Arquitetura de Software Baseada em Riscos (Risk-Driven Approach)

Como especialistas, compreendemos que o sucesso de um sistema não reside apenas na sua funcionalidade, mas na capacidade do arquiteto em equilibrar o esforço de design com as incertezas inerentes ao projeto. Projetar em excesso é um desperdício de capital, mas ignorar o design é um convite ao desastre.

A engenharia só alcança o sucesso ao antecipar e evitar sistematicamente os modos de falha

Na engenharia, definimos risco através da fórmula fundamental: $Risco = Probabilidade\ de\ Falha \times Impacto\ da\ Falha$

A Metodologia do "Risk-Driven Approach"

A metodologia divide-se em três passos cíclicos:

1. **Identificar e Priorizar Riscos:** Determinar o que pode falhar e qual a gravidade de cada cenário.
2. **Selecionar e Aplicar Técnicas:** Escolher ferramentas de design (modelação, prototipagem, padrões) que ataquem diretamente esses riscos.
3. **Avaliar a Redução de Risco:** Validar se o risco foi mitigado para um nível aceitável.

Taxonomia e Tipologia de Riscos

Os riscos dividem-se em duas grandes categorias:

- **Riscos de Gestão de Projetos:** Envolvem a viabilidade logística.
- **Riscos de Engenharia de Software:** Focam-se na integridade técnica, como a incapacidade de escala do servidor ou bugs em algoritmos críticos. Além destes, devemos estar atentos aos Riscos Prototípicos de cada domínio, o que nos permite antecipar problemas antes que eles surjam:
- **Domínio IT:** Foca-se em saber se estamos a resolver o problema real (o risco de "solução errada"), na integração com sistemas legados e na modifiabilidade do software.
- **Sistemas de Engenharia:** Os riscos críticos são a concorrência, a fiabilidade e a composição de subsistemas complexos.
- **Web:** Domina a preocupação com a segurança, escalabilidade massiva e produtividade/expressividade do desenvolvedor.

Mitigação de Riscos

Uma organização de engenharia madura MUST (deve obrigatoriamente) manter um "Handbook de Técnicas" que correlacione problemas com soluções.

As técnicas dividem-se em:

- **Técnicas de Análise:** Como cálculos de stress, análise térmica ou modelação de throughput. Pense no exemplo histórico das aeronaves Comet: a falha catastrófica deveu-se à concentração de stress nos cantos das janelas quadradas. Uma análise de "linhas de stress" (como as que vemos em modelos de carga) teria revelado que janelas redondas dissipam a força, mitigando o risco de fadiga do metal. No software, fazemos o mesmo ao modelar a carga para evitar o colapso do sistema.
- **Técnicas de Solução:** Como o uso de **Design Patterns**, prototipagem ou o uso de Botaréus (Flying Buttresses). Tal como as catedrais góticas usam botaréus para suportar o peso das paredes e evitar que estas abram sob pressão, nós usamos padrões arquiteturais para dar suporte estrutural ao sistema contra riscos de performance ou segurança.

Se o risco é nulo, o design deve ser zero. Se o risco é extremo, o design deve ser exaustivo.

Estilos de Design Arquitetural: NDUF, BDUF e EDUF

A forma como aplicamos o design depende do estilo adotado, que pode ser ilustrado pela analogia do desenvolvimento de um veículo:

1. **Evolutionary Design (NDUF - No Up-Front):** É como um jardim ou uma árvore que cresce organicamente. Não há plano inicial; o design surge da implementação. Embora práticas como TDD e Refactorização ajudem a controlar o caos, em sistemas de grande escala este estilo é perigoso.
2. **Planned Design (BDUF - Big Up-Front):** É como desenhar um carro com todos os detalhes técnicos (motor de 6 cilindros, estofos em pele, consumos exatos) ou uma ponte antes de colocar o primeiro tijolo. É útil para equipas em paralelo, mas **o excesso de detalhe inicial pode ignorar o feedback do código real**. Arquitetura detalhada antes da construção
3. **Minimal Planned Design (EDUF/LDUF - Enough/Little Up-Front):** É o equilíbrio pragmático. Desenhamos o "esboço" do carro (capacidade para 6 pessoas, autonomia >500km, velocidade >80km/h) para garantir que a estrutura base suporta os maiores riscos. O resto — a cor, os acabamentos — é decidido evolutivamente.