**ANÁLISE DO ARTIGO**
**“HOTSPOT PATTERNS”**
**VINÍCIUS GOMES RODRIGUES**
**ENGENHARIA DE SOFTWARE - PUC MINAS**
**BELO HORIZONTE - 2025**

O artigo "Hotspot Patterns: The Formal Definition and Automatic Detection of Architecture Smells" apresenta uma pesquisa sobre problemas recorrentes em arquiteturas de software que causam altos custos de manutenção. Os autores propõem a identificação de "hotspots" - padrões arquiteturais problemáticos - através de uma combinação de análise estrutural e histórica do código. Embora o trabalho traga contribuições interessantes, apresenta também limitações significativas que merecem discussão.

A principal contribuição do artigo está na identificação e formalização de cinco padrões problemáticos em arquiteturas de software. Os pesquisadores desenvolveram uma ferramenta automatizada que analisa tanto a estrutura atual do software quanto seu histórico de modificações para detectar esses problemas.

Os padrões identificados incluem interfaces instáveis, dependências ocultas entre módulos, hierarquias de herança problemáticas e ciclos de dependência. A abordagem é inovadora por considerar não apenas como o software está organizado no momento, mas também como ele evoluiu ao longo do tempo.

Um aspecto forte do trabalho é a validação em múltiplos projetos de software real, incluindo tanto projetos de código aberto quanto um projeto comercial. Os resultados mostram correlação entre a presença desses padrões e problemas de manutenção, sugerindo que a abordagem pode ser útil na prática. A ferramenta desenvolvida parece oferecer visualizações que ajudam os desenvolvedores a entender não apenas onde estão os problemas, mas também suas possíveis causas. Isso é valioso para equipes que precisam priorizar esforços de refatoração.

No entanto, o artigo apresenta várias limitações. A abordagem depende fortemente da disponibilidade de histórico detalhado do projeto, o que pode não estar presente em todos os contextos de desenvolvimento. Além disso, os parâmetros usados para detectar os problemas parecem ser definidos de forma arbitrária, sem uma justificativa clara sobre como foram escolhidos.

Outra limitação importante é que o estudo não compara a abordagem proposta com ferramentas já existentes de análise de código. Fica difícil avaliar se o método realmente oferece vantagens significativas em relação às soluções já disponíveis no mercado.

A aplicabilidade da ferramenta também parece restrita a projetos em Java, não ficando claro se funcionaria igualmente bem em outras linguagens de programação ou em arquiteturas diferentes, como microsserviços.

O artigo apresenta uma ideia interessante para identificar problemas arquiteturais em software, mas ainda parece estar em estágio inicial de desenvolvimento. A abordagem tem potencial, mas necessitaria de mais refinamentos para se tornar amplamente útil na indústria.