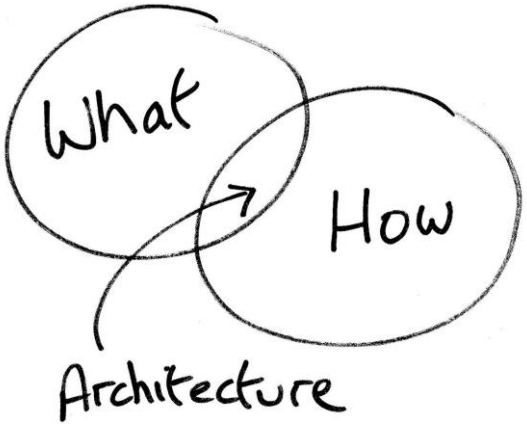
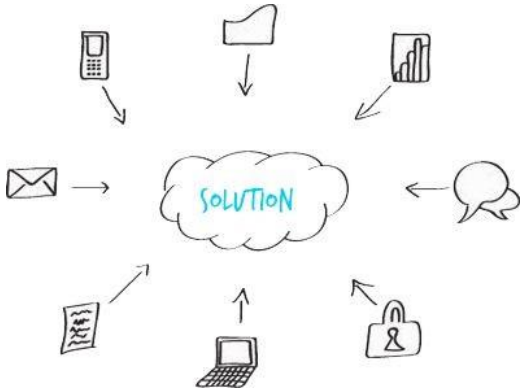


Arquitetura de Software: Qualidade

SSC -0527 Engenharia de Software – 2017
Profa. Dra. Elisa Yumi Nakagawa

Tiago Volpato



Índice

Requisitos

Requisitos abrangem as seguintes categorias:

1. **Requisitos Funcionais**: o que o sistema deve fazer e como deve reagir em tempo de execução a estímulos. São satisfeitos com a atribuição de responsabilidades a elementos durante o projeto.
2. **Atributos de Qualidade**: **qualificações** dos requisitos funcionais ou do produto global. São satisfeitos com as várias estruturas concebidas para a arquitetura, e os comportamentos e interações dos elementos dessas estruturas.
3. **Restrições**: uma **decisão** de design com zero graus de liberdade. São satisfeitas ao aceitar a decisão e conciliá-la com outras decisões de design afetadas.

Atributos de Qualidade

Se um requisito funcional é "*Quando o usuário pressiona o botão verde, o diálogo Opções aparece*", um **atributo de qualidade** de desempenho, por exemplo, pode descrever "*o quão rapidamente o diálogo irá aparecer*", enquanto um atributo de qualidade de disponibilidade pode descrever "*a frequência com esta função irá falhar e quão rápido ela será reparada*".

A arquitetura provê a fundação para alcançar atributos de qualidade. No entanto essa fundação não é suficiente se o desenvolvimento não foi eficiente.

Cenários de Atributos de Qualidade

Existia o problema de **especificação** de atributos de qualidade devido a divergências entre partes interessadas (*stakeholders*) e a comunidade com relação a:

- diferentes definições e vocabulário
- classificação de requisitos em qualidades
- validação dos atributos de qualidade

A solução para estes problemas é a utilização de **cenários de atributos** de qualidade, como um meio de caracterizar os atributo, e a **definição da descrição** detalhada de cada atributo.

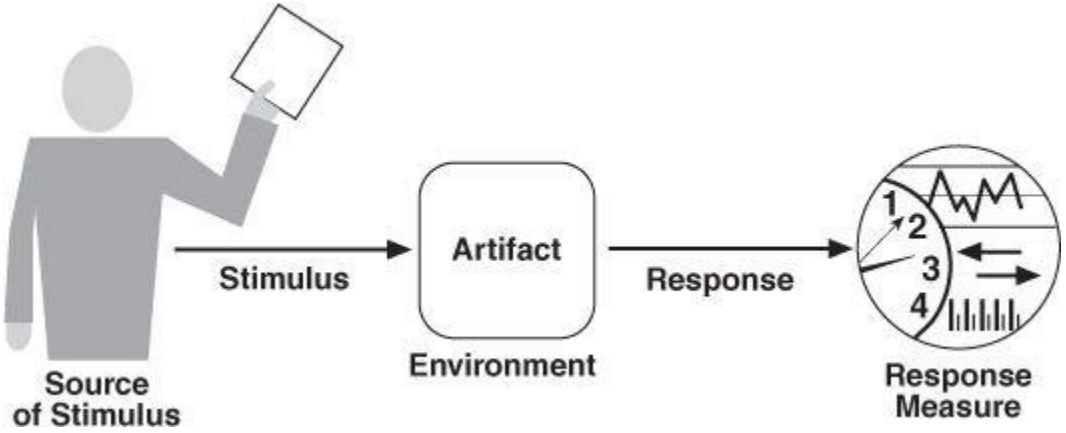
Cenários de Atributos de Qualidade

A especificação por meio de cenários de atributos de qualidade contém 6 partes:

1. **Estímulo**: um evento que chega no sistema (e.g., mensagem, operação de usuário, ataque).
2. **Fonte de estímulo**: entidade que gera um estímulo.
3. **Artefato**: a porção do sistema na qual o requisito se aplica.
4. **Ambiente**: é o conjunto de circunstâncias em que o cenário acontece.
5. **Resposta**: é como o sistema deve reagir ao estímulo recebido.
6. **Medida de Resposta**: forma de medir a resposta e determinar se é satisfatória.

Cenários de Atributos de Qualidade

Os **requisitos** de um sistema **particular** devem ser **especificados** por meio de cenários. Cenário genéricos devem ser "traduzidos" para requisitos específicos.



Bass et al. 2012

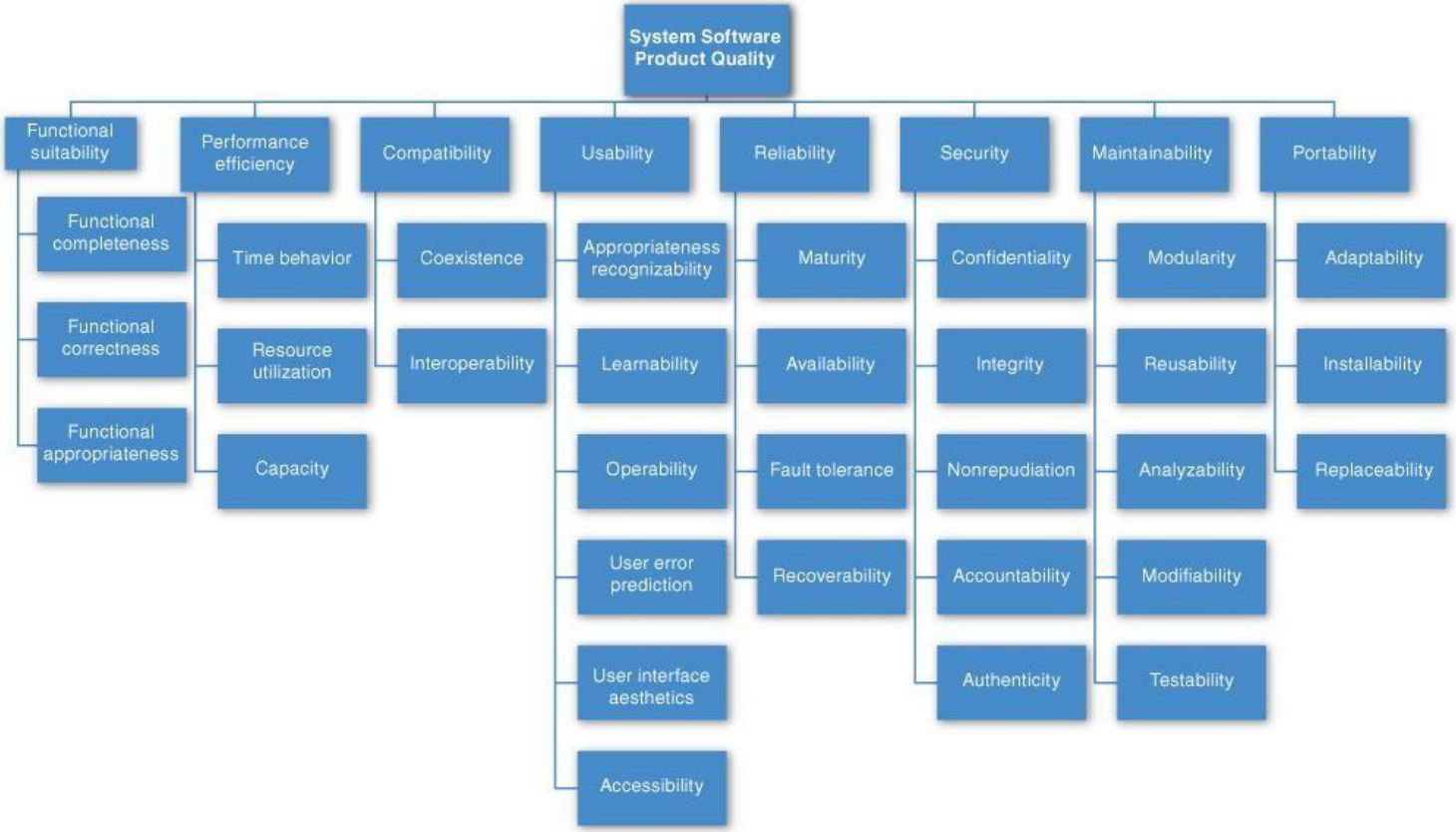
Modelo de Qualidade ISO/IEC 25010:2011

O **modelo de qualidade** externa e interna da ISO/IEC 25010 **categoriza** atributos de qualidade de software em seis características:

1. funcionalidade
2. confiabilidade
3. usabilidade
4. eficiência
5. manutenibilidade
6. portabilidade

as quais são subdivididas em subcaracterísticas.

Modelo de Qualidade ISO/IEC 25010:2011



Modelo de Qualidade ISO/IEC 25010:2011

Observação:

Disponibilidade é a capacidade de um produto de software de estar pronto para executar uma função requisitada num dado momento sob condições específicas de uso.

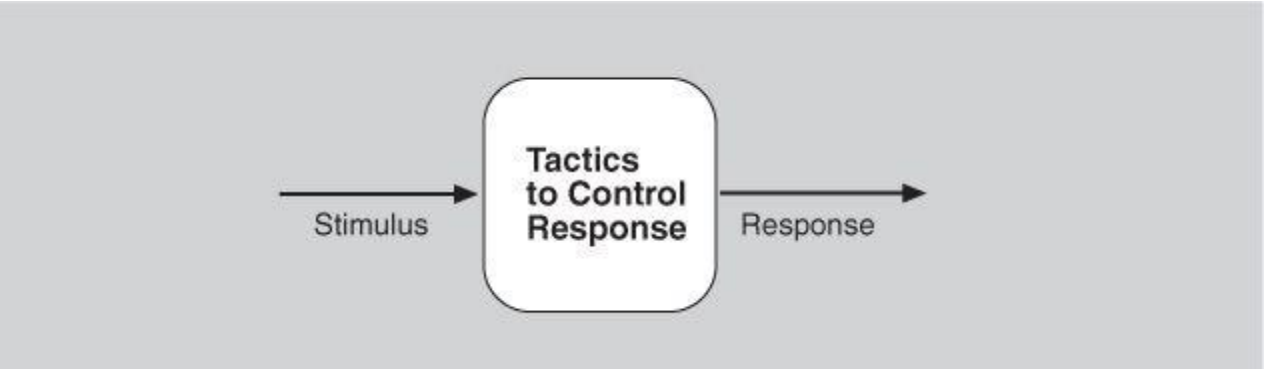
A **disponibilidade** é a combinação de **maturidade**, **tolerância a falhas** e **recuperabilidade**.

(Bass et al. 2003: Disponibilidade, Modificabilidade, Desempenho, Segurança, Testabilidade, Usabilidade)

Técnicas para Alcançar Atributos de Qualidade

As técnicas que um arquiteto pode usar para atingir os atributos de qualidade exigidos são chamadas **táticas arquiteturais**.

Uma tática é uma **decisão de design** que influencia a concretização de uma resposta de um atributo de qualidade.



Bass et al. 2012

Atributos de Qualidade

1. Disponibilidade
2. Modificabilidade
3. Desempenho
4. Segurança
5. Testabilidade
6. Usabilidade
7. Outros Atributos:
    a. Qualidades de Negócio
    b. Qualidades de Arquitetura

Bass et al. 2003

Disponibilidade

Disponibilidade refere-se a propriedade do software de estar **acessível e pronto** para executar uma tarefa quando solicitada.

Refere-se à capacidade de um sistema em **mascarar ou reparar falhas** de modo que o período cumulativo de interrupção de serviço não seja superior a um valor desejado num intervalo de tempo especificado.

Em outras palavras, **disponibilidade é sobre minimizar o tempo de interrupção do serviço ao atenuar falhas.**

A disponibilidade pode ser calculada: $\frac{MTBF}{(MTBF + MTTR)}$

Disponibilidade

Erro (*fault*) x Falha (*failure*): um erro pode se transformar em uma falha se não for corrigido ou mascarado.

4 tipos de erros:

- **Omissão**: um componente não responde a uma entrada (*input*).
- ***Crash***: o componente sofre falhas de omissão repetidamente.
- ***Timing***: um componente responde mas sua resposta ocorre antes ou depois do esperado.
- **Resposta**: um componente responde com um valor incorreto.

Disponibilidade

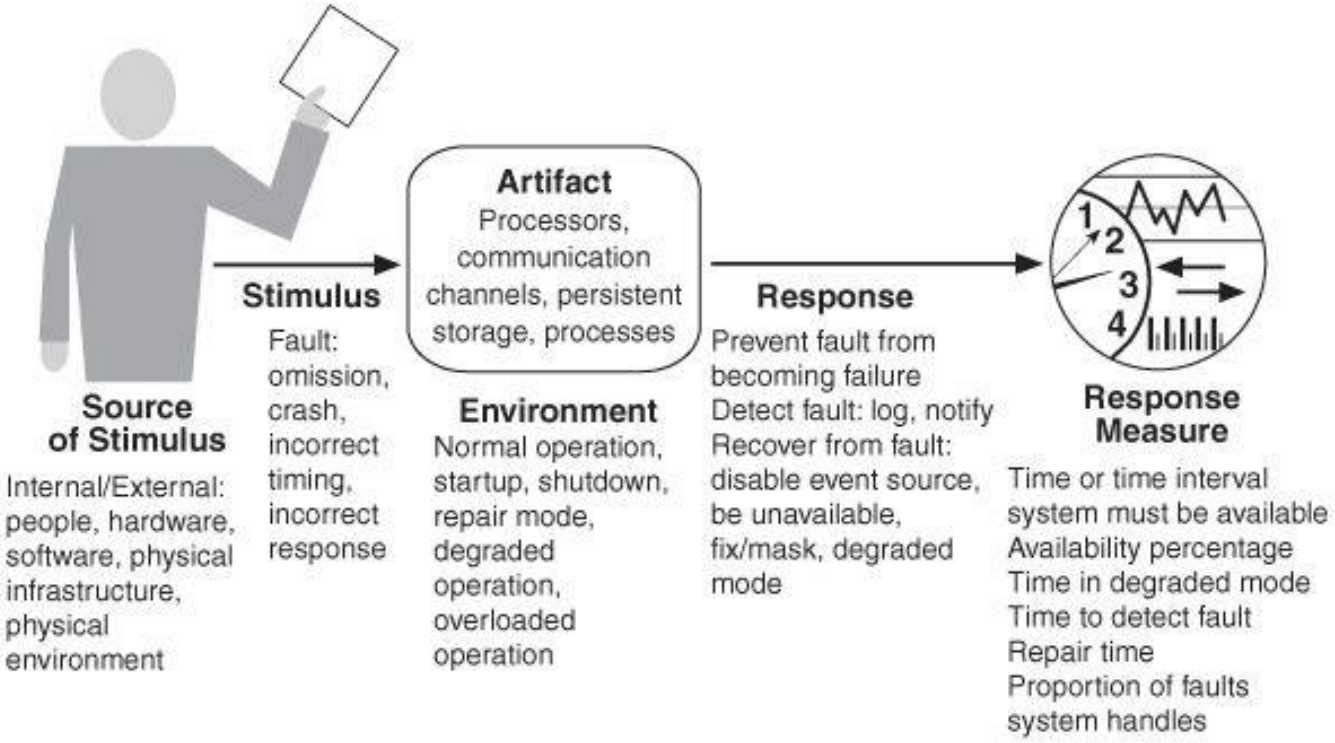
Cenário Geral

Portion of Scenario	Possible Values
Source	Internal/external: people, hardware, software, physical infrastructure, physical environment
Stimulus	Fault: omission, crash, incorrect timing, incorrect response
Artifact	Processors, communication channels, persistent storage, processes
Environment	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation
Response	Prevent the fault from becoming a failure Detect the fault: ▪ Log the fault ▪ Notify appropriate entities (people or systems) Recover from the fault: ▪ Disable source of events causing the fault ▪ Be temporarily unavailable while repair is being effected ▪ Fix or mask the fault/failure or contain the damage it causes ▪ Operate in a degraded mode while repair is being effected
Response Measure	Time or time interval when the system must be available Availability percentage (e.g., 99.999%) Time to detect the fault Time to repair the fault Time or time interval in which system can be in degraded mode Proportion (e.g., 99%) or rate (e.g., up to 100 per second) of a certain class of faults that the system prevents, or handles without failing

Bass et al. 2012

Disponibilidade

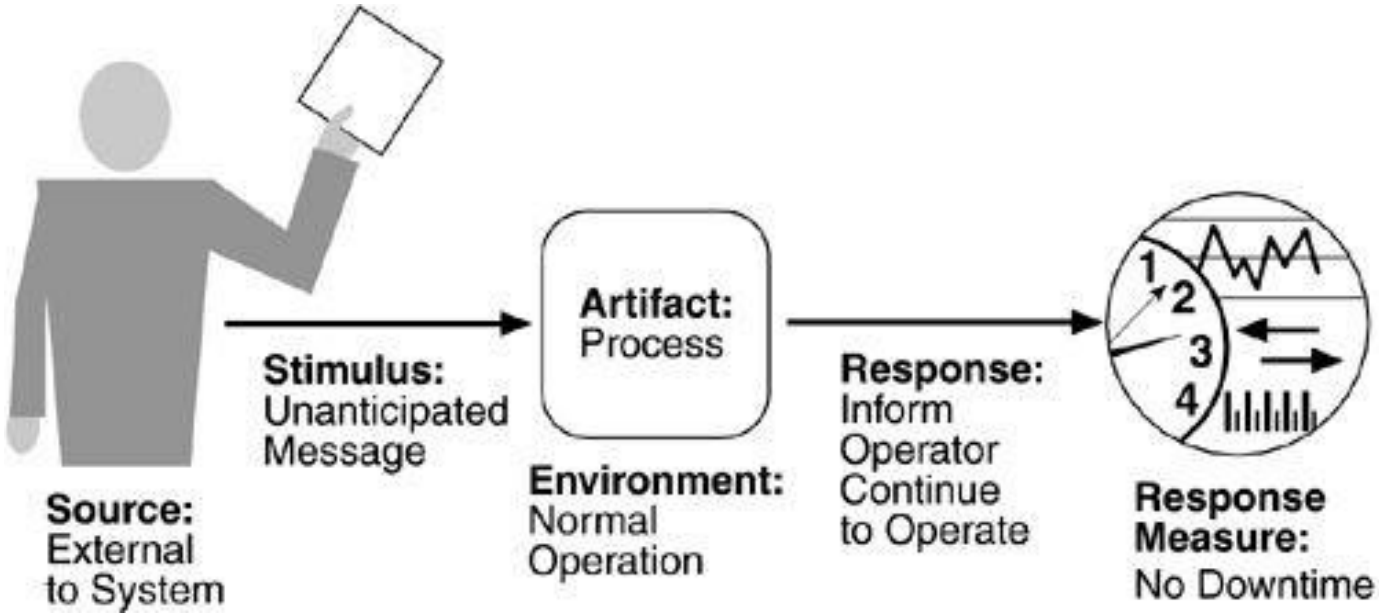
Cenário Geral



Bass et al. 2012

Disponibilidade

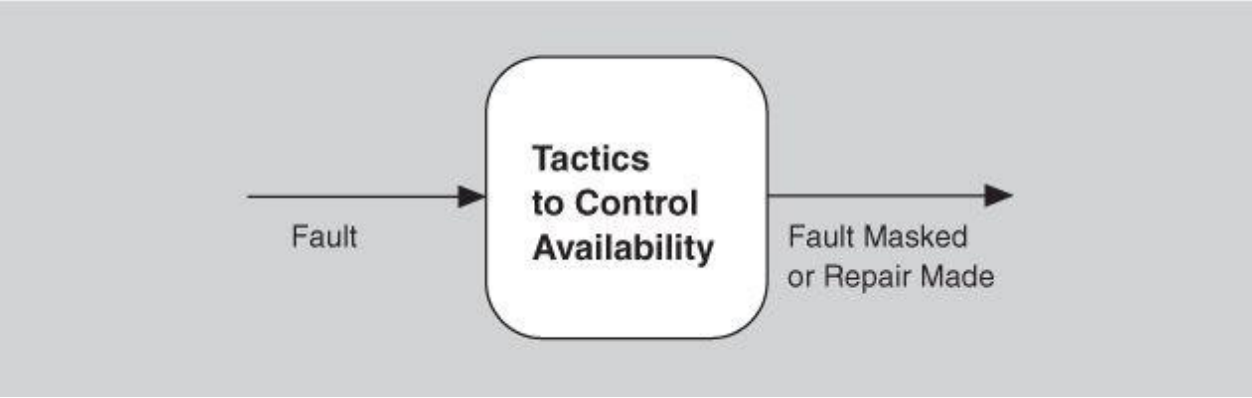
Exemplo de Cenário Concreto



Bass et al. 2003

Disponibilidade

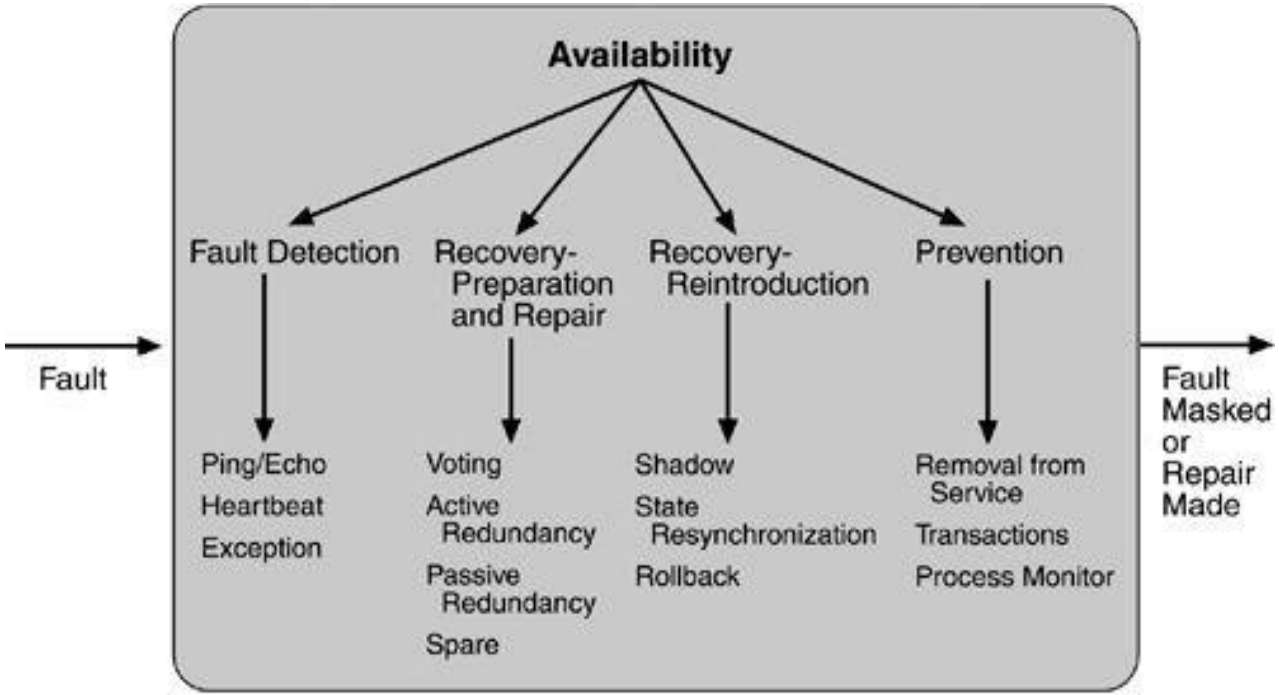
A aplicação de uma **tática arquitetural** para controlar a disponibilidade do sistema permite que o **erro** seja **reparado** ou **mascarado**, de acordo com o erro ocorrido e estratégia escolhida.



Bass et al. 2012

Disponibilidade

Táticas



Bass et al. 2003

Modificabilidade

A modificabilidade está preocupada com os **custo de mudanças no sistema**. Vários **tipos de mudanças** podem ser necessários (e.g., funcionalidades, mudança de plataforma, aumento de capacidade) em **diferentes fases do projeto**.

Questões que devem ser consideradas: O que mudar? Probabilidade? Quando? Por quem? Custo?

A aplicação de **uma tática diminui o custo de mudanças**:

N x Custo da Mudança ≤ Custo do Mecanismo + (N x Custo da Mudança)

Onde N é o número estimado de modificações.

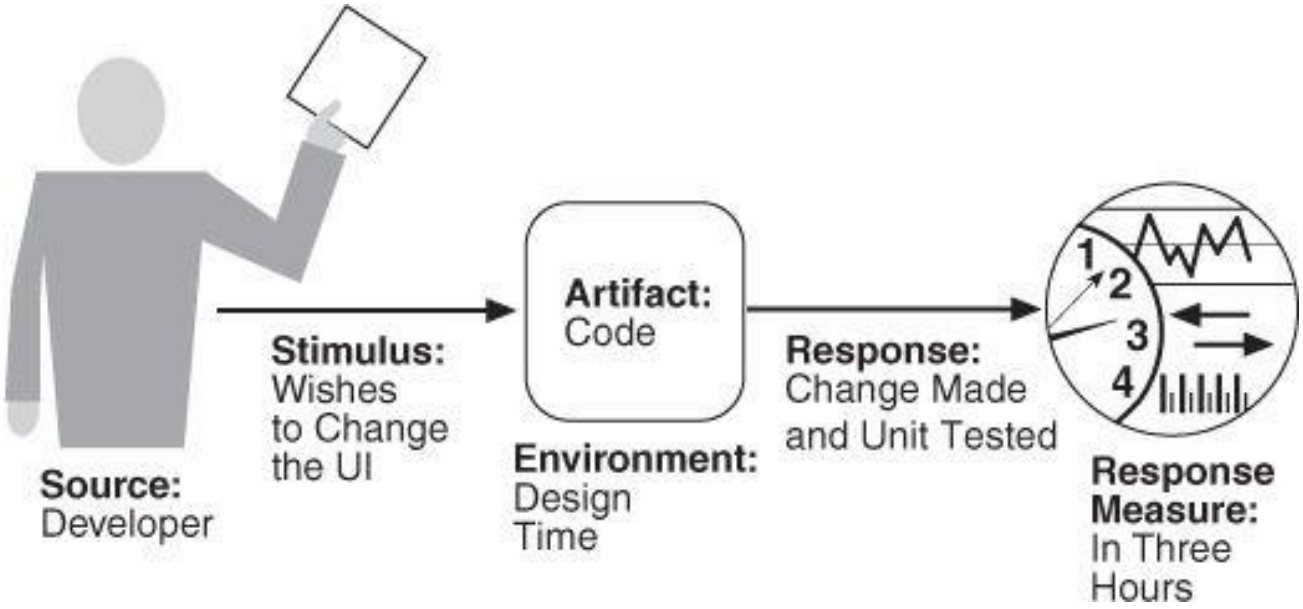
Modificabilidade Cenário Geral

Portion of Scenario	Possible Values
Source	End user, developer, system administrator
Stimulus	A directive to add/delete/modify functionality, or change a quality attribute, capacity, or technology
Artifacts	Code, data, interfaces, components, resources, configurations, . . .
Environment	Runtime, compile time, build time, initiation time, design time
Response	One or more of the following: ▪ Make modification ▪ Test modification ▪ Deploy modification
Response Measure	Cost in terms of the following: ▪ Number, size, complexity of affected artifacts ▪ Effort ▪ Calendar time ▪ Money (direct outlay or opportunity cost) ▪ Extent to which this modification affects other functions or quality attributes ▪ New defects introduced

Bass et al. 2012

Modificabilidade

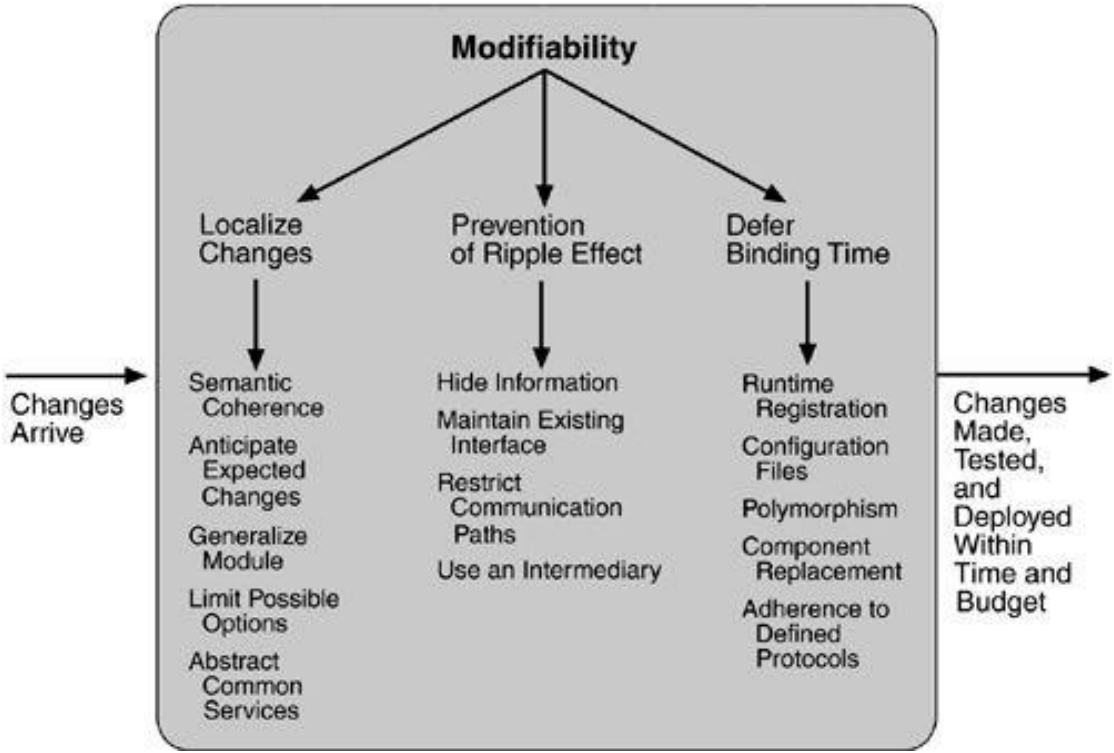
Exemplo de Cenário Concreto



Bass et al. 2012

Modificabilidade

Táticas



Bass et al. 2003

Desempenho

O desempenho está relacionado ao tempo: **quanto tempo o sistema demora para responder** quando um determinado **evento** acontece.

Historicamente, o desempenho tem sido um **atributo chave** da arquitetura de sistemas. Ao mesmo tempo, é geralmente o atributo **mais comprometido** para alcançar os outros.

Dentro do cenário é essencial caracterizar o **tipo de evento** (e.g., requisições, mensagens, interrupções) e o **tempo** relacionado esperado (e.g., eventos por segundo, tempo de processamento).

Desempenho

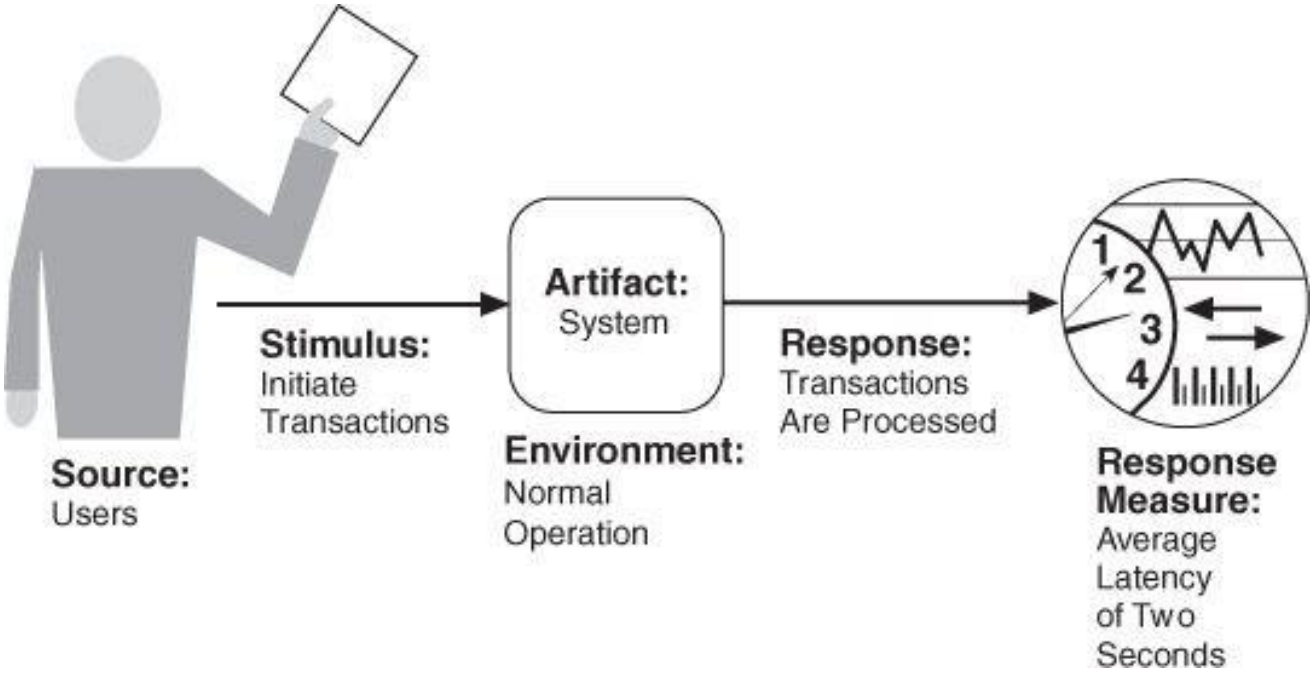
Cenário Geral

Portion of Scenario	Possible Values
Source	Internal or external to the system
Stimulus	Arrival of a periodic, sporadic, or stochastic event
Artifact	System or one or more components in the system
Environment	Operational mode: normal, emergency, peak load, overload
Response	Process events, change level of service
Response Measure	Latency, deadline, throughput, jitter, miss rate

Bass et al. 2012

Desempenho

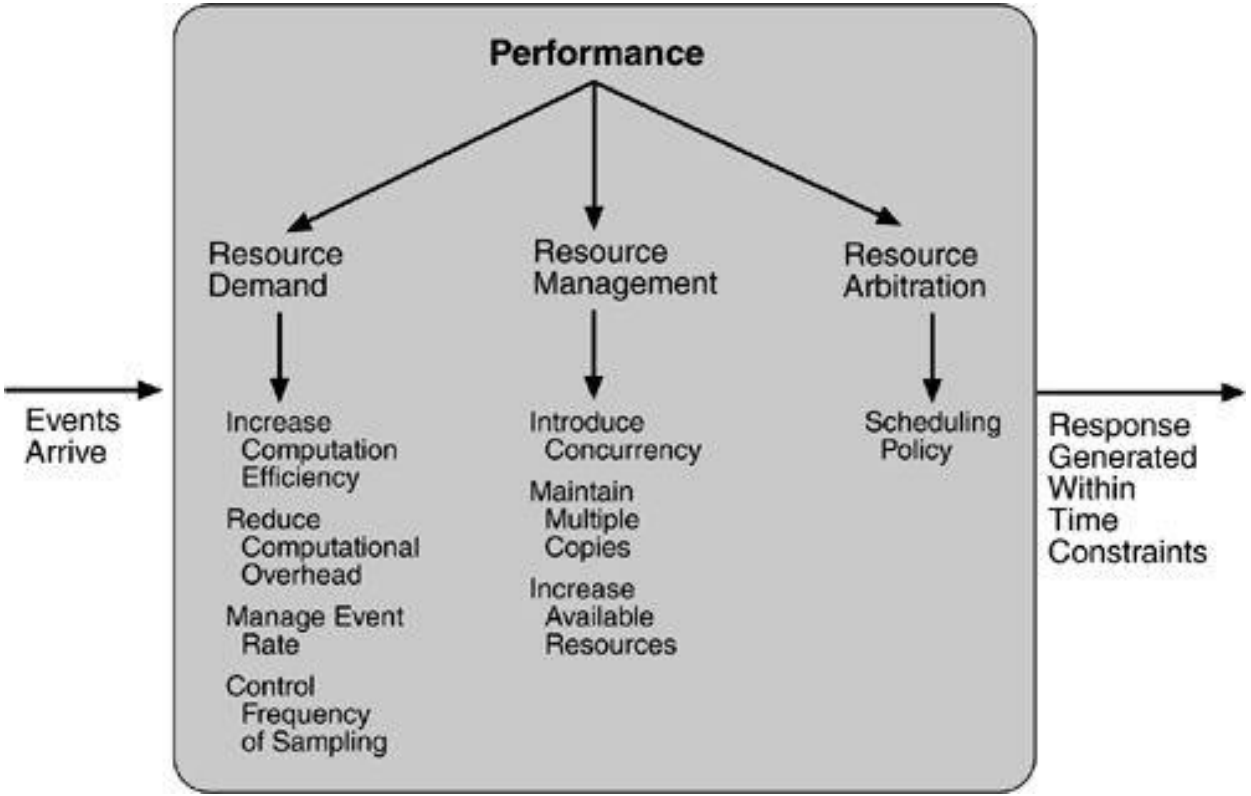
Exemplo de Cenário Concreto



Bass et al. 2012

Desempenho

Táticas



Bass et al. 2003

Segurança

Segurança é a medida da habilidade de um sistema de **proteger dados e resistir a acessos não autorizados** enquanto proveem serviços a usuários e sistemas que são autorizados. Segurança pode ser caracterizada pelas características:

1. **Confidencialidade**: garante que dados ou serviços são protegidos contra o acesso não autorizado.
2. **Integridade**: garante que dados ou serviços não estão sujeitos a manipulação não autorizada.
3. **Disponibilidade**: garante que o sistema estará disponível para uso legítimo.
4. **Autenticação**: garante ambas as partes de uma transação são quem eles dizem ser.
5. **Não Repudiação**: garante que o remetente de uma mensagem não pode negar ter enviado e que o destinatário não pode negar ter recebido.
6. **Auditoria**: garante o rastreamento de atividades de forma que o sistema possa reconstruí-las.

Segurança

Cenário Geral

Portion of Scenario	Possible Values
Source	Human or another system which may have been previously identified (either correctly or incorrectly) or may be currently unknown. A human attacker may be from outside the organization or from inside the organization.
Stimulus	Unauthorized attempt is made to display data, change or delete data, access system services, change the system's behavior, or reduce availability.
Artifact	System services, data within the system, a component or resources of the system, data produced or consumed by the system
Environment	The system is either online or offline; either connected to or disconnected from a network; either behind a firewall or open to a network; fully operational, partially operational, or not operational.
Response	Transactions are carried out in a fashion such that ▪ Data or services are protected from unauthorized access. ▪ Data or services are not being manipulated without authorization. ▪ Parties to a transaction are identified with assurance. ▪ The parties to the transaction cannot repudiate their involvements.

Bass et al. 2012

Segurança

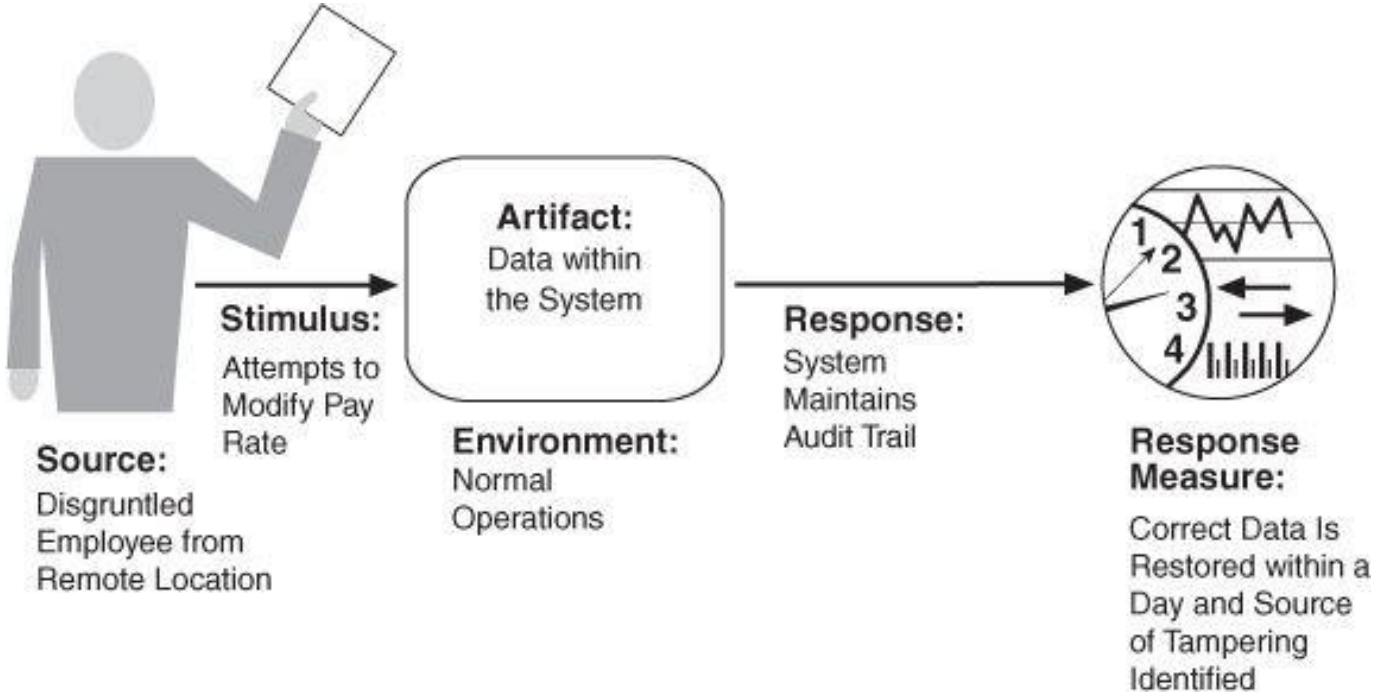
Cenário Geral

	▪ The data, resources, and system services will be available for legitimate use. The system tracks activities within it by ▪ Recording access or modification ▪ Recording attempts to access data, resources, or services ▪ Notifying appropriate entities (people or systems) when an apparent attack is occurring
Response Measure	One or more of the following: ▪ How much of a system is compromised when a particular component or data value is compromised ▪ How much time passed before an attack was detected ▪ How many attacks were resisted ▪ How long does it take to recover from a successful attack ▪ How much data is vulnerable to a particular attack

Bass et al. 2012

Segurança

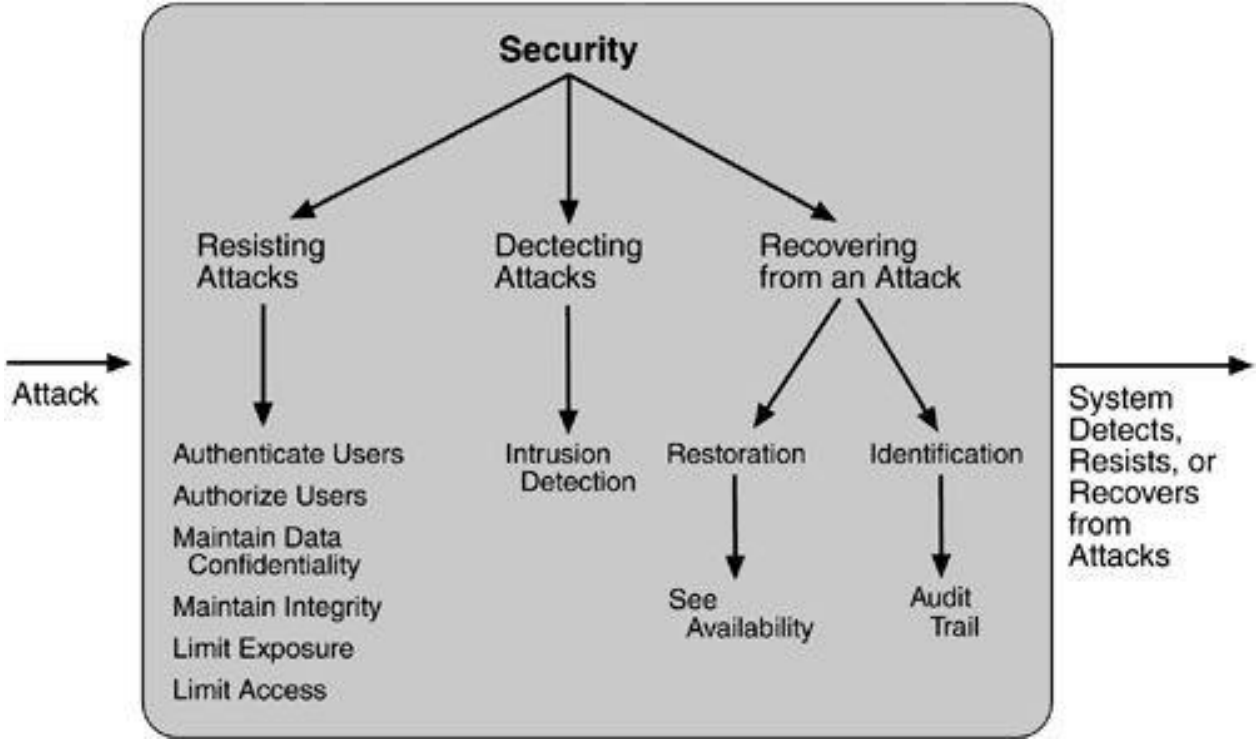
Exemplo de Cenário Concreto



Bass et al. 2012

Segurança

Táticas



Bass et al. 2003

Testabilidade

Se refere à **facilidade** com que um software pode ser feito para **demonstrar seus erros por meio de testes**.

Testes são uma **etapa cara** do processo de desenvolvimento. Se arquitetos conseguirem fazer um sistema mais facilmente testável, o custo de desenvolvimento cai bastante.

Um sistema é considerado altamente testável se revela suas falhas facilmente.

Testabilidade

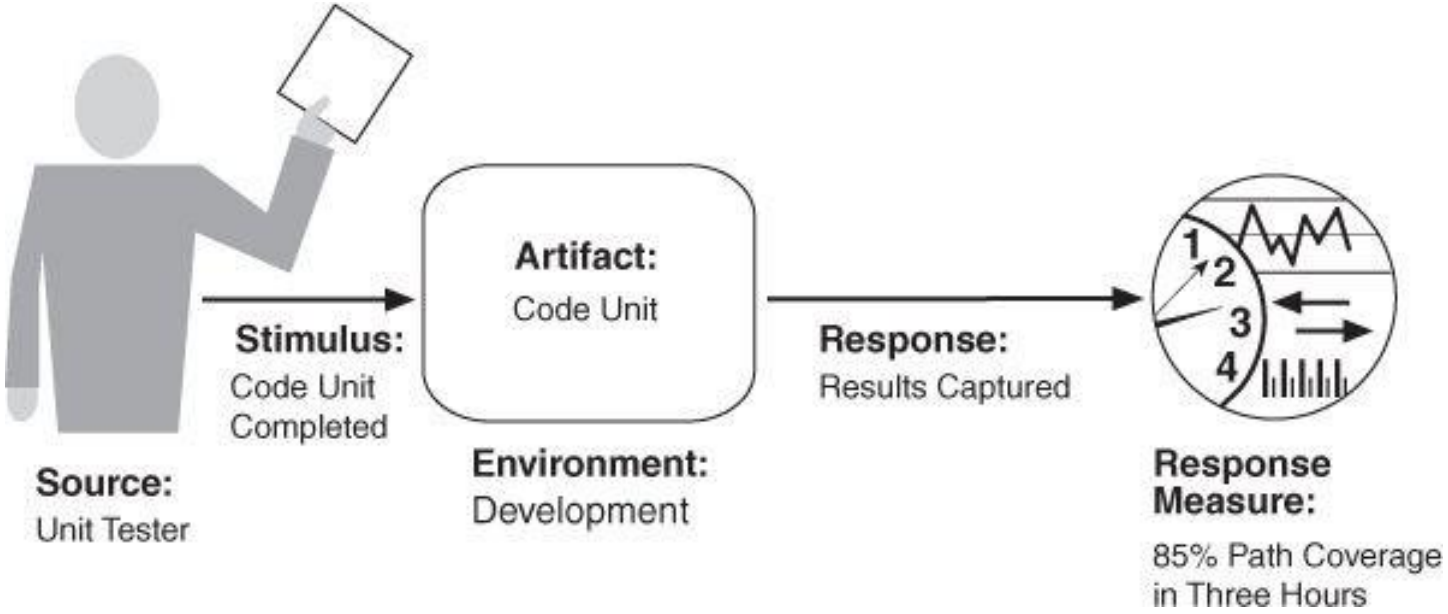
Cenário Geral

Portion of Scenario	Possible Values
Source	Unit testers, integration testers, system testers, acceptance testers, end users, either running tests manually or using automated testing tools
Stimulus	A set of tests is executed due to the completion of a coding increment such as a class layer or service, the completed integration of a subsystem, the complete implementation of the whole system, or the delivery of the system to the customer.
Environment	Design time, development time, compile time, integration time, deployment time, run time
Artifacts	The portion of the system being tested
Response	One or more of the following: execute test suite and capture results, capture activity that resulted in the fault, control and monitor the state of the system
Response Measure	One or more of the following: effort to find a fault or class of faults, effort to achieve a given percentage of state space coverage, probability of fault being revealed by the next test, time to perform tests, effort to detect faults, length of longest dependency chain in test, length of time to prepare test environment, reduction in risk exposure (size(loss) × prob(loss))

Bass et al. 2012

Testabilidade

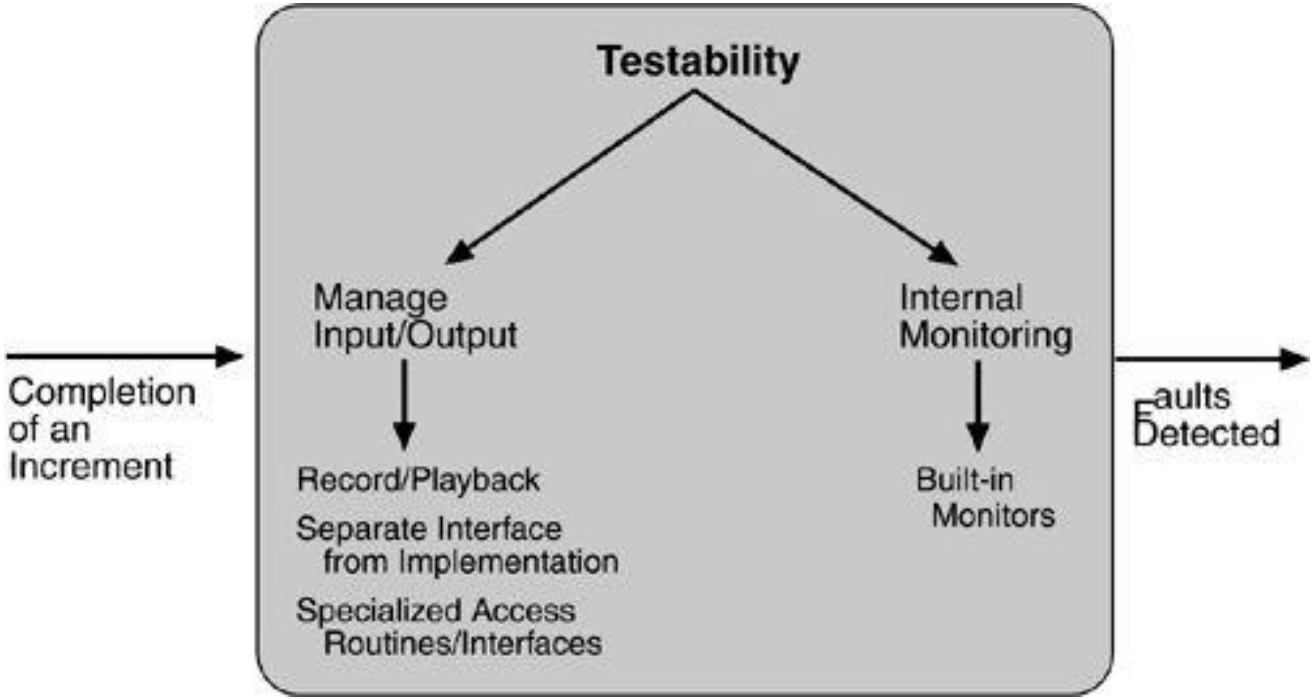
Exemplo de Cenário Concreto



Bass et al. 2012

Testabilidade

Táticas



Bass et al. 2003

Usabilidade

A usabilidade se preocupa com **o quão fácil é para o usuário para realizar tarefas** e o **tipo de suporte ao usuário** que o sistema oferece. A usabilidade compreende os seguintes tópicos:

- Aprendizado de recursos do sistema
- Eficiência de uso
- Minimização do impacto de erros
- Adaptação do sistema às necessidades do usuário
- Aumentar a confiança e satisfação do usuário

Usabilidade

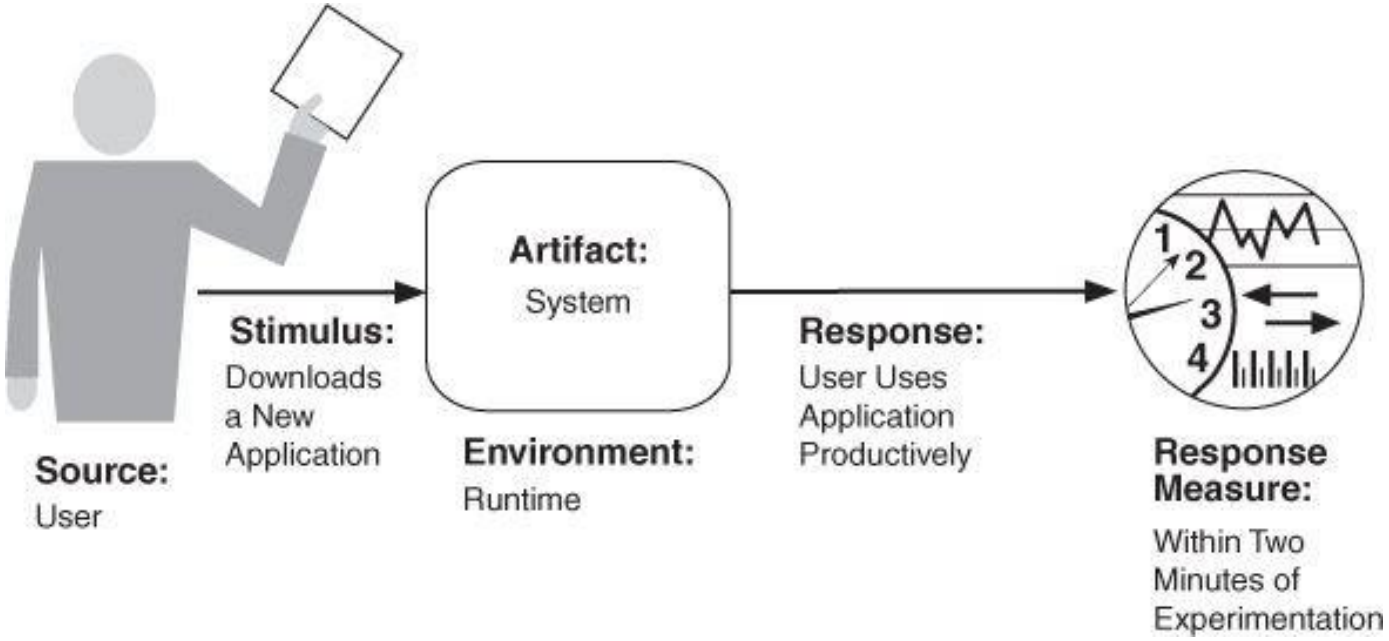
Cenário Geral

Portion of Scenario	Possible Values
Source	End user, possibly in a specialized role
Stimulus	End user tries to use a system efficiently, learn to use the system, minimize the impact of errors, adapt the system, or configure the system.
Environment	Runtime or configuration time
Artifacts	System or the specific portion of the system with which the user is interacting
Response	The system should either provide the user with the features needed or anticipate the user's needs.
Response Measure	One or more of the following: task time, number of errors, number of tasks accomplished, user satisfaction, gain of user knowledge, ratio of successful operations to total operations, or amount of time or data lost when an error occurs

Bass et al. 2012

Usabilidade

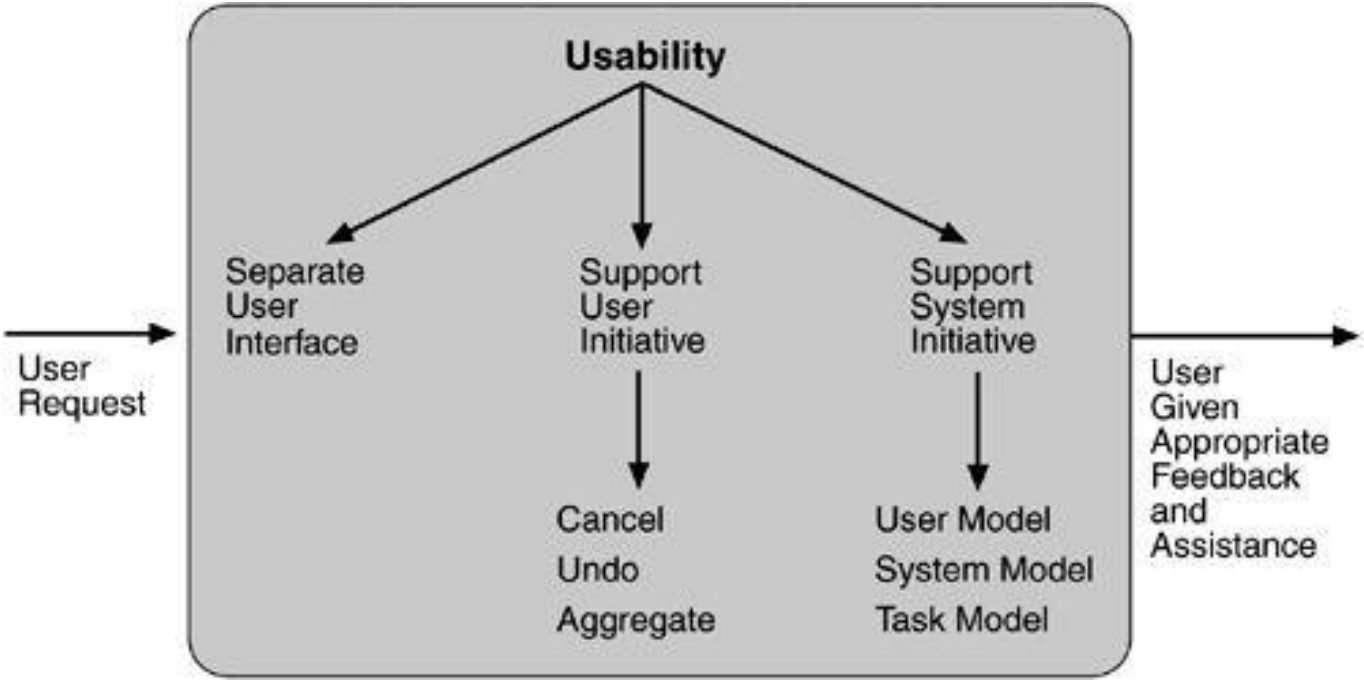
Exemplo de Cenário Concreto



Bass et al. 2012

Usabilidade

Táticas



Bass et al. 2003

Requisitos Arquiteturais

- Requisitos de potencial valor para a arquitetura de um sistema
- Esses requisitos são como a parte submersa de um iceberg. Ninguém vê! Mas acredite! É o que dá sustentação.
- Nem todos os requisitos têm igual importância no que diz respeito à arquitetura

Requisitos Arquiteturalmente Significantes

- Desempenham um papel importante na determinação da arquitetura do sistema

A inclusão desses requisitos, muito provavelmente, resultará em uma arquitetura diferente daquela em que eles não foram incluídos.

- São um subconjunto dos requisitos que precisam ser atendidos antes que a arquitetura possa ser considerada "estável".

"***Significante***" é o termo chave da definição!

Requisitos Arquiteturalmente Significantes

A seleção de requisitos que são considerados "**arquiteturalmente significantes**" pode-se orientar por vários fatores:

- O **benefício** dos requisitos para os *stakeholders*: crítico, importante ou útil .
- O **impacto** arquitetural do requisito: nenhum , extensível, ou modificável.
- Os **riscos** a serem mitigados: desempenho, a disponibilidade de um produto, e adequação de um componente.
- Outros objetivos ou restrições táticas, como demonstração para o usuário, e assim por diante.

Requisitos Arquiteturalmente Significantes

Exemplos de ASRs:

- O sistema deve gravar cada modificação de registros dos clientes para fins de auditoria.
- O sistema deve responder no prazo de 5 segundos.
- O sistema deve ser implantado no Microsoft Windows XP e Linux.
- O sistema deve criptografar todo o tráfego de rede.

Coletando ASRs

ASRs assumem muitas vezes, **mas nem sempre**, a forma de atributos de qualidade

Existem algumas técnicas para alcançar e priorizar os atributos de qualidade para determinar ASRs:

- Documentos de requisitos
- Entrevistas com *stakeholders* (*Workshop* de Atributos de Qualidade - QAW)
- Entendimento dos objetivos de negócio
- Árvore de utilidade
- FURPS+

Coletando ASRs

Documentos de Requisitos:

Embora os documentos de requisitos não conte a um arquiteto toda a história, eles são uma importante fonte de ASRs.

Documentos de requisitos falham em:

1. A maioria do que está em uma especificação de requisitos não afeta a arquitetura.
2. Muito do que é útil para um arquiteto não está nem mesmo no melhor documento de requisitos.

Coletando ASRs

Workshop de Atributos de Qualidade (QAW):

Método fácil e focado nos *stakeholders*

Utilizado para gerar, priorizar e refinar cenários de atributos de qualidade

Procedimentos:

Passo 1. Apresentação do QAW
Passo 2. Apresentação do Negócio / Missão
Passo 3. Apresentação do Plano Arquitetural
Passo 4. Identificação dos condutores arquiteturais
Passo 5. Cenário de *brainstorming*
Passo 6. Cenário de consolidação
Passo 7. Cenário de priorização
Passo 8. Cenário de refinamento

Coletando ASRs

Objetivos de negócio:

Um método para elicitar e documentar os objetivos de negócio é o **Pedigreed Attribute eLicitation Method (PALM)**

- Procurar requisitos que estão faltando no início do ciclo de vida
- Descobrir e obter informações adicionais sobre os requisitos existentes.
- Examinar requisitos de atributos de qualidade complicados para ver se eles podem serrelaxados.

Coletando ASRs

Árvore de Utilidade:

Utilizada para gravar os requisitos em um só lugar

Procedimentos:

1. A árvore começa com a palavra "***utilidade***" como o nó raiz
2. Lista-se os atributos de maior qualidade que o sistema requer que sejam apresentados
3. Faz-se um refinamento específico dos atributos de qualidade relevantes para o sistema
4. Avalia-se com os ASRs com base em dois critérios:
   (a)o valor comercial do candidato a ASR e (b) o impacto arquitetural de incluí-lo.

Coletando ASRs

Árvore de Utilidade:

Uma vez que você tem uma árvore de utilidade preenchido, você pode usá-lo para fazer verificaçõesimportantes:

- Um QA ou um refinamento de QA sem qualquer ASR não é necessariamente um erro
- ASRs que são classificados com (H, H) são, obviamente, os que merecem mais atenção
- Revisão pelos*stakeholders*

É uma boa maneira de capturar ASRs junto com sua ordem de prioridade.

Coletando ASRs

FURPS+:

Sistema para classificar requisitos arquiteturais e assegurar que declarações valiosas para o sistema não sejam negligenciadas.

O acrônimo "FURPS" representa as seguintes categorias:

- **F** = Functionality (Funcionalidade)
- **U** = Usability (Usabilidade)
- **R** = Reliability (Confiabilidade)
- **P** = Performance (Execução)
- **S** = Supportability (Suportabilidade)
- **+** = Restrições (de projeto, de implementação, de integração, físicas)

Coletando ASRs

FURPS+:

Procedimentos:

1. Manter uma lista completa dos requisitos arquiteturais
2. Formular questões que possam ajudar no processo de especificação
3. Ajudar os *stakeholders* mostrando os impactos em potencial
4. Capturar as respostas dos *stakeholders* para cada uma das questões.
5. Atribuir prioridade ou peso para cada requisito arquitetural

Informações Úteis

O termo ***Requisito Arquiteturalmente Significante*** foi criado pelo grupo Software Architecture Review and Assessment (SARA)

O Open Group Architecture Framework fornece um template completo para documentar cenários de negócios que contenham um informações úteis.

A fonte de referência definitiva para o Workshop de Atributos de Qualidade é http://www.sei.cmu.edu/reports/03tr016.pdf

Os detalhes completos da Palm pode ser encontrado em "*Relating Business Goals to Architecturally Significant Requirements for Software Systems*"

Referências

Bass, L., Clements, P., & Kazman, R. (2003). Software Architecture in Practice. Addison-Wesley Professional, 2nd edition.

Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice. Addison-Wesley Professional, 3rd edition.

ISO (2011). ISO/IEC 25010:2011: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models. Thecnical report, International Organization for Standardization (ISO), Geneva, Switzerland.

Barbosa, G. (2009). Um Livro-texto para o Ensino de Projeto de Arquitetura de Software. Master's thesis, Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.

BASTOS JUNIOR, P. R. D. O. (2005). Elicitação de requisitos de software através da utilização de questionários. Master's thesis, PUC-Rio, Departamento de Informática.