Modelagem de Sistemas

Guilherme Henrique Pasqualin Algeri

guilherme.algeri@sistemafiep.org.br

FIEP SESI SENAI IEL

SENAI

Arquitetura de Software

Arquitetura de Software

O projeto da arquitetura de software é uma etapa essencial no desenvolvimento de sistemas de software de grande porte e complexos

Arquitetura de Software

Dentro deste contexto, a arquitetura de software é fundamental para o desenvolvimento de linhas de produção de software

Arquitetura de Software

onde se tem um conjunto de funcionalidades concebidas e implementadas a partir da mesma arquitetura (de software) base

Arquitetura de Software

Entretanto, antecedendo a etapa de projeto da arquitetura de software, há a necessidade de fazer o levantamento dos requisitos do sistema

Arquitetura de Software

De um modo geral, o conjunto de requisitos de um sistema é definido durante as fases iniciais do processo de desenvolvimento

Arquitetura de Software

Tal conjunto de requisitos é visto como uma especificação do que deveria ser implementado

Arquitetura de Software

Os requisitos são descrições de como o sistema deveria se comportar, e contêm informações do domínio da aplicação e restrições sobre a operação do sistema

Arquitetura de Software

Durante a fase de elicitação de requisitos, um projetista ou arquiteto de software faz uso de sua experiência a fim levantar os requisitos, buscando identificar características do sistema a ser desenvolvido

Arquitetura de Software

Além disso, informações do domínio juntamente com informações de estilos arquiteturais existentes podem ser utilizados como fontes de dados que auxiliam na identificação dos requisitos

Arquitetura de Software

Outro recurso que pode ser usado pelo projetista é construir cenários. Os cenários de uso oferecem suporte a requisitos específicos e visam tanto a elicitação quanto a análise de requisitos

Arquitetura de Software

Uma vez que o conjunto de requisitos tenha sido obtido, o projetista/arquiteto de software estará em condições de iniciar o projeto da arquitetura de software, conforme ilustrado

Arquitetura de Software

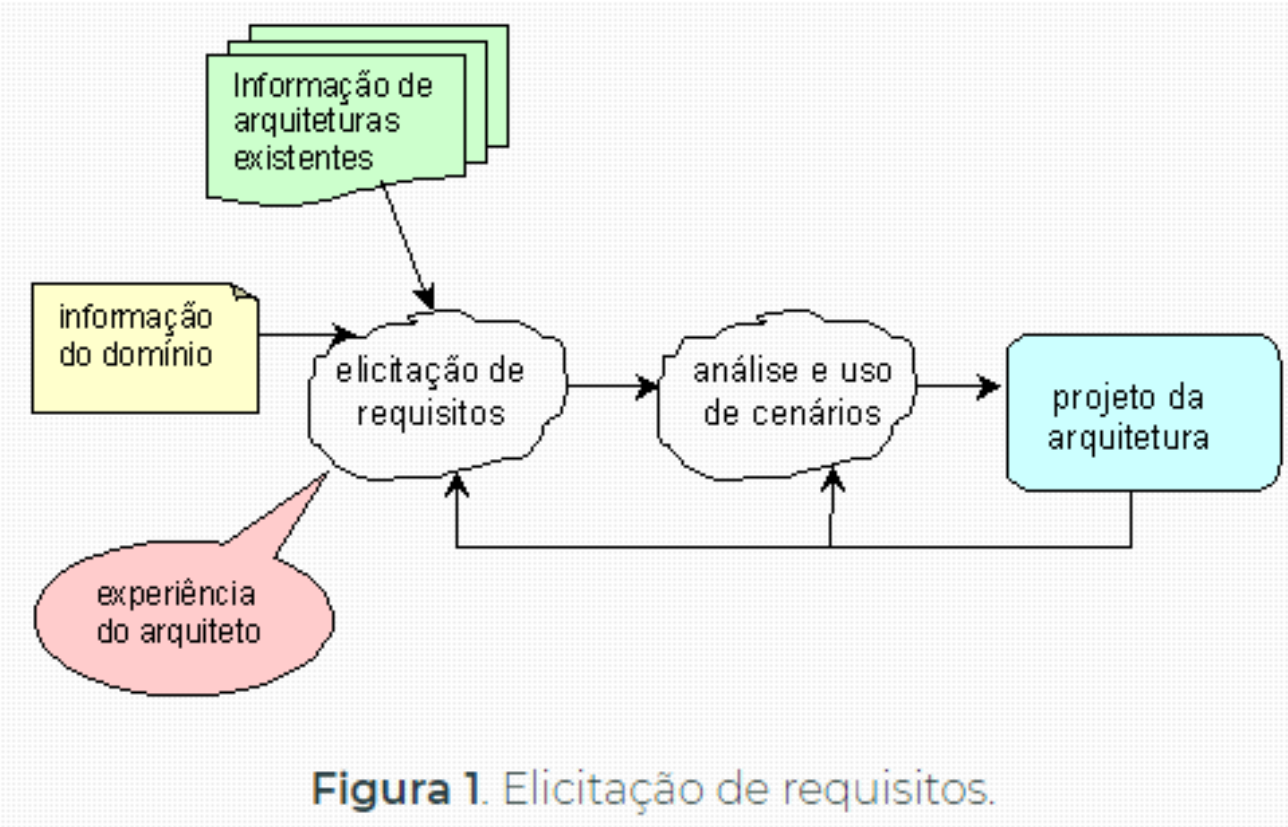

Figura 1. Elicitação de requisitos.

Arquitetura de Software

É importante observar que a etapa de projeto arquitetural pode precisar fazer uso de cenários de uso ou mesmo uma re-análise a fim de refinar a arquitetura a ser empregada no sistema a ser desenvolvido

Arquitetura de Software

O processo de desenvolvimento baseado na arquitetura considera a arquitetura de software como fator orientador do processo

Arquitetura de Software

Isto acarreta em **colocarmos os requisitos não funcionais** associados à arquitetura como principais aspectos do processo de desenvolvimento

Arquitetura de Software

Note que o desenvolvimento de um sistema de software centrado na arquitetura inicia-se com um arquiteto de software, de posse de um conjunto de requisitos do sistema

Arquitetura de Software

Nesse momento, busca-se identificar qual estilo ou combinação destes oferece suporte mais adequado a esses requisitos e, portanto, derivar uma arquitetura de software que atenda às características do sistema a ser desenvolvido

Arquitetura de Software

Vale ressaltar que a complexidade de um sistema de software é determinada tanto por seus requisitos funcionais – **o que ele faz** – quanto requisitos não funcionais – **como ele faz**

Arquitetura de Software

Tal distinção é baseada nas seguintes definições

FIEP
SESI
SENAI
IEL

SENAI

Requisitos Funcionais

Requisitos Funcionais

A organização das funcionalidades de software de um sistema é explicitada através da **arquitetura de software**

Requisitos Funcionais

Há uma grande quantidade de estilos arquiteturais que trazem características peculiares a cada estilo, e estes, por sua vez, podem ser combinados gerando novos estilos

Requisitos Funcionais

Um requisito de sistema de software que especifica uma função que o sistema ou componente deve ser capaz de realizar

Requisitos Funcionais

Estes são requisitos de software que definem o comportamento do sistema, ou seja, o processo ou transformação que componentes de software ou hardware efetuam sobre as entradas para gerar as saídas

Requisitos Funcionais

Esses requisitos capturam as funcionalidade sob o ponto de vista do usuário

Requisitos Não Funcionais

Requisitos Não Funcionais

Em engenharia de sistemas de software, um requisito não funcional de software é aquele que descreve não o que o sistema fará, mas como ele fará

Requisitos Não Funcionais

Assim, por exemplo, têm-se requisitos de desempenho, requisitos da interface externa do sistema, restrições de projeto e atributos da qualidade

Requisitos Não Funcionais

A **avaliação dos requisitos não funcionais** é feita, em parte, por meio de testes, enquanto que outra parte é avaliada de maneira subjetiva

Requisitos Não Funcionais

Perceba que tanto os requisitos funcionais quanto os **não funcionais** possuem importância no desenvolvimento de um sistema de software

Requisitos Não Funcionais

Entretanto, os **requisitos não funcionais**, também denominados de atributos de qualidade, têm um papel relevante durante o desenvolvimento de um sistema

Requisitos Não Funcionais

atuando como critérios na seleção e/ou composição de uma arquitetura de software, dentre as várias alternativas de projeto

Requisitos Não Funcionais

Cabe salientar que à medida que os sistemas tornam-se maiores e mais complexos, o **suporte a requisitos não funcionais** depende cada vez mais de decisões tomadas no projeto da arquitetura de software

Requisitos Não Funcionais

Trata-se de uma visão compartilhada pelos profissionais da área e, especificamente, pela comunidade de arquitetura de software

Requisitos Não Funcionais

Requisitos não funcionais são aqueles que não estão diretamente relacionados à funcionalidade de um sistema

Requisitos Não Funcionais

Desconsiderar ou não considerar adequadamente tais requisitos é dispendioso, pois torna difícil a correção uma vez que o sistema tenha sido implementado

Requisitos Não Funcionais

Suponha, por exemplo, que uma decisão tenha sido feita de modularizar a arquitetura de um sistema de modo a facilitar sua manutenção e adição de novas funcionalidades

Requisitos Não Funcionais

Entretanto, modularizar um sistema adicionando uma camada a mais pode comprometer um outro requisito, o de desempenho

Requisitos Não Funcionais

Portanto, faz-se necessário definir logo cedo quais requisitos não funcionais serão priorizados na definição de uma arquitetura

Requisitos Não Funcionais

Os requisitos não funcionais abordam aspectos de qualidade importantes em sistemas de software

Requisitos Não Funcionais

Se tais requisitos não são levados em consideração, então o sistema de software poderá ser inconsistente e de baixa qualidade, conforme discutido acima

Requisitos Não Funcionais

Para tanto, quanto mais cedo forem definidos os critérios arquiteturais, mais cedo o projetista pode identificar o estilo, ou combinação de estilos, mais apropriado ao sistema considerado

Requisitos Não Funcionais

Ao desenvolver um novo sistema de software bem como sua arquitetura, os projetistas ou engenheiros de software apresentam um conjunto de atributos de qualidade ou requisitos não funcionais que o sistema deveria suportar

Requisitos Não Funcionais

Exemplo destes requisitos são desempenho, portabilidade, manutenibilidade e escalabilidade

Requisitos Não Funcionais

A arquitetura de software deveria oferecer suporte a tais requisitos. **Isto resulta da associação existente entre arquitetura de software e requisitos não funcionais**

Requisitos Não Funcionais

Importante observar que cada estilo arquitetural (isto é, a forma na qual o código do sistema é organizado) suporta requisitos não funcionais específicos

Requisitos Não Funcionais

A estruturação de um sistema é determinante no suporte oferecido a um requisito não funcional

Requisitos Não Funcionais

Por exemplo, o uso de camadas permite melhor separar as funcionalidades de um sistema, tornando-o mais modular e facilitando sua manutenção

Requisitos Não Funcionais

Considere, por exemplo, o padrão IEEE-Std 830-1993 [IEEE 1993] que lista um conjunto de 13 requisitos não funcionais a serem considerados no documento de especificação de requisitos de software

Requisitos Não Funcionais

Este padrão inclui, dentre outros, requisitos de desempenho, confiabilidade, portabilidade e segurança

Requisitos Não Funcionais

Embora haja um conjunto de propostas, consideradas como complementares, concentraremos nossas atenções num conjunto de requisitos diretamente associados a um sistema de software

Requisitos Não Funcionais

Este conjunto é baseado numa classificação apresentada por Sommerville, onde é feita a distinção entre requisitos externos, de produto, e de processo [Sommerville 2007].

Requisitos Não Funcionais

A **Figura 2** é uma adaptação da proposta de Sommerville, onde consideramos os requisitos de produto, associados à arquitetura de software

Requisitos Não Funcionais

É importante observar que a Figura 2 mostra um **subconjunto de requisitos não funcionais**, denominados de requisitos de produtos os quais estão associados à arquitetura de um sistema de software

Requisitos Não Funcionais

Note que a classificação apresentada em [Sommerville 2007] ainda considera os requisitos de processo e requisitos externos como sendo requisitos não funcionais, além dos requisitos de produtos

Requisitos Não Funcionais

A figura exibe um conjunto de 7 requisitos não funcionais, sendo alguns destes ainda decompostos

Requisitos Não Funcionais

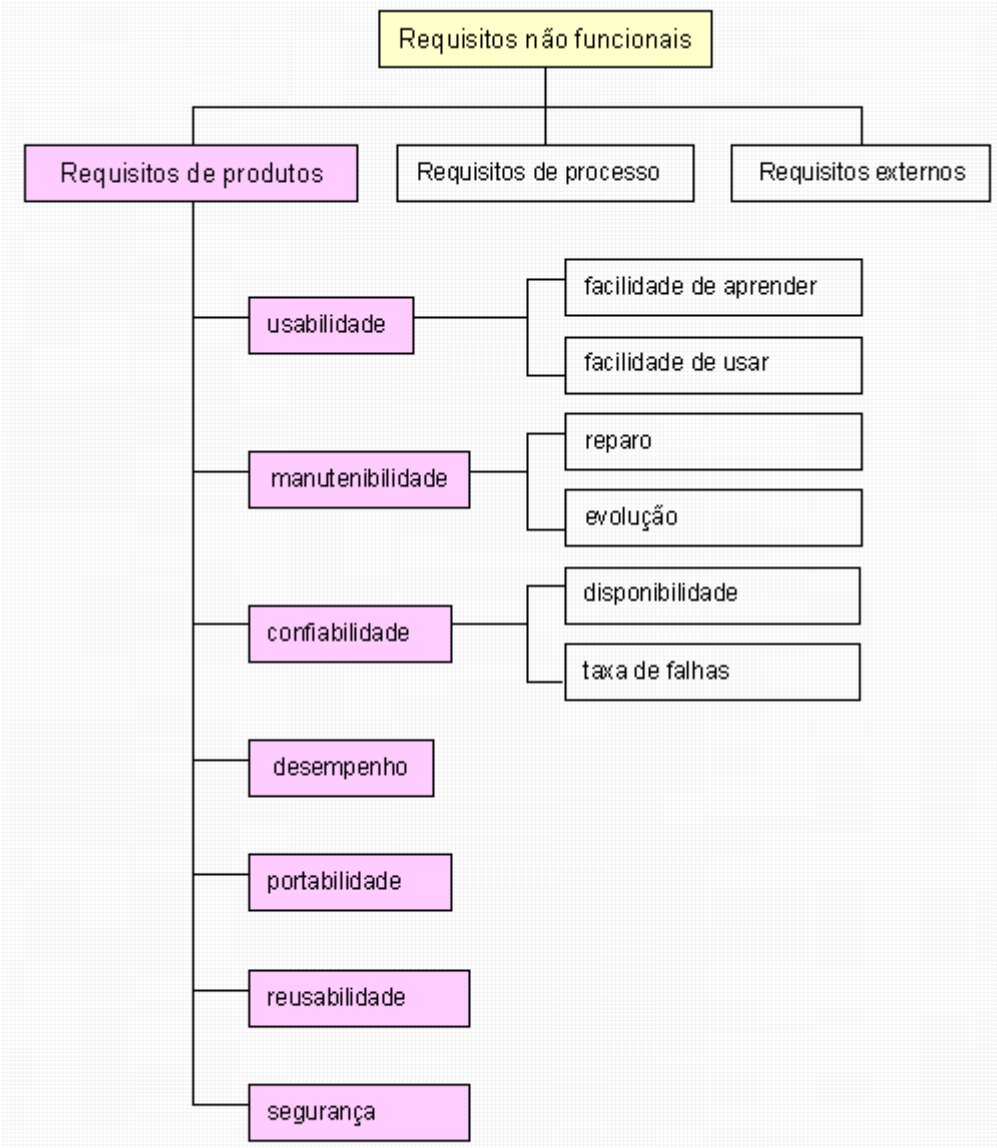

Figura 2. Tipos de requisitos não funcionais.

Usabilidade

Usabilidade

Usabilidade é um dos atributos de qualidade ou requisitos não funcionais de qualquer sistema interativo, ou seja, no qual ocorre interação entre o sistema e seres humanos

Usabilidade

A noção de usabilidade vem do fato que qualquer sistema projetado para ser utilizado pelas pessoas deveria ser fácil de aprender e fácil de usar, tornando assim fácil e agradável a realização de qualquer tarefa

Usabilidade

Requisitos de usabilidade especificam tanto o nível de desempenho quanto a satisfação do usuário no uso do sistema. Dessa forma, a usabilidade pode ser expressa em termos de:

Usabilidade

Facilidade de aprender: Associado ao tempo e esforço mínimo exigido para alcançar um determinado nível de desempenho no uso do sistema

Usabilidade

Facilidade de uso: Relacionado à velocidade de execução de tarefas e à redução de erros no uso do sistema

Usabilidade

Os requisitos de usabilidade são coletados juntamente com outros requisitos (de dados e funcionais) usando algumas das técnicas de elicitação de requisitos como entrevistas ou observação

Usabilidade

A coleta desses dados pode ocorrer, por exemplo, verificando o log de ações do usuário quando do uso de funcionalidade do sistema

Usabilidade

Esses requisitos de usabilidade podem ser expressos através de métricas de usabilidade, expressas em termos de medidas de desempenho

Usabilidade

Tyldesley apresentou um conjunto de critérios que podem ser utilizados durante a medição de usabilidade [Tyldesley 1988]

Usabilidade

A seleção de critérios a serem usados para mensurar a usabilidade depende do tipo de sistema. Exemplos de critérios de medição de usabilidade são apresentados na **Figura 3**

Usabilidade

1. Tempo para realizar um tarefa.
2. Percentual de tarefa concluído.
3. Percentual de tarefa concluído por unidade de tempo.
4. Taxa de sucessos/falhas.
5. Tempo consumido com erros.
6. Percentual de erros.
7. Número de comandos utilizados.
8. Número de comandos disponíveis não utilizados.
9. Freqüência de uso de *ajuda* (help) ou documentação.
10. Número de vezes que o usuário expressa satisfação ou frustação.

Figura 3. Critérios de medição de usabilidade.

Usabilidade

A definição das metas de usabilidade através de métricas é parte do processo denominado de engenharia de usabilidade

Usabilidade

Neste processo, faz-se necessário também estabelecer os níveis desejados de usabilidade

Usabilidade

Se, por exemplo, o usuário tem dificuldade em encontrar a funcionalidade desejada no sistema e, conseqüentemente, precisa recorrer à ajuda (Help) ou expressa insatisfação,

Usabilidade

então se observa que dois dos critérios da Figura 3 são considerados

Usabilidade

A quantidade de vezes que essas ocorrências são observadas serve de indicador do suporte oferecido à usabilidade pelo sistema

Usabilidade

A usabilidade é um dos atributos de qualidade de um sistema e tem sido cada vez mais levada em consideração durante o desenvolvimento de software

Usabilidade

A usabilidade pode ser afetada pelos componentes funcional (ou de aplicação) e de apresentação de um sistema

Usabilidade

Mesmo que esses componentes sejam bem projetados, ainda assim a usabilidade poderá ficar comprometida se a arquitetura do sistema não levar em consideração a facilidade de efetuar uma modificação

FIEP SESI SENAI IEL

SENAI

Manutenibilidade

Manutenibilidade

O termo manutenção de software é geralmente empregado quando nos referimos às modificações feitas após o sistema de software ter sido disponibilizado para uso

Manutenibilidade

Na realidade, o termo manutenibilidade é um tanto abrangente já que ele envolve tanto a atividade de reparo (de algum defeito existente no sistema de software)

Manutenibilidade

quanto a atividade de alteração/evolução de características existentes ou adição de novas funcionalidades não previstas ou capturadas no projeto inicial

Manutenibilidade

O reparo de um sistema de software ocorre quando defeitos são detectados, fazendo-se necessária a correção deles

Manutenibilidade

A capacidade de efetuar um reparo depende do número de componentes do sistema

Manutenibilidade

Por exemplo, se o sistema é monolítico, ou seja, constituído de um único componente, então tornar-se mais difícil efetuar o reparo se este sistema de software é de grande porte

Manutenibilidade

No entanto, se o sistema de software é modularizado, então tende a ser mais fácil analisar e reparar o existente

Manutenibilidade

Podemos dizer que a modularidade favorece a capacidade de fazer reparo, permitindo que defeitos fiquem confinados a poucos módulos, considerando-se que se tenha a funcionalidade adequadamente separada

Manutenibilidade

Uma observação importante é que a necessidade de fazer reparo é minimizada à medida que a confiabilidade do sistema aumenta

Manutenibilidade

Similarmente a outros sistemas, os sistemas de software evoluem ao longo do tempo, seja com a adição de novas funcionalidades ou com a modificação das existentes

Manutenibilidade

Esta capacidade de evolução de um sistema de software é também influenciada pela modularidade do sistema

Manutenibilidade

Note que se a arquitetura do sistema não considerar sua possibilidade de evolução por meio da adição e/ou alteração de funcionalidades do sistema, modificações tornar-se-ão cada vez mais difíceis à medida que o software evolui

Manutenibilidade

De um modo geral, a manutenibilidade é um dos requisitos mais relacionados com a arquitetura de um sistema de software

Manutenibilidade

A facilidade de fazer alteração no sistema existente, seja adicionando ou modificando alguma funcionalidade, depende muito da arquitetura deste

Manutenibilidade

Se considerarmos a necessidade de incrementar a quantidade de componentes encarregados do processamento de uma ligação telefônica (numa central telefônica)

Manutenibilidade

ou de transações eletrônicas, então esta adição de novos componentes deve ocorrer sem a necessidade de modificar a arquitetura existente e ainda comprometer pouco (ou nada) o desempenho atual do sistema

Manutenibilidade

Tal suporte à manutenibilidade (seja para correção ou evolução do sistema) deve ser facilmente acomodada pela arquitetura de software

Manutenibilidade

É importante lembrar que uma arquitetura de software define os componentes e conexões entre estes e, portanto, também definem sob que circunstâncias componentes ou conectores podem ser alterados

Manutenibilidade

Dessa forma, uma arquitetura ou estilo arquitetural de um sistema de software deveria efetivamente acomodar as modificações que precisarem ser feitas tanto durante seu desenvolvimento quanto após o sistema entrar em operação

Confiabilidade

Confiabilidade

Confiabilidade de software é a probabilidade de o software não causar uma falha num sistema durante um determinado período de tempo sob condições especificadas

Confiabilidade

A probabilidade é uma função da existência de defeitos no software. Assim, os estímulos recebidos por um sistema determinam a existência ou não de algum defeito

Confiabilidade

Em outras palavras, a confiabilidade de software, geralmente definida em termos de comportamento estatístico, é a probabilidade de que o software irá operar como desejado num intervalo de tempo conhecido

Confiabilidade

Também, a confiabilidade caracteriza-se um atributo de qualidade de software o qual implica que um sistema executará suas funções como esperado

Confiabilidade

Os requisitos de confiabilidade compreendem restrições sobre o comportamento do sistema de software em tempo de execução

Confiabilidade

Na realidade, tem-se um conjunto de métricas de confiabilidade de software associadas a esses requisitos

Confiabilidade

Geralmente, as falhas de um componente de software são de natureza transitória, ou seja, elas ocorrem apenas para algumas entradas (estímulos) enquanto o sistema poderá continuar operando normalmente em outras circunstâncias

Confiabilidade

Isto distingue o software do hardware já que as falhas no segundo são de natureza permanente

Confiabilidade

Vale ressaltar que falha é o que se observa pelos usuários (externamente) enquanto que os defeitos, de origem interna ao sistema, são os motivadores das falhas

Confiabilidade

Não é tão simples relacionar a disponibilidade de um sistema de software a uma falha existente, pois isto depende de vários fatores,

Confiabilidade

tais como grau de corrupção de dados em decorrência de algum defeito de software e tempo de re-inicialização, dentre outros

Confiabilidade

Exemplos de métricas utilizadas para avaliar a confiabilidade de software compreendem:

Confiabilidade

Disponibilidade: Esta é uma medida de quão disponível o sistema estaria para uso, isto é, quão disponível o sistema estaria para efetuar um serviço solicitado por algum usuário

Confiabilidade

Por exemplo, um serviço de um sistema de software terá uma disponibilidade de 999/1.000

Confiabilidade

Isto significa que dentre um conjunto de 1.000 solicitações de serviço, 999 deverão ser atendidas. Esta métrica é muito importante em sistemas de telecomunicações, por exemplo

Confiabilidade

Taxa de ocorrência de falha: Esta é uma medida da frequência na qual o sistema falha em prover um serviço como esperado pelo usuário, ou seja, a frequência na qual um comportamento inesperado é provável de ser observado

Confiabilidade

Por exemplo, se temos uma taxa de ocorrência de falha de 2/1.000, isto significa que 2 falhas são prováveis de acontecerem para cada 1.000 unidades de tempo

Confiabilidade

Probabilidade de falha durante fase operacional: Esta é uma medida da probabilidade que o sistema irá comporta-se de maneira inesperada quando em operação

Confiabilidade

Esta métrica é de suma importância em sistemas críticos que requerem uma operação contínua

Confiabilidade

Tempo médio até a ocorrência de falha ou Mean Time To Failure (MTTF): Esta é uma medida do tempo entre falhas observadas

Confiabilidade

Note que esta métrica oferece um indicativo de quanto tempo o sistema permanecerá operacional antes que uma falha aconteça

Confiabilidade

Qualquer métrica que venha a ser utilizada para avaliar a confiabilidade de um sistema dependerá da forma como o sistema é usado

Confiabilidade

Note ainda que o tempo é um fator considerado nas métricas. Ele é escolhido de acordo com a aplicação

Confiabilidade

Há sistemas de software que operam de forma contínua, enquanto outros operam de maneira periódica

Confiabilidade

Por exemplo, considere um caixa eletrônico de banco

Confiabilidade

Este é um exemplo de sistema que opera periodicamente, isto é, um caixa eletrônico encontra-se parte do tempo em operação, enquanto no restante do tempo fica ocioso

Confiabilidade

No exemplo de um caixa eletrônico de banco, uma unidade de tempo mais adequada é o número de transações

Confiabilidade

Assim, um exemplo de falha seria a perda de dados entrados por um usuário. Neste caso, a especificação de confiabilidade poderia ser a ocorrência de uma falha dessa natureza a cada 10.000 transações

FIEP
SESI
SENAI
IEL

SENAI

Desempenho

Desempenho

Desempenho é um atributo de qualidade importante para sistemas de software

Desempenho

Considere, por exemplo, um sistema de uma administradora de cartões de crédito

Desempenho

Em tal sistema, um projetista ou engenheiro de software poderia considerar os requisitos de desempenho para obter uma resposta de tempo para autorização de compras por cartão

Desempenho

Note que os requisitos de desempenho têm impacto mais global sobre o sistema e, por essa razão, estão entre os requisitos não funcionais mais importantes

Desempenho

Contudo, é geralmente difícil lidar com os requisitos de desempenho e com outros requisitos não funcionais uma vez que eles estão em conflito, conforme discutido acima

Desempenho

No início da atividade de projeto da arquitetura de software torna-se necessário definir quais requisitos não funcionais serão priorizados, dada a possibilidade de conflito entre eles

Desempenho

Adicionalmente, desempenho é importante porque afeta a usabilidade de um sistema

Desempenho

Se um sistema de software é lento, ele certamente reduz a produtividade de seus usuários ao ponto de não atender às suas necessidades

Desempenho

Também, se o sistema de software requer muito espaço em disco para armazenamento de informações, pode ser oneroso utilizá-lo

Desempenho

Por exemplo, se um sistema de software exige muita memória para ser executado, ele pode afetar outras aplicações que são executadas no mesmo ambiente

Desempenho

Além disso, ele pode executar tão lentamente que o sistema operacional tenta balancear o uso de memória entre as diversas aplicações

Desempenho

De um modo geral, o requisito de desempenho pode ser decomposto em termos de tempo e espaço, como ilustrado na **Figura 4**

Desempenho

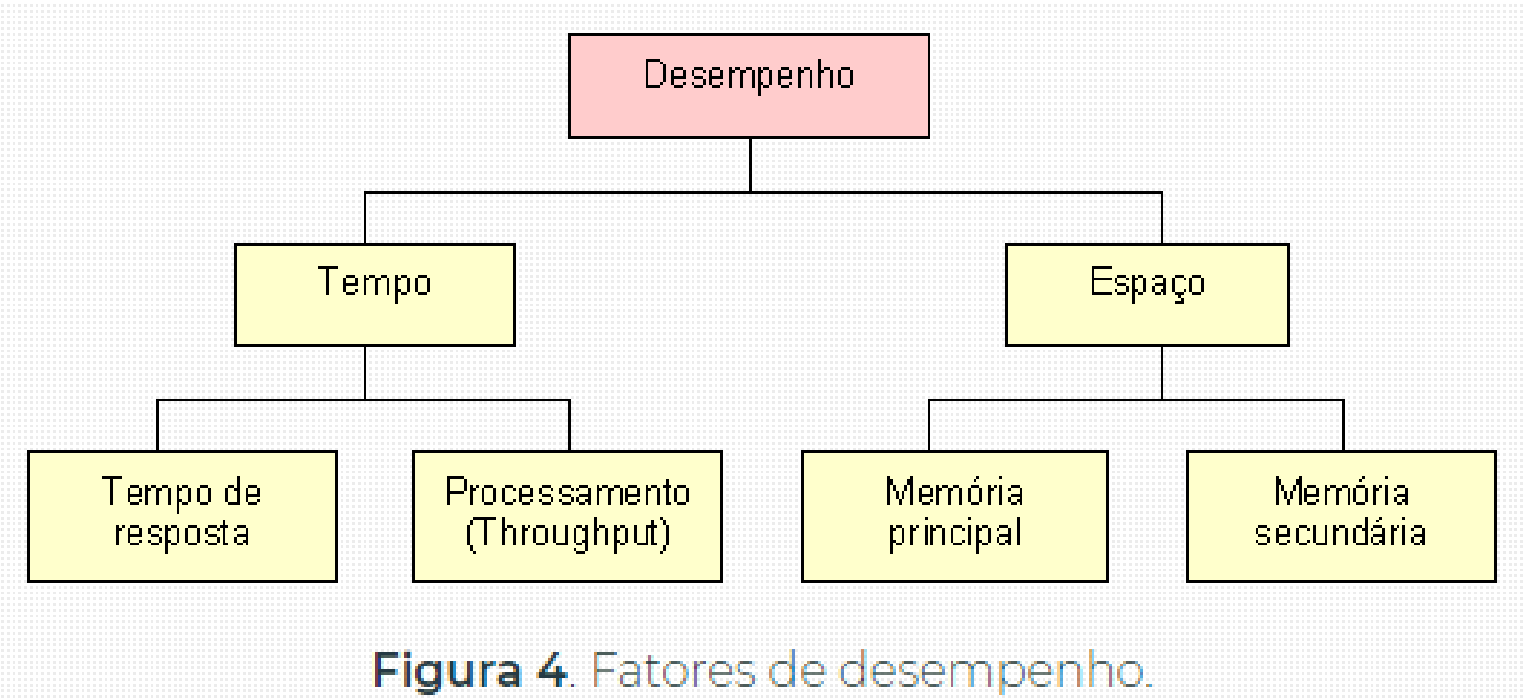

Figura 4. Fatores de desempenho.

Desempenho

O requisito de desempenho restringe a velocidade de operação de um sistema de software. Isto pode ser visto em termos de:

Desempenho

Requisitos de resposta:

Especificam o tempo de resposta de um sistema de software aceitável para usuários

Desempenho

Neste caso, um projetista poderia especificar que o sistema deveria responder à solicitação de um serviço específico de um usuário dentro de um intervalo de 2 segundos

Desempenho

Por exemplo, num caixa eletrônico, após o usuário inserir o cartão magnético do banco no local apropriado (leitor do equipamento),

Desempenho

o sistema deveria exibir uma nova tela, num intervalo de 2 segundos, requerendo que o usuário digite sua senha de conta corrente

Desempenho

Numa outra situação, o usuário pode ser solicitado a digitar sua senha e não o faz dentro de um período de 20 segundos, quando então um timeout ocorre e o sistema retorna à tela inicial

Desempenho

Requisitos de processamento (throughput):

Estes requisitos especificam a quantidade de dados que deveria ser processada num determinado período de tempo

Desempenho

Um exemplo seria exigir que o sistema de software possa processar, no mínimo, 6 transações por segundo

Desempenho

Requisitos de temporização:

Este tipo de requisito especifica quão rapidamente o sistema deveria coletar dados de entrada de sensores antes que outras leituras de dados de entrada, feitas posteriormente, sobrescrevam os dados anteriores

Desempenho

Assim, por exemplo, poderia ser especificado que o sistema deveria efetuar leitura de dados 5 vezes por segundo, como condição mínima

Desempenho

Requisitos de espaço:

Em alguns casos, os requisitos de espaço podem ser considerados. Aqui, podemos nos referir à memória principal ou secundária

Desempenho

Por exemplo, a memória principal para executar uma aplicação poderia ser considerada como um requisito de desempenho uma vez que ela está relacionada ao comportamento do sistema em tempo de execução

Desempenho

É importante observar que o desempenho depende da interação existente entre os componentes de um sistema de software e, portanto, está muito associado à arquitetura

Desempenho

Neste caso, os mecanismos de comunicação utilizados pelos componentes de um sistema têm influência sobre o desempenho obtido

Desempenho

Conforme vimos anteriormente, desempenho está relacionado a outros requisitos não funcionais

Desempenho

Por exemplo, a confiabilidade melhora com o uso de componentes redundantes. Todavia, o desempenho fica bastante comprometido, implicando na sua redução

Obrigado!

Guilherme Henrique Pasqualin Algeri
guilherme.algeri@sistemafiep.org.br
(42) 9 9148-8117