Universidade São Paulo
IME
MAC5715 - Tópicos Avançados em Programação Orientada a Objetos
Professor Fábio Kon

Anti-Padrão de Desenvolvimento: Herança Maldita

Vinicius Gama Pinheiro
Mário Henrique Cruz

História-Evidência

- O que faz essa classe *Visitante*?

- Ah, ela herda vários métodos da superclasse pai, e outros que estão definidos na superclasse dessa superclasse, a classe avô... Mas esta última, na verdade, herda de duas outras classes, então... o que você perguntou mesmo?

- Esquece... Este outro método está definido onde?

- Hum, me lembro que ele foi declarado na superclasse *TipoDeUsuario*, porém não lembro se esta classe *TipoDeUsuario* ainda é necessária pois não sei ao certo o que está também está herdando, ou se outra classe extende funcionalidades dela. Eu não mexeria nisso."

Forma Geral

O anti-padrão *Herança Maldita* ocorre com frequência em sistemas cujas classes modelam entidades que possuem muitas especializações aninhadas, e/ou utilizam fortemente herança múltipla.

Geralmente é acarretado pelo mal uso da característica de herança por desenvolvedores com pouca experiência em sistemas orientados a objeto. O sistema acaba contando uma extensa árvore hierárquica cujas classes herdam muitos métodos, obrigando um constante "sobe e desce" pela árvore a fim de acompanhar o fluxo de execução. Essas classes apresentam comportamentos mais variados possíveis, o que acarreta na perda de coesão, em um forte acoplamento no sistema e uma enorme dificuldade em compreender a função de cada classe.

Devido a esta dificuldade em entender o código, algumas classes podem eventualmente conter código herdado cuja utilidade se restringe somente a classes superiores. Ou então podem existir classes que extendem funcionalidades que não são mais úteis ao sistema, como um subsistema de acesso a disquetes (em um sistema que não utilize mais disquetes) por exemplo, e mesmo assim permanecerem por anos no código.

Sintomas

- classes com muitos métodos herdados implementados em vários níveis hierárquicos;
- herança múltipla utilizada desmedidamente;
- grande profundidade e largura da árvore de herança;
- métodos herdados que parecem deslocados do escopo da classe.

Consequências

- dificuldade em saber onde um determinado método é definido na hierarquia de classes;
- dificuldade em entender as funcionalidades herdadas por uma classe;
- dificuldade em saber quais atributos são necessários para o funcionamento de quais métodos;
- devido ao forte acoplamento, é muito difícil manter o sistema, sem perder muito tempo "desvendando" as relações entre classes, superclasses, métodos, atributos.

Solução refatorada

É complicado estabelecer um conjunto de medidas que possam ser consideradas em todos os casos em que ocorrem este anti-padrão. A melhor solução neste caso é evitar que ele ocorra, procurando sempre ter uma clara noção da arquitetura do sistema e do papel de cada classe na hierarquia de heranças.

Durante a evolução do código, classes podem se tornar obsoletas a ponto de serem removidas, ou terem parte de suas funcionalidades movidas para classes superiores. Heranças múltiplas devem ser utilizadas com parcimônia de forma a evitar que as classes herdeiras perca a sua coesão. Utilizar delegação ao invés de herança é uma forma de manter a árvore com uma produndidade aceitável. Por fim, evitar o aprofundamento demasiado da árvore de heranças, desde que possível, é recomendável. Para tal é fortemente recomendável o uso de composição ao invés de herança, desde que se as interfaces das classes estejam bem codificadas e documentadas.[1]

Exceções Conhecidas

Versões iniciais de frameworks podem apresentar uma complexa estrutura na sua hierarquia de classes. Contudo, a medida que o framework evolui, ele deve ser refatorado para utilizar mais composições (*blackbox frameworks*) e menos heranças (*whitebox frameworks*).[2]

Anti-Padrões Relacionados

Lava Flow:[3] o *Herança Maldita* também pode ser encontrado em sistemas que começaram como experimentação e que, ao invés de refatorados desde o início, tiveram suas classes extendidas e acabaram indo para produção, mantendo relações equivocadas entre as classes novas e antigas. Classes e métodos sem utilidade presentes no *Herança Maldita* também podem ser encarados como fluxos de lava.

Referências

[1] Improving the Design of Exisiting Code – Martin Fowler
[2] Components, Frameworks, Patterns – Ralph E. Johnson
[3] www.antipatterns.com/lavaflow.htm (acessado às 16:00 em 23/10/2006)