Sobre a Importância da Arquitetura de Software no Desenvolvimento de Sistemas de Software

Antonio Mendes da Silva Filho[1]

[1]União dos Institutos Brasileiros de Tecnologia
52050-002 – Recife – PE – Brasil
Antonio.mendes@unibratec.com.br

Abstract. *A key attribute to achieve maturity in the engineering field is to use in a usual way the existing solutions for developing new systems. In a reuse strategy, the main focus is on the architecture-centered reuse aiming the software systems development. Within this context, software architecture supports the system organization, allowing its understanding in terms of components, relationships, and properties consistent throughout the time and implementations. This work presents an introduction to this topic.*

Resumo. *Um dos principais atributos para se alcançar a maturidade em uma disciplina de engenharia é o uso rotineiro de soluções existentes no desenvolvimento de novos sistemas. Em uma estratégia de reuso de software, enfatiza-se a importância do reuso centrado na arquitetura, objetivando o desenvolvimento de sistemas de software. Dentro deste contexto, a arquitetura de software serve como uma organização do sistema, permitindo seu entendimento em termos de componentes, inter-relacionamentos e propriedades consistentes ao longo do tempo e implementações. Este trabalho apresenta uma introdução a este tópico.*

1. Introdução

Há, aproximadamente, quatro décadas atrás, software constituía uma pequena senão ínfima parcela dos sistemas computacionais quando comparado ao hardware. Naquela época, os custos de desenvolvimento e manutenção de software eram desprezíveis. Hoje, porém, software é responsável por significativa porção dos sistemas computacionais. Encontramos software nas mais diversas aplicações. No uso doméstico, fazemos uso de processadores de texto e planilhas eletrônicas. Entretanto, software tem sido elemento chave e muito utilizado em sistemas de grande porte. Podemos exemplificar seu uso no controle e supervisão dos sistemas de geração/distribuição de energia bem como em sistemas de telecomunicações, onde ele é encarregado do roteamento e controle de milhares de ligações telefônicas. Hoje em dia, empresas e pessoas têm conseguido otimizar suas atividades, geralmente, fazendo uso de sistemas computacionais. O computador tem se tornado um *companheiro* e sido uma ferramenta fundamental em nosso cotidiano.

Todavia, ao mesmo tempo em que os sistemas computacionais têm tornado-se quase ubíquos em inúmeras aplicações, tamanho e complexidade do software têm aumentado tanto que as técnicas de abstração utilizadas até o final do anos 80 (como,

por exemplo, tipos de dados abstratos, linguagens de programação de alto nível e técnicas de decomposição modular) já não são mais suficientes para lidar com esses novos problemas envolvendo o projeto de software a nível de sistema. Isto refere-se ao nível de organização de um sistema ou, mais propriamente, o nível de arquitetura de software.

Perceba que abstração é uma forma de lidar com a complexidade de problemas e sistemas. Como seres humanos, costumamos reconhecer que estamos, regularmente, reconstruindo um padrão já utilizado em um contexto diferente. Exemplo disso seria a especificação ou parte de um código. Poderíamos dizer que a essência da abstração é reconhecer um padrão, atribuindo-lhe nome e definindo seu uso, e analisá-lo buscando formas para especificá-lo.

Diferentemente do uso de técnicas que empregam algoritmos, estruturas de dados e as linguagens de programação que as implementam, o crescimento em escala dos sistemas de software demandou notações para conectar módulos, técnicas para gerenciar configurações e controlar versões. Entretanto, à medida que os sistemas tornavam-se maiores e mais complexos, outros fatores como, por exemplo desempenho e qualidade, passaram a ser considerados também. Tais fatores estão intimamente relacionados à organização global do sistema de software o qual será objeto de estudo deste artigo.

2. Evolução Tecnológica

A busca incessante do conhecimento é uma característica peculiar do ser humano. Ao longo de séculos, o homem tem buscado respostas a muitas questões que lhe vêm à mente bem como encontrado soluções a uma enorme de gama de problemas que lhe são apresentados. Resultado disto, têm sido as grandes invenções tecnológicas que testemunhamos no século XX e tantos outros que remontam às origens humana. Já o século XXI é vislumbrado como século do conhecimento onde a informação é praticamente ubíqua, as fronteiras de tempo e espaço já não parecem mais existir, e passamos a viver num mundo caracterizado cada vez mais por uma interdependência global.

Dentre as tecnologias que surgiram no século passado, destacamos os primeiros esforços para construir o primeiro computador (ENIAC[1]) de propósito geral completamente eletrônico que ocorreu na Universidade da Pensilvânia (EUA), entre 1942 e 1945. Participaram do projeto e desenvolvimento do ENIAC J. Presper Eckert e John W. Mauchly. Em 1944, John von Neumann integrou a equipe que havia desenvolvido inicialmente o ENIAC e então o grupo deu início ao desenvolvimento do EDVAC[2]. No período de 1946 a 1952, von Neumann e seus colaboradores desenvolveram o computador do Instituto de Estudos Avançados de Princeton (EUA). Durante a década de 50 até quase metade da década de 1960, havia um produção limitada de software e naquela época o tamanho dos programas eram geralmente pequeno.

[1] Electronic Numeric Integrator and Computer.
[2] Electronic Discrete Variable Computer.

Entretanto, pouco a pouco, software passa a constituir uma tecnologia chave, sendo usada nas mais diversas áreas, desde aplicações financeiras e comerciais a aplicações complexas, tais como controle de usinas de energia e missões espaciais. Aos poucos, a indústria de computadores passa a ser um dos ramos de negócios mais competitivos do planeta e o componente software torna-se a força prevalecente em termos de inovação tecnológica.

Apesar da importância que esta nova tecnologia passa a ter na época, inexiste ferramentas oferecendo suporte ao desenvolvimento de software e pouco se conhece sobre seu processo de desenvolvimento. Concomitante a estes fatos, as aplicações começam a tornar-se maiores e mais complexas, demandando uma parcela cada vez maior de software. Ademais, outro fator a destacar é o crescente papel da manutenção. Todos esses fatores contribuíram para o surgimento da Engenharia de Software, termo que foi cunhado numa conferência seminal do NATO em Garmisch, Alemanha, em 1968 [Naur 69].

A idéia central de engenharia de software é a de fazer uso de princípios de engenharia a fim de produzir a baixo custo software que operem corretamente e com eficiência em equipamentos, onde o software seja instalado. Assim, à medida que a tecnologia de software conquistava mais espaço, diversos paradigmas de engenharia de software e programação surgiam. Face à necessidade de lidar com sistemas de software cada vez maiores e mais complexos, a técnica de abstração foi utilizada objetivando reconhecer padrões, analisar e especificar sistemas. De fato, o desenvolvimento de técnicas de abstração tem sido uma das principais fontes da melhoria nas práticas de programação. Contudo, o crescimento continuado do tamanho de sistemas de software aliada à complexidade tem requerido em conjunto mais confiabilidade, economia e desempenho. Estes requisitos estão associados a organização global dos sistemas de software e será abordado ao longo desse texto.

3. A Natureza dos Sistemas de Software

O termo software não se restringe apenas aos programas de computadores associados com uma aplicação, mas também envolve toda a documentação necessária para instalação, uso, desenvolvimento e manutenção dos programas. Uma questão que vem à mente é: quando um sistema pode ser considerado como bem *arquitetado* (i.e., concebido e desenvolvido utilizando-se princípios de engenharia) ? Alguém poderia dizer que um sistema é bem bolado se ele faz o que o(s) usuário(s) quer(em). Todavia, uma avaliação mais geral da qualidade e quão bem projetado é um sistema requer a identificação de atributos os quais desejaríamos encontrar num software bem arquitetado. Há dois atributos importantes que merecem nossa atenção: confiabilidade e manutenibilidade.

Um nível adequado de confiabilidade é essencial a um sistema de software independente de seu uso. A confiabilidade é um importante atributo da qualidade de software, implicando que a aplicação associada a ele realizará suas funções como esperado. Geralmente, a confiabilidade de software é definida em termos de

comportamento estatístico, ou seja, da probabilidade de que o software irá operar como desejado ao longo de um período especificado de tempo.

A exemplo de outros sistemas, software é sujeito à mudanças regulares e, portanto, torna-se importante que o software seja codificado e documentado de modo a não incorrer em custos proibitivos. Neste contexto, a manutenebilidade possui dois aspectos distintos: a necessidade de reparo e evolução dos sistemas. Um sistema de software precisará de reparo se existe defeitos que pedem correção. Por outro lado, os sistemas podem evoluir a fim de satisfazer novos requisitos de usuários. Perceba que embora um sistema de software possa deteriorar-se, requerendo novas versões a fim de atender a natureza evolutiva dos sistemas e a demanda por novas características, ele não sofre de desgaste como ocorre com outros dispositivos e sistemas. Hardware, por exemplo, desgasta-se com o decorrer do tempo pela ação de uso indevido, acúmulo de poeira e funcionamento em temperaturas elevadas.

Adicionalmente, é importante observar que (componente de) software existe em duas formas básicas: código executável ou não executável. Contudo, apesar de podermos, hoje em dia, trabalhar a nível de componentes, ainda não podemos encomendar um componente de software da forma que fazemos com o hardware. A idéia é que um componente deveria ser projetado e implementado de modo que pudesse ser reutilizado em muitas aplicações.

A reusabilidade é um atributo importante em componentes de software com elevada qualidade. Como exemplo, poderíamos citar o utilização de componentes reutilizáveis de interfaces gráficas que permitem a criação de janelas, menus e outras formas de interação com usuário.

Apesar dos avanços alcançados até o presente momento em termos de reusabilidade, tem havido um crescimento continuado tanto em termos de tamanho quanto da complexidade dos sistemas de software. Aliados a esses aumentos, há uma demanda cada vez mais acentuada por sistemas com maior confiabilidade e desempenho bem como menor custo. Tais ocorrências redirecionam a ênfase do processo de desenvolvimento de software para o nível de organização ou arquitetural do sistema.

4. O que é Arquitetura de Software?

Para lidar com a complexidade e o tamanho de sistemas, engenheiros de software têm feito uso de princípios de projeto como, por exemplo, a ocultação de informações. Entretanto, à medida que os sistemas tornam-se cada vez maiores, o uso de uma disciplina deve ser enfatizado de modo a obter resultados de baixo custo e maior qualidade. Dentro deste contexto, arquitetura de software tem entrado em cena de modo a lidar com sistemas grandes e complexos.

Perceba que o crescimento em termos de tamanho e complexidade dos sistemas de software faz com que o problema de desenvolvimento extrapole o projeto das estruturas de dados e algoritmos para o sistema. Em outras palavras, projetar a estrutura global de um sistema emerge como um problema novo, ou seja, desenvolvimento de

software baseado na arquitetura. O desenvolvimento de software no nível arquitetural compreende questões estruturais, dentre as quais, destacam-se:

- seleção de alternativas de projeto;
- escalabilidade e desempenho;
- organização e estrutura geral de controle;
- protocolos de comunicação, sincronização;
- atribuição de funcionalidade a componentes de projeto.

Ademais, torna-se cada vez mais evidente que processos de engenharia de software requerem projeto arquitetural de software. Primeiro, é importante ser capaz de reconhecer estruturas comuns utilizadas em sistemas já desenvolvidos possibilitando aos projetistas de software compreender as relações existentes entre sistemas e desenvolver novos sistemas baseado em variações de sistemas antigos. Segundo, o entendimento de arquiteturas de software existentes permite aos engenheiros tomarem decisão sobre alternativas de projeto. Terceiro, uma descrição arquitetural do sistema é essencial a fim de analisar e descrever propriedades de um sistema complexo. Quarto, o conhecimento de notações formais para descrição de paradigmas arquiteturais permite que engenheiros possam apresentar novos projetos de sistemas para outras pessoas e instituições.

Fica claro que a estrutura global ou, mais propriamente, a arquitetura de software constitui-se num fator de suma importância no projeto de sistemas de software de grande porte. Baseado em discussões realizadas no Software Engineering Institute da Carnegie Mellon University, David Garlan e Dewayne Perry [Garlan 95] definiram arquitetura de software como:

> *"A estrutura dos componentes de um programa/sistema, seus inter-relacionamentos, e princípios e diretrizes guiando o projeto e evolução ao longo do tempo."*

Arquitetura de software é uma área relativamente nova dentro da engenharia de software. Este tópico não havia despertado o interesse de pesquisadores e profissionais até final da década de 80 quando Mary Shaw [Shaw 89] aponta a necessidade de considerar o nível organizacional ou arquitetural dos sistemas. Em seguida, em meados da década de 90, foi publicado o livro seminal sobre arquitetura de software de Mary Shaw e David Garlan [Shaw 96] que tratou de muitas questões desta nova área. Note ainda que arquitetura de software tem foco no estudo da organização global dos sistemas de software bem como do relacionamento entre subsistemas e componentes. Desde suas origens, quando descrições qualitativas de organizações de sistemas eram consideradas como úteis, arquitetura de software tem amadurecido ao longo da última década, buscando englobar e explorar notações, ferramentas e técnicas de análise. Também, tem-se observado sobreposição e interação da pesquisa de arquitetura de software com trabalhos sobre reuso baseado em componentes, projeto de software, classes de componentes, análise de programas e linhas de produto. Embora não se considere muito produtivo tentar fazer distinções entre essas áreas, sempre que oportuno as diferenças serão salientadas.

5. A Necessidade e o Papel da Arquitetura de Software

Baseado no que foi apresentado até aqui, tem-se que a idéia básica de arquitetura de software é aquela de que um sistema de software em elevado nível de abstração pode ser descrito através de subsistemas ou componentes distintos, sendo estes interrelacionados. Note quão fundamental a arquitetura de um sistema de software é e quão essencial também é o projeto arquitetural dentro do processo de desenvolvimento de um sistema. Perceba ainda que uma *arquitetura de referência* ou *base* constitui um elemento fundamental para o desenvolvimento de linhas de produção de software, onde múltiplos sistemas com funcionalidades distintas podem ser concebidos.

A arquitetura de software tenta analisar as propriedades do software a nível de subsistema ou módulo. Examinando o papel da arquitetura em um contexto mais amplo do gerenciamento de processo de software e da tentativa de ordenar as diversas técnicas novas que têm surgido, alguns possíveis benefícios podem ser apontados. Dessa forma, tem-se que a arquitetura pode:

- atuar como uma estrutura a fim de atender as requisitos de sistema;
- ser utilizada como aspecto técnico para o projeto de sistema bem como suporte na estimação de custos e gerência de processo;
- servir de base para análise de consistência e dependência;
- prover suporte ao reuso.

Vamos tentar entender mais o seu papel. Considere a Figura 1 que ilustra um afunilamento nas descrições arquiteturais de um sistema.

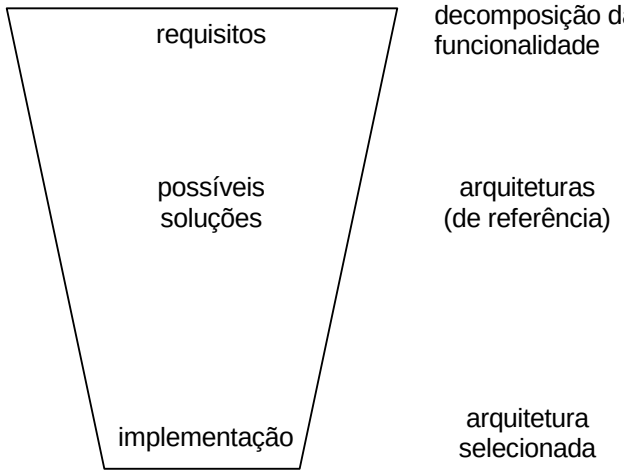

Figura 1 – Níveis de descrição de um sistema.

O processo de desenvolvimento de um sistema de software vai desde a concepção do sistema, quando requisitos são elicitados e analisados até sua concreta implementação. Conforme mostrado na Figura 1, na fase inicial do processo há um interesse em compreender a funcionalidade que o sistema deverá prover sem que haja

necessidade de se tomar qualquer decisão a nível estrutural ou arquitetural. Uma forma de entender a funcionalidade e serviços a serem oferecidos pelo sistema é compreendendo a funcionalidade oferecida por sistemas similares. É fácil perceber que nesta fase inicial existe uma maior gama de soluções.

Por outro lado, durante a implementação, decisões já foram tomadas e uma única arquitetura é implementada. Entretanto, antes da implementação de uma arquitetura, a funcionalidade do sistema é particionada a fim de identificar possíveis subsistemas ou módulos. Isto permitirá analisar os módulos compondo o sistema bem como os requisitos, visando encontrar ou apresentar uma arquitetura que satisfaça às necessidades do sistema. Neste caso, as arquiteturas podem ser vislumbradas como referências que retratam as funcionalidades levantadas durante a decomposição funcional, acarretando numa alocação funcional a uma descrição arquitetural. Em outras palavras, a arquitetura de um sistema envolve a divisão de funções entre subsistemas ou módulos bem como os mecanismos de interação entre os módulos e a representação da informação compartilhada.

As alternativas de arquiteturas disponíveis a um projetista poderiam ser descritas e classificadas, compondo assim as arquiteturas de referência. Considerando as arquiteturas de referência, poderíamos formular regras de projeto que indicariam boas e más opções de projeto. Estas regras podem ser norteadas por requisitos funcionais e não funcionais. Desse modo, este texto pode ser visto como uma forma de codificar conhecimento de projeto de software, possibilitando seu uso na educação de novos engenheiros de software bem como em atividades cotidianas de profissionais da área.

Considerando o que vimos até agora, percebe-se que um arquiteto de software possui uma posição estratégica. Ele precisa ter de conhecimento profundo do domínio onde o sistema a ser desenvolvido será utilizado, das tecnologias relevantes e de processos de desenvolvimento.

Além disso, o arquiteto de software deverá ter um foco nas implicações que os objetivos organizacionais terão sobre as opções técnicas, isto é, uma visão global do sistema. Ele deverá construir modelos para o problema em mãos, buscando achar uma solução e explorando abordagens alternativas. Documentação é imprescindível a fim de apresentar e discutir com os demais membros da equipe de desenvolvimento e com o gerente responsável pelo projeto.

Resumindo, poderíamos dizer que um arquiteto de software deveria reunir um conjunto de habilidades a fim de realizar satisfatoriamente as funções que lhe são atribuídas. A Tabela 1 apresenta habilidades desejadas de um arquiteto de software juntamente com possíveis tarefas que este deveria realizar.

Embora não tenha sido explicitamente acrescentado à tabela acima, um arquiteto deve ter facilidade em trabalhar em vários níveis de abstração (fazendo uso de técnicas de abstração) bem como ser hábil em buscar múltiplas alternativas de projeto. Ser criativo e fazer investigações devem ser habilidades inerentes ao arquiteto.

Habilidades desejadas	Tarefas atribuídas
compreensão profunda do domínio e tecnologias pertinentes	modelagem
entendimento de aspectos técnicos para desenvolver sistema com sucesso	análise de compromissos/viabilidade
técnicas de elicitação, técnicas de modelagem e métodos de desenvolvimento	prototipação, simulação, realização de experimentos
entendimento das estratégias de negócios da instituição onde atua	análise de tendências de tecnologias
conhecimento de produtos, processos e estratégias de concorrentes	atuar como mentor de arquitetos novatos

Tabela 1 – Habilidades e tarefas de um arquiteto de software.

As seções subsequentes apresenta um sumário de um conjunto de tópicos pertinentes a arquitetura de software, tais como estilos arquiteturais e análise arquitetural. Essa apresentação tem como objetivo despertar o interesse do leitor, motivando-o a conhecer mais sobre esse novo campo de estudo. A apresentação detalhada pode ser encontrada em [Silva Filho 2002].

7. Sumário de Temas de Arquitetura de Software

Na seção anterior, verificamos que a arquitetura de software de um sistema possui papel importante na obtenção de metas de qualidade e manutenibilidade. A arquitetura de software de um sistema é composta de subsistemas ou componentes (responsáveis pelo comportamento do sistema), de conectores (interligando os componentes) e de padrões de conexão (possibilitando vários tipos de interação e compartilhamento de informações entre esses componentes). Objetivando apresentar temas importantes desse novo campo de estudo ao leitor, é apresentado um sumário de 7 dos principais tópicos de arquitetura de software os quais são detalhados em [Silva Filho 2002].

A seção anterior deste texto fez uma introdução de um novo tópico de estudo, dentro da engenharia de software, o qual é visto como um aspecto crítico no projeto de qualquer sistema de software de grande porte - a arquitetura de software - caracterizando o nível arquitetural do projeto de software. Arquitetura de software é a estrutura global do sistema capturada através da organização deste descrita em elevado nível de abstração, em termos dos elementos computacionais pertinentes e das interações entre esses elementos. Numa discussão sobre a evolução tecnológica, vê-se que ao longo das três últimas décadas, a estrutura de software tem tornado-se numa das questões de maior interesse. Apesar disto, apenas recentemente a arquitetura de software tem surgido com novo campo de estudo para pesquisadores e profissionais de engenharia de software. Evidência desta tendência é visível face à grande quantidade de trabalhos em diversas áreas, tais como linguagem de interface de módulos, arquiteturas específicas de domínio, linguagens para descrição de arquiteturas, e padrões de projeto, temas estes explorados nos capítulos subsequentes.

Um dos principais aspectos do projeto arquitetural é o uso de padrões de organização de um sistema. Muitos desses padrões, também chamados de estilos arquiteturais, têm sido desenvolvidos ao longo do tempo à medida que projetistas de sistemas reconhecem o valor de princípios de organização bem como estruturas para certas categorias de software. Assim, no capítulo 2 de [Silva Filho 2002] é feito a introdução de um novo tópico de estudo apresentado um conjunto de estilos arquiteturais. São apontados compromissos e características associadas à várias opções de estilo arquitetural.

Outro aspecto importante envolve um processo genérico de elicitação de requisitos arquiteturais. Elementos de entrada desse processo compreende os principais influenciadores do sistema, envolvendo projetista, arquiteto e usuários. A experiência do arquiteto de software é grande importância. Como saída deste processo tem-se um conjunto de requisitos funcionais identificados com o suporte de casos de uso, uma lista de requisitos arquiteturais específicos e um conjunto de cenários de qualidade. Um conjunto de atributos de projeto e atributos de qualidade (ou requisitos não funcionais) são apresentados. Esses requisitos são utilizados para desenvolver cenários de qualidade que caracteriza o uso do sistema de software em situações específicas e permite que haja a validação de uma arquitetura de software proposta. O leitor é referenciado ao capítulo 3 de [Silva Filho 2002].

Além disso, torna-se importante discutir um conjunto de propriedades de arquitetura de software consideradas intrínsecas à estrutura do sistema. Essa apresentação é feita no capítulo 4 de [Silva Filho 2002]. Estas propriedades juntamente com os requisitos arquiteturais, envolvendo requisitos não funcionais e atributos de projeto, apresentados no capítulo 3 servem de base para a análise arquitetural. Neste capítulo, dois métodos de análise arquitetural (SAAM e ATAM) são apresentados, ilustrando-se os passos a proceder a análise. Adicionalmente, discute-se o papel do requisitos não funcionais ou atributos de qualidade como mecanismo para indicar tendências na qualidade do sistema. Finalmente, considera-se como estes métodos de análise poderiam ser aplicados sob a óptica de minimizar custos e agregar benefícios.

Um outro tópico abordado engloba a estrutura global ou modelo arquitetural de sistemas de software a qual pode ser estudada através do uso de dimensões de projeto. Isto é discutido do capítulo 5 de [Silva Filho 2002]. Uma dimensão de projeto identifica as opções funcionais e arquiteturais feitas durante o projeto de um sistema e classifica as alternativas existentes para cada escolha. Adicionalmente, regras em formato de narrativa são formuladas, objetivando correlacionar as opções existentes a uma dimensão de projeto. O conjunto dessas regras constituem uma ferramenta de projeto e oferece uma solução promissora para uma orientação automática de projeto arquitetural. Exemplos de regras de projeto para sistema interativos são apresentadas. O projeto arquitetural é discutido, observando-se o uso de requisitos arquiteturais e dimensões de projeto bem como sua natureza altamente iterativa.

Tem havido considerável interesse em se desenvolver arquiteturas de software para domínios específicos ou DSSAs (Domain Specific Software Architectures). Uma DSSA oferece suporte a uma abordagem de desenvolvimento de domínio específico que

possibilita a configuração de componentes reutilizáveis. Verifica-se que uma DSSA é composta de: (i) um modelo de domínio, (ii) requisitos de referência, (iii) uma arquitetura de referência, (iv) a infra-estrutura ou ambiente de suporte, e (v) um processo de desenvolvimento. Essas DSSAs fornecem uma organização estrutural adequada para uma classe de aplicações ou domínio como, por exemplo, sistemas de informações (envolvendo processamento de dados e computação heterogênea distribuída), sistemas de tempo real, sistemas inteligentes e sistemas de comunicações. Essas classes de aplicações são apresentadas e discutidas no capítulo 6 de [Silva Filho 2002]. Adicionalmente, um conjunto de benefícios decorrentes do uso de DSSAs são apresentados.

Os desenvolvedores de sistemas interativos necessitam tomar decisões de projeto, objetivando otimizar o processo de desenvolvimento desses sistemas bem como melhorar a qualidade do produto final. A tomada de decisão tem de ser feita entre metas desejáveis, embora algumas vezes conflitantes, tais como a necessidade de minimizar os futuros efeitos de alterar a tecnologia e melhorar o desempenho do tempo de execução de um sistema. O capítulo 7 apresenta aos desenvolvedores de sistemas interativos um conjunto de arquiteturas de software candidatas que podem ser utilizadas em tais sistemas. Também, discute-se os compromissos oriundos quando da escolha de uma arquitetura candidata em detrimento de outra.

Finalizando, pode-se observar que a idéia principal deste texto é servir o leitor e motivá-lo a conhecer uma nova área dentro da Ciência da Computação. Adicionalmente, busca-se apresentar uma breve retrospectiva do campo de arquitetura de software discutindo os problemas iniciais de organização de sistemas de software bem como discutindo os caminhos explorados até os dias atuais.

Referências

Garlan, D. and Perry, D., “Introduction to the Special Issue on Software Architecture”, IEEE Transactions on Software Engineering, April 1995, pp. 269-274.

Naur, P., Randell, B. and Buxton, J. (Eds.), “Software Engineering: A Report on a Conference Sponsored by NATO Science Committee, NATO, 1969.

Parnas, D. L., “On the Criteria To Be Used in Decomposing Systems into Modules”, Communications of the ACM, Vol. 15, No. 12, Dec. 1972, pp. 1053-1058.

Shaw, M. “Larger Scale Systems Require Higher-Level Abstractions”, Proc. of the International Workshop on Software Specification and Design, Pittsburgh, PA, May 1989, pp. 143-146.

Shaw, M. and Garlan, D. “Software Architecture – Perspectives on na Emerging Discipline, Prentice Hall, Upper Saddle River, New Jersey, 1996.

Silva Filho, A. M., “Arquitetura de Software”, Editora Campus, 2002.