



Buscar

comentários post favorito (4)

Arquitetura de Software: Desenvolvimento orientado para arquitetura

Artigo da Revista Engenharia de Software.



Curtir 3

Gostei (16) (2)

[Esse artigo faz parte da revista Engenharia de Software edição especial. Clique aqui para ler todos os artigos desta edição](#)



Projeto

Arquitetura de Software

Desenvolvimento orientado para arquitetura

Software, de modo genérico, é uma entidade que se encontra em quase constante estado de mudança. As mudanças ocorrem por necessidade de corrigir erros existentes no software ou de adicionar novos recursos e funcionalidades. Igualmente, os sistemas computacionais (isto é, aqueles que têm software como um de seus elementos) também sofrem mudanças frequentemente. Essa necessidade evolutiva do sistema de software o torna 'não confiável' e predisposto a defeitos, atraso na entrega e com custos acima do estimado. Concomitante com esses fatos, o crescimento em tamanho e complexidade dos sistemas de software exige que os profissionais da área raciocinem, projetem, codifiquem e se comuniquem por meio de componentes de software. Como resultado, qualquer concepção ou solução de sistema passa então para o nível arquitetural, onde o foco recai sobre os componentes e relacionamentos entre eles num sistema de software.

Arquitetura de software

Quase cinco décadas atrás software constituía uma insignificante parte dos sistemas existentes e seu custo de desenvolvimento e manutenção eram desprezíveis. Para perceber isso, basta olharmos para a história da indústria do software (ver seção Links). Encontramos o uso do software numa ampla variedade de aplicações tais como

sistemas de manufatura, software científico, software embarcado, robótica e aplicações Web, dentre tantas. Paralelamente, observou-se o surgimento de várias técnicas de modelagem e projeto bem como de linguagens de programação. Perceba que o cenário existente, décadas atrás, mudou completamente.

Antigamente, os projetos de sistemas alocavam pequena parcela ao software. Os componentes de hardware, por outro lado, eram analisados e testados quase exaustivamente, o que permitia a produção rápida de grandes quantidades de subsistemas e implicava em raros erros de projetos. Entretanto, a facilidade de modificar o software, comparativamente ao hardware, tem servido como motivador para seu uso. Além disso, a intensificação do uso do software numa larga variedade de aplicações o fez crescer em tamanho e complexidade. Isto tornou proibitivo analisá-lo e testá-lo exaustivamente, além de impactar no custo de manutenção.

Um reflexo dessa situação é que as técnicas de abstração utilizadas até o final da década de 1980 (como decomposição modular, linguagens de programação de alto nível e tipos de dados abstratos) já não são mais suficientes para lidar com essa necessidade.

Diferentemente do uso de técnicas que empregam algoritmos e estruturas de dados e das linguagens de programação que implementam tais estruturas, o crescimento dos sistemas de software demanda notações para conectar componentes (módulos) e descrever mecanismos de interação, além de técnicas para gerenciar configurações e controlar versões. A **Tabela 1** apresenta o contexto da arquitetura de software. Na programação estruturada, é feito uso de estruturas de seqüência, decisões e repetições como 'padrões' de controle nos programas. Já a ocultação de informações é um recurso do paradigma orientado a objetos que permite ao programador, por exemplo, ocultar dados tornando-os seguros de qualquer alteração accidental. Além disso, na programação orientada a objetos, dados e funções podem ser 'encapsulados' numa entidade denominada objeto, o que resulta em mais simplicidade e facilidade na manutenção de programas. Por outro lado, os estilos arquiteturais capturam o 'padrão' de organização dos componentes de software num programa, caracterizando a forma na qual os componentes comunicam-se entre si.

--	--	--

Abordagem	Foco	Padrões
Programação estruturada	Sistemas de pequeno porte	Estruturas de controle
Abstração e modularização	Sistemas de médio porte	Encapsulamento e ocultação de informações
Componentes e conectores	Sistemas de grande porte	Estilos arquiteturais

Tabela 1. Contexto da arquitetura de software.

Perceba que a categorização, apresentada na **Tabela 1**, teve o objetivo de capturar uma visão geral de abordagens aplicadas a sistemas de software. Nada impede, por exemplo, o uso da programação estruturada em sistemas de grande porte ou da ênfase de um estilo arquitetural num sistema de pequeno porte. Entretanto, essa prática não é comum.

Note que à medida que tamanho e complexidade dos sistemas de software aumentam, o problema de projeto extrapola as estruturas de dados e algoritmos de computação. Ou seja, projetar a arquitetura (ou estrutura geral) do sistema emerge como um problema novo. Questões arquiteturais englobam organização e estrutura geral de controle, protocolos de comunicação, sincronização, alocação de funcionalidade a componentes e seleção de alternativas de projeto. Por exemplo, nos sistemas Web, uma solução que tem sido empregada faz uso de múltiplas camadas separando componentes cliente, servidores de aplicações, servidores Web e outras aplicações (que possam ter acesso a esse sistema). Essa estruturação em camadas objetiva facilitar a alocação da funcionalidade aos componentes. O uso de camadas oferece suporte à flexibilidade e portabilidade, o que resulta em facilidade de manutenção. Outro aspecto a destacar da arquitetura em camadas é o uso de interfaces padrões visando facilitar reuso e manutenção. Interfaces bem definidas encapsulam componentes (com

funcionalidades definidas) já testados, prática que permite o reuso e também auxilia na manutenção, já que toda e qualquer alteração necessária estaria confinada àquele componente.

Importância da Arquitetura de software

Todos esses fatores compreendem o projeto no nível arquitetural e estão diretamente relacionados com a organização do sistema e, portanto, afetam os atributos de qualidade (também chamados de requisitos não funcionais) como desempenho, portabilidade, confiabilidade, disponibilidade, entre outros. Se fizermos uma comparação entre arquitetura de software (caracterizada, por exemplo, pelo estilo em camadas) e arquitetura 'clássica' (relativa à construção de edificações), podemos observar que o projeto arquitetural é determinante para o sucesso do sistema.

A **Tabela 2** destaca aspectos da representação de projeto que captura elementos característicos da arquitetura enquanto que as restrições estão associadas a atributos de qualidade e, portanto, servem como determinantes nas decisões do projeto arquitetural. Por exemplo, embora o uso de múltiplas camadas facilite a manutenção de um sistema de software, também contribui para degradar o desempenho do sistema. Uma tática tem sido reduzir o nível de acoplamento entre componentes para não comprometer o desempenho do sistema. Dessa forma, se adotarmos uma redução no nível de acoplamento dos componentes, eles terão menor necessidade de comunicação entre si, o que resulta num melhor desempenho.

<i>Categorias arquiteturais</i>	<i>Representações de projeto</i>	<i>Restrições</i>
Arquitetura Clássica	Modelos, desenhos, planos, elevações e perspectivas	Padrão de circulação, acústica, iluminação e ventilação
Arquitetura de Software	Modelos para diferentes papéis, múltiplas visões	Desempenho, confiabilidade,

Tabela 2. Comparação de aspectos arquiteturais.

Hoje em dia, os processos de engenharia de software requerem o projeto arquitetural de software. Por quê?

- É importante poder reconhecer as estruturas comuns existentes de modo que arquitetos de software (ou engenheiros de software realizando o papel de arquiteto de software – conforme **Tabela 3**) possam entender as relações existentes nos sistemas em uso e utilizar esse conhecimento no desenvolvimento de novos sistemas.
- O entendimento das arquiteturas permite aos engenheiros tomarem decisões sobre alternativas de projeto.
- Uma especificação arquitetural é essencial para analisar e descrever propriedades de um sistema complexo, permitindo o engenheiro ter uma visão geral completa do sistema.
- O conhecimento de notações para descrever arquiteturas permite engenheiros comunicarem novos projetos e decisões arquiteturais tomadas a outros membros da equipe.

Cabe destacar que, para que haja o entendimento da arquitetura, faz-se necessário ao engenheiro de software conhecer os estilos arquiteturais existentes, conforme apresentado adiante. As propriedades de cada arquitetura, portanto, são dependentes do estilo arquitetural adotado. Por exemplo, o uso de uma notação padrão como a UML ajuda na representação de componentes e compartilhamento de informações do projeto.

Esses aspectos servem como indicadores de uma maturidade inicial de engenharia de software. Outros aspectos compreendem uso e reuso de soluções existentes no desenvolvimento de novos sistemas. Para tanto, a prototipagem tem sido usada em

projetos de natureza inovadora (bem antes da implementação ou aceitação de um produto acontecer).

Além disso, o aumento da complexidade e quantidade de requisitos dos sistemas dificulta cada vez mais atender às restrições de orçamento e cronograma. Atualmente, empresas têm procurado incorporar estratégia de reuso de software, enfatizando o reuso centrado na arquitetura para obter melhores resultados no desenvolvimento de sistemas. Note que a arquitetura de software serve como uma estrutura através da qual se tem o entendimento dos componentes de um sistema e de seus inter-relacionamentos. Em outras palavras, ela define a estrutura do sistema, de modo consistente para implementações, já que está diretamente relacionada aos atributos de qualidade como confiabilidade e desempenho. A organização dos componentes num sistema de software impacta sobre a qualidade apresentada por ele. Por exemplo, a adoção de uma arquitetura em camadas serve para modularizar o sistema bem como facilitar modificações. Entretanto, um número elevado de camadas (4 ou 5) pode degradar o desempenho do sistema se houver um elevado grau de acoplamento entre os componentes.

Diversos benefícios decorrem da incorporação da arquitetura de software como 'elemento norteador' do processo de desenvolvimento de software. Cabe destacar que a arquitetura pode:

- Prover suporte ao reuso – seus componentes definidos e testados podem ser reaproveitados em novas aplicações.
- Servir de base à estimativa de custos e gerência do projeto – a existência de uma arquitetura bem definida permite ao gerente de projeto adequadamente alocar tarefas de, por exemplo, implementação de componentes e melhor estimar o tempo e tamanho de equipe necessária para realização de um projeto.
- Servir de base para análise da consistência e dependência – o arquiteto de software pode verificar se a arquitetura de software adotada suporta os atributos de qualidade desejados de modo consistente e avaliar o nível de dependência dos atributos de qualidade em relação à arquitetura. Para tanto, ele faz a análise arquitetural que verifica o suporte oferecido pela arquitetura a um conjunto de atributos de qualidade (como desempenho, portabilidade e confiabilidade).

- Ser utilizada para determinar atributos de qualidade do sistema – o arquiteto de software faz a análise arquitetural a fim de determinar os atributos de qualidade. Trata-se de um processo iterativo.
- Atuar como uma estrutura para atender os requisitos do sistema – a arquitetura ajuda a definir os requisitos funcionais, que compreendem o conjunto de funcionalidades do sistema de software, e requisitos não funcionais (ou atributos de qualidade) que determinam as características visíveis ao usuário como desempenho e confiabilidade.

Uma questão que você deve estar se fazendo é: Por que apenas recentemente houve o foco na arquitetura de software?

A resposta é simples: economia e reuso.

Anteriormente não havia forte ênfase na disciplina de engenharia de software, fato este ocorreu com o amadurecimento desta nova área ao longo de toda a década de 1990. Tudo motivou o surgimento de um novo profissional: o arquiteto de software.

Habilidades do Arquiteto de software

Perceba que o arquiteto de software tem um papel de suma importância para estratégia adotada pela empresa. Ele precisa ter profundo conhecimento do domínio, das tecnologias existentes e de processos de desenvolvimento de software. Uma síntese de um conjunto desejado de habilidades para um arquiteto de software e das tarefas atribuídas a ele são apresentados na **Tabela 3**. Note que a prototipagem é uma tarefa comum onde o arquiteto desenvolve um protótipo para 'testar' uma possível solução. Já a simulação pode ser empregada quando ele necessita avaliar o suporte oferecido a determinado atributo de qualidade como o desempenho. Por outro lado, a experimentação pode ocorrer quando o arquiteto precisa testar um novo componente recém implementado.

<i>Habilidades desejadas</i>	<i>Tarefas atribuídas</i>
-------------------------------------	----------------------------------

Conhecimento do domínio e tecnologias relevantes	Modelagem
Conhecimento de questões técnicas para desenvolvimento de sistemas	Análise de compromisso e de viabilidade
Conhecimento de técnicas de levantamento de requisitos, e de métodos de modelagem e desenvolvimento de sistemas	Prototipagem, simulação e experimentação
Conhecimento das estratégias de negócios da empresa	Análise de tendências tecnológicas
Conhecimento de processos, estratégias e produtos de empresas concorrentes	'Evangelizador' de novos arquitetos

Tabela 3. Habilidades e tarefas de um arquiteto de software.

Entendo o Estilo Arquitetural

O estilo arquitetural serve para caracterizar a arquitetura de software de um sistema, possibilitando a:

- Identificação de componentes – o arquiteto identifica quais os principais elementos que tem funcionalidades bem definidas como, um componente de cadastro de (informações de) usuários e um componente de autenticação de usuário numa aplicação Web.
- Identificação de mecanismos de interação – a comunicação entre objetos por meio da troca de mensagens constitui uma forma através da qual os componentes de software interagem entre si.
- Identificação de propriedades – o arquiteto pode analisar as propriedades

oferecidas por cada estilo baseado na organização dos componentes e nos mecanismos de interação, conforme discutido abaixo.

O estilo arquitetural considera o sistema por completo, permitindo o engenheiro ou arquiteto de software determinar como o sistema está organizado, caracterizando os componentes e suas interações. Em outras palavras, ele determina uma estrutura para todos os componentes do sistema. O estilo arquitetural compreende o vocabulário de componentes e conectores, além da topologia empregada. Mas, você pode estar se questionando: Por que saber o estilo arquitetural é importante?

Os sistemas de grande porte exigem níveis de abstração mais elevados (justamente onde se têm os estilos) que servem de apoio à compreensão do projeto e comunicação entre os participantes do projeto. Ele é determinante no entendimento da organização de um sistema de software.

Mas, o que se ganha em saber o estilo arquitetural? Ele oferece:

- Suporte a atributos de qualidade (ou requisitos não funcionais);
- Diferenciação entre arquiteturas;
- Menos esforço para entender um projeto;
- Reuso de arquitetura e conhecimento em novos projetos.

Conhecer o estilo arquitetural permite ao engenheiro antecipar, por meio da análise (arquitetural), o impacto que o estilo (isto é a classe de organização do sistema) terá sobre atributos de qualidade. Adicionalmente, facilita a comunicação do projeto, além do reuso da arquitetura (da solução).

A caracterização e existência de estilos arquiteturais constituem sinais de amadurecimento da engenharia de software, uma vez que permite ao engenheiro organizar e expressar o conhecimento de um projeto de modo sistemático e útil. Note que uma forma de codificar conhecimento é dispor de um vocabulário de um conjunto de conceitos (terminologia, propriedades, restrições), estruturas (componentes e

conectores) e padrões de uso existentes. Conectores são empregados na interação entre componentes como, tubo (pipe) no estilo tubos e filtros, e mensagens no estilo de objetos.

Exemplificando o estilo pipes (tubos) e filtros

O estilo arquitetural de tubos e filtros considera a existência de uma rede através da qual dados fluem de uma extremidade à outra. O fluxo de dados se dá através tubos e os dados sofrem transformações quando são processados nos filtros.

O tubo provê uma forma unidirecional do fluxo de dados uma vez que ele atua como uma espécie de 'condutor' para o fluxo de dados entre a fonte até um destino. O exemplo mais comumente conhecido do estilo tubos e filtros é aquele usado no sistema operacional Unix, isto é:

```
who | sort
```

A linha de comando acima executa o comando `who` (uma única vez) e encaminha sua saída ao programa `sort`, conforme ilustrado na **Figura 1**. O resultado da execução do programa `who` é uma lista de todos os usuários que se encontram conectados (a um servidor específico) naquele momento, enquanto que o programa `sort` ordena essa lista de usuários em ordem alfabética (de login).

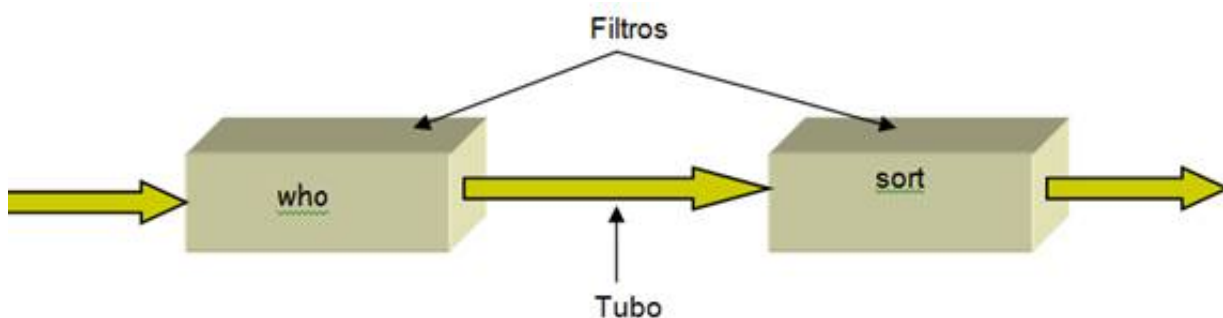


Figura 1. Exemplo do estilo arquitetural de tubos e filtros

Outro exemplo compreende a arquitetura básica de um compilador como ilustrado na

Figura 2. Observe que cada etapa do processo de compilação é realizada separadamente por um componente (ou seja, um filtro).

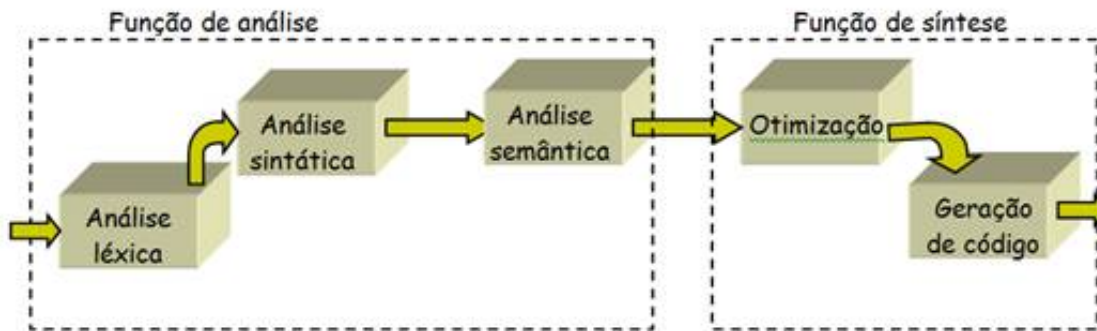


Figura 2. Arquitetura básica de um compilador (segundo estilo arquitetural de tubos e filtros)

Um compilador tem duas funções básicas: análise e síntese. A função de análise é implementada por três componentes: analisadores léxico, sintático e semântico. Já a função de síntese compreende os componentes de otimização e geração de código. Observe que essa arquitetura oferece suporte à portabilidade e reuso.

Entretanto, essa arquitetura evoluiu com a introdução de um componente gerador intermediário de código, conforme ilustrado na **Figura 3, a fim de tornar a arquitetura do compilador 'mais portátil' a várias plataformas com o objetivo de reduzir custos no desenvolvimento de diferentes produtos, ou seja, compiladores para diferentes plataformas.**

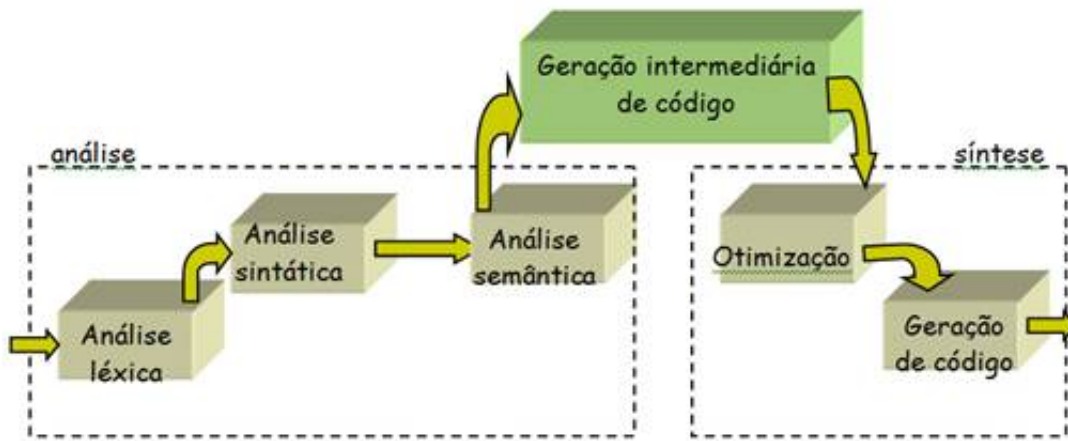


Figura 3. Evolução inicial da arquitetura de um compilador.

Uma nova evolução da arquitetura de compiladores a fim de atender a necessidade de integração (do compilador) com outras ferramentas, tais como editor e depurador, resultou na arquitetura mostrada na **Figura 4**.

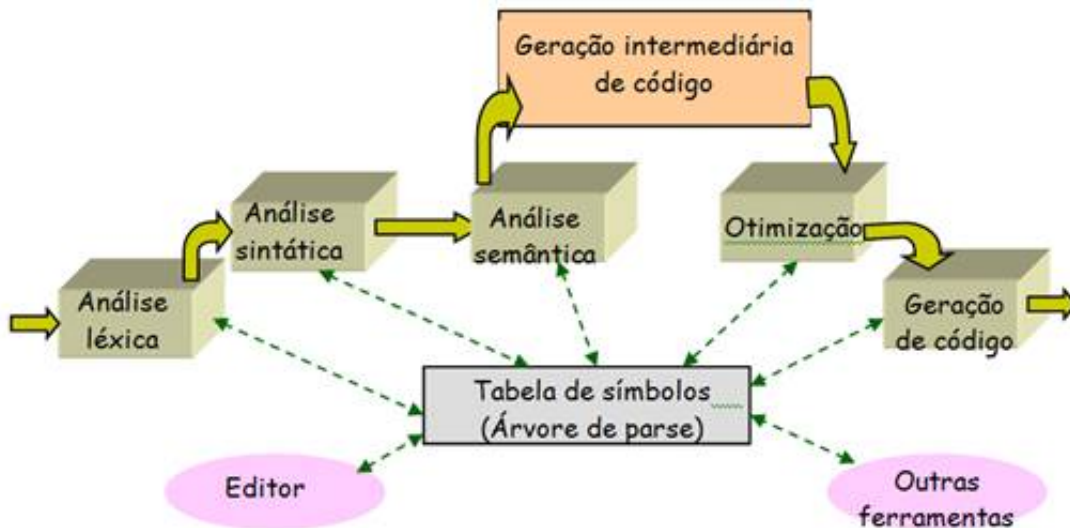


Figura 4. Nova evolução da arquitetura de um compilador.

É importante salientar que a evolução da arquitetura de compilador foi resultado, principalmente, da necessidade de dar suporte ao requisito de portabilidade. Nesse sentido, pode-se destacar como vantagens:

- Problema ou sistema pode ser decomposto de forma hierárquica;
- Função do sistema é vista como composição de filtros;
- Facilidade de reuso, manutenção e extensão, que emprega abordagem caixa preta, onde cada componente tem funcionalidade e interface bem definida, facilitando alterações nos mesmos;
- Desempenho pode ser incrementado através do processamento paralelo de filtros, já que a ativação e uso do componente ocorre com o fluxo de dados, permitindo que componentes com funcionalidades independentes sejam executados de forma concorrente.

Apesar das vantagens acima apontadas, o estilo de tubos e filtros coloca ênfase no modo 'batch', tornando difícil seu uso em aplicações interativas e em situações que exija ordenação de filtros. Outra questão técnica a ser observada é a possibilidade de haver deadlock com o uso de buffers finitos (para armazenamento temporário de dados). Esse estilo arquitetural tem sido empregado em razão das vantagens anteriormente destacadas.

Exemplos de outros estilos arquiteturais compreendem:

- *Camadas* – a arquitetura do sistema de software é organizada num conjunto de camadas, oferecendo maior flexibilidade e suporte a portabilidade. A identificação do nível de abstração nem sempre é evidente e perde-se desempenho à medida que o número de camadas cresce. Exemplo desse estilo compreende os sistemas Web de múltiplas camadas que separa cliente, servidores de aplicação, servidores Web e outros clientes Web.
- *Objetos* – essa arquitetura combina dados com funções numa única entidade (objeto), facilitando a decomposição do problema, manutenção e reuso. É comum utilizar a arquitetura orientada a objetos em sistemas de informação como sistemas de consulta e empréstimos online de bibliotecas de instituições de ensino que dispõem de componentes de cadastro de usuários e componentes de autenticação de usuários. Note que componentes similares existem em outros sistemas de informações, tais como sites de conteúdos (jornais e revistas) que exigem cadastro e autenticação de

qualquer usuário antes de disponibilizar o conteúdo.

- *Invocação implícita* – diferentemente do estilo baseado em objetos, no qual um componente invoca outro diretamente por meio da passagem de mensagens, a invocação implícita requer que componentes interessados em receber ou divulgar eventos se registrem para receber ou enviar. Um exemplo de sistema empregando mensagens são listas de notícias e fórum que possuem componentes de registro de novos usuários acoplados ao componente de autenticação. Perceba que esse tipo de sistema apenas permite que o usuário tenha acesso ao conteúdo se este for devidamente autenticado e registrado.
- (Sistemas orientados a) *Eventos* – trata-se de um estilo no qual os componentes podem ser objetos ou processos e a interface define os eventos de entrada e saída permitidos. Conectores são implementados através do 'binding' evento-procedimento. Assim, eventos são registrados junto com eventos e os componentes interagem entre si através do envio de eventos. Dessa forma, quando um evento é recebido, o(s) procedimento(s) associado(s) a este evento é(são) invocado(s). Um interessante exemplo deste estilo são jogos online, como discutido na seção seguinte.
- Quadro negro (ou Blackboard) – esse estilo faz uso de um repositório central de dados circundado por um conjunto de componentes (ou células) de informações. Esses componentes contêm informações necessárias à solução de problemas. Os dados da solução de problemas são mantidos na base de dados compartilhada (o repositório), o qual é denominado de quadro negro. O exemplo mais comum desse estilo é um sistema especialista.
- Combinação de estilos – Outros sistemas, na prática, combinam estilos arquiteturais resultando numa heterogeneidade, como ilustrado na **Figura 5**, que agrega o estilo de objetos e camadas.

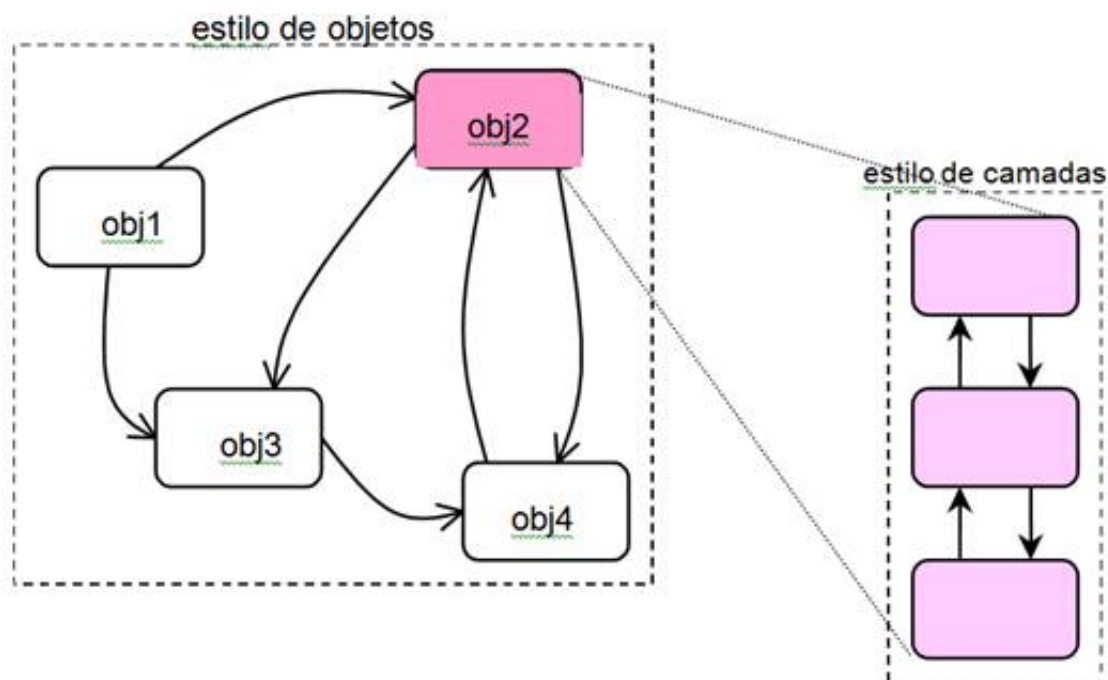


Figura 5. Combinação de estilos.

Exemplificando o estilo combinado de objetos e eventos

Jogos online e de computador têm se tornado comuns atualmente com a popularidade da Internet. Costuma-se categorizar os jogos em:

- Baseados na vez – são jogos nos quais cada ação é baseada na vez do jogador como jogo da velha e xadrez.
- Baseados em eventos – são jogos onde eventos podem ocorrer em qualquer instante e estes ditam o ritmo do jogo. Exemplos compreendem simuladores de vôo e corridas de carro.

Por exemplo, quando os jogos são disponibilizados na Internet, costuma-se denominá-los de jogos online ou jogos para Internet, possibilitando ao usuário jogar contra a máquina (computador). Um exemplo desse tipo de jogo de computador é o Connect4 que tem como meta para cada jogador conectar quatro peças da mesma cor, verticalmente, horizontalmente ou diagonalmente. Cada jogador deve depositar uma peça na parte superior da coluna selecionada e esta cai até preencher a lacuna inferior da coluna (selecionada), conforme ilustrado na **Figura 6**. Note que o quadro contém 7 colunas e 6 linhas, um indicador de status do jogo e um seletor manual (utilizado para selecionar a coluna na qual uma peça será depositada).

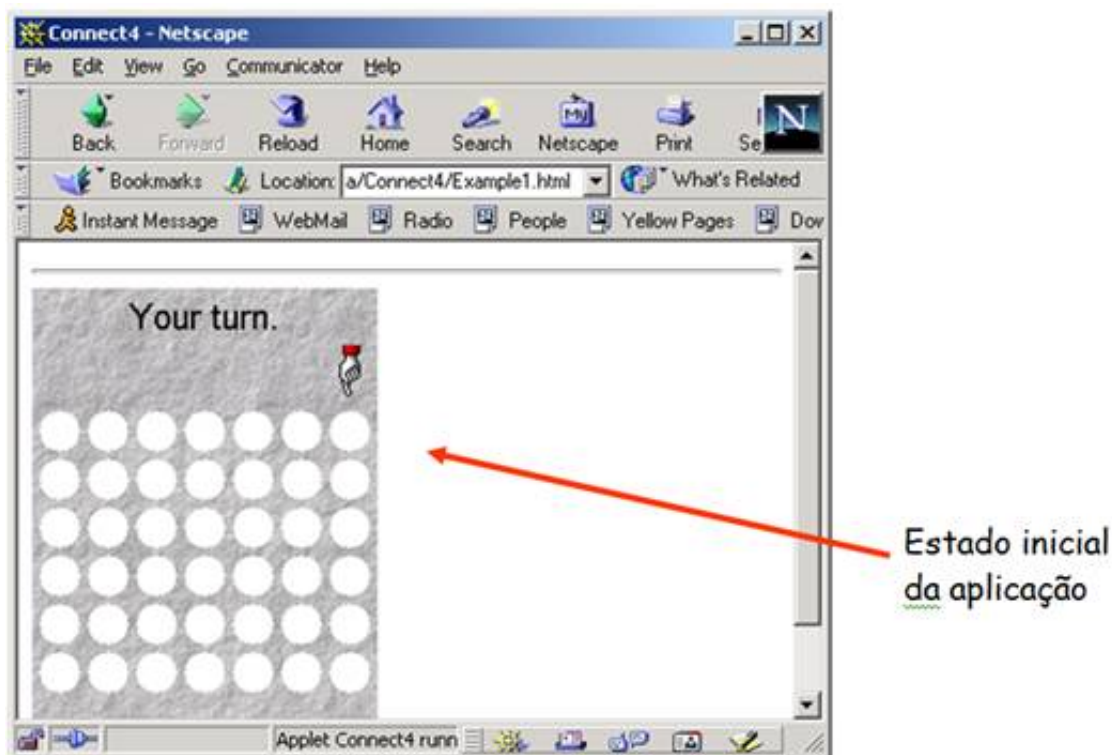


Figura 6. Jogo Connect4.

A arquitetura de software dessa aplicação é apresentada na **Figura 7**. O componente jogador computacional (que dispõe de recursos de inteligência artificial para simular um jogador humano) contém uma classe Connect4State que trata a maioria das requisições para verificar se algum jogador venceu o jogo, e também dispõe de mecanismo para atualizar o estado do jogo após uma jogada do jogador computacional.

O componente (máquina) de inferência contém uma classe para tratar as jogadas feitas pelos jogadores bem como determinar a melhor jogada para o jogador computacional.

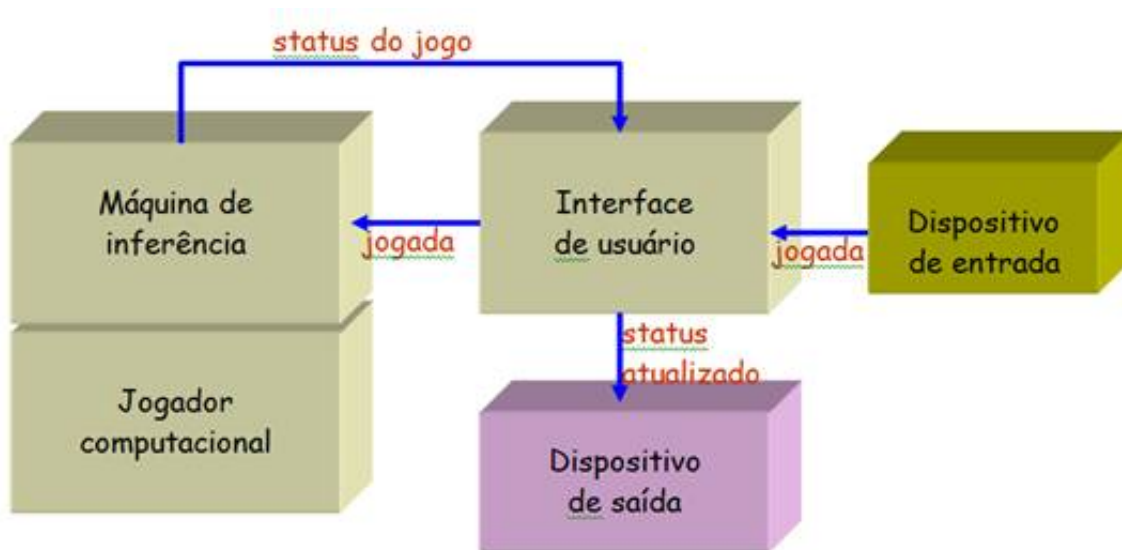


Figura 7. Arquitetura do Connect4

O componente de interface de usuário contém uma classe que carrega os arquivos de imagem e áudio, trata as requisições feitas através do mouse e invoca um método que checa se houve vitória, derrota ou empate, além de fazer a atualização da interface gráfica. Um possível estado final desse jogo é mostrado na **Figura 8**, onde na terceira linha há um conjunto de quatro peças na cor azul, indicando que o computador completou primeiro a conexão de quatro peças na mesma cor (a cor do usuário humano é vermelha).

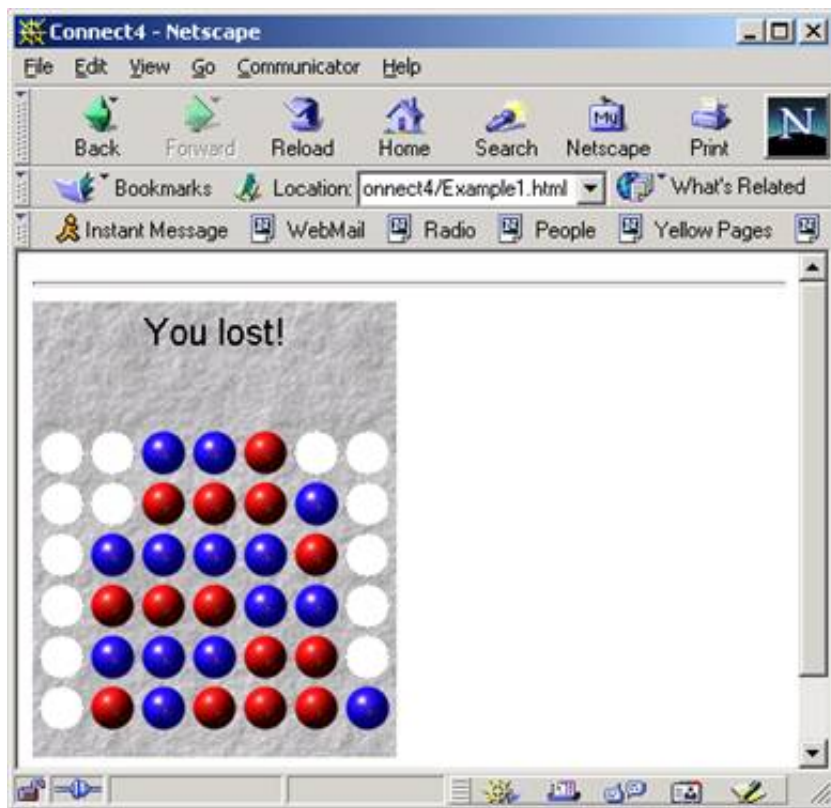


Figura 8. Possível estado final do jogo Connect4

A combinação estilo arquitetural orientado para eventos e objetos permite a decomposição de um sistema em termos de objetos (componentes de inferência, componente de interface gráfica e componente jogador computacional) que são mais independentes além de possibilitar que atividades de computação e coordenação (de eventos) sejam realizadas separadamente. Há ainda a facilidade de reuso e manutenção, já que novos objetos podem ser facilmente adicionados. Essa característica, e interfaces bem definidas, facilitam ainda a integração.

O mecanismo de invocação é não determinístico (isto é, ocorre de forma aleatória) uma vez que considera a recepção de eventos. Adicionalmente, os componentes têm seus dados preservados de qualquer modificação acidental já que essas informações são encapsuladas em objetos, facilitando também a integração.

Importância do Reuso

É importante perceber de tudo o que foi apresentado e discutido que um modelo arquitetural oferece suporte ao reuso de várias formas. Por exemplo, pode se ter arquiteturas reusáveis que forneçam a organização e o modelo de coordenação,

permitindo seu uso em diversos sistemas. Além disso, podem-se reutilizar componentes, já testados, em mais de um sistema. Disso tudo, o mais importante é o reuso do conhecimento que tem impacto direto na definição da arquitetura de software candidata à solução de um problema (ou sistema).

A ampla variedade de plataformas e utilitários, juntamente com a pressão do mercado para reduzir o tempo de entrega de produtos de software e elevar a produtividade, faz com que o reuso seja apontado como uma das chaves de sucesso para empresas.

O reuso de artefatos (ou componentes) é possível quando o projeto arquitetural está incorporado e orienta o processo de desenvolvimento de software. Isto, como discutido, permite antever os atributos de qualidade que o sistema suporta e também administrar o cronograma de execução do projeto. Portanto, impactando positivamente o reuso e economia do sistema.

O quadro apresentado na **Tabela 4** sumariza as principais características da arquitetura de software e pontos importantes na capacitação e reciclagem de engenheiros e arquitetos de software.

<i>Características de arquitetura de software</i>	<i>Uso prático da arquitetura de software</i>
Constitui um artefato reutilizável	Como um arquiteto de software pode organizar o projeto e código de um sistema
Dispõe de mecanismos de interconexão	Como um arquiteto avalia e implanta arquiteturas de software em sistemas
Oferece um vocabulário de projeto e separa funcionalidades	Como um arquiteto de software atua no processo de desenvolvimento de software

Vincula o projeto a atributos de qualidade	Como um arquiteto avalia a qualidade do código baseada em métricas de produto
Suporta o desenvolvimento baseado em componentes e linha de produto (quando os requisitos são considerados para uma família de sistemas)	Como usar arquitetura como parâmetro para reduzir custos de manutenção e amortizar custos de desenvolvimento

Tabela 4. Características e uso prático da arquitetura de software

Conclusão

Embora arquitetura de software seja um tema relevante no contexto atual para desenvolvimento de sistemas de software que têm seu foco no reuso e, portanto, consideram tanto o aspecto econômico quanto a produtividade, sua incorporação nos processos de desenvolvimento de software tem sido observada apenas em empresas de grande porte e em poucas de médio porte. Entretanto, esse cenário começa a mudar dada a crescente necessidade de integração de sistemas a qual tem a arquitetura de software como premissa. Assim, o fator econômico tem sido e será o determinante da sobrevivência da empresas de software. Novas estratégias de desenvolvimento de sistemas devem ser *para* o reuso e *com* reuso (de componentes de software) e o pilar principal dessas estratégias é a arquitetura de software. Há, portanto, uma necessidade premente de formação de capital humano com tais qualificações.

Links

História da Indústria de Software

www.softwarehistory.org

An introduction to software architecture

www.cs.cmu.edu/afs/cs/project/able/www/paper_abstracts/intro_softarch.html

SEI's Software Architecture Technology Initiative

www.sei.cmu.edu/architecture/sat_init.html

Worldwide Institute of Software Architects

www.wwisa.org/wwisamain/index.htm

The Software Architecture Portal


www.softwarearchitectureportal.org/



Antonio Mendes Da Silva Filho

Professor e consultor em área de tecnologia da informação e comunicação com mais de 20 anos de experiência profissional, é autor dos livros Arquitetura de Software e Programando com XML, ambos pela Editora Campus/Elsevier, tem mai [...]

O que você achou deste post?

 Gostei (16)  (2)

Comentário | Tire sua dúvida



Juliano Tiago Rinaldi

Gostei muito do artigo, me fez lembrar os tempos de faculdade, o quanto é importante Engenharia de Software, mas não damos a atenção dedicada a arquitetura e processo bem definido, e com isto somos cobrados a frente nos projetos com custo elevadíssimo e descontrole de manutenções.

Vejo ainda um problema grande quanto a venda de produtos, o cliente deseja claro o custo menor possível, e a concorrência desleal e mal profissionalizada, se prostitui para ganhar migalhas do mercado, e com isto criam softwares com ciclo de vida muito pequenos e mal elaborados.

Aguardo ansiosamente a mudança desse mercado.

[há +1 ano] - Responder



Serviços

Inclua um comentário

Adicionar aos Favoritos

Marcar como lido/assistido

Incluir anotação pessoal

Versão para impressão

+Engenharia de
software

Mais posts

Artigo

TestLink: Gerenciando atividades de teste

Artigo

GodVis: Monitoramento da Dívida Técnica com God Classes

Artigo

Aplicação de jogos no ciclo de desenvolvimento de sistemas

Artigo

Gerenciamento e controle de mudanças de requisitos

Artigo

Pontos de função: Realizando uma contagem estimada e uma contagem detalhada

Artigo

PMBok vs Agilidade: são mesmo opostos? Miguel Santos

Artigo

Metodologia RUP: Como trabalhar com requisitos em Scrum

Revista

Revista Engenharia de Software Magazine 74

Revista

Revista Engenharia de Software Magazine 73

Listar mais conteúdo



Anuncie | Loja | Publique | Assine | Fale conosco



DevMedia

Curtir

54.607 pessoas curtiram [DevMedia](#).



Plug-in social do Facebook

Hospedagem web por [Porta 80 Web Hosting](#)