

Engenharia de Software na Prática

Hélio Engholm Jr.

Novatec

CAPÍTULO 1

Desenvolvimento de software para o valor de negócios

Com base na importância cada vez maior do software no dia-a-dia das empresas, devemos nos preocupar com a maneira com que ele agrega valor aos negócios das mesmas, aumentando a produtividade e diminuindo custos. Desse modo, este capítulo introdutório apresenta o valor da tecnologia da informação para as empresas e a importância de utilizarmos processos adequados e engenharia de software na produção de sistemas com qualidade e menores custos, tanto de desenvolvimento quanto de manutenção. Com a finalidade de atender a esses objetivos, a área de engenharia de software destina parte de sua atenção ao quesito qualidade na construção de software, utilizando a definição de modelos e processos para melhoria da qualidade e diminuição de custos no desenvolvimento e na manutenção de sistemas. Existem hoje em dia várias propostas de modelo buscando melhorar o processo de desenvolvimento de software e a qualidade envolvida.

1.1 Qualidade na construção de software

A qualidade contempla uma série de objetivos da construção de software, conhecidos como requisitos não-funcionais, tais como extensibilidade, capacidade de manutenção, reutilização de código, desempenho, escalabilidade, usabilidade e confiabilidade nos dados apresentados pela aplicação. Esses tipos de requisitos não-funcionais serão explicados no item 3.4 deste livro.

Podemos presenciar uma série de problemas enfrentados por empresas que investem no desenvolvimento de sistemas sem a utilização de engenharia de software, seja por desenvolvimento interno ou pela contratação de empresas que constroem sistemas. Para clarificar esses problemas, são apresentadas a

seguir as consequências práticas de desenvolver software de modo ad hoc, sem utilização de processos definidos, orientação a objetos e melhores práticas:

- Softwares difíceis de se dar manutenção, tanto corretiva quanto evolutiva. Será apresentado no item 1.8.3, Tipos de manutenção, uma estatística de mercado que mostra que os profissionais envolvidos gastam, em média, cerca de 50% do esforço somente tentando entender o código pronto.
- Softwares difíceis de se implementar alterações, tanto corretivas quanto evolutivas.
- Reutilização de código mal elaborado e sujeito a geração/propagação de erros em outras partes do sistema em desenvolvimento.
- Sistemas com baixo desempenho e escalabilidade inadequada.
- Baixa eficiência no desenvolvimento, com analistas desenvolvendo as mesmas funcionalidades diversas vezes.
- Falta de confiança nos dados apresentados pelo sistema, fazendo com que usuários deixem de utilizar o sistema por não confiar nas informações apresentadas.
- Baixa qualidade de código.

Em contrapartida aos efeitos do desenvolvimento ad hoc, temos o maior desejo de todas as empresas que desenvolvem software: ter menores custos e tempo de desenvolvimento nos processos de implementação e manutenção de sistemas. Esse desejo faz com que muitas empresas prefiram não utilizar processos de engenharia de software, pensando na falsa economia de tempo e diminuição de custos. Posso afirmar que essa ideia é falsa, pois a não utilização de métodos adequados no desenvolvimento gera softwares de má qualidade, que trazem frustração aos usuários, custos adicionais relacionados à manutenção corretiva e problemas na utilização do sistema. Desse modo, todo o tempo e dinheiro economizado no desenvolvimento será gasto em correções, trazendo prejuízos à imagem do projeto e ao próprio sistema.

Para termos uma ideia mais clara dos motivos que nos levam a utilizar a engenharia de software, é apresentado a seguir a armadilha associada à não utilização da mesma no desenvolvimento de sistemas e, em seguida, como a tecnologia da informação agrega valor ao negócio, ratificando a importância de métodos e procedimentos adequados no desenvolvimento de sistemas para as empresas.

1.2 A armadilha

Quem gerencia projetos, ou já contratou empresas para executá-los, sabe que uma das maneiras de diminuir custos e prazos de entrega é diminuindo a qualidade do produto. É justamente aqui que se arma uma grande armadilha. Os envolvidos podem não estar cientes das consequências geradas a partir do momento da entrega do sistema desenvolvido. Outro ponto de extrema importância é saber que, caso o sistema não seja amigável para uso, apresentando informações consideradas não confiáveis e com erros de funcionamento, nenhum usuário se interessará em utilizar o mesmo, levando o projeto ao fracasso e perdendo o investimento realizado.

Imagine o cenário de um sistema de cobrança bancária, para o qual você poderia estar participando de um processo de manutenção corretiva, sem ter a documentação do sistema e sem ter conhecimento das regras de negócio implementadas. Imagine também que essa manutenção está relacionada à correção de informações, que estão sendo geradas incorretamente. Agora, para finalizar, imagine que o sistema tem milhares de linhas de código e você é novo na equipe, contratado recentemente, sendo que os programadores que implementaram o sistema não estão mais na empresa.

Por mais que esse cenário seja assustador, podemos encontrá-lo nesse exato momento em diversos lugares enquanto você lê este parágrafo. Mas o que faltou no projeto de desenvolvimento desse sistema? Tecnicamente, poderia citar os seguintes exemplos:

- Engenharia de software e utilização de processos definidos.
- Utilização de paradigma de desenvolvimento orientado a objetos.
- Componentização.
- Utilização de padrões de design.
- Documentação adequada.

Do lado humano, poderia citar:

- Participação no projeto de profissionais com conhecimento dos itens técnicos apresentados.
- Falta de percepção do patrocinador do projeto do valor e dos benefícios de utilizar os itens técnicos relacionados.
- Falta de conhecimento dos envolvidos no projeto, das melhores práticas de mercado para desenvolvimento de software e de seus benefícios.

Já tive a oportunidade de presenciar empresários e executivos fazendo o seguinte comentário: “Prefiro o desenvolvimento de software sem utilizar orientação a objetos e processos de desenvolvimento, tipo CMMI, pois fica mais barato e mais rápido disponibilizar o sistema para os usuários”.

Pessoas que pensam assim provavelmente desconhecem o que será apresentado no item 1.8 deste livro chamado “Crise do legado”, no qual serão apresentadas informações que deixam evidente a grande armadilha de quem escolhe esse tipo de procedimento no desenvolvimento de software, que pode gerar as consequências listadas a seguir:

- Implantação de sistemas repletos de erros, constantemente entrando em manutenção e interrompendo o trabalho dos funcionários, o atendimento a clientes e/ou as funcionalidades disponibilizadas a clientes.
- Sistemas que podem não atender às expectativas dos stakeholders.
- Softwares com alto custo de manutenção.
- Softwares difíceis de serem utilizados, não atendendo às necessidades dos stakeholders.
- Grande insatisfação dos usuários.
- Prejuízos de imagem junto a clientes.
- Fracasso do projeto.

O desenvolvimento ad hoc de sistemas sem utilizar processos acaba custando muito mais caro, gerando insatisfação. Já presenciei várias vezes pessoas e empresas que simplesmente não querem e não utilizam sistemas disponibilizados por pelo menos um dos seguintes motivos:

- Não confiavam nos dados por eles apresentados.
- Porque não atendiam a suas expectativas e necessidades do dia-a-dia.
- Porque tinham usabilidade ruim.
- Porque viviam apresentando problemas de funcionamento.
- Porque costumavam ficar indisponíveis.

O detalhe mais triste é que as empresas detentoras desses sistemas investiram dinheiro e tentaram disponibilizar um sistema para melhorar a produtividade e diminuir custos. O resultado foi exatamente o contrário: além de não diminuir custos nem aumentar produtividade, as empresas perderam todo o investimento realizado, e o sistema caiu em desuso, literalmente, sendo jogado fora.

1.3 Valor da tecnologia da informação

Neste tópico será mostrado como a tecnologia da informação pode agregar valor ao negócio da empresa.

1.3.1 Gerenciando TI para o valor de negócio

Na hora de um executivo de determinada empresa defender investimentos em sistemas, seja desenvolvimento interno, externo ou a compra de um sistema pronto, ele necessita defender sua proposta. Pelo menos, os seguintes itens devem estar presentes:

- Contribuição e benefícios dados pelos investimentos propostos.
- Alinhamento da proposta com o negócio da empresa.
- Como construir ou contratar a área de TI para entregar e sustentar o software proposto.
- Custo total de propriedade.
- Retorno do investimento.

É justamente nos quesitos investimento em software, custo de propriedade e retorno de investimento que a engenharia de software tem muito a contribuir. Utilizando-se das melhores práticas de engenharia de software de mercado no desenvolvimento de software, temos menores custos na manutenção de sistemas, agilizando o retorno de investimentos e diminuindo o custo de propriedade caso o sistema disponibilizado não seja eficiente como o pretendido e apresente falhas constantes, não atendendo aos requisitos dos usuários e não agregando valor, situação em que será difícil para quem propôs o projeto mostrar as contribuições e benefícios que prometeu ao apresentar e defender o projeto. Como justificar o retorno de investimento realizado nesse tipo de projeto? Como defender que o projeto proposto está alinhado a qualquer negócio se ele gerou um sistema repleto de falhas, que não atende às expectativas dos envolvidos?

1.3.2 Metodologia do Índice de Valor de Negócio

Desenvolvido pela Intel para verificar o potencial de investimentos futuros, comparar opções e priorizar investimentos com base no ambiente de negócios corrente, esse índice considera os seguintes pontos:

- **Valor de negócio:** mede o impacto corporativo do projeto na estratégia da empresa e em suas prioridades.

- **Eficiência de TI:** mede como o investimento vai utilizar ou melhorar a infraestrutura existente.
- **Atratividade financeira:** mede o nível de investimento e o valor presente do projeto.

Analisando o que é considerado para o índice proposto pela Intel para softwares, além dos mesmos deverem estar alinhados a necessidades de negócio das empresas, eles devem ser desenvolvidos de forma eficiente, com o menor custo possível e menor custo total de propriedade, considerando neste os custos relacionados à manutenção dos mesmos após a implantação.

1.4 Relações entre o ciclo de vida do projeto e o ciclo de vida do produto

Muitos projetos estão ligados ao trabalho em andamento da organização executora. Algumas organizações aprovam formalmente os projetos somente após o término de um estudo de viabilidade, um plano preliminar ou alguma outra forma equivalente de análise. As motivações que criam o estímulo para um projeto são normalmente chamadas de problemas, oportunidades ou necessidades de negócios.

No estudo de caso apresentado neste livro, mostra-se um projeto de criação de comércio eletrônico de uma empresa que comercializa miniaturas de automóveis famosos, visando a prospecção e intensificação de clientes, além da divulgação da empresa e de seus produtos pela Internet.

É necessário ter cuidado para distinguir o ciclo de vida do projeto do ciclo de vida do produto. Por exemplo, um projeto de desenvolvimento de um novo sistema é apenas um aspecto do ciclo de vida do produto, o sistema a ser entregue.

O que não está claro para muitos profissionais e empresas consiste nos custos relacionados à manutenção dos sistemas pós-implantação. Dedicarei um item inteiro deste livro para mostrar claramente esses custos do produto de sistema, após implantação em produção. A importância disso é mostrar o retorno do investimento de se utilizar processos formais no desenvolvimento de sistemas. Em nosso estudo de caso, além dos custos de manutenção evolutiva do sistema on-line, teremos também custos para atualização do catálogo de produtos e administração do relacionamento com o cliente, entre outros.

1.4.1 Relação entre o produto e os ciclos de vida do projeto

A figura 1.1 apresenta o diagrama do ciclo de vida de um produto qualquer, mas que, em nosso contexto, deve ser considerado para um sistema de software.

Observe que depois de desenvolvido o produto, em nosso caso, um sistema de comércio eletrônico de miniaturas de automóveis desenvolvido e implantado em produção, o mesmo passa para a fase de operacionalização. É nessa fase que aparecem os custos de manutenção. A ISO/IEC 12207 [4] considera o desenvolvimento e a manutenção do software como processos que compõem o ciclo de vida de software.

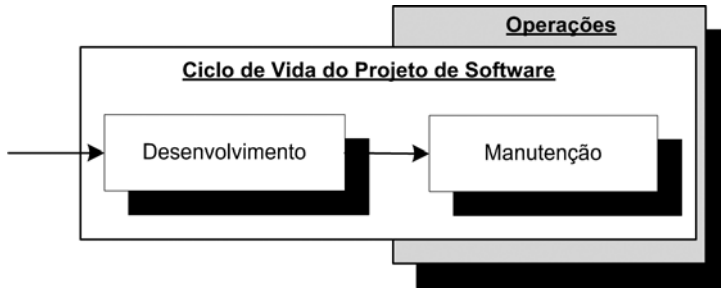


Figura 1.1 – Ciclo de vida do projeto de software.

1.4.2 Competências e capacidade

O negócio ganha vantagens competitivas quando inventa e implementa aplicações estratégicas mais rapidamente, com mais funcionalidades e com custos menores que seus concorrentes. Um dos objetivos da tecnologia da informação é diminuir os custos das empresas e aumentar a produtividade. Para isso, precisamos ter sistemas que aumentem a produtividade e não envolvam altos custos de manutenção utilizando engenharia de software.

1.4.3 Valor de TI

Pode ser descrita como envolvendo três processos:

- **Inovação:** imaginar a solução.
- **Entrega da solução:** construir a solução.
- **Provisionamento de serviço/produto:** entregar e dar suporte à solução.

Utilizando-se de processos propostos no mercado e na engenharia de software, podemos otimizar a construção e entrega e diminuir custos de suporte relacionados aos sistemas desenvolvidos. Como exercício, tente responder como você poderia atender a alguns dos processos propostos para agregar valor de TI na hora de propor nosso sistema de comércio eletrônico de miniaturas de automóveis.

1.4.4 Gerenciamento de TI

Podemos listar as seguintes preocupações do gerenciamento da tecnologia da informação dentro das empresas:

- Qual a aplicação da TI na empresa?
- Quais recursos de TI são compartilhados e qual será a sequência de investimentos futuros?
- Quanto a TI investirá e quando?
- Quais aplicações de negócio são necessárias e em qual prioridade?
- Quanto a TI pode gastar e quando?

Em geral, todas as empresas querem e precisam gastar o mínimo possível na aquisição e manutenção de sistemas, sendo que as empresas que desenvolvem software querem produzi-lo no menor custo e prazo. Novamente, a engenharia de software pode ajudar muito a atingir essas metas.

1.4.5 Estratégias para redução de custos

A seguir são apresentadas algumas estratégias para redução de custos na área de tecnologia da informação propostas pelas empresas:

- Renegociar com fornecedores.
- Trocar fornecedores.
- Ajustar acordos de nível de serviço.
- Realizar reengenharia de processos de negócios.
- Rastrear gastos.
- Considerar alternativas de outsourcing.
- Trocar recursos para localização de custos mais baixos.
- Substituir tecnologias/sistemas ultrapassados.
- Gerenciar portfólios de tecnologia da informação.
- Minimizar custos operacionais com a tecnologia da informação.

Dos pontos apresentados, podemos verificar aqueles alinhados aos objetivos da engenharia de software, a saber:

- Substituir tecnologias/sistemas ultrapassados, utilizando-se de outros mais alinhados aos requisitos não-funcionais do sistema.
- Minimizar custos operacionais com TI, utilizando sistemas com custos de manutenção menor.

1.5 Custo total de propriedade

O Gartner Group introduziu o conceito de custo total de propriedade como uma medida de aproximação para gerenciar e reduzir custos, mostrando que a manutenção é o maior componente de custo de soluções. Envolvendo pessoas, processos e tecnologia, suporte ao usuário final, suporte à TI, hardware e software, a avaliação desse custo é fundamental para o gerenciamento do valor de negócio representado pela tecnologia da informação.

É extremamente importante ressaltar esse custo, pois muitos empresários não têm uma noção clara do quanto custa manter um sistema de baixa qualidade em operação nem o custo para implementar manutenção evolutiva em sistemas mal desenvolvidos. Aliado a isso, temos também problemas de funcionários que não conseguem executar suas tarefas da maneira esperada e o custo das horas em que esses mesmos funcionários não estão produzindo por estarem aguardando o sistema voltar a operar corretamente, além de impactos na imagem da empresa quando clientes tentam utilizar sistemas com problemas.

Já presenciei uma empresa com legado composto por diversos sistemas sem documentação, baixa qualidade e que, frequentemente, apresentavam problemas de funcionamento. Além das paradas no negócio da empresa, que geravam perdas imensuráveis, relacionadas à sua imagem perante os clientes, a mesma periodicamente tinha que arcar com os custos relacionados a manutenções corretivas. Você continuaria utilizando uma operadora de telefonia celular que vivesse fora do ar? Para quantas pessoas você falaria mal dessa empresa, e qual seria o impacto disso nos negócios da companhia? Qual o prejuízo gerado?

O que já vi várias vezes, com a certeza de que verei novamente, são pessoas que acreditam que desenvolver software utilizando processos e o paradigma orientado a objetos não é necessário, e até que é perda de tempo. Simplesmente acham que não vale a pena gastar e esperar mais para ter seus sistemas desenvolvidos utilizando as melhores práticas de engenharia de software de mercado.

Para essas pessoas, devemos apresentar as informações do item sobre manutenção de software apresentado neste livro, no qual mostro os custos para manter softwares mal elaborados e estruturados. Na verdade, tudo o que se economiza desenvolvendo software ad hoc se gasta muitas vezes mais nos processos de manutenção dos sistemas, além de eventuais impactos de imagem junto ao cliente.

Outro detalhe é que, mesmo que o sistema esteja estável e sem apresentar erros, manutenções evolutivas são muito mais caras e demoradas, devido ao desenvolvimento caótico e desorganizado utilizado para disponibilizá-lo, geralmente sem documentação apropriada. Esta pode ser considerada a maior armadilha gerada pela falta de visibilidade de quem defende a preferência pela não utilização de processos e engenharia de software: achar que obterá vantagens quando de fato está programando armadilhas, custos futuros e insatisfações com o software de má qualidade.

1.6 Valor do investimento em software

Na hora de defender a aquisição ou desenvolvimento de um novo software, devemos ter pelo menos os seguintes objetivos em mente:

- Aumento da produtividade.
- Diminuição de custos operacionais.
- Diminuição de erros operacionais.
- Retorno do investimento.

O Retorno do Investimento (ROI – Return On Investment) representa o tempo necessário para recuperar os investimentos realizados em um projeto e para a obtenção de lucros. Desse modo, todo investimento deveria ter a análise de ROI elaborada para se ter uma ideia clara das vantagens e desvantagens de realizar o projeto.

1.6.1 Defesa da realização de um investimento em TI

Para defender investimentos em um projeto de software, podemos considerar pelo menos os seguintes itens:

- A aprovação de qualquer investimento TI deveria ter como base um caso de negócio.
- A avaliação quantitativa do caso de negócio é o retorno de investimento.
- ROI é utilizado para auxiliar nas decisões de investimento em TI.
- ROI é a métrica mais utilizada para justificar investimentos em TI.

O importante é visualizar que todo o investimento realizado em um sistema de software pode desaparecer se apresentarmos, no final, um sistema de baixa qualidade, com baixo desempenho, difícil de utilizar, que mostre dados não confiáveis e com alto custo de manutenção.

1.7 Processo de manutenção de software

Apresentarei agora algumas informações relacionadas ao custo da manutenção de software.

1.7.1 Introdução

Este item tem por objetivo passar uma ideia clara do processo de manutenção de software, mostrando principalmente os custos e dificuldades relacionados e como a engenharia de software pode colaborar para minimizar esses aspectos. Depois de fornecer detalhes sobre tipos de manutenção e os custos relacionados, será apresentado o conceito de crise do legado.

1.7.2 Tipos de manutenção

A manutenção de software pode estar associada a motivos diferentes, originando diferentes tipos de manutenção. Entre estes podemos listar:

- **Corretiva:** modificação do software com o objetivo de corrigir falhas.
- **Evolutiva:** relacionada à inclusão de novas funcionalidades ou ao desempenho.
- **Adaptativa:** relacionada à adaptação do software a ambientes operacionais diferentes.

Com base nos vários tipos de manutenção, podemos dizer que as manutenções estão associadas a pelo menos um dos itens a seguir:

- Remoção de defeitos.
- Adição de funcionalidades no sistema.
- Alteração de funcionalidades.
- Evolução das funcionalidades.
- Ajustes.
- Atualização.
- Melhora de desempenho.

1.7.3 Custo da manutenção

Existe uma série de motivos que influenciam os custos de manutenção de um sistema. Entre eles podemos enumerar os seguintes pontos como diretamente relacionados a esses custos:

- Documentação existente relacionada ao sistema e qualidade da mesma.
- Qualidade do software.
- Design do sistema.
- Paradigma de programação utilizado.
- Tamanho do software, relacionado à quantidade de linhas de código.
- Qualidade técnica da equipe responsável pela manutenção.
- Regras de negócio embarcadas no sistema.
- Plataforma de desenvolvimento.

Mostrarei resultados de estudos que comprovam que os envolvidos em manutenção gastam, em média, 50% do esforço tentando entender o código. Isso geralmente é resultado de código mal ou nada documentado, podendo ser também resultado da falta de utilização de engenharia de software e de documentação do sistema durante o desenvolvimento.

1.8 Crise do legado

De acordo com o Information Technology Research Institute, ELTIS-project, o custo relacionado à manutenção de software e ao gerenciamento de sua evolução representa atualmente cerca de 90% do custo total. Isso foi referido como crise do legado por Seacord e colaboradores (2003).

1.8.1 Custo proporcional relacionado à manutenção de software

Veja na tabela 1.1 uma série de informações relacionadas ao custo da manutenção de software fornecidos pelo próprio Information Technology Research Institute.

Tabela 1.1 – Custo proporcional de manutenção de software

Ano	Proporção de custos de manutenção de software	Definição
2000	>90%	Custo de software relacionado à manutenção e evolução de sistemas/custo total de software
1993	0,75	Manutenção de software/orçamento de sistemas de informação (em 1000 empresas)
1990	>90%	Custo de software relacionado à manutenção e evolução de sistemas/custo total de software
1990	60-70%	Manutenção de software/orçamento total de operação de sistemas da informação (Management Information Systems – MIS)

1988	60-70%	Manutenção de software/orçamento operacional do total de sistemas de informação gerencial (Management Information Systems – MIS)
1984	65-75%	Esforço gasto na manutenção de sistemas/total disponível relacionado a engenharia de software
1981	>50%	Tempo gasto pela equipe de manutenção/tempo total (em 487 organizações)
1979	0,67	Custos de manutenção/custo total de software

Observando os dados da tabela 1.1, podemos ter uma ideia da importância de termos softwares bem desenvolvidos e com qualidade, justificando os custos associados nos processos de desenvolvimento dos mesmos.

1.8.2 Custo absoluto relativo à manutenção de software

Podemos encontrar as seguintes informações relacionadas a custos de manutenção de software:

- O custo anual relacionado à manutenção de software nos estados Unidos está estimado em mais de US\$ 70 bilhões (Sutherland, 1995; Edelstein, 1993).
- Por exemplo: nos EUA, o governo federal gastou sozinho cerca de US\$ 8,38 bilhões durante o período de cinco anos devido ao bug do milênio.
- A Nokia Inc. utilizou cerca de US\$ 90 milhões relacionados a correções preventivas do bug do milênio.

1.8.3 Tipos de manutenção

Além das manutenções corretivas, estudos estimam que cerca de 75% dos custos de manutenção são relativos a fornecer evolução ao software, na forma de manutenção adaptativa e evolutiva (Martin, 1983; Nosek & Palvia, 1990; van Vliet, 2000). Esses estudos mostram também que cerca de 50% do tempo é gasto no processo de entendimento do código a ser mantido (Fjeldstad & Hamlen, 1983; Standish, 1984).

Essas informações comprovam o valor de sistemas bem implementados, documentados e fáceis de serem mantidos.

1.9 Crise do software

Nos anos 70, apareceu o termo crise do software, quando a engenharia de software praticamente inexistia. Esse termo estava relacionado às dificuldades

enfrentadas no desenvolvimento de software, inerentes ao aumento das demandas e da complexidade delas, aliado à inexistência de técnicas apropriadas para resolver esses desafios.

Imagine a complexidade de um software utilizado em aparelhos celulares com reconhecimento de voz usando inteligência artificial. No início muitos acreditavam nisso como ficção científica. Com a evolução, softwares de milhões de linhas de instrução apareceram, como sistemas operacionais de mercado conhecidos mundialmente. Além do aumento grandioso de complexidade, temos também o fenômeno da urgência em se desenvolver esses softwares, por exemplo, devido à necessidades de mercado.

Esses fatos fizeram aparecer o conceito de crise do software, podendo ser verificada por vários sintomas, tais como:

- Software de baixa qualidade.
- Projetos com prazos e custos maiores que os planejados.
- Software não atendendo aos requisitos dos stakeholders.
- Custos e dificuldades no processo de manutenção.

Analisando os sintomas expostos e o que vemos hoje em dia, podemos verificar que a crise ainda está presente. Mesmo dispondo de técnicas apropriadas na atualidade, ainda presenciaremos esses problemas em um futuro próximo.

Mesmo empresas que têm conhecimento das técnicas propostas às vezes não conseguem praticá-las por pressão do próprio cliente, por exemplo, em relação a prazos.

1.10 Problemas, expectativas e metas relacionados a software

Afinal de contas, quais são os problemas, expectativas e metas relacionados a software?

1.10.1 O problema

Na atualidade, encontramos uma crise de desenvolvimento de software que, além de englobar os sintomas relacionados à crise do software, está relacionada aos pontos a seguir:

- Grande insatisfação de clientes.
- Custos relativos ao desenvolvimento de sistemas aumentando, enquanto os relativos a hardware diminuem.

- Prazo de desenvolvimento de software tornando-se mais longo enquanto os custos de manutenção tornam-se maiores.
- Erros de software tornando-se mais frequentes, refletindo em custos de manutenção e de retrabalho.
- Processos estruturados de desenvolvimento sendo inflexíveis e ainda muito utilizados.
- Desenvolvimento de aplicativos/sistemas com alto custo, baixa qualidade e com demanda cada vez maior, aliados a uma complexidade também cada vez maior.
- Pressão das empresas clientes para entregas rápidas dos sistemas.

1.10.2 Expectativas

Com base no grande número de falhas apresentadas por softwares, custos e manutenções corretivas em sistemas, podemos enumerar as seguintes expectativas relacionadas ao desenvolvimento de software:

- Software confiável, flexível e de fácil manutenção.
- Maior velocidade no desenvolvimento de software, acarretando diminuição dos custos e de tempo de entrega ao cliente.
- Reusabilidade.
- Diminuição do número de falhas nos aplicativos desenvolvidos, diminuindo os custos relacionados à manutenção.
- Portabilidade.
- Computação distribuída: comunicação distribuída, comunicação interprocessos, compartilhamento e replicação de dados e segurança.

Dessas expectativas, surge, entre outros, o paradigma orientado a objetos e a engenharia de software como ferramentas para a solução de problemas. Neste livro forneço uma série de conceitos e melhores práticas de mercado desenvolvidos para se alcançar essas expectativas e diminuir os sintomas da crise do software comentada no item 1.9 deste livro.

1.11 Engenharia de software

A engenharia de software surgiu com o objetivo de utilizar princípios de engenharia no desenvolvimento de software para aumentar a qualidade dos produtos oferecidos, diminuir os custos e riscos relacionados e criar processos

repetíveis e eficazes para serem utilizados nos ciclos de manutenção e desenvolvimento de software.

Este item apresenta uma série de conceitos iniciais que serão explorados nos próximos capítulos do livro.

1.11.1 Definição de engenharia de software

Segundo Friedrich Ludwig Bauer, engenharia de software é a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe eficientemente em máquinas reais. O próprio significado de engenharia já envolve os conceitos de criação, construção, análise, desenvolvimento e manutenção.

A engenharia de software se concentra nos aspectos práticos da produção de um sistema de software sendo criada na tentativa de tratar a crise do software apresentada no item 1.9, realizando tratamento de engenharia sistemático e controlado no desenvolvimento de softwares.

O estudo de caso deste livro irá mostrar, na prática, um projeto de desenvolvimento de software utilizando processos definidos pela engenharia de software, percorrendo todas as fases do ciclo de desenvolvimento de software, desde a requisição da área de negócio de uma empresa até a elaboração do plano de testes e a implantação do mesmo. A dinâmica será realizada por meio da utilização de vários processos envolvidos, junto com templates que poderão, se desejado, ser utilizados por você e sua empresa como ponto de partida em projetos reais.

1.11.2 Áreas de conhecimento

Segundo o SWEBOK (Software Engineering Body of Knowledge – Corpo de Conhecimento da Engenharia de Software), as áreas de conhecimento da engenharia de software são:

- Requisitos de software.
- Análise e desenho de software.
- Implementação de software.
- Teste de software.
- Manutenção de software.
- Gerência de configuração de software.

- Gerência de engenharia de software.
- Processos de engenharia de software.
- Ferramentas e métodos de engenharia de software.
- Qualidade de software.

Utilizando estudos de caso, mostrarei os processos de gerenciamento de requisitos, análise e design, implementação, testes, gerenciamento de mudanças e gerenciamento de riscos na prática, apresentando vários templates que podem ser utilizados por você em sua rotina de trabalho.

1.11.3 Metas de engenharia de software para utilização do Paradigma Orientado a Objetos (POO)

Na tentativa de se produzir softwares com qualidade e de fácil manutenção, algumas décadas atrás foi desenvolvido o Paradigma Orientado a Objetos para desenvolvimento de software. Pelo fato de esse paradigma estar embarcado na engenharia de software, podemos apresentar os seguintes itens como metas de engenharia de software relacionada à orientação a objetos:

- Aumento do encapsulamento, agrupando todas as peças de um objeto dentro de um pacote limpo.
- Aumento da ocultação de informações com o uso de um objeto, fazendo com que ele não saiba o que ocorre dentro de si (Caixa Preta).
- Aumento da reusabilidade de código.
- Diminuição dos custos e do tempo de desenvolvimento.
- Facilidade de manutenção, tanto corretiva quanto evolutiva.

1.11.4 Modelos de maturidade

Os modelos de maturidade surgiram para avaliar a qualidade dos processos de software aplicados em uma organização (empresa ou instituição). Entre os mais populares, temos o Capability Maturity Model Integration (CMMI), do Software Engineering Institute – SEI. Nesse modelo podemos entender a capacidade de um processo como a habilidade com que ele alcança o resultado desejado. Já um modelo tem como objetivo estabelecer, com base em estudos, históricos e conhecimento operacional, um conjunto de melhores práticas que devem ser utilizadas para um fim específico. O CMMI pode ser organizado de duas formas, a contínua e a estagiada.

Representação contínua

A representação contínua possibilita que a empresa utilize a ordem de melhoria que melhor atende seus objetivos de negócio. É caracterizado pelos seguintes Níveis de Capacidade (Capability Levels):

- Nível 0: incompleto (Ad hoc).
- Nível 1: executado (definido).
- Nível 2: gerenciado/gerido.
- Nível 3: definido.
- Nível 4: quantitativamente gerenciado/gerido quantitativamente.
- Nível 5: em otimização ou otimizado.

Como vantagens da representação contínua podemos relacionar:

- Fornece maior flexibilidade focando em áreas de processo específicas de acordo com metas e objetivos de negócio.
- Permite a comparação de áreas de processo entre diferentes organizações.
- Estrutura familiar para aqueles que estão migrando da comunidade de engenharia de sistemas.
- Foco bem-definido nos riscos específicos de cada área de processo.
- Estrutura compatível com o padrão ISO/IEC 15504.

Representação por estágios

Disponibiliza uma sequência predeterminada para melhoria baseada em estágios que não deve ser desconsiderada, pois cada estágio serve de base para o próximo. Pelo modelo estagiado, mais tradicional e mantendo compatibilidade com o CMM, uma organização pode ter sua maturidade medida em 5 níveis de Maturidade (Maturity Levels):

- Nível 1: Inicial (Ad hoc).
- Nível 2: Gerenciado/Gerido

Áreas-chave do processo para o nível 2:

- Gerenciamento de requisitos.
- Planejamento de projetos.
- Controle e monitoramento de projetos.
- Gerência de subcontratos.

- Análise e medidas.
- Garantia de qualidade.
- Gerência de configuração.
- **Nível 3: Definido**
Áreas-chave do processo para o nível 3:
 - Desenvolvimento dos requisitos.
 - Solução técnica.
 - Integração de produtos.
 - Verificação.
 - Validação.
 - Foco no processo organizacional.
 - Definição do processo organizacional.
 - Programa de treinamento.
 - Gerenciamento de projetos integrado.
 - Gerenciamento de riscos.
 - Integração de equipes.
 - Gerenciamento integrado de fornecedores.
 - Análise e resolução de decisões.
 - Integração do ambiente organizacional.
- **Nível 4: Gerenciado quantitativamente**
Áreas-chave do processo para o nível 4:
 - Gerenciamento quantitativo de processos.
 - Gerenciamento da qualidade de software.
- **Nível 5: Em otimização** – Gerenciamento de mudanças no processo, utilizando a informação quantitativa para melhorar continuamente e gerenciar o processo de software.
Áreas-chave do processo para o nível 5:
 - Inovação organizacional e implantação.
 - Análise de causas e resoluções.

Vantagens da representação por estágios:

- Fornece uma rota de implementação por meio de:
 - Grupos de área de processo.
 - Implementação em sequência.
 - Cada nível funciona como fundamento para o próximo nível.
- Estrutura familiar para aqueles que estão migrando do SW-CMM.
- Habilidade de gerenciar processos na organização.
- Atribuição de uma nota de classificação do nível de maturidade em que a organização se encontra pelos resultados das avaliações, permitindo, dessa forma, a comparação de forma direta entre as organizações.

O SEI criou recentemente o CMMI como um modelo de maturidade tendo por objetivo agrupar as diferentes formas de utilização que foram dadas a seu predecessor, o CMM. O CMMI foi construído sobre a estrutura do CMM, o modelo de melhoria de processo mais difundido e utilizado na comunidade de software, com as seguintes características:

- É mais abrangente, englobando diversas disciplinas em um único modelo, com uma única estrutura, metodologia comum e nomenclatura padrão.
- Pode ser utilizado no desenvolvimento de produtos, serviços e manutenção.
- Reúne melhores práticas de outros modelos.

1.11.5 Áreas de processo

O modelo CMMI v1.2 (CMMI-DEV) contém 22 áreas de processo, representadas na tabela 1.2.

Tabela 1.2 – Custo proporcional de manutenção de software

Área de processo	Área de processo
Análise causal e resolução	Planejamento de projeto
Gerenciamento de configuração	Garantia da qualidade de processo e produto
Análise de decisão e resolução	Integração de produto
Gerenciamento integrado de projeto	Gerenciamento quantitativo de projeto
Medição e análise	Gerenciamento de requisitos
Inovação organizacional e implantação	Desenvolvimento de requisitos

Definição de processo organizacional	Gerenciamento de riscos
Foco de processo organizacional	Gerenciamento de acordo com fornecedor
Desempenho de processo organizacional	Solução técnica
Treinamento organizacional	Validação
Monitoração e controle de projeto	Verificação

1.12 O CMM e o processo de desenvolvimento de software

O CMMI descreve o que deve ser feito em relação ao processo de desenvolvimento de software, com a finalidade de gerar qualidade. Ele diz que devemos, por exemplo, gerenciar requisitos, riscos e mudanças, além de planejar e monitorar o projeto.

Para institucionalizar as práticas previstas no modelo, utilizamos Metodologia de Desenvolvimento de Sistemas utilizando instruções (workflows), guias, modelos, listas de verificação, ferramentas e assim por diante. Para o processo de certificação, deve-se apresentar ao SEI como a empresa está endereçando o modelo CMMI em seus processos internos, o que é verificado por meio da avaliação SCAMPI (Standard CMMI Appraisal Method for Process Improvement). Nas avaliações SCAMPI níveis A e B verifica-se como o processo está sendo endereçado aos projetos, ou seja, verifica-se a aderência dos projetos ao modelo.

Este livro apresentará alguns dos processos que devem ser utilizados por empresas que desejam seguir as práticas de maturidade propostas pelo CMMI.

1.13 Por que utilizar engenharia de software?

A partir das últimas duas décadas temos presenciado uma constante diminuição nos custos de hardware. Por outro lado, devido ao aumento no tamanho e na complexidade dos softwares, eles tem se tornado o principal item no orçamento da computação. Esse aumento da complexidade dos softwares ressaltou alguns problemas relacionados ao desenvolvimento de software, a saber:

- Imprecisão na estimativa de prazos e custos.
- Dificuldade de se atender às demandas.
- Dificuldade de se obter profissionais qualificados para atender às demandas.

- Qualidade dos softwares inferior à necessária, causando insatisfação dos usuários.

Abordar o desenvolvimento de software como engenharia e aliar o assunto com ferramentas e técnicas voltadas ao aumento da produtividade e qualidade dos artefatos gerados foi uma solução encontrada. Devido ao alto custo do desenvolvimento de software e à sua importância cada vez maior, a engenharia de software passa a ter um valor muito grande na garantia de sistemas eficientes e de menor custo.