Resenha: Software Architecture: a Roadmap

O artigo "Software Architecture: a Roadmap", de David Garlan, publicado em 2000, oferece uma visão abrangente do campo da arquitetura de software no final do século XX, destacando seu progresso, desafios e aspirações futuras. Garlan argumenta que, embora a arquitetura de software tenha ganhado atenção significativa como um subcampo importante da engenharia de software, ainda há muito a ser feito para que se estabeleça como uma disciplina de engenharia madura.

Os Papéis da Arquitetura de Software

Garlan inicia o artigo descrevendo os múltiplos papéis que a arquitetura de software desempenha no desenvolvimento de sistemas. Ele a posiciona como uma ponte crucial entre os requisitos e a implementação. Os principais papéis incluem:

- **Compreensão:** A arquitetura simplifica a capacidade de entender grandes sistemas, apresentando-os em um nível de abstração que facilita a compreensão do design de alto nível. Ela expõe as restrições de alto nível e a lógica por trás das escolhas arquitetônicas.
- **Reutilização:** A arquitetura suporta a reutilização em múltiplos níveis, não apenas de componentes, mas também de frameworks e padrões de design.
- **Construção:** Uma descrição arquitetônica fornece um projeto parcial para o desenvolvimento, indicando os principais componentes e suas dependências. Ela documenta limites de abstração e restringe como as partes do sistema podem depender dos serviços fornecidos por outras partes.
- **Evolução:** A arquitetura pode expor as dimensões ao longo das quais um sistema deve evoluir, permitindo que os mantenedores compreendam as ramificações das mudanças e estimem com mais precisão os custos de modificação. A separação entre componentes e mecanismos de interação facilita a adaptação a novas preocupações, como desempenho e interoperabilidade.

- **Análise:** Descrições arquitetônicas abrem novas oportunidades para análise, incluindo verificação de consistência do sistema, conformidade com restrições de estilo arquitetônico, conformidade com atributos de qualidade, análise de dependência e análises específicas de domínio.
- **Gerenciamento:** O sucesso de projetos muitas vezes depende do reconhecimento de uma arquitetura de software viável como um marco fundamental. A avaliação crítica da arquitetura leva a uma compreensão mais clara dos requisitos, estratégias de implementação e riscos potenciais.

O Estado da Arte: Ontem e Hoje (até 2000)

Garlan contrasta o estado da arquitetura de software "ontem" (uma década antes da publicação do artigo) com o "hoje". No passado, a arquitetura era em grande parte um assunto *ad hoc*, com descrições informais e pouca capacidade de análise formal. As escolhas arquitetônicas eram idiossincráticas, e a transmissão do conhecimento era difícil.

"Hoje", no entanto, houve avanços significativos. A arquitetura tornou-se uma atividade de design mais visível e explícita. Três avanços importantes são destacados:

Linguagens e Ferramentas de Descrição de Arquitetura (ADLs)

A informalidade dos diagramas de caixa e linha levou a problemas de clareza e análise. Em resposta, surgiram as ADLs (Architecture Description Languages), que fornecem um framework conceitual e uma sintaxe concreta para caracterizar arquiteturas de software. Exemplos incluem Adage, Aesop, C2, Darwin, Rapide, SADL, UniCon, Meta-H e Wright. Essas linguagens vêm acompanhadas de ferramentas para parsing, exibição, compilação, análise e simulação de descrições arquitetônicas. Garlan menciona o Acme como uma linguagem de intercâmbio arquitetônico, vista como o "XML da descrição arquitetônica".

Linhas de Produto e Padrões

A busca por explorar a comunalidade entre múltiplos produtos levou ao desenvolvimento de linhas de produto e padrões de integração entre fornecedores. As arquiteturas de linha de produto exigem uma arquitetura reutilizável que pode ser instanciada para cada produto. Padrões de integração entre fornecedores, como HLA

(High Level Architecture) para Simulação Distribuída e EJB (Enterprise JavaBeans) da Sun, fornecem frameworks arquitetônicos que permitem a configuração de uma ampla variedade de sistemas específicos.

Codificação e Disseminação

A falta de um corpo de conhecimento compartilhado era um impedimento. Livros e cursos sobre design arquitetônico ajudaram a codificar e disseminar estilos arquitetônicos padrão. Um estilo arquitetônico especifica um vocabulário de design, restrições de uso e suposições semânticas (ex: pipe-and-filter, blackboard, cliente-servidor, orientado a eventos, orientado a objetos). O reconhecimento do valor de padrões bem documentados também contribuiu para a maturidade do campo.

O Futuro: Tendências Emergentes e Desafios (até 2010)

Garlan especula sobre as tendências emergentes e os desafios para a arquitetura de software na década seguinte. Ele identifica três tendências proeminentes:

Mudança no Equilíbrio entre Construir e Comprar

A pressão econômica para reduzir o tempo de lançamento no mercado está mudando drasticamente o equilíbrio entre construir software internamente e comprar componentes ou sistemas prontos. As empresas estão se tornando mais integradoras de sistemas do que desenvolvedoras de software, usando milhares de linhas de código externo para cada linha que escrevem. Isso aumenta a necessidade de padrões industriais (como COM, JavaBeans, CORBA) e leva a novos processos de subcontratação de software, onde a conformidade arquitetônica é crucial.

Computação Centrada em Rede

A transição de um modelo computacional centrado no PC para um modelo centrado em rede (como a Internet) apresenta novos desafios. Sistemas abertos, como a Internet, carecem de controle centralizado e exigem arquiteturas que escalem para o tamanho e a variabilidade da rede. A necessidade de suportar coalizões dinamicamente formadas de recursos autônomos e a integração de provedores de

serviços de aplicativos comerciais (ASPs) exigirão novas técnicas para gerenciar modelos arquitetônicos em tempo de execução e avaliar suas propriedades.

Computação Ubíqua (Pervasive Computing)

A tendência em direção à computação ubíqua, onde uma vasta variedade de dispositivos heterogêneos (torradeiras, carros inteligentes, etc.) povoa o universo computacional, trará desafios relacionados ao uso crítico de recursos, maior flexibilidade arquitetônica para lidar com dispositivos que aparecem e desaparecem, e melhor gerenciamento da mobilidade do usuário. As arquiteturas precisarão fornecer controle mais automatizado sobre os serviços computacionais.

Conclusão

Garlan conclui que o campo da arquitetura de software experimentou um crescimento considerável e continuará a crescer. À medida que amadurece como disciplina de engenharia, enfrentará desafios significativos. Muitas soluções virão da disseminação e maturação das práticas e tecnologias existentes, mas as mudanças no cenário da computação exigirão inovações significativas. O artigo serve como um roteiro valioso para entender a evolução e as direções futuras da arquitetura de software, enfatizando a necessidade contínua de pesquisa e prática disciplinada para atender às demandas de sistemas cada vez mais complexos e interconectados.