PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

ENGENHARIA DE SOFTWARE

ANÁLISE COMPARATIVA DE MANUTENIBILIDADE: PLATAFORMAS LOW-CODE OPEN SOURCE VS. FRAMEWORKS WEB TRADICIONAIS

DAVI FERREIRA

BELO HORIZONTE

PANORAMA DAS AMEAÇAS À VALIDADE

Como não posso controlar o ambiente, a defesa se baseia em seleção rigorosa e estatística robusta.

CONSTRUCTO (MEDIÇÃO)

Estamos medindo o conceito correto?

INTERNA (CAUSA)

A relação de causa e efeito é clara?

EXTERNA (GENERALIZAÇÃO)

Os resultados podem ser aplicados em outros casos?

CONCLUSÃO (ESTATÍSTICA)

Os dados estatísticos são confiáveis?

VALIDADE DE CONSTRUCTO

MTTR (Mean Time to Repair) inclui tempo de espera, finais de semana, burocracia - Inconsistência nas Labels de Bug

VALIDADE DE CONSTRUCTO

O desafio é garantir que estamos medindo a coisa certa.

Mitigação: Usar a Triangulação de Métricas:

- Code Churn (linhas alteradas)
- Files Changed (quantos arquivos foram mexidos)
- Reopen Rate (bugs que voltaram)

Auditoria manual nas labels dos repositórios para garantir que apenas defeitos reais sejam analisados.

Manutenibilidade = facilidade e rapidez de diagnóstico e correção de defeitos

Medida por: combinação de tempo, esforço de código e estabilidade.

VALIDADE INTERNA

Viés de Seleção - Fator de Confusão: Tamanho/atividade da comunidade - Fatores Históricos

VALIDADE INTERNA

Risco de confundir a causa. É a tecnologia que Causa o efeito ou outro fator?

Mitigação: Selecionar apenas repositórios > 10K estrelas.

Meço explicitamente fatores de confusão como **Time to First Response** e **External Contributor Ratio**.

Exemplo: Se bugs no Low-Code são corrigidos mais lentamente, é porque Low-Code é difícil?

Ou porque React tem 100k devs enquanto Appsmith tem 10k, então há mais gente para consertar?

VALIDADE EXTERNA (GENERALIZAÇÃO)

Open Source vs Proprietário - JavaScript/TypeScript - Limitação de Generalização

VALIDADE EXTERNA

Existe uma ameaça de que os resultados não se apliquem a ferramentas privadas, que têm arquiteturas internas diferentes.

Mitigação: Delimitação Clara do Escopo: Conclusões serão válidas para o ecossistema de ferramentas Open Source baseadas em JS.

- Plataformas **Open Source** (Appsmith, Budibase)
- Ecossistema **JavaScript/TypeScript** (React, Vue, etc.)
- Projetos com mais de **10.000 estrelas**
- Dados extraídos do **GitHub**

VALIDADE DE CONCLUSÃO

Viés de Seleção - Tamanho/atividade da comunidade - Fatores Históricos

VALIDADE DE CONCLUSÃO

Tempo de resolução de bugs não seguem uma curva normal;

Mitigação: Usar Estatística Não-Paramétrica (teste Mann-Whitney U), que é robusta para esses casos.

Estatísticas Robustas: em vez de média (afetada por extremos), uso mediana e quartis.

Exemplo: dados de tempo de resolução não são normais. Imagina: 80% dos bugs são rápidos, mas 20% ficam abertos por meses.

Isso é uma distribuição enviesada.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

SÍNTESE

Identifiquei e mitigarei ameaças em todas as quatro categorias de validade:

BELO HORIZONTE

VALIDADE DE CONSTRUCTO

Uso triangulação de métricas (MTTR + Code Churn + Files Changed + Reopen Rate). Auditoria manual das labels e de 5% dos dados para garantir que 'bug' é bug mesmo

VALIDADE INTERNA

Seleciono apenas repositórios com >10k estrelas para equiparar comunidades. Meço explicitamente fatores de confusão como Time to First Response e External Contributor Ratio.

VALIDADE EXTERNA

Deixo claro no relatório que os resultados são válidos apenas para Low-Code Open Source em JavaScript com >10k estrelas.

VALIDADE DE CONCLUSÃO

Coleto 1.000 issues por grupo para poder estatístico elevado (>80%). Uso testes não-paramétricos (Mann-Whitney U) ao invés de T-test que assume normalidade. Mediana ao invés de media.