Resumo: Hotspot Patterns: A Definição Formal e Detecção Automática de "Cheiros" de Arquitetura

Este artigo, escrito por Ran Mo, Yuanfang Cai, Rick Kazman e Lu Xiao, propõe e valida empiricamente um conjunto de "hotspot patterns" de arquitetura, que são problemas recorrentes em sistemas de software complexos e que resultam em altos custos de manutenção. Esses padrões são definidos com base na teoria de regras de design de Baldwin e Clark, combinando informações de arquitetura e histórico de evolução do projeto. O estudo mostra que a detecção desses padrões não só identifica os arquivos mais propensos a erros e mudanças, mas também aponta problemas de arquitetura específicos que podem ser a causa raiz desses problemas de manutenção.

Introdução

O artigo começa abordando a pesquisa existente em predição de defeitos e localização de problemas, que utiliza métricas estruturais e histórico de evolução para identificar falhas no código-fonte. Em um trabalho anterior, os autores introduziram o conceito de

Design Rule Space (DRSpace), um modelo de arquitetura que representa um sistema como um conjunto de regras de design e módulos. Eles observaram que os arquivos mais propensos a erros são geralmente altamente conectados arquitetonicamente e podem ser capturados por poucos DRSpaces. Além disso, notaram que esses DRSpaces problemáticos frequentemente contêm múltiplos problemas de arquitetura, que se repetem em diversos projetos.

Esses problemas recorrentes, ou "hotspot patterns," são a base do artigo. Eles são distintos de "code smells" ou "anti-patterns" porque, embora causem altos custos de manutenção, não são facilmente detectáveis por ferramentas existentes, como o Sonar[1]. A pesquisa se concentra em como os gerentes de projeto e arquitetos podem priorizar a manutenção e o refatoramento, identificando quais arquivos "complexos" realmente precisam ser corrigidos.

Hotspot Patterns Definidos

Com base na teoria de regras de design, os autores formalizaram cinco "hotspot patterns" que são frequentemente a causa de altos custos de manutenção:

1. **Unstable Interface:** Uma interface (regra de design) que é muito influente e muda frequentemente em conjunto com outros arquivos. As interfaces mais influentes, que são os "leading files" em um DRSpace, deveriam permanecer estáveis, mas neste padrão, elas estão envolvidas em um alto número de co-mudanças, o que as torna propensas a erros e mudanças.
2. **Implicit Cross-module Dependency:** Ocorre quando dois módulos que parecem estruturalmente independentes na arquitetura são alterados juntos frequentemente no histórico de revisões. Isso indica uma dependência oculta prejudicial que deveria ser eliminada.
3. **Unhealthy Inheritance Hierarchy:** Detecta hierarquias de herança que violam a teoria de regras de design ou o princípio de substituição de Liskov. Dois problemas comuns são: uma classe pai que depende de uma de suas classes filhas, ou uma classe cliente que depende tanto da classe base quanto de todas as suas classes filhas.
4. **Cross-Module Cycle:** Um ciclo de dependência onde os arquivos não pertencem todos ao mesmo módulo. Nem todos os ciclos são igualmente prejudiciais, mas os que ocorrem entre módulos de um DRSpace estão associados a mais defeitos.
5. **Cross-Package Cycle:** Um ciclo de dependência entre dois pacotes, o que é geralmente considerado prejudicial.

Unstable Interface e Implicit Cross-module Dependency são novos, enquanto os outros se baseiam em conceitos conhecidos, mas são formalizados usando a abordagem DRSpace.

Ferramenta de Suporte e Avaliação

Para detectar esses padrões, os autores criaram uma ferramenta chamada

Hotspot Detector. A ferramenta usa três tipos de arquivos de entrada: um DSM (Design Structure Matrix) com dependências estruturais (

SDSM), um DSM com informações de acoplamento evolutivo baseado no histórico de revisões (**HDSM**), e um arquivo de agrupamento de arquivos. A ferramenta, então, gera um resumo dos problemas de arquitetura e os arquivos envolvidos.

O estudo avaliou os padrões quantitativa e qualitativamente usando nove projetos de código aberto da Apache e um projeto comercial.

- **Análise Quantitativa:** Foram usadas quatro métricas para quantificar o esforço de manutenção: frequência de bugs (bugFreq), "churn" de bugs (bugChurn),

frequência de mudanças (changeFreq) e "churn" de mudanças (changeChurn). Os resultados mostraram que os arquivos envolvidos em "hotspots" tinham valores significativamente maiores para todas as métricas em comparação com os arquivos médios. Além disso, a correlação de Pearson mostrou uma forte dependência: quanto mais "hotspots" um arquivo estava envolvido, maior era o esforço de manutenção que ele causava. Dos quatro padrões em nível de arquivo, o

Unstable Interface e o **Cross-Module Cycle** foram os que mais contribuíram para a propensão a erros e mudanças.

- **Análise Qualitativa:** Um estudo de caso com uma empresa de software ("CompanyS") confirmou a praticidade da abordagem. O arquiteto do projeto confirmou que a ferramenta detectou a maioria dos problemas de arquitetura que causavam dor de cabeça na manutenção. O artigo ressalta que a visualização em DSMs forneceu orientação sobre como realizar o refatoramento. O arquiteto também destacou que a ferramenta identificou um problema de Implicit Cross-module Dependency que não era detectável por outras ferramentas.

Discussão e Conclusão

O artigo reconhece algumas limitações, como a dependência de Unstable Interface e Implicit Cross-module Dependency no histórico de evolução, a necessidade de ajustar thresholds, e o fato de terem avaliado apenas cinco padrões. No entanto, conclui que os "hotspot patterns" são ubíquos e que os detectados pela ferramenta estão fortemente correlacionados com altos custos de manutenção. A pesquisa fornece evidências de que quanto mais um arquivo está envolvido em problemas arquitetônicos, mais propenso a erros ele se torna.

O estudo é um passo importante, pois não só identifica os problemas, mas também oferece orientação sobre como corrigi-los, ligando diretamente a análise de arquitetura com a prática de refatoração.