Software Architecture: A Roadmap - Quando a Visão Encontra a Realidade

David Garlan escreveu "Software Architecture: a Roadmap" no ano 2000, mas suas previsões sobre o futuro da arquitetura de software soam quase proféticas hoje. O que mais impressiona não é sua capacidade de antever tendências, mas como ele conseguiu capturar os dilemas fundamentais que ainda enfrentamos.

Garlan começa definindo arquitetura de software como a "estrutura bruta" de um sistema - uma descrição que parece simples, mas carrega peso. Para ele, arquitetura não é apenas sobre componentes e suas conexões, mas sobre decisões de design de alto nível que definem propriedades críticas como performance, confiabilidade e escalabilidade. É uma visão que vai além do técnico e abraça o estratégico.

O artigo mapeia seis papéis fundamentais da arquitetura: entendimento, reuso, construção, evolução, análise e gerenciamento. Cada um ressoa com experiências práticas. Quantas vezes não vimos sistemas que funcionam, mas ninguém consegue explicar como? Ou projetos onde a arquitetura virou uma caixa-preta que só o arquiteto original entendia?

A retrospectiva que Garlan faz dos "velhos tempos" - há apenas dez anos antes de 2000 - é quase cômica. Diagramas de caixas e linhas informais, escolhas arquiteturais idiossincráticas, impossibilidade de análise formal. Parece idade da pedra, mas muitas organizações ainda operam dessa forma hoje.

A mudança no equilíbrio build-versus-buy foi visionária. Em 2000, a pressão por time-to-market já estava forçando empresas a usar mais código externo do que desenvolver internamente. Garlan previu que muitas empresas se tornariam essencialmente integradoras de sistemas, escrevendo principalmente "código cola". Hoje, com microservices, APIs e cloud computing, isso é nossa realidade.

A computação pervasiva, com milhares de dispositivos heterogêneos, acabou se materializando através da IoT. Garlan já identificava os desafios fundamentais: gerenciamento automático de recursos, reconfiguração dinâmica, e mobilidade de usuários. São exatamente os problemas que edge computing e IoT enfrentam hoje.

Garlan acertou na importância crescente de padrões arquiteturais e linguagens de descrição de arquitetura (ADLs). Embora suas ADLs específicas (Wright, Aesop, Acme) não tenham dominado o mercado, a necessidade de formalizar arquiteturas se concretizou através de ferramentas como Terraform, Kubernetes YAML, e Infrastructure as Code.

A previsão sobre product lines e padrões de integração foi certeira. Vemos isso hoje em plataformas como Salesforce, AWS, e ecossistemas de APIs. A discussão sobre arquiteturas que suportam billing, segurança, e composição por usuários finais antecipou o modelo SaaS.

Algumas lacunas são compreensíveis para a época. Garlan não previu o impacto transformador do open source, que mudou drasticamente a dinâmica build-versus-buy. Também subestimou como linguagens de programação evoluiriam para abraçar conceitos arquiteturais - veja Go com goroutines, Rust com ownership, ou TypeScript com interfaces.

O valor duradouro do artigo não está nas previsões específicas, mas nos princípios que estabelece. Garlan entendeu que arquitetura é fundamentalmente sobre trade-offs conscientes. Não existe arquitetura "certa" ou "errada" - existem escolhas que se alinham melhor com restrições e objetivos específicos.

A ênfase na separação entre concerns funcionais e conectividade continua relevante. Microservices exemplificam isso: mantemos lógica nos endpoints (serviços) e usamos protocolos simples para comunicação. É a mesma filosofia de "endpoints inteligentes, pipes simples" que seria popularizada mais tarde.

A discussão sobre architectural mismatch permanece atual. Garlan reconheceu que componentes desenvolvidos independentemente raramente funcionam bem juntos sem adaptação. Container technologies e service meshes são respostas modernas a esse problema.

Algumas previsões foram excessivamente otimistas. Garlan esperava que usuários finais pudessem fazer composição de sistemas com garantias de funcionamento. Na prática, isso se mostrou mais complexo - mesmo desenvolvedores experientes lutam com dependências e compatibilidade.

A discussão sobre análise formal de arquiteturas, embora tecnicamente sólida, não considerou suficientemente as realidades organizacionais. Ferramentas formais exigem investimento em treinamento e disciplina que muitas organizações não conseguem sustentar.

O artigo também subestima a importância do fator humano. Conway's Law - que Garlan menciona de passagem - acabou sendo mais determinante para arquiteturas do que muitas das considerações técnicas que ele enfatiza.

Relendo Garlan hoje, o mais impressionante é sua compreensão de que arquitetura de software não é apenas uma disciplina técnica, mas uma ponte entre requisitos de negócio e implementação. Essa visão holística continua sendo o diferencial entre arquitetos que meramente desenham diagramas e aqueles que realmente influenciam o sucesso de sistemas.

Suas previsões sobre a crescente importância de padrões, a necessidade de automação pesada, e os desafios de sistemas distribuídos se materializaram, mesmo que através de tecnologias diferentes das que ele imaginou. O roadmap que ele traçou não foi seguido literalmente, mas os destinos que ele identificou foram alcançados por caminhos alternativos.

A maior lição é que arquitetura de software evoluiu de uma atividade ad hoc para uma disciplina com princípios, padrões e ferramentas estabelecidos. Não chegamos ao ponto de

ser uma "engenharia madura" como Garlan esperava, mas definitivamente não estamos mais na idade das pedra dos diagramas informais.

Para quem trabalha com arquitetura hoje, o artigo de Garlan oferece perspectiva histórica valiosa sobre como chegamos onde estamos e lembretes sobre princípios fundamentais que transcendem modas tecnológicas. É uma leitura que muda como você pensa sobre o próprio papel da arquitetura no desenvolvimento de software.