

Cortesia de **IBM**

IBM edição limitada

Engenharia de Sistemas

PARA
LEIGOS®

Aprenda:

- O que é a engenharia de sistemas
- Como a engenharia de sistemas pode ajudá-lo a desenvolver produtos inteligentes, conectados
- Como acelerar a colocação do produto no mercado, garantir a agilidade dos negócios e oferecer produtos inteligentes de alta qualidade
- Como cortar os custos



Cathleen Shamieh

Engenharia de Sistemas

PARA

LEIGOS®

IBM EDIÇÃO LIMITADA

Cathleen Shamieh



WILEY

Wiley Publishing, Inc.

Engenharia de Sistemas Para Leigos® IBM Edição Limitada

Publicado por

Wiley Publishing, Inc.

111 River Street

Hoboken, NJ 07030-5774 EUA

www.wiley.com

Copyright © 2011 by Wiley Publishing, Inc., Indianapolis, Indiana EUA

Publicado por Wiley Publishing, Inc., Indianapolis, Indiana EUA

Nenhuma parte desta publicação pode ser reproduzida, armazenada em um sistema de recuperação ou transmitida de qualquer forma ou por qualquer meio eletrônico, mecânico, de fotocópia, gravação, digitalização ou outro, exceto se permitido sob as Seções 107 ou 108 do 1976 United States Copyright Act, sem a aprovação prévia por escrito da Editora. Solicitações de autorização para a Editora devem ser enviadas para Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, EUA, (201) 748-6011, fax (201) 748-6008, ou online para <http://www.wiley.com/go/permissions>.

Marcas comerciais: Wiley, o logotipo da Editora Wiley, Para Leigos, o logotipo do personagem da série A Reference for the Rest of Us!, The Dummies Way, Dummies.com, Making Everything Easier, e imagens comerciais relacionadas são marcas comerciais ou marcas registradas da John Wiley & Sons, Inc. e/ou suas afiliadas nos Estados Unidos e em outros países, e não podem ser usadas sem autorização por escrito. Todas as outras marcas comerciais são de propriedade de seus respectivos proprietários. A Wiley Publishing, Inc., não é associada a nenhum produto ou fornecedor mencionado neste livro.

LIMITE DE RESPONSABILIDADE/ISENÇÃO DE GARANTIA: A EDITORA E O AUTOR NÃO OFERECEM REPRESENTAÇÕES OU GARANTIAS DE QUALQUER NATUREZA COM RESPEITO À PRECISÃO OU INTEGRIDADE DO CONTEÚDO DESTE TRABALHO E ESPECIFICAMENTE SE ISENTAM DE TODA E QUALQUER GARANTIA, INCLUINDO, SEM LIMITAÇÃO, GARANTIA DE ADEQUAÇÃO A UMA DETERMINADA FINALIDADE. NENHUMA GARANTIA PODE SER CRIADA OU CONCEDIDA POR MATERIAIS DE VENDA OU PROMOCIONAIS. AS RECOMENDAÇÕES E ESTRATÉGIAS AQUI CONTIDAS PODEM NÃO SER ADEQUADAS A TODAS AS SITUAÇÕES. ESTE LIVRO É VENDIDO COM O ENTENDIMENTO DE QUE A EDITORA NÃO SE COMPROMETE A PRESTAR SERVIÇOS JURÍDICOS, CONTÁBEIS OU OUTROS SERVIÇOS PROFISSIONAIS. EM CASO DE NECESSIDADE DE ASSISTÊNCIA PROFISSIONAL, DEVE-SE PROCURAR OS SERVIÇOS DE UM PROFISSIONAL COMPETENTE. NEM A EDITORA NEM O AUTOR DEVEM SER RESPONSABILIZADOS POR PREJUÍZOS DECORRENTES DO USO DESTE MATERIAL. O FATO DE QUE UMA ORGANIZAÇÃO OU SITE SEJA MENCIONADO NESTE TRABALHO COMO CITAÇÃO E/OU FONTE PÔTENCIAL PARA INFORMAÇÕES ADICIONAIS NÃO SIGNIFICA QUE A EDITORA OU O AUTOR ENDOSSE AS INFORMAÇÕES QUE A ORGANIZAÇÃO OU SITE POSSA FORNECER OU AS RECOMENDAÇÕES QUE POSSA FAZER. ALÉM DISSO, OS LEITORES DEVEM LEVAR EM CONTA QUE OS SITES DA INTERNET APRESENTADOS NESTE LIVRO PODEM TER SIDO ALTERADOS OU DESATIVADOS APÓS A PUBLICAÇÃO DO PRESENTE MATERIAL.

Para informação geral sobre os nossos outros produtos e serviços, entre em contato com o nosso Business Development Department nos Estados Unidos pelo telefone 317-572-3205. Para detalhes sobre como criar um livro *Para Leigos* personalizado para a sua empresa ou organização, entre em contato com info@dummies.biz. Para informação sobre o licenciamento da marca *Para Leigos* para produtos serviços, entre em contato com BrandedRights&Licenses@Wiley.com.

ISBN: 978-1-118-37062-9

Fabricado nos Estados Unidos da América

10 9 8 7 6 5 4 3 2 1



Agradecimentos da Editora

Temos muito orgulho deste livro e das pessoas que trabalharam nele. Para detalhes sobre como criar um livro *Para Leigos* para a sua empresa ou organização, entre em contato com info@dummies.biz. Para detalhes sobre o licenciamento da marca *Para Leigos* para produtos serviços, entre em contato com BrandedRights&Licenses@Wiley.com.

Aqui estão algumas das pessoas que ajudaram a colocar este livro no mercado:

Compras, Edição e Desenvolvimento de mídia

Editora de projetos: Carrie A. Burchfield

Gerente de edição: Rev Mengle

Editora sênior de compras: Katie Feltman

Representante de desenvolvimento de negócios: Sue Blessing

Especialista de projetos de publicação personalizada: Michael Sullivan

Serviços de redação

Coordenadora sênior de projetos: Kristie Rees

Layout e artes gráficas: Melanee Habig

Revisora: Jessica Kramer

Publicação e edição, Série Para Leigos de tecnologia

Richard Swadley, Vice-Presidente e Editor executivo do grupo

Andy Cummings, Vice-Presidente e Editor

Mary Bednarek, Diretora Executiva, Compras

Mary C. Corder, Diretora de Edição

Publicação e edição, Série Para Leigos de consumidores

Diane Graves Steele, Vice-presidente e editora, Série Para Leigos de consumidores

Serviços de redação

Debbie Stailey, Diretora de serviços de redação

Desenvolvimento de negócios

Lisa Coleman, Diretora, Desenvolvimento de novos mercados e marcas

Visão geral do conteúdo

Introdução.....	1
Capítulo 1: Criação de produtos inteligentes.....	3
Capítulo 2: Domando a fera com a Engenharia de Sistemas.....	13
Capítulo 3: Requisitos revolucionadores	23
Capítulo 4: Abstração do sistema de modelagem.....	35
Capítulo 5: Garantia da qualidade de primeira	43
Capítulo 6: Capacitação da colaboração e gerenciamento das mudanças por equipes grandes	51
Capítulo 7: Dez maneiras de ganhar com a Engenharia de Sistemas	61

Introdução

produtos inteligentes são encontrados em qualquer lugar. Eles acompanham seus pacotes, controlam os sinais de trânsito, voam aeronaves e o orientam para o seu destino. Eles são o centro dos sistemas e serviços que você usa todos os dias — dos smartphones aos carros inteligentes, sistemas médicos, e sistemas aeroespaciais e de defesa.

Produtos inteligentes, instrumentados e interconectados revolucionam a forma como as pessoas interagem e executam suas tarefas diárias. Com uma combinação de eletrônicos, softwares, sensores e outros hardwares, temos a tecnologia necessária para criar produtos multifuncionais personalizados. E, com a nossa imaginação ilimitada, temos o potencial de definir centenas de sistemas e serviços inovadores personalizados voltados para o valor.

O grande desafio ao criar produtos inteligentes é a organização: como podemos integrar com eficácia uma combinação complexa de tecnologias para criar um “sistema de sistemas” inteligentes que cumpram o que prometem e atinjam o seu potencial? A solução está com a engenharia de sistemas.

Engenharia de Sistemas Para Leigos, IBM Edição Limitada, explica o que é a engenharia de sistemas e como ela pode ajudá-lo a controlar a complexidade natural do desenvolvimento de produtos inteligentes conectados. Se quiser acelerar a colocação do produto no mercado, garantir a agilidade dos negócios e oferecer produtos inteligentes de alta qualidade e, ao mesmo tempo, reduzir os custos, *Engenharia de Sistemas Para Leigos*, IBM Edição Limitada, é o livro para você.

Como este livro está organizado

Os sete capítulos deste livro têm por objetivo ajudá-lo a entender o problema que a engenharia de sistemas tenta resolver e todas as etapas para resolvê-lo. Eis aqui um resumo do seu conteúdo:

- ✓ Capítulo 1 define os produtos inteligentes e explica porque eles justificam uma nova abordagem ao desenvolvimento de sistemas.
- ✓ Capítulo 2 é um resumo de alto nível da engenharia de sistemas.
- ✓ Capítulo 3 expõe o papel fundamental que os requisitos desempenham no ciclo do desenvolvimento de sistemas.
- ✓ Capítulo 4 mostra como os modelos podem aprimorar o seu entendimento da estrutura e do comportamento do sistema.
- ✓ Capítulo 5 explica como verificar se você construiu o sistema certo (validação) e construiu o sistema corretamente (verificação).
- ✓ Capítulo 6 sugere formas de aprimorar a colaboração dos times de desenvolvimento.
- ✓ Capítulo 7 fornece exemplos de algumas empresas reais que incorporaram a engenharia de sistemas nas suas principais práticas de negócios.

Ícones usados neste livro

Ao ler este livro, você verá vários ícones chamativos criados para destacar certa informação.



Este ícone destaca os conceitos principais que você deveria memorizar.

Este ícone é exibido próximo às sugestões viáveis que têm por objetivo facilitar, e muito, a sua vida.

Estes são o oposto de uma dica. São sugestões que, quando ignoradas, certamente irão dificultar, e muito, a sua vida.

Sei que você não precisa saber de tudo o que sei, mas este ícone dá um pouco mais de detalhe sobre o que é absolutamente necessário, por isso você pode pular se quiser.

Capítulo 1

Criação de produtos inteligentes

Neste capítulo

- ▶ Como lidar com a demanda de sistemas inteligentes interconectados
 - ▶ Reconhecimento dos desafios na estrada para o sucesso
 - ▶ Mudança de marcha para abranger um cenário mais amplo
-

pense nisso: Ao dar ré para sair de casa, o seu carro envia um sinal para a sua casa para armar o sistema de alarme e fechar a porta da garagem. O seu celular se sincroniza automaticamente com o sistema de comando de voz do seu carro e o sistema de posicionamento global (GPS) integrado analisa a situação do tráfego ao vivo e sugere uma rota alternativa que gaste menos tempo para ir para o trabalho. O seu carro avisa que está na hora de trocar o óleo, confere a sua agenda no seu smartphone, sugere horários possíveis para compromissos, e oferece para ligar para o seu posto de gasolina favorito. O carro até informa sobre a condição potencial de enchentes na sua volta do trabalho.

É possível que um veículo que já foi conhecido como uma “carruagem sem cavalo” possa ser tão inteligente e útil? Claro que sim!

Neste capítulo, você irá descobrir qual é a força motriz dos produtos inteligentes, como eles trabalham em conjunto, e porque você deveria adotar novos processos de negócios no seu desenvolvimento.

O que os produtos inteligentes têm de tão inteligentes?

Os produtos inteligentes são a grande febre do momento. É difícil imaginar a vida antes dos aparelhos eletrodomésticos de cozinha programáveis, videogames interativos e celulares multitarefa que gravam vídeo, imprimem fotos, surfam a Internet e tocam música. Uma aeronave pode achar o seu caminho evitando colisões e contamos com drones inteligentes e outros sistemas de defesa de precisão para manter a nossa segurança.

E por que estes e outros objetos inanimados podem realizar feitos tão incríveis?

Entrega de valor com os sistemas inteligentes

Sistemas inteligentes movidos por software estão surgindo em todo o ecossistema:

- ✓ **Saúde:** Software personalizado oferece uma acesso confiável e seguro às imagens médicas complexas e relatórios através de dispositivos móveis, viabilizando que os profissionais médicos revejam a informação do paciente fora do consultório e acelerem os diagnósticos de emergência.
- ✓ **Serviços públicos:** A comunicação bidirecional entre o fornecedor de energia e os consumidores através de uma “grade inteligente” facilita o controle inteligente do consumo de energia. Por exemplo, as lavadoras de roupas podem ser ligadas pela grade quando a energia é mais barata, e certos aparelhos eletrodomésticos podem ser desligados durante os horários de pico de consumo. Os carros inteligentes são outro exemplo nesta área.

✓ **Aparelhos inteligentes:** Os aparelhos conectados da casa fornecem status atualizado do consumo de energia com interfaces intuitivas de usuário. O controle remoto destes aparelhos viabiliza que os consumidores atinjam o nível de conforto desejado e reduzam o consumo de energia.

✓ **Entretenimento:** As TVs inteligentes fornecem acesso sem fio à Internet, permitindo que os consumidores aluguem vídeos, façam compras, consultem a previsão do tempo, estabeleçam novas páginas personalizadas e façam download de aplicativos.

✓ **Automotivo:** Os sistemas anticollision inteligentes integram uma tecnologia GPS diferencial, comunicação sem fio, e displays gráficos a bordo para alertar o motorista sobre veículos próximos e até mesmo fazer uma manobra de emergência.

Mistura de ingredientes tecnológicos

Atualmente os produtos e serviços inteligentes são o resultado da convergência das tecnologias de manufatura, eletrônica e da informação. Os fabricantes de pensamento rápido perceberam que poderiam aproveitar os tremendos avanços dos microeletrônicos, software, dispositivos mecânicos, sensores e atuadores para criar produtos para impressionar os clientes — e engolir os concorrentes. Por isso eles pegam um pouquinho disso, um pouquinho daquilo, misturam (com uma certa ajuda dos amigos engenheiros), e — *ta-rá* — inventam produtos que até mesmo George Jetson acharia inovador!



Os produtos inteligentes são de todos os formatos e tamanhos, mas em geral, podem ser caracterizados com três adjetivos:

- ✓ **Instrumentados:** Dispositivos esportivos de produtos inteligentes, como câmeras, detectores de movimento, sensores de posição, receptores sem fio, som, calor e umidade leve, e campos magnéticos que monitoram constantemente as operações e farejam a vizinhança. Com o estabelecimento deste tipo de contexto, os produtos inteligentes podem se adaptar ao ambiente em tempo real.
- ✓ **Interconectados:** Quando dois ou mais produtos interagem entre si e compartilham informação, eles podem oferecer valor que vai além da capacidade individual de cada produto. Conecte-os na Internet e no sistema de apoio ou outros sistemas de IT, e o céu é o limite!
- ✓ **Inteligentes:** Com o uso de dados sensoriais, tendências históricas e informação do perfil do usuário, os produtos bem projetados podem fazer previsões, otimizar os resultados e personalizar a experiência do usuário.

Juntando tudo isso com software

Não há dúvida que o condutor mais importante da revolução do produto inteligente é o crescimento fenomenal da capacidade de processamento. Como o cérebro por trás de cada produto inteligente, os microprocessadores integrados executam algoritmos, analisam dados, fazem cálculos numéricos complicados, e controlam todos os componentes mecânicos e eletrônicos de um produto inteligente.

Para fazer o trabalho, os microprocessadores executam milhares — às vezes milhões — de linhas de códigos de software. Por exemplo, hoje em dia, os carros topo de linha contêm dezenas de microprocessadores que executam provavelmente 100 milhões de linhas de códigos — com o objetivo expresso de oferecer 250 a 300 funções para os motoristas e passageiros.

Com os fabricantes de semicondutores criando microprocessadores cada vez mais potentes, a oportunidade para os desenvolvedores de software criarem novas funções geniais está subindo como um foguete. E isto é uma boa coisa — porque muitos hardwares que diferenciam os produtos estão se transformando em commodities. Cada vez mais é o software, e não os eletrônicos, que destaca um produto e determina quais produtos irão conquistar a fatia do mercado.



A flexibilidade inerente do software oferece inúmeras oportunidades para o desenvolvimento de recursos e funções adicionais, permitindo que os fabricantes atualizem seus produtos de acordo com as expectativas de novidades por parte dos clientes. Por exemplo, os MP3 mais populares não são simples tocadores, eles armazenam bibliotecas de música, transmitem vídeo, dão suporte a mensagens, fornecem games e executam aplicativos de terceiros. Os melhores produtos podem ser facilmente atualizados para incluir funções para que acompanhem as mudanças do mercado.

Tornando-se viral com padrões abertos



Em 2013, um número incrível de 1,2 bilhões de dispositivos eletrônicos de consumo conectados estarão em 800 milhões de casas com acesso de banda larga. Se você tem algumas boas ideias sobre como oferecer serviços de valor agregado pela Internet, deve estar salivando com estes números.

Imagine se todos os dispositivos eletrônicos fossem criados no vácuo, com cada fabricante desenvolvendo a sua própria forma de se conectar com a Internet e se comunicar com o mundo exterior. Poderia haver muito bate-papo nas linhas, mas nenhuma delas seria útil. Que perda de oportunidade!

Não se preocupe! Algumas pessoas realmente inteligentes com grandes visões do futuro tiveram a ideia de criar *padrões abertos*, ou concordaram em maneiras para enviar mensagens dos dispositivos para provedores de serviço. Quanto maior o número de empresas que usam os protocolos de comunicação definidos pelos padrões abertos, maiores as oportunidades para

todos nós. A população está evoluindo rapidamente para uma “Internet de coisas” — um ecossistema global de produtos e serviços inteligentes e conectados.

Atenção aos clientes

Grande parte da população mundial consiste em usuários experientes de produtos inteligentes.



Somente em 2010, cerca de 40 milhões de dispositivos de navegação pessoal foram vendidos no mundo e, em 2008, mais de 55 milhões de pessoas arrebataram iPods.

Se você viaja frequentemente, nem se preocupa com o tempo ruim durante um voo pelo país porque sabe que o avião tem um equipamento inteligente que garante a segurança da sua viagem. Os usuários experientes conhecem muito bem a capacidade dos produtos inteligentes de hoje em dia — e estão exigindo produtos mais inteligentes ainda.

Os consumidores atualmente exigem produtos confiáveis, interativos em tempo real — e querem este produto agora! E se você tiver sorte (e for inteligente) o suficiente para fazer a entrega na hora, não deve relaxar porque seus clientes já esperam um upgrade!



Com a incorporação da capacidade de personalização e de integração nos designs, é possível criar produtos voltados para as necessidades individuais do usuário e adaptá-los ao ambiente do usuário. Os clientes querem produtos que prometem facilitar e tornar suas vidas mais divertidas, no entanto, cada um deles tem uma forma única de fazer as coisas, com um conjunto de valores distintos, implicâncias e idiossincrasias. Os clientes querem produtos inteligentes tão fáceis de personalizar quanto baixar o aplicativo mais recente de smartphone!

Produtos Inteligentes não são ilhas

Você pode perceber que não é mais suficiente criar um produto independente, legal, cheio de recursos. Os produtos inteligentes de hoje devem ser inteligentes o suficiente para interagir com outros produtos inteligentes.

Vejamos um carro moderno, por exemplo (veja Figura 1-1). Um carro topo de linha típico contém um conjunto de subsistemas de software sofisticados criados para tornar a experiência divertida, melhorar a segurança e otimizar a eficiácia do combustível: freios antitravamento, sistema anticolidão, controle de conforto, segurança e muito mais. O projeto e criação de um carro inteligente — que na realidade é um “sistema de sistemas” — envolve a integração complexa de componentes mecânicos, elétricos e eletrônicos, sem mencionar a execução adequada de, aproximadamente, 100 milhões de linhas de códigos.

Para complicar ainda mais as coisas, os carros inteligentes interagem com diversos outros sistemas externos ao próprio carro. Os serviços baseados em locais, sistemas de diagnóstico de veículos e sistemas de recarga de carro híbrido/elétrico são apenas alguns dos muitos sistemas com os quais um carro pode interagir em um ecossistema automotivo maior.

Parece complicado, não é? Mas é mesmo! Mas também é incrivelmente útil. Por exemplo, se o sistema de segurança do seu carro é projetado para interagir com centros de resposta de emergência, ele pode fornecer detalhes do acidente para os socorristas baseado na coleta de dados dos sensores do carro. Os detalhes essenciais, como a força do impacto, por exemplo, podem auxiliar o operador do serviço de emergência a determinar quais recursos de salvamento mais apropriados devem ser enviados.

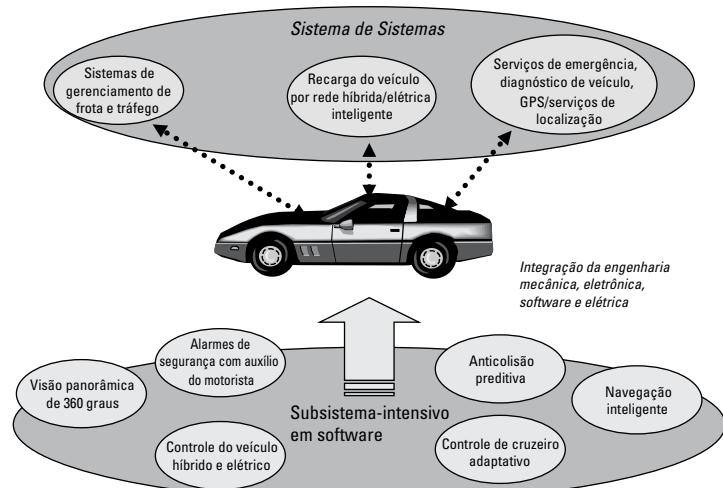


Figura 1-1: Um carro inteligente é um sistema de sistemas de um ecossistema maior.



Normalmente, os recursos mais valiosos de um produto inteligente não estão totalmente dentro do próprio produto mas são fornecidos com a interação com outros produtos ou serviços do ecossistema. Da mesma forma como as pessoas, os produtos inteligentes precisam colaborar e compartilhar a informação!



Poder compartilhar os dados não quer dizer que você *deveria!* A privacidade, a segurança e os regulamentos (entre outras coisas) podem ter um grande impacto no seu design!

Identificação de desafios novos e empolgantes

O desenvolvimento de um produto inteligente de sucesso que ofereça uma experiência personalizada para diversos tipos de clientes inconstantes e exigentes que queiram tudo para ontem é, como dizer, um trabalho nada fácil. Você pode ser um expert de renome na sua área, tentando desesperadamente aceitar o fato de que nem a sua educação de elite nem a sua vasta experiência o preparou para os enormes desafios que você enfrenta atualmente.



Antes de escalar a montanha, para desenvolver produtos inteligentes de sucesso, você precisa abordar as seguintes questões específicas:

- ✓ **Domínio de múltiplas capacidades:** Você precisa ser especialista em diversas áreas técnicas, incluindo manufatura, eletrônicos, engenharia mecânica e engenharia de software. Embora a maioria das empresas seja forte em uma ou duas destas áreas, esta especialização é raramente encontrada sob um mesmo teto.
- ✓ **Ser uma casa de software de categoria internacional:** Se você é um fabricante de produtos que considera o software como um mal necessário, é melhor procurar um hipnotizador porque a maioria dos “inteligentes” dos produtos inteligentes são do software — que ainda não se codifica a si mesmo (pelo menos, ainda não).
- ✓ **Integração do trabalho de desenvolvimento de hardware e de software:** Com as funções voltadas para o software tomando o centro do palco, você precisa fazer com que seus times de hardware e de software *realmente* trabalhem juntos — e não apenas joguem os módulos acabados para integração e teste.
- ✓ **Gerenciamento eficaz de times distribuídos:** Se os seus times de desenvolvimento estão localizados em cidades, fusos horários e/ou

países diferentes, tente facilitar os trabalhos em colaboração para garantir resultados eficientes, precisos e cooperativos.

- ✓ **Interação com outros sistemas:** Você não deve lançar o único produto do mercado que não consiga interagir com a Internet, sistemas de apoio de IT e outros sistemas interconectados. Os produtos independentes não são tão inteligentes assim — e muito provavelmente serão fracassos independentes.
- ✓ **Conformidade garantida:** Mesmo que o seu produto não esteja ligado diretamente aos padrões regulatórios ou da indústria, ele pode interagir com um produto ou serviço que esteja, por isso previna-se — para que todos os outros garotos queiram jogar com você.
- ✓ **Mudança das prioridades do design:** Priorize o seu design para capacidade de atualização e não estabilidade do produto. Concentre-se nas prioridades de design *certas*. Para a área aeroespacial pode ser a segurança, para a defesa pode ser a proteção, para os eletrônicos de consumo pode ser a capacidade de atualização.
- ✓ **Redução do ciclo de vida do produto:** Com a demanda de novos recursos causando a redução da vida do serviço até mesmo dos melhores produtos, o seu trabalho é entrar no mercado na hora certa com o conjunto certo de recursos.
- ✓ **Adaptação aos requisitos sempre diferentes:** A mudança das necessidades do cliente, as condições dinâmicas dos concorrentes e do mercado, e as metas corporativas repensadas são motivos válidos para os pedidos irritantes de mudança. Se você for inteligente, descubra como lucrar com a mudança.

Pesquisa dos perigos potenciais

A falha de abordar os desafios relacionados com a criação de um novo produto inteligente que seja uma bobagem pode deixá-lo em maus lençóis — isso sem falar no prejuízo para a sua imagem e para os lucros.

Os erros que possam parecer pequenos e fáceis de corrigir em um produto independente normalmente são aumentados e distribuídos através de um design de sistema interconectado. Um bug de software pode criar um caos quando não detectado no início do processo de desenvolvimento, aumentando os custos e causando atrasos na programação. Com a quantidade de softwares nos dispositivos dobrando a cada dois anos, é fácil entender porque 66% dos designs de software dos dispositivos ultrapassam o orçamento, e porque 24% dos grandes projetos são cancelados devido a atrasos irrecuperáveis da programação.



Se você não puder desenvolver produtos complexos em um ciclo mais curto sem comprometer a qualidade, você está fadado a perder a renda e manchar a sua marca. No entanto, os produtos inteligentes e interconectados normalmente têm centenas — ou até milhares — de requisitos exclusivos, sendo difícil imaginar ser possível manter a qualidade e reduzir o ciclo de desenvolvimento.

Se você não estiver preparado para responder facilmente às novas demandas do mercado ou ameaças dos concorrentes, não se espante ao perder a fatia do mercado para organizações mais ligeiras. Para muitas organizações, somente chegar ao design inicial certo já é um desafio: quase um terço de todos os dispositivos produzidos hoje em dia simplesmente falham na conformidade com as especificações de performance ou funcionais. Você pode apostar que estes dispositivos estarão fora das prateleiras muito rapidamente!



Não é porque você tem um talento técnico especial trabalhando no design do sistema que o sucesso é garantido. Muitos sistemas falham devido a erros nas especificações da interface do sub-sistema, fidelidade com os requisitos ou comunicações do conhecimento principal — e não de engenharia.

Reconhecimento da necessidade da mudança do paradigma

Depois de aceitar o fato de que para ter sucesso no mercado atual você precisa mudar como o valor é entregue com os seus produtos, você pode iniciar a examinar novamente o planejamento, gerenciamento e processos de desenvolvimento do seu produto. E imediatamente você verá a necessidade de mudar para se concentrar na inovação e na mudança — com o software como a base para este diferenciador.

Antigamente, quando o hardware mandava, o Design auxiliado por computador 3D (CAD) e o gerenciamento da Lista técnica de materiais (BOM) eram tudo para o desenvolvimento de produto sequencial (veja Figura 1-2). O seu time de desenvolvimento de hardware usou um sistema CAD para desenhar o hardware para atender um conjunto de requisitos, enquanto que o time de software trabalhou com um conjunto de requisitos diferentes, porém relacionadas, para produzir o código necessário. O design CAD e o BOM foram entregues para a fábrica que calculou a forma mais rápida e mais barata de manufaturar o produto. Finalmente, os engenheiros de integração e de teste (de outro departamento) carregaram o software e fizeram o teste geral do produto.

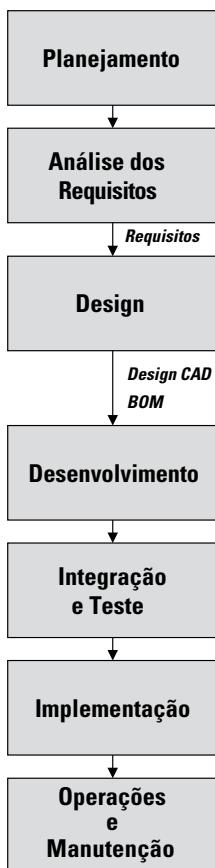


Figura 1-2: Desenvolvimento de produto sequencial.

No mundo de hoje, este tipo de processo de desenvolvimento sequencial baseado em documentos simplesmente não dá mais certo. Imagine tentar responder rapidamente aos eventos inesperados do mercado ou aos pedidos de novos recursos. Cada mudança exige voltar para o início e refazer todo o processo sequencial. Sua empresa iria definharia num piscar de olhos!

Se a sua empresa depende de um sistema inteligente para sobreviver, é preciso deixar de lado os processos antigos sequenciais baseados em documento e desenvolver todo um novo conjunto de processos centrais para o futuro.



Capítulo 2

Domando a fera com a Engenharia de Sistemas

Neste capítulo

- Avaliação da engenharia de sistemas
- Ajuste da euforia do projeto com a realidade do mundo
- Modelagem do seu design

V

ocê já pensou como a NASA conseguiu administrar com sucesso o desenvolvimento de um sistema tão complexo como a da aeronave espacial Apollo? Como times tão diferentes de engenheiros de projeto, programadores, fabricantes terceiros e outras pessoas conseguiram trabalhar juntos no primeiro voo tripulado para a lua? Bom, na realidade isto não poderia ter sido realizado sem a ajuda da engenharia de sistemas.

A *engenharia de sistemas* é uma abordagem interdisciplinar da criação de sistemas maiores e complexos que atendam a um conjunto definido de requisitos de negócios e técnicas. As indústrias aeroespacial e de defesa usam a engenharia de sistemas há muito tempo e muito do que elas aprenderam está sendo aplicado em outras indústrias. Com os sistemas de transporte, redes elétricas, e sistemas de telefonia e de rede tornando-se cada vez mais inteligentes, é preciso métodos dos voos espaciais para construí-los.



A engenharia de sistemas existe desde os anos 1940s, mas começou a ganhar força fora da NASA nos anos 1990s. Foi nessa época que os fabricantes começaram a transformar os produtos ordinários em sistemas inteligentes com a incorporação da tecnologia da informação — passando a ser um ponto crucial no desenvolvimento de produtos. Somente recentemente o software passou a assumir um papel central.

Com a capacitação de funções sem limites nos produtos, o software passou a ter um papel central em todos os tipos de produtos, viabilizando muitos novos tipos de interconexões das peças e do produto com o mundo. O maior número de conexões aumentou exponencialmente a complexidade do sistema — deixando as empresas sem alternativa, tendo que eliminar os silos de desenvolvimento e criar novas maneiras de gerenciar a complexidade.

Neste capítulo, você saberá exatamente o que é a engenharia de sistemas, como ela pode ajudá-lo a gerenciar a complexidade e a desenvolver produtos mais inteligentes — e inovar até chegar no topo.

Engenharia de Sistemas

Se você perguntar a cinco especialistas exatamente o que é a engenharia de sistemas, vai receber provavelmente cinco — talvez até seis — respostas diferentes! O motivo é que o termo *engenharia de sistemas* é usado para várias coisas diferentes e, em parte, porque o conceito de sistemas tem se modificado há décadas, e por isso a *engenharia de sistemas* não teve escolha, e se modificou também!

Não se preocupe! Enquanto alguns doutores possam se preocupar com os detalhes e o escopo da engenharia de sistemas, a maioria dos especialistas concorda com a sua fundação. Nesta seção, veremos o que é esta fundação — e como construí-la.

Enxergar a floresta e as árvores



A engenharia de sistemas é uma prática e um processo ao mesmo tempo (veja Figura 2-1):

- ✓ Como prática, se preocupa com cenário geral: como um sistema funciona e se comporta em geral, como ele faz a interface com os seus usuários e outros sistemas, como seus subsistemas interagem e como reunir várias disciplinas de engenharia para um trabalho conjunto.
- ✓ Como processo, define uma abordagem estruturada sólida para o desenvolvimento do sistema que possa ser aplicado ao sistema de sistemas ou dentro das disciplinas específicas da engenharia.

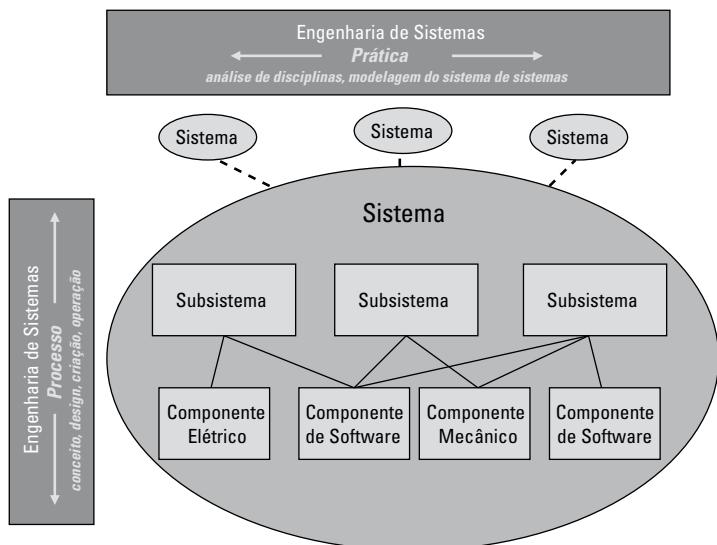


Figura 2-1: A engenharia de sistemas é uma prática e um processo ao mesmo tempo.

Não importa o tipo de análise, a engenharia de sistemas é a aplicação da disciplina ao processo de desenvolvimento do sistema. E esta disciplina tem dois sabores distintos:

- ✓ **Disciplina técnica** garante a execução rigorosa de um processo de desenvolvimento sensato, desde o conceito até a produção e a operação.
- ✓ **Disciplina do gerenciamento** organiza o trabalho técnico em todo o ciclo de vida do sistema, incluindo a facilitação da colaboração, definição do fluxo de trabalho e implementação das ferramentas de desenvolvimento.

Alguns princípios de orientação a seguir

Com a união da amplitude com a profundidade, a engenharia de sistemas tem por objetivo ajudá-lo a administrar os detalhes de um trabalho complexo de desenvolvimento sem esquecer das metas gerais do projeto.



Comece com a adesão ao seguinte conjunto de princípios de orientação:

- ✓ **Fique focado no prêmio.** Defina o resultado desejado para um projeto desde o início e não se afaste da sua meta, mesmo que a coisa fique louca.
- ✓ **Envolva pessoas interessadas.** Peça a opinião dos clientes, usuários, operadores, gerentes de nível C e outras pessoas ao tomar as decisões nas diferentes etapas do processo de desenvolvimento.
- ✓ **Defina o problema antes de encontrar a solução.** Mantenha a mente aberta quanto aos meios para os fins, para examinar várias alternativas — e escolha a melhor solução para o resultado desejado.
- ✓ **Divida o problema em partes administráveis.** Decomponha o sistema em subsistemas menores, e divida cada subsistema nos componentes de hardware ou software. Outro princípio é o gerenciamento das interfaces destes pedaços para a garantia da integração e entrega da capacidade emergente exigida.
- ✓ **Espera para escolher a tecnologia específica.** Espere até estar bem adiantado no processo antes de selecionar os componentes físicos para evitar o compromisso com uma tecnologia desatualizada ou desnecessária quando estiver pronto para implementar o design.
- ✓ **Conekte os requisitos com o design.** Confirme se você pode justificar as decisões de design conectando-as com as necessidades técnicas e de negócios específicas.
- ✓ **Teste logo, teste frequentemente.** Aproveite os protótipos, simuladores, emuladores e qualquer outro método para que todos envolvidos no projeto vejam logo o sistema. Confira se os testes provam o atendimento dos requisitos com a sua conexão.

Exploração do processo da engenharia de sistemas

Muito bem, você tem fortes princípios orientadores, mas como colocá-los em ação? Bem, uma maneira de fazer isso é desenvolver um processo consistente para a engenharia de sistemas que abranja estes princípios.

Aproximadamente nos últimos 20 anos, os especialistas de sistemas complexos desenvolveram e refinaram o conhecido *Modelo V* do processo de engenharia de sistemas (veja Figura 2-2). O *Modelo V* é uma representação gráfica de uma série de etapas e procedimentos para o desenvolvimento de sistemas complexos.

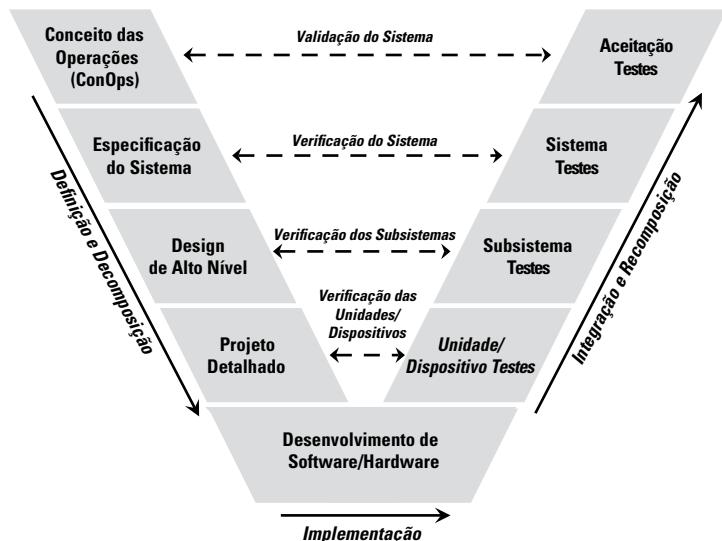


Figura 2-2: O Modelo V do processo de engenharia de sistemas.

Ao traçar o “V” da esquerda para a direita, você executa o processo de engenharia de sistemas em uma série de etapas, assim:

- ✓ **Conceito de Operações (ConOps):** Identificação e documentação das necessidades dos principais interessados, capacidade geral do sistema, funções e responsabilidades e medidas da performance para a validação do sistema no final do projeto.
- ✓ **Especificação do Sistema:** Detalhe dos requisitos verificáveis do sistema que atendam às necessidades dos interessados definidas no ConOps.
- ✓ **Design de Alto Nível:** Design de uma arquitetura de sistema de alto nível que atenda os requisitos do sistema e preveja a manutenção, atualizações e integração com outros sistemas.

- ✓ **Projeto Detalhado:** Exame do design do sistema, desenvolvimento dos requisitos de componentes que dão suporte à compra de hardware dentro do orçamento.
- ✓ **Desenvolvimento de Software/Hardware:** Seleção e procura da tecnologia apropriada, e desenvolvimento do hardware e software para o cumprimento das especificações detalhadas do projeto.
- ✓ **Teste de Unidade/Dispositivo:** Teste de cada componente da implementação de hardware, verificação da funcionalidade de acordo com os requisitos dos componentes.
- ✓ **Teste do Subsistema:** Integração dos componentes de hardware e de software em subsistemas. Teste e verificação de cada subsistema de acordo com os requisitos de alto nível.
- ✓ **Teste do Sistema:** Integração dos subsistemas e teste de todo o sistema de acordo com os requisitos do sistema. Confirmação de que todas as interfaces foram implementadas adequadamente e de que todos os requisitos e limites foram atendidos.
- ✓ **Teste de Aceitação:** Confirmação de que o sistema cumpre com os requisitos e atinge as metas desejadas.

Durante o processo de engenharia de sistemas, criação e refinamento da documentação do sistema. Em cada etapa à esquerda do “V” na Figura 2-2, crie os requisitos para a próxima etapa, bem como um plano para a verificação da implementação do atual nível de decomposição. Por exemplo, durante a etapa de ConOps, você cria um documento com os requisitos de alto nível do sistema que conduz à etapa de Especificação do Sistema, e cria o Plano de Validação do Sistema que conduz ao Teste de Aceitação. Em cada etapa à direita do “V”, você cria a documentação de suporte ao treinamento, uso, manutenção, instalação e teste do sistema.



Com a conexão de todas as etapas à esquerda do “V” de todos os requisitos e referindo-se a estes requisitos durante o trabalho em direção à direita do “V,” você estará muito mais propenso a permanecer fiel à sua missão original e manter a objetividade durante o processo. Estas conexões são conhecidas na engenharia de sistemas como *rastreamento*. O Capítulo 3 abrange os requisitos e o rastreamento com detalhes.

Gerenciamento da complexidade com modelos

Alguns modelos são úteis, principalmente para um projeto de engenharia complexo. Se você puder desenvolver métodos relativamente baratos para o design, teste e verificação do sistema *antes* de produzi-lo, é possível economizar bastante tempo e dinheiro — e até mesmo o seu trabalho! Uma maneira de fazer isso é usar modelos para fazer o design e refiná-lo durante o processo de desenvolvimento.

Com os modelos do sistema é possível capturar a complexidade em diversos níveis, incluindo o sistema de sistemas (também conhecido como ecossistema), sistema, subsistema e níveis de componentes. Com eles é possível explorar os detalhes de cada um destes níveis, conhecidos como *níveis de abstração*, independentemente e ocultar ou expor os detalhes de acordo.

Os modelos podem ter diferentes formatos. No nível mais simples, pode ser apenas uma planilha usada para calcular alguma propriedade empírica do sistema. Por outro lado, pode ser uma simulação de computador interativa altamente complexa.

Por exemplo, para entender como o sistema automotivo (tradução: carro) que você está desenvolvendo capta e encaminha os dados do impacto em uma colisão para um sistema de resposta de emergência. É preciso explorar a informação sobre os sensores do carro e como o carro interage com o sistema externo, mas você provavelmente não deve se distrair com detalhes externos, como o diagrama de fiação e colocação de componentes. O modelo certo pode mostrar o que é necessário sem detalhes extra.

Os modelos podem proporcionar os seguintes benefícios:

- ✓ Permitir a concentração na informação relevante para a presente questão, e garantir a consistência do seu design.
- ✓ Captura da estrutura (arquitetura) e comportamento (funcionalidade) de um sistema, ilustrando o relacionamento e a interação dos elementos do sistema.



- ✓ É possível usar os modelos na previsão do comportamento em diversos níveis de abstração, permitindo a exploração de diferentes arquiteturas no início do processo de desenvolvimento e a execução de estudos de compensação para avaliar as opções de design que façam mais sentido.
- ✓ É possível desenvolver modelos executáveis para a validação do design de acordo com os requisitos antes da construção do sistema.
- ✓ Os modelos permitem a visualização dos detalhes de um sistema com diversas perspectivas (por exemplo, arquitetura, lógica, função, dados físicos, dados, e usuário), para a abordagem de questões específicas.
- ✓ É possível explorar modelos que ajudem a administração do trabalho de desenvolvimento.
- ✓ A modelagem oferece uma tremenda visibilidade do sistema, proporcionando um ponto focal potente para a discussão e entendimento mútuo.
- ✓ Os modelos oferecem um espaço para os pensamentos e critérios de decisões, facilitando a colaboração entre você e seu time de desenvolvimento.
- ✓ E, talvez o mais importante, o modelo do sistema fornece um ponto de sincronização de diversas disciplinas de engenharia, e uma solução para um dos problemas mais significantes para o desenvolvimento de sistemas inteligentes: como coordenar o hardware e o software.

Como falar o mesmo idioma

Da mesma forma como um arquiteto usa um conjunto de padrões para representar elementos de um prédio em um desenho a ser interpretado por um mestre de obras, um time de desenvolvimento deve usar uma linguagem comum para representar os modelos de sistema que promovam a compreensão geral.

Em 2001, o International Council on Systems Engineering (INCOSE), juntamente com o Object Management Group (OMG), iniciaram o desenvolvimento de uma linguagem de modelagem comum para as aplicações de engenharia de sistemas e, em poucos anos, a Linguagem de Modelagem de Sistemas (SysML) nasceu. Criada como uma adaptação da Linguagem de Modelagem Unificada (UML) concentrada no software, a SysML é o verdadeiro idioma padrão para sistemas de modelagem e sistemas de sistemas.

A SysML usa uma abordagem de diagrama simples para modelar os sistemas (tão simples que algumas pessoas os chamam de pintura de caverna), onde a unidade básica da estrutura — um bloco — pode ser usada para representar o hardware, software, instalações, pessoal ou qualquer outro elemento de um sistema. Com uma série de *diagramas de estrutura* em ninho, define-se a estrutura interna e o uso desejado de um elemento particular do sistema (por exemplo, um sistema de freio antitravamento). Então, com uma série separada de *diagramas de comportamento* em ninho, é possível mostrar como aquele elemento do sistema interage com outros elementos, e com os *atores* (usuários, sistemas externos ou o ambiente), para cumprir ou *realizar* o comportamento.

Além da modelagem da estrutura (arquitetura) e do comportamento dinâmico (funcionalidade) do sistema, a SysML também permite a modelagem dos requisitos e dos parâmetros da performance. Por exemplo, em um sistema automotivo, é possível criar um diagrama de requisitos para especificar um limite, como “parar a 105 kph em uma distância de 55 m em uma superfície limpa e seca” e diagramas paramétricos para especificar as equações que controlam o movimento do carro. E o melhor de tudo, é possível usar as novas ferramentas de software potentes — pense como se fosse a construção de um simulador de um novo carro de corrida. Isto significa que você pode fazer um test-drive para verificar a direção antes mesmo de construí-lo!



Com a definição e a organização de modelos de construção, a SysML força os engenheiros e arquitetos de sistemas a ser bastante claros e precisos ao projetar o sistema. Isto reduz a ambiguidade e leva a uma maior qualidade, ciclos de desenvolvimento mais curtos e custos reduzidos.

Uma boa imagem vale mil (ou um bilhão) de palavras

A melhor maneira de entender a modelagem com a SysML é olhar o diagrama da SysML. A Figura 2-3 mostra um *diagrama do contexto* simples de um carro moderno que tem um receptor integrado do Sistema de Posicionamento Global (GPS) e um controle remoto do sistema de controle de segurança residencial.

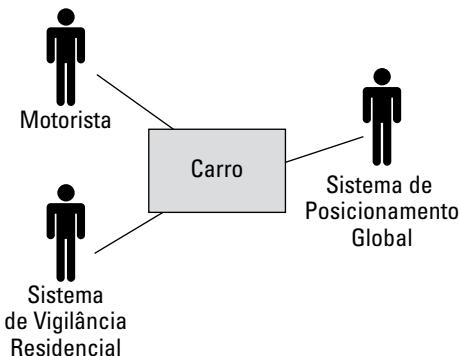


Figura 2-3: Um simples diagrama do contexto de um carro.

Use o diagrama do contexto para definir os limites, ou contexto, do sistema. Neste caso, o sistema é o carro, que interage com três *atores* externos do sistema: o motorista, o sistema GPS e o sistema de segurança residencial. Um ator é qualquer coisa com o qual o sistema interage, um usuário, outro sistema ou o ambiente. Use o diagrama do contexto para fornecer os detalhes do sistema sendo considerado.



Com a definição do sistema e dos seus atores, você identifica os relacionamentos importantes e, com isso, os requisitos e interfaces. Ai então tem início o mapeamento das especificações da interface e do fluxo de dados entre o sistema e os seus atores.



Sem um diagrama do contexto, é possível deixar de ver um ator ou dois — e cometer falhas quanto aos requisitos do sistema. Por exemplo, com a falha na definição do sistema de segurança residencial como um ator do ecossistema, o seu carro deixa de ser inteligente o suficiente para armar o sistema de alarme.

Capítulo 3

Requisitos revolucionadores

Neste capítulo

- ▶ Reconhecimento dos benefícios da mudança
 - ▶ Cobertura de todos os requisitos com a inclusão de casos de utilização
 - ▶ Requisitos em cascata no processo de desenvolvimento
 - ▶ Análise do impacto da mudança
 - ▶ Compreensão (do gerenciamento dos requisitos)
-

Antigamente os requisitos eram criados no início de um projeto e o produto era projetado ao redor destes requisitos e, quando o departamento de marketing informava que o cliente queria fazer uma mudança (ou duas ou três), você batia o pé, fazia cara feia e reclamava do processo de mudanças dispendioso dos requisitos que exigia 57 assinaturas.

Mas isso está mudando. Em um mercado volátil e competitivo cujo sucesso está atrelado a satisfazer cada vez mais os clientes, você tem duas opções: mudar ou reclamar. Neste capítulo, você descobrirá como projetar os requisitos do sistema já pensando nas mudanças para garantir que o design do sistema satisfaça estes requisitos.

Adoção da filosofia de mudança

Nesta nova era de produtos inteligentes, é preciso responder rapidamente à dinâmica do mercado, como as mudanças das necessidades do cliente, novas ameaças de concorrência ou o mais recente padrão regulatório. Os criadores de produtos também observaram durante os últimos anos que os requisitos *devem* mudar com o melhor entendimento das necessidades durante o processo de desenvolvimento. Para o processo de desenvolvimento de sistema tradicional, a mudança é o inimigo. E o que você deve fazer?

Criar um novo processo.

A boa notícia é que outras pessoas que passaram por isso estabeleceram um processo de *projeto de requisitos* totalmente novo. Mais do que simplesmente uma análise e definição antecipada dos requisitos, a engenharia de requisitos define completamente o processo para o estabelecimento dos requisitos, atrelando-os aos testes, e facilitando a mudança.



Os requisitos funcionam melhor quando projetados com o pensamento na mudança. A filosofia da engenharia de requisitos é que a mudança é bem-vinda. Na realidade, a mudança é uma meta!

Compreensão do contexto

Antes de estabelecer um conjunto inicial de requisitos para um produto ou sistema inteligente, vale a pena pensar um pouco no problema que o seu produto está tentando resolver. Por exemplo, se a intenção é criar um carro, é essencial que você entenda exatamente como e por quem ele será usado. O carro será para a cidade, estrada ou corrida? O mercado alvo consiste principalmente em motoristas mais velhos, homens jovens ou pais que não trabalham? O carro terá que enfrentar condições difíceis, como frio extremo, rodovias com sal, calor extremo, terreno montanhoso ou alta altitude?



A compreensão do *contexto* é essencial para o desenvolvimento de sistemas que alcancem as metas de marketing e de negócios. O contexto significa um conjunto de *atores* (por exemplo, usuários, outros sistemas e o ambiente) com o qual o sistema interage, e como as interações do sistema com seus atores são realizadas.

A Figura 3-1 mostra um diagrama do contexto para um carro com um sistema de navegação integrado e um sistema de controle de segurança de residência voltado para uso em climas frios. Neste diagrama, o carro é tratado como uma “caixa preta” que interage com os seguintes atores: o motorista, o sistema GPS e o sistema de segurança residencial e o clima frio. A definição do contexto ajuda a entender qual funcionalidade é exigida do sistema e que tipo de trocas ocorrem entre o sistema e seus atores.

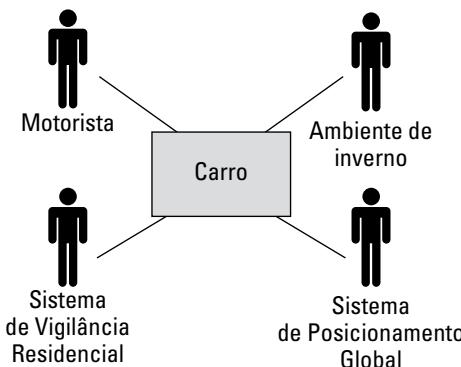


Figura 3-1: O contexto de um sistema delinea os limites e especifica as interfaces.

A compreensão do contexto também ajuda a garantir de que todos os requisitos, relacionamentos e interfaces necessários sejam considerados antes do início do processo de desenvolvimento. Por exemplo, a omissão do ator “ambiente frio” para o carro pode resultar na falha de especificar requisitos de um “pacote para conveniência climática” com cabos de bateria pesados e um revestimento protetor do chassi.

No mais alto nível de abstração, o seu sistema é uma “caixa preta” que interage com um conjunto externo de atores. A caixa preta tem um conjunto de subsistemas interconectados (por exemplo, freios antitravamento, sistema de navegação, etc.) que compõe o sistema. É possível decompor cada subsistema em um conjunto de componentes interconectados (e, talvez, subsistemas). Em cada nível da decomposição (por exemplo, sistema, subsistema, componente) do seu sistema, o contexto muda.

Compare o diagrama do contexto do carro (veja Figura 3-1) com o diagrama do contexto do subsistema de navegação integrado do carro (veja Figura 3-2). No nível do sistema, o carro é a caixa preta que interage com atores externos. No nível do subsistema, o subsistema de navegação é a caixa preta que interage com diferentes conjuntos de atores: o motorista, o subsistema elétrico do carro e o sistema GPS.

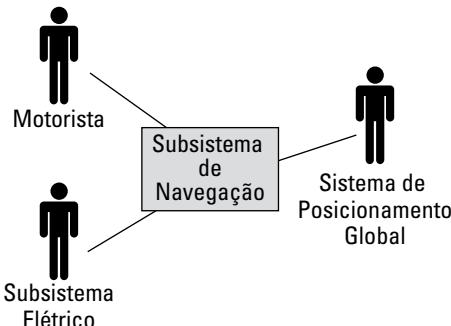


Figura 3-2: O contexto muda com a exploração dos diferentes níveis do sistema.



A compreensão do contexto delinea os limites do sistema e define as interfaces, estabelecendo a base para uma especificação precisa dos requisitos do sistema.

Mergulho nos requisitos

Pense nos requisitos como três amplas categorias ou níveis:

- ✓ **Requisitos de origem.** São os requisitos obtidos dos clientes ou interessados. Podem ser amplos e gerais, detalhados e específicos, abrangentes ou fragmentados — ou mais provavelmente, um pouco de tudo. Como diz o ditado, os clientes não seguem nenhuma regra para oferecer os requisitos — o que você conseguir já é lucro.
- ✓ **Requisitos da missão ou dos negócios.** São os requisitos que especificam o contexto operacional do sistema — não o que o sistema faz, e sim o papel que ele tem no mundo. Para uma nova aeronave militar, pode descrever os tipos de missões a serem realizadas. Para um novo smartphone, os requisitos dos negócios podem descrever a operação da infraestrutura de comunicação da operadora do celular.

✓ **Requisitos do sistema/subsistema.** São os requisitos que definem o que o sistema deve fazer. Eles iniciam em um alto nível do sistema e são analisados e decompostos para produzir os requisitos dos subsistemas de nível mais baixo. Eles podem ser expressados com sentenças com o termo “deve” ou em formatos mais avançados como modelos e diagramas.

Em cada nível da abstração, os requisitos definem o que um sistema deve fazer e como ele deve ser feito, mas não como deve ser implementado.

Com os sistemas inteligentes tornando-se cada vez mais complexos, é quase impossível chegar aos requisitos do sistema certos no princípio. E nas áreas como de eletrônicos de consumo, por exemplo, no qual a mudança é mais importante do que a estabilidade, você não deve congelar os requisitos no início do processo de desenvolvimento.

Busca de mais informações

Vale a pena obter a maior quantidade de informações possível para os requisitos da origem, do maior número possível de interessados desde o início do processo de engenharia dos requisitos. Você deve iniciar com os clientes, é claro, mas também deve obter informação sobre os padrões regulatórios, da indústria e de segurança que governam o seu sistema, interfaces e troca de dados com outros sistemas (como por exemplo, o sistema GPS), e as limitações dos negócios e de marketing.



O objetivo é obter os requisitos que descrevam a capacidade e não as funções do sistema. (Por exemplo, o que o sistema deve fazer e não como deveria fazer.) Pergunte a um cliente que esteja descrevendo a função, o que ele quer. Isto deve levar à capacidade.

Também é importante especificar os requisitos *funcionais* e *não-funcionais*. Use os requisitos funcionais para descrever o que o sistema deve fazer, de acordo com certas informações. Por exemplo, em um determinado ponto inicial e de destino, o seu sistema de navegação deve apresentar um mapa e exibir uma rota. Use os requisitos não funcionais para especificar os requisitos de performance ou de qualidade, ou para impor limites ao design. Inclua nestes requisitos coisas como velocidade, capacidade, confiabilidade, peso, uso e escalabilidade.

Para ter certeza que cobriu todos os ângulos, desenvolva *casos de utilização* que descrevam todas as possibilidades de utilização das funções do sistema. Por exemplo, a função “mapa da rota” do seu sistema de navegação pode ser usado para “encontrar o posto de gasolina mais próximo” ou para “mostrar os hotéis na vizinhança do meu destino”. Os casos de utilização normalmente são compostos de sequências de uma ou mais funções do sistema. Os casos de utilização contam histórias concretas da utilização do sistema e podem ser usados em três níveis de requisitos — origem, missão/negócios e sistema/subsistema.



Com a captura da utilização durante o desenvolvimento dos requisitos, é mais provável que você projete um sistema que proporcione um valor real.

Satisfação e obtenção de requisitos

Após avaliar o contexto do sistema e definir um conjunto inicial de requisitos de alto nível do sistema, use estes requisitos para chegar ao processo de design de alto nível (veja Figura 3-3). Durante o processo de design são desenvolvidos os requisitos adicionais, como o Plano de Verificação do Sistema a ser atingido durante o teste do sistema, ou os limites do design arquitetônico que devem ser alcançados pelo projeto detalhado.

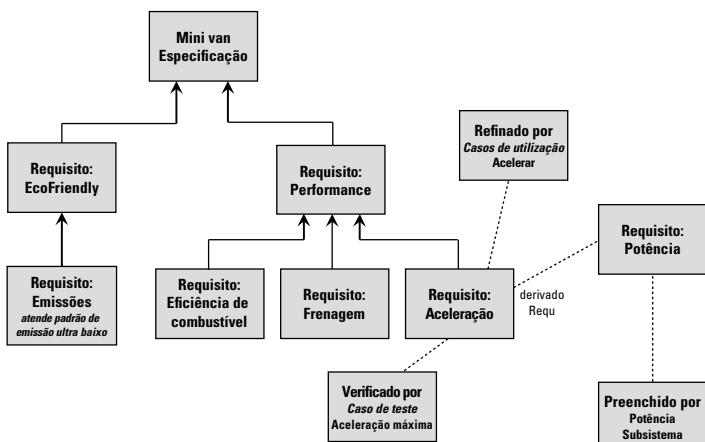


Figura 3-3: Processo do design de alto nível.



O exame do alcance do seu design atende os requisitos desenvolvidos no nível de design anterior (mais alto) e trazem requisitos que devem ser usados pelo nível do teste correspondente, bem como do nível de design seguinte (mais baixo).

Como você pode ver, a definição de “requisitos” está expandindo, pois as linhas que diferenciam os requisitos do design estão cada vez mais tênues, e a conexão entre o teste e o design tornam-se críticas. O conjunto final de requisitos é uma combinação dos requisitos iniciais do sistema de alto nível com os requisitos obtidos durante o processo de design.

Criação dos diagramas dos requisitos

Embora os requisitos antigos baseados em texto ainda sejam necessários para grandes projetos com obrigações contratuais, normalmente não são suficientes no mundo atual dos produtos inteligentes complexos, flexíveis e focados no cliente. Felizmente, a Linguagem de Modelagem de Sistemas (SysML) amplamente adotada fornece uma base de trabalho para os requisitos de modelagem. Enquanto os grandes conjuntos de requisitos de texto ainda existem em um banco de dados, principalmente os enviados pelo cliente, existem agora novas formas melhores de visualizar e trabalhar com conjuntos de requisitos afins.

A SysML permite a criação de modelos de requisitos hierárquicos que ilustram as dependências, classificam os requisitos como originais ou derivados, e captam a lógica da opção do design. São dois os níveis de requisitos de sistema de alto nível: EcoFriendly e Performance. (Veja Figura 3-3.) O requisito EcoFriendly tem um sub-requisito específico que é Emissões.

O requisito Performance tem três sub-requisitos, sendo um deles a Aceleração. O sub-requisito Aceleração é mais refinado em casos de utilização específicos. É definido um plano de teste para aceleração máxima (um requisito que o processo deve atender). Finalmente, o sub-requisito Aceleração gera um requisito derivado para energia, atendido pelo subsistema Potência.

Os diagramas dos requisitos SysML permitem a fácil visualização dos relacionamentos e a resposta de perguntas do tipo: “Quais requisitos o subsistema Potência atende?” ou “Qual pode ser o impacto da mudança

do requisito Tamanho da Carga?” Quando os requisitos são tratados como parte integral da arquitetura do sistema, é muito mais fácil visualizar o impacto das mudanças dos requisitos do seu sistema. E esta não é uma das metas da engenharia de sistemas?

Direcionamento das decisões do design com os requisitos

Com os requisitos em cascata no processo de design, a natureza de alguns deles podem levar a opções de design específicas. Por exemplo, ao projetar um sistema anticolisão de avião, o início do processo pode apresentar três opções de design, cada uma delas satisfazendo os requisitos funcionais para evitar a colisão:

- ✓ Sistema de radar de bordo
- ✓ Transponder
- ✓ Principalmente manual (comunicação com o controle de tráfego aéreo)

Cada opção de design tem o seu conjunto único de componentes, requisitos de espaço e custos, e envolve certo nível de colaboração dos sistemas com seus atores (por exemplo, controle de tráfego aéreo). Com a maior decomposição do design, é possível encontrar um requisito não funcional, como “espaço máximo da cabine de comando” que limita as opções de design (o sistema de radar não cabe!).

Os engenheiros de sistemas usam uma técnica chamada *estudo de compensação* para avaliar as muitas opções de design que devem ser adotadas. É basicamente a mesma coisa que todos fazem ao tomar uma decisão importante. Identificar os fatores importantes para a decisão, pontuar cada alternativa de cada fator, adicionar ajustes de ponderação e tomar a decisão. Os estudos de compensação normalmente exigem vastas pesquisas para a avaliação total das alternativas.

O acompanhamento de todos os requisitos e como eles afetam as outras partes do sistema são um mal necessário para a oferta de produtos de sucesso. As ferramentas de software prontas para uso ajudam a administração dos requisitos do sistema.



A omissão de um requisito pode causar problemas sérios. Por exemplo, é possível projetar um porta-copos para automóveis mais fenomenal do mundo, mas se ele ficar localizado embaixo do som do carro — fazendo com que o seu *café latte grande* bloquee os controles — porque você esqueceu de levar em consideração o requisito de “deixar um amplo espaço livre” no design, você vai errar feio. Erros como esse podem sair muito caro.

Malabarismo com os requisitos e Designs

No exemplo de anticolisão para aviões na seção anterior, vamos supor que você selecionou a opção de design “transponder do sistema”. Esta opção semiautomática envolve o monitoramento dos sinais do transponder do avião na área por parte do controle de tráfego aéreo. A opção de design dispara sub-requisitos de um design de subsistema de transponder e um plano de teste do subsistema.

Você já está bem adiantado no design do subsistema quando recebe um pedido de mudança de requisito. O novo requisito especifica que a aeronave deve evitar colisões de modo independente. A sua opção de design não atende a este novo requisito, por isso é preciso colocar este design de lado e trabalhar com uma opção alternativa, o sistema de radar.

Certas mudanças podem produzir requisitos derivados de um design de subsistema que podem propagar-se até o design do sistema. Por exemplo, as restrições de custo podem limitar a escolha de um componente do sistema que, por sua vez, limitam a funcionalidade do subsistema, afetando os requisitos originais. Antigamente (alguns anos atrás!) este processo de propagação da mudança consumia muitas horas ou até semanas, com os engenheiros rastreando e modificando os documentos e designs relevantes nos documentos em processador de texto, slides de apresentação e planilhas. A introdução de modelos no processo ajuda muito, conforme discutido no Capítulo 4.



Estabeleça um processo formal para mudanças de requisitos e assim entender facilmente o impacto das mudanças no design do seu sistema.

Vinculação dos requisitos com os testes

Você concluiu o design de um novo sistema de navegação de automóvel super-sensível. O protótipo já foi construído e o time de testes informou que o sistema está funcionando muito bem. Na realidade, ele calcula a velocidade da rota entre dois pontos com tal velocidade (graças ao seu algoritmo brilhante), que deve acabar com a concorrência. O seu chefe, todo orgulhoso, oferece o protótipo para o seu CEO usar durante um dia. O CEO fica impressionado até tentar encontrar o restaurante chinês mais próximo — e perder a paciência esperando pela resposta.

Você tenta descobrir o que aconteceu e percebe que o seu algoritmo foi otimizado para cálculos de rota de ponta-a-ponta mas não para encontrar locais dentro de uma certa distância. E, embora o plano do teste do seu sistema tenha testado a funcionalidade do sistema, não testou todos os casos de utilização (ou talvez alguém tenha esquecido de criar este caso de utilização).



Um componente principal do processo de engenharia de sistemas é o estabelecimento de vínculos entre os requisitos e os testes. Tenha certeza de construir o sistema que você planejou — e não apenas um sistema que “funcione”.

Em cada nível da decomposição do sistema, com o ajuste e a derivação dos requisitos, você cria e refina os planos de teste para confirmar que o sistema atende aos requisitos. Quando os requisitos mudam, os planos do teste também devem mudar. Normalmente, o ato de escrever um critério para testes de um conjunto de requisitos ajuda a melhorar e a refinar os próprios requisitos.



É sempre bom avaliar os requisitos e confirmar que eles *podem* ser testados e até especificar no início *como* eles devem ser testados.

Não esqueça de deixar um rastro

Para a garantia de que todos os requisitos sejam implementados adequadamente, é preciso rastrear cada um deles durante o processo de desenvolvimento e de teste.



Rastreamento dos requisitos é a capacidade de vincular cada requisito com três itens relacionados:

- ✓ As necessidades dos interessados (requisitos da origem) que são atendidas
- ✓ Os elementos que implementam ou executam o sistema
- ✓ O caso de teste que o verifica

O rastreamento ajuda a garantir a conformidade com os regulamentos padrões, evita o esquecimento de requisitos e mantém o foco nas metas gerais do projeto. Com o estabelecimento de vínculos de rastreamento completos, é possível avaliar exatamente o impacto da mudança de requisitos mais recente ou uma opção de design alternativa — antes que a mudança seja feita.

Recompensa do seu trabalho tão duro

Nos sistemas grandes e complexos, o gerenciamento dos requisitos pode ser um pesadelo. Muitos times de desenvolvimento consistem em centenas — e até milhares — de arquitetos e engenheiros, e todos eles tocam os requisitos, seja para criar e editar ou simplesmente para rever e entender estes requisitos. O rastreamento normalmente atravessa quatro níveis de decomposição enquanto os requisitos dos interessados são percorridos em cascata até o design do componente e os requisitos de teste. Além disso, estes níveis de decomposição normalmente atravessam as fronteiras da cadeia de suprimentos, tornando a vida ainda mais desafiadora.

Um gerenciamento eficaz dos requisitos envolve muitas disciplinas de engenharia, incluindo o design do sistema, arquitetura, software, mecânica, eletricidade e engenharia do teste. As funções dos negócios, como o marketing e compras, também estão interessadas no gerenciamento dos requisitos.



As ferramentas de software podem ajudar no entendimento do processo de gerenciamento dos requisitos difíceis. Estas ferramentas são projetadas para manter o histórico das auditorias, preservar todas as mudanças,

conduzir análises de impacto, e automatizar o processo de gerenciamento da mudança. Elas também podem alertar sobre os requisitos ignorados, designs muito trabalhados e não conformidade.

O gerenciamento dos requisitos é muito difícil, mas com a ajuda das ferramentas certas é possível ter um grande retorno em termos de custo, agendamento e sucesso do projeto — e manter a sanidade mental.

Capítulo 4

Abstração do sistema de modelagem

Neste capítulo

- ▶ Decomposição do sistema em níveis de abstração
- ▶ Visualização de como os elementos se encaixam
- ▶ Como o sistema deve se comportar
- ▶ Ajuste dos modelos do sistema com a iteração

Uma fórmula é um tipo de modelo. Ela capta o relacionamento matemático das variáveis inseridas e o representa com uma estrutura matemática que é aplicada a diversos tipos de entradas. A visualização da fórmula ajuda o entendimento da relação dos elementos da estrutura. E, com o uso da fórmula, é possível testar vários tipos de possibilidades diferentes até encontrar a melhor fórmula.

Neste capítulo, vamos explorar como é possível usar os modelos do sistema para administrar a complexidade, o relacionamento abstrato essencial do sistema e executar testes econômicos antes da construção do produto.

Arquitetura do sistema de modelagem

Os modelos de arquitetura viabilizam a captação da natureza estática do sistema, incluindo a estrutura e o uso pretendido do sistema. Por exemplo, considere o modelo arquitetônico simplificado de um carro mostrado nas Figuras 4-1 e 4-2.

O modelo da estrutura parcial da Figura 4-1 decompõe o sistema em diversos níveis. No nível superior, vemos o sistema geral: o carro. Um nível abaixo, vemos dois dos muitos subsistemas de um carro: o sistema de freio antitravamento (ABS) e o chassi.

O subsistema do chassi se decompõe ainda mais em outro subsistema (o conjunto dos cubos), um componente mecânico (o pneu), e outros subsistemas e componentes não mostrados neste diagrama. O subsistema do conjunto dos cubos consiste em um componente eletrônico (sensor) e diversos componentes mecânicos não mostrados na figura.

O subsistema ABS se decompõe em um componente mecânico (rotor) e outro subsistema (o controlador antitravamento). O subsistema controlador antitravamento consiste em dois componentes eletrônicos: o detector de tração e o modulador do freio. As linhas pontilhadas que conectam o sensor do conjunto dos cubos com o controlador antitravamento mostram que os dois se relacionam, embora o sensor não seja um elemento do subsistema do controlador antitravamento.

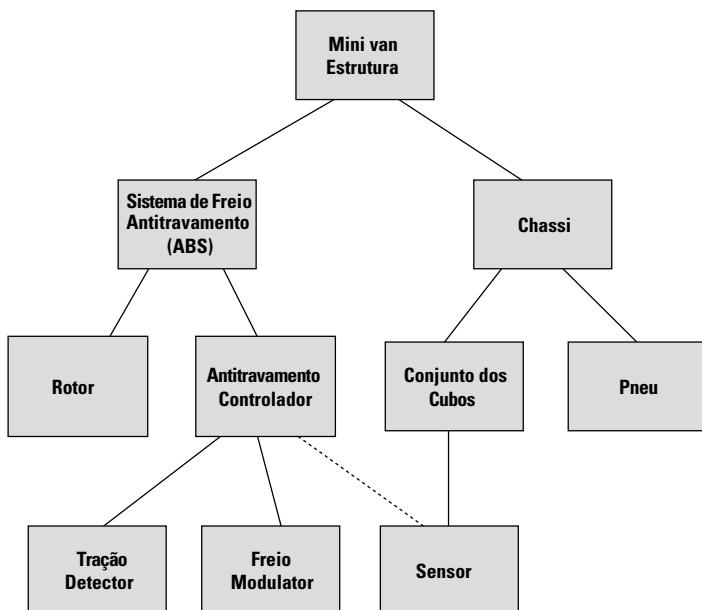


Figura 4-1: Um modelo arquitetônico simplificado de um sistema de minivan.



Os modelos estruturais, como o modelo arquitetônico simplificado mostrado na Figura 4-1, captam a estrutura hierárquica da arquitetura do sistema e ilustram as conexões dos elementos do modelo.

A Figura 4-2 mostra outra peça do modelo arquitetônico completo: uma visão interna de como as peças do subsistema do controlador antitravamento interagem entre elas e com o sensor do conjunto dos cubos. Neste caso, a saída do sensor é conectada à saída do detector de tração e a saída do detector de tração é conectada à saída do modulador do freio. Este diagrama simples ilustra as conexões para indicar o uso *pretendido* mas não especifica as condições nas quais as interações acontecem. Os modelos de comportamento (discutidos na próxima seção) são usados para descrever a sequência de eventos que devem acontecer para que as interações ocorram (por exemplo, a leitura do sensor indica a perda de tração, e isso faz com que o detector de tração acione o modulador do freio).

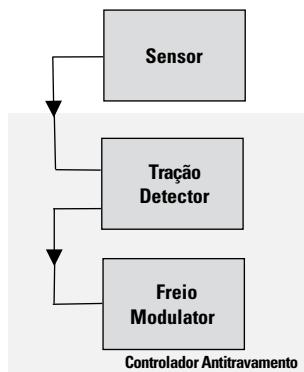


Figura 4-2: Modelos arquitetônicos captam a informação de uso.

Os modelos estruturais como estes podem mostrar a arquitetura física, como no exemplo acima, ou podem ser usados para mostrar a arquitetura lógica. As arquiteturas lógicas são usadas cada vez mais no desenvolvimento de sistemas complexos porque permitem que os engenheiros calculem as interações funcionais sem especificar uma estrutura física particular. Vejamos o carro da Figura 4-1. Há algumas décadas, os subsistemas de carros, como freios, motor e o rádio, eram completamente separados, e era apenas necessário entender a arquitetura física para construir um carro. Pense agora na arquitetura lógica de um carro que tenha elementos como estes:

- ✓ Propulsão, que ultrapassa as fronteiras físicas dos freios, parte elétrica, motor e controles
- ✓ Gestão da Informação, que inclui toda a informação recebida, armazenada, enviada ou usada no carro
- ✓ Entretenimento, que pode ter componentes físicos em comum com a propulsão e a informação
- ✓ Segurança, que pode estar vinculada a muitos outros sistemas do carro como freios, gestão da informação e a interface de usuário do motorista

Ao considerar a arquitetura lógica separadamente — e, o ideal, antes da implementação física — os engenheiros de sistemas podem adotar abordagens melhores e mais elegantes. Uma abordagem das arquiteturas lógica e física é uma grande forma de administrar a complexidade dos novos produtos inteligentes integrados — que não são mais projetados como o carro do seu avô.

Comportamento do sistema de modelagem

Para entender como um sistema se comporta, é preciso entender como os componentes da arquitetura (lógica e física) de um sistema interagem. Os modelos de comportamento do sistema captam a informação dinâmica do sistema, como as transições de estado, ações que o sistema executa em resposta a eventos específicos, e interações das peças que colaboram dentro de um sistema. Os modelos também captam o fluxo de dados e o controle das atividades, como por exemplo, como os dados de saída do sensor passam para as atividades ocorridas no controlador antitravamento de um carro, ou como o controle passa de uma peça para outra do sistema.

A Figura 4-3 mostra um diagrama do estado simplificado que descreve os estados operacionais de um carro. Este carro permanece em um dos quatro estados — desligado, marcha lenta, acelerando ou freando — até que um evento específico ocorra (por exemplo, “frear”) que faz com que o carro passe para outro estado.

É possível mapear as conexões dos modelos de comportamento com os modelos estruturais para aplicar a consistência através de um processo chamado *alocação*. Por exemplo, alocação da ação “detectar perda de tração” para o componente “detector de tração” do modelo estrutural do sistema, e alocação da ação “força do modulador do freio” para o componente “modulador do freio”.

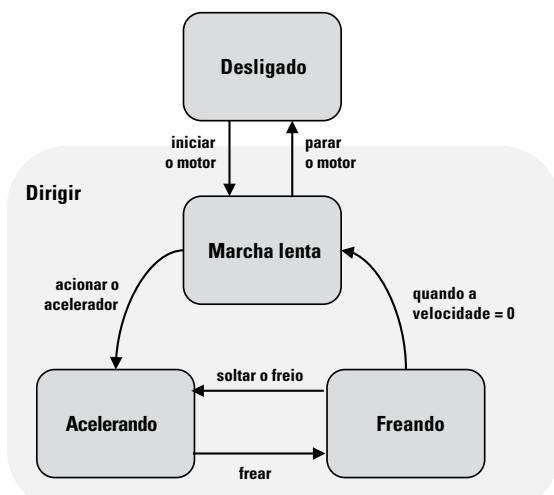


Figura 4-3: Modelo de comportamento mostrando as transições dos estados operacionais.

Nos sistemas complexos, é preciso muitos modelos de pequenos comportamentos para representar toda a atividade ocorrida. Em muitos casos, uma atividade aciona uma ação em outra parte do sistema, por isso os modelos individuais são interconectados. Ao final da modelagem de toda a atividade, você tem um modelo grande, complexo, uniforme e consistente do comportamento geral do sistema.



Com o uso da construção e da semântica da modelagem padrão, como as fornecidas pela Systems Modeling Language (SysML), é possível usar ferramentas de software comercial para automatizar a execução dos modelos de comportamento do sistema. As ferramentas podem traduzir automaticamente as construções de modelagem, como os diagramas de transição do estado, em sentenças com código “se-então-executar”. Desta forma, é possível simular o comportamento do sistema no software, viabilizando a execução dos estudos “e-se”, ver as alternativas do design e realizar análises de impacto antes da construção do sistema. Os estudos de compensação, que normalmente levam horas ou até dias, podem ser realizados em minutos desta forma.

Mapeamento dos modelos

Os padrões da indústria, como o SysML, são a base para o entendimento comum da modelagem de sistemas em todos estes estágios. Os modelos

SysML são um conjunto de *diagramas* que representam a estrutura, comportamento, requisitos e limites quantitativos de um sistema. Os diagramas SysML vêm em quatro tipos diferentes:

- ✓ **Estrutura:** Descreve que elementos arquitetônicos (lógico e físico), conhecidos como *blocos*, estão contidos no sistema ou subsistema e como eles se conectam. Por exemplo, um subsistema de freio antitravamento contém um detector de tração e um modulador do freio.
- ✓ **Comportamento:** Descreve o comportamento do sistema ou subsistema, incluindo as transições de estado, sequências de atividades, funções e interações. (Por exemplo, “detectar perda de tração” aciona “força do modulador do freio.”).
- ✓ **Requisitos:** Descreve os requisitos específicos do sistema ou do subsistema. (Por exemplo, fornecimento da distância de parada específica.)
- ✓ **Paramétrico:** Descreve os limites paramétricos do sistema ou subsistema. (Por exemplo, a equação da força de frenagem é uma equação paramétrica.)

Exploração dos quatro estágios da modelagem do sistema

Os sistemas são por natureza estruturas recursivas que consistem em diversos níveis começando com o sistema geral no mais alto nível, e são decompostos em subsistemas e, então, em componentes. A melhor maneira de modelar um sistema completo é usar um processo recursos de multietapas — começando de cima — composto dos seguintes quatro estágios. Os estágios formam a palavra CURE, pense como a cura da complexidade:

- ✓ **Contexto:** Estabelece os limites do sistema, identifica as pessoas e os sistemas com os quais o sistema interage (também conhecidos como atores) e descreve as interfaces (como eles se comunicam com o sistema e o que eles trocam). Juntos, os elementos deste modelo contextual são conhecidos como a *empresa* — significando o sistema e seu ambiente. Use o diagrama do bloco SysML para isso.
- ✓ **Uso:** Descreve todas as maneiras que os atores usam o sistema. Inclui quem ou o que usa o sistema, e quem ou o que o sistema usa. Isto é realizado melhor com histórias específicas, passo-a-passo do tipo

narração do uso do sistema, como casos de utilização, e ilustradas como diagramas de atividade SysML. Ajusta os requisitos do sistema, e os torna específicos e completos (não esqueça, os requisitos de origem podem ignorar muita coisa importante). Os mesmos cenários de utilização tornam-se a base para os testes posteriores do sistema — de acordo com a informação fornecida pela análise de utilização.

- ✓ **Realização:** Define os modelos da estrutura (arquitetura) e comportamento (função) que, juntos, descrevem como cada utilização é alcançada pelo sistema através da colaboração dos elementos da arquitetura do sistema. O comportamento exigido é realizado (torna-se realidade) nos elementos do sistema. Isto é muito diferente do processo de design tradicional que aloca os requisitos para os componentes físicos e force para que funcione com a utilização real. Aqui, a utilização é definida especificamente e o design do sistema é feito de acordo com as utilizações específicas, e você *sabe* que o sistema é projetado para atender estes requisitos de utilização.
- ✓ **Execução:** Executa os modelos de comportamento para demonstrar que o design atende estes requisitos. Os modelos simples executáveis, até mesmo os níveis mais altos de abstração são uma forma excelente e econômica de descobrir os problemas complicados, má comunicação, ausência de requisitos ou requisitos ambíguos, e outras questões que atrapalham a programação desde o início. Todos chegam a um acordo antes de qualquer coisa ser construída.

Inicia no nível mais alto da decomposição do design do sistema: o nível do sistema (por exemplo, uma minivan). Após estabelecer os modelos de contexto e utilização, os modelos arquitetônicos e de comportamento de alto nível são definidos de acordo com os requisitos do sistema. Ai então, os modelos são executados para demonstrar que o sistema faz exatamente o que deveria fazer. Após concluir o processo do nível do sistema, o processo é repetido no nível de decomposição seguinte: o nível do subsistema.

Continue a examinar os níveis de decomposição, mudando o contexto ao passar para cada nível do modelo, até alcançar o nível mais baixo — o nível do componente — onde a implementação física do design é especificada (por exemplo, eletrônicos, software ou design mecânico).

Em cada nível, chegue horizontalmente “ao V” e execute a verificação e validação, usando o modelo como base. Use modelos executáveis e outros tipos de simulações matemáticas e de design para realizar a



maior quantidade possível de verificações antes de chegar ao estágio da implementação.

O ponto sagrado aqui é um modelo completo do sistema — um tipo de versão virtual da realidade do sistema real. Isso ainda não é possível, mas com as ferramentas de modelagem cada vez melhores e aprendendo a se interconectar com as disciplinas de engenharia, estamos cada vez mais perto.

Porque a modelagem está na moda

Pode parecer que a modelagem da arquitetura e do comportamento de um sistema complexo não vale a pena — até lembrarmos como foi difícil o último projeto, quando ninguém do time de desenvolvimento assumiu a responsabilidade por ter ignorado um requisito essencial, e a reunião post-mortem fez a Inquisição Espanhola parecer uma festa.

Com a modelagem é possível captar todos os detalhes cabeludos do design do sistema de maneira organizada, permitindo a visualização, entendimento e assimilação das complexidades da estrutura e do comportamento do sistema. Ela permite a exploração de opções diferentes de arquitetura e design, a execução de estudos de compensação, e a avaliação do impacto das mudanças antes do início da construção do sistema — para a redução dos riscos e dos custos de desenvolvimento do projeto.

Modelos fornecem um contexto para a discussão das questões do sistema em diversos níveis. É possível usar modelos para explorar diversas visões de um sistema — planejamento, requisitos, arquitetura, design, implementação, acionamento, comportamento, entrada de dados, saída de dados e mais — para que você e seus colegas possam analisar as grandes questões tão facilmente quanto as pequenas questões do design.

Com o uso das linguagens e técnicas de modelagem padrão da indústria, como SysML, é possível reduzir a ambiguidade e remover as barreiras da linguagem que possam existir entre os membros de diversos times de desenvolvimento e fornecer uma única fonte mestre para o status e a documentação do desenvolvimento do projeto. Uma colaboração melhor e mais clara com documentos precisos proporciona maior eficácia, menores ciclos de desenvolvimento e, acima de tudo, maior qualidade.

Capítulo 5

Garantia da qualidade de primeira

Neste capítulo

- ▶ Integração da qualidade no processo de desenvolvimento de sistemas
- ▶ Iteração dos testes e do design
- ▶ Uso de modelos para revelar erros antecipadamente

Antigamente a garantia de qualidade era um processo realizado no final do ciclo de desenvolvimento como se a qualidade fosse um recurso tangível que pudesse ser adicionada ao produto antes da entrega. E quando os defeitos eram descobertos, era preciso um grande trabalho para identificar e resolver a raiz do problema.

No mundo de hoje onde os produtos inteligentes com softwares cada vez mais complexos, a qualidade passou a ser parte integral do processo de desenvolvimento de sistemas, trazendo a esperança da entrega de produtos sem defeitos e que demonstrem adequação para o objetivo. Neste capítulo, veremos como a engenharia de sistemas ajuda a identificar as questões de qualidade com a validação e a verificação antecipadas no ciclo de desenvolvimento, para aumentar a possibilidade de que o projeto tenha sucesso.

Exploração dos níveis de teste

Ao projetar um produto inteligente com vários subsistemas e milhares (ou milhões) de linhas de código, você pode pensar como é possível verificar e validar todo o sistema. Onde começar? Como ter certeza de que todas as peças do quebra-cabeça se *encaixam* e também *funcionam* adequadamente para oferecer a funcionalidade do produto que os interessados (e acionistas) buscam? Bem, na realidade, o processo de qualidade começa no início do design do sistema.

Os sistemas grandes e complexos são decompostos em diversos níveis, incluindo o nível do sistema, um ou mais níveis do subsistema e, finalmente, o nível do componente.



Embora possa parecer intuitivo esperar até que o sistema esteja pronto para fazer os testes, a boa prática de engenharia de sistemas inclui técnicas mais sofisticadas para que o sistema tenha qualidade durante a sua construção. Durante a construção, cada componente do sistema (hardware ou software) é testado independentemente e depois integrado com outros para formar o subsistema. A seguir, o subsistema é testado independentemente e depois integrado com outros para formar o sistema, e o sistema é testado. Por último, todo o sistema é testado juntamente com o ambiente operacional (contexto) para verificar se todo o conjunto faz o que deve fazer no ambiente real.

Além de testar o sistema durante a sua construção, é possível verificar e validar os modelos e simulações do sistema para descobrir os problemas no início — antes que qualquer placa do circuito seja montada ou soldada. É possível até verificar as atividades durante a etapa inicial de obtenção de requisitos para conferir se a interpretação da necessidade do cliente está correta.

Teste da unidade

O teste do nível mais inferior do sistema está intimamente ligado à implementação do seu design. Cada componente de hardware ou software é testado, os defeitos são corrigidos e a implementação é refeita. Quanto ao software, isto pode envolver diversos ciclos iterativos de codificação, testes e recodificação até o software estar completamente depurado.

Após abordar todos os defeitos conhecidos, é hora de verificar se o componente atende aos requisitos alocados. Lembra quando você desenvolveu o design detalhado do sistema? Bom, parte da fase do design detalhado envolveu a definição dos requisitos específicos para os componentes e a criação de um Plano de Verificação da Unidade. Agora, os casos de teste são definidos no Plano de Verificação da Unidade para verificar se o componente atende aos requisitos alocados. Quando os defeitos são descobertos, volte para a implementação para corrigi-los. Após o exame detalhado dos componentes, eles podem ser integrados em conjuntos ou subsistemas de nível superior.



Certifique-se de que o seu Plano de Verificação da Unidade documenta detalhadamente os casos e os resultados de testes específicos de cada componente e que uma matriz de rastreamento seja usada para vincular os testes aos requisitos específicos que foram verificados. Com isso, quando todos os testes são aprovados, o sistema atende a todos os requisitos.

Integração e verificação do subsistema

Os componentes de hardware e de software completamente verificados estão prontos para a integração com os módulos ou subsistemas. Se as interfaces foram testadas primeiro, isto não deve ter problemas.



O objetivo desta etapa de testes é garantir que todas as interfaces entre os componentes e os conjuntos tenham sido implementadas adequadamente e que todos os requisitos e limites do subsistema tenham sido atendidos.

Dependendo do grau de complexidade do subsistema, pode ser necessário desenvolver uma plano de integração que defina a ordem para a integração dos componentes e sub-conjuntos de nível mais inferior. Se possível, planeje a integração para que as peças que devem trabalhar em conjunto também estejam prontas para a integração. Em cada etapa da integração, compare a funcionalidade do subconjunto com o conjunto de requisitos apropriados, usando o Plano de Verificação de Subsistema definido na fase do design.



Cuidado para não ignorar os testes realizados para verificar os requisitos no nível do componente, pois muitos requisitos abrangem como cascata diversos níveis da decomposição do sistema. Por exemplo, quando um requisito do sistema especifica que uma tela de display deve ficar em branco quando um usuário pressiona um botão, é preciso verificar se o componente da tela pode ficar em branco (teste de nível de componente), e é preciso verificar também se com o apertar do botão a tela fica em branco (teste de nível de subsistema).

Teste do sistema

Com a iteração progressiva, é possível integrar, testar e verificar os subsistemas até chegar ao nível do sistema (veja Figura 5-1). Cada iteração envolve um teste cuidadoso e detalhado — com atenção especial

para as interfaces — e a verificação de que os requisitos apropriados foram atendidos. No nível mais alto, são realizados testes para verificar se o sistema em geral atende aos requisitos do sistema de nível alto definidos no início da fase do design. Repetindo, os testes são baseados no plano de verificação definido em paralelo com os requisitos do sistema.



É importante documentar os resultados de cada caso de teste e anotar quaisquer respostas inesperadas ou outras anomalias. No entanto, resista à tentação de corrigir um defeito recém-descoberto para evitar a perda do controle da configuração. Ao invés disso, documente o problema, analise a causa e defina um plano de ação para resolver o problema dentro do contexto de um processo sistemático.

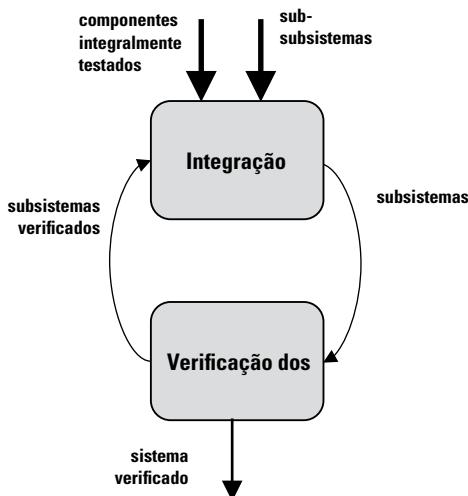


Figura 5-1: Integração e verificação são um processo iterativo.

Quando tudo corre bem, o resultado é um sistema verificado que pode ser apresentado para os interessados. É possível provar que todos os requisitos do sistema foram atendidos, confirmando que o sistema foi construído corretamente.

Teste de aceitação do sistema

“É só construir que os interessados aparecem.” Esta filosofia pode não ser muito verdadeira. O sistema pode ser perfeitamente projetado e

implementado, atender a todos os requisitos e muito mais mas, se não atender ao uso pretendido, está fadado a ser um fracasso.

Vejamos, por exemplo, o caso de um sistema de controle de sinais luminosos de trânsito bem projetado. Habilmente projetado para controlar a sequência dos sinais luminosos de trânsito de uma grande cidade, este sistema pode falhar em atender o uso *pretendido* — reduzir o congestionamento com a agilização do fluxo do tráfego — se ninguém se preocupou em estudar os padrões típicos do tráfego da cidade e em transformá-los em um mapa da sequência de tempo para os requisitos do sistema.



O objetivo do teste de aceitação do sistema é confirmar se o sistema atende ao objetivo pretendido.

Na fase conceitual do projeto são identificadas as necessidades principais do interessado, a capacidade geral do sistema, cenários de utilização (CONOPS e casos de utilização) e medidas de performance para validação do sistema. Você definiu um Plano de Validação do Sistema e, se foi inteligente, o guardou em um cofre e não permitiu que ninguém o alterasse para acomodar objetivos indesejados como reduzir a qualidade para economizar muito pouco. O Plano de Validação do Sistema é a base sólida com a qual você deve provar que o sistema atinge os objetivos pretendidos.

Realize uma validação do sistema com usuários reais e meça a performance de acordo com o plano, incluindo até mesmo a satisfação do cliente. A validação pode exigir a coleta de dados antes, durante e depois da implementação do sistema. Após documentar cuidadosamente a performance do sistema, reúna-se com os interessados, analise todos os dados e avalie o sucesso do projeto.

Quando um problema é observado no sistema durante os testes, normalmente há uma falha no sistema que pode ser corrigida com uma modificação do sistema. Observe que o problema pode ser outra coisa. Quando o plano de verificação ou o procedimento de teste está errado ou desatualizado o teste pode falhar sem significar falha do sistema. Por outro lado, um requisito errado, ambíguo ou incompleto (principalmente um requisito de interface), podendo ser a fonte do problema — é mais um motivo para a concentração em obter requisitos e planos de testes corretos desde o início do processo.



Quando os testes de verificação indicam um problema, retroceda e confira os requisitos e design, faça os ajustes necessários e repita os testes.

Por exemplo, suponhamos que você esteja projetando um limpador de para-brisas com sensor de chuva (RSW). O objetivo do RSW é limpar o para-brisas automaticamente ao detectar gotas de chuva na parte externa do para-brisas. A arquitetura de alto nível do sistema RSW inclui um sensor óptico com um alcance operacional específico conectado na superfície interior do para-brisas, uma unidade de controle eletrônico e um software. O design exige que o sensor e o software interajam para identificar a presença de água e ativar o limpador de para-brisas.

Os testes dos componentes individuais indicam que o sistema RSW funciona de acordo com o projeto dentro da faixa operacional do sensor. Ai então são integrados o RSW, o para-brisa e outros subsistemas em um carro. No entanto, o teste da operação do RSW com todo o sistema, falha.

Após uma extensa análise da causa do problema, a fonte do problema é descoberta: as características físicas do para-brisa (mais especificamente, o índice e a espessura ópticos) são incompatíveis com a faixa operacional do sensor. Você conclui que não definiu um requisito para as características físicas do para-brisa que garantisse a compatibilidade com o sistema RSW. Neste caso, é preciso voltar para a fase do design, adicionar o requisito e redesenhar o para-brisa.

O que causou esta confusão foi uma *suposição* não expressada e não examinada, um clássico bicho-papão para os engenheiros de sistemas. A hipótese foi que o sensor funcionaria com qualquer tipo de material do para-brisa. Se você tivesse expressado esta hipótese, alguém poderia ter dito: “Não com qualquer material!” e identificado o requisito que estava faltando. Esta experiência dolorosa é um grande exemplo da importância de testes frequentes e desde o início.

Testes frequentes desde o início

É mais barato consertar um defeito descoberto no início. Como podemos ver na Figura 5-2, o conserto dos defeitos descobertos após o lançamento do produto podem custar até 100 vezes mais do que o conserto dos defeitos descobertos durante o processo dos requisitos. Mas isto é apenas um exemplo. Com a adesão estrita ao processo de testes, é possível reduzir drasticamente o custo do conserto dos defeitos.

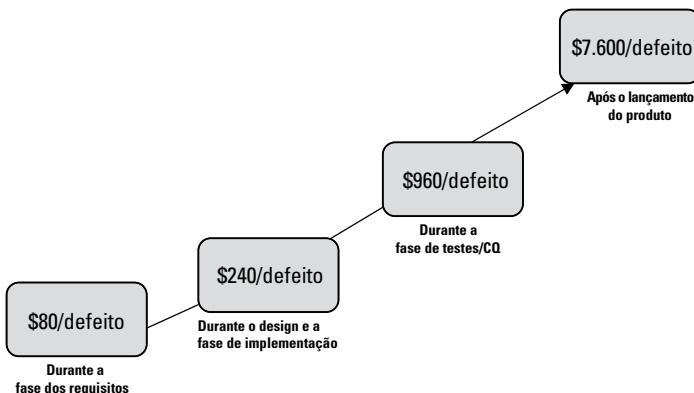


Figura 5-2: O custo da correção dos defeitos aumenta drasticamente com a progressão do processo de desenvolvimento.

O grande problema são as . . . Interfaces

Um dos grandes problemas do desenvolvimento e testes de grandes sistemas complexos é que as peças desenvolvidas independentemente nem sempre funcionam em conjunto de acordo com o planejado. Você está frito se esperar até que todas as peças estejam prontas *para* realizar a integração e o teste, e encontrar problemas na integração — porque provavelmente já ultrapassou toda a sua programação e orçamento.



Este provavelmente é o principal fator que causa atrasos na programação e estouro de custos: descoberta tardia de problemas com a integração no processo de desenvolvimento. É um *grande* fator de risco para o sucesso do programa.

E o que você pode fazer? Os engenheiros de sistemas elaboraram duas abordagens principais para minimizar este risco:

- Verificar as interfaces e interações dos principais subsistemas e componentes no *início* com a substituição de alguns (ou até todos) subsistemas e componentes por modelos e simulações
- Integrar as peças gradativamente (ou por iteração) e ver como elas funcionam, e não esperar até que tudo esteja pronto para iniciar a integração e encontrar os problemas



A primeira abordagem, a modelagem do sistema (veja Capítulo 4), pode ser de grande ajuda. Os modelos de utilização, expressados como diagramas de atividades e diagramas de sequências, ajudam a identificar quais peças do sistema falam entre si e quais interfaces são necessárias entre elas. Com este conhecimento, é possível fazer com que as peças “falem” entre si para uma integração antecipada — reduzindo a simulação e emulação necessárias. Ao forçar os times dos subsistemas a trabalharem em conjunto assim que possível podemos identificar as questões logo no início do processo do design — quando é mais fácil e mais econômico corrigi-los.

Com os modelos também é possível testar o seu entendimento das interações dos subsistemas antes que sejam construídos — com a execução de modelos e a confirmação de que o que você pensou realmente vai acontecer. A transformação do pensamento em diagrama de execução pode revelar discrepâncias em uma etapa onde seja mais simples e mais barato consertar os problemas.

Com modelos que simulam a operação dos componentes que ainda não foram construídos, é possível testar as interfaces muito antes de vincular todas as peças de aço e circuitos do sistema. Pense em um simulador de voo, por exemplo, que permite que os engenheiros testem os aspectos da operação de uma aeronave antes mesmo que ela decole.

A segunda abordagem é a incorporação da maior quantidade possível de integração e testes (iteração) progressivos no ciclo do desenvolvimento. Os engenheiros de software usam ciclos de desenvolvimento iterativos há décadas. É claro que é mais fácil usar a iteração no desenvolvimento de software do que, por exemplo, na construção de componentes eletrônicos ou da fuselagem (porque normalmente o hardware exige um prazo de entrega mais longo), mas as mesmas ideias podem ser aplicadas.

Os ciclos de iteração antecipados podem identificar as peças de alto risco do produto com a integração de uma combinação de protótipos ou hardware de simulação com o software recém-desenvolvido. Com todas estas criações preliminares do sistema, porém executáveis, é possível resolver os problemas de integração, aperfeiçoar as interfaces, e confirmar as opções do estudo de compensação.

Alguns engenheiros de sistemas com pensamento moderno sugerem que a integração deve ser tentada antes dos testes dos componentes individuais, como parte do subsistema integrado. Embora possa parecer um contra-senso, na realidade, isso identifica antecipadamente os riscos antes de esperar a integração do sistema para que as coisas quebrem.

Capítulo 6

Capacitação da colaboração e gerenciamento das mudanças por equipes grandes

Neste capítulo

- ▶ Em busca de um ponto comum para interações humanas eficazes
- ▶ Automação dos processos do trabalho
- ▶ Vinculação das ferramentas e dos dados do ciclo de vida

Equipes pequenas e mais íntimas de desenvolvimento de produto sabem realmente como trabalhar em conjunto com eficácia: eles compartilham a informação essencial, usam as mesmas ferramentas de desenvolvimento e compartilham a mudança de um requisito, descoberta de um defeito ou até fazem uma festa!

Multiplique o tamanho da equipe por, vejamos, 100, atribua uma lista de desejos de aproximadamente 700 requisitos, informe que eles têm seis meses para construir o produto — e diga adeus ao seu trabalho (e todos os convites para festas).

Como é possível capturar a mágica que existe nos pequenos times de desenvolvimento e adaptá-la para grandes times de desenvolvimento? Este capítulo mostra como.

Facebook

Temos muito o que aprender com a exploração deste site de rede social de tanto sucesso, o Facebook. Lançado em 2004 quando a maioria dos donos de computadores já tinha uma conta de e-mail, o Facebook decolou

imediatamente e, no início de 2011, se orgulhava de ter 600 milhões de usuários. E por que o Facebook é tão popular?

Os fundadores do Facebook descobriram uma maneira de desenvolver uma plataforma em tempo real fácil de usar para relacionamento social organizada ao redor de interações sociais típicas com amigos, parentes, e outras pessoas com interesses semelhantes. Esta plataforma proporciona uma interface fácil de usar para compartilhar novidades, fotos, curtir, não curtir, status de relacionamento e outras informações. Ao invés de forçar a comunicação das pessoas em um formato tecnológico como o e-mail, por exemplo, o Facebook usa a tecnologia como uma plataforma para dar suporte a uma comunicação voltada para as pessoas.

Agora, imagine a aplicação da “mudança de paradigma do Facebook” no mundo da engenharia de sistemas. Imagine uma plataforma baseada na tecnologia que capitaliza a maneira como as equipes de desenvolvimento, engenheiros e interessados interagem. Como o Facebook, esta plataforma aproveitaria a conexão com a Internet para unir as pessoas — tornando uma equipe grande e dispersa tão eficaz quanto uma equipe pequena que estivesse trabalhando em uma mesma sala.

Como todos podem chegar a um acordo

Uma engenharia brilhante não é suficiente para criar um produto inteligente que venha a ter sucesso no mercado. As pesquisas mostram que um terço de todos os dispositivos produzidos não cumprem os requisitos de performance ou da funcionalidade e que 24 porcento de todos os projetos são cancelados devido aos atrasos irrecuperáveis. Muitas vezes o motivo para uma falha tão catastrófica do sistema não está relacionado com o design da engenharia do sistema, e sim com as falhas do conhecimento ou da comunicação.

Por isso, não é de se estranhar que os trabalhos de desenvolvimento de sistemas sofram com a má comunicação. A maioria das grandes equipes de desenvolvimento está espalhada pelas cidades, empresas e países. As barreiras do idioma e da cultura dificultam a comunicação, e o fuso horário normalmente atrapalha a colaboração. Até mesmo para os funcionários de uma mesma empresa, os silos organizacionais podem impedir a comunicação, reduzir a produtividade e propagar o “jogo da acusação”.

A má comunicação pode causar muitos problemas, como:

- ✓ Falta de clareza das metas do sistema
- ✓ Múltiplas interpretações dos requisitos do sistema
- ✓ Requisitos incompletos ou ignorados
- ✓ Perda de tempo com a obtenção manual de informação de múltiplas fontes
- ✓ Equipes que trabalham com documentos desatualizados
- ✓ Diferenças ou redundâncias das responsabilidades

Para aumentar ainda mais estes problemas, as equipes de desenvolvimento estão sendo muito pressionados para aumentar a produtividade — mesmo com o aumento da complexidade do sistema. E para completar, os produtos inteligentes com enormes quantidades de software exigem mais documentações e a curva de aprendizado dos novos membros da equipe é muito acentuada.



A melhor maneira de superar a dificuldade da comunicação natural de uma equipe grande é proporcionar uma fundação comum para o desenvolvimento, estabelecimento e a manutenção de um idioma comum para a comunicação nesta fundação.

Estabelecimento da fundação

Muitos dos sistemas inteligentes de hoje contêm diversos subsistemas de várias fontes e milhões de linhas de código — desenvolvidos pela equipe de engenharia de diversas empresas, países e culturas. Pense em uma aeronave moderna — a fuselagem é construída em um país por uma empresa, o motor por outra, os aviônicos por terceiros, e o software de integração por outra! Para simplificar os processos de desenvolvimento e de testes, é essencial oferecer uma plataforma unificada para o desenvolvimento dos sistemas.

O uso de uma plataforma de desenvolvimento comum elimina as barreiras das equipes, e permite que os engenheiros trabalhem em conjunto durante o ciclo do desenvolvimento. Uma plataforma unificada facilita a integração do trabalho e do compartilhamento do conhecimento das equipes distribuídas, reduzindo o tempo precioso do ciclo do desenvolvimento. Com a redução da má comunicação e a simplificação dos fluxos de trabalho, podemos esperar uma melhora substancial da qualidade — e uma maior satisfação da equipe. Além disso, os

engenheiros podem compartilhar o status do projeto com todos da equipe com quadros de gerenciamento, para que os gerentes de desenvolvimento mantenham o projeto na linha.

Como falar o mesmo idioma

Não existe nada melhor para unir diversas equipes de desenvolvimentos com várias culturas e disciplinas de engenharia do que adotar um design voltado para modelagem baseado em um idioma comum independente do domínio. Ao fornecer uma referência visual para o design do sistema a modelagem elimina as barreiras do idioma e facilita para que todos da equipe de desenvolvimento entendam o sistema e compartilhem do conhecimento das funções. E uma compreensão compartilhada se traduz diretamente em uma melhor produtividade, pois os engenheiros não perdem mais tempo resolvendo mal-entendidos e com o retrabalho do projeto.

A Linguagem de Modelagem de Sistemas (SysML) está emergindo como o padrão aceito para o desenvolvimento dos sistemas voltados para a modelagem. Ela dá suporte a todas as fases do desenvolvimento do sistema, incluindo a especificação dos requisitos, análise e design do sistema, verificação e validação dos sistemas que consistem no hardware, software, dados, pessoal, e até mesmo locais. Ela também tem a vantagem de ser completamente compatível com a Linguagem de Modelagem Unificada (UML) que facilita a passagem dos modelos do ponto de vista do sistema para o software.

São muitas as ferramentas de software disponíveis comercialmente que dão suporte ao desenvolvimento com a SysML, e cada uma delas tem o seu próprio ambiente de desenvolvimento. Para facilitar a colaboração eficaz, os times devem padronizar uma plataforma de trabalho comum com as ferramentas mais eficazes disponíveis no mercado.

Além do e-mail e do compartilhamento dos documentos

O estabelecimento de uma plataforma de desenvolvimento comum é um passo gigantesco na direção certa para um trabalho de equipe eficaz — mas isso

não é suficiente. Quando um membro da equipe de desenvolvimento faz uma mudança (por exemplo, de um modelo do requisito ou do sistema), e isso não é comunicado imediatamente para o resto da equipe, o resultado é o caos.

Acompanhamento das mudanças

Durante a engenharia de um sistema, a informação essencial sempre muda — de acordo com o projeto. A engenharia de sistemas envolve (entre outras coisas) o design iterativo e os processos de teste: Desenho do sistema, desenvolvimento dos modelos, teste dos modelos, repetição dos desenhos para a correção dos defeitos, e assim por diante. Por isso, os modelos, resultados dos testes e as outras informações são sempre atualizados. Os requisitos também podem mudar, pois o mercado e os negócios precisam evoluir.

Tradicionalmente, as grandes equipes de engenharia contam com a documentação em texto como a fonte de toda a informação do projeto. Os arquivos cheios de documentos dos requisitos, documentos de arquitetura de alto nível, documentos do design, e outros documentos normalmente servem como a fundação de todo o conhecimento do sistema. Mas os documentos são limitados pois simplesmente *registram* a informação e não permitem uma fácil mudança a informação. Você não deve contar exclusivamente com a documentação pois ela se torna obsoleta antes mesmo de ser aprovada!

As equipes de desenvolvimento de sistemas precisam de um mecanismo consistente e flexível para obter a informação essencial e facilitar a atualização, sem perder a integridade.



Comunicação das mudanças

O método tradicional para a criação de algumas dezenas de documentos essenciais e a troca de informação por e-mail não funcionam bem nos ambientes de desenvolvimento complexos de hoje em dia. Com as centenas ou até mesmo milhares de requisitos dos sistemas complexos, a troca de e-mails apenas cria caos e confusão.



As equipes de desenvolvimento de sistemas precisam de um veículo que facilite a troca eficaz entre os membros da equipe. Uma plataforma robusta para o compartilhamento da informação é a única maneira de evitar problemas que podem ser causados quando diversas versões dos documentos essenciais são espalhadas por todos os lugares.

Escolha do que deve ser compartilhado

Se você tivesse que sentar e fazer uma lista dos diferentes tipos de informações essenciais que devem ser administradas para o desenvolvimento de um sistema complexo, provavelmente nunca chegaria ao fim. Por isso, ao escolher qual informação deve ser compartilhada com todos da equipe de desenvolvimento, cuidado para evitar colocar na lista todas as peças dos dados que descrevem o sistema.



O objetivo é facilitar o compartilhamento e a colaboração dos membros de uma equipe de desenvolvimento de sistemas, e não sobrecarregar a equipe com informações supérfluas. Faça uma lista com a informação mínima que deve ser compartilhada para facilitar a colaboração, como por exemplo:

- ✓ Prioridades do desenvolvimento
- ✓ Aprovações do projeto
- ✓ Agendas
- ✓ Funções e responsabilidades dos funcionários
- ✓ Requisitos
- ✓ Alteração de pedidos
- ✓ Modelos conceituais
- ✓ Casos de utilização
- ✓ Planos de teste
- ✓ Defeitos
- ✓ Assuntos essenciais
- ✓ Dados de rastreamento
- ✓ Informação do orçamento
- ✓ Informação de aquisições

Maneiras de facilitar o compartilhamento

Não é realista esperar que cada uma das potenciais centenas de empresas que participam do processo de desenvolvimento de produto usem exatamente o mesmo conjunto de ferramentas do mesmo fornecedor. Por isso, é preciso haver uma forma para que todos da equipe virtual possam compartilhar os dados de desenvolvimento — sem importar que equipe ou parceiro que os criou.



Uma colaboração eficaz inicia com uma plataforma no qual o processo de desenvolvimento geral pode ser administrado e controlado. Com esta plataforma, os interessados e engenheiros processam e produzem dados que são compartilhados, analisados e relatados com eficiência durante o ciclo de desenvolvimento dos sistemas.

As ferramentas de automação e de federação que aproveitam a tecnologia para simplificar a comunicação e a automação dos fluxos de trabalho, são o próximo grande desenvolvimento da colaboração.

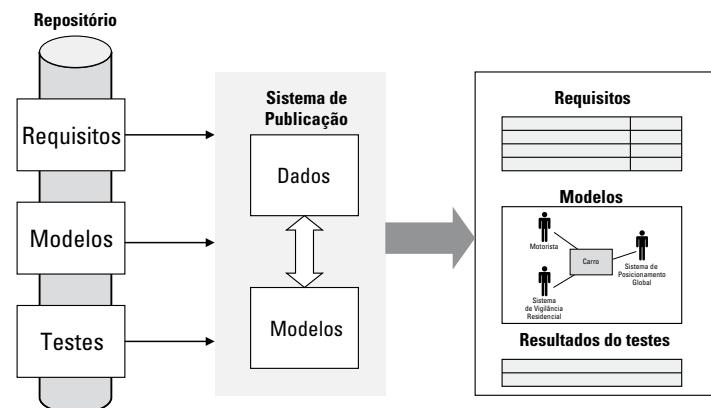


Figura 6-1: Um repositório virtual da informação do design pode ajudar a aumentar a colaboração.

Criação de um repositório virtual de informação ao vivo

As pessoas precisam ter acesso em tempo real à informação atualizada do projeto localizada em qualquer local e em qualquer formato de dados. Tradicionalmente os engenheiros de sistemas gastam bastante tempo para acompanhar a informação e garantir que todos estejam trabalhando com os mesmos documentos.



Com a organização da informação essencial em um repositório virtual cria-se uma fonte única de conhecimento dos aspectos essenciais do processo de desenvolvimento, para a automatização do processo do compartilhamento da informação.

Um repositório *virtual* não existe fisicamente, os dados são obtidos quando necessário dos locais em todo o mundo, de acordo com a necessidade. Por exemplo, os dados da engenharia mecânica podem ser obtidos de um repositório de dados físicos administrados pelo time de engenharia mecânica de Seattle, enquanto que os dados de engenharia elétrica podem ser obtidos de um banco de dados de engenharia elétrica de Tóquio. Ninguém precisa saber (ou se importar) com o local onde os dados físicos estão localizados — desde que eles possam ser obtidos quando necessários.

A Figura 6-1 mostra um modelo conceitual deste repositório virtual e como ele pode ser usado. Um gerente de processo dos negócios atua como o cérebro da operação, administrando a execução das atividades de desenvolvimento. Ele pode acessar os dados de acordo com o que é necessário (e quem precisa) com os padrões abertos. Ao invés de passar documentos de textos dos gerentes de produto para os arquitetos, engenheiros de design, engenheiros de teste e assim por diante, todos os envolvidos no processo de desenvolvimento podem acessar os dados necessários de forma transparente.

Um repositório virtual como esse é uma fonte única de informações atualizadas que todos podem acessar de qualquer lugar. Além disso, ele permite a troca de informação e de ideias, e a captura das decisões e processos mentais — quase como se estivessem trabalhando no mesmo local.

Por que um repositório “virtual”?

Por que um repositório “virtual” e não um “real”? Bem, na realidade, são muitos os motivos porque isso é uma melhor ideia do que “enfiar tudo em um banco de dados”.

- ✓ Primeiro, a definição de um “banco de dados universal” que possa armazenar tudo o desejado ou necessário seria um trabalho gigantesco!
- ✓ Segundo, seria preciso dar suporte as ferramentas de todos os fornecedores que existam atualmente ou que venham a existir no futuro.
- ✓ E terceiro, um banco de dados maciço acessado de todo o

planeta teria uma boa performance para certas coisas e seria um desastre para outras, isso sem mencionar as atualizações para novas versões, pontos únicos de falhas (da rede ou do hardware), e muitos outros desafios de um único armazenamento físico dos dados.

Virtual é definitivamente melhor — distribuído, otimizado para os dados e performance de cada ferramenta, com atualização simples e fácil conexão com terceiros.

Aviso: Uma solução que “coloca tudo em um só local” pode prendê-lo a uma solução proprietária e a um fornecedor, limitando a sua opção como consumidor!

Simplificação da integração da ferramenta

Embora o compartilhamento dos recursos e ativos com um repositório virtual pareça ser excelente a princípio, na realidade, isso não é fácil. As ferramentas internas, projetos de fonte aberta e diversos fornecedores podem criar barreiras com os formatos de dados incompatíveis e outros fatores.

Nos últimos anos, os fornecedores de ferramentas populares de desenvolvimento de ciclos de sistemas, como os tipos de ferramentas relacionadas na Tabela 6-1, passaram a definir maneiras para facilitar a integração das ferramentas de ciclo dos sistemas. Com a criação de uma comunidade conhecida como Open Services for Lifecycle Collaboration (OSLC), estas empresas se dedicam à promoção de novas formas de colaboração através da eliminação das barreiras das ferramentas.

Inspirada pela arquitetura da Internet, a OSLC especifica um conjunto de padrões ligeiramente agrupados, formatos de recursos comuns, e serviços criados para facilitar o compartilhamento dos recursos do sistema. A OSLC facilita o uso conjunto das ferramentas de qualquer fornecedor, e simplifica o compartilhamento dos dados do ciclo do sistema, como os requisitos, solicitações de alteração, casos de teste e defeitos.

Tabela 6-1 Principais ferramentas da engenharia de sistemas

Tipo de ferramenta	Principais capacidades
Gerenciamento e rastreamento dos requisitos	Rastreamento completo ao vivo dos requisitos da origem, missão, e sistema/subsistema
Desenvolvimento de sistemas baseados em modelagem	Requisitos do sistema, funcionalidade do sistema, realização, estudos de compensação, execução e validação
Gerenciamento da mudança e da configuração	Gerenciamento da colaboração, mudança, repositório compartilhado e configuração
Geração automatizada da documentação	Geração de documentos de requisitos, design e especificação
Engenharia integrada de sistemas e de software	Desenvolvimento integrado do fluxo descendente dos requisitos e modelos

Automação da produção dos documentos

Até mesmo em um modelo de desenvolvimento de modelagem a documentação e os relatórios são necessários para as obrigações contratuais, conformidade, revisões técnicas e administração do projeto. A documentação normalmente envolve trabalho físico, como a execução de um diagrama de uma ferramenta de modelagem, a captura de tela e a colagem no documento. Além disso, os engenheiros gastam bastante tempo na coleta da informação de diversas fontes, para a obtenção, resumo e reformulação da informação mais recente para a produção de um relatório personalizado.



As ferramentas de automação de documentos podem simplificar a produção de relatórios personalizados, garantindo a consistência da informação em cada relatório.

As ferramentas de automação permitem o acesso ao repositório central da informação, identificação e seleção da informação necessária, e pedido de relatório. Além de simples, este processo também facilita a reutilização e a consistência da informação. Além disso, é possível consolidar a informação essencial de diversas fontes — até mesmo produtos de diversos fornecedores — em um único relatório.

Certamente isto economiza tempo com a produção dos documentos, mas a grande vantagem é quando alguma coisa muda. É só fazer as mudanças necessárias dos requisitos e modelos, apertar um botão, e pronto — os documentos estão revisados.

Capítulo 7

Dez maneiras de ganhar com a Engenharia de Sistemas

Neste capítulo

- ▶ Correção das falhas do design no início do processo de desenvolvimento
- ▶ Desenvolvimento de modelos de negócios flexíveis
- ▶ Controle do software integrado complexo
- ▶ Melhoria do gerenciamento dos requisitos
- ▶ Reutilização do código para acelerar o lançamento dos produtos
- ▶ Uso de ferramentas de colaboração
- ▶ Lançamento de soluções complexas no mercado dentro do prazo
- ▶ Uso de uma plataforma de desenvolvimento integrada
- ▶ Teste logo, teste frequentemente
- ▶ Compartilhamento dos requisitos para maior eficácia

Aengenharia de sistemas oferece uma vantagem competitiva necessária para o sucesso no desenvolvimento de produtos inteligentes que ofereçam um valor tangível para os clientes (e para o resultado final). A engenharia de sistemas como parte do processos de negócios centrais aumenta a probabilidade da produção de produtos que as pessoas realmente queiram, com menos defeitos caros, e maior capacidade de responder à dinâmica do mercado.

Neste capítulo, veremos dez exemplos de empresas que adotaram as melhores práticas da engenharia de sistemas — e que obtiveram resultados reais e mensuráveis.

Correção das falhas do design antes que se tornem viral



Um dos grandes desafios da engenharia de produtos inteligentes é identificar os defeitos no início do processo de desenvolvimento. A maioria dos defeitos surge durante o processo de design, mas só é identificada na fase de testes, ou até mesmo depois do produto entrar na fase de produção.

Nos produtos caros de produção em massa, a não descoberta das falhas de design pode custar muito caro, e nos produtos de defesa produzidos em massa, os defeitos não detectados também podem ser perigosos. Para complicar ainda mais as coisas, a maioria dos sistemas de defesa é composta de múltiplos subsistemas complexos projetados e construídos por diversos subcontratados aprovados.

Como subcontratada do Departamento de Defesa dos Estados Unidos, a Brockwell Technologies de Huntsville, Alabama, cria aplicativos de sistemas integrados de armamento em tempo real e executa diagnósticos integrados de veículos militares. O desenvolvimento de sistemas de armamentos interconectados apresenta um desafio único: como fazer modificações em um sistema sem afetar os outros sistemas.

Para reduzir a possibilidade de introdução de defeitos com a interação dos subsistemas, a Brockwell Technologies incorporou um design e testes do sistema baseados em modelagem nos seus processos de negócios. Com a modelagem da estrutura e do comportamento do sistema, os engenheiros da Brockwell criam protótipos de sistemas complexos e virtualizam o seu funcionamento. Com a técnica de modelagem preditiva os engenheiros podem identificar as falhas do design no início do processo de desenvolvimento — antes que os sistemas sejam produzidos em massa.

O investimento da Brockwell na engenharia de sistemas é vantajoso tanto para o Departamento de Defesa quanto para a Brockwell: A Brockwell melhorou a confiabilidade e a segurança dos seus sistemas de armamentos e reduziu em 40 porcento o tempo para a colocação do produto no mercado.

Engenharia de um modelo de negócios flexível

Se estiver planejando iniciar um novo negócio ou entrar para um novo mercado, a coisa mais inteligente a ser feita é desenvolver uma infraestrutura de negócios flexível projetada para ser um exemplo para a engenharia de sistemas. E foi exatamente isso que uma empresa de Atlanta fez quando entrou no mercado quente da telemática há alguns anos.

A empresa viu uma oportunidade de passar à frente dos veteranos do setor com o estabelecimento de uma infraestrutura de negócios que permitisse a adaptação rápida aos relacionamentos de negócios e modelos de entrega de serviço. Com uma estrutura de trabalho flexível do processo criada com uma tecnologia aberta, a empresa concluiu que poderia entrar em novas oportunidades de serviço antes que as provedoras de telemáticas existentes — com a sua tecnologia própria e modelos rígidos de negócios — pudessem reagir.

A empresa projetou todos os processos centrais dos negócios — incluindo os da linha de frente, de retaguarda, e as operações — com um objetivo em mente: o lançamento rápido de novos serviços. Além disso, para simplificar o desenvolvimento dos novos serviços, a empresa instalou uma plataforma comum de desenvolvimento de sistemas e um conjunto de ferramentas de colaboração. A infraestrutura de desenvolvimento de sistemas da empresa permitiu o gerenciamento do trabalho dos produtos e das entregas, armazenagem das entregas em um repositório central, a colaboração das equipes espalhadas pelo mundo, manutenção do controle da versão e rápida comunicação das atualizações.

Com a combinação dos sistemas abertos, processos flexíveis e colaboração aprimorada, a empresa consegue lançar novos produtos no mercado em menos de 30 dias. Com o ponto focal das telemáticas mudando da própria tecnologia para serviços inovadores que usam de forma diferente os dados do veículo, a empresa realmente fica bem mais posicionada para ganhar a competição.

Controle do software integrado complexo

As empresas inteligentes sabem que, quando o produto é apenas parte de uma solução de sistema de sistemas maior, a garantia da qualidade

consistente do produto é de suma importância. Afinal de contas, ninguém quer que a sua empresa seja conhecida como o “elo fraco” do sistema. Mas, com o crescimento do tamanho e da complexidade dos softwares integrados, fica cada vez mais difícil garantir a qualidade.

Durante muito tempo, o desenvolvimento manual de software foi o ganha-pão de uma provedora de serviço de tecnologia alemã especializada em medição e controle da engenharia de tecnologia e processo. Mas quando decidiu desenvolver um software integrado complexo para sistemas que gerenciam e controlam de forma remota sistemas fotovoltaicos, a empresa descobriu que teria que adotar um novo ambiente de software.

Com quatro objetivos específicos em mente — reduzir os defeitos do produto, melhorar o rastreamento, aumentar a reutilização dos módulos de software, e garantir um produto com qualidade consistente — a empresa selecionou uma plataforma que usa o desenvolvimento de modelos na engenharia de sistemas em tempo real e integrada. O novo sistema permite que a empresa identifique e conserte os problemas no início com os testes de modelos durante a fase de design — garantindo um software com alta qualidade e consistente. Como bônus, a empresa agora pode criar módulos de código fonte e subsistemas reutilizáveis, dando para a empresa uma vantagem sobre a concorrência.

Melhora da eficiência com requisitos consistentes

Para criar o produto certo — e construir o produto certo — é preciso conhecer profundamente as necessidades do cliente e dos interessados, e definir cuidadosamente os requisitos do sistema ao redor destas necessidades. Sem um gerenciamento eficaz dos requisitos, é muito fácil perder de vista os objetivos — e os seus clientes.

Com centenas de desenvolvedores espalhados pelo mundo usando diferentes ferramentas de gerenciamento de requisitos, uma empresa australiana estava tendo dificuldades com a ineficiência do desenvolvimento e o rastreamento inadequado dos requisitos. A gerência estava cada vez mais preocupada por não poder testar os requisitos de forma uniforme, pois cada equipe de desenvolvimento tinha seus próprios métodos de gerenciar os requisitos. E o pior de tudo era que a falta de consistência aumentava a possibilidade de

erros e tinha o potencial de prejudicar o relacionamento da empresa com o seu maior cliente, a Força de Defesa Australiana.

A organização implementou uma solução unificada para o gerenciamento dos requisitos projetada para dar suporte à análise dos requisitos em toda a empresa. Com o acesso universal ao repositório centralizado de requisitos, a solução elimina a confusão, o retrabalho caro e a duplicação do trabalho. E o mais importante, a solução facilita o rastreamento total dos requisitos — garantindo que o produto lançado atenda às necessidades do cliente.

Aproveitamento do código reutilizável para acelerar o lançamento do produto

Se o seu portfólio tem uma linha de produtos com recursos básicos semelhantes, provavelmente muitos códigos de software similares podem ser aproveitados. As empresas inteligentes estruturam seus códigos em módulos reutilizáveis para acelerar o desenvolvimento dos produtos futuros.

E foi exatamente isso que a Océ N.V. tinha em mente quando decidiu produzir a impressora de folhas soltas mais rápida do mundo. Líder do mercado de tecnologia e serviços de gerenciamento de documentos digitais, a Océ desenvolve aplicativos de software avançados que entregam documentos e dados através de redes internas e da Internet para dispositivos de impressão e arquivos locais e em todo o mundo.

Normalmente, a Océ codifica cada nova impressora do zero, mas quando assumiu a enorme tarefa de coordenar os códigos de 17 processadores distribuídos na nova impressora, a empresa decidiu examinar novamente seus processos de desenvolvimento.

A Océ utilizou ferramentas de desenvolvimento voltadas para modelos para decompor o sistema da impressora em subsistemas menores e projetados mais facilmente. Com isso, os desenvolvedores puderam fazer a modelagem de um conjunto de máquinas de estado finito simultâneas que foram codificadas em um conjunto de módulos reutilizáveis para diversos ambientes de destino. Agora, sempre que uma mudança é necessária, a Océ simplesmente atualiza os modelos e gera novamente o código — reduzindo o tempo de resposta para as solicitações de mudança.



Atualmente, mais de 50 porcento dos códigos de software da empresa são reutilizáveis, e isso traz uma enorme melhora na eficiência e na qualidade. Na realidade, a Océ até lançou um protótipo operável de uma nova impressora em apenas dois meses — seis meses antes do previsto.

Simplificação do desenvolvimento com as ferramentas de colaboração

As empresas que desenvolvem produtos complexos, inteligentes e altamente instrumentados sabem que o sucesso depende tanto do gerenciamento do trabalho técnico quanto à excelência da engenharia. Sem uma disciplina a nível do sistema para coordenar os trabalhos de diversos grupos de engenharia, os produtos complexos estão fadados a se tornar famosos fracassos.

Quando a General Motors (GM) resolveu criar o Chevy Volt, trabalhou intensamente em estabelecer as melhores práticas para a engenharia de sistemas. A GM examinou as práticas de desenvolvimento e as práticas de gerenciamento do trabalho técnico visando melhorar as duas.

Tradicionalmente, os engenheiros de sistemas da GM gastam grande parte do seu tempo conferindo as cargas de trabalho dos desenvolvedores e garantir que todos tenham a mesma versão dos requisitos e dos outros documentos. Para liberar os engenheiros para que pudessem se concentrar na funcionalidade e na qualidade, a GM decidiu utilizar uma ferramenta comercial para coordenar os times de desenvolvimento e gerenciar uma única versão dos requisitos e de outros documentos.

A GM então implementou práticas de desenvolvimento de sistemas baseadas em modelagem para ajudar o gerenciamento da complexidade (o Volt executa aproximadamente 10 milhões de linhas de códigos). Os desenvolvedores da GM usaram modelos para visualizar as interações dos sistemas integrados e executar simulações para testar os modelos.

A combinação das ferramentas de colaboração com o desenvolvimento de sistemas baseados em modelagem valeu a pena: A GM concluiu o desenvolvimento do Volt em apenas 29 meses — um recorde para a GM, que normalmente desenvolve um novo carro em até cinco anos ou mais.

Entrega de soluções complexas no mercado dentro do prazo

Para ter sucesso no mundo ultra-competitivo de hoje, as empresas que desenvolvem sistemas complexos extremamente grandes sabem que precisam ser espertos ao focar o processo de desenvolvimento nos requisitos. A capacidade de capturar e de gerenciar os requisitos pode fazer a diferença entre ganhar ou perder um grande contrato.

Uma empresa de defesa queria reduzir os riscos do lançamento de soluções de sistema de sistemas complexas de muitos milhões de dólares no mercado dentro do prazo. A empresa teve uma ideia inovadora: reunir o gerenciamento dos requisitos e a estrutura de trabalho da arquitetura da empresa de forma a permitir que a empresa lançasse os sistemas complexos baseados em requisitos dentro do prazo.

A empresa baseou sua solução nos produtos disponíveis comercialmente para o gerenciamento dos requisitos e o planejamento da arquitetura do sistema. Agora, a empresa pode definir as soluções tecnológicas, mas também as soluções operacionais, técnicas, de treinamento e de sistemas — durante todo o ciclo. Com isso, ela lança sistemas grandes, complexos e de alta qualidade no mercado mais rápido do que nunca. Além disso, quando os requisitos do cliente mudam, a empresa pode mostrar o impacto das mudanças bem rapidamente.

Melhora da produtividade com uma plataforma de desenvolvimento integrada

As empresas que desenvolvem sistemas essenciais para a segurança normalmente precisam demonstrar que o seu processo de desenvolvimento está em conformidade com os padrões de segurança nacionais e internacionais. Quando o ambiente de desenvolvimento é fragmentado, fica difícil demonstrar esta conformidade.

Com um conjunto diverso de ferramentas de desenvolvimento e de times de desenvolvimento em dois grandes locais, foi necessário outra empresa australiana para fazer algumas mudanças para garantir o sucesso dos projetos futuros. Ela oferece soluções de sinalização e controle ferroviário essenciais para a segurança em todo o mundo, e seus clientes exigem compatibilidade com versões anteriores do equipamento ferroviário.

A empresa precisava de um ambiente de desenvolvimento integrado que promovesse a conformidade com padrões de segurança e confiabilidade global. A empresa implementou uma solução uniforme de gerenciamento dos requisitos e da configuração que atenderam aos requisitos do cliente e também viabilizaram a colaboração total das equipes de desenvolvimento em diversos locais.

Teste logo, teste frequentemente

A entrega de um protótipo ou de um modelo executável para as mãos dos interessados no início tem um valor incrível. O feedback do conceito no início do processo evita problemas mais tarde.



A execução de testes simples o mais cedo possível também é útil durante os testes finais. O acesso desde o início para as equipes de testes, mesmo quando eles sabem que o produto ainda não está pronto ou é apenas um protótipo, também ajuda a melhorar os planos e procedimentos dos testes do mesmo modo que ajuda os designers verificarem suas ideias.

Compartilhamento dos requisitos para a economia de tempo e dinheiro

Normalmente as grandes organizações de desenvolvimento têm duas ou mais equipes localizadas em áreas diferentes que trabalham em versões paralelas com requisitos compartilhados. Poderia-se economizar muito tempo e dinheiro se essas equipes tivessem um mecanismo para o compartilhamento dos requisitos.

Uma empresa de Michigan é uma fornecedora global líder de sistemas eletrônicos móveis e de transporte. Um dos objetivos da empresa era melhorar a comunicação das equipes de desenvolvimento globais para que eles pudessem ser mais produtivos durante o trabalho nas versões paralelas com os requisitos compartilhados.

A implementação de uma ferramenta de gerenciamento de requisitos facilita o compartilhamento dos requisitos pelos desenvolvedores da empresa. Eles podem importar os requisitos de um repositório para que possam ser reutilizados em um novo projeto, evitando a duplicação desnecessária do trabalho, economizando tempo e reduzindo os custos do desenvolvimento. A ferramenta de gerenciamento dos requisitos promove a consistência e aprimora a colaboração, permitindo que a empresa entregue produtos de maior qualidade em menos tempo.

Entendimento do problema que a engenharia de sistemas tenta resolver

A engenharia de sistemas é uma abordagem interdisciplinar de criação de sistemas maiores e complexos que atendem a um conjunto definido de requisitos comerciais e técnicos. As indústrias aeroespacial e de defesa usam a engenharia de sistemas há muito tempo e muito do que aprenderam está sendo aplicado em outras indústrias. Os carros, telefones e TVs mais inteligentes exigem métodos espaciais para a sua construção. Com a *Engenharia de Sistemas Para Leigos*, IBM Edição Limitada, você entende os princípios do tópico e investiga maneiras para resolver os problemas e criar melhores negócios.

- *O que são os produtos inteligentes — porque eles justificam uma nova abordagem ao desenvolvimento de sistemas.*
- *Entendimento do processo — do desenvolvimento de sistemas — uma visão de alto nível da engenharia de sistemas*
- *A modelagem — como aprimorar o seu entendimento da estrutura e do comportamento do sistema*
- *Colaboração aprimorada — união dos times de desenvolvimento*



Abra o livro e encontre:

- Empresas reais que se beneficiam com a engenharia de sistemas
- Como criar o sistema certo com validação e verificação
- Como gerar produtos inteligentes

Making Everything Easier!™

Visite [Dummies.com®](https://www.dummies.com)
para vídeos, exemplos
passo a passo, artigos com
instruções, ou para comprar!

Para Leigos®
A Branded Imprint of
 WILEY

ISBN: 978-1-118-37062-9
Não para revenda