Suporte a Padrões no Projeto de Software

Alexandre Dantas, Gustavo Veronese
Alexandre Correa, José Ricardo Xavier, Cláudia Werner
{alexrd, veronese, alexcorr, xavier, werner}@cos.ufrj.br
COPPE/UFRJ – Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
Caixa Postal 68511 – CEP. 21945-970
Rio de Janeiro – Brasil

Resumo

Conhecimento de projeto de software pode ser encontrado disperso na literatura e nas mentes dos desenvolvedores mais experientes. Este conhecimento é um guia importante para a obtenção de um projeto de software de sucesso, contendo características favoráveis à reutilização e flexibilidade. Este artigo apresenta uma organização deste conhecimento na forma de heurísticas, padrões e anti-padrões. São descritos os mecanismos implementados em um ambiente de desenvolvimento de software que oferecem suporte para estes conceitos durante o projeto de software orientado a objetos.

Abstract

A body of knowledge of software design can be found dispersed over the literature and at developers' minds through their learned experiences. This knowledge represents an important guidance towards a successful software design with suitable features for reusability and flexibility. This paper proposes to organize this design knowledge as design heuristics, patterns and anti-patterns. We describe the mechanisms implemented within a software development environment to support these concepts during the object oriented software design.

1 – Introdução

Fundamentalmente, o projeto de software envolve uma sistemática de decomposição da solução, a começar pela descrição em mais alto nível dos principais elementos do sistema e, em seguida, criando uma visão mais detalhada de como as características e funções deste sistema deverão estar integradas [1]. Projetar software orientado a objetos é uma tarefa difícil, e projetar software orientado a objetos reutilizável e flexível é ainda mais difícil. É desejável evitar ou minimizar a necessidade de reestruturação do projeto como resultado da incorporação de novas funcionalidades [2].

Este trabalho apresenta mecanismos de suporte à representação, sugestão e identificação de conhecimentos obtidos a nível de projeto, no ambiente de desenvolvimento de software Odyssey [3]. Na próxima seção, são apresentados brevemente os elementos envolvidos, que são as heurísticas, padrões e anti-padrões. Na terceira seção, são apresentados os mecanismos de apoio à utilização destes elementos implementados no Odyssey. Finalmente, são apresentadas algumas conclusões sobre o trabalho.

2 – Representação do Conhecimento de Projeto

Projetistas de sistemas de software vêm, ao longo dos anos, reconhecendo a importância de se representar e explorar o conhecimento obtido durante a construção de sistemas. Uma das formas consiste no reconhecimento e utilização de boas práticas, idéias e soluções já aplicadas em situações de sucesso e fracasso em projetos de software, como heurísticas de projeto, padrões e anti-padrões [4].

2.1 – Heurísticas de Projeto

Uma heurística de projeto é uma diretriz resultante de conhecimento e experiência que serve como um conselho para tomada de decisões de projeto, sendo de grande importância no sentido de guiar o projetista na elaboração de boas soluções ao longo do desenvolvimento [5]. É importante notar que uma heurística não deve ser vista como uma regra inviolável que deva ser seguida em todas as circunstâncias. Possíveis violações a estas heurísticas devem ser consideradas como avisos ou

indicadores de que alguma decisão de projeto pode ter sido tomada incorretamente. As diversas heurísticas hoje existentes estão dispersas em várias publicações, descritas de maneira não uniforme e muitas vezes implícita, tornando difícil o seu reconhecimento como tal. Exemplos de heurísticas podem incluir:

- Todo módulo deve se comunicar com o mínimo possível de outros módulos.
- Toda informação a respeito de um módulo deve ser privativa do mesmo.
- Classes derivadas devem conhecer as suas classes base, por definição, mas classes base não devem ter qualquer tipo de conhecimento a respeito de suas classes derivadas.

Uma representação proposta para heurísticas é formada pelas seguintes informações: Título da heurística, sua descrição, a motivação, fontes bibliográficas, e os padrões e anti-padrões que são relacionados a ela.

2.2 – Padrões

Podemos pensar em um padrão como a reutilização da essência de uma solução para determinados problemas similares. Sintetizando as definições encontradas na literatura, podemos dizer que um padrão resolve um problema recorrente, em um determinado contexto, fornecendo uma solução que comprovadamente funcione, além de informar os resultados e compromissos da sua aplicação, e subsídios para que seja possível adaptar esta solução a uma variante do problema.

Ao contrário das heurísticas, os padrões disponíveis na literatura estão descritos de forma explícita e organizada. Embora existam várias formas de descrição de um padrão, de modo geral, a descrição de um padrão deve conter as seguintes informações: Nome do padrão, o problema, o contexto, a solução, as conseqüências, o uso conhecido e os padrões que são relacionados.

Segundo Buschmann [6], quanto maior o número de padrões em um sistema de padrões, maior é a dificuldade de entendê-los e utilizá-los. Um esquema de organização que permita agrupar padrões é importante para minimizar o esforço dos desenvolvedores em encontrar aqueles mais adequados para um determinado problema. Neste trabalho, estamos focalizando os padrões arquiteturais, ou seja, aqueles que visam apoiar o desenvolvimento de sistemas através da ênfase na organização estrutural do software, e os padrões de projeto, aplicados para refinar e estender a arquitetura fundamental do sistema.

2.3 – Anti-Padrões

Um anti-padrão pode resultar da falta de conhecimento de uma solução mais adequada, ou ainda da aplicação de um padrão (teoricamente, uma boa solução) no contexto incorreto. Os anti-padrões descrevem soluções inadequadas para um problema que resultam em uma situação ruim, ou então descrevem como sair de uma situação ruim e chegar a uma boa solução [7]. A presença de "bons" padrões em um sistema bem sucedido pode não ser suficiente. É preciso mostrar que estes padrões geralmente não ocorrem em sistemas mal sucedidos, e que determinadas construções inadequadas (anti-padrões) encontradas em sistemas mal sucedidos geralmente não estão presentes em sistemas bem sucedidos.

Brown [8] propõe uma estrutura para a descrição de anti-padrões, composta pelas seguintes seções: Nome do anti-padrão, sua forma geral, sintomas e conseqüências, causas típicas, variações, a nova solução indicada, algumas exceções conhecidas e as soluções que são relacionadas.

3 – Suporte a Padrões no Ambiente Odyssey

A partir da representação do conhecimento de projeto através dos conceitos descritos na seção anterior, foi implementado um conjunto de mecanismos em um ambiente de desenvolvimento de software visando apoiar a utilização deste conhecimento durante o projeto de software orientado a objetos. Nesta seção, apresentamos detalhes sobre a implementação e funcionamento destes mecanismos.

3.1 – Catálogo de Padrões

O Odyssey é um ambiente de desenvolvimento de software baseado em reutilização que oferece ferramentas para apoio automatizado tanto para o desenvolvimento para reutilização através da criação de modelos de domínios, quanto para o desenvolvimento de aplicações com reutilização a partir dos modelos de domínios criados [3]. Dentre as ferramentas disponíveis, podemos citar a de apoio a captura de conhecimento sobre domínios, de documentação de componentes, a navegação inteligente, a máquina de processos baseada em agentes, a engenharia reversa, e a ferramenta de diagramação em UML com sistema de críticas. Toda a implementação foi feita na linguagem de programação Java.

Como base dos mecanismos de suporte a padrões do ambiente Odyssey é necessário descrever e armazenar o conhecimento de projeto em que estamos interessados. Neste sentido, há um catálogo de padrões, anti-padrões e heurísticas que permitem ao usuário criar, armazenar e visualizar o conhecimento desses elementos em função da representação proposta para cada um deles. A representação para um padrão de projeto inclui, também, um diagrama de classes que descreve explicitamente a estrutura física do padrão catalogado. A base atual que forma o catálogo consta com alguns padrões de projetos descritos em [2], anti-padrões e heurísticas descritos em [4] e padrões arquiteturais descritos em [9].

A partir do catálogo de conhecimento de projeto descrito acima e ilustrado pela Figura 1, foram implementados mecanismos de suporte a padrões para apoiar a atividade de projeto de software no Odyssey: instanciação de padrões de projeto; seleção de padrões arquiteturais a partir de critérios de qualidade; e a detecção de padrões e anti-padrões.

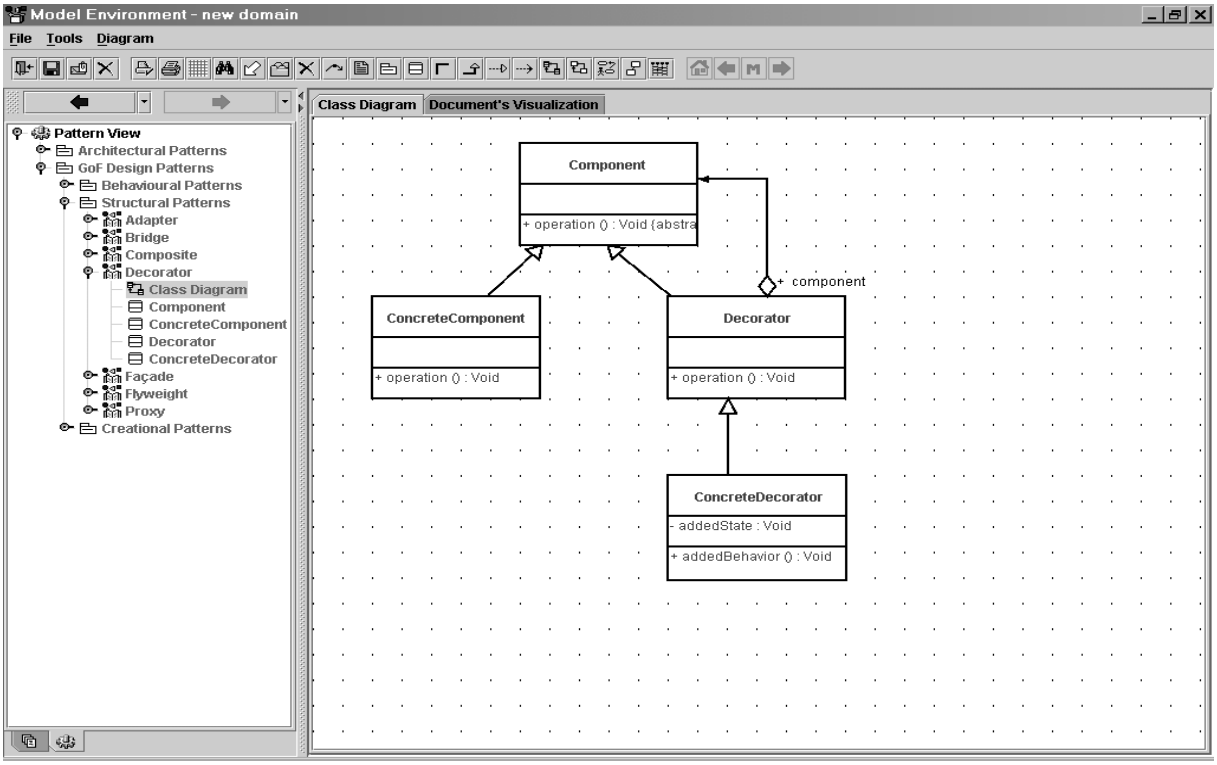

Figura 1 - Catálogo de Heurísticas, Padrões e Anti-Padrões no Ambiente Odyssey

3.2 – Instanciação de Padrões de Projeto

O mecanismo de instanciação de padrões de projeto procura oferecer ao usuário projetista uma forma de replicar uma solução com base em um padrão identificado para uma situação adequada ao seu projeto em desenvolvimento. Para isto, o padrão, conforme representado no catálogo, é transportado para o diagrama de classes do projetista. Este transporte é caracterizado pela cópia da estrutura de classes e relacionamentos que correspondem ao padrão.

Há duas formas de realizar a instanciação dos padrões de projeto: a absoluta e a relativa. Na instanciação absoluta, a estrutura do padrão é totalmente adicionado ao diagrama de classes do projetista, sem que haja nenhuma relação com algum elemento prévio do diagrama. Por outro lado, na instanciação relativa, o projetista tem a opção de associar partes já existentes do seu diagrama de projeto aos elementos constituintes do padrão desejado. Neste caso, o transporte do padrão para o diagrama do usuário procura se adequar aos elementos já existentes, complementando-os e criando novos elementos para que adquiram a estrutura conhecida do padrão.

3.3 – Seleção de padrões arquiteturais

Um dos aspectos críticos de se desenvolver software com ênfase arquitetural é a seleção de um estilo, ou padrão arquitetural [9]. Neste sentido, identificamos a oportunidade de apoiar a seleção de padrões arquiteturais utilizando uma abordagem orientada pela necessidade de se obter determinadas características de qualidade para o software a ser produzido. Estas características são baseadas na norma ISO/IEC 9126-1.

O suporte a padrões do ambiente Odyssey oferece um mecanismo para realização de uma comparação entre os requisitos de qualidade arquiteturais identificados para a aplicação e os obtidos pela utilização de cada padrão catalogado, com a intenção de se chegar a um indicador de padrão que melhor represente uma solução para o atendimento aos requisitos esperados.

A realização desta busca se baseia no conceito matemático de distância euclidiana. Consideramos que as características arquiteturais de qualidade são as dimensões do espaço vetorial e que cada padrão arquitetural é um elemento neste espaço dimensional. O vetor que representa cada padrão arquitetural se baseia na observação do aparente grau de esforço de desenvolvimento aplicado à fase de projeto arquitetural, utilizando o padrão avaliado em relação às características de qualidade. Analogamente, a representação vetorial do objetivo de projeto a ser atingido com a utilização dos padrões arquiteturais é baseada no valor da importância da ocorrência de cada característica de qualidade arquitetural no contexto específico de desenvolvimento em questão. A Figura 2 mostra como são feitas as avaliações.

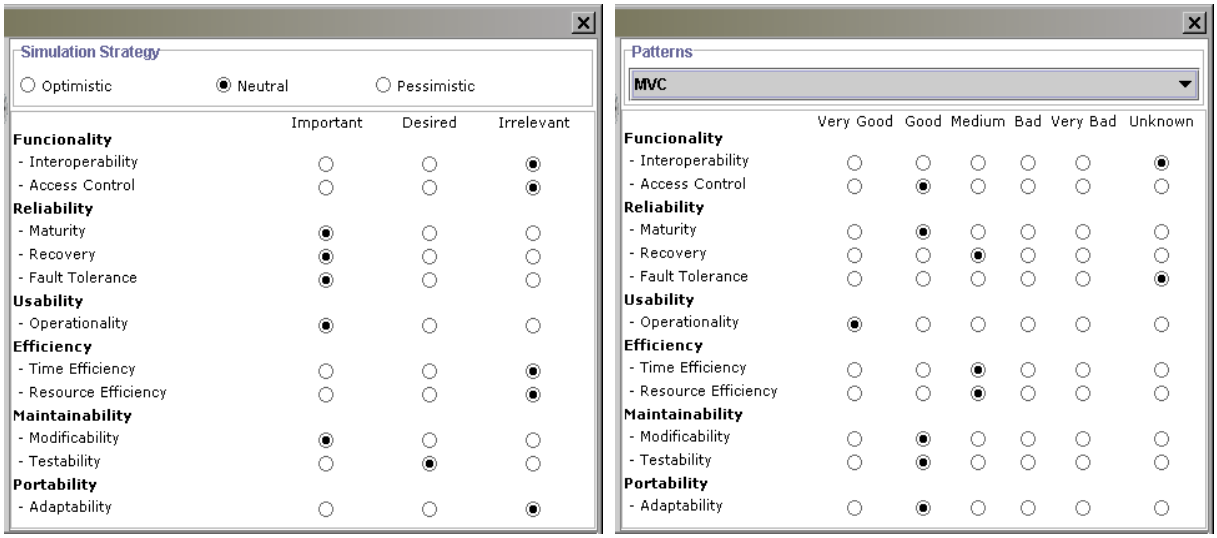

Figura 2 - Avaliação dos requisitos desejados e de um padrão arquitetural

O cálculo das distâncias entre o vetor correspondente a um padrão arquitetural e a especificação das características arquiteturais de referência permite que, ao fim deste processo, tenhamos uma classificação dos resultados, sendo possível a criação de uma ordenação que informa quais os padrões arquiteturais estão mais próximos da especificação desejada.

3.4 – Deteção de Padrões e Anti-Padrões

O suporte a padrões do ambiente Odyssey fornece um mecanismo de detecção de construções boas, assim como ruins. O principal objetivo desta detecção é fornecer suporte para a reestruturação de sistemas orientados a objetos legados, e também para a avaliação de sistemas ainda em desenvolvimento. A detecção de construções problemáticas (anti-padrões) permite que determinadas estruturas que comprometem uma futura expansão ou modificação possam ser detectadas e substituídas por outras mais adequadas. A detecção de construções típicas (padrões) permite que o projeto do sistema seja entendido em um nível maior de abstração, além de permitir uma avaliação sobre a utilização destes padrões no projeto.

A tarefa de identificar padrões e anti-padrões em um projeto orientado a objetos é muito difícil de ser realizada manualmente em função de vários motivos, dentre os quais:

- Sistemas legados de interesse e necessidade em reestruturação, normalmente, correspondem a sistemas de médio/grande porte.
- Geralmente, a única fonte segura de informação sobre o projeto está no código fonte. Modelos, quando existem, estão desatualizados ou são superficiais demais para a detecção da maior parte dos problemas. A análise manual do código fonte limita o escopo dos problemas que podem ser detectados de forma economicamente viável.
- Desenvolvedores muitas vezes não sabem que tipo de construção procurar. A existência de uma base de construções de projeto, englobando tanto padrões como anti-padrões, pode fornecer um suporte valioso nesta busca.

O mecanismo de detecção atua sobre um modelo orientado a objetos extraído pela ferramenta de engenharia reversa [10] ou sobre um modelo previamente construído no ambiente de modelagem. A partir do modelo é gerada uma representação Prolog, segundo o meta-modelo descrito por [4]. Aplicam-se as consultas (correspondentes a cláusulas Prolog) a esta representação do modelo. Uma visão geral da estrutura do mecanismo é ilustrada na Figura 3.

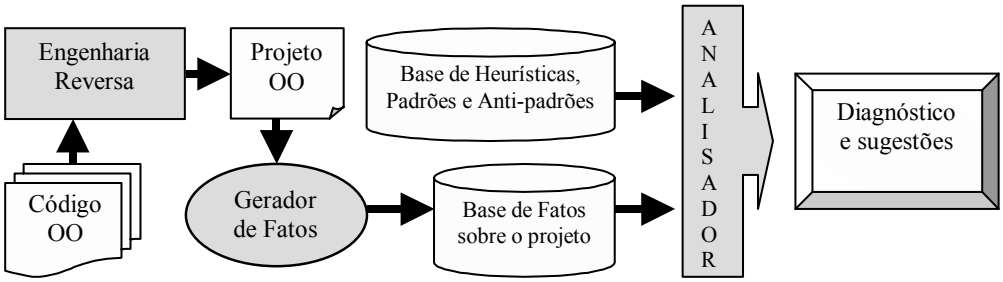

Figura 3 - Visão geral da estrutura do mecanismo de detecção

Seguindo o fluxo da figura, um modelo pode ser extraído com auxílio da ferramenta de engenharia reversa. Em seguida, é gerada, de maneira transparente, uma representação Prolog deste modelo. O usuário, então, seleciona os padrões, anti-padrões ou heurísticas que devem ser detectados no modelo. De acordo com os itens selecionados, a base de heurísticas, padrões e anti-padrões é consultada. Com o auxílio de uma máquina de inferência Prolog, o analisador detecta os elementos do modelo (classes, relacionamentos etc.) que manifestam os itens selecionados na detecção.

O mecanismo de detecção possibilita o relacionamento de padrões, anti-padrões e heurísticas de projeto. Um padrão pode ser solução para um determinado anti-padrão e/ou pode reforçar uma determinada heurística. Um anti-padrão pode violar uma determinada heurística. Desta forma, é importante que o usuário registre estes relacionamentos.

Dos três conceitos analisados, os padrões e anti-padrões são detectáveis diretamente no modelo extraído em fatos, pois possuem representação em cláusulas Prolog. As heurísticas não podem ser detectadas diretamente, por não possuírem representação formal (são apenas guias). Entretanto, uma heurística pode ser avaliada indiretamente pelas suas relações com os conceitos de padrões e anti-

padrões. Pode-se analisar, através do mecanismo de detecção, se uma heurística está sendo violada, caso um ou mais dos anti-padrões a ela relacionados sejam detectados.

4 – Conclusão

Os mecanismos de suporte ao desenvolvimento com padrões do ambiente Odyssey integram os conceitos de heurísticas, padrões e anti-padrões de projeto, de forma organizada e padronizada, tornando explícito os relacionamentos porventura existentes entre eles. A base construída através do catálogo possibilita a disseminação de um conhecimento hoje disperso, geralmente presente de forma implícita na literatura.

Entretanto, os mecanismos apresentam algumas limitações que podem ser alvo de melhorias futuras, principalmente no que diz respeito a uma ação integrada de todos os mecanismos. É importante, primeiramente, o levantamento de uma base mais extensa para formar o catálogo. A instanciação relativa de padrões pode ser ampliada no sentido de permitir que um conjunto de classes do usuário seja associado a um tipo de classe do padrão desejado, ao invés de ser uma relação de um para um. O mecanismo de seleção de padrões arquiteturais poderá estar diretamente ligado ao mecanismo de instanciação a partir do resultado da busca realizada. O mecanismo de detecção, por sua vez, poderá oferecer uma indicação explícita dos elementos participantes das heurísticas, padrões e anti-padrões avaliados dentro dos modelos de projeto do usuário, além da emissão de um relatório permitindo a navegação entre esses elementos.

5 – Referências Bibliográficas

[1] PFLEEGER, S.L., Design the System, In: S*oftware Engineering: Theory and Practice*, Prentice-Hall, 1998.

[2] GAMMA, E., HELM, R., JONHSON, R., *et alli.*, *Design Patterns – Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.

[3] WERNER, C., BRAGA, R., MATTOSO, M., *et alli. Odyssey: Estágio Atual.* Caderno de Ferramentas do XIV Simpósio Brasileiro de Engenharia de Software (XIV SBES). João Pessoa, PB, Brasil. Outubro, 2000.

[4] CORREA, A.L., *Uma Arquitetura de Apoio para Análise de Modelos Orientados a Objetos*, Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil. Julho, 1999.

[5] RIEL, A., *Object Oriented Design Heuristics*, Addison-Wesley, 1996.

[6] BUSCHMANN, F., MEUNIER R., ROHNERT, H., *et alli. Pattern-Oriented Software Architecture, A System of Patterns.* John Wiley & Sons, 1996.

[7] KOENIG, A., Patterns and antipatterns, In: *Journal of Object Oriented Programming*, 8(1), Março, 1995.

[8] BROWN, W., MALVEAU, R. McCORNICK III, MOWBRAY, T., *et alli*, *Anti-patterns – Refactoring Software, Architectures, and Projects in Crisis,* Wiley Computer Publishing, 1998.

[9] XAVIER, J.R, WERNER, C., TRAVASSOS, G. *Uma abordagem para a Seleção de Padrões Arquiteturas baseada em características de qualidade*, XVI Simpósio Brasileiro de Engenharia de Software (XVI SBES). Gramado, RS, Brasil. Outubro, 2002.

[10] VERONESE, G.O., CORREA, A., WERNER, C., NETTO, F.J., *ARES: Uma Ferramenta de Engenharia Reversa Java-UML*. Caderno de Ferramentas do XVI Simpósio Brasileiro de Engenharia de Software (XVI SBES). Gramado, RS, Brasil, Outubro, 2002.