Faculdade XPe

Capítulo 03

ANÁLISE DE REQUISITOS ARQUITETURAIS

Vinicius Fernandes de Almeida

2024

Priorização de requisitos

Uma tarefa extremamente importante no trabalho de analistas de requisitos é a priorização para garantir a máxima geração de valor das atividades do time de desenvolvimento. Há duas ferramentas que podem ser úteis para a priorização:

- Kano Model.
- Cost of Delay (Custo do Atraso).

Kano Model

Cost of Delay

Kano Model

Este modelo divide a funcionalidade conforme o comportamento delas, quando analisamos um gráfico com dois eixos. No eixo vertical analisamos a satisfação dos usuários com uma determinada funcionalidade. No eixo horizontal analisamos o investimento feito na funcionalidade, resultando em algo melhor ou pior implementado. Temos então três categorias de funcionalidades:

1. **Expectativas básicas (basic expectations):** são funcionalidades básicas que necessariamente devem estar presentes e bem implementadas. Caso esteja mal implementada vão ter um impacto muito grande na satisfação dos usuários.

2. **Satisfação (Satisfiers):** são funcionalidades com um comportamento linear entre investimento e satisfação. Se estiverem bem implementadas resultarão em grande satisfação e se estiverem mal implementadas terão péssimo impacto na satisfação.

3. **Encantamento (Delighters):** são funcionalidades que se estiverem ausentes não terão impacto na satisfação dos usuários, mas se estiverem presentes a satisfação com aquela funcionalidade será muito alta.

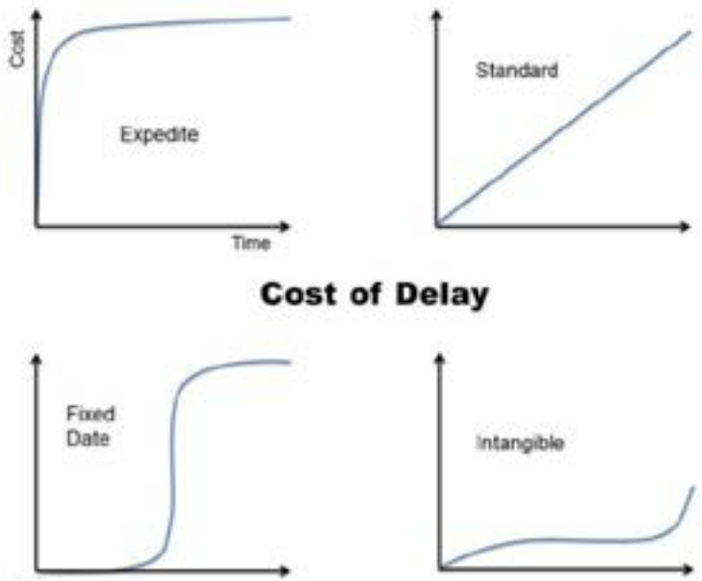
Figura 1 - Gráfico Kano Model.

Fonte: google.com

Cost of Delay (Custo do Atraso)

Na técnica de Cost of Delay consideramos o custo que uma organização tem ao “não ter” uma funcionalidade. Descobrimos “quanto custa ficar sem uma funcionalidade” para poder estimar o custo do atraso. Há alguns comportamentos em relação ao tipo de funcionalidade, como podemos ver na Figura:

Figura 2 - Evolução do custo do atraso conforme o tipo de demanda.



Fonte: google.com

Para calcular um índice de priorização conforme o atraso, dividimos o custo do atraso pela duração da funcionalidade. Quanto maior o resultado, mais prioritária deve ser uma tarefa. Dessa forma, ligamos a priorização diretamente aos custos financeiros.

Figura 3 - Índice de custo do atraso dividido pela duração.

Fonte: google.com

Na Figura 4 podemos visualizar um exemplo de cálculo com três funcionalidades com custos de atraso e duração diferentes.

Figura 4 - Exemplos de cálculo de custo do atraso usando o CD3

	CoD por semana	Duração Planejada	Score
Funcionalidade A	R$ 2.000	5 semanas	400
Funcionalidade B	R$ 30.000	6 semanas	5.000
Funcionalidade C	R$ 8.000	2 semanas	4.000

Fonte: google.com

Escrevendo Histórias de Usuários

Uma história de usuário é, essencialmente, uma breve descrição de um requisito do ponto de vista de um usuário do sistema. A descrição é feita como se o usuário estivesse descrevendo uma de suas necessidades e justificando-a. Um exemplo de história de usuário para um sistema de gestão de projetos seria:

Eu, gerente, preciso acompanhar as atividades de minha equipe porque preciso garantir que as atividades sejam executadas dentro do prazo.

As histórias de usuários visam promover a conversa entre a equipe. Ao contrário de uma especificação formal de requisitos, ela é incompleta por definição. Dessa forma é essencial que a equipe de desenvolvimento esteja em contato constante com o Dono do Produto para que as dúvidas e os pontos em aberto sejam resolvidos e esclarecidos.

Para entender a forma incompleta das histórias de usuário, usamos a analogia da foto. Se você vê uma foto de um amigo na praia tomando uma água de coco, vai pensar apenas “Aqui está meu amigo na praia tomando água de coco”. Agora imagine outra situação, em que seu amigo te conta uma história dessa foto: ele conta como ele teve dificuldades para chegar na praia, diz que torceu seu pé no caminho até lá e que estava com tanta sede que teve que pagar R$ 50,00 naquele coco e que nem achou tão gostoso. Quando você olhar novamente para a foto não vai mais pensar apenas “Aqui está meu amigo na praia tomando água de coco”. Dessa vez você vai se lembrar de cada detalhe que ele te contou, vai pensar no quão caro foi aquele coco e pensar como o pé do seu amigo estaria doendo.

Nos dois casos temos a mesma foto, mas com a conversa a foto passa a ter muito mais valor e conteúdo. Da mesma forma funcionam as histórias de usuários. Quando vemos uma história de usuário ela está incompleta, mas quando conversamos com o Dono do Produto para entender as necessidades dos clientes tudo se torna mais claro. A partir da conversa a história tem mais valor. O time de desenvolvimento entende o problema e passa a enxergá-lo do ponto de vista do usuário. Trabalha-se então para resolver o problema do usuário.

As histórias de usuário são simples, mas é necessário bastante experiência para se escrever boas histórias e para explicá-las da melhor forma possível para toda a equipe. O objetivo é o alinhamento de conhecimento sobre o problema, e para isso a conversa constante e abundante é essencial. A disponibilidade do Dono do Produto para o time de desenvolvimento deve ser constante.

Débitos técnicos

Ao longo das atividades de desenvolvimento, o time se depara com situações em que se decide não usar as melhores práticas de codificação. Isso pode ser por questão do prazo de uma entrega, pela instabilidade dos requisitos etc. Quando isso acontece, um débito técnico é colocado no produto.

É importante gerenciar o débito técnico para que se garanta que os problemas serão resolvidos com o tempo, pois sem a devida gestão cada vez mais problemas são colocados e não são resolvidos. A longo prazo, o produto aumentará de complexidade e sua manutenção fica cada vez mais custosa.

Com a devida gestão o débito técnico é acompanhado e atividades de refatoração são realizadas para corrigir os problemas mais críticos.

Acessibilidade

Acessibilidade é a possibilidade de acesso a um lugar ou conjunto de lugar. No caso da web, uma interface com boa acessibilidade provê boa experiência de navegação para qualquer pessoa, independentemente de terem alguma deficiência ou não.

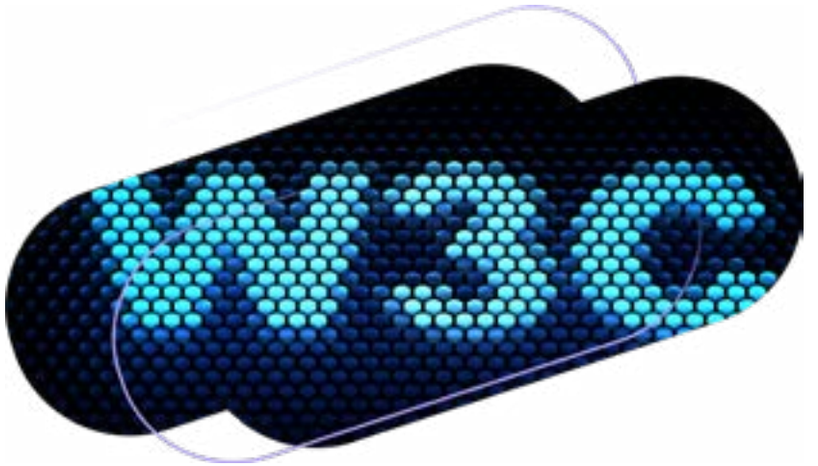
Problemas de sites não acessíveis

Há diversos tipos de deficiência que podem atrapalhar a utilização de um site não acessível. A tabela abaixo relaciona algumas dessas deficiências e os potenciais problemas que podem ter:

Tabela 1 - Exemplos de Problemas.

Deficiência	Exemplos de problemas em sites não acessíveis
Cegueira	Sem utilizar corretamente as semânticas HTML, descrição de imagens ou links bem nomeados, é difícil, até impossível, navegar no site com leitores de texto.
Daltonismo	Links diferenciados apenas pela cor podem não ser identificados por usuários daltônicos.
Problemas motores	Links e áreas clicáveis pequenas podem ser difíceis de acessar por usuários que não podem controlar o mouse com precisão.
Dislexia e dificuldades de aprendizagem	Conteúdo complexo e sem imagens instrucionais podem dificuldade a compreensão do conteúdo.
Deficiências de visão	Textos e imagens pequenas, e sem possibilidade de ampliação, dificultam a leitura e o entendimento por usuários com problemas de visão.

Fonte: google.com



Acessibilidade também diz respeito Fs recomendações do W3C para sites acessíveis. Existem diversos validadores disponíveis para verificar automaticamente se o site está aderente Fs recomendações da W3C.

O governo federal criou o eMAG – Modelo de Acessibilidade em Governo Eletrônico. Todos os sites e portais do governo brasileiro têm obrigação de implementá-lo. Disponível em: http://www.governoeletronico.gov.br/acoes-e- projetos/e-MAG

Por que investir em acessibilidade

Estudos mostram que cerca de 20% da população brasileira possui algum tipo de deficiência. Certamente nem todas essas pessoas possuem incapacidades que as impeçam de utilizar um site não acessível. Mesmo que o percentual de usuário incapazes de utilizar um site não acessível seja 10% ou até 5%, é imprudente para qualquer negócio excluir tal percentual de potenciais clientes.
Para uma loja on-line, por exemplo, alcançar 5% a mais de público pode representar mais vendas. Para uma rede social, representa mais usuários. Projetar um site acessível, então, é um investimento que deve ser considerado em projetos web.

Desempenho

As tecnologias de desenvolvimento web tem evoluído muito nos últimos anos. Com isso, estamos tentando fazer mais e mais coisas em nossas aplicações. Consequentemente, as aplicações estão ficando mais robustas e mais pesadas, podendo levar um longo tempo de carregamento e demora para os usuários.
Dessa forma um problema tem ficado muito evidente: desempenho. Quanto mais tentamos fazer sistemas robustos, capazes de fazer muitas coisas, mais o desempenho se torna uma questão relevante para desenvolvedores.

Existe um objetivo primário para se preocupar com desempenho: dar ao nosso usuário uma experiência melhor. Empresas conseguem aumentar consideravelmente a retenção de clientes e a rentabilidade de seus sites ao melhorar o desempenho da aplicação e, consequentemente, a experiência de seus usuários.
Aplicações são essenciais para cada vez mais atividades fundamentais para as pessoas e o desempenho das aplicações, que não deve ser uma barreira para que elas consigam alcançar seus objetivos.
Veremos várias formas de melhorar o desempenho das aplicações, mas você não precisa se preocupar em usar todas. Qualquer coisa que você puder fazer para melhorar o desempenho já poderá afetar positivamente a experiência do usuário.
Os problemas de desempenho se concentram em dois pontos principais da aplicação: no carregamento da página e na renderização na página. O carregamento da página é quando todos os recursos estáticos são carregados no servidor para o navegador. Isso significa que quanto maiores forem os arquivos, maior será o tempo de carregamento. O tempo de renderização diz respeito ao tempo que demora depois que todos os arquivos essenciais foram carregados, e o navegador vai precisar trabalhar para compilar, interpretar tudo e dispor os elementos de uma maneira que seja entendível visualmente.

Devemos tratar a questão de desempenho como uma questão estratégica para nossa aplicação. Vários estudos mostram que aplicações com melhor desempenho tem melhores resultados. Um site de vendas, por exemplo, poderá perder usuários ou perder vendas por conta de demora na resposta da aplicação. O usuário poderá fazer o cadastro no sistema porque aplicação demora muito, e ele acabou desistindo. Então devemos tratar desempenho como atributo diretamente ligado aos objetivos de negócio, por isso devemos decidir estrategicamente quanto tempo e esforço vamos investir para melhorar o desempenho da aplicação.

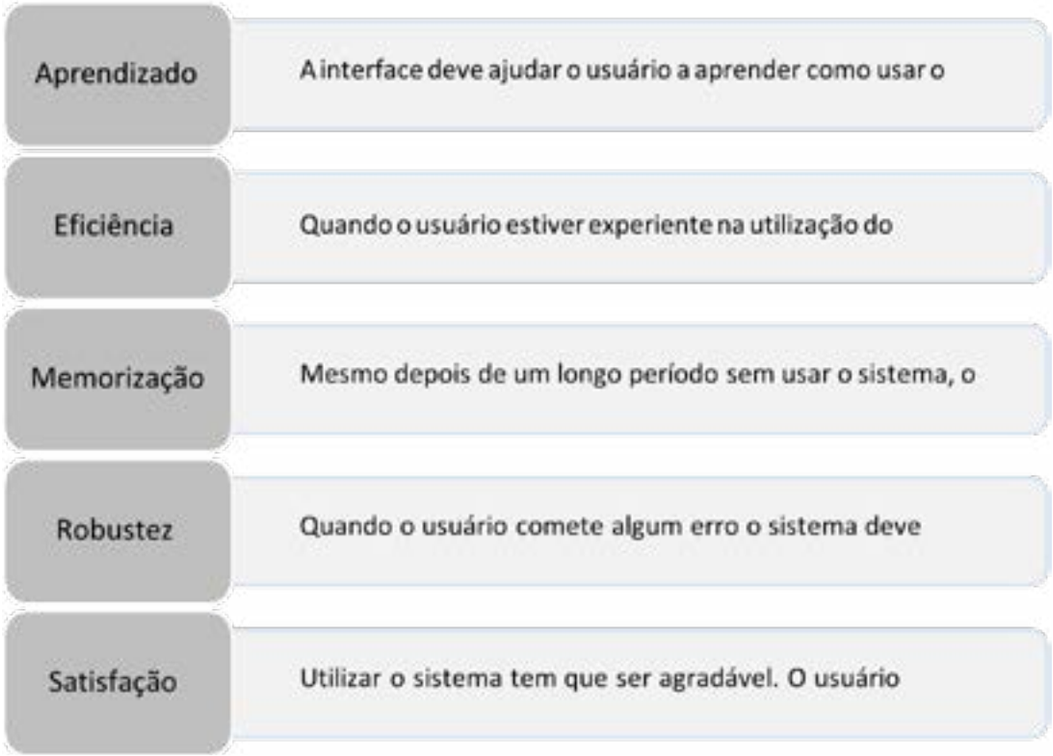
Usabilidade

Usabilidade define a facilidade com que as pessoas podem empregar uma ferramenta para realizar uma tarefa específica e importante. Dessa forma, a interface web deve ser construída para ajudar o usuário a realizar seu trabalho e chegar F informação importante de maneira efetiva. Conforme a ISO 9241-11: “Usabilidade é a medida pela qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com efetividade, eficiência e satisfação em um contexto de uso específico”.

Atributos de usabilidade

Para garantir alto grau de usabilidade, as interfaces do sistema devem contemplar os cinco atributos de usabilidade listados abaixo:

Figura 5 - Usabilidades.



Fonte: elaborado pelo autor

Exemplo: sistema bancário

Um sistema bancário é um exemplo de sistema de difícil aprendizado, mas muito eficiente. Um funcionário do banco deve ter vários treinamentos para entender os conceitos e as funcionalidades do sistema. Um usuário leigo certamente não conseguiria utilizar um software bancário sem treinamento.
Entretanto, quando o usuário está familiarizado com o sistema, ele consegue chegar a seu objetivo facilmente. Utilizando apenas o teclado, com poucos comandos o funcionário do banco consegue resolver o problema de um cliente. Apesar de ser de difícil aprendizado, é bastante eficiente.

Referências

ABRAHANSSON, P.; SALO, O.; RONKAINEN, J.; WARSTA, J. Agile Software Development Methods. Review and Analysis. Espoo 2002, v. VTT Publicação 478, 2002.

AGILE ALIANCE WEB SITE. Agile Alliance [On-line]. Disponível em: <http://agilealliance.org>. Acessado em: abr. 2023.

AGILE MODELING WEB SITE. Agile Documentation [On-line]. Disponível em: <http://aligemodeling.com/essays/agileDocumentation.htm>. Acessado em: abr. 2023.

AMBLER, S. A., Test-Driven Development. Agile Data, 2003. Disponível em: <http://agiledata.org>. Acessado em: abr. 2023.

BECK, K. TDD by Example. 1. ed. Addison-Wesley Professional, 2002.

BOOCH, G., RUMBAUGH, J., JACOBSON, I., The Unified Modeling Language User Guide Reading. Addison Wesley, 1999.

CHRISTEL, M. G.; KANG, K. C. Issues in Requirements Elicitation. Software Engineering Institute, Technical Report CMU/SEI-92-TR-012 ESC-TR-92-012, 2012. Disponível em: <http://www.sei.cmu.edu/pub/documents/92.reports/pdf/tr12.92.pdf>. Acessado em: abr. 2023.

DAHLSTEDT, A., Requirements Engerneering - Chapter 11. Department od Computer Science, University of Skovde, 2003. Disponível em: <htt://www.ida.his.se/ida/kurser/informationsystems_engineering/kursmaterial/foreelasningar/Chapter11_2003.pdf>. Acessado em: abr. 2023.

EVANS, E., Domain Driven Design. Addison-Wesley, 2004.

FAULK, S. R.; Software Requirements: A tutorial. In: Thayer R. G.; Dorfman, M.; Software Requirement Engineering, 2nd ed. Wiley-IEEE Computer Society Press, 1997.

FRANCE, R., KOBRYN, C.; UML for Software Engineer. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 23. 2001.

HOOKS, I. Writing Good Requirements. In: INTERNATIONAL SYMPOSIUM OF THE INCOSE, 3., 1993. Disponível em: <http;//www.incose.org/rwg/writinggoodrqs_hooks.htm>. Acessado em: abr. 2023.

JEFFRIES, R. et al. TDD: The art of fearless programming, 2001.

SAWYER, P.; KOTONYA, G.; Chapter 2: Software Requirements. In: Guide to the Software Engineering Body of Knowledge, SWEBOK. A Project of the Software Engineering Coordinating, 2001. Disponível em: <htt://www.swebok.org/stoneman/trial_1_00.htm>. Acessado em: abr. 2023.

SCHARER, L., Pinpointing Requeriments. In: SYSTEM AND SOFTWARE REQUIREMENTS ENGINEERING. IEEE Computer Society Press, 1990.

SCHWABER, K.; SUTHERLAND, J., The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game. 2017.

SOMMERVILE, I.; SAWYER, P. Requirements Engineering: A good practice guide. 1. ed. New York: John Wiley & Sons, 1997.

VERHEYEN, G. Scrum: Framework, não metodologia. 2013.

WAZLAWICK, R. S. Engenharia de Software: Conceitos e Práticas. 2013.

WILDT, Daniel et al. eXtreme Programming: práticas para o dia a dia no desenvolvimento ágil de software. 2015.

YOUNG, R. R., Recommended Requirements Gathering Practices. CROSSTALK. In: The Jornal od Defense Software Engineering. Disponível em: <http://www.stsc.hill.af.mil/crosstalk/2002/04/young.html>. Acessado em: abr. 2023.

Faculdade XPe

xpeducacao.com.br