização de gráficos tridimensionais para visualização de software e apresenta-se uma representação tridimensional para integrar visualizações centradas em classes e colaborações, com informação relativa aos padrões de projeto nos quais cada classe participa. Estas visualizações foram construídas utilizando um framework para desenho 3D, que estende o sub-framework de visualização de Luthier, o qual implementa uma versão orientada a objetos do padrão para desenho 3D, OPEN-GL.

3.2.1 Visualização de Software e 3D: Vantagens e Limitações

Considerando a necessidade de aproveitar as capacidades cognitivas dos humanos para se manejarem em três dimensões, a terceira dimensão abre possibilidades interessantes para explorar espaços de informação. Como Koike observa, grande parte dos problemas existentes com as visualizações bidimensionais podem ser resolvidos com a utilização de visualizações 3D [KOI 93]. A utilização de uma terceira dimensão para visualização de informação apresenta várias vantagens, a saber:

- É possível visualizar uma maior quantidade de objetos, devido à existência de um espaço conceitual maior. Isto implica num aumento na densidade de informação (quantidade informação / espaço físico) mostrada [ROB 93], o qual permite aumentar o poder de expressividade das visualizações.
- Podem realizar-se apresentações com maior legibilidade. Existindo mais espaço físico para visualizar informação, é possível distribuir melhor a mesma. Por exemplo, a visualização de um grafo com muitos arcos em 2D produz muitos cruzamentos de elos não desejados, enquanto que esta mesma visualização em 3D pode ser muito melhor distribuída e, por tanto, minimizar-se-ão a quantidade de cruzamentos.
- É possível distribuir os objetos na tela de acordo com alguma semântica particular. Por exemplo, se podem agrupar os elementos que pertencem a tarefas comuns mais

próximos entre si. Isto permite ao usuário se concentrar na parte específica de seu interesse de um diagrama.

- Dois ou mais gráficos bidimensionais podem ser unificados numa única visualização . Isto possibilita enxergar simultaneamente diferentes relacionamentos entre os objetos. De esta forma, o usuário no necessita criar um modelo com a união de várias apresentações, já que este esta dado pelo diagrama 3D.
- É possível integrar tanto a informação local como global numa única visualização [MAC 91]. Isto es possível realizando projeções e mudando os pontos de vista do usuário a respeito da informação apresentada.

Este tipo de visualizações, entretanto, apresenta alguns inconvenientes, a saber:

- Para obter tempos de resposta adequados para seu uso, é necessário hardware especializado para desenho 3D, o qual não é facilmente disponível nas estações de trabalho atualmente utilizadas.
- A manipulação direta neste tipo de interfaces requer de dispositivos especializados que podem inibir a sua utilização conjunta com a manipulação de texto.
- Representações 3D de conceitos que não possuam um mapeamento direto com as construções visualizadas podem interferir na facilidade de compreensão inicial da própria visualização, aumentando a curva de aprendizado de uma ferramenta.
- A implementação de visualizações tridimensionais é muito mais complexa que representações em duas dimensões, mesmo com as bibliotecas de suporte para gráficas 3D hoje disponíveis.

Independente destas limitações, visualizações 3D podem ser de grande utilidade quando pensadas como um complemento de visualizações diagramáticas bidimensionais. Por exemplo, visualizações 3D podem ser de utilidade para representar em forma conjunta o fluxo de controle entre os diferentes níveis das hierarquias de classes que compõem um framework e as classes construídas pelo usuário. Da mesma forma, a utilização de visualizações tridimensionais pode ajudar a condensar em uma mesma apresentação baseada em classes a informação relativa aos padrões de projeto nos quais cada classe participa, tal com é descrito na seção seguinte.

3.2.2 Visualização de Padrões

Utilizando um espaço tridimensional, uma classe pode ser visualizada como um volume composto pelos três semi-eixos que representam o espaço. Uma classe concreta é visualizada como uma esfera, enquanto uma classe abstrata (isto é, uma classe com pelo menos um método não implementado) é visualizada como uma pirâmide. Cada eixo representa uma das três categorias básicas nas quais os padrões são classificados, isto é, comportamentais, estruturais e criacionais. Cada padrão, por sua vez, é representado por um poliedro de forma

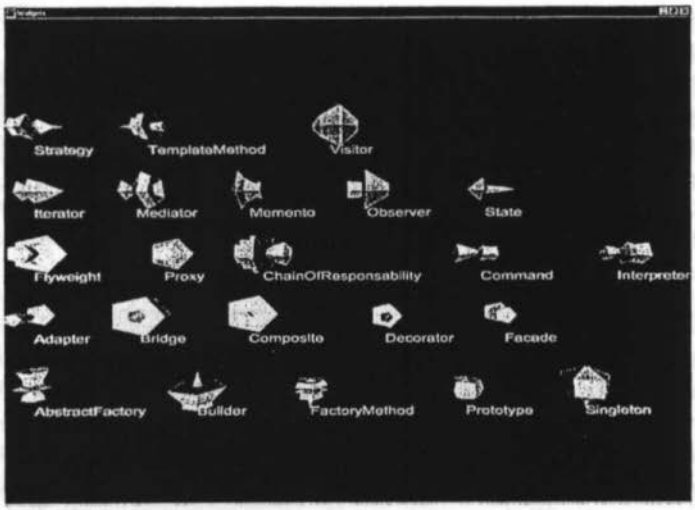

Fig. 4- Poliedros correspondentes com o espaço de padrões de projeto

característica. A Fig. 4 apresenta as formas e orientação associadas com cada padrão de acordo com a categoria à qual pertencem.

A Fig. 5 apresenta uma visualização de uma classe, chamada *GraphLayoutStrategy*, segundo esta representação. A classe intervém em padrões pertencentes às três categorias, *FactoryMethod* da categoria criacional no eixo *y* (para acima), *Decorator* da categoria

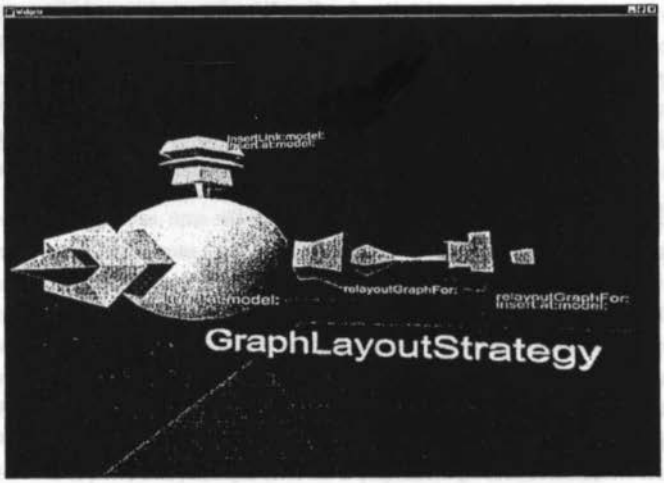

Fig. 5- Exemplo da visualização de uma classe e as três categorias de padrões

estrutural no eixo z (para frente) e *Strategy* e *TemplateMethod* da categoria comportamental no eixo x. A cor do poliedro que representa um padrão relaciona as classes associadas pelo padrão.

Através desta representação é possível visualizar, simultaneamente, as classes que compõem uma aplicação, como estão relacionadas pelos padrões nos quais intervém e quais os métodos que determina cada padrão. Este aspecto é de grande importância para facilitar a compreensão da estrutura global de um framework tomando em consideração os padrões que estão envolvidos em cada classe e permite identificar visualmente quais classes estão relacionadas através de um dado padrão. Uma vez que o usuário consegue identificar a forma característica do padrão, o qual pode não resultar evidente em princípio, é capaz de identificar imediatamente o tipo de relacionamento que existe entre um número potencialmente grande de classes e subclasses. Uma análise mais detalhada, pode permití-lhe uma compreensão adequada dos métodos que devem ser redefinidos e quais as classes que devem ser especializadas. Definitivamente, este aspecto representa um ganho importante no que diz respeito às facilidades que fornece uma ferramenta para aproveitar as capacidades cognitivas do usuário.

3.2.3 Visualização da Estrutura Global e Padrões Envolvidos

A Fig. 6 apresenta um exemplo da visualização parcial da estrutura hierárquica de classes e os padrões envolvidos no framework Luthier. As linhas que unem classes representam relacionamentos de classes e subclasses. Cada hierarquia de classes desenha-se como uma *árvore cônica* (*cone-tree* [MAC 91]), a qual dispõe a raiz da hierarquia no vértice do cone e seus componentes num nível inferior em forma circular. Esta representação facilita a visualização das hierarquias de classes definidas pelos padrões, particularmente estruturais. A representação visual de cada padrão é repetida nas subclasses que o implementam, enquanto as cores ajudam a identificar rapidamente quais as hierarquias de classes relacionadas por um dado padrão.

Os paneis inferiores representam os controles de navegação providos pela ferramenta. Através desta interface de controles de navegação, o usuário pode navegar pelo espaço tridimensional deslocando a câmara. Este deslocamento pode ser realizado variando a posição em cada eixo independentemente ou através de controles que permitem avançar, retroceder e virar a câmara. Também é possível fixar o foco da câmara em objetos específicos utilizando o mouse. Alternativamente, um *joystick* pode ser utilizado para realizar o deslocamento da câmara.

Este tipo de manipulações, tem como inconveniente a adaptação inicial do usuário, tornando difícil sua utilização para usuários iniciais. No entanto, após uma prática razoável um usuário é capaz de se manejar no espaço tridimensional como facilidade, sendo capaz de explorar detalhadamente os relacionamentos entre classes de uma forma mais natural que com as barras de deslocamento das interfaces bidimensionais.

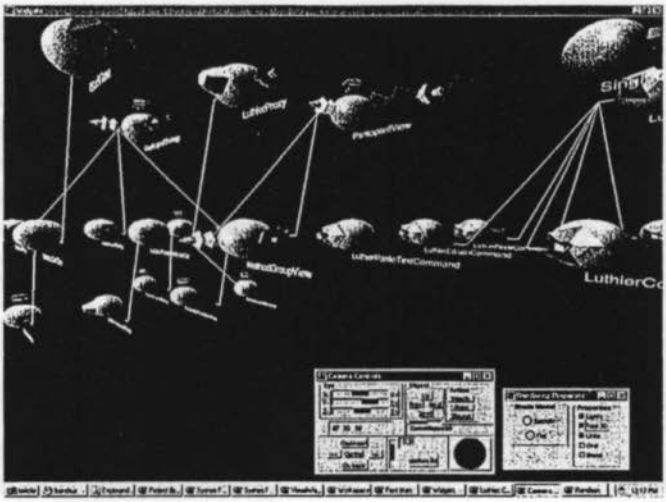

Fig. 6- Visão parcial da estrutura de classes e padrões do framework Luthier

Do ponto de vista cognitivo, é necessário considerar que quantidade de informação existente na visualização torna necessária uma análise detalhada. Isto é, não é possível em visualizações complexas processar de maneira imediata a informação provida, mas a possibilidade de visualizar essa informação em forma integrada elimina a sobrecarga produzida pela navegação entre múltiplos diagramas bidimensionais.

3.2.4 Gerenciamento de Níveis de Abstração

O framework Luthier fornece o mecanismo denominado *abstratores* para gerenciar automaticamente o nível de detalhe das visualizações [CAM 96]. O grau de detalhe da informação apresentada pode ser controlado pelos abstratores, permitindo implementar mecanismos de transformação gradual do nível de detalhe à maneira de um zoom contínuo da visualização. Este mecanismo de zoom pode ser controlado através da definição de *escalas de abstração*. Uma *escala de abstração* define uma seqüência ordenada de conceitos que representam diferentes níveis de abstração de uma hierarquia. Por exemplo, a escala abaixo pode ser utilizada para definir o grau de detalhe de uma visualização baseada em padrões:

(abstractHierarchy creationalPatterns structuralPatterns behavioralPatterns concreteHierarchy relationships methods)

Através desta escala uma visualização mostrará inicialmente só as classes abstratas de um framework. Variando interativamente o nível na escala a visualização é transformada

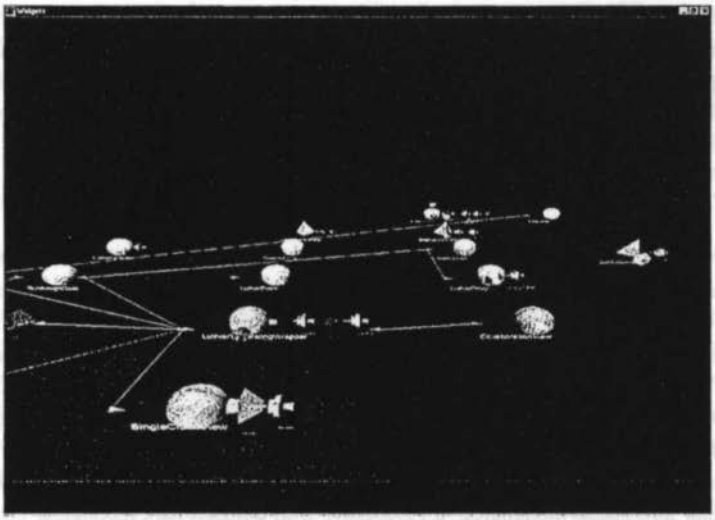

Fig. 7- Visão integrada de classes, colaborações e padrões

gradualmente para mostrar os padrões envolvidos, as subclasses concretas, relacionamentos entre classes e os métodos.

A Fig. 7 apresenta um visão global do browser de relacionamentos entre classes, no nível de relacionamentos entre as classes abstratas do framework. Nesta representação, pirâmides representam classes abstratas e esferas representam classes concretas. As setas simples representam relacionamentos cliente-servidor, enquanto as setas duplas indicam um relacionamento cliente-servidor mútuo entre duas classes. A cor dos relacionamentos dá uma noção relativa da ordem no tempo na qual esses relacionamentos se estabelecem através da troca de alguma mensagem. Os relacionamentos mostrados neste nível representam conjuntos de mensagens, assim, através da interface de seletores de atributos, a semântica da cor utilizada pode ser variada interativamente para representar a ordem relativa dentro de um *thread* [1]de acordo a três critérios:

- ordem da primeira mensagem trocada entre duas classes
- ordem da última mensagem trocada entre duas classes
- ordem da mensagem mais freqüentemente trocada entre duas classes

Quando o usuário seleciona um botão deste painel, a visualização é automaticamente atualizada, variando a cor das setas de acordo com a opção selecionada.

[1]Utiliza-se o termo *thread* para referenciar genericamente um caminho de controle dentro do fluxo de controle total do framework, não indicando, necessariamente, uma execução concorrente desses caminhos.

Nesta visualização a profundidade da apresentação representa a ordem de instanciação das classes no tempo na execução dos exemplos analisados para gerar a visualização. Deste modo, pode se observar que a disponibilidade de uma terceira dimensão permite condensar em uma mesma apresentação uma grande quantidade de informação que requereria múltiplos diagramas bidimensionais diferentes para ser representada.

4. Trabalho Relacionado

Um dos primeiros trabalhos realizados na área de visualização tridimensional aplicada foi *SemNet* [FAI 88] construído para apoio na manutenção de grandes bases de conhecimento através da visualização de grafos 3D. Na área de visualização de programas Lieberman [LIE 89], utilizou uma estrutura tridimensional para representar a estrutura de programas como caixas, visualizando suas execuções com animações. O *Information Visualizer* [MAC 91], [ROB 93] é um ambiente de visualização de informação tridimensional. A contribuição mais relevante deste trabalho é a classificação e definição de diagramas 3D para diferentes tipos de estruturas de informação, como por exemplo o *ConeTree*, utilizado neste trabalho, para representar informação hierárquica, *a Perspective Wall* utilizada para representar informação linear, etc.

Na área de depuração de programas orientados a objetos, Vion-Dury e Santana [VIO 94] apresentam a utilização de visualização tridimensional baseada no conceito de imagens virtuais (*virtual images*), para a depuração de sistemas orientados a objetos distribuídos. Uma imagem virtual é uma representação de um objeto utilizando um modelo espacial 3D. Neste modelo os objetos são representados por poliedros que possuem formas, cores, volumes e orientações específicas. O conceito de *variável visual* é utilizado para associar os valores que o olho humano pode perceber e processar durante uma unidade de percepção com propriedades que identificam atributos particulares dos objetos observados. A ferramenta fornece diferentes visões estáticas 3D das hierarquias de classes e relacionamentos entre objetos distribuídos. Estas representações oferecem as vantagens típicas dos sistemas de visualização 3D devido as facilidades inerentes de navegação e múltiplas perspectivas. Este trabalho justifica muito bem as vantagens da utilização de gráficos 3D para a visualização de sistemas de objetos, mas não resulta evidente como a utilização de formas poliédricas pode substituir completamente o texto.

Técnicas básicas de visualização 3D foram utilizadas também na área de visualização arquitetônica. *Mirror* [ORO 95] é um protótipo de ambiente de desenvolvimento de aplicações reflexivas C++, que utiliza visualizações tridimensionais simples baseadas em grafos, para visualizar arquiteturas reflexivas de meta-objetos. Os diferentes níveis da arquitetura são representados por planos normais ao eixo *y*. Os relacionamentos entre objetos de um mesmo nível são representados por linhas que unem os objetos relacionados, enquanto os relacionamentos entre objetos e meta-objetos são representados por linhas que unem os dois níveis. Cores são utilizadas para ressaltar a transferência de controle entre os diferentes

níveis durante a execução do programa. A utilização de representações 3D é de utilidade, neste caso, para prover o usuário com um modelo semelhante ao utilizado mentalmente para imaginar a composição de uma arquitetura reflexiva. Esta similaridade ajuda muito para facilitar a compreensão geral da arquitetura. O mecanismo de navegação por translação da câmara permite o deslocamento do foco de atenção a diferentes pontos da estrutura, bem como o mecanismo inerente de *zoom* permite visualizar tanto estrutura global como estrutura detalhada dos componentes.

5. Conclusões

Nas seções precedentes apresentou-se sinteticamente uma extensão da ferramenta MetaExplorer para a visualização de padrões de projeto utilizando técnicas 3D. A principal contribuição deste artigo é a apresentação de uma abordagem que permite a visualização integrada de aspectos que não podem ser facilmente representados por uma única notação bidimensional, como é o caso de padrões e classes que possuem um relacionamento unívoco, e a distribuição espacial de categorias de padrões na representação de classes.

A efetividade da abordagem deve ser ainda testada em exemplos reais de utilização com usuários não especialistas na ferramenta. No entanto, representa uma abordagem promissora tanto para facilitar a compreensão de frameworks em níveis altos de abstração quanto das possibilidades que abre para a utilização de dispositivos especializados para a navegação de espaços tridimensionais.

A implementação atual apresenta a limitação de um desempenho bastante pobre quando a quantidade de informação a ser visualizada é relativamente grande, mas este problema pode se facilmente resolvido com dispositivos dedicados de desenho 3D de relativo baixo custo.

6. Referências

[BUH 92] BUHR, R.; CASSELMAN, R. Architectures with Pictures. In: CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, LANGUAGES AND APPLICATIONS, 7., 1992, Vancouver, Canadá. **Proceedings...** New York:ACM Press, Oct.1992.

[CAM 96a] CAMPO, M.; PRICE, R. Um Framework Reflexivo para Ferramentas de Visualização de Software. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 10., 1995, São Carlos, Brasil. **Anais...** São Carlos:SBC, 1996. p. 153-169.

[CAM 97] CAMPO, M.; PRICE, R. Compreensão Visual de Frameworks através da Introspeção de Exemplos. Porto Alegre: CPGCC da UFRGS, Março 1997. Tese de doutorado.

[DEP 93] DE PAUW, W. et al. Visualizing the Behavior of Object-Oriented Programms. **SIGPLAN Notices**, New York ,v.28, n.10, p.326-337, Oct.1993.

[FAI 88] Fairchild K. M. Poltrock, S.E. Furnas G.W. SemNet: Three dimensional graphic representation for large knowledge bases In: Cognitive Science ant its Applications For Human Computer Interaction. Lawrence Erlbaum Associates: Hillsdale, New Jersey. 1998

[GAM 94] GAMMA, E. et al. **Design Patterns:** Reusable Elements of Object-Oriented Design. Reading: Addison-Wesley, 1994.

[LAN 95] LANGE, D.; NAKAMURA Y. Interactive Visualization of Design Patterns Can Help in Framework Understanding. **SIGPLAN Notices**, New York, v.30, n.10. Oct. 1995.

[MAC 91] MACKINLAY, J.; ROBERTSON, G.; CARD, K. The Perspective Wall: Detail and Context Smoothly Integrated. In: CONFERENCE COMPUTER-HUMAN INTERACTION, 1991. **Proceedings...** NewYork: ACM Press, 1991,p. 173-179.

[MAE 88] MAES, P. Issues in Computational Reflection. In: MAES,P.; NARDI, D. (Eds.). **Meta-Level Architecture and Reflection.** Amsterdam: Elsevier Science, 1988. p. 21-35.

[MIL 90] MILLER, G. The Magical Number Seven, Plus or Minus Two: Limits of Our Capacity to Process Information. In: GLINERT, E. (Eds.). **Visual Softwre Development Environments**: Applications and Issues. California:IEEE Press, 1990. p. 276-291.

[MUT 95] MUTHUKUMARASAMY, J.; STASKO, J. **Visualizing Program Executions on Large Data Sets using Semantic Zooming**. Georgia:Georgia Institute of Technology, 1995. (Tech. Report. GIT-GVU-95-02).

[ORO 95] OROSCO, R.; CAMPO, M.; SOLÉ, J. Mirror: Visually Reflecting C++, In: TECHNOLOGY OF OBJECT-ORIENTED LANGUAGES AND SYSTEMS, 15., 1995, Santa Bárbarã. **Proceedings**... California:Prentice-Hall, Aug.1995.

[ROB 93] ROBERTSON, G. Information Visualization using 3D Interactive Animation. **Communications of the ACM**, New York, v.36, n. 4, Apr. 1993.

[TAE 89] TAENZER, D.; GANTI, M.; PODAR, S. Object-Oriented Software Reuse: The Yo-Yo Problem. **Journal of Object-Oriented Programming**, New York, v.2, Sept. 1989.

[VIO 94] VION-DURY, J.; SANTANA, M. Virtual Images: Interactive Visualization of Distributed Object-Oriented Systems. In: CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, LANGUAGES AND APPLICATIONS, 9., 1994, Portland, Oregon. **Proceedings...** New York: ACM, 1994.

[WIL 92] WILDE, N.; HUIT, R. Maintenance Support for Object-Oriented Programs. **IEEE Transactions on Software Engineering**, New York, v.18, n.12, Dec.1992.