Alexandre Marcos Lins de Vasconcelos
Sandro Ronaldo Bezerra Oliveira
(Organizadores)

# QUALIDADE, GESTÃO E PROCESSOS DE SOFTWARE

Editora UFPE

# QUALIDADE, GESTÃO E PROCESSOS DE SOFTWARE

Alexandre Marcos Lins de Vasconcelos
Sandro Ronaldo Bezerra Oliveira
Gustavo Henrique da Silva Alexandre
Jussara Adolfo Moreira
Kamila Nayana Carvalho Serafim
Leonardo da Silva Leandro
Marcela da Conceição Oliveira de Souza
Raquel Godoi do Amaral

# QUALIDADE, GESTÃO E PROCESSOS DE SOFTWARE

Editora UFPE
Março/2016

Catalogação na fonte:
Bibliotecária Joselly de Barros Gonçalves, CRB4-1748

Q1 Qualidade, gestão e processos de software [recurso eletrônico] / Alexandre Marcos Lins de Vasconcelos... [et al.]. – Recife : Editora UFPE, 2016.

Inclui referências bibliográficas.
ISBN 978-85-415-0733-2 (online)

1. Software – Controle de qualidade. 2. Software – Desenvolvimento. I. Vasconcelos, Alexandre Marcos Lins de.

005.1 CDD (23.ed.) UFPE (BC2016-006)

# PRÓLOGO

## O objetivo e a origem do livro

O conteúdo deste livro aborda conceitos, teorias e, principalmente, as principais abordagens sobre gestão, qualidade e processos de software tratadas em âmbito nacional pela indústria de software. Seu objetivo é apresentar e ilustrar como estas abordagens devem ser tratadas e usadas no cenário de desenvolvimento de software para resolver problemas de gestão, qualidade e processos de software. O livro pode atuar como como um complemento ao ensino-aprendizagem sobre as melhores práticas usadas para atingir um programa de melhoria organizacional.

Este livro é fruto dos resultados obtidos em uma disciplina ministrada no ano de 2015 no Centro de Informática da UFPE para os alunos de Pós-Graduação (Mestrado e Doutorado) em Ciência da Computação, que tinham o objetivo de: apresentar seminários sobre temas no assunto do livro; discutir em sala de aula sobre as informações tratadas nos seminários; apresentar uma revisão sintática e semântica (com críticas

e sugestões de melhoria) sobre o material apresentado; e desenvolver/refinar as informações tratadas nos seminários na forma de um artigo, sendo este acompanhado pelos professores responsáveis pela disciplina, Prof. Alexandre Vasconcelos (Professor Doutor do Centro de Informática da UFPE) e Prof. Sandro Bezerra (Professor Doutor do Programa de Pós-Graduação em Ciência da Computação da UFPA). Estes artigos, obrigatoriamente, tinham como meta apresentar os resultados da experiência da aplicação e/ou da utilização na indústria de software dos temas tratados neste livro. É direcionado a alunos, gestores, implementadores, avaliadores e outros interessados no tema gestão, qualidade e processos de software.

Por último, este livro não é autossuficiente. Ele exige que o leitor utilize as referências bibliográficas que apresentamos de materiais superlativos. Se algo destacamos deste livro é que a bibliografia vale a pena por si só.

## As vertentes do livro

O título deste livro mistura três ideias poderosas. Fala de gestão em desenvolvimento de software, de qualidade no processo e produto de software e de assuntos que tratam sobre abordagens para a melhoria dos processos de software. Qualquer uma das três ideias merece um livro independente, de maneira que escrever só um, e no curto prazo com que contaram os autores, implica uma escolha de conteúdos. Este é um livro sobre as abordagens que são utilizadas em consultorias de melhoria de processos.

Não é um livro de consultoria, estes existem e são muito bons, escritos por excelentes consultores. No entanto, há muitos conselhos sobre como realizar as coisas importantes,

as que levam a mudanças sérias, que estão contidas nos temas tratados. A escolha destes temas deu-se por se tratarem de abordagens técnicas que costumamos introduzir, de um modo ou de outro, em nossas consultorias na indústria de software e em ensino nas Universidades.

Embora favoreça a melhoria do processo organizacional, não é um livro sobre modelos de melhoria do processo, preferimos que o leitor aprenda sobre esta robusta área nos seus próprios guias e nos cursos autorizados que são oferecidos. No entanto, não há nada no livro que não tenha sido escrito com os conceitos de melhoria do processo em mente.

## Relevância do livro

Em várias regiões do mundo, existem centros de desenvolvimento tecnológico em software, com o objetivo de acompanhar as áreas de Gestão, Qualidade e Processo de Software, empenhados em repassar para organizações interessadas o estado da prática empregado com sucesso em projetos reais.

No Brasil, desde 1993 de modo mais intenso, têm surgido várias iniciativas na área de gestão, qualidade e processo de software, tanto por parte do governo, como de centros de pesquisas, universidades, associações de classes e organismos normativos, com objetivos semelhantes aos já citados.

O setor de software é uma das prioridades governamentais na área de informática. Nesse setor, o governo brasileiro vem incentivando a criação de novas empresas e a capacitação das já existentes. Algumas das iniciativas são: o SOFTEX, com o Programa Nacional para a Excelência em Software, desde 2002; o PBQP Software – Programa Brasileiro de Qualidade e Produtividade, com o objetivo de estimular a adoção de normas, métodos, técnicas e ferramentas da qualidade e da

Engenharia de Software; a SEPIN – Secretaria de Política de Informática do Ministério da Ciência, Tecnologia e Inovação (MCTI), que tem conduzido periodicamente pesquisas sobre qualidade e produtividade entre as empresas de software no país; a Associação Brasileira de Normas Técnicas – ABNT, criou em 1993 o Subcomitê de Software, com o objetivo de gerar normas brasileiras relacionadas à Qualidade de Software.

Apesar da extrema relevância sobre o assunto de gestão, qualidade e processos de software, pode-se perceber na literatura especializada pouco material bibliográfico sobre estes assuntos escrito em português. Fazendo-se uma busca sobre estes assuntos, pode-se notar um total de 5 (cinco) livros tratando de maneira sucinta sobre todas as abordagens apresentadas neste livro de maneira detalhada e com exemplos de utilização. Além disso, pode-se mencionar que o trabalho apresentado neste livro foi realizado numa fase de reestruturação do cenário nacional da indústria de software, onde organizações buscam constantemente a implementação de programas de melhoria a fim de favorecer a qualidade dos seus processos de desenvolvimento e dos seus produtos de software.

01

# INTRODUÇÃO

*Sandro Ronaldo Bezerra Oliveira*
*Alexandre Marcos Lins de Vasconcelos*

Desenvolver software tem se mostrado uma atividade bastante complexa, pois nem todas as organizações desenvolvedoras de software conseguem atingir o escopo definido, assegurar que os prazos sejam cumpridos, que os custos e recursos sejam estimados corretamente e ainda que o software desenvolvido possua a qualidade esperada pelo usuário [the Standish Group International, 2009].

Devido à globalização, cada vez mais os produtos e serviços de software estão sujeitos a novas exigências de mercado, a alta competitividade e a concorrência internacional. Neste cenário, a qualidade torna-se uma arma competitiva, tendo em vista a equiparação com padrões internacionais, conformidade com a especificação e a satisfação do cliente. No entanto, apesar dos inúmeros avanços na Engenharia de Software[1], muito ainda é discutido acerca da baixa qualidade

1 A Engenharia de Software é uma área da ciência da computação voltada à especificação, desenvolvimento e manutenção de sistemas de *software*, com aplicação de tecnologias e práticas de gerência de projetos e outras disciplinas, visando organização, produtividade e qualidade.

e produtividade da indústria mundial de software, refletindo-se na insatisfação dos seus usuários e em prejuízos financeiros de enormes proporções.

A qualidade pode ser definida como a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas [ABNT, 1994]. Entre os benefícios da qualidade estão redução de defeitos, aumento da confiabilidade do produto, redução do esforço de retrabalho, redução de custo de desenvolvimento e manutenção, e maior índice de satisfação dos clientes.

Neste contexto, a Qualidade de Software constitui uma área cuja demanda está crescendo significativamente, pois os usuários exigem cada vez mais eficiência, eficácia, dentre outras características de qualidade importantes para um produto tão especial como o software [Guerra e Colombo, 2009]. Paralelamente à demanda do mercado, existe um movimento nacional e internacional, no sentido de estabelecer normas na área de Engenharia de Software, como é o caso da ISO – *International Organization for Standardization*, no Subcomitê de Engenharia de Software, e da ABNT – Associação Brasileira de Normas Técnicas.

As normas recomendam modelos de qualidade e processos de avaliação da qualidade de software que se têm firmado como valioso auxílio à obtenção de produtos de software com qualidade aprimorada e mais confiável. Por um lado, surge a oportunidade do desenvolvimento de metodologias para avaliar/medir a qualidade de produto de software, com base em atributos de qualidade das normas reconhecidas internacionalmente, dando subsídios para que compradores e usuários adquiram produtos com características mais próximas de suas necessidades. Por outro, surge para produtores de software a possibilidade de fornecer produtos de melhor qualidade.

Buscando uma maior inserção no mercado de desenvolvimento de software, diversas corporações começaram a fazer grandes investimentos para desenvolver sistemas diferenciados com mais qualidade. Para isto, tem-se investido também na melhoria do processo de desenvolvimento de software e se passou a buscar a adoção de modelos e normas de qualidade de processos de software com reconhecimento internacional que possam certificar e avaliar que os sistemas desenvolvidos pela organização são sinônimos de qualidade. Em consequência, foram criados diversos modelos e normas de qualidade de processos de software, como MR-MPS-SW – Modelo de Referência do MPS para Software [SOFTEX, 2012a], o CMMI - *Capability Maturity Model Integration* [SEI, 2010], a ISO/IEC 15504 – *Information Technology – Process Assessment* [ISO/IEC, 2003a] e a ISO/IEC 12207 – *Systems and Software Engineering – Software Life Cycle Process* [ISO/IEC, 2008], que têm como objetivo buscar a garantia da qualidade do produto através da definição e normatização de processos organizacionais a serem aplicados durante o desenvolvimento do software. Tais modelos destacam a importância da avaliação da qualidade dos processos de desenvolvimento e a necessidade do desenvolvimento das habilidades do capital humano.

A comunidade científica, o governo e as próprias organizações, em resposta a essas necessidades, cada vez mais estão buscando a qualidade de software, por meio de abordagens voltadas tanto para o produto final quanto para o processo de desenvolvimento e manutenção de software.

Esta proliferação de vários modelos e normas de qualidade de software e respectivos métodos de avaliação tem levado à dificuldade de intercâmbio de conhecimento entre seus usuários e também à dificuldade de estabelecer comparações entre esses modelos a fim de contrastar seus prós e contras com

o objetivo de selecionar o modelo mais adequado para as necessidades de um projeto específico ou organização. Modelos e normas de qualidade de processos de software têm em comum algumas características (requisitos), mas também têm características particulares que os diferenciam. No entanto, não se pode atestar que algum destes modelos e normas é melhor ou pior que outro. A escolha da adoção de um modelo/norma ou outro depende de vários fatores [Oliveira, 2007] (ex: alvo de mercado a ser atingido, cliente, decisões internas da organização, entre outros), que por vezes não são satisfeitos por um único modelo/norma de forma isolada.

Estes modelos e normas são compostos por processos, que por sua vez possuem práticas que orientam as organizações na implementação de um programa de melhoria do processo de software. As práticas destes processos possuem correlação entre si a fim de prover uma integração entre as melhorias do processo organizacional, como por exemplo o processo de Solução Técnica (cujo o propósito é de selecionar, projetar e implementar soluções para requisitos) possui referência aos processos de Desenvolvimento de Requisitos (definindo conceitos e cenários operacionais), Verificação (realizando revisão por pares), Análise e Resolução de Decisão (usando um processo de avaliação formal para analisar possíveis decisões), Gerência do Desenvolvimento Organizacional (selecionando e implantando melhorias) e Gerência de Requisitos (garantindo consistência entre os requisitos e os produtos de trabalho). Vale ressaltar que estas práticas não esclarecem o "como" devem ser implementadas no escopo organizacional, ficando a critério das organizações definir as boas práticas para a sua implementação, de acordo com a cultura organizacional definida para o desenvolvimento dos seus produtos de software. Assim, o provimento de métodos, técnicas, artefatos,

ferramentas de software, metodologias, entre outros, para apoiar nesta implementação é de extrema importância.

Assim, este livro está dirigido a:

- gerentes de projeto interessados em entender melhor algumas práticas para o planejamento, acompanhamento e controle dos projetos de software;
- responsáveis pela área de qualidade das organizações de software que queiram aprimorar seus conhecimentos sobre alguns dos modelos de qualidade para o processo e produto de software existentes na comunidade;
- responsáveis pela concepção, elaboração, construção e avaliação do processo de software organizacional que desejam melhor entender algumas técnicas e métodos existentes para apoiar nos seus trabalhos;
- engenheiros de software;
- professores de graduação e pós-graduação em Engenharia de Software;
- alunos de graduação e pós-graduação em Engenharia de Software.

Nos próximos capítulos este livro fornecerá:

- conhecimentos importantes para a qualidade de software, abordando modelos para o processo e o produto de software (Parte I – Capítulos 2 a 5);
- os conceitos relacionados à gestão de processos e projetos de software, a partir da apresentação de métodos e boas práticas (Parte II – Capítulos 6 a 9);
- conhecimento sobre algumas técnicas aplicadas ao processo de desenvolvimento de software (Parte III – Capítulos 10 a 13);
- um apanhado geral sobre os assuntos tratados no livro, bem como as suas principais contribuições (Capítulo 14).

# QUALIDADE DE SOFTWARE

*Sandro Ronaldo Bezerra Oliveira*
*Alexandre Marcos Lins de Vasconcelos*

A origem da Qualidade de Software confunde-se com a da Engenharia de Software, já que no início não havia a definição clara das subáreas da engenharia de software. Inclusive ainda hoje, a qualidade de software é ubíqua na engenharia de software [SOFTEX, 2012a], ou seja, está ao mesmo tempo em toda a parte.

A qualidade de software é a área de conhecimento que tem como objetivo principal garantir a qualidade do software através da definição e normatização de processos de desenvolvimento [Schulmeyer e Mcmanus, 1999]. Por este motivo, os modelos aplicados na garantia da qualidade de software focam no processo de desenvolvimento como forma de garantir a qualidade final do produto.

Compreender de fato o que é qualidade de software é essencial para entender como alcançá-la. Ao longo dos anos vários autores e organizações têm tentado definir o significado do termo "Qualidade de Software". O BNQP - *Baldrige National Quality Program* refere-se à qualidade como "qualidade dirigida

ao cliente" e declaram a satisfação do cliente como ponto mais relevante. Segundo a norma ISO/IEC 9000 [ISO/IEC, 2005], a qualidade é o grau em que um conjunto de características inerentes a um produto, processo ou sistema cumpre os requisitos inicialmente estipulados para estes.

Desde as primeiras pesquisas sobre qualidade de software diferentes soluções foram sugeridas para melhoria do produto de software, onde as mais comuns são os *frameworks* de processos, tais como RUP – *Rational Unified Process* [Kruchten, 2000] e MSF – *Microsoft Solutions Framework* [Turner, 2006]. Modelos de maturidade, como o CMMI [SEI, 2010] e a Família de normas ISO, também são constantemente indicados como caminho para obter sucesso no desenvolvimento de software.

A busca pela qualidade de software apresenta resultados substanciais. Muitas empresas reportam dados de projetos de sucesso e demonstram os resultados positivos de investimento em iniciativas de melhoria de qualidade. O SEI – *Software Engineering Institute*, por exemplo, sumarizou os resultados em melhoria de prazo, custo, qualidade do produto, retorno de investimento e outras métricas relacionadas à performance de organizações em um relatório técnico contendo *cases* de 10 diferentes organizações [Gibson, 2006]. Isso demonstra que o investimento em qualidade de software, em particular a implantação de modelos de maturidade de software, quando conduzido da maneira correta, trazem benefícios substanciais para as organizações envolvidas.

*Benchmarkings* mundiais também nos mostram dados importantes sobre o sucesso e o fracasso de projetos, no contexto da qualidade de software. O SPR - *Software Productivity Research*, já coletou dados de mais de 9000 projetos de software pelo mundo ao longo de 12 anos [Jones, 2000]. Segundo eles os fatores que mais influenciam na produtividade do projeto

são: reúso de produtos de alta de qualidade, seguido por experiência do gerente e dos desenvolvedores. Em quarto lugar estão os processos de software, como pode ser visto na Tabela 1.

**Tabela 1.** Fatores que influenciam positivamente em projetos de software [Jones, 2000]

| Novos Fatores de Desenvolvimento | Faixa Positiva |
|---|---|
| Reúso de Produtos de Alta Qualidade | 350% |
| Alta Experiência em Gestão | 65% |
| Elevada Experiência Pessoal | 55% |
| Método/Processo Efetivo | 35% |
| Ferramentas de Gerenciamento Efetivas | 30% |
| Efetivas Ferramentas CASE* Técnicas | 27% |
| Linguagens de Programação de Alto Nível | 24% |
| Ferramentas de Estimativas de Qualidade | 19% |
| Ocupações Especializadas | 18% |
| Participação Efetiva do Cliente | 18% |

* CASE - *Computer-Aided Software Engineering* é uma classificação que abrange todas ferramentas baseadas em computadores que auxiliam atividades de engenharia de software, desde análise de requisitos e modelagem até programação e testes. Podem ser consideradas como ferramentas automatizadas que tem como objetivo auxiliar o desenvolvedor de sistemas em uma ou várias etapas do ciclo de desenvolvimento de software.

Neste contexto, a área de qualidade de software tem se desenvolvido bastante pautada na dualidade produto-processo. De maneira geral, normas e modelos de qualidade de processo definem boas práticas que devem estar presentes em processos ou resultados esperados dos mesmos. Já normas e modelos de qualidade relacionados a produtos de software focam em características de qualidade que um produto deve

apresentar e medidas que podem ser utilizadas para se avaliar tais características [Dal Moro e Falbo, 2008].

Pode-se perceber que há aspectos bastante diferentes em cada perspectiva. As normas e modelos de qualidade de processo estão focados principalmente em processos, propósitos, resultados e atividades. As normas de qualidade de produto, por sua vez, enfocam características dos produtos e como medi-las. Entretanto, há também aspectos comuns. Ambas as perspectivas reconhecem a importância de se avaliar a qualidade das respectivas entidades (processo ou produto). Ou seja, trata-se de entidades que precisam ser mensuradas para ter sua qualidade avaliada [Dal Moro e Falbo, 2008].

Assim, esta parte do livro fornecerá:

- conhecimentos sobre a gestão da qualidade total e gerência de processos de negócios (Capítulo 2);
- os conceitos relacionados à comparação do modelo MR-MPS-SW ao modelo CMMI-DEV (Capítulo 3);
- conceitos sobre o modelo CMMI-Services (Capítulo 4);
- conhecimentos sobre a norma ISO/IEC 25000 para a qualidade do produto de software (Capítulo 5).

02

# GESTÃO DA QUALIDADE TOTAL E GESTÃO DE PROCESSOS DE NEGÓCIOS

*Marcela da Conceição Oliveira de Souza*

A competitividade no setor privado evoluiu em paralelo com os conceitos da qualidade nas últimas décadas. No início do século, como consequência da primeira Guerra Mundial, o foco da qualidade estava no produto obtido pelo aperfeiçoamento dos processos (inspeção final e descarte), caracterizando este ciclo inicial. Na década de 60, o segundo ciclo, preponderou os conceitos de Demin e Juran, conhecidos como "Quality of mind" e administração participativa. Tem-se no terceiro ciclo, na década de 80, conhecida como a "Era do Cliente", a qualidade medida através da confiabilidade do produto, associada aos preços baixos e entregas rápidas, apoiadas no TQM – *Total Quality Management*. O quarto e atual estágio coloca a qualidade como premissa e obrigação para atingimento da estratégia da competividade total, envolvendo *design*, certificações e responsabilidade social.

A qualidade consiste justamente na capacidade de atender as necessidades dos clientes por um preço que eles possam pagar. Acrescenta ainda que, numa organização, todos

devem fazer o melhor que podem, porém, devem também saber o que fazer.

Na Era do Conhecimento, o ambiente é agente influenciador no desenvolvimento de mercados competitivos e neste aspecto o setor público assume papel de indutor, atuando como propulsor da criação e da sustentabilidade do meio competitivo e atrativo para o desenvolvimento tecnológico e inovador. Incumbida do cumprimento desta função, a administração pública tem que saber o que buscar, que caminho seguir e o que atingir.

Aos poucos, as empresas estão modificando a forma de gestão e apostando em projetos de BPM – *Business Process Management*. Esta tendência visa substituir uma administração departamental dos sistemas de qualidade por uma visão sistêmica do negócio, baseada em processos. No entanto, tal mudança pode encontrar resistências por parte das equipes condicionadas no modelo anterior. É por isso que se recomenda uma transição gradativa para a nova modalidade de gestão de processos. Esta estratégia pode identificar as dificuldades do trabalho, alinhar o plano de voo e minimizar os riscos de fracasso durante a implantação.

Neste contexto, os processos de negócios podem oferecer vantagens competitivas, logo, para serem efetivadas, as organizações devem ser capazes de definir, analisar, melhorar, medir e controlar os seus processos. Uma das abordagens crescentes no mercado com este objetivo é o gerenciamento de processos de negócio. De acordo com Korhonen (2007), o BPM é um paradigma-chave da computação empresarial para incrementar agilidade nas organizações. Segundo o CBOK – *Common Body of Knowledge* [ABPMP, 2009] o BPM é uma abordagem disciplinada para identificar, desenhar, executar, documentar, medir, monitorar, controlar e melhorar processos

de negócio automatizados ou não para alcançar os resultados pretendidos consistentes e alinhados com as metas estratégicas de uma organização.

De acordo com Stergiou e Johnson (1998), a transformação organizacional tem sido amplamente discutida e praticada. Os autores falam em um "vazio" entre negócios e tecnologia da informação, como o grande problema das organizações e sistemas. O desenvolvimento de uma organização tornou-se um processo cada vez mais complexo, não dependendo apenas da resolução de problemas técnicos, ou da utilização de modernas tecnologias, mas também altamente dependente dos processos de negócios, objetivando o alinhamento da organização com as necessidades reais de negócios [de La Vara, 2011].

O sucesso de uma organização está condicionado à eficácia com que os seus processos de negócio são executados. Segundo Andrade *et al.* (2004), por meio da modelagem de processos de negócio é possível identificar as partes que compõem os processos da organização, como elas são estruturadas, e como interagem para prover as funções oferecidas a seus clientes.

Neste contexto, a fase da "melhoria contínua", conhecida como "ciclo PDCA – *Plan, Do, Check and Action*", é um método iterativo de gestão de quatro passos, utilizado para o controle e melhoria contínua de processos e produtos. O método visa melhores resultados antes, durante e depois das certificações. O modelo de gestão que utiliza ciclos facilita a adaptação da empresa e permite um aprendizado de forma consciente. Unterkalmsteiner *et al.* (2012) consideram uma questão importante em muitas organizações hoje em dia, que é a busca contínua pela qualidade. Sendo assim, o objetivo geral deste capítulo é investigar os principais conceitos de BPM e TQM.

Baldam *et al*. (2007) destacam que a abordagem BPM facilita a adequação dos processos aos requisitos de qualidade, segurança ou legislação as quais as recentes inovações são submetidas.

No contexto do BPM, a modelagem de processos de negócios BPMN – *Business Process Modeling Notation* pode ser vista como uma das principais etapas, a ser melhor discutida no Capítulo 6. O mapeamento do processo atual, sua análise, melhoria e proposta de mudança são de fundamental importância em todo o ciclo de BPM [Baldam *et al*., 2007].

## 2.1. Gestão da Qualidade Total (TQM)

Refletindo sobre a ineficiência das instituições públicas em nosso país e vislumbrando como uma das causas desse mau funcionamento dificuldades de ordem gerencial, buscou-se aprofundar o estudo de duas estratégias metodológicas de gestão utilizadas em órgãos da administração pública no Brasil. Uma delas foi a Gestão da Qualidade Total, metodologia difundida a partir da década de 80 em todo mundo como consequência do grande crescimento econômico alcançado pelo Japão do pós-guerra, país de origem do método. O objetivo era verificar a sua adequação para consecução de maior qualidade e eficiência nos resultados atingidos por órgãos da administração pública.

Com a massificação da produção surgiram os sistemas de inspeção. Taylor desenvolveu os fundamentos da gestão científica criando a função de inspetor, sendo este o responsável pela qualidade do produto. A inspeção completa de todas as peças fabricadas tornava os procedimentos demorados e onerosos tanto para o fabricante quanto para os consumidores. Face ao exposto, um grupo de pesquisadores liderados

por Shewant e Edwards criaram técnicas para o controle do processo produtivo, as quais permitiam reduzir ao mínimo o número de peças defeituosas. Desenvolveram igualmente técnicas de inspeção por amostragem, surgindo então o controle estatístico de processos. Entre 1950 e 1960 a prevenção passou a ser avaliada. A era da gestão pela qualidade total surgiu no final da década de 70, desdobrando-se em duas linhas de pensamento distintas: a ocidental e a oriental. Os motivos preconizadores da temática em causa foram: a crise do petróleo aliada ao aumento da exportação de produtos de alta qualidade da indústria japonesa.

Estudos recentes das teorias de gestão organizacional demonstram que há, em curso, um processo de mudança no sistema de organização do trabalho [Salerno, 1992; Dankbaar, 1993]. A partir da "revolução" institucional causada pela introdução do taylorismo/fordismo, buscou-se a solução para as crises de acumulação do capital, ao longo dos anos, com a introdução de novas tecnologias. Com o passar do tempo verificou-se que essa estratégia mostrava-se ainda insuficiente para um cenário marcado por rápidas mudanças no mercado e aumento de competitividade. A procura, então, voltou-se para novas formas de organização do trabalho e práticas de gerenciamento que atendessem a dois objetivos básicos: tornar a produção mais flexível e aumentar a produtividade, aumentando, ao mesmo tempo a qualidade dos produtos, fator de competitividade. Neste contexto, o grande desenvolvimento econômico atingido pelo Japão, após a II Guerra Mundial despertou o interesse de estudiosos do assunto, que começaram a tratar as inovações introduzidas no setor produtivo japonês como "modelo" para a implantação de reestruturação produtiva em muitos setores da economia. Entretanto, há autores que afirmam que o Modelo Japonês

em seu conjunto – ou Gestão da Qualidade Total, como é comumente conhecido no Ocidente – pela sua complexidade institucional [Coriat, 1994; Phostuma; Fleury, 1994; Watanabe, 1995] seria intransferível para outros países.

No contexto atual existem três valências principais nas organizações: a tecnologia, a globalização e a crescente complexidade e interdependência. Um dos paradigmas com maior impacto nas organizações que surgiu nos finais do século XX foi à gestão total da qualidade. No entanto, tal como a qualidade este conceito pode ser interpretado de diversas formas.

O objetivo da gestão da qualidade total é a melhoria contínua, através de três princípios básicos: ênfase no cliente, melhoria dos processos e envolvimento total. A qualidade total é um sistema da empresa orientada pela filosofia do melhoramento contínuo dos processos e dos recursos produtivos, que resultará no aperfeiçoamento dos produtos e serviços, objetivando a satisfação de todos os envolvidos na cadeia produtiva. Esse sistema é constituído por princípios, valores, ferramentas (recursos), técnicas (forma de utilizar as ferramentas) e comportamento de pessoas, e necessita integrar à cultura organizacional.

Outros estudos sobre o modelo Japonês, sobretudo no Brasil, mostram que não se pode partir do pressuposto de que tais programas são aplicados de forma idêntica em todos os contextos [Hirata, 1993; Coriat, 1994]. Como nos alerta Lima (1996): "na verdade, temos uma considerável variedade de aplicações desses programas, dependendo das condições econômicas da empresa em questão, da compreensão alcançada a respeito dos seus procedimentos ou da forma como são recebidos pelos trabalhadores e por suas instâncias representativas. Atualmente, encontramos no Brasil algumas empresas que conseguiram importar mais fielmente

o “modelo japonês” e uma grande maioria que restringiu-se apenas a adotar alguns de seus procedimentos, em geral, voltados para a qualidade do produto, a redução dos estoques, a padronização de processos, o controle de desperdícios, etc.”

De acordo com Mears (1993), a gestão pela qualidade total é um sistema permanente e de longo prazo, voltado para o alcance da satisfação do cliente através de um processo de melhoria contínua dos produtos e serviços gerados pela empresa. Sendo que, de caráter geral, uma gestão pela qualidade total que efetivamente tenha controle sobre a qualidade, tem como necessidade a participação de todos os membros da empresa, incluindo gerentes, supervisores, trabalhadores e seus executivos, na busca do objetivo de melhoria contínua.

Segundo Ishikawa (1990), pode-se enfocar a gestão pela qualidade total de duas maneiras distintas. A pequena qualidade é aquela que se limita às características de produtos e serviços consideradas importantes para seus usuários e compradores. A grande qualidade envolve a satisfação comum de várias pessoas, grupos e comunidades envolvidos na vida de uma organização. A pequena qualidade, no longo prazo, não passa de consequência da grande qualidade. Pelo exposto, nota-se que a “grande” qualidade tem um enfoque bastante abrangente e total em relação à organização. Este enfoque exige uma mudança aguda na filosofia tradicional que se pratica em termos de produção nas empresas, uma mudança de um foco baseado em custos e produtividade para um fundado em qualidade e na visão do cliente/consumidor. Este novo enfoque, segundo Garvin (1992) tem cinco pressupostos básicos:

- Qualidade é definida do ponto de vista do cliente;
- Qualidade é relacionada com lucratividade em ambos os lados, do mercado e de custos;
- Qualidade é visualizada como uma arma competitiva;

- Qualidade é construída desde o processo de planejamento estratégico;
- Qualidade requer um compromisso que abranja todos os membros da organização.

Segue alguns aspectos importantes do controle da qualidade total:

- Controle do processo: controle de todas as fases do processo durante a produção. Esta tarefa requer uma quantidade elevada de inspetores caso a qualidade não seja responsabilidade da produção. Cada posto de trabalho é, também, um posto de inspeção e controle da qualidade do processo;
- Visibilidade da qualidade: estabelecimento de objetivos de qualidade mensuráveis e exposição da situação da produção em relação a estes objetivos, através de quadros e cartazes por toda a empresa. Dessa forma, todos podem estar cientes da situação referente à qualidade;
- Disciplina da qualidade: enquadramento das atitudes de todos em relação às metas de qualidade, não permitindo um relaxamento. Para tal, é necessário o total comprometimento da gestão de alto nível;
- Paralisação das linhas de produção: prioridade total para a qualidade, ficando em segundo lugar a quantidade produzida. As linhas de produção devem reduzir a sua velocidade, ou mesmo parar, caso a qualidade não seja satisfatória, para que os problemas sejam resolvidos.

Diversos estudos de empresas de consultoria demonstram que as organizações que aceitaram processos de implementação da gestão total da qualidade tiveram retornos de 20 a 36% [Stark, 1998], assim como, demonstram a preferência dos gestores pela gestão total da qualidade enquanto ferramenta de gestão [Rigby, 1998].

No entanto, um ponto importante para a discussão é alertar para as dificuldades que as organizações enfrentam durante o processo de implementação da gestão total da qualidade. Como consequências das dificuldades referidas no ponto anterior, inúmeros acadêmicos preconizaram uma visão bastante pessimista quanto ao futuro do movimento da gestão total da qualidade [Hackman e Wageman, 1995]. Este movimento acentua-se com a publicação dos seguintes artigos: "*The Cracks in Quality*" [THE ECONOMIST, 1992]; "*Is TQM Dead?*" [USA TODAY, 1995]; "*Is Total Quality Management (TQM) yesterday's news or does it still shine?*" [WALL STREET JOURNAL, 1995]. Zhu e Scheurmann (1999) enumeram algumas razões para a falha da gestão total da qualidade: individualismo, competitividade, orientação apenas para solução de problemas, orientação para o controle, pensamento linear, falta de comprometimento da gestão de alto nível, falta de confiança dos colaboradores e obsessão por ganhar prémios.

Segundo Revere e Black (2003), uma das dificuldades da gestão total da qualidade é a medição. As equipes devem estabelecer critérios para avaliar os desvios, desenvolver meios para identificá-los e trabalhar para minimizá-los. De acordo com Oakland (1994), as medidas tradicionais de desempenho proporcionam pouco incentivo aos esforços de implementação da gestão total da qualidade, porque são incompatíveis com medidas de aperfeiçoamento da qualidade. Porém, as diversas histórias de sucesso empresarial, como: Ford Motor Company, Phillips Semiconductor, Motorola, Toyota Motor Company, General Motors, Boeing, IBM, entre outras, demonstram o contrário.

A gestão da qualidade global tem-se debruçado sobre a gestão da qualidade em organizações globais. Segundo Kim e Chang (1995) esta define-se como um planejamento

estratégico com interação de produtos e processos, para que se obtenha uma elevada aceitação do cliente e baixa disfuncionalidade organizacional através de mercados em diferentes países. Esta visão incorpora a filosofia central básica da gestão total da qualidade, reconhecendo, contudo as diferenças dos mercados globais, implicando assim na necessidade de balanceamento dos requisitos locais dos clientes com as capacidades de redes de fornecimento das organizações. Kim e Chang (1995) acrescentam ainda que a evolução do conceito da qualidade em torno da gestão da qualidade global estaria estruturada ao longo de cinco dimensões básicas: orientação do mercado; orientação da produção; orientação da produção; sistema de informação; e, rede tecnológica.

Em suma, pode-se afirmar que o fracasso da gestão total da qualidade em inúmeras organizações ocorre, primordialmente, pelo pouco comprometimento da gestão de alto nível com as suas práticas. A sociedade e o mercado exigem maior qualidade dos produtos e serviços, contudo, de forma ingênua alguns gestores proclamaram uma gestão "cosmética" da qualidade nas suas organizações, vinculado ao fracasso.

## 2.2. Gerenciamento de Processos de Negócios (BPM)

Todo trabalho importante realizado nas empresas faz parte de algum processo [Graham e Lebaron, 1994]. Estes processos estão inseridos nas organizações, e passam por diversas áreas funcionais de sua estrutura e podem ser entendidos como a ordenação de atividades de trabalho através do tempo e do espaço, com um início e fim e um conjunto claramente definido de entradas e de saídas [Davenport, 1994]. Os processos utilizam os recursos da organização para oferecer resultados objetivos aos seus clientes [Harrington, 1991].

Cada organização tem missão, objetivos e processos próprios e é importante dar atenção à modelagem desses itens. Alencar (1999), *apud* Pádua *et al.* (2004), destaca alguns objetivos da modelagem organizacional como: fornecer um objeto, que seja uma representação compartilhável e reutilizável da cadeia de fornecimento de informação e conhecimento; suportar tarefas da cadeia de fornecimento, pela habilitação de respostas a questionamentos, que não estão explicitamente representados no modelo; definir os objetos de maneira precisa, de forma que sejam consistentemente aplicados, por meio dos domínios e interpretados pelos usuários; e suportar visualização do modelo, de forma intuitiva, simples e consistente.

A organização pode ser analisada sob três diferentes níveis hierárquicos: o nível estratégico sendo o mais alto, onde é elaborado o planejamento estratégico e são definidas as atribuições do nível tático; o nível tático que, por sua vez, elabora um planejamento para execução de suas atribuições, delegando tarefas ao nível operacional; e o nível operacional, que também planeja a execução dessas tarefas e as executa [Rezende e Abreu, 2001; Chiavenato, 2000a]. O planejamento elaborado em cada nível precisa ser acompanhado através de indicadores que tratem informações quantitativas e qualitativas sobre o desempenho da organização, e assim permitir a análise de cada indicador de forma a detectar se o comportamento da organização está em conformidade com o comportamento esperado. Neste ponto também é importante definir as metas. Outra situação que precisa estar bem definida é a missão e visão da organização, pois estes devem orientar a definição dos objetivos estratégicos da organização. A missão de uma organização representa sua finalidade, enquanto a visão descreve o que ela gostaria de ser dentro de um determinado período de tempo [Chiavenato, 2000a].

Na era industrial, a alocação de recursos era puramente financeira e física, utilizava-se de índices financeiros e de produtividade para mensurar o desempenho das empresas. Essas premissas, no entanto, tornaram-se obsoletas na era da informação. Em um ambiente complexo, para se obter vantagem competitiva é preciso muito mais [Kaplan, 2001]. Os executivos necessitam hoje de indicadores sobre vários aspectos do ambiente e desempenho organizacional, sem os quais não teriam como manter o rumo da excelência empresarial. Os funcionários devem agregar valor pelo que sabem e pelas informações que podem fornecer. Esse conhecimento passou a ser um fator crítico de sucesso à medida que as organizações investem, gerenciam e exploram esse conhecimento [Kaplan e Norton, 1997].

Segundo Ferrari (2011), métricas e indicadores são mais que simples elementos matemáticos e estatísticos. São também elementos de impacto psicológico e comportamental nas pessoas e organizações, já que a forma com que são medidas determina com grande força suas reações. Logo, sejam para análises estatísticas ou para alinhar pessoas e equipes em torno de visões e objetivos comuns, métricas e indicadores exercem papel fundamental na gestão das empresas.

A busca por flexibilidade está vinculada à necessidade das organizações de se adaptarem a mudanças frequentes e sem precedentes em seus ambientes. Segundo Araújo (2001), ao longo das últimas décadas, teorias e práticas administrativas como organização, sistemas e métodos (OS&M), gestão da qualidade total e reengenharia, entre outros, vieram à tona trazendo contribuições importantes para o desenvolvimento empresarial e colaborando no atendimento das novas demandas do mercado.

O ponto comum entre estas abordagens é a introdução do conceito de processos de negócios como forma de gerenciar

e estruturar as empresas. De acordo com Korhonen (2007), BPM é um paradigma-chave da computação empresarial para incrementar agilidade nas organizações. Hammer e Champy (1995) conceituam processos de negócios como "um conjunto de atividades com uma ou mais espécies de entrada e que cria uma saída de valor para o cliente".

Segundo Sordi (2005), as empresas que adotam esta modalidade de gestão são intituladas de "orientadas a processos", enquanto as empresas que permanecem focadas apenas em suas funções empresariais são rotuladas como "baseadas em funções".

Abordagens de negócios orientadas por processos têm o objetivo de obter e especificar os requisitos para desenvolver sistema e suportar os processos de negócios de uma organização. Estas abordagens concentram-se na compreensão dos objetivos, no apoio da criação de modelos de processos de negócios, além de um modelo de dados e as especificações funcionais para novas formas de execução de processos de negócio [Vasconcellos, 2012].

Segundo Vasconcellos (2012), a necessidade de modelagem de processos de negócio para entendimento dos serviços prestados pelas organizações tem sido amplamente reconhecida no meio acadêmico. Nos últimos anos, percebe-se um interesse crescente, da indústria e da academia, na utilização de BPM como estratégia para elicitação, especificação e modelagem conceitual de requisitos de sistemas de informação que, por sua vez, devem apoiar os processos de negócios das organizações.

Eriksson e Penker (2000) explicam que, para modelar e entender o ambiente do sistema, as organizações devem responder algumas perguntas, como:

- Como os diferentes atores interagem?

- Quais atividades fazem parte do seu trabalho?
- Quais são os principais objetivos de seu trabalho?
- Quais pessoas, sistemas ou recursos estão envolvidos e não aparecem como atores do sistema específico?
- Quais as regras que regem suas atividades e estruturas?
- Existe outra maneira dos atores executarem a atividade de forma mais eficiente?

Segundo Andrade *et al.* (2004), os modelos representam a arquitetura do negócio, essa representação é realizada descrevendo-se as partes que compõem os processos da organização, como elas são estruturadas e como interagem para prover as funções oferecidas a seus clientes.

Conforme Gonçalves (2000), os processos de negócio (PN) caracterizam a atuação de uma organização. Nesse sentido, os PN existem para concretizar os objetivos estratégicos, porque é por meio deles que as organizações podem entregar o que está descrito na sua missão. Vale enfatizar que o conceito de PN é aplicado a qualquer tipo de organização, independente da sua natureza.

O que deve ficar claro é que um PN caracteriza a atividade-fim ou um conjunto de atividades principais de uma organização [Melão e Pidd, 2000]. Logo, os PN refletem a essência do que uma empresa faz.

O BPM é uma abordagem de gestão que entende as organizações como um conjunto de PN e que tem despertado a atenção dos gestores, sobretudo, a partir da virada do século, conforme pondera Baldam *et al.* (2007). Um dos principais insumos de um PN é o conhecimento que as organizações produzem, a partir do seu próprio ambiente (interno e externo) de atuação. Como esse ambiente é dinâmico, os conhecimentos demandados pelos PN mudam ao longo do tempo. Assim, o BPM, também, apresenta um ciclo de vida.

A Figura 1 delimita o ciclo de vida de gerenciamento de PN. Ele reflete uma fusão das abordagens de *Aalst et at.* (2003), Jung, Choi e Song (2006), Baldam *et al.* (2007), Kor *et al.* (2009) e Houy *et al.* (2009), sobre gestão de PN. De acordo com tais autores, as fases desse ciclo possuem as seguintes características:

- Planejar: tem o propósito de definir os PN que efetivamente sustentam as atividades principais de uma organização;
- Modelar: essa fase é constituída de atividades que permitem identificar e/ou criar conhecimentos sobre o processo atual (modelagem AsIs) e, principalmente, identificar melhorias que serão incorporadas ao modelo de processo desejado (modelagem To-Be), gerando inovações em produtos e serviços e, em muitos casos, até a identificação de novos segmentos de negócios;
- Implementar: a partir da compreensão detalhada e integrada proporcionada pela fase de modelagem, modelos do processo são construídos, contemplando a criação de regras de negócio, inclusão de boas práticas, determinação de padrões, métodos e automatização de tarefas. Esses modelos são armazenados em repositórios de processos para que todos tenham acesso a eles;
- Executar: corresponde à fase em que se verifica se o processo de negócio gera valor para mercados, sociedade e cidadão;
- Monitorar: aqui, encontram-se as atividades relacionadas ao monitoramento geral do processo. Essa fase provê aos tomadores de decisão (de todos os níveis organizacionais) informações sobre o comportamento dos PN;
- Otimizar: os processos são refinados e inovações são implementadas de acordo com os dados e informações geradas pela fase de monitoramento e controle. A redução do tempo de ciclo é um dos resultados mais significativos dessa fase.

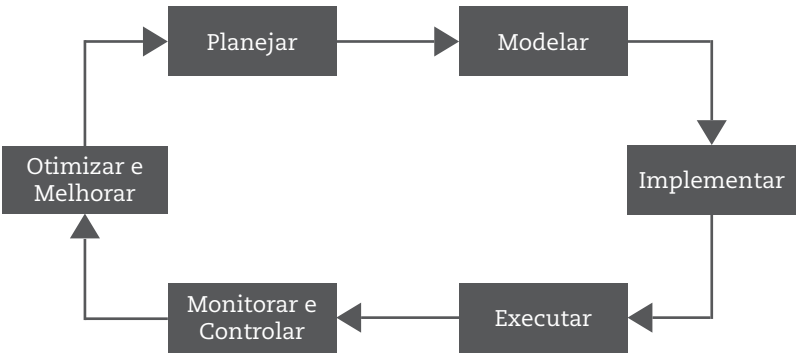



**Figura 1.** Ciclo de Vida do BPM, Adaptado de Baldam *et al.* (2007) e Houy *et al.* (2010)

- O impacto da aplicação de BPM abrange desde alterações na forma de realização do trabalho individual até a maneira pela qual as empresas trabalham juntas em processos interorganizacionais, passando pela redefinição da maneira pela qual os grupos de pessoas realizam suas tarefas coletivas [Gonçalves, 2000].

A implementação de BPM minimiza consideravelmente a possibilidade de ocorrência de erros e aperfeiçoa tempo e recursos. Já vem ocorrendo ao longo do tempo a percepção do interesse das organizações por estas práticas de gestão ou melhoria de processos, as quais vem crescendo e se consolidado a cada ano [Dale *et al.*, 2001]. Conforme dados de uma pesquisa realizada pela TIBCO, a maioria absoluta das empresas que implementaram um projeto de BPM perceberam aumento de produtividade, com 95% informando melhora na qualidade de serviço e 82% mencionando redução de custos operacionais [Tibco Customer Research, 2009]. No entanto, existem decisões e mudanças organizacionais que dependem do gestor e do comprometimento dos funcionários. O processo de gestão empresarial é um campo muito complexo, pois envolve desafios de vários domínios, como organizacional,

gerencial, sistemas de informação e até mesmo problemas sociais [Trkman, 2010].

Segundo Jeston e Nelis (2008), existem três aspectos críticos para um processo de melhoria de projetos: processos, pessoas e tecnologia; além disso, existe um quarto fator que é a base destes pilares e se constitui de todas as boas práticas em gestão de projetos (boa comunicação interna, liderança, estratégia, definição de serviços, envolvimento de todos). Quando algum destes fatores está mal desenvolvido, ou o processo não está alinhado com as estratégias da empresa, percebe-se que a empresa não está apta a implementar projetos BPM.

Desta forma, para que a iniciativa de adotar a gestão por processos possa ser bem sucedida, todos precisam estar dispostos a participar e compreender seus erros e pontos a serem melhorados. Além disso, é preciso promover a melhoria contínua, ou seja, acompanhar o andamento dos processos e garantir mudanças e adaptações sempre que necessário.

É necessário primeiro elaborar o planejamento estratégico, pois é preciso identificar o que será necessário no futuro para que medidas que consomem um tempo maior possam ser iniciadas antes. Este é um momento de reflexão sistemática sobre os negócios da empresa, sua trajetória e situação atual, suas forças e fraquezas, o contexto mercadológico em que está atuando, as oportunidades e ameaças que se apresentam, benefícios e riscos de investimentos, fatores limitantes sistêmicos dos processos e, finalmente, os próprios anseios e objetivos dos seus diretores. Foi identificado que este é um processo de gestão que oferece subsídio para definição da estrutura de governança de processos atuarem por toda empresa de forma alinhada aos objetivos estratégicos [Ferrari, 2011].

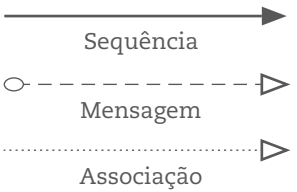
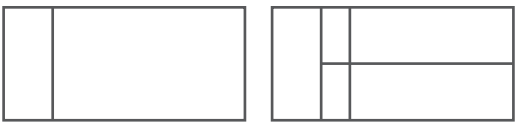
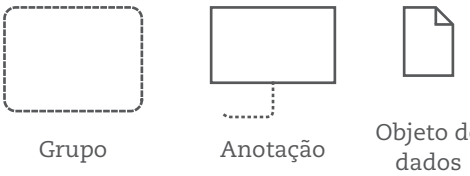
## 2.3. BPMN e Seus Padrões

A BPMN - *Business Process Modeling Notation* é um padrão cada vez mais importante para a modelagem de processos e tem desfrutado de altos níveis de atenção e aceitação na prática de BPM. Desenvolvida pelo BPMI - *Business Process Management Initiative*, onde este em seguida foi incorporado pelo OMG - *Object Management Group* [OMG, 2009]. A notação BPMN provê um padrão que preenche o espaço, e faz uma ligação, entre modelos de negócio e sua implementação [Vara *et al.*, 2008]. Os diagramas de processos de negócios criados utilizando a notação BPMN são chamados de BPD - *Business Process Diagrams*. Um BPD é uma rede de objetos gráficos que representam atividades e um fluxo de execução que definem a ordem de execução destas atividades. Estes objetos gráficos, ou elementos, permitem o desenvolvimento fácil de diagramas simples de compreender pela maioria dos analistas de negócio [ABPMP, 2009].

Os elementos utilizados para construção de um BPD foram classificados em categorias específicas, de forma que os leitores de BPD possam reconhecer facilmente os elementos básicos e entender os diagramas [ABPMP, 2009]. As quatro categorias básicas do BPMN são: Objetos de Fluxo, Objetos de Conexão, Raias e Artefatos; apresentadas na Tabela 2.

**Tabela 2.** Elementos básicos das categorias da BPMN.

| Categoria | Elementos |
|---|---|
| **Objetos de Fluxo** *(Flow Objects)* São os principais elementos gráficos. Definem o comportamento dos processos de negócios. |  Eventos  Gateway  Atividades |

| Categoria | Elementos |
|---|---|
| **Objetos de Conexão** *(Connecting Objects)* Realizam a conexão entre objetos de fluxo ou destes objetos com outras informações para criar o esqueleto básico do processo. |  |
| **Raias** *(Swimlanes)* Organizam as atividades de forma visual, a fim de representar responsabilidades (papéis) ou capacidades funcionais diferentes. |  |
| **Artefatos** *(Artifacts)* Permitem que a notação básica seja estendida e que informações referentes ao contexto do processo de negócio sejam adicionadas ao modelo para fins de análise do modelo criado. |  |

Dentro das categorias básicas da notação BPMN existem variações as quais podem ser utilizadas para desenhos de processos mais complexos. Um padrão industrial para modelagem de processos, é de fato uma linguagem rica que possibilita definir uma vasta quantidade de cenários de negócios, variando de coreografias de processos internos para arquitetura de processos interorganizacionais, interações de serviços e exceções de fluxo de trabalho. É possível inclusive, construir um código de linguagem de execução de processos de negócio (BPEL - *Business Process Execution Language*) a partir dos modelos de BPMN mapeados.

Algumas das vantagens do BPMN são [Braconi e Oliveira, 2009]:

- Padrão de notação com suporte em diversas ferramentas;
- Permite evolução para o padrão XPDL 2.0, que é uma linguagem de descrição de fluxo;
- Permite a conversão direta (e automática) para BPEL, reduzindo assim a lacuna entre o desenho do processo e sua implantação (automação);
- Incorpora facilidades de técnicas como UML e IDEF;
- Notação mais facilmente compreendida e usada por todos os envolvidos nos processos de negócio.

São desvantagens do BPMN [Braconi e Oliveira, 2009]:

- Integração do BPMN em outras ferramentas é parcialmente atendida por ser somente uma notação gráfica depende da sua representação textual;
- Por ser focado em processos, dificulta o manuseio de diferentes visões.

Um conjunto de diagramas pode ser elaborado usando uma combinação dos modelos de processos. No entanto, nem sempre o resultado gerado pode ser um mapeamento em linguagem executável. Torna-se de extrema importância para uma análise BPM a modelagem do estado atual, a obtenção de um diagnóstico do processo existente, mesmo que não existam práticas documentadas é algo necessário ao processo de análise do problema. Este mapeamento pode ser apresentado através de representação gráfica ou de forma descritiva. Também pode ser chamado de "Modelagem do estado atual" ou simplesmente de "Modelagem AsIs". De certa forma, é visto por alguns especialistas, os quais defendem que o primeiro passo em qualquer projeto BPM (exceto, tratando-se de um processo novo), é entender o processo existente e

identificar suas falhas, ou seja, desenvolver a visão do estado atual e assim fazer a modelagem do AsIs.

## 2.4. Indicadores

Para garantir o alinhamento, entre os aspectos estratégico, tático e operacional da gestão de processos, é de vital importância realizar uma avaliação real sobre os resultados alcançados, no qual o processo de planejar deve cobrir a tradução de metas estratégicas em ações realizáveis e controláveis. Logo, a partir dessa ideia, é preciso definir métricas e indicadores, que possibilitem a tomada de decisão de maneira mais eficiente. Indicadores representam informações a partir das quais, é possível avaliar uma situação e sua evolução histórica. Contudo, indicadores mal definidos podem levar a conclusões equivocadas.

Ao definir indicadores com o objetivo de monitorar o desempenho dos processos, muitas organizações definem seus painéis de monitoramento com base em métricas que focam apenas no resultado desejado e perdem, assim, a valiosa oportunidade de monitorar não apenas o resultado do processo, mas a sua execução. Dessa forma, o resultado é conhecido apenas quando não há mais nada a se fazer para que a conclusão do processo seja favorável à organização. Diante deste cenário, existem os indicadores direcionadores (*drivers*), que monitoram a causa antes do efeito e caracterizam-se pela possibilidade de alterar o curso para o alcance de um resultado e os indicadores de resultados (*outcome*), que monitoram o efeito e não permitem mais alterar um dado resultado.

O monitoramento é a observação, bem como o registro regular relativo às atividades de um processo. É um trabalho rotineiro de acúmulo de dados, os quais devidamente

estruturados transformam-se em informações. Monitorar é checar o progresso das atividades, ou seja, uma observação sistemática e com propósitos, visando dar ciência dessas informações de desempenho do processo aos responsáveis pelo mesmo. Um avanço sobre esse tema configura-se quando se estabelecem medidores, com metas, relativas a cada informação tratada, em cada processo. Assim, com auxílio de sistema informatizado, as informações são coletadas e comparadas com as metas e os resultados já são apresentados com algum *display* sobre o seu estado, em relação às metas: atingido, não atingido ou na faixa de tolerância. Esses medidores são chamados de indicadores de desempenho (em inglês KPI: *Key Performance Indicator*).

Monitorar é uma maneira de seguir, coletar e oferecer informação de retorno sobre eficiência ou eficácia do processo, compreende o rastreamento da execução dos processos, de modo que as informações sobre seu estado possam ser facilmente vistas e entendidas, e assim constituam elemento base para a tomada de decisão, por parte do gestor responsável. A medição de desempenho é a forma de prover a informação necessária para que os gestores tomem as decisões corretas sobre a alocação de recursos em suas operações de negócio, de forma a atingir os objetivos previamente estabelecidos. Além disso, é um elemento crítico para o ciclo de vida de BPM ao prover informações valiosas para atividades como análise, desenho e transformação dos processos, bem como a revisão da sua eficiência e eficácia. Abaixo algumas justificativas para a medida do desempenho:

- Maior assertividade na tomada de decisão;
- Melhorar a eficiência / eficácia das atividades;
- Transparência na divulgação dos resultados;

- Criação de uma cultura de excelência para a empresa;
- Prover resposta ao gestor sobre o desempenho de algum elemento do processo.

Diante deste contexto, surge uma das técnicas mais populares na definição de métricas de negócios, a abordagem GQM – *Goal, Question, Metrics*, na qual são elaboradas metas visando avaliar a situação dos projetos, e a relação existente com os objetivos estratégicos da empresa, afinal "não se pode controlar aquilo que não se pode medir" [Tom de Marco, 1986]. Segundo Basili (1992), GQM é uma abordagem orientada a metas e utilizada em engenharia de software para a medição de produtos e processos de software, e tem como ideia básica derivar métricas de software a partir de perguntas e objetivos. Logo, essa técnica permite ajudar na identificação de métricas úteis e relevantes, apoiar a análise e interpretação dos dados coletados, de modo a avaliar os resultados obtidos.

## 2.5. Automação de Processos de Negócios

Atualmente um novo patamar da gestão por processos foi estabelecido através da automação dos processos, o que se tornou possível graças aos avanços realizados na área de tecnologia da informação. A implantação de soluções e ferramentas de *workflow* (fluxo de trabalho) foram as primeiras a serem desenvolvidas para este fim e, posteriormente, ferramentas chamadas BPMS - *Business Process Management System* surgiram com novas funcionalidades como simulação e monitoramento do processo gerenciado, além do simples acompanhamento da trajetória de um processo.
Algumas mudanças aconteceram, primeiro houve uma mudança no tipo dos problemas de processos que as pessoas de

negócio estão tentando resolver. Por exemplo, as pessoas de negócio migraram de um cenário onde resolviam de uma forma mais ou menos estável, processos procedimentais, para um cenário onde tentam construir soluções para processos mais dinâmicos. Segundo, houve uma mudança nos tipos de ferramentas de softwares que os fornecedores estão oferecendo, variando de ferramentas de *workflow* para suítes que combinam *workflows*, EAI - *Enterprise Application Integration*, regras de negócio e mineração de processos.

O IDC (2007) cita que ferramentas de BPMS são as plataformas escolhidas para desenvolvimento de aplicações feitas sob medida, e indica alguns fatores responsáveis pelo aquecimento deste mercado:

- A penetração de sistemas BPMS ainda é pequena e sua adoção por grandes corporações encoraja novos compradores;
- Muitas implantações de BPMS são feitas com escopo reduzido, geralmente um único projeto, possibilitando uma expansão para incluir processos adicionais em ferramentas já instaladas;
- Os fornecedores de BPMS estão enriquecendo suas ferramentas, transformando-as em grandes suítes;
- O modelo de software como serviço (SaaS - *Software as a Service*) está apenas emergindo como um modelo de negócios no mercado e sua disseminação ajudará na adoção de sistemas BPMS no *midmarket*, pequenos negócios e arenas de B2B - *Business to Business*.

A crescente adoção de sistemas BPMS só se tornou viável pelo conjunto de tecnologias emergentes, sob a arquitetura orientada a serviços (SOA - *Service Oriented Architecture*), permitindo que as mesmas possuam características atraentes a sua adoção. Tal arquitetura permite a manutenção de

sistemas existentes, que terão seus serviços disponibilizados para consumo pelos processos implantados no sistema BPMS.

Sistemas BPMS são destinados à facilitação da gestão por processos de negócio. Atuam fortemente no aumento da velocidade de execução dos processos de negócio através da automação de determinadas tarefas, e integração de sistemas legados, reduzindo assim o tempo gasto em navegação de telas pelo usuário durante a execução de uma tarefa [Verner, 2004]. Um BPMS pode ser definido como um conjunto de ferramentas ou instrumentos que buscam a melhoria do sistema de gestão (sob a visão por processos).

Por ser um conjunto de software e tecnologias existentes, os sistemas BPMS podem ter focos diferentes, o que acarreta em distintos tipos de BPMS [Chang, 2006]:

- Centrado em Dados (*Data-Centric*): Produtos deste tipo são focados na extração, transformação e transporte de dados principalmente através de sistemas de banco de dados. Essa classe de produto é limitada à execução de processos Sistema-Sistema;
- Centrado em Aplicações: Essa classe de produtos possui o foco na integração de aplicações. São destinados à implantação de processos dos tipos Sistema-Sistema e Pessoa-Sistema. Como meio de integração utilizam mensagens e componentes. Possuem também uma camada de gestão por processo, um *designer* de processo e um motor de execução, podendo incorporar ferramentas de monitoramento de indicadores. Podem também ter uma camada de fluxo humano e assim permitir a execução de processos Pessoa-Pessoa, no entanto, tais produtos possuem deficiências com relação ao tratamento do fluxo de trabalho humano, não sendo ideais para a implantação de processos complexos do tipo Pessoa-Pessoa;

- Centrado em Processos: São produtos com origem em antigos sistemas *workflow* ou construídos a partir do zero. Possuem capacidades robustas para o desenho dos processos através de uma ferramenta de desenho com mais funcionalidades do que a categoria anterior de sistemas. Certas ferramentas oferecem capacidades de simulação de cenários para os processos desenhados. Oferecem também portais de trabalho e listagem de tarefas sendo mais bem utilizados para processos do tipo Pessoa-Pessoa do que as duas categorias anteriores de sistema. Permitem também o trabalho conjunto de analistas de negócio e analistas de TI. São componentes principais desse tipo de sistema:
    - » Modelador de processo de negócio;
    - » Ferramenta de simulação;
    - » Motor de execução de processos;
    - » Ferramentas de interação humana (p.ex., portal de trabalho);
    - » Serviços de integração;
    - » Monitor de processos.

Linguagens de regras de negócio focam na especificação dos requisitos para tomada de ações, em contraste ao BPMN que foca em como algo é realizado. A necessidade de regras de negócio dependerá da complexidade dos processos de cada organização e cada sistema BPMS que oferece ferramentas mais complexas para modelagem de regras de negócio usa sua própria combinação de BPMN e linguagem de regra de negócio (não há um padrão definido para tais linguagens), como pode ser visto na Figura 2.

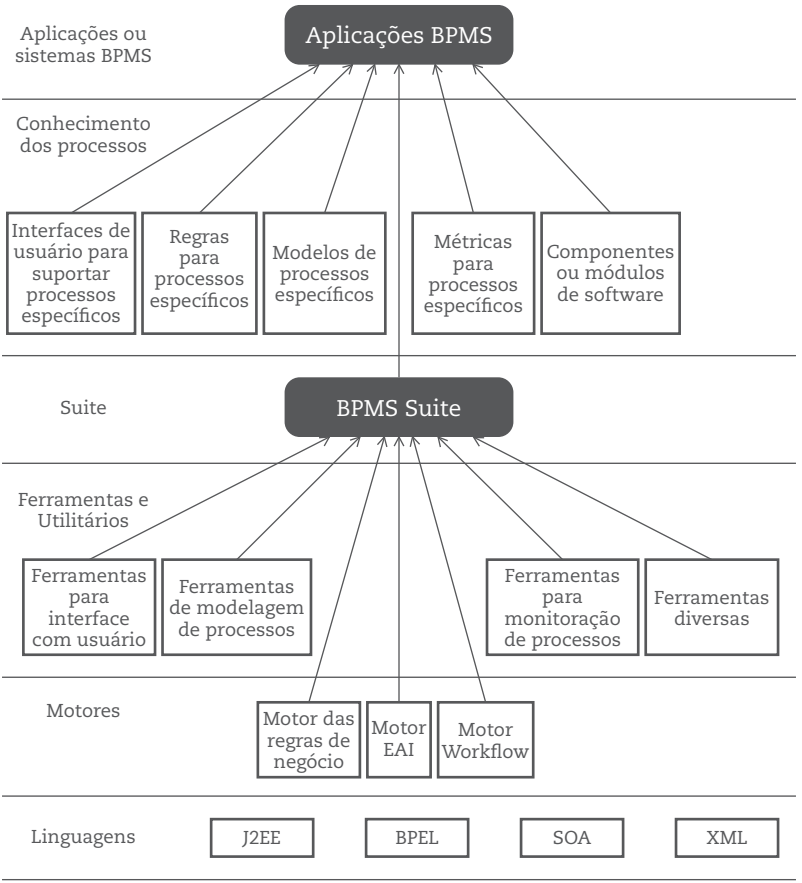



**Figura 2.** Modelo genérico de software BPMS

Os sistemas BPMS possuem como característica a integração de tecnologias na orquestração dos processos de uma organização. Tal característica exige das tecnologias uma padronização, para que sejam intercambiáveis, como pode ser visto na Figura 3.

| Entidade | Padrão | Tipo de padrão | Descrição do padrão |
|---|---|---|---|
| WfMC | XPDL (XML Process Definition Language) | Definição de processo | Similar ao BPEL, também e baseada em XML. |
| WfMC | Wf-XML 2.0 | Interação de processo | Linguagem XML para comunicação entre web services e engines de *workflow*. |
| OASIS | BPEL4 (Business Process Execution Language) | Definição de processo | Linguagem mais disseminada para BPMS. Representa um processo em linguagem baseada em XML e permite conexões com web services. |
| OASIS | BPEL (Business Process Execution Language) | Interação de processo | Linguagem mais disseminada para BPMS. Representa um processo em linguagem baseada em XML e permite conexões com web services. |
| BPMI/W3C | BPML (Business Process Modeling Language) | Definição de processo | Similar ao BPEL, também é baseada em XML. |
| BPMI/W3C | WSCI (Web Service Choreography Interface) | Interação de processo | Linguagem XML para coreografia em web services. |
| WfMC | *Workflow* API | Interação de processo | API funcional e administrativa com definições em C, IDI e COM. |
| W3C | WS-CDL (Web Service Choreography Description Language) | Interação de processo | Linguagem XML para coreografia. Linguagem oficial da W3C. |
| W3C | WSCL (Web Service Conversation Language) | Interação de processo | Linguagem XML básica para coreografia. |
| OMG | BPRI (Business Process Runtime Interface) | Interação de processo | Modelo MDA para API de BPM funcional e administrativa. |
| Microsoft | XLANG | Definição de processo | Uma das primeiras linguagens de processo em XML. Influenciou a BPEL. |
| IBM | WSFL (Web Services Flow Language) | Definição de processo | Uma das primeiras linguagens de processo em XML. Influenciou a BPEL. |
| OASIS | BPSS (Business Process Specification Schema) | Interação de processo | Linguagem de processos para colaboração B2B. |

**Figura 3.** Padrões associados a sistemas BPMS [Chang, 2006; Baldam et al., 2007; Havey, 2005]

No entanto, alguns padrões são mais utilizados entre os fabricantes das ferramentas de BPM, a seguir são apresentados os padrões que, de acordo com Jeston e Nelis (2008), lideram em BPM:

- BPEL - *Bussiness Process Execution Language*: é atualmente o principal componente na execução que orquestra os processos de negócio usando *web services*, e permite que várias aplicações BPM estejam ligadas e integradas;
- BPML - *Business Process Modeling Language*: este compete interage diretamente com BPEL como metalinguagem para a modelação de processos de negócio;
- BPMN - *Business Process Management Notation*: esta é a notação padrão de modelagem de processos de negócio. O principal objetivo deste padrão é de usar uma notação gráfica comum para a modelagem de processos de negócio independentemente das ferramentas e aplicações usadas para a modelagem;
- Wf-XML - *Workflow XML*: este fornece a interoperabilidade entre mecanismos da BPM, tornando-a possível executar um processo de negócio por um largo período de tempo que abrange vários mecanismos;
- XPDL - *XML Process Definition Language*: é uma linguagem de definição de processos de negócio que descreve um processo inteiro, e pode ser utilizado para integrar componentes de BPM para a modelagem de processos, execução e controle dentro de um produto. É também amplamente utilizado em produtos de BPM de código aberto.

A vertente do negócio dentro da organização dita a gestão por processos e a de TI cada vez mais caminha em direção à arquitetura orientada a serviços. Assim, sistemas BPMS fazem conexões através da SOA, e a área de TI trabalha nos serviços com registros e outros meios de conectá-los [Jedd, 2007].

## 2.6. Considerações Finais

Depois de entrar em contato com o universo da Qualidade Total, percebe-se que a razão de ser de uma empresa são os seus clientes. Portanto, toda sua administração deve estar voltada para a qualidade, que é a busca contínua da satisfação das necessidades dos clientes e está claro que a empresa é o meio para atingir a satisfação das necessidades de todas as pessoas (clientes, acionistas, empregados e vizinhos).

Acontece que as necessidades das pessoas mudam continuamente e os concorrentes estão se desenvolvendo e melhorando. Ninguém pode parar e esperar. Diante deste quadro, para que a empresa possa sobreviver é necessário desenvolver novos produtos ou serviços (melhores, mais baratos, mais seguros, de entrega mais rápida, de manutenção mais fácil etc. que os concorrentes). Para produzir estes novos produtos ou serviços são necessários novos processos (melhores, mais fáceis, de menor dispersão, mais baratos, mais rápidos, mais seguros etc.). Este processo de inovação tem como referências o cliente e os concorrentes e se constitui na garantia da própria sobrevivência da empresa.

03

# MR-MPS-SW E SUA COMPARAÇÃO COM O CMMI-DEV

*Jussara Adolfo Moreira*

Muitas organizações que desenvolvem software estão optando por utilizar modelos de qualidade em seus processos para otimizar as atividades de desenvolvimento, aumentar a produtividade, minimizar esforço e custo, e certificar a empresa com um modelo de maturidade de software. O objetivo principal dessas organizações é aumentar a eficiência e efetividade das soluções de software desenvolvidas para apoiar as necessidades de clientes e dos mercados.

As organizações de software devem se adaptar para enfrentar mudanças, pois os clientes percebem suas necessidades na medida em que o software é delineado e construído, além da constante evolução das tecnologias, e com isso garantir o sucesso do negócio. Da mesma forma, para melhorar sua habilidade de inovação e eficiência dos processos organizacionais, as organizações devem melhorar continuamente sua capacidade de desenvolver software. Para atingir este objetivo, organizações necessitam ser mais produtivas, aumentar a qualidade de produtos de software, diminuir o esforço e

custos e lidar com o *time-to-market* para produtos comerciais [Pfleeger, 2001].

Neste contexto, diversos modelos e normas internacionais de qualidade de processos de software foram definidos para atender as necessidades das empresas em melhoria de processos de software ISO/IEC 12207 [ISO/IEC, 2008], ISO/IEC 15504 [ISO/IEC, 2003a], CMMI-DEV [SEI, 2010].

No contexto nacional, o modelo de Referência para Melhoria de Processo do Software Brasileiro (MR-MPS) [SOFTEX, 2012a] foi definido com base nestes modelos e normas, objetivando disseminar o uso de normas e modelos de qualidade de forma mais acessível às empresas brasileiras. Diversas empresas do Brasil adotaram o modelo a partir do trabalho prestado por Instituições Implementadoras (II) devidamente credenciadas pelo Fórum de Credenciamento e Controle do MR-MPS.

Várias organizações estão optando em realizar a implantação do MPS.BR-SW e depois realizar a migração para o CMMI-DEV. Este fato acontece por vários motivos entre eles destaca-se o tempo de implantação que se torna menor e com isso o custo de implantação também diminui, além de existir no Brasil incentivos financeiros de empresas de fomento para minimizar o custo por aderir a um modelo de qualidade. Avaliações conjuntas MPS/CMMI-DEV podem ser adotadas pelas organizações para otimizar o tempo e o esforço do processo [SOFTEX, 2012b] uma vez que a implementação do modelo MPS possui compatibilidade com o modelo CMMI-DEV. Estima-se que a implantação e avaliação do modelo no Nível F (Gerenciado) em empresas no modo cooperado, e sem nenhum processo de melhoria adotado anteriormente, requer um esforço de aproximadamente um ano e seis meses, com custos inferiores ao CMMI [SOFTEX, 2013].

## 3.1. MR-MPS-SW

O MPS.BR é um programa mobilizador, de longo prazo, criado em dezembro de 2003, coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), que conta com apoio do Ministério da Ciência, Tecnologia e Inovação (MCTI), Financiadora de Estudos e Projetos (FINEP), Serviço Brasileiro de Apoio às Micro e Pequenas Empresas (SEBRAE) e Banco Interamericano de Desenvolvimento (BID/FUMIN) [SOFTEX, 2012a].

O programa MPS.BR surgiu com o objetivo de buscar a Melhoria de Processo de Software e Serviços com duas metas a alcançar a médio e longo prazos: Uma meta técnica e outra de negócio. A meta técnica visa a criação e melhoria do modelo MPS, e a meta de negócio visa disseminação do modelo, de modo que diversas organizações brasileiras, em todas as regiões do país, de pequeno, médio e grande porte, possam conhecer e adotar o modelo a um custo razoável, sejam estas públicas ou privadas [SOFTEX, 2012b]. O modelo vem ganhando repercussão não apenas no Brasil, mas também na América Latina, pois este modelo possui grande potencial de replicabilidade em outros países que possuem características similares no que se refere ao setor de software [Weber *et al.*, 2004a e 2004b].

O programa MPS.BR conta com uma unidade de execução do programa (UEP), responsável por executar o programa do modelo de qualidade e duas estruturas de apoio para a execução de suas atividades: o Fórum de Credenciamento e Controle (FCC) e a Equipe Técnica do Modelo (ETM). O FCC emite parecer que subsidie decisão da SOFTEX sobre o credenciamento e monitoração de Instituições Implementadoras e Instituições Avaliadoras; podendo emitir parecer sobre descredenciamento

se for o caso. O ETM apoia a SOFTEX sobre os aspectos técnicos, criação e aprimoramento relacionados ao Modelo de Referência MPS para Software (MR-MPS-SW), Modelo de Referência MPS para Serviços (MR-MPS-SV) e Método de Avaliação (MA-MPS), para: além da capacitação de pessoas por meio de cursos, provas e workshops [SOFTEX, 2012b].

Através destas estruturas de apoio o MPS.BR pode contar com a participação de representantes de universidades, instituições governamentais, centros de pesquisa e de organizações privadas, os quais contribuem com suas visões complementares que agregam valor e qualidade ao programa [SOFTEX, 2012b]. A participação destes torna o processo mais robusto, pois agrega vários setores com sua visão de qualidade.

O MR-MPS-SW contém a definição dos níveis de maturidade, dos processos e dos atributos do processo relacionados a cada nível de maturidade. A base técnica para a construção e o aprimoramento do modelo é composta pelas normas ISO/IEC 12207 [ISO/IEC, 2008] e ISO/IEC 15504 [ISO/IEC, 2003a].

A iniciativa para elaboração deste modelo teve como objetivo aumentar a qualidade do processo das empresas de desenvolvimento de software, de forma que pequenas e médias empresas, possam melhorar seus processos com um custo inferior ao que seria necessário para implantar o CMMI-DEV, com possibilidades de conseguir incentivos financeiros do BID, BNDES e do SEBRAE para implantação do modelo MPS.BR. O recurso financeiro fornecido pelos Bancos visam o desenvolvimento do país e devem ser utilizados para obter a certificação, caso a empresa não alcance a certificação, o dinheiro deve ser devolvido. O SEBRAE aporta o recurso, mas a empresa tem que pagar 10% do valor. De modo que essas empresas, de forma acessível, possam implantar seus modelos de qualidade e possuir certificado de qualidade em seus processos.

O progresso e o alcance de um determinado nível de maturidade do MR-MPS-SW obtêm-se quando são atendidos os resultados esperados dos processos e dos atributos de processos estabelecidos para aquele nível, como pode ser visto na Figura 4. Os atributos de processo (AP) são uma característica mensurável da capacidade do processo. Embora não seja detalhado no processo, para atender aos AP é necessário atender aos resultados esperados dos atributos do processo (RAP) para todos os processos correspondentes ao nível de maturidade [SOFTEX, 2012a].

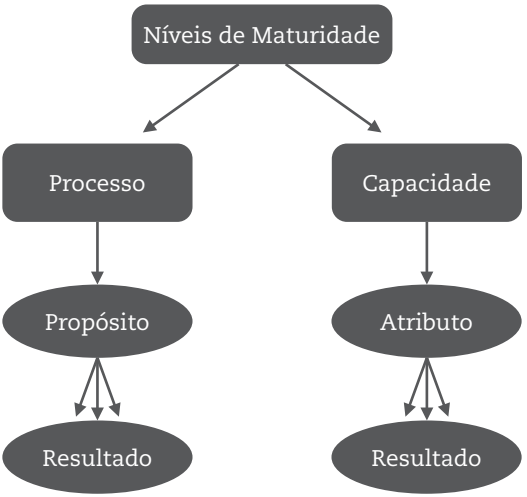



**Figura 4.** Estrutura do MR-MPS-SW [SOFTEX 2012a]

A Figura 4 mostra a estrutura do modelo MPS.BR, composta por níveis de maturidade, onde cada nível é uma combinação dos processos e da capacidade dos processos. Os processos são descritos segundo o propósito e os resultados esperados. O propósito descreve o objetivo a ser atingido com a execução do processo e os resultados esperados estabelecem os que

devem ser obtidos com a efetiva implementação do processo. Já a capacidade do processo é representada por um conjunto de atributos descritos em termos de resultados esperados [SOFTEX, 2012a].

O níveis de maturidade do MPS.BR são descritos na Tabela 3 conforme seus processos e atributos de processo. O modelo possui uma escala de maturidade que é dividida em sete níveis, começa no nível G e evolui até o nível A, quando a organização atinge a alta maturidade. Para cada processo ser atingido é necessário que os resultados esperados dos atributos do processo sejam evidenciados. Os atributos do processo do nível anterior são requeridos nos níveis posteriores. Então, para uma empresa estar no nível F é necessário possuir os atributos de processo do nível G e F. Os níveis que vão demandar maior esforço são os níveis G e F, pois requer uma adaptação no processo da empresa para atender os requisitos estabelecidos neste modelo.

**Tabela 3.** Níveis de Maturidade do MR-MPS-SW [SOFTEX, 2012a]

| Nível de maturidade | Processo | Atributos de Processo |
|---|---|---|
| A | - | AP 1.1, AP 2.1, AP 2.2, AP 3.1, AP 3.2, AP 4.1, AP 4.2, AP 5.1 e AP 5.2 |
| B | Gerência de Projetos - GPR (evolução) | AP 1.1, AP 2.1, AP 2.2, AP 3.1, AP 3.2, AP 4.1, e AP 4.2 |
| C | Gerência de Riscos - GRI | AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2 |
| | Desenvolvimento para Reutilização - DRU | |
| | Gerência de Decisões - GDE | |

| Nível de maturidade | Processo | Atributos de Processo |
|---|---|---|
| D | Verificação - VER | AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2 |
| | Validação - VAL | |
| | Projeto e Construção do Produto - PCP | |
| | Integração do Produto - ITP | |
| | Desenvolvimento de Requisitos - DRE | |
| E | Gerência de Projetos - GPR (evolução) | AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2 |
| | Gerência de Reutilização - GRU | |
| | Gerência de Recursos Humanos - GRH | |
| | Definição do Processo Organizacional - DFP | |
| | Avalização e Melhoria do Processo Organizacional - AMP | |
| F | Medição - MED | AP 1.1, AP 2.1 e AP 2.2 |
| | Garantia da Qualidade - GQA | |
| | Gerência de Portfólio de Projetos - GPP | |
| | Gerência de Configuração - GCO | |
| | Aquisição - AQU | |
| G | Gerência de Requisitos - GRE | AP 1.1 e AP 2.1 |
| | Gerência de Projetos - GPR | |

O MR-MPS-SW [SOFTEX, 2012a] define sete níveis de maturidade, sequenciais e cumulativos, a seguir: A (Em Otimização), B (Gerenciado Quantitativamente), C (Definido), D (Largamente Definido), E (Parcialmente Definido), F

(Gerenciado) e G (Gerenciado Parcialmente). Se comparado ao CMMI-DEV [SEI, 2010], o modelo possui três níveis avaliáveis a mais, o que permite melhor atender às médias, pequenas e microempresas, que poderão alcançar os objetivos de melhoria em etapas intermediárias, fornecendo mais visibilidade do progresso [SOFTEX, 2012b]. Cada nível de maturidade possui atributos do processo a serem atingidos. Cada atributo do processo é requerido nos níveis do modelo e são descritos a seguir na Tabela 4.

Cada atributo do processo possui uma descrição e um propósito. O intuito da descrição é determinar o objetivo a ser atingido com a execução do processo e o propósito evidenciar os resultados esperados que devem ser obtidos com a implementação do processo.

**Tabela 4.** Atributos de Processo [SOFTEX, 2012a]

| Atributo de Processo | Descrição | Propósito |
|---|---|---|
| AP 1.1 | O processo é executado | Este atributo evidencia o quanto a execução do processo é gerenciada. |
| AP 2.1 | O processo é gerenciado | Este atributo evidencia o quanto a execução do processo é gerenciada. |
| AP 2.2 | Os produtos de trabalho do processo são gerenciados | Este atributo evidencia o quanto os produtos de trabalho produzidos pelo processo são gerenciados apropriadamente. |
| AP 3.1 | O processo é definido | Este atributo evidencia o quanto um processo padrão é mantido para apoiar a implementação do processo definido. |
| AP 3.2 | O processo está implementado | Este atributo evidencia o quanto o processo padrão é efetivamente implementado como um processo definido para atingir seus resultados. |

| Atributo de Processo | Descrição | Propósito |
|---|---|---|
| AP 4.1 | O processo é medido | Este atributo evidencia o quanto os resultados de medição são usados para assegurar que a execução do processo atinge os seus objetivos de desempenho e apoia o alcance dos objetivos de negócio definidos. |
| AP 4.2 | O processo é controlado | Este atributo evidencia o quanto o processo é controlado estatisticamente para produzir um processo estável, capaz e previsível dentro de limites estabelecidos. |
| AP 5.1 | O processo é objeto de melhorias incrementais e inovações | Este atributo evidencia o quanto as mudanças no processo são identificadas a partir da análise de defeitos, problemas, causas comuns de variação do desempenho e da investigação de enfoques inovadores para a definição e implementação do processo. |
| AP 5.2 | O processo é otimizado continuamente | Este atributo evidencia o quanto as mudanças na definição, gerência e desempenho do processo têm impacto efetivo para o alcance dos objetivos relevantes de melhoria do processo. |

## 3.2. CMMI-DEV

O CMMI-DEV [SEI 2010] é um modelo de maturidade e capacidade de processos de software criado pelo SEI – *Software Engineering Institute* e que consiste de boas práticas de engenharia de software para o desenvolvimento e manutenção de produtos e serviços.

O CMMI possui cinco níveis de maturidade começa no nível 1 e evolui até o nível 5, quando a organização atinge a alta maturidade. Para cada área de processo ser atingida é necessário

que os resultados esperados dos objetivos específicos e genéricos sejam evidenciados. As áreas de processo do nível anterior são requeridos nos níveis de posteriores. Então para uma empresa estar no nível 3 é necessário evidenciar as áreas de processo do nível 1 e 2, como pode ser visto na Tabela 5.

**Tabela 5.** Níveis de capacidade e de maturidade do CMMI-DEV [SEI 2010]

| Nível | Nível de capacidade (representação contínua) | Nível de maturidade (representação por estágios) |
|---|---|---|
| 0 | Incompleto | - |
| 1 | Realizado | Inicial |
| 2 | Gerenciado | Gerenciado |
| 3 | Definido | Definido |
| 4 | - | Gerenciado quantitativamente |
| 5 | - | Em otimização |

O CMMI-DEV [SEI, 2010] possui dois tipos de representação: contínua e por estágios. Na representação contínua, as áreas de processos são organizadas em categorias e a implementação da melhoria ocorre por níveis de capacidade, enquanto que na representação por estágios, as áreas de processos são organizadas em níveis de maturidade. Na primeira, as áreas de processos podem ser avaliadas individualmente, segundo a estratégia e objetivos de negócio da organização. Já na representação por estágios, a avaliação é realizada em todas as áreas de processos que compõem o nível de maturidade selecionado pela organização [SOFTEX, 2012b].

O CMMI-DEV é formado por componentes agrupados em três categorias: componentes requeridos, componentes esperados e componentes informativos, que auxiliam na interpretação do modelo, conforme apresentado na Figura 5.

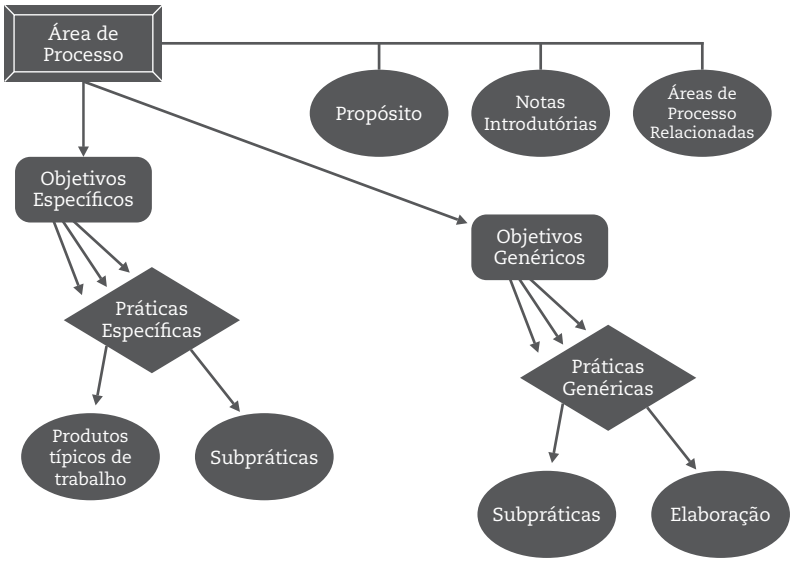



**Figura 5.** Componentes do CMMI-DEV [SEI, 2010]

Cada área de processo possui objetivos específicos e objetivos genéricos a serem atingidos. Os objetivos específicos são atingidos a partir dos resultados esperados das práticas específicas e os objetivos genéricos são atingidos a partir dos resultados esperados das práticas genéricas.

O propósito, os objetivos específicos – relacionados ao processo – e os objetivos genéricos – relacionados a todos os processos e à organização – compõem as áreas de processo. A função dos objetivos específicos é definir as características únicas que devem estar presentes para que uma determinada área de processo seja satisfeita. Por sua vez, os objetivos genéricos estão associados a mais de uma área de processo e definem as características que devem estar presentes para institucionalizar os processos que implementam a área de processo [SEI, 2010].

Em processos de avaliação da implementação do CMMI-DEV [SEI, 2010], a equipe de avaliação visa buscar evidências de que os objetivos e práticas, específicos e genéricos, estão atendidos nos projetos, para as respectivas áreas de processos avaliadas.

O CMMI-DEV [SEI 2010] é composto por 22 áreas de processos, que podem ser observadas na Tabela 6, com os respectivos níveis de maturidade e categorias [SOFTEX 2012b].

**Tabela 6.** Áreas de Processo do CMMI-DEV [SEI 2010]

| Nível de maturidade | Área de Processo | Categoria |
|---|---|---|
| 2 | Monitoração e Controle do Projeto (PMC) | Gerência de Projeto |
| 2 | Planejamento do Projeto (PP) | Gerência de Projeto |
| 2 | Gerência de Requistos (REQM) | Gerência de Projeto |
| 2 | Análise e Medição (MA) | Apoio |
| 2 | Garantia da Qualidade do Processo e do Produto (PPQA) | Apoio |
| 2 | Gerência de Configuração (CM) | Apoio |
| 3 | Gerência de Fornecedor Integrada (SAM) | Gerência de Projeto |
| 3 | Gerência de Projeto Integrada (IPM) | Gerência de Projeto |
| 3 | Gerência de Riscos (RSKM) | Gerência de Projeto |
| 3 | Definição do Processo Organizacional (OPD) | Gerência de Processo |
| 3 | Foco no Processo Organizacional (OPF) | Gerência de Processo |
| 3 | Treinamento Organizacional (OT) | Gerência de Processo |
| 3 | Desenvolvimento de Requisitos(RD) | Engenharia |

| Nível de maturidade | Área de Processo | Categoria |
|---|---|---|
| 3 | Integração do Produto (PI) | Engenharia |
| 3 | Solução Técnica (TS) | Engenharia |
| 3 | Validação (VAL) | Engenharia |
| 3 | Verificação (VER) | Engenharia |
| 3 | Análise de Decisão e Resolução (DAR) | Apoio |
| 4 | Gerência Quantitativa do Projeto (QPM) | Gerência de Projeto |
| 4 | Desempenho do Processo Organizacional (OPP) | Gerência de Processo |
| 5 | Gerência do Desempenho Organizacional (OPM) | Gerência de Processo |
| 5 | Resolução e Análise Causal (CAR) | Apoio |

## 3.3. Comparação do MR-MPS-SW com o CMMI-DEV

O MR-MPS-SW define níveis de maturidade que são uma combinação de processos e atributos de processo. Processos estão descritos através de seu propósito e resultados esperados do processo, enquanto que os atributos de processo estão descritos através de resultados de atributos de processos (RAP). Os componentes requeridos no MR-MPS-SW são os Resultados esperados de processos e os Resultados de atributos de processos.

O CMMI-DEV está organizado em áreas de processos, com objetivos e práticas específicos, e em objetivos e práticas genéricos. No CMMI-DEV, os componentes são agrupados em três categorias: requeridos, esperados e informativos. Em uma avaliação, os componentes requeridos e esperados do

modelo CMMI-DEV têm o mesmo impacto dos resultados esperados do processo e dos resultados de atributos de processos do MR-MPS-SW, que obrigatoriamente devem estar atendidos pela organização avaliada.

A Figura 6 ilustra a árvore estrutural dos dois modelos, as similaridades entre os elementos avaliados nos níveis de maturidade do MR-MPS-SW e CMMI-DEV. Com relação à maturidade, os processos do MR-MPS-SW são compatíveis com as áreas de processo do CMMI-DEV. Os resultados esperados dos processos do MR-MPS-SW, são comparados com as práticas específicas das áreas de processo do CMMI-DEV. Com relação à capacidade, os resultados de atributos de processos do MR-MPS-SW, são comparados com as práticas genéricas do CMMI-DEV.

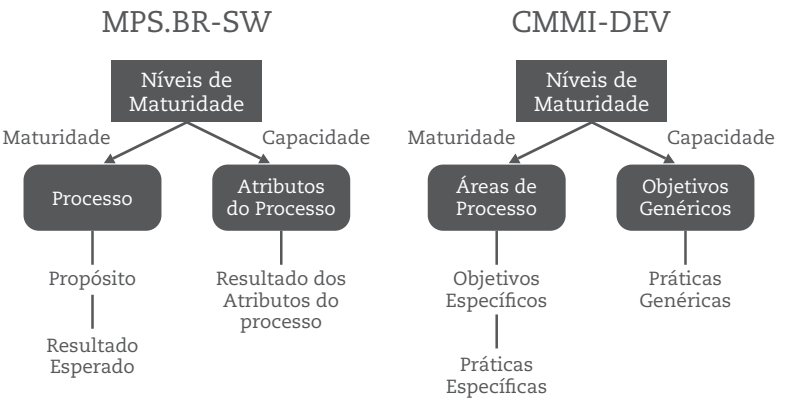



**Figura 6.** Comparação entre as Estruturas do MR-MPS-SW e CMMI-DEV

A Tabela 7 apresenta uma visão geral dos processos do MR-MPS-SW e seus correspondentes no modelo CMMI-DEV. A Gerência de Portfólio, Gerência de Reutilização e Desenvolvimento para reutilização são processos que não existem no CMMI-DEV.

**Tabela 7.** Processos do MPS-SW e Áreas de Processos do CMMI-DEV [SOFTEX, 2012b]

| Processos do MR-MPS-SW | Áreas de Processos do CMMI-DEV |
|---|---|
| Gerência de Projetos | Planejamento de Projeto |
| | Monitoração e Controle de Projeto |
| | Gestão Integrada de Projeto |
| | Gestão Quantitativa de Projeto |
| Gerência de Requisitos | Gestão de Requisitos |
| Aquisição | Gestão de Contrato com Fornecedores |
| Gerência de Configuração | Gestão de Configuração |
| Garantia da Qualidade | Garantia da Qualidade de Processo e Produto |
| Gerência de Portfólio de Projetos | - |
| Medição | Medição e Análise |
| Avaliação e Melhoria do Processo Organizacional | Foco nos Processos da Organização |
| Definição do Processo Organizacional | Definição dos Processos da Organização |
| Gerência de Recursos Humanos | Tratamento na Organização |
| Gerência de Reutilização | - |
| Desenvolvimento de Requisitos | Desenvolvimento de Requisitos |
| Integração do Produto | Integração de Produto |
| Projeto e Construção do Produto | Solução Técnica |
| Validação | Validação |
| Verificação | Verificação |
| Desenvolvimento para Reutilização | - |

| Processos do MR-MPS-SW | Áreas de Processos do CMMI-DEV |
|---|---|
| Gerência de Decisões | Análise e Tomada de Decisões |
| Gerência de Riscos | Gestão de Riscos |
| - | Análise e Resolução de Causas |
| - | Gerência do Desempenho Organizacional |
| - | Desempenho dos Processos da Organização |

As áreas de processo do modelo CMMI-DEV que não têm um processo correspondente foram mapeadas por possuir correspondência com resultados de atributos de processos do modelo MPS, o que está refletido no mapeamento definido em [SOFTEX, 2012b].

No CMMI existe a representação por estágios e representação contínua. Segundo a documentação do MPS.BR, o modelo trabalha apenas com representação por estágios, podendo acontecer certificações por áreas de processo, nos casos onde a empresa estiver realizando algum nível de certificação. No MR-MPS-SW seria necessário implementar o nível inteiro e selecionar quais processos a empresa deseja adicionar. No CMMI é diferente, pois não é necessário avaliar todo o nível, e sim apenas escolher quais áreas de processos seriam avaliados.

No MPS a empresa escolhe o nível a ser implementado, e se esta desejar pode determinar um ou mais processos de níveis posteriores. Então, esta pode receber a certificação do nível pretendido e do(s) processo(s) solicitados a mais.

O Quadro 1 expõe a compatibilidade entre os níveis de maturidade do MR-MPS-SW e do CMMI-DEV, que possibilita a migração do primeiro para o segundo, realizando adequações

nos processos implantados. Resende (2009) apresenta um relato de experiência de uma empresa brasileira que implantou de forma simultânea o modelo MR-MPS-SW no nível F e o CMMI-DEV no nível 2. Segundo o autor, a migração entre os modelos acorreu devido à compatibilidade já estabelecida e os recursos financeiros disponíveis.

**Quadro 1.** Mapeamento entre níveis de Maturidade MPS e CMMI [Guedes 2013]

| Nível MPS | G | F | E | D | C | B | A |
|---|---|---|---|---|---|---|---|
| Nível CMMI | - | 2 | - | - | 3 | 4 | 5 |

Para a realização das avaliações conjuntas e complementares do MR-MPS-SW e do CMMI-DEV foram geradas planilhas específicas (como visto na Figura 7), definidas a partir da planilha oficial de avaliação MPS, adaptadas para inclusão do componente correspondente do modelo CMMI-DEV, da classificação definida no mapeamento e das considerações relacionadas. Assim, foi mantida a mesma estrutura da planilha oficial MPS, com introdução de informações resultantes do mapeamento realizado [SOFTEX, 2012b].

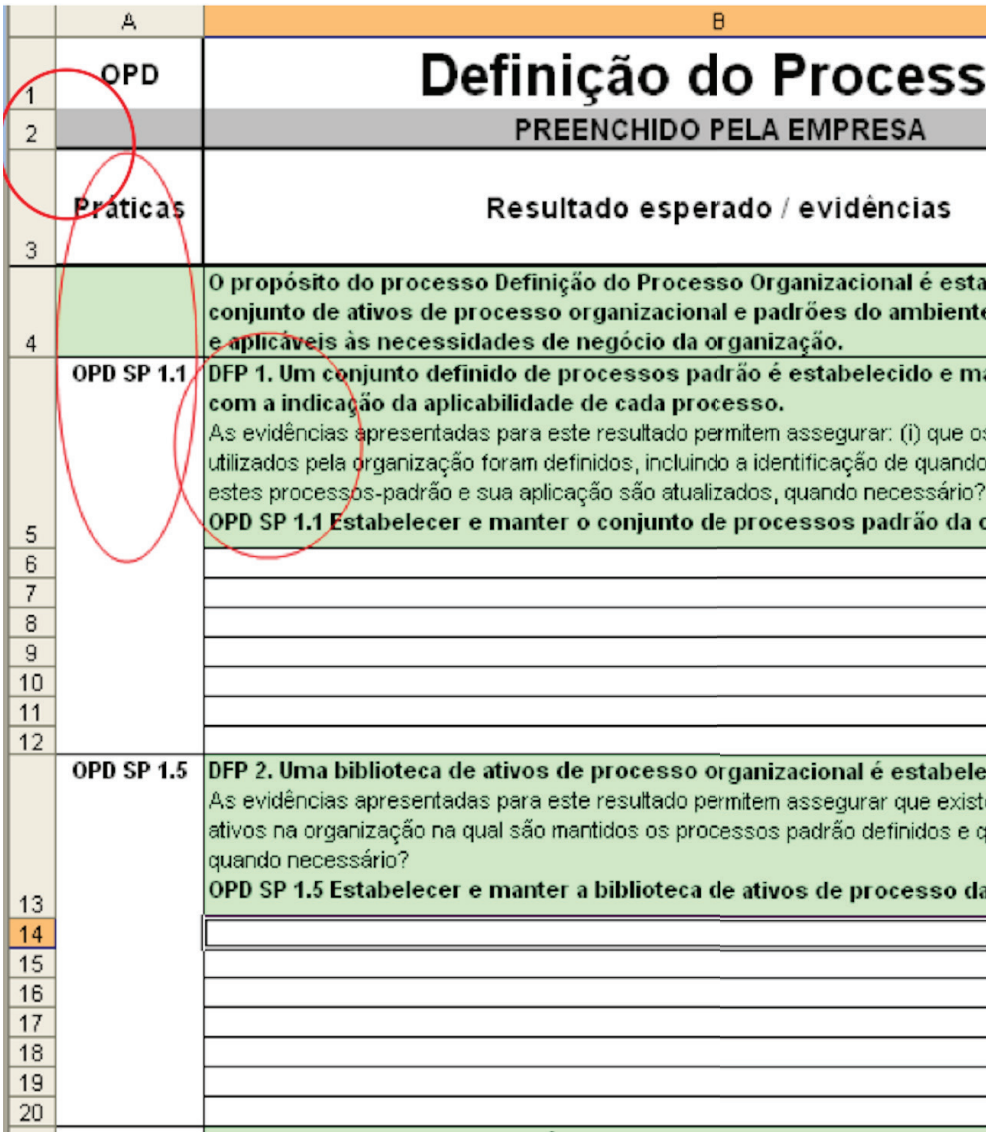
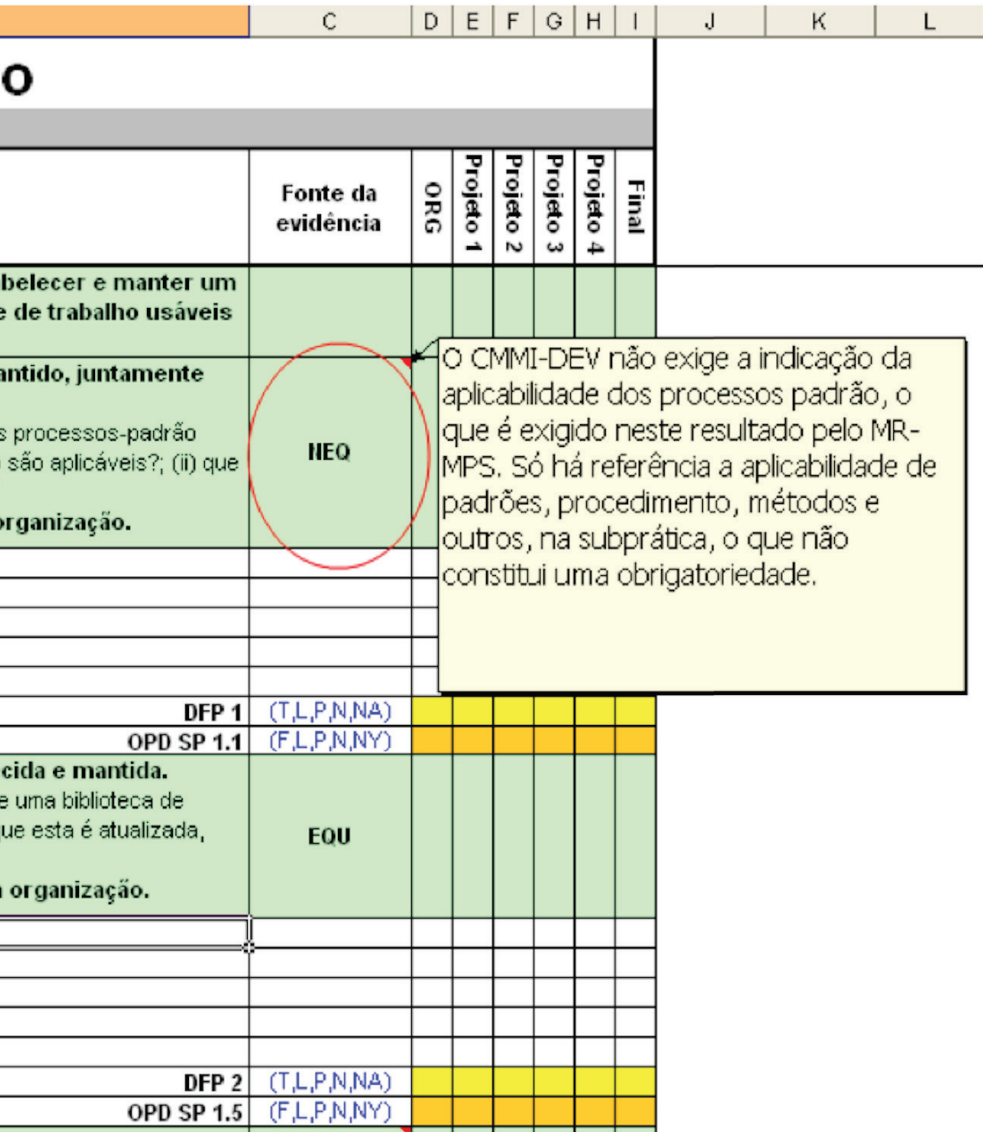



**Figura 7.** Planilha para Avaliação Conjunta MPS-SW/CMMI-DEV [SOFTEX, 2012b]



| | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|
| o | | | | | | | | | | |
| | Fonte da evidência | ORG | Projeto 1 | Projeto 2 | Projeto 3 | Projeto 4 | Final | | | |
| belecer e manter um e de trabalho usáveis | | | | | | | | | | |
| antido, juntamente s processos-padrão são aplicáveis?; (ii) que rganização. | NEQ | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| DFP 1 | (T,L,P,N,NA) | | | | | | | | | |
| OPD SP 1.1 | (F,L,P,N,NY) | | | | | | | | | |
| cida e mantida. e uma biblioteca de ue esta é atualizada, organização. | EQU | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| DFP 2 | (T,L,P,N,NA) | | | | | | | | | |
| OPD SP 1.5 | (F,L,P,N,NY) | | | | | | | | | |

O CMMI-DEV não exige a indicação da aplicabilidade dos processos padrão, o que é exigido neste resultado pelo MR-MPS. Só há referência a aplicabilidade de padrões, procedimento, métodos e outros, na subprática, o que não constitui uma obrigatoriedade.

Nas áreas em destaque da planilha para avaliações conjuntas (Figura 7) são apresentados: a área de processo (célula A1), a sigla e a declaração da prática específica do CMMI-DEV correspondente ao resultado esperado do processo no MR-MPS-SW (células A5 e A13 e células B5 e B13, respectivamente), o critério de classificação da comparação (células C5 e C13) e as considerações relacionadas, que foram incluídas na planilha por meio de comentários [SOFTEX, 2012b].

Inicialmente o modelo de qualidade é implementado, depois avaliado. Na implementação do modelo, em todos os processos que serão avaliados para o nível requerido, é necessário registrar as evidências de cada projeto na planilha, gerando assim um *hiperlink* para o(s) documento(s) que comprova(m) a evidência.

São escolhidos 4 projetos de acordo com o grau de relevância para organização, geralmente são 2 concluídos e 2 em andamento. O avaliador escolhe em conjunto com a organização.

O processo de avaliação consiste em uma avaliação inicial e outra final. Na avaliação inicial o avaliador fica em uma sala analisando as evidências registradas. A avaliação inicial no CMMI-DEV, não obrigatória, chama-se *readiness* e a final chama-se de avaliação oficial.

Na avaliação inicial estas evidências são verificadas. O avaliador marca a célula da planilha com uma das cores: verde (adequado), amarelo (precisa ser melhorado) ou vermelho (Não aceito). Todas as células marcadas com as cores: amarelo e vermelho são deixados como pontos requeridos. A empresa tem de 15 dias a 6 meses para ajustar os pontos nos projetos e corrigir todos os pontos fracos.

Na avaliação final o avaliador vai verificar se todos os pontos amarelos e vermelhos precisam ser revistos pois eles precisam tornar-se verdes, precisam estar adequados. Se na

avaliação final houver uma célula vermelha, o processo de avaliação final é extinto, pois não pode existir pontos fracos na avaliação final. Caso todas as células estejam verdes, o avaliador deve definir o grau de implementação.

A avaliação final é realizada com entrevistas para evitar que as evidências tenham sido fraudadas. Todos os funcionários são reunidos para que os avaliadores os entrevistem sobre os processos da empresa, identificando assim que as evidências foram realmente produzidas pelo grupo. Se algo divergir na entrevista a avaliação pode ser impugnada.

O objetivo do avaliador nas entrevistas vai identificar a veracidade das evidências geradas e recomendar à alta administração o que o nível operacional está solicitando.

A implantação do modelo MR-MPS-SW ao ser comparado com o CMMI-DEV torna-se bem mais simples de ser implementado, pois o MPS possui mais níveis minimizando o esforço no processo inicial de implementação do modelo.

Na verdade o MPS.BR não é um novo modelo, o foco dele é fazer com que os modelos que já existem no mercado possam ser implementados no Brasil de maneira mais acessível de forma que as empresas nacionais possam implementar. O MPS.BR é um programa para fazer com que os modelos que já existentes sejam de fato utilizados no Brasil a custo bem inferior.

## 3.4. Considerações Finais

É possível perceber que os modelos são totalmente compatíveis, e que a iniciativa do modelo MR-MPS-SW possibilita que as empresas brasileiras e da América Latina possam aumentar a qualidade de seus processos com um custo inferior ao do modelo CMMI, e que existem recursos financeiros que

podem ser alcançados pelas empresas Brasileiras para a implantação do modelo sem que haja devolução do valor, pois o objetivo dos financiadores é o desenvolvimento do país.

04

# CMMI-SERVIÇOS

*Raquel Godoi Amaral Ferraz*

Serviço pode ser definido como o conjunto de componentes relacionados entre si que são ofertados para suportar um ou mais processos de negócio, também relativos ao que o usuário interage.

A gestão de serviços é organizada por processo, ferramentas e pessoas, como se pode verificar na Figura 8, que colaboram para garantir que os serviços sejam atendidos de acordo com o contrato estabelecido entre prestador e cliente. A partir da definição bem sucedida dos processos que envolverão a gestão de serviços é importante estabelecer indicadores que permitam o gestor medir, controlar e traçar estratégias a partir dos resultados. Observa-se também que a junção de tecnologia com serviços é algo que proporciona maior independência e sucesso na execução dos serviços, mas são as pessoas que representam a base para que o serviço seja efetivamente adequado aos consumidores.

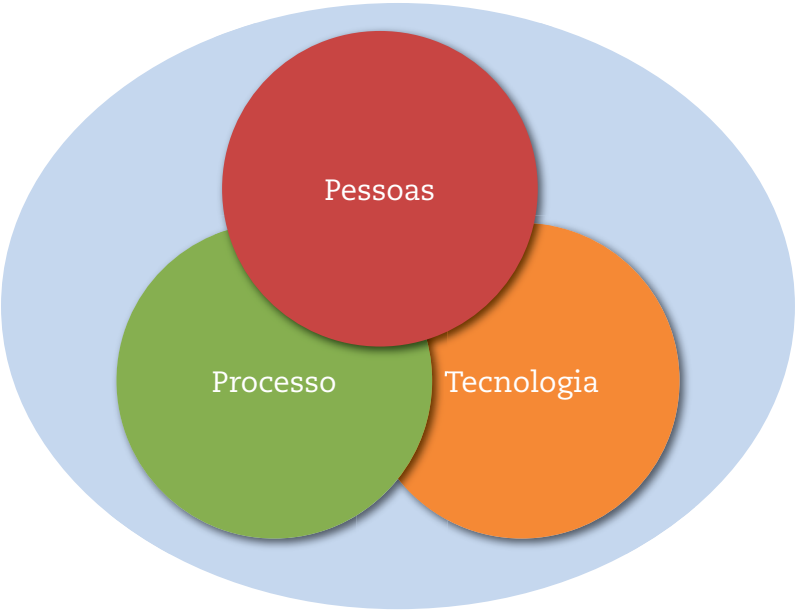



**Figura 8.** Base da Gestão de Serviços.

Um ponto importante da gestão de serviços é a diferença entre cliente e usuário. O cliente busca o serviço e paga por ele. Já o usuário é a pessoa que utiliza os serviços oferecidos.

Processo representa o conjunto de atividades que se relacionam entre elas com algum objetivo específico. Para estabelecer um processo é necessário estabelecer as entradas, informações, ferramentas, envolvidos, itens ou artefatos de saída. O processo pode ser estabelecido da seguinte forma:

- Estabelecer o dono do projeto;
- Estabelecer tarefas para atingir o objetivo;
- Incluir entradas e saídas nas tarefas;
- As tarefas serão executadas por uma função, que pode ser humana ou utilizando tecnologia;
- Definir regras da execução das funções.

Com o uso do gerenciamento de serviços é preciso instituir metas, métricas e objetivos para a organização, garantindo que este gerenciamento passe por ciclos de melhoria e que seja contínua. A cada novo ciclo devem ser avaliados os objetivos, as métricas e as metas estabelecidas e gerar novos insumos para garantir que o processo esteja de acordo com o esperado e trazendo melhorias para a organização.

De acordo com o CMMI Institute, o setor de serviços é um importante motor para o crescimento econômico a nível mundial. O modelo CMMI para serviços (CMMI-SVC) é orientado ao desenvolvimento e as melhores práticas de serviços maduros [CMMI Institute, 2010]. O CMMI-SVC é um modelo projetado para satisfazer a necessidade atual.

Segundo Kotler (1993), o serviço pode ser definido como qualquer ato do desempenho essencialmente intangível que uma parte pode oferecer a outra e que não tem como resultado a propriedade de algo. A execução de um serviço pode estar ou não ligada a um profundo físico.

A necessidade pela qualidade de produtos e serviços, decorrente quase sempre do aumento de concorrência de variadas naturezas, motivou uma transformação radical no cenário e, consequentemente, na atitude dos gestores, sobretudo quando se descobriu que a decisão gerencial entre produzir ou produzir com qualidade estava sendo substituída pela decisão estratégica de produzir com qualidade ou por em risco a sobrevivência da organização [Paladini, 2012].

Este capítulo apresenta o modelo de qualidade CMMI-SVC V1. 3 desde o surgimento do modelo, contextualizando com exemplos vivenciados nos serviços públicos e privados, motivação para adotá-lo e os benefícios que a organização obterá caso faça uso de seus conceitos.

## 4.1. O CMMI-Serviços

Em 1930 Walter Shewhart começou a trabalhar com melhoria de processos e estabeleceu os princípios de controle estatísticos da qualidade. Esses princípios foram refinados e aplicados por Deming, Crosby e Juran. Watts Humphrey, Ron Radice, entre outros refinaram, estenderam e aplicaram estes princípios na IBM – *International Business Machines* e no SEI. No seu livro, Humphrey (1989) apresenta a descrição dos princípios e os conceitos que os modelos do CMM – *Capability Maturity Models* são baseados.

O objetivo do CMM é concentrar na melhoria dos processos da organização, fazendo com que a organização trilhe um caminho de melhoria contínua. Em 1994 o SEI criou o primeiro CMM que é designado às organizações de software, publicando o livro The Capability Maturity Model: Guidelines for Improving the Software Process [Paulk, 1994].

O CMMI para serviços [CMMI Institute, 2010] baseia-se nos conceitos e práticas do CMMI e outros padrões e modelos de serviço como o ITIL, CobiT, ISO/IEC 20000 e o ITSCMM. O modelo abrange as atividades necessárias para estabelecer, entregar e gerenciar serviços. Conforme definido no contexto do CMMI, um serviço é um produto intangível, não armazenável.

A ISO/IEC 20000 [ISO/IEC, 2011] é a norma que tem como escopo definir requisitos para o correto gerenciamento de uma organização prestadora de serviços de TI – Tecnologia da Informação para garantir a entrega aos clientes de serviços de qualidade. São requisitos da norma a definição de políticas, objetivos, procedimentos e processos de gerenciamento para garantir a qualidade e a efetividade na prestação de serviços de TI. Os processos da ISO/IEC 20000 são:

- Processos de Planejamento e implementação;

- Processos de entrega de serviços;
- Processos de relacionamento;
- Processos de solução, liberação e controle.

A ISO/IEC 20000 adota a metodologia conhecida como PDCA – *Plan, Do, Check, Action* para os processos de planejamento e implementação de serviços, que consiste de quatro tarefas básicas:

- Planejar: estabelece os objetivos e processos necessários para a entrega dos serviços de acordo com os requisitos dos clientes e das políticas da organização;
- Fazer: implementa os processos estabelecidos no plano;
- Checar: monitora e mede os processos e serviços em relação às políticas, objetivos e requisitos, e relata os resultados;
- Agir: toma medidas para melhorar continuamente o desempenho dos processos;

Metas e práticas do CMMI-SVC, portanto, são potencialmente relevantes para qualquer organização preocupada com a prestação de serviços, incluindo empresas em sectores tais como a defesa, a tecnologia da informação, saúde, finanças e transporte.

## 4.2. Processos Específicos de Serviço do Modelo

O modelo de qualidade CMMI para serviços [CMMI Institute, 2010] está estruturado com 24 processos e 4 áreas de processos que abordam o Gerenciamento de Projetos, Gerenciamento de Processo, Suporte e Estabelecimento e Entrega de Serviço. A Figura 9 apresenta a divisão dos processos que envolvem os modelos do CMMI. Percebe-se que o modelo de serviços também inclui os processos verificados no modelo CMMI-DEV. Pode-se também identificar que no relatório das organizações certificadas com CMMI para serviços no CMMI Institute,

que algumas organizações apresentam a certificação para serviços também foram certificados em desenvolvimento.

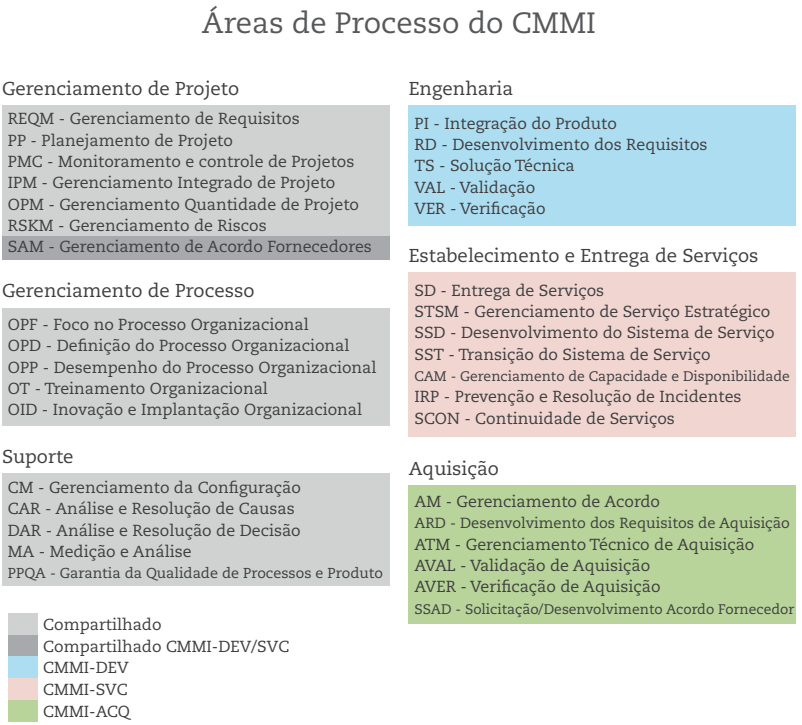



**Figura 9.** Áreas de Processo e Categorias dos modelos do CMMI.

Neste capítulo serão tratadas apenas os sete processos relativos apenas ao serviço propriamente, ou seja, os processos que não se encontram no CMMI para desenvolvimento de software. Os sete processos envolvidos são Entrega de Serviço, Gerenciamento de Serviço Estratégico, Desenvolvimento do Sistema de Serviços, Transição do Sistema de Serviços, Gerenciamento da Capacidade e Disponibilidade, Prevenção e Resolução de Incidentes e Continuidade dos Serviços.

### 4.2.1. Entrega de Serviço

Os objetivos que são esperadas para esta área de processo são contratos de serviços, preparação para a prestação de serviços e entrega do serviço. Um contrato de serviços deve ser estabelecido anteriormente à prestação de serviços, auxiliando o cliente e a organização a refinar e aprimorar os serviços prestados. Além disso, o contrato para a prestação de serviços pode ser revisado ao longo do trabalho prestado com base nos resultados do serviço oferecido.

Percebe-se que o contrato de serviços estabelecido é um documento dinâmico, que sofre alterações ao longo do tempo em que os serviços estão sendo prestados e existe o retorno dos resultados, que devem ser analisados pelo prestador de serviços e pelo cliente.

De maneira geral este processo visa garantir que sejam realizados os serviços de acordo com o que foi acordado com o cliente. As atividades envolvidas são: estabelecer e manter este acordo de serviço; receber e processar os pedidos de serviço; além de preparar e realizar os serviços acordados. As possíveis evidências que podem garantir o uso deste processo para a organização englobam o contrato, conhecido também como SLA – *Service-Level Agreement*, Pesquisa de Satisfação e *Status* para o cliente sobre os serviços estabelecidos.

### 4.2.2. Gerenciamento de Serviços Estratégicos

O objetivo do gerenciamento de serviços estratégicos é implementar e manter planos que caracterizam o negócio, onde sejam verificadas as necessidades existentes da organização e do mercado, abrangendo a capacidade e as necessidades dos clientes, juntamente com os acordos estabelecidos.

Em linhas gerais este processo visa garantir as informações necessárias para a tomada decisões estratégicas eficazes sobre o conjunto de serviços padrão que a organização mantém, onde seja extraído o máximo de proveito dos serviços para captação de negócios, redução de custos, erros e tempo para prestar os serviços, agregando valor para a organização e cliente.

As atividades que podem ser adicionadas neste processo são realizar reuniões de portfólio onde sejam verificadas pela alta gestão as oportunidades de mercado e melhorias, análise de mercado e análise da satisfação dos clientes, assim como possíveis evidências através destes canais.

### 4.2.3. Desenvolvimento do Sistema de Serviço

O propósito deste processo é realizar o mapeamento dos sistemas de serviços como análise, desenvolvimento, integração, verificação e validação, incluindo os componentes de serviços. Assim, o intuito é a formação do escopo de serviços existentes para que o fornecimento de serviços esteja alinhado com o acordo realizado.

Os componentes de serviço representam um processo, cliente, produto de trabalho ou pessoa que agregue valor ao serviço oferecido. Pode-se identificar os componentes de serviço como do cliente ou de terceiros, tomando como exemplo o componente serviço pessoa, este componente garante realizar parte do serviço para que este esteja disponível.

É necessário que ao desenvolver o sistema de serviço sejam vistas as necessidades do cliente para definir as suas expectativas. Pode-se encontrar estas necessidades em acordos de serviços, políticas organizacionais e comunicação com os usuários finais, clientes e outras partes interessadas. Além disso, abordar também o que será entregue e identificar a

hierarquia de como o serviço será atendido, analisar os atributos de qualidade que são propriedades do serviço e de serviço do sistema como capacidade de resposta, segurança e disponibilidade que são críticos para a satisfação do cliente e à satisfação das necessidades das partes interessadas.

### 4.2.4. Transição do Sistema de Serviços

Neste processo verifica-se que o objetivo é o gerenciamento da implantação de novos ou componentes significantes do sistema de serviços, enquanto os serviços oferecidos permanecem em curso.

São verificados os aspectos de planejamento, comunicação, gerenciamento e implantação para a transição dos serviços ou componentes do sistema de serviços. A transição deve garantir que seja realizada uma avaliação no ambiente do serviço, impactos sobre o sistema de serviços que será modificado ou removido para um novo sistema de serviço, impactos nos sistemas de serviços que compartilham interfaces ou recursos e os impactos da continuidade do serviço.

Elaborar um planejamento detalhado da transição do serviço é fundamental para a execução deste processo. Deve-se inserir no plano as estratégias que serão adotadas para a transição do serviço e também esclarecimentos para a parte interessada. É necessário rever os conceitos operacionais, os critérios de aceitação do sistema de serviços para garantir o trabalho efetivo, além de elaborar o plano de transição, realizar a identificação dos problemas associados à transição através da análise de como o atual sistema está operando. O plano de transição deve identificar todas as atividades e os recursos que são necessários para a transição e estar em funcionamento até que todo o processo seja concluído. As mudanças e os impactos devem ser apresentados para as partes

interessadas dos sistemas de serviço, e implementar estratégia de notificações para os envolvidos.

### 4.2.5. Gerenciamento da Capacidade e Disponibilidade

Este processo envolve a garantia que os sistemas e recursos envolvidos no serviço são fornecidos de forma eficaz. As atividades verificadas neste processo são estabelecer e manter uma estratégia de gestão de capacidade e disponibilização do fornecimento do serviço, compreender o fornecimento dos serviços atuais, reportar análises do fornecimento futuro e determinar ações corretivas para garantir a capacidade adequada (acordada). Essas atividades devem estar de acordo com um custo apropriado.

A capacidade para serviços é vista como o montante de prestação de serviços ou o número máximo de serviços que os sistemas de serviços podem atender com sucesso dentro de um período estabelecido de tempo. O contrato deve definir a capacidade e a medição, pois estes podem ser diferentes para diferentes tipos de serviços. Esse processo prevê o gerenciamento da capacidade, que tem como foco principal a definição da melhor forma de fornecer recursos para atender aos requisitos do serviço. É característica única do serviço oferecer simultaneamente disponibilidade e capacidade, quando o cliente requisita um serviço e este não tem o tempo adequado de resposta, o cliente acaba tendo de esperar e o serviço tende a trazer resultados negativos para quem oferta o serviço.

Neste processo é esperado que exista uma preparação para o gerenciamento da capacidade e disponibilidade, onde sejam verificadas as condições do sistema de serviço, entendimento da capacidade e disponibilidade acordada. Pode ser utilizado também o desempenho do sistema de serviços para mapear essas características.

### 4.2.6. Prevenção e Resolução de Incidentes

Para atender ao processo de prevenção e resolução de incidentes é necessário garantir que a resolução de incidentes seja realizada efetivamente e no tempo acordado. Algumas atividades que estão envolvidas neste processo são: identificar e analisar os incidentes; monitorar os incidentes até seu encerramento; e escalar o incidente quando necessário. Para que a prevenção dos incidentes mapeie os erros conhecidos, deve-se realizar cadastros destes e mitigar esses erros, criar uma base de dados a partir deste mapeamento.

A resolução de incidentes e sua prevenção são elaboradas geralmente antes do início da prestação de serviços e documentados em um contrato de serviços. Os incidentes podem causar ou ser indicações de interrupção de serviço. Para atuar nos incidentes de forma eficaz são necessárias definições no acordo de termos do contrato do serviço. Um sistema pode ser estabelecido para o armazenamento dos incidentes, não é obrigatório que seja automatizado, mas devem estar disponível para a organização interessada o histórico dos incidentes, suas causas e resoluções para apoiar a gestão do incidente.

A comunicação é o fator critico neste processo, onde usuários finais e o cliente bem informados podem ajudar no sucesso da resolução do incidente, além disso a equipe que trata dos incidentes pode evitar ações de resolução que possam interferir na prestação dos serviços em curso.

É importante realizar soluções alternativas para tratar os incidentes ou soluções reutilizáveis, estes pontos devem estar documentados e confirmados o seu uso eficaz antes de serem utilizados pelos prestadores de serviço.

### 4.2.7. Continuidade dos Serviços

Este processo tem o propósito de estabelecer e manter planos para garantir a continuidade dos serviços durante e após qualquer interrupção das operações normais dos serviços. Estas práticas descrevem como preparar os sistemas de serviços e os recursos dos quais dependem para ajudar a garantir que um nível mínimo de serviços pode continuar se um risco significativo acontecer.

Normalmente, a interrupção do serviço é uma situação que envolve um evento ou sequências de eventos que tornam praticamente impossível para um prestador de serviços realizar negócios como de costume.

As atividades envolvidas são baseadas nas práticas da gestão de risco, como levantamento dos potenciais perigos e ameaças para entregar o serviço. A área de processo de Gerenciamento de Riscos descreve uma abordagem sistemática em geral para identificar e mitigar todos os riscos para minimizar pró-ativamente seu impacto sobre o trabalho.

No entanto, a gestão de risco genérica não garante a continuidade dos serviços. É necessário fazer uso das praticas genéricas do processo, que são: a identificação e priorização dos recursos essenciais; o estabelecimento dos planos de continuidade de serviços; o fornecimento dos treinamentos de serviços de continuidade; a avaliação do serviço de formação da continuidade; a verificação e a validação do plano de continuidade do serviço; e a análise dos resultados de verificação e validação do plano de continuidade do serviço.

Após a elaboração dos planos de continuidade de serviço, é necessário que ocorram as suas validações, para não se testar o plano apenas quando ocorrer alguma interrupção do serviço. Os envolvidos que irão realizar os procedimentos no plano de continuidade de serviço devem ser treinados em como executar

estes procedimentos. Além disso, testes periódicos devem ser realizados para determinar se o plano de continuidade de serviço é eficaz em uma emergência real ou perturbação significativa, e que mudanças no plano são necessárias para permitir que a organização continue a fornecer um serviço confiável.

Pode-se observar no processo de gerenciamento da continuidade dos serviços as seguintes atividades:

- Proporcionar segurança ao negócio reduzindo impactos ou falhas nos serviços;
- Através do uso da gestão de risco, assegurar que seja reduzida a vulnerabilidade do serviço devido a análise realizada dos riscos;
- Prevenir perda de segurança para o cliente e usuário;
- Produzir o plano de continuidade de serviços.

## 4.3. Motivação e Benefícios do CMMI-Serviços

Encontram-se registrados no CMMI Institute um total de 407 organizações que foram certificadas até julho de 2015 no modelo CMMI-Serviços. Pode-se observar na Figura 10 que o número de organizações certificadas com o CMMI-Serviços vem crescendo ao longo dos anos.

Pode-se conjecturar que como o modelo é recente, ainda não existe uma grande aplicação se comparado ao CMMI-DEV, mas no futuro as empresas podem despertar para o uso do CMMI-Serviços como aliado ao fornecimento e prestação de serviços juntamente com o desenvolvimento. Além disso, pode-se verificar que o modelo para serviços engloba processos do CMMI-DEV estando assim coerente com as atividades que são observadas nas empresas de TI, cujo foco seja o desenvolvimento e manutenção de software, e atendam a chamados de incidentes e mudanças.

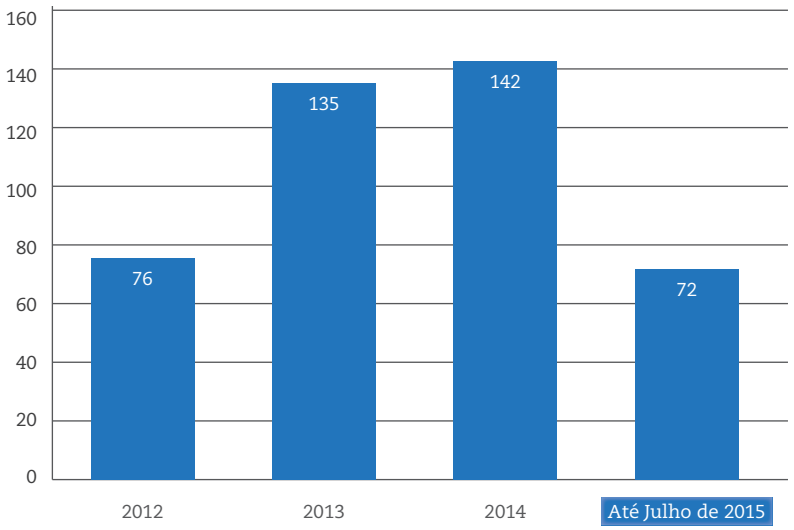



**Figura 10**. Crescimento das organizações certificadas com CMMI-SVC v1.3. Fonte: CMMI Institute

No Brasil também não tem sido diferente, existe o crescimento pela obtenção da certificação do CMMI-Serviços, a partir da Figura 11, cujos dados foram extraídos de Julho de 2015. Assim, totalizam 14 empresas certificadas no modelo CMMI-Serviços no Brasil desde 2013, apresentando 3 empresas certificadas em 2013, 8 empresas em 2014 e até julho de 2015 3 empresas foram certificadas.

| Organization<br>Organizational Unit | Team Leader<br>Sponsor | Appraisal End Date | Model (Representation): Maturity Level |
|---|---|---|---|
| A5 Arquitetura Ltda<br>Architecture and Interior | Ana Rouiller<br>Eduardo Paulino | 12/05/2014 | CMMI-SVC v1.3(Continuous): Maturity Level 2 |
| Accion Ltda<br>Implementation and support of management and process improvement using Seven Software | Ana Rouiller<br>Edney Marcos Mossambani | 05/15/2015 | CMMI-DEV v1.3(Continuous): Maturity Level 2<br>CMMI-SVC v1.3(Continuous): Maturity Level 2 |
| Consenso Soluções em Tecnologia da Informação Ltda<br>Software Development and Support | Ana Rouiller<br>Carlos Elmano Ferreira | 10/17/2014 | CMMI-DEV v1.3(Continuous): Maturity Level 2<br>CMMI-SVC v1.3(Continuous): Maturity Level 2 |
| Elotech Sistemas Ltda<br>Software Development and Support | Ana Rouiller<br>Marco Andrade | 11/28/2014 | CMMI-DEV v1.3(Continuous): Maturity Level 2<br>CMMI-SVC v1.3(Continuous): Maturity Level 2 |
| EMPREL<br>DSI - Department 02 | Ana Rouiller<br>Homero Cavalcanti | 06/11/2015 | CMMI-DEV v1.3(Continuous): Maturity Level 2<br>CMMI-SVC v1.3(Continuous): Maturity Level 2 |
| Hotsoft Informatica Ltda<br>Hotsoft Informatica Ltda | Ana Rouiller<br>Euclides Gomes | 02/28/2013 | CMMI-DEV v1.3(Continuous): Maturity Level 2<br>CMMI-SVC v1.3(Continuous): Maturity Level 2 |
| IVIA Serviços de Informática LTDA<br>IT Outsourcing - Recife | Ana Rouiller<br>Marcio Braga | 12/19/2014 | CMMI-SVC v1.3(Continuous): Maturity Level 2 |
| Lider Consultoria e Serviço Ltda<br>SLE Serviços e Tecnologia Ltda<br>Technology and BPO operations | Andre de Pinho<br>Luiz Eduardo Alvarenga | 07/30/2014 | CMMI-SVC v1.3(Staged): Maturity Level 3 |
| Neurotech Tecnologia da Informação S.A<br>Software Factory and Service Desk | Ana Rouiller<br>Domingos Monteiro | 12/13/2014 | CMMI-DEV v1.3(Continuous): Maturity Level 2<br>CMMI-SVC v1.3(Continuous): Maturity Level 2 |
| Oncase Soluções em TI<br>Business Inteligence | Ana Rouiller<br>Breno Mendonça | 10/10/2014 | CMMI-DEV v1.3(Continuous): Maturity Level 2<br>CMMI-SVC v1.3(Continuous): Maturity Level 2 |
| SG Sistemas Ltda<br>All services and projects | Ana Rouiller<br>JOAQUIM TAVARES | 12/20/2013 | CMMI-DEV v1.3(Continuous): Maturity Level 2<br>CMMI-SVC v1.3(Continuous): Maturity Level 2 |
| Sintese Compras On-line Ltda<br>Online Trading Services For Hospital Products | Ana Rouiller<br>Bertrand Gourgue | 04/10/2015 | CMMI-SVC v1.3(Continuous): Maturity Level 2 |
| SIVSA, Soluciones Informaticas, S.A.<br>Software Development Projects and HOSIX Software Maintenance Service. | Ramiro Carballo<br>Jose Antonio Román Jiménez | 12/20/2013 | CMMI-DEV v1.3(Continuous): Maturity Level 2<br>CMMI-SVC v1.3(Continuous): Maturity Level 2 |
| SUB100 Sistemas Ltda<br>All Services and Projects | Ana Rouiller<br>Walcir Franzone | 11/29/2013 | CMMI-DEV v1.3(Continuous): Maturity Level 2<br>CMMI-SVC v1.3(Continuous): Maturity Level 2 |

**Figura 11**. Organizações Brasileiras com a certificação CMMI-Serviços. Fonte: CMMI Institute

Os principais benefícios observados com os sete processos apresentados neste capítulo para a organização adotar o CMMI-Serviços são:

- Qualidade no serviço oferecido;

- Capacidade atual e futura vislumbrada com maiores detalhes;
- Maior satisfação do cliente com um serviço oferecido com qualidade e agilidade;
- Continuidade dos serviços oferecidos;
- Elaboração de relatórios eficientes para a gerência, possibilitando definir estratégias de mercado;
- Pessoas envolvidas conhecendo o seu papel e as atividades a serem trabalhadas, possibilitando motivação no trabalho;
- Gerenciamento da expectativa do cliente e das pessoas envolvidas com a prestação de serviços;
- Serviços oferecidos com mais segurança e estabilidade.

## 4.4. Considerações Finais

As Tecnologias de Informação são parte integrante de todas as organizações, principalmente daquelas que são lideres, que se destacam nos seus setores e que envolvem estas ferramentas nas suas decisões estratégicas. As organizações tentam definir objetivos e estratégias que se refletem na Gestão de Serviços de Tecnologias de Informação, numa tentativa de otimizar os recursos relacionados com a infraestrutura de TI. Diariamente são anunciadas ferramentas relacionadas com a governança de TI.

O CMMI-Serviços fornece um conjunto abrangente e integrado de orientações para a prestação de serviços de qualidade, atuando no fornecimento de serviços operacionais de TI. Basicamente disponibiliza um guia para a aplicação das boas práticas num organismo prestador de serviços, que têm como objetivo criar atividades que possam facultar serviços de qualidade aos clientes e usuários.

05

# ISO/IEC 25000

*Kamila Nayana Carvalho Serafim*

É de fundamental importância que as empresas tenham como prioridade a preocupação com o produto oferecido ao mercado, pois pessoas, instituições e vidas dependem da garantia e consistência dos dados processados e armazenados nos softwares. Mas só a preocupação não é o suficiente, a qualidade deve ser usada pela satisfação do cliente.

Uma das formas empregadas por organizações para fazer a avaliação do produto de software é o uso de normas de qualidade. A norma é usada para aferir a qualidade do produto, assim para que a avaliação seja mais efetiva, é importante que o processo de avaliação seja bem definido, estruturado e o uso de modelos de qualidade permitam estabelecer e avaliar requisitos de qualidade.

Vários modelos e processos de avaliação existem no mercado, com finalidade do próprio desenvolvedor de software e do usuário final possam ter parâmetros para que a avaliação da qualidade de software seja mais eficaz e confiável,

As normas que fazem parte do guia SQuaRE – *Software product Quality Requirements and Evaluation* descrevem um modelo

de qualidade, um conjunto de métricas que podem ser utilizadas para realizar a avaliação de um produto de software de acordo com várias perspectivas e um processo de avaliação.

Este capítulo objetiva detalhar a norma 25000 da ISO mostrando como se pode medir várias características de um software e posteriormente propõe um processo para a avaliação da qualidade de um produto de software através de um estudo de caso [Nascimento, 2010]. O estudo de caso é baseado nas normas ISO/IEC 9126 e ISO/IEC 14598, que compõem a ISO/IEC 25000, e deverá avaliar o software de acordo com um subconjunto dos atributos de qualidade.

Não menos importante, o capítulo deixa claro para o usuário o porquê da importância do uso da qualidade para o cliente e quais as vantagens conseguidas, já que qualidade ainda é uma área nova para muitas empresas ou é vista apenas como um custo maior e não é dada a devida importância à melhoria do produto.

## 5.1. Qualidade do Produto de Software

A qualidade de software é uma área de conhecimento da engenharia de software que objetiva garantir a qualidade do software através da definição e normatização de processos de desenvolvimento. Apesar dos modelos aplicados na garantia da qualidade de software atuarem principalmente no processo, o principal objetivo é garantir um produto final que satisfaça às expectativas do cliente, dentro daquilo que foi acordado inicialmente [Koscianski, 2007].

Segundo a norma ISO/IEC 9000 [ISO/IEC, 2005], a qualidade é o grau em que um conjunto de características inerentes a um produto, processo ou sistema cumpre os requisitos inicialmente estipulados para estes.

No desenvolvimento de software, a qualidade do produto está diretamente relacionada à qualidade do processo de desenvolvimento, desta forma, é comum que a busca por um software de maior qualidade passe necessariamente por uma melhoria no processo de desenvolvimento.

Um produto de software compreende os programas e procedimentos de computador e a documentação e dados associados, que foram projetados para serem liberados para o usuário. Da mesma forma como existem diversas interpretações para qualidade de um modo geral, também existem diversas interpretações para qualidade de um produto de software.

Para um produto de software ter qualidade, ele precisa ter as seguintes características:

- Fabricação usando processo de desenvolvimento e qualidade;
- Equipe focada, motivada e com expertise na tecnologia usada;
- Deve ser um software utilizável nas plataformas acordadas previamente com o cliente;
- O software deve satisfazer as necessidades do cliente;
- O cliente deve conseguir executar completamente todas as operações que foram requisitadas enquanto o projeto era desenvolvido;
- O cliente deve ter facilidade para usar o software;
- O número de defeitos deve ser o mínimo possível.

## 5.2. O Histórico da ISO/IEC 25000

A definição da arquitetura das normas SQuaRE teve início em 1999 e é orientada pelas normas ISO/IEC 9126 [ISO/IEC, 2001a] e ISO/IEC 14598 [ISO/IEC, 1999a]. Foi desenvolvido pelo grupo de trabalho WG6 do Subcomitê de Sistemas de Software (SC7)

da ISO/IEC – *International Organization for Standardization and International Electrotechnical Comission*.

Surgiu da necessidade de construir um conjunto harmônico de documentos, visto que faltava clareza na utilização das normas de qualidade de produto. Suas normas visam obter uma série logicamente organizada e unificada com abrangência de dois processos principais: especificação de requisitos e avaliação da qualidade de software, apoiados por um processo de mediação, como pode ser visto na Figura 12.

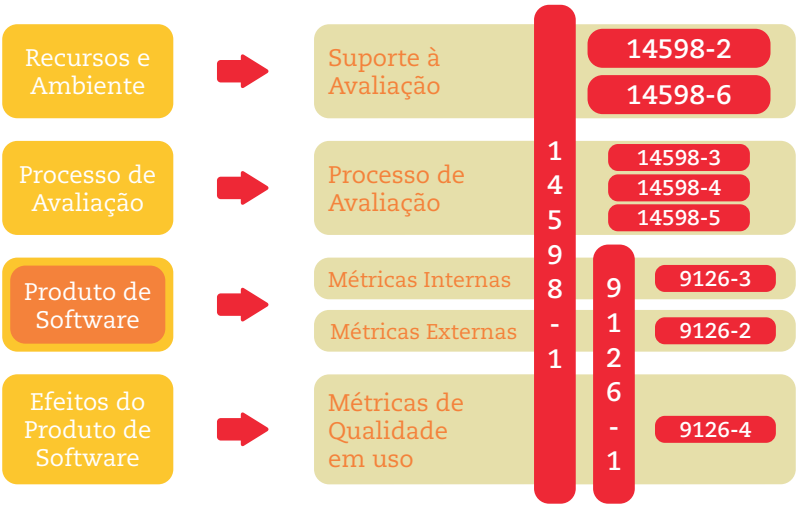



**Figura 12**. Relacionamento entre as séries 9126 e 14598 Adaptado do modelo SQuaRE

### 5.2.1. A ISO/IEC 9126

Segundo a ISO/IEC 9126-1 [ISO/IEC, 2001a], convém que a qualidade de produtos de software seja avaliada usando um modelo de qualidade definido e que este modelo seja usado durante o estabelecimento de metas de qualidade para produtos de software finais e intermediários. A série de normas

ISO/IEC 9126 descreve um modelo de qualidade para produtos de software, categorizando a qualidade hierarquicamente em um conjunto de características e subcaracterísticas. Esta série também propõe métricas que podem ser utilizadas durante a avaliação dos produtos de software (medição, pontuação e julgamento dos produtos de software).

A série de normas ISO/IEC 9126 é dividida em quatro partes:

- ISO/IEC 9126-1 - Modelo de qualidade [ISO/IEC, 2001a]: descreve um modelo de qualidade para produtos de software, contendo um conjunto de características de qualidade e suas subcaracterísticas correspondentes;
- ISO/IEC 9126-2 - Métricas externas [ISO/IEC, 2003b]: estas métricas demonstram a perspectiva externa da qualidade do produto de software quando ele está pronto para a fase de execução, bem como as funcionalidades do sistema, os testes de usabilidade e negócios. Os atributos das características de qualidade definidas na ISO/IEC 9126-1;
- ISO/IEC 9126-3 - Métricas internas [ISO/IEC, 2003c]: apresenta métricas internas para medir os atributos das características de qualidade do código, pode ser feito pelos desenvolvedores para testar sua unidade ou componente.
- ISO/IEC 9126-4 - Métricas de qualidade em uso [ISO/IEC, 2004]: apresenta métricas de qualidade em uso pelo cliente a partir dos atributos das características de qualidade. Estas métricas representam a concepção do usuário para a qualidade do produto de software.

### 5.2.2. A ISO/IEC 14598

A ISO/IEC 14598 [ISO/IEC, 1999a] diz que a avaliação de produtos de software deve ser objetiva, ou seja, baseada em observação e não em opinião. Também deve ser reprodutiva, de forma que avaliações do mesmo produto, para a mesma

especificação de avaliação, executadas por diferentes avaliadores produzam resultados aceitos como idênticos e repetíveis. Assim, um processo de avaliação deve ser definido. Este processo deve conter, cinco passos: análise dos requisitos de avaliação; especificação da avaliação; projeto e planejamento da avaliação; execução da avaliação; e documentação dos resultados. A série de normas ISO/IEC 14598 descreve um processo para avaliação de produtos de software, que consiste de quatro passos. O padrão definido distingue três perspectivas de avaliação: desenvolvedor, adquirente e avaliador.

A série de normas ISO/IEC 14598 é dividida em:

- ISO/IEC 14598-1 - Visão geral [ISO/IEC , 1999]: fornece uma visão geral do processo de avaliação da qualidade dos produtos de software e define toda a estrutura de funcionamento da série de normas ISO/IEC 14598;
- ISO/IEC 14598-2 - Planejamento e gestão [ISO/IEC, 2000a]: refere-se ao planejamento e gestão do processo de avaliação apresentando requisitos, recomendações e orientações para uma função de suporte ao processo;
- ISO/IEC 14598-3 - Processo para desenvolvedores [ISO/IEC, 2000b]: define o processo para desenvolvedores. Destina-se ao uso durante o processo de desenvolvimento e manutenção de software;
- ISO/IEC 14598-4 - Processo para adquirentes [ISO/IEC, 1999b]: define o processo para adquirentes, estabelecendo um processo sistemático para avaliação de: produtos de software tipo pacote, produtos de software sob encomenda, ou ainda modificações em produtos já existentes;
- ISO/IEC 14598-5 - Processo para avaliadores [ISO/IEC, 1998]: define o processo para avaliadores, fornecendo orientações para a implementação prática de avaliação de produtos de software (quando diversas partes necessitam entender, aceitar e confiar em resultados da avaliação);

- ISO/IEC 14598-6] - Documentação de módulos para avaliação [ISO/IEC, 2001]: fornece orientação para documentação de módulos de avaliação. Estes módulos contêm a especificação do modelo de qualidade, as informações e dados relativos à aplicação prevista do modelo e informações sobre a real aplicação do modelo.

## 5.3. Modelo SQuaRE

O núcleo principal do SQuaRE é composto de cinco divisões de normas e uma sequência prevista para extensão do modelo:

- ISO/IEC 2500n – Divisão Gestão da Qualidade,
- ISO/IEC 2501n – Divisão Modelo de Qualidade,
- ISO/IEC 2502n – Divisão Medição da Qualidade,
- ISO/IEC 2503n – Divisão Requisitos de Qualidade,
- ISO/IEC 2504n – Divisão Avaliação da Qualidade, e
- ISO/IEC 2505n – ISO/IEC 25099 – Extensão do SQuaRE

Assim, pode-se dizer que a ISO/IEC 25000 é uma revisão da ISO/IEC 9126, e incorpora as mesmas características de qualidade de software com algumas alterações:

- A segurança foi adicionada como uma característica, em vez de uma subcaracterística de funcionalidade;
- Portabilidade foi dividida em transferência e compatibilidade, incluindo a interoperabilidade;
- Foram adicionadas as subcaracterísticas robustez, utilidade, acessibilidade técnica, modularidade, reutilização e portabilidade;
- Qualidade em uso foi dividida em usabilidade em uso, flexibilidade na utilização e segurança.

Como objetivos da Normal ISO/IEC 25000 pode-se listar:

- Transformar o conjunto SQuaRE em um conjunto de normas logicamente organizadas;
- Abranger Qualidade de Software, Especificação de Requisitos e um Processo de Medição;
- Ajudar no desenvolvimento e aquisição de sistemas e produtos de software com especificação e avaliação de requisitos de qualidade;
- Incluir um modelo de qualidade para alinhar definições de atributos do cliente com qualidade e processo de desenvolvimento.

Assim como, os principais benefícios desta norma são:

- Coordenação das orientações sobre sistemas de medição e qualidade dos produtos de software e avaliação;
- Orientação para a especificação de sistemas e requisitos de qualidade do produto de software;
- Harmonização com as normas ISO/IEC 9126 e ISO/IEC 14598;
- O produtor poderá assegurar a qualidade do produto final;
- Redução nos custos com a manutenção do software;
- O usuário ficará mais satisfeito, pois estará adquirindo um produto de qualidade;
- O vendedor poderá usar como argumento de venda a qualidade assegurada do produto que está vendendo;
- Organizações poderão exigir critérios de qualificação com propósitos específicos.

### 5.3.1. ISO/IEC 2500n - Divisão Gestão da Qualidade

Contém uma visão geral do uso da série SQuaRE, dando uma visão geral do seu conteúdo, de seus modelos de referência, definições, o relacionamento entre todos os documentos da série. Fornece também orientações para planejamento e

gestão para especificação de requisitos e avaliação de produto. As normas que fazem parte da divisão Gestão da Qualidade esclarecem todos os modelos e termos referidos por todas as outras normas da série 25000.

### 5.3.2. ISO/IEC 2501n - Divisão Modelo de Qualidade

Fornece dois modelos de qualidade: um modelo que inclui características para qualidade interna, que é medida pelo desenvolvedor através dos testes unitários ou de componentes; e externa, que é medida pelos testes funcionais do produto. A qualidade interna e externa são decompostas em subcaracterísticas, como pode ser visto na Figura 13. O outro modelo é definido como qualidade em uso, que define a avaliação da qualidade para os sistemas que estão em produção, como visto na Figura 14. Também são fornecidas orientações práticas para o uso de modelos de qualidade.

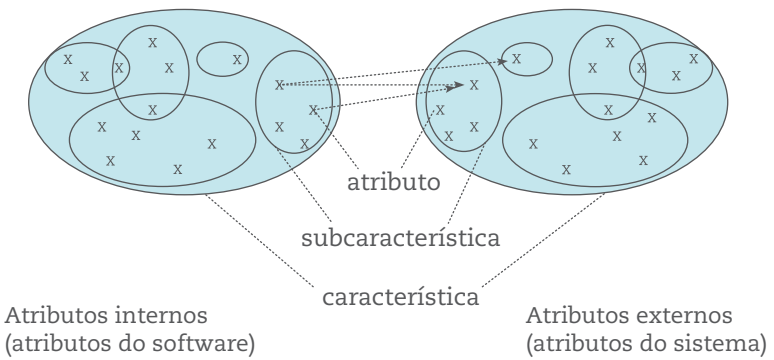



**Figura 13**. Estrutura utilizado para o modelo de qualidade, Adaptada da ISO 2501n

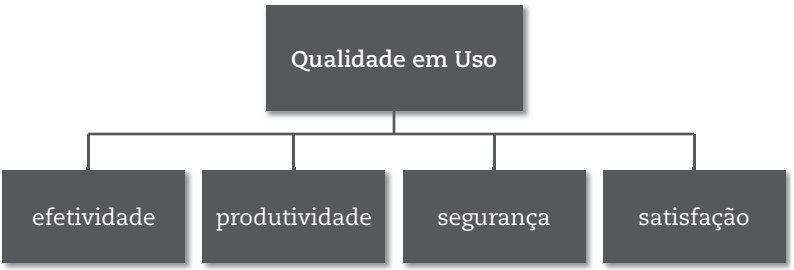



**Figura 14**. Modelo de qualidade em uso segundo a ISO/IEC 9126-1

Segundo a ISO/IEC 9126-1, as definições das características de qualidade em uso são:

- **Eficácia**: capacidade do produto de software de permitir que usuários atinjam metas específicas com acurácia e completude, em um contexto de uso específico;
- **Produtividade**: capacidade do produto de software de permitir que seus usuários empreguem quantidade apropriada de recursos em relação à eficácia obtida, em um contexto de uso especificado;
- **Segurança**: capacidade do produto de software de apresentar níveis aceitáveis de riscos de danos a pessoas, negócios, software, propriedades ou ambiente, em um contexto de uso especificado;
- **Satisfação**: capacidade do produto de software de satisfazer usuários, em um contexto de uso especificado.

As características pretendem abranger todos os aspectos de qualidade de software, de forma que se possa especificar qualquer requisito de qualidade utilizando uma das seis características, como visto na Figura 15.


**Figura 15**. Modelo de qualidade interna e externa, segundo a ISO/IEC 9126-1

Segundo a ISO/IEC 9126-1, as definições das características e subcaracterísticas de qualidade interna e externa são:

- **Funcionalidade**: capacidade do produto de software de prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições específicas. As subcaracterísticas são:
    - » Adequação: capacidade do produto de software de prover um conjunto apropriado de funções para tarefas e objetivos do usuário especificados;
    - » Acurácia: capacidade do produto de software de prover, com o grau de precisão necessário, resultados ou efeitos corretos ou conforme acordados;
    - » Interoperabilidade: capacidade do produto de software de interagir com um ou mais sistemas especificados;
    - » Segurança de Acesso: capacidade do produto de software de proteger informações e dados, de forma que pessoas ou sistemas não autorizados não possam lê-los nem modificá-los e que não seja negado o acesso às pessoas ou sistemas autorizados;

    - » Conformidade: capacidade do produto de software de estar de acordo com normas, convenções ou regulamentações previstas em leis e prescrições similares relacionadas à funcionalidade;
- **Confiabilidade**: capacidade do produto de software de manter um nível de desempenho especificado, quando usado em condições específicas. As subcaracterísticas são:
    - » Maturidade: capacidade do produto de software de evitar falhas decorrentes de defeitos no software;
    - » Tolerância a Falhas: capacidade do produto de manter um nível de desempenho especificado em casos de defeitos no software ou de violação de sua interface especificada;
    - » Recuperabilidade: capacidade do produto de software de restabelecer seu nível de desempenho especificado e recuperar os dados diretamente afetados no caso de uma falha;
    - » Conformidade: capacidade do produto de software de estar de acordo com normas, convenções ou regulamentações relacionadas à confiabilidade.
- **Usabilidade**: capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições específicas. Suas subcaracterísticas são:
    - » Inteligibilidade: capacidade do produto de software de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso específicas;
    - » Apreensibilidade: capacidade do produto de software de possibilitar ao usuário aprender sua aplicação;

    - » Operacionalidade: capacidade do produto de software de possibilitar ao usuário operá-lo e controlá-lo;
    - » Atratividade: capacidade do produto de software de ser atraente ao usuário;
    - » Conformidade: capacidade do produto de software de estar de acordo com normas, convenções, guias de estilo ou regulamentações relacionadas à usabilidade.
- **Eficiência**: capacidade do produto de software de apresentar desempenho apropriado, relativo à quantidade de recursos usados, sob condições específicas. Suas subcaracterísticas são:
    - » Comportamento em relação ao tempo: capacidade do produto de software de fornecer tempos de resposta e de processamento, além de taxas de transferência, apropriados, quando o software executa suas funções, sob condições estabelecidas;
    - » Comportamento em relação aos recursos: capacidade do produto de software usar tipos e quantidades apropriados de recursos, quando o software executa suas funções, sob condições estabelecidas;
    - » Conformidade: capacidade do produto de software de estar de acordo com normas e convenções relacionadas à eficiência.
- **Manutenibilidade**: capacidade do produto de software ser modificado. As modificações podem incluir correções, melhorias ou adaptações do software devido a mudanças no ambiente e nos seus requisitos ou especificações funcionais. Suas subcaracterísticas são:

    - » Analisabilidade: capacidade do produto de software de permitir o diagnóstico de deficiências ou causas de falhas no software, ou a identificação de partes a serem modificadas;
    - » Modificabilidade: capacidade do produto de software que uma modificação específica seja implementada;
    - » Estabilidade: capacidade do produto de software de evitar efeitos inesperados decorrentes de modificações no software;
    - » Testabilidade: capacidade do produto de software de permitir que o software, quando modificado, seja validado;
    - » Conformidade: capacidade do produto de software de estar de acordo com normas ou convenções relacionadas à manutenibilidade.
- **Portabilidade**: capacidade do produto de software de ser transferido de um ambiente para outro. Suas subcaracterísticas são:
    - » Adaptabilidade: capacidade do produto de software de ser adaptado para diferentes ambientes especificados, sem necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software considerado;
    - » Capacidade de ser instalado: capacidade do produto de software ser instalado em um ambiente especificado;
    - » Coexistência: capacidade do produto de software de coexistir com outros produtos de software independentes, em um ambiente comum, compartilhando recursos comuns;
    - » Capacidade para substituir: capacidade do produto de software de ser usado em substituição

a outro produto de software especificado, com o mesmo propósito e no mesmo ambiente;

» Conformidade: capacidade do produto de software de estar de acordo com normas ou convenções relacionadas à portabilidade.

### 5.3.3. ISO/IEC 2502n - Divisão Medição da Qualidade

Contém um modelo de referência para medição da qualidade do produto de software, algumas definições analíticas para medidas da qualidade de software e orientações práticas para aplicação.

Alguns exemplos de como se pode medir as características:

- Funcionalidade: Criar casos de testes para as funcionalidades do sistema, executá-los e verificar a quantidade de quantos testes falharam, a partir do resultado é possível medir quais pontos precisam de mais atenção. Pode estabelecer um parâmetro para saber o nível de qualidade do software;
- Confiabilidade: Considerando um sistema baseado em computador, uma simples medida de confiabilidade é o tempo médio entre falhas (MTBF), onde: MTBF = MTTF + MTTR, MTTF = tempo médio para a falha e MTTR = tempo médio para o reparo. Além da medida de confiabilidade, deve-se desenvolver uma medida de disponibilidade, a qual é a probabilidade de que um programa esteja operando de acordo com os requisitos num dado ponto no tempo, e é definida como: Disponibilidade = MTTF/(MTTF + MTTR) X 100% [Bueno e Campelo, 2010];
- Usabilidade: Fazer testes com usuários principiantes e usuários da população alvo onde sejam realizadas tarefas pré-definidas do sistema. É possível medir verificando se usuários conseguem acessar a funcionalidade

com sucesso, verificar o tempo que o usuário passou a realizar a funcionalidade de acordo com o seu perfil;

- Eficiência: Todos os testes apresentados anteriormente comprovam a eficiência de um software, além dos testes funcionas é possível fazer testes de carga, testes de estresse e teste de performance, que verificam o tempo de resposta e a quantidade de usuário que podem acessar o software ou a funcionalidade de uma só vez;
- Manutenibilidade: Testes de regressão podem ser feitos para avaliar uma mudança, esses testes podem ser feitos em cima da funcionalidade alterada e das funcionalidades impactadas usando uma matriz de rastreabilidade e testes automatizados;
- Portabilidade: Ao mudar o ambiente do software fazer testes funcionais e não funcionais no sistema, para que possa verificar a quantidade de defeitos e falhas e a partir de um parâmetro medir a qualidade.

### 5.3.4. ISO/IEC 2503n - Divisão Requisitos de Qualidade

Auxilia na especificação de requisitos de qualidade tratando de aspectos como clareza, testabilidade, não ambiguidade, para servirem de base para o processo de elicitação de requisitos com o cliente, para um produto que será desenvolvido, isto é, no início do seu ciclo de vida ou posteriormente, como entrada para um processo de avaliação.

### 5.3.5. ISO/IEC 2504n - Divisão Avaliação da Qualidade

Fornecem requisitos, recomendações e orientações para o processo de avaliação de produto de software. Apresenta também uma maneira formal de documentar uma medida, utilizando um módulo de avaliação a partir dos resultados encontrados na medição. Além disso, ela apresenta uma

estrutura para a avaliação da qualidade de produto de software. Essas estruturas são provenientes das normas ISO/IEC 9126-1 e 14598-1, 14598-3, 14598-4, 14598-5 e 14598-6.

## 5.4. Estudo de Caso

A Intertel é uma empresa de manutenção predial cujas principais atividades são a instalação e manutenção de: portões eletrônicos, centrais de interfones, circuitos fechados de TV (CFTV) e antenas coletivas [Nascimento, 2010].

O sistema Intertel foi desenvolvido principalmente para agilizar o atendimento da empresa aos seus clientes. Para isso ele mantém os dados dos clientes e funcionários da empresa, dos contratos de manutenção vigentes, e também das solicitações de serviço por parte dos clientes (ordens de serviço, ou OS), permitindo assim que os funcionários tenham acesso a essas informações mais rapidamente e possam gerenciá-las mais eficientemente [Nascimento, 2010].

### 5.4.1. Metodologia de Avaliação

Esta seção apresenta a metodologia que foi usada na avaliação do produto de software em questão. A metodologia foi concebida com base nas normas ISO/IEC 9126 e 14598 e é formada pelas seguintes atividades [Nascimento, 2010]:

- Estabelecer requisitos de avaliação;
- Selecionar o perfil dos interessados na avaliação;
- Especificar as características, subcaracterísticas e atributos de qualidade;
- Especificar métricas e pesos utilizados na avaliação;
- Definir as métricas da avaliação;

- Associar pesos às características, subcaracterísticas e atributos de qualidade;
- Realizar a avaliação;
- Realizar a avaliação do produto, levando em conta os requisitos, as métricas e os pesos definidos nas atividades 1 e 2.

### 5.4.2. Especificação das Características, Subcaracterísticas e Atributos do usuário

Com base em discussões com os interessados na avaliação, ficou definido que, entre as seis características definidas pela norma ISO/IEC 9126, a característica de maior importância para o perfil selecionado seria a Funcionalidade, portanto a avaliação foi focada nesta característica [Nascimento, 2010], cujos atributos foram os definidos na Figura 16.

1. Funcionalidade
    1.1. Adequação
        1.1.1. Quantos requisitos presentes no sistema são considerados úteis pelos usuários?
        1.1.2. Quantos requisitos presentes no sistema são considerados inúteis pelos usuários?
        1.1.3. Quantos requisitos não estão presentes no sistema, mas são importantes para o usuário?
    1.2. Acurácia
        1.2.1. Quantos requisitos presentes no sistema são considerados úteis pelos usuários e estão corretos?
        1.2.2. Quantos requisitos presentes no sistema são considerados úteis pelos usuários, mas precisam de modificação?

**Figura 16**. Atributos de qualidade para a característica Funcionalidade

### 5.4.3. Definição das Métricas de Avaliação

Esta etapa consiste em definir as métricas que serão usadas na avaliação dos atributos de qualidade especificados na seção anterior. Para cada atributo especificado será definida uma métrica que irá estabelecer o quanto o atributo avaliado está próximo do ideal.

Com base na norma ISO 9126 foram criadas as métricas para a avaliação do software, conforme mostram as Tabelas 8 e 9, [Nascimento, 2010].

**Tabela 8.** Métricas para avaliação da característica Funcionalidade e subcaracterística Adequação [Nascimento, 2010]

| 1. Funcionalidade | | | |
|---|---|---|---|
| | 1.1. Adequação | | |
| Nome da Métrica | 1.1.1. Requisitos úteis | 1.1.2. Requisitos inúteis | 1.1.3. Requisitos inexistentes |
| Propósito da Métrica | Quantos requisitos são úteis? | Quantos requisitos são inúteis? | Quantos requisitos que os usuários consideram importantes não existem no software? |
| Método de Aplicação | Contar o número de requisitos existentes no software que os usuários consideram úteis | Contar o número de requisitos existentes no software que os usuários cosideram inúteis | Contar o número de requisitos que os usuários consideram importantes e que não estão presentes no software |
| Medição | X = A/B<br>A = número de requisitos úteis<br>B = número de requisitos existentes no software | x = A/B<br>A = número de requisitos inúteis<br>B = número de requisitos existentes no software | X = A/(A+B)<br>A = número de requisitos importantes que não estão presentes no software<br>B = número de requisitos existentes no software |

| 1. Funcionalidade | | | |
|---|---|---|---|
| | 1.1. Adequação | | |
| Interpretação | 0<=X<=1 Quanto mais perto de 1 melhor | 0<=X<=1 Quanto mais perto de 0 melhor | 0<=X<=1 Quanto mais perto de 0 melhor |
| Escala | Absoluto | Absoluto | Absoluto |
| Tipo da Medida | X = quantidade/ quantidade<br>A = quantidade<br>B = quantidade | X = quantidade/ quantidade<br>A = quantidade<br>B = quantidade | X = quantidade/ quantidade<br>A = quantidade<br>B = quantidade |

**Tabela 9.** Métricas para avaliação da característica Funcionalidade e subcaracterística Acurácia [Nascimento, 2010]

| 1. Funcionalidade | | |
|---|---|---|
| | 1.2. Acurácia | |
| Nome da Métrica | 1.2.1. Requisitos corretos | 1.2.2. Requisitos incorretos |
| Propósito da Métrica | Quantos requisitos são úteis e o software os executa corretamente? | Quantos requisitos são úteis e não executados corretamente pelo software? |
| Método de Aplicação | Contar o número de requisitos existentes no software que os usuários consideram úteis e que o software os executa corretamente | Contar o número de requisitos existentes no software que os usuários cosideram úteis e que não são executados pelo softaware corretamente |
| Medição | X = A/B<br>A = número de requisitos corretos<br>B = número de requisitos úteis existentes no software | x = A/B<br>A = número de requisitos incorretos<br>B = número de requisitos úteis existentes no software |
| Interpretação | 0<=X<=1 Quanto mais perto de 1 melhor | 0<=X<=1 Quanto mais perto de 0 melhor |
| Escala | Absoluto | Absoluto |

| 1. Funcionalidade | | |
|---|---|---|
| | 1.2. Acurácia | |
| Tipo da Medida | X = quantidade/ quantidade<br>A = quantidade<br>B = quantidade | X = quantidade/ quantidade<br>A = quantidade<br>B = quantidade |

### 5.4.4. Mapeamento das Métricas em Atributos

Nesta fase as métricas foram mapeadas às subcaracterísticas e aos atributos, para que haja rastreabilidade entre os mesmos, como pode ser visto na Tabela 10.

**Tabela 10**. Mapeamento das métricas em atributos [Nascimento, 2010]

| Métrica | Subcaracterística | Atributo |
|---|---|---|
| Requisitos úteis | Adequação | Quantos requisitos presentes no sistema são considerados úteis pelos usuários? |
| Requisitos inúteis | Adequação | Quantos requisitos presentes no sistema são considerados inúteis pelos usuários? |
| Requisitos inexistentes | Adequação | Quantos requisitos não estão presentes no sistema, mas são importantes para o usuário? |
| Requisitos corretos | Acurácia | Quantos requisitos presentes no sistema são considerados úteis pelos usuários e estão corretos? |
| Requisitos incorretos | Acurácia | Quantos requisitos presentes no sistema são considerados úteis pelos usuários, mas precisam de modificação? |

### 5.4.5. Associação de pesos às características, subcaracterísticas e atributos de qualidade

Esta etapa tem como objetivo determinar o grau de importância das subcaracterísticas e atributos de qualidade no contexto do domínio da aplicação. Isto é feito através da associação de pesos (números inteiros) a cada um dos itens detalhados, como pode ser visto na Tabela 11 (contendo os valores inteiros) e na Figura 17 (contendo a associação) [Nascimento, 2010].

**Tabela 11**. Descrição dos pesos [Nascimento, 2010]

| Valor | Significado |
|---|---|
| 1 | Pouco importante |
| 2 | Importante |
| 3 | Muito importante |

1. Funcionalidade
    - 1.1. Adequação (3)
        - 1.1.1. Quantos requisitos presentes no sistema são considerados úteis pelos usuários? (2)
        - 1.1.2. Quantos requisitos presentes no sistema são considerados inúteis pelos usuários? (1)
        - 1.1.3. Quantos requisitos não estão presentes no sistema, mas são importantes para o usuário? (3)
    - 1.2. Acurácia (2)
        - 1.2.1. Quantos requisitos presentes no sistema são considerados úteis pelos usuários e estão corretos? (3)
        - 1.2.2. Quantos requisitos presentes no sistema são considerados úteis pelos usuários, mas precisam de modificação? (2)

**Figura 17**. Definição dos pesos das subcaracterísticas e atributos de qualidade. Os pesos de cada item estão entre parênteses [Nascimento, 2010]

### 5.4.6. Cálculo de subcaracterísticas

O cálculo das notas de cada item será realizado numa estratégia *bottom-up*, ou seja, primeiro serão calculadas as notas dos atributos, em seguida serão calculadas as notas das subcaracterísticas e por fim as notas das características.

Como descrito na seção anterior, as notas dos atributos serão calculadas através das métricas definidas seguindo o mapeamento mostrado na Tabela 11 e Figura 17. Após as notas dos atributos terem sido calculadas, serão calculadas as notas das subcaracterísticas. Esse cálculo será feito através da média ponderada das notas de seus atributos usando os pesos definidos na Figura 17. Dessa forma todos os resultados obtidos estarão no intervalo entre 0 e 1, como pode ser visto nas Tabelas 12 e 13.

**Tabela 12**. Melhor caso para a subcaracterística Acurácia [Nascimento, 2010]

| Subcaracterísticas e métricas | Valor da métrica | Peso | Cálculo subcaracterísitcas |
|---|---|---|---|
| **1.2. Acurácia** | | | $\frac{3 * 1 + 2 * 0}{5} = 0,6$ |
| 1.2.1. Requisitos corretos | 1 | 3 | |
| 1.2.2. Requisitos incorretos | 0 | 2 | |

**Tabela 13**. Pior caso para a característica acurácia [Nascimento, 2010]

| Subcaracterísticas e métricas | Valor da métrica | Peso | Cálculo subcaracterísitcas |
|---|---|---|---|
| **1.2. Acurácia** | | | $\frac{3 * 0 + 2 * 1}{5} = 0,4$ |
| 1.2.1. Requisitos corretos | 0 | 3 | |
| 1.2.2. Requisitos incorretos | 1 | 2 | |

### 5.4.7. Cáclculo das subcaracterísticas com normalização

Para normalizar os resultados dessas duas métricas, seu resultado deve ser subtraído de 1. Por exemplo, se seu resultado tiver sido 0,4, então o valor correspondente que será usado no cálculo das subcaracterísticas será o resultado da seguinte operação: 1 – 0,4, ou seja, 0,6. Ao realizar esta operação a interpretação do resultado de ambas as métricas ficará semelhante à interpretação das demais, assim pode-se calcular o resultado da subcaracterística através da média ponderada de seus atributos. O cálculo do melhor e pior caso da subcaracterística *1.2 Acurácia* utilizando a normalização da métrica *1.2.2 Requisitos incorretos* pode ser visto nas Tabelas 14 e 15.

**Tabela 14**. Melhor caso para a característica acurácia [Nascimento, 2010]

| Subcaracterísticas e métricas | Valor da métrica | Peso | Cálculo subcaracterísitcas |
|---|---|---|---|
| **1.2. Acurácia** | | | $\frac{3 * 1 + 2 * (1 - 0)}{5} = 1$ |

| Subcaracterísticas e métricas | Valor da métrica | Peso | Cálculo subcaracterísitcas |
|---|---|---|---|
| 1.2.1. Requisitos corretos | 1 | 3 | |
| 1.2.2. Requisitos incorretos | 0 | 2 | |

**Tabela 15**. Pior caso para a característica acurácia [Nascimento, 2010]

| Subcaracterísticas e métricas | Valor da métrica | Peso | Cálculo subcaracterísitcas |
|---|---|---|---|
| **1.2. Acurácia** | | | $\frac{3 * 0 + 2 * (1 - 1)}{5} = 0$ |
| 1.2.1. Requisitos corretos | 0 | 3 | |
| 1.2.2. Requisitos incorretos | 1 | 2 | |

### 5.4.8. Cálculo das Métricas

A coleta dos dados foi realizada através de questionários de acordo com as funcionalidades do sistema. Para cada perfil de usuário identificado no software foi elaborado um questionário considerando apenas as funcionalidades do sistema que um usuário daquele perfil tem acesso. Cada usuário deveria responder apenas o questionário de seu respectivo perfil, dessa forma a integridade dos dados obtidos é garantida, evitando que um usuário prejudique a avaliação dando uma opinião incorreta a respeito de uma funcionalidade que ele não tenha acesso. Para cada funcionalidade a qual o usuário tem acesso, ele deve assinalar uma das seguintes opções de acordo com seu julgamento sobre a funcionalidade: Correto, Incorreto ou Inútil. Através das respostas dos usuários para essa questão, as métricas *1.1.1 Requisitos* úteis, *1.1.2 Requisitos*

*inúteis*, *1.2.1 Requisitos corretos* e *1.2.2 Requisitos incorretos* podem ser calculadas [Nascimento, 2010].

A identificação dos requisitos inexistentes para o cálculo da métrica *1.1.3 Requisitos inexistentes* foi feita através da seguinte pergunta em todos os questionários: "Existe alguma funcionalidade que você gostaria que existisse no sistema, mas atualmente o sistema não possui? Em caso afirmativo cite-as com uma breve descrição". Para obter o total de requisitos inexistentes, necessário para realizar o cálculo da métrica, cada uma das respostas dessa questão foi confrontada com as demais, de forma a identificar requisitos semelhantes entre elas. Por fim, foi elaborada uma lista com todos os requisitos funcionais distintos sugeridos pelos usuários. O total de requisitos desta última lista é a quantidade de requisitos inexistentes que será utilizado para o cálculo da métrica, como pode ser visto na Tabela 16 [Nascimento, 2010].

**Tabela 16**. Cálculo das métricas [Nascimento, 2010]

| Métricas | A | B | X |
|---|---|---|---|
| 1.1.1 Requisitos úteis | 45 | 45 | $\frac{45}{45} = 1$ |
| 1.1.2 Requisitos inúteis | 0 | 45 | $\frac{0}{45} = 0$ |
| 1.1.3 Requisitos inexistentes | 28 | 45 | $\frac{28}{(28 + 45)} = 0$ |
| 1.2.1 Requisitos corretos | 36 | 45 | $\frac{36}{45} = 0{,}8$ |
| 1.2.2 Requisitos incorretos | 9 | 45 | $\frac{9}{45} = 0{,}2$ |

### 5.4.9. Cálculo das Características

Após o cálculo das métricas, foram calculados os resultados das subcaracterísticas e características a partir do resultado da seção anterior, como pode ser visto na Tabela 17.

**Tabela 17**. Resultado de cada características e subcaracterística [Nascimento, 2010]

| Características, subcaracterísticas e métricas | Valor da métrica | Peso | Cálculo subcaracterísticas | Cálculo características | Total |
|---|---|---|---|---|---|
| **1. Funcionalidade** | | - | | $\frac{3 * 0,81 + 2 * 0,8}{5}$ = 0,81 | 0,81 |
| **1.1 Adequação** | | 3 | $\frac{2 * 1 + 1 * (1 - 0) + 3 * (1 - 0,38)}{6}$ = 0,81 | | |
| 1.1.1 Requisitos úteis | 1 | 2 | | | |
| 1.1.2 Requisitos inúteis | 0 | 1 | | | |
| 1.1.3 Requisitos inexistentes | 0,38 | 3 | | | |
| **1.2 Acurácia** | | 2 | $\frac{3 * 0,8 + 2 * (1 - 0,2)}{5}$ = 0,8 | | |
| 1.2.1 Requisitos corretos | 0,8 | 3 | | | |
| 1.1.1 Requisitos incorretos | 0,2 | 2 | | | |

### 5.4.10. Análise dos Resultados

Analisando os resultados apresentados nas duas últimas tabelas, pode-se verificar que, com relação às métricas Requisitos úteis e Requisitos inúteis, o software é realmente adequado aos usuários, pois todos os requisitos presentes

no mesmo são úteis e nenhum deles é inútil. Mas a métrica Requisitos inexistentes for analisada, pode-se verificar que o software não está atendendo a todas as necessidades do usuário, visto que o valor obtido para esta métrica foi relativamente alto. Isto leva à conclusão de que a elicitação de requisitos do sistema não foi bem elaborada. Por conta do resultado dessa última métrica, o resultado da subcaracterística Adequação foi bastante prejudicado [Nascimento, 2010].

Continuando a análise com as métricas Requisitos corretos e Requisitos incorretos, pode-se ver que seus resultados podem ser considerados razoavelmente bons, o que indica que poucos requisitos do produto de software precisam de correções. Assim, pode-se concluir que a subcaracterística Acurácia está sendo parcialmente atendida [Nascimento, 2010].

## 5.5. Considerações Finais

A criação do SQuaRE possibilitou uma melhor organização das normas, pois uniu as duas antigas séries ISO/IEC 9126 e ISO/IEC 1498, que eram bastante inter-relacionadas, mas tinham conflitos, facilitando os usuários a manterem o foco em apenas uma norma para garantir a qualidade do produto. As divisões sobre medição e avaliação do produto servem de alicerce para saber o nível de qualidade do sistema e fazer o controle do mesmo, a fim de que se tenha sempre um resultado satisfatório para o cliente.

A introdução de orientações para uso prático em forma de exemplos facilitou o entendimento para a utilização da norma, porque guia o usuário com mais efetividade e dá a ele opções e ideias de como medir, como avaliar e como controlar.

# GESTÃO DE SOFTWARE

*Sandro Ronaldo Bezerra Oliveira*
*Alexandre Marcos Lins de Vasconcelos*

O software é o elemento mais importante e custoso de muitos sistemas baseados em computador. Apesar dos inúmeros avanços, ainda continuamos sob o efeito da chamada "Crise do Software"[2] que se manifesta através das seguintes constatações [Yoshidome, 2014]:

- Grandes sistemas de software não funcionam adequadamente;
- Os projetos estão frequentemente atrasados, isto é, não são entregues nos prazos estipulados;
- Os custos são frequentemente maiores que os previstos.

Este fenômeno é consequência direta da falta de uma política coerente para o gerenciamento do desenvolvimento de

2 A **crise do software** foi um termo utilizado nos anos 1970, quando a engenharia de software era praticamente inexistente. O termo expressava as dificuldades do desenvolvimento de software frente ao rápido crescimento da demanda por software, da complexidade dos problemas a serem resolvidos e da inexistência de técnicas estabelecidas para o desenvolvimento de sistemas que funcionassem adequadamente ou pudessem ser validados.

software. Segundo Fernandes (1995), a gestão dos projetos de software é uma tarefa bastante complexa que consiste na aplicação de procedimentos, práticas e tecnologias que apoiam:

- O planejamento das atividades que compõem o processo de desenvolvimento de software, de acordo com restrições de prazos e de orçamento;
- A definição da estrutura organizacional utilizada para desenvolver e manter os produtos;
- Alocação de recursos, de uma forma que otimize o processo;
- A direção, controle e monitoramento contínua do processo de desenvolvimento, a fim de acompanhar a evolução do desenvolvimento de acordo com as expectativas.

De fato, a preocupação com a gerência deve acompanhar todo o processo de desenvolvimento de software, monitorando a realização correta das atividades e, deste modo, garantindo a melhor distribuição dos recursos por todo o ciclo de vida do software.

A gerência de projetos, portanto, envolve as funções necessárias para conduzir um grupo de atividades (incluindo os profissionais e recursos) visando atingir os objetivos do projeto [Fernandes, 1995]. Neste contexto, o gerente do projeto de software exerce um papel fundamental para a correta definição e realização dos projetos. Algumas atividades desempenhadas pelo gerente são:

- Avaliação dos Requisitos de um Projeto: através de uma cuidadosa revisão destes requisitos, o gerente deve estabelecer cuidadosamente os objetivos que devem ser atingidos pelo projeto;
- Definição de como o software deve ser construído: o gerente deve equilibrar os diversos custos do projeto (por exemplo, a utilização dos recursos) para atender os

requisitos (restrições de tempo e orçamentárias previstas para o projeto).

A gerência de desenvolvimento de software tem como objetivos as estimativas mais precisas (prazos, custos, entre outros), custos mais baixos (quanto à alocação correta de recursos e produtividade), melhor controle do processo (síndrome do 90% pronto), qualidade do produto, possível certificação da qualidade do produto, manutenção facilitada e até mesmo a reutilização [Reis, 1999].

Desta forma, esta parte do livro fornecerá:

- conceitos sobre a gestão ágil de projetos de software (Capítulo 6);
- as especificações sobre as práticas de gestão de pessoas (Capítulo 7) e das técnicas de gestão de riscos (Capítulo 8) em projetos de software;
- os conhecimentos sobre o conjunto de boas práticas de gestão para serem aplicadas na infraestrutura, operação e manutenção de serviços de tecnologia da informação a partir do ITIL (Capítulo 9).

06

# GESTÃO ÁGIL DE PROJETOS DE SOFTWARE

*Leonardo da Silva Leandro*

Nos últimos anos tem se falado sobre métodos ágeis de gerenciamento de projeto. Essa popularidade surgiu das muitas empresas de tecnologia da informação que começaram a ganhar espaço no mercado com a adoção dos métodos ágeis de gerenciamento e desenvolvimento de software.

A ascensão das metodologias ágeis deixou claro que os projetos não precisam ser gerenciados e desenvolvidos sempre sob uma forma tradicional e engessada de gerenciamento. Projetos diferentes requerem formas diferentes de gerir, de acordo com uma série de características que as metodologias ágil e tradicional possuem. Afinal, o que qualquer empresa deseja e precisa são processos que possibilitem a mesma atingir seus objetivos de negócio, com a satisfação de todas as partes interessadas: clientes, colaboradores, acionistas, etc.

Este capítulo tem por objetivo descrever as características da gestão ágil de projetos desde o manifesto ágil. Incialmente é realizada uma breve descrição sobre gestão, dando um enfoque em gestão de projetos. Em seguida é descrito em

detalhes o gerenciamento tradicional de projetos, suas características, em que tipo de frente eles são preferencialmente aplicados e alguns exemplos de uso.

Posteriormente, inicia-se o estudo sobre a gestão ágil de projetos, apresentando o que motivou a entrada dos métodos ágeis de uma vez por todas no mercado, ou seja, o Manifesto Ágil. E então é detalhado o gerenciamento ágil, usando algumas das metodologias ágeis usadas no mercado como forma de apresentar as características desta forma de gerenciar.

## 6.1. A Gestão

A gestão é um ramo das ciências humanas que trata do relacionamento com grupos de pessoas. Procura manter a sinergia entre elas, a estrutura da empresa e os recursos existentes. Gestão também quer dizer gerenciamento e/ou administração. Tem como objetivo o crescimento de uma entidade como um todo [Significados, 2015].

O conceito de gestão surgiu após a revolução industrial. Foi naquela época que os profissionais decidiram buscar soluções para problemas que não existiam antes, usando vários métodos de ciências para administrar os negócios da época, o que deu inicio à ciência da administração, pois é necessário o conhecimento e aplicação de modelos e técnicas administrativas [Significados, 2015].

### 6.1.1. Gestão de Projetos

O gerenciamento de projetos é a aplicação de conhecimento, habilidades, ferramentas e técnicas às atividades do projeto a fim de atender aos seus requisitos. Gerenciar um projeto inclui também identificação de requisitos, adaptação às diferentes necessidades (visão do cliente e visão da equipe)

e balanceamento das restrições conflitantes do projeto, tais como: Escopo, Qualidade, Cronograma, Orçamento, Recursos e Risco [PMI, 2009].

No tocante a projetos de software, há a necessidade do gerenciamento de projetos porque a engenharia de software está sempre sujeita a restrições de orçamento e de prazo. Tais restrições são potenciais agravantes na conclusão e entrega de determinado produto ou serviço, pois existe uma série de variáveis que devem ser consideradas durante o ciclo de vida de um projeto.

O fracasso de muitos projetos de software, na década de 60 e no início da década de 70, foi a primeira indicação das dificuldades de gerenciamento de software. A falha residia na abordagem de gerenciamento utilizada. Técnicas de gerência provenientes de outras disciplinas da engenharia eram aplicadas e mostravam-se ineficazes para o desenvolvimento de software [Sommerville, 2010].

Um Projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo. Por ser temporário, um projeto tem início e término definidos [PMI, 2009]. Pode-se listar como exemplos de projetos: o desenvolvimento de um novo produto ou serviço; efetuar uma mudança de estrutura, de pessoal ou de estilo de uma organização; desenvolvimento ou aquisição de um sistema de informação novo ou modificado; implementação de um novo procedimento ou processo de negócios.

Uma vez que não existe uma forma universal para resolver todos os problemas, projetos podem ser geridos de formas diferentes. Contudo, as fases de um projeto, de uma forma geral, são basicamente as mesmas: Início, Planejamento, Execução, Monitoramento e Controle e Finalização. A Figura 18 apresenta as cinco fases de um projeto.

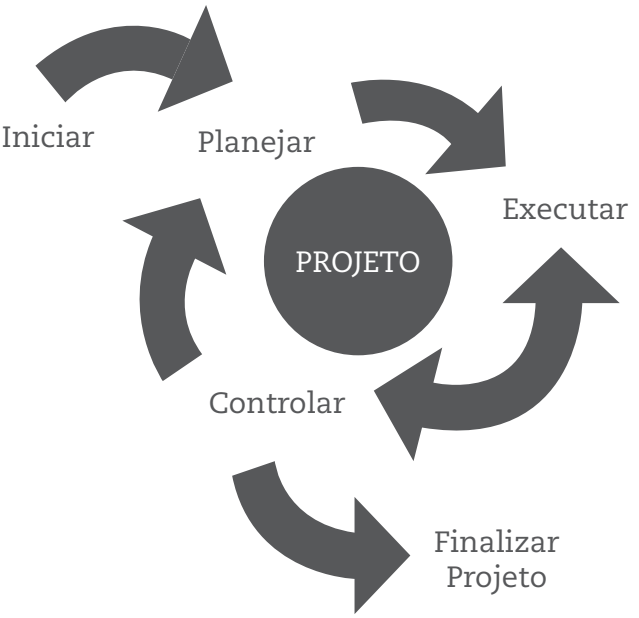



**Figura 18**. Ciclo de vida de um projeto

Com relação à estrutura do ciclo de vida de um projeto gerenciado de forma tradicional, pode-se ver que frequentemente é referenciada na comunicação com a alta administração ou outras entidades menos familiarizadas com os detalhes do projeto. De forma genérica, a estrutura do ciclo de vida geralmente apresenta as seguintes características (ver a Figura 19) [PMI, 2009]:

- Os níveis de custo e de pessoal são baixos no início, atingem um valor máximo enquanto o projeto é executado e caem rapidamente conforme o projeto é finalizado (observar a linha pontilhada na Figura 19 que ilustra esse padrão típico);
- A influência das partes interessadas, os riscos e as incertezas (conforme ilustrado na Figura 20) são maiores

durante o início do projeto. Estes fatores caem ao longo da vida do mesmo;

- A capacidade de influenciar as características finais do produto do projeto, sem impacto significativo sobre os custos, é mais alta no início e torna-se cada vez menor conforme o projeto progride para o seu término. A Figura 20 ilustra a ideia de que os custos das mudanças e correções de erros geralmente aumentam significativamente conforme o projeto aproxima-se do término.

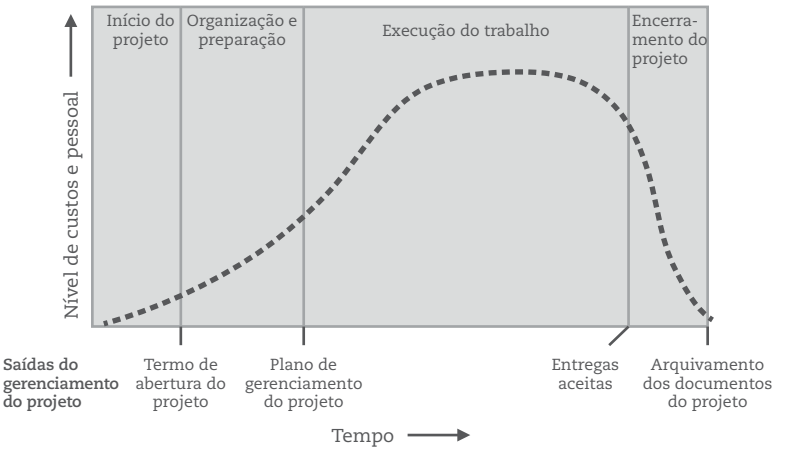



**Figura 19**. Nível típico de custos e pessoal ao longo do seu ciclo de vida

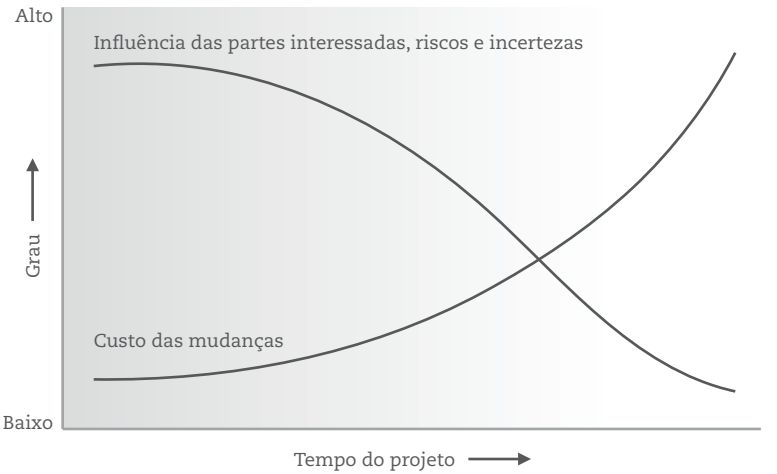



**Figura 20**. Impacto da variável com base no tempo decorrido do projeto

Existem duas grandes frentes de gerenciamento de projetos, as quais possuem características particulares que as tornam propícias para certos tipos de projeto: Gerenciamento Tradicional e Gerenciamento Ágil. Ambas são capazes de transformar completamente o ambiente de trabalho em virtude da metodologia que movimenta as atividades durante o ciclo de vida do projeto. Serão apresentadas nas próximas seções uma descrição em detalhes sobre cada uma dessas duas formas de gerenciar.

## 6.2. Gestão de Projetos Tradicional

Na gestão tradicional o ciclo de vida de um projeto costuma ser rigorosamente respeitado. Esta forma de gerir aborda basicamente todas as técnicas de gerência descritas pelo PMI - *Project Management Institute*. Sendo assim, fazem parte dos entregáveis de um projeto gerido de forma tradicional ao

conjunto de documentos apresentados pelo PMBOK – *Project Management Body of Knowledge* [PMI, 2009]. Muitas vezes os clientes nem se quer fazem uso de 70% dos documentos que são entregues, mas como é padrão criá-los, então os mesmos não deixam de ser produzidos.

O produto final ou serviço é visto como uma unidade indivisível de entrega e, sendo assim, apenas é considerado para entrega quando está 100% concluído.

O relacionamento entre a equipe e os *stakeholders* é praticamente inexistente durante a etapa de desenvolvimento do produto. Por esta razão, na reunião de planejamento e elicitação de requisitos é solicitado que os requisitos sejam descritos da forma mais detalhada possível e sem ambiguidades. Com uma descrição completa e não ambígua, a equipe de desenvolvimento procura evitar que alguma funcionalidade deixe de estar presente no produto final, ou evitar que determinada funcionalidade esteja errada.

No que diz respeito ao escopo e ao custo do projeto, os mesmos são fechados na etapa de planejamento e são mantidos fixos até a etapa final. Indiferente se o cronograma não for respeitado e o produto ou serviço requisitado demore a ser entregue, o escopo e o custo são imutáveis.

A Figura 21 mostra a pirâmide Requisitos/Escopo x Recursos x Cronograma, onde se pode ter conhecimento do que é fixo e do que é variável na visão da gerência tradicional de projetos. Note que no centro da pirâmide está escrito "*Plan-Driven*", ou seja, na visão da gestão tradicional o projeto é motivado pelo planejamento. Em outras palavras, a fase de planejamento é o momento de maior valor, pois se trata de um momento único durante todo o ciclo tradicional onde muita informação é apresentada de uma vez e tudo é extremamente importante para a evolução do projeto.

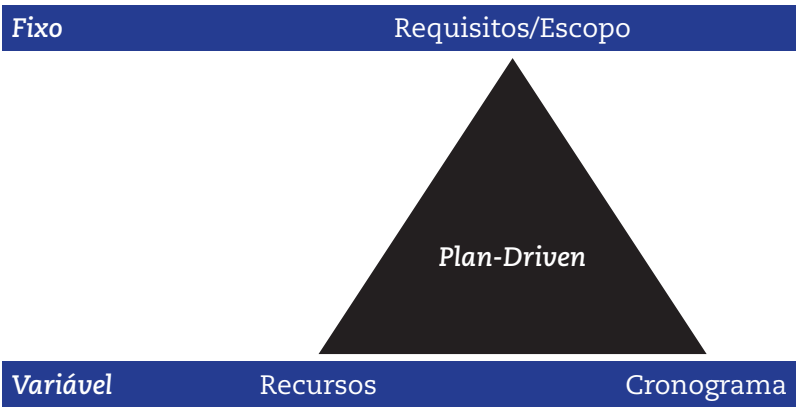



**Figura 21**. Pirâmide dos valores fixos e variáveis na gestão tradicional de projetos.

### 6.2.1. Aplicações da Gestão Tradicional no Mercado

Para dar uma melhor ideia de que tipo de projeto a gestão tradicional é mais bem aplicada, serão descritas duas situações reais da aplicação da gestão tradicional no mercado.

#### 6.2.1.1. Contratação de Empresa de Desenvolvimento de Software por Meio de Edital de Licitação

No primeiro cenário, a prefeitura de um determinado município do estado brasileiro contrata, por meio de licitação, uma empresa B de desenvolvimento de software para desenvolver um Sistema de Gestão de Assistência Social e controle de demandas da população. Tudo que é necessário para a implementação do projeto está descrito em um edital que foi apresentado no processo de licitação.

Durante a reunião de planejamento do projeto a equipe detalhou o edital, procurando compreender todos os requisitos descritos no documento, os quais foram escritos de forma clara, detalhada e não ambígua. Além dos requisitos,

o documento também apresentava o custo do projeto e um possível cronograma.

A partir de então, cria-se o plano de projeto com o cronograma, os recursos alocados, o custo, os requisitos e as etapas do projeto. Durante todas as fases seguintes (execução, monitoramento e controle e conclusão) o cliente fica quase que completamente isento de qualquer envolvimento com o projeto.

Ao final do desenvolvimento, o produto é criado e entregue em sua completude ao cliente, da forma como estava descrito no edital, ou seja, respeitando o escopo e o custo fixados e procurando ao máximo respeitar também o cronograma.

#### 6.2.1.2 Desenvolvimento de uma Primeira Versão de um Software

No segundo cenário, uma empresa de desenvolvimento de software é contratada para desenvolver um produto X. Durante a reunião de planejamento do projeto, o gerente, o arquiteto de software e o cliente discutem a respeito do escopo, do cronograma, do custo, dos recursos alocados e dos requisitos. O cliente descreve com clareza e detalhes tudo que ele espera de funcionalidade no software requisitado.

Após a reunião de planejamento, o time de desenvolvimento é acionado para iniciar a implementação do software. O cliente não mais tem participação nesta fase. Os requisitos que foram detalhados durante a fase de planejamento, são usados como insumo pela equipe de desenvolvimento.

No final, a primeira versão do software é entregue para o cliente, em sua totalidade, com tudo que era esperado pelos *stakeholders*, com todas as funcionalidades descritas no documento de requisitos. Com a primeira versão entregue, o cliente poderá certificar-se de que o software funciona como esperado e poderá inclusive perceber novas funcionalidades que poderiam ser aderidas, o que levaria a um novo ciclo de projeto.

#### 6.2.1.3 Conclusão sobre os Cenários Apresentados

Em virtude dos casos reais citados, conclui-se que o gerenciamento tradicional possui características que se adequam a certos tipos de projetos, os quais não demandam entregas parciais, contato frequente com o cliente, entre outras coisas.

Projetos em editais de licitação ou uma primeira versão completa de um software são projetos com escopo e custo fixos, e recursos e cronograma variáveis. Para tanto, os requisitos são descritos em sua totalidade na reunião inicial de planejamento.

Apesar do nível de detalhamento, da clareza e da não ambiguidade na descrição dos requisitos, esta forma de gerenciamento de requisitos é muito propensa a falhas, uma vez que o cliente não mantém contato constante com a equipe de desenvolvimento para alinhar se o que está sendo feito de fato é o que se espera para o produto ou serviço final.

## 6.3. Gestão Ágil de Projetos

O conceito de agilidade fala da capacidade de alterar a posição de um corpo de forma eficiente, ou seja, requerendo a integração de competências de movimentos isolados utilizando uma combinação de equilíbrio, coordenação, velocidade, reflexos, resistência e força. Tem como princípio geral o aumento da socialização, do companheirismo, da colaboração mútua, e ainda diminui o medo e a inibição.

Trazendo este conceito para o lado da gestão de projetos quase nada muda nos conceitos, apenas é preciso realizar adaptações nas palavras. Agilidade em gestão de projetos também trata da capacidade de alterar algo, mas nesse caso seria a alteração do estado de uma entrega, de uma funcionalidade ou mesmo do projeto como um todo. Isso é feito por

meio da integração das competências individuais de um time de desenvolvimento. É preciso equilíbrio entre os participantes de forma a manter a velocidade de produção.

O fato da agilidade promover a colaboração mútua entre os membros de uma equipe em virtude de se alcançar um resultado positivo, faz com que os participantes fiquem mais unidos, conheçam-se melhor e saibam como se ajudar.

O gerenciamento ágil de projetos veio como uma forma de rebater a gestão tradicional, tida muitas vezes como "engessada" em virtude de seguir um padrão muito rigoroso, com criação obrigatória de documentos que nem sempre são aproveitados, atividades orientadas a processos deixando as equipes muito individualizadas e com um afastamento muito grande entre a equipe e o cliente.

Para resolver certos desvios existentes na gestão tradicional, os teóricos do gerenciamento ágil investiram em melhorias nos principais pontos que penalizam a evolução de um projeto:

- Plano de Projeto, permitindo que seja realizado sucessivas vezes, com um menor grau de detalhe por vez, priorizando as entregas mais importantes;
- Escopo, deixando o cliente livre para fazer uma descrição abrangente do projeto de forma macro e desafiadora, ambígua e metafórica. Direcionam o foco para o que agrega mais valor ao cliente;
- Controle de tempo do projeto, realizado através da identificação de mudanças por meio do ambiente físico (controles visuais, reuniões diárias);
- Medição do progresso, a qual pode ser orientada para resultados tangíveis, tais como protótipos, demonstrações, desenhos e artefatos visuais. Mas que também

pode ser orientada pela frequência de *feedback* da equipe para com o gerente de projetos.

A Figura 22 mostra a pirâmide dos três valores (Recursos x Cronograma x Requisitos/Escopo) na visão da gestão ágil. Nota-se que no centro da pirâmide, diferentemente da gestão tradicional que mantém o projeto orientado ao planejamento, a gestão ágil procura orientar o projeto para o que agrega mais valor ao cliente.

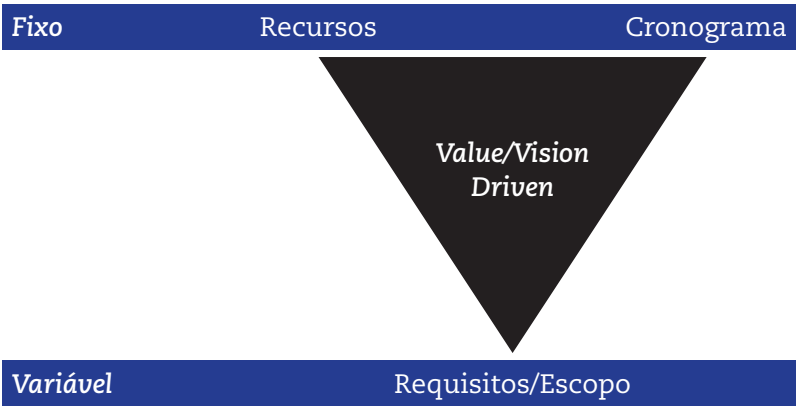



**Figura 22**. Pirâmide dos valores fixos e variáveis na gestão ágil de projetos.

Muitos dos métodos ágeis já existiam no mercado, mas de forma muito pontual e não havia divulgação. O que tornou estas práticas difundidas entre as empresas, bem como o que motivou a criação de novas metodologias ágeis de gerenciamento foi o evento que marcou a implantação da gestão ágil de projetos no mercado: O Manifesto Ágil.

Nesta próxima subseção será detalhada a gestão ágil, iniciando com o Manifesto Ágil (contexto histórico, valores e princípios), logo após serão detalhadas três metodologias ágeis de gerenciamento de projetos de forma que seja possível ver como os valores e princípios defendidos pelos

signatários (articuladores do manifesto ágil) são colocados em prática nas atividades e no dia-a-dia das equipes ágeis.

### 6.3.1. O Manifesto Ágil

Com um índice muito alto de fracassos nos projetos de software, começou-se a analisar o que estaria causando tamanha crise na engenharia de software. Muitos pontos foram levantados, como por exemplo, o relacionamento entre o cliente e a equipe de desenvolvimento, o relacionamento entre os membros da equipe, a valorização do planejamento mais do que o produto propriamente dito, etc.

O que se pode concluir é que os problemas interconectam-se e causam uma "bola de neve" que vai crescendo a cada fase do projeto. Sistemas são entregues com atrasos e/ou orçamentos estourados. Além da entrega atrasada, o produto apresenta falta de funcionalidades (requisitos não atendidos). Consequentemente, a insatisfação do cliente é latente, bem como a sua falta de fé na equipe que está desenvolvendo, que acaba trabalhando muitas horas por semana para dar conta dos cronogramas atrasados, sob uma disciplina muito estática de trabalho.

O manifesto ágil atacou cada item que se mostrava ameaçador nas diferentes etapas do desenvolvimento de um produto ou serviço. O "evento" propriamente dito ocorreu em Fevereiro de 2001, no estado de Utah, nos Estados Unidos [Davies e Sedley, 2009]. Contou com a colaboração/participação de 17 homens, teóricos e práticos da engenharia de software, como Martin Fowler.

O manifesto apresentou diferentes praticas de gerenciamento e desenvolvimento de software que se mostravam bastante flexíveis e aplicáveis a um conjunto de projetos.

Como forma de marcar o evento com uma frase de efeito, eles deixaram o seguinte comentário:

*"Estamos evidenciando maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a valorizar:*

- *Interação entre pessoas MAIS QUE processos e ferramentas;*
- *Software em funcionamento MAIS QUE documentação abrangente;*
- *Responder a mudanças MAIS QUE seguir um plano;*
- *Colaboração com o cliente MAIS QUE negociação de contratos.*
- *Ou seja, mesmo tendo valor os itens à direita, valorizamos mais os itens à esquerda."* [Rasmusson, 2015].

Os criadores do Manifesto Ágil apresentaram também os princípios que regem a prática:

- Satisfazer o cliente, entregando o software em tempo hábil e continuamente;
- Aceitar as mudanças de requisitos, em qualquer fase do projeto (processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente);
- Entregar software na menor escala de tempo possível;
- Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto;
- Construir projetos com indivíduos motivados e comprometidos com o resultado;
- Usar a comunicação efetiva;
- Ter o software em funcionamento é a principal medida do progresso;
- Atenção contínua à excelência técnica;
- As melhores arquiteturas, requisitos e projetos provêm de equipes organizadas;

- Refletir sobre como se tornar mais eficaz, ajustando e adaptando o comportamento da equipe;
- Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente;
- Simplicidade é essencial, a arte de maximizar a quantidade de trabalho não realizado.

### 6.3.2. Métodos Ágeis

O Manifesto Ágil foi o embasamento filosófico de todos os métodos ágeis e diversos métodos de desenvolvimento de software estão alinhados a ele. A maioria deles utiliza-se de ciclos curtos, que são chamados de iterações e normalmente têm duração de poucas semanas, dessa forma garantindo *feedback* frequente e respostas rápidas às mudanças [Gomes, 2014].

Métodos ágeis assumem imprevisibilidade natural do desenvolvimento de software, por isso, considera-se que o cliente também está aprendendo sobre o que precisa e, que a cada *feedback* pode mudar de ideia e alterar o escopo do projeto.

Scrum, *eXtreme Programming* (XP) e Kanban são exemplos de métodos ágeis. Cada um deles possui uma abordagem particular, fazendo uso dos valores e princípios defendidos pelo manifesto ágil.

#### 6.3.2.1. Scrum

Entre os inúmeros métodos ágeis existentes, o Scrum é considerado o mais focado em aspectos gerenciais do desenvolvimento de software muito pela sua organização durante as iterações. É um processo iterativo e incremental para

desenvolvimento de qualquer produto ou gerenciamento de qualquer trabalho.

O Scrum procura empregar os eventos com duração fixa (*time-boxes*) de forma a criar regularidade. Entre os elementos do Scrum que possuem duração fixa, pode-se citar o Planejamento da Versão para Entrega, a *Sprint*, a Reunião Diária (*stand up meeting*), a Revisão da *Sprint* e a Retrospectiva do *Sprint* [Schwaber, 2009]. O motor do Scrum são as *Sprints*, as quais possuem duração entre 15 dias e um mês. Como resultado, a Sprint procura entregar um incremento do produto final, contudo já com funcionalidades usáveis.

**a) Artefatos**

No tocante aos artefatos, o Scrum faz uso de quatro principais: *Product Backlog, Sprint Backlog, Burndown Deliverable, Burndown Sprint*.

O *Product Backlog* nada mais é do que a lista com tudo que pode ser necessário para a implementação do produto ou serviço a ser entregue. Esta lista é priorizada, sempre procurando prover o que agrega mais valor ao cliente. O *Sprint Backlog* é a lista de atividades que foram designadas para determinada *Sprint*. Estas atividades são originadas do *Product Backlog* e são selecionadas de forma que ao final da *Sprint* um incremento do produto potencialmente entregável possa ser produzido. O gráfico *Burndown Deliverable* fornece à equipe uma medição do *Product Backlog* restante ao longo do tempo de um plano de entrega. O gráfico *Burndown Sprint* provê à equipe uma visão do que já foi realizado do *Sprint Backlog* diante do que ainda falta realizar.

A Figura 23 mostra um gráfico do *Burndown Sprint*. Neste caso, a linha contínua, que representa o estado real da *Sprint*, inicia bem acima do desejável nos primeiros dias de atividade,

o que simboliza que ocorreram problemas de atraso no término das atividades. Se a linha contínua voltar a ficar no mesmo patamar da linha pontilhada, então isso simboliza que a equipe conseguiu progredir nas atividades de forma a contornar os atrasos iniciais.

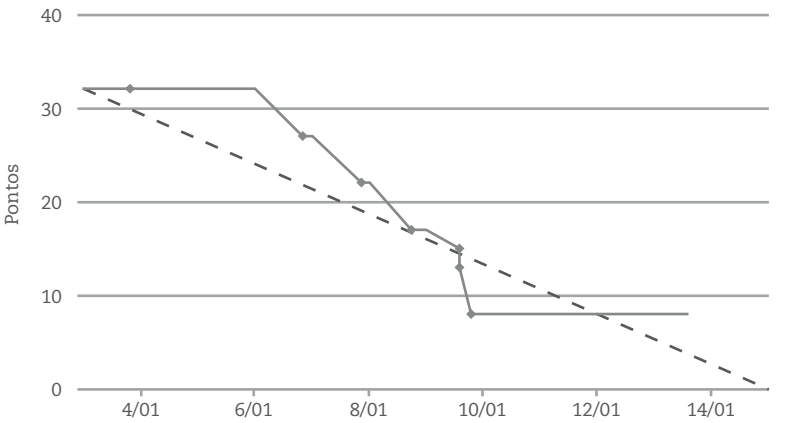



**Figura 23**. Gráfico do *Burndown Sprint*

### b) Papéis no Scrum

No Scrum existem três papéis: *Product Owner*, *Scrum Master* e a Equipe [Gomes, 2014]. O *Product Owner* (também conhecido como dono do produto) é a pessoa que tem um conhecimento amplo sobre o negócio. Em outras palavras, ele representa o cliente dentro da equipe (algumas vezes o próprio cliente pode ser o *Product Owner*). É ele quem define o que deve ser priorizado nas *Sprints*, tendo em vista os interesses do cliente e também é o responsável por criar o *Product Backlog*.

O *Scrum Master* é o responsável por coordenar a *Sprint*. Ele não é o gerente, tampouco precisa ser exclusivamente um

líder de projetos. Quem está alocado a este papel deve ministrar as reuniões diárias, discutir sobre o andamento da *Sprint* e procurar mitigar problemas junto à equipe. Em muitos lugares tornou-se prática que se faça um rodízio com relação a este papel, ou seja, que em cada nova *Sprint* o *Scrum Master* seja um membro diferente da equipe com relação à *Sprint* anterior. Esse tipo de abordagem permite que todos possam vivenciar essa experiência, fazendo com que a equipe como um todo contribua para o melhoramento das iterações.

Finalmente, mas não menos importante, tem a Equipe, composta pelos desenvolvedores, engenheiros de testes, *designers*, etc. A Equipe dita o ritmo da *Sprint*, uma vez que é responsável pelas atividades alocadas em cada nova iteração. Cada membro da Equipe decide quais atividades da *Sprint* deseja ficar responsável por fazer, pois cada um sabe exatamente o que tem condições de fazer. Também é a Equipe quem estima o tempo de conclusão de cada atividade listada no *backlog*, para assim poder-se estimar o tempo total de cada *Sprint*.

**c) Sprint**

A Sprint representa cada iteração do Scrum, tendo como resultado uma potencial funcionalidade ou serviço que já poderá ser aproveitado pelo cliente. Toda *Sprint* inicia com a reunião de planejamento, a qual pode ser dividida em duas etapas: a etapa estratégica, na qual se decide o que será feito, uma meta; e a etapa tática, onde é discutido como serão feitas as atividades.

Durante o período de execução da *Sprint*, diariamente são realizadas as *stand up meeting*, que, por padrão, duram entre 15 e 30 minutos. Estas reuniões são de extrema importância para a integração da Equipe, fazendo com que todos possam compartilhar sobre o andamento das atividades, problemas

existentes e também podem ser apresentadas sugestões. Normalmente elas ocorrem no início do dia e todos ficam em pé [Britto, 2015].

Ao final da iteração são realizadas duas reuniões. A primeira é a Revisão da *Sprint*, onde são realizadas as demonstrações do que foi produzido, apresentado resultados, etc. A segunda reunião realizada no final da *Sprint*, trata da Retrospectiva da *Sprint*. Esta reunião é muito interessante e de grande importância para a evolução da Equipe, tanto individualmente como coletivamente. Isso porque neste momento cada membro da Equipe expressa suas lições aprendidas e suas observações referentes ao período da *Sprint*. Sugestões de melhorias para as próximas iterações podem ser também apresentadas nesse momento.

Em empresas maiores, com projetos que possuem subdivisões, ocorre um evento no final da *Sprint* que é chamado Scrum de Scrums. Esse evento reúne todas as subequipes para se apresentar os resultados que cada subgrupo alcançou, bem como discutir sobre a relação entre eles [Britto, 2015].

A Figura 24 ilustra todo o processo da *Sprint*, desde o planejamento até a entrega do produto ou funcionalidade. É possível visualizar na parte debaixo da imagem os quadros com os papéis, cerimonias e artefatos existentes na iteração.

**d) Aplicação no Mercado**

Uma empresa chamada XSoft, conhecida pela sua forma ágil de desenvolver projetos, foi contratada para desenvolver uma ferramenta de criação de testes para TV Digital. A ferramenta deveria cobrir um conjunto de áreas funcionais descritas numa norma. O projeto tinha cronograma fechado de 6 meses, com possibilidade de extensão.

Durante a reunião de planejamento do projeto, o gerente do projeto juntamente com o arquiteto de software solicitaram ao cliente tudo que eles consideravam necessário para o desenvolvimento do produto. Sendo assim, um documento de requisitos foi gerado ao final da reunião, contendo informações gerais do que era esperado na entrega. O plano de projeto foi fechado com o custo, o escopo, os recursos que foram usados e o cronograma.

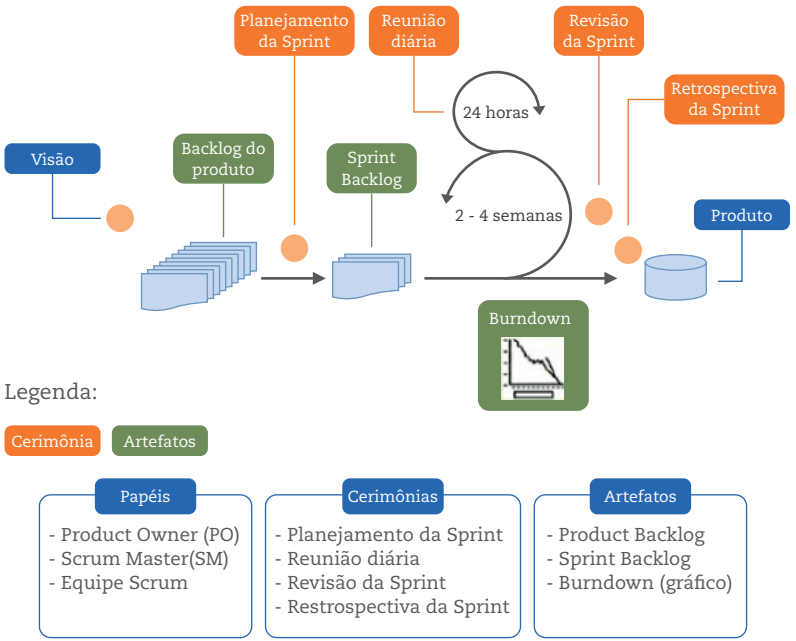



**Figura 24**. Esquema completo de uma Sprint

A partir de então, a Equipe juntou-se (gerente, arquiteto e desenvolvedores) para discutir sobre o projeto. A Equipe não era grande, possuía apenas 5 desenvolvedores, além do arquiteto e do gerente.

Foi criado um *Product Backlog* contendo todas as atividades que a equipe julgou necessárias para a implementação do produto, a partir dos requisitos levantados pelo cliente. As atividades foram distribuídas em *Sprints*, às quais foram ordenadas por prioridade, de acordo com o que mais agregaria valor ao cliente. Inicialmente priorizou-se a modelagem da arquitetura do software e sua implementação. O marco da primeira Sprint foi a base da ferramenta de geração testes, com uma interface de apresentação. A partir das iterações seguintes, começaram a ser entregues pequenos *engines* de geração de teste para cada área funcional existente na norma.

Por ser uma Equipe pequena, o próprio gerente, tendo um conhecimento geral do produto e das necessidades do cliente, ficou como o *Product Owner*. O *Scrum Master* iniciou como sendo o arquiteto de software, mas no decorrer das *Sprints* foi sendo revezado por cada membro da Equipe.

Durante a iteração inicial, percebeu-se que a curva do gráfico de *Burndown* subiu com relação à curva ideal. Nas reuniões diárias foi discutido sobre a subida e alguns membros da Equipe relataram problemas durante as atividades, o que gerou certo atraso no fechamento das mesmas. Contudo, faltando poucos dias para o final da *Sprint*, a curva voltou a ficar próxima da curva ideal, estando abaixo desta, simbolizando que a entrega na data prevista não estava ameaçada.

A cada final da iteração, o cliente participou junto com toda a equipe da reunião de revisão da *Sprint*, onde foi apresentada a funcionalidade ou incremento acordado com o ele para aquele momento. Estando tudo na forma esperada, o cliente aprova e a próxima *Sprint* é planejada.

#### 6.3.2.2. eXtreme Programming

Normalmente conhecida como XP, está abordagem ágil é um conjunto de técnicas condensadas de engenharia de software tradicional, adaptadas para seu uso em equipes pequenas que repetem rapidamente seus ciclos de desenvolvimento.

XP não tenta resolver todos os aspectos da engenharia de software, centrando-se fundamentalmente em quatro atividades: Codificar, Testar, Escutar e Desenhar. Kent Beck (o pioneiro de XP) dizia o seguinte sobre essas quatro atividades: "*Codifica-se porque se não codificarmos, não se fez nada. Testa-se porque se não testarmos, não se sabe se terminou a codificação. Escuta-se porque se não escutarmos, não se sabe o que codificar ou o que testar. E se desenha para poder seguir codificando, testando e escutando indefinidamente*" [Boria, Rubinstein e Rubinstein, 2013].

O XP é movido sobre um princípio chamado YAGNI - *You aren't gonna need it* (Você não vai precisar disto). Sendo assim, o objetivo é manter o *design* sempre simples, evitando aumentar a complexidade com a criação de funcionalidades que não sejam realmente necessárias.

Ao manter o *design* simples e fazer somente aquilo que for necessário, poupa-se tempo, pois se deixa de escrever código que não precisa e se tem mais tempo para concentrar-se na qualidade do que realmente é importante e que vai agregar valor para a demanda real e atual do cliente [Gomes, 2014].

**a) Esquema de funcionamento do XP**

A *eXtreme Programming* é uma metodologia que procura a máxima imersão possível do cliente no projeto, inclusive nas etapas de codificação e teste de aceitação.

Na etapa de planejamento de uma iteração, é solicitado ao cliente que ele descreva de forma bem livre tudo que espera

do produto. Esta descrição feita pelo cliente é chamada "histórias de usuário". Esse documento geralmente é escrito de forma metafórica, ambígua e genérica. Também é solicitado que o cliente atribua prioridades a cada história.

A equipe de desenvolvimento coleta as histórias criadas pelo cliente e passa para a fase de planejamento da iteração. Nesta fase, as histórias são refinadas através de processos repetitivos conhecidos como *Spikes*. A *ideia* por trás dessa etapa é de eliminar riscos, definir tarefas e estimativas de alto nível, e criar um *design* simples, sempre focando no que agrega valor ao cliente.

A fase seguinte é a interação, onde ocorre a codificação e a integração contínua. A codificação é feita usando o método de *Pair Programming* (Programação em Pares). A ideia por trás disso é que com duas pessoas trabalhando juntas fica muito mais fácil e rápido de resolver problemas complexos.

A codificação em XP é orientada a testes de unidade. Sendo assim, primeiramente são criados os testes para então a partir deles iniciar-se a codificação das funcionalidades. O fato de ser uma etapa de puramente codificação não exclui a participação do usuário final. O contato com o cliente nesse momento ajuda a equipe a alinhar pontos de dúvida nas funcionalidades que naturalmente surgem durante esse momento.

Ao final de cada ciclo de implementação em pares, é realizada uma integração através da prática de integração contínua; e o código-fonte é coletivo, ou seja, pertence a todos os membros da equipe, e deve ser escrito de acordo com os padrões definidos pelo próprio time [Gomes, 2014].

As entregas em XP são feitas em períodos curtos, idealmente a cada duas semanas. Esta é uma forma de manter o cliente sempre interessado e comprometido com o projeto, já que o *feedback* assim obtido aumenta o valor do produto

[Boria, Rubinstein e Rubinstein, 2013]. Ao receber o entregável, o cliente realiza os testes de aceitação. Esta é uma etapa muito crucial, pois de acordo com o *feedback* dado pelo cliente é que a equipe saberá se o incremento está de acordo com o esperado e poderá ser realizada uma *release* parcial.

Na Figura 25 está ilustrado o esquema completo da metodologia XP. É possível visualizar as diferentes fases e a comunicação entre elas.

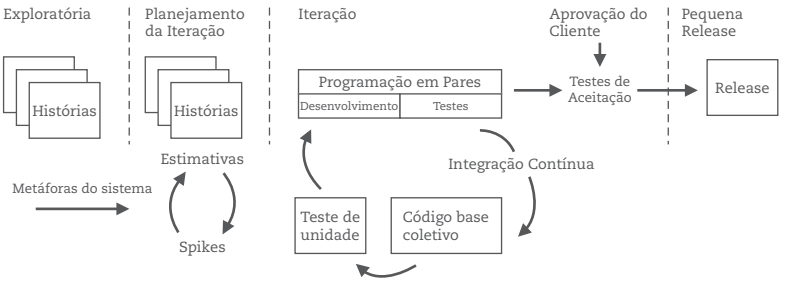



**Figura 25**. Esquema de funcionamento da metodologia ágil XP

### 6.3.2.3. Kanban

O método Kanban é parte de uma família de sistemas que se denomina "pull", ou de puxar, contra o enfoque tradicional de sistemas "push" ou de empurrar. Outra maneira de ver a diferença entre uns e outros sistemas é que o sistema "pull" é guiado pela demanda, enquanto que o sistema "push" é guiado pela produção [Boria, Rubinstein e Rubinstein, 2013].

O Kanban não possui as iterações que foram vistas anteriormente no Scrum e no XP. Ele é desacoplado no que diz respeito a planejamento, priorização, desenvolvimento e entrega. Em outras palavras, estas atividades podem desta forma ter sua própria cadência (repetições que se sucedem em

intervalos regulares) de forma a melhor ajustarem-se às necessidades que o processo demanda.

Em virtude desta forma desacoplada de trabalho, o Kanban funciona como um fluxo contínuo, de forma que as entregas ocorrem ainda que apenas uma parte do que foi planejado esteja pronto. É evidente que assim como as demais metodologias ágeis, deve-se entregar algo funcional e útil para o cliente, sempre priorizando o que agrega valor e evitando-se o desperdício. Sendo assim, não se deve pensar que o fato do Kanban ter essa ideia "livre" de entrega ocasionará numa entrega "quebrada" sem que haja algo funcional.

Dentre os métodos ágeis, o Kanban é considerado o menos prescritivo. Para se ter uma ideia de quão pouco prescritivo é o Kanban, ele apenas possui três prescrições:

- Visualizar o fluxo de trabalho (*workflow*);
- Limitar o trabalho em progresso;
- Gerenciar e medir o fluxo.

Kanban é uma palavra japonesa que significa "cartões visuais" e é usada como uma ferramenta no sistema de produção da Toyota. Quando aplicado no desenvolvimento de software, cada cartão é indexado com uma atividade que representa um valor existente na história de usuário. Em outras palavras, os cartões são usados como forma de controlar o que está sendo feito, o que foi feito e o que falta fazer [Davies e Sedley, 2009].

As Figuras 26 e 27 mostram exemplos de quadros de atividades do Kanban. Nota-se que cada coluna representa um estado da atividade. Ou seja, quando uma atividade ainda não foi iniciada, ela se encontra no estado "A Fazer". A partir do momento que ela é iniciada, o responsável por ela move o cartão para a coluna "Fazendo". Ao finalizar-se uma atividade, o cartão é movido para a última coluna, simbolizando que está "Feito".

No exemplo mostrado na figura 26, é possível visualizar a coluna "Item de *Backlog*" que vem antes da coluna "A Fazer". Esta primeira contém todas as atividades (todos os cartões) que foram atribuídos para esse fluxo contínuo. Vale frisar que o quadro de atividades não é sempre do mesmo jeito. O Kanban é uma das metodologias mais adaptativas, permitindo uma grande flexibilidade. No caso do quadro, ele pode ter mais colunas, detalhando um pouco mais as etapas (ver figura 27).

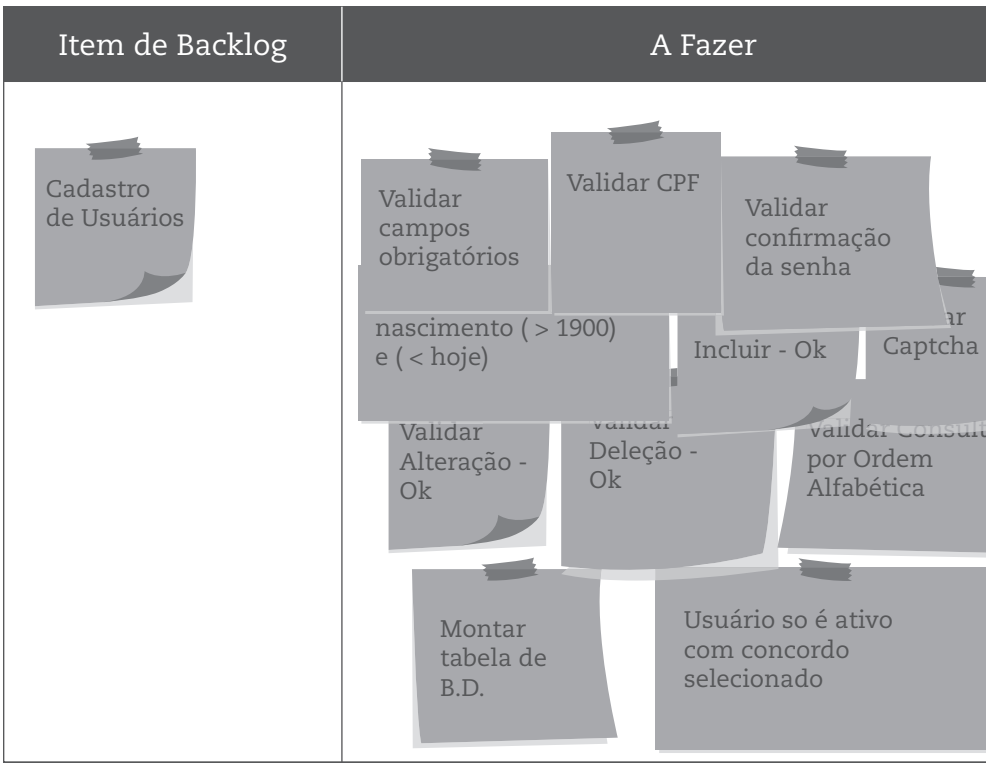
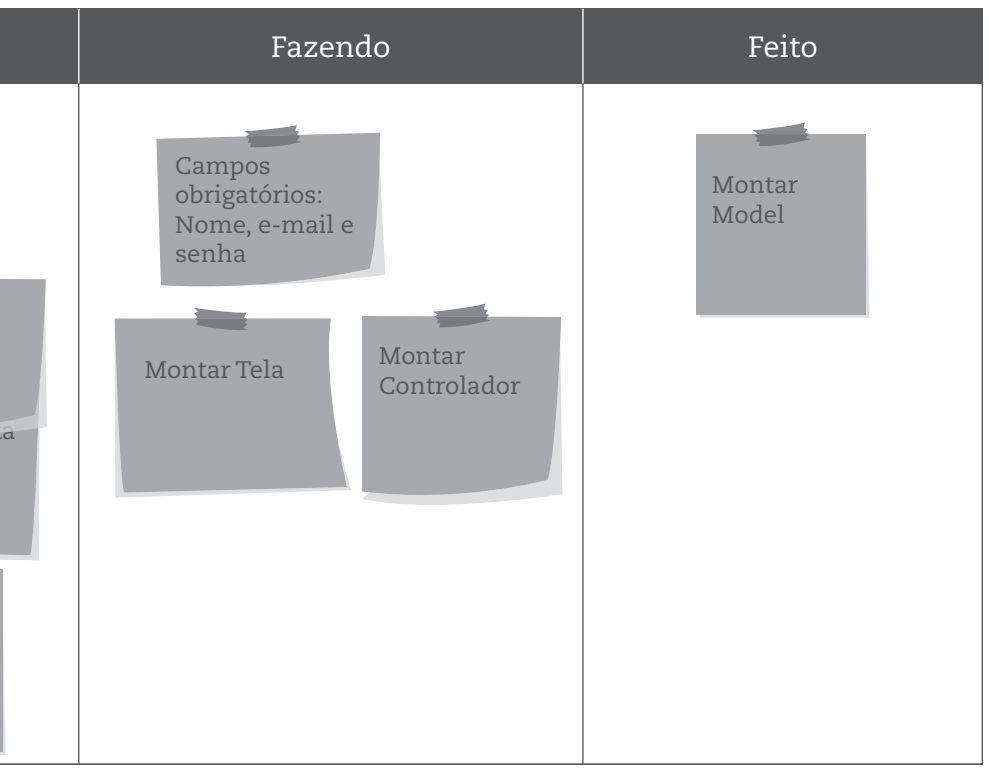



**Figura 26**. Quadro de atividades do Kanban


Fazendo
Feito
Campos obrigatórios: Nome, e-mail e senha
Montar Tela
Montar Controlador
Montar Model

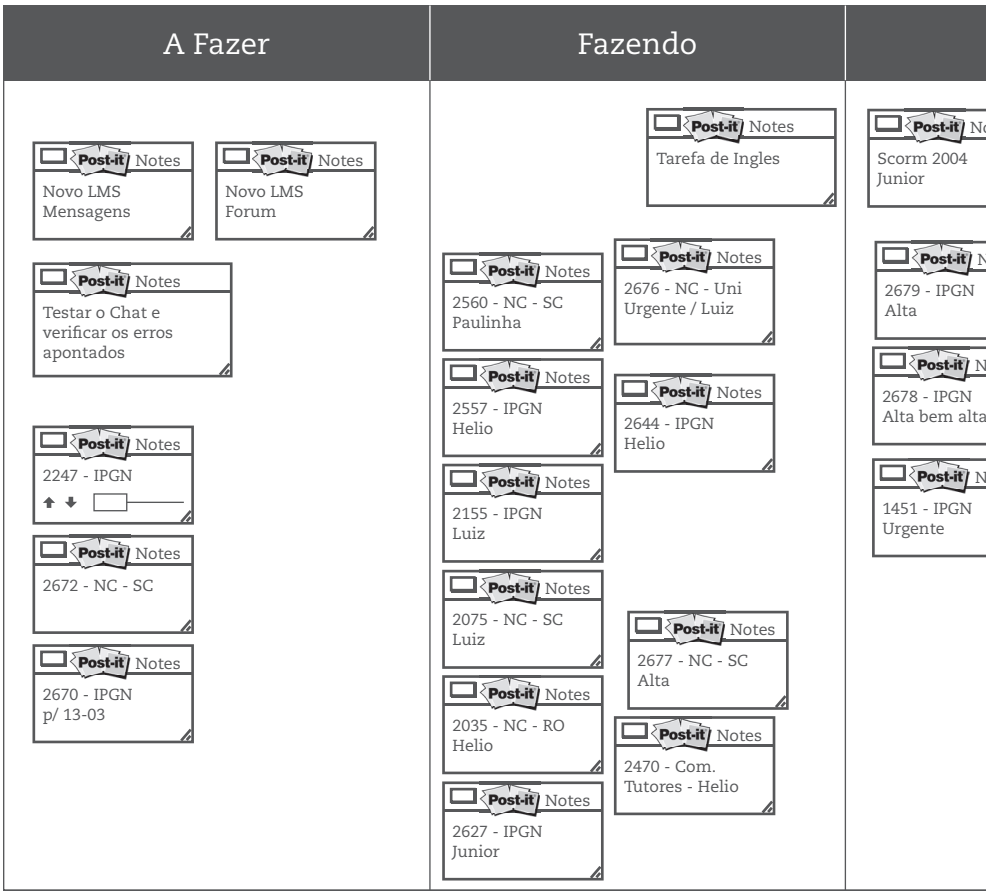



**Figura 27**. Um quadro de atividades do Kanban com um maior nível de detalhes (mais colunas).


Feito
Post-it Notes
17/03
SBQS
Checado
Post-it Notes
2660 - IPGN
Post-it Notes
2659 - IPGN
Post-it Notes
2656 - IPGN
Post-it Notes
2655 - IPGN
Post-it Notes
2591 - IPGN
Post-it Notes
2581 - IPGN
Post-it Notes
2648 - JF - SC
Post-it Notes
2654 - NC - SC
Post-it Notes
2647 - NC - SC
Post-it Notes
2646 - NC - SC
Post-it Notes
2129 - NC - SC
Post-it Notes
2584 - NC - Uni
Post-it Notes
2584 - NC - Uni
Não Planejado
Post-it Notes
Post-it Notes
Post-it Notes
Post-it Notes
2652 - NC / SE
Luiz
Post-it Notes
Post-it Notes
2665 - IPGN
Não efetivado
Problemas
Post-it Notes
2674 - NC - Uni
Junior
Post-it Notes
17/03
SBQS

No método Kanban existe um limite para o número de tarefas em cada coluna. O Objetivo é identificar claramente a capacidade do sistema de equilibrar a demanda em relação à competência da equipe [Boria, Rubinstein e Rubinstein, 2013]. Para ajudar na questão do limite de tarefas em cada coluna, existe uma variável usada no gerenciamento de tempo chamada *lead time*, que é o tempo médio para se completar um item, de forma que o processo deve ser continuamente optimizado para tornar esse tempo cada vez menor e mais previsível [Gomes, 2014].

## 6.4. Considerações Finais

Em virtude do que foi discutido neste capítulo, pode-se concluir que não existe uma metodologia universal para gerir projetos. Seja ágil ou tradicional, a metodologia usada deve ser aquela que melhor adequa-se ao tipo de projeto a ser desenvolvido.

As metodologias ágeis surgiram para preencher lacunas que a gestão tradicional de projetos não conseguia suprir. Os precursores do manifesto ágil estudaram com muita cautela as razões da crise que envolvia o gerenciamento de projetos. Focaram em elaborar princípios e valores que pudessem transformar o ambiente de trabalho e a forma como lidar com o cliente. Isso de fato é um ponto de grande melhoria.

Contudo, isso não quer dizer que ser ágil é sinônimo de ser bom. Forçar a usar métodos ágeis em qualquer tipo de cenário pode não ser proveitoso e acabar prejudicando as entregas e a evolução de um projeto.

O mais correto a se fazer é estudar bem a proposta e saber em qual forma de gestão ela melhor encaixa-se, tendo em vista as necessidades do cliente e conhecendo bem a equipe de desenvolvimento.

A Tabela 18 apresenta um resumo sobre a forma como cada uma das duas formas de gestão (tradicional e ágil) comporta-se em determinados cenários [Eder, Conforto e Amaral, 2015].

**Tabela 18**. Comportamento das gestões ágil e tradicional na forma de lidar com determinadas características.

| Característica | Abordagem de gerenciamento de projetos tradicional | Abordagem de gerenciamento ágil de projetos |
|---|---|---|
| 1) A forma de elaboração do plano do projeto | Há um único plano de projeto, que abrange o tempo total do projeto e contém os produtos, entregas, pacotes de trabalho e atividades. | Há dois planos de projeto: a) um plano geral que considera o tempo total de duração do projeto, mas que contém apenas os produtos principais do projeto; b) um plano de curto prazo (iteração) que contém apenas as entregas e atividades referentes a uma fração de tempo do projeto. |
| 2) A forma como se descreve o escopo do projeto | Descrição exata do resultado final por meio de texto, com normas do tipo contratuais, números objetivos e indicadores de desempenho. | Descrição do resultado final de maneira abrangente, desafiadora, ambígua e metafórica. |
| 3) O nível de detalhe e padronização com que cada atividade do projeto é definida | As atividades são descritas de maneira padronizada e organizadas em listas do tipo WBS. Contêm códigos e são classificadas em conjuntos de pacotes de trabalho, entregas e produtos do projeto. | Não há um padrão para a descrição das atividades, que podem ser escritas na forma de estórias, problemas, ações ou entregas. E não há uma tentativa de organização, apenas a priorização do que deve ser executado no momento. |
| 4) O horizonte de planejamento das atividades da equipe de projeto | As listas de atividades são válidas para o horizonte total do projeto. | As listas de atividades são válidas para uma iteração, que é definida como uma fração do tempo total do projeto. |

| Característica | Abordagem de gerenciamento de projetos tradicional | Abordagem de gerenciamento ágil de projetos |
|---|---|---|
| 5) A estratégia utilizada para o controle do tempo do projeto | Empregam-se relatórios com indicadores de desempenho, documentos escritos, auditorias e análises de transições de fase. As reuniões da equipe não são frequentes. | Empregam-se dispositivos visuais que indicam entregas físicas do resultado final (cartazes, autoadesivos, etc.). As reuniões são curtas e frequentes. |
| 6) A estratégia utilizada para a garantia do atingimento do escopo do projeto. | O gerente de projeto avalia, prioriza, adiciona ou altera as atividades do projeto para que os resultados estejam em conformidade com o escopo do projeto assinado com o cliente. | O cliente avalia, prioriza, adiciona ou altera o produto final do projeto, conforme a experiência com os resultados alcançados. A equipe altera as atividades para obter os resultados propostos pelo cliente. |

07

# GESTÃO DE PESSOAS EM SOFTWARE

*Gustavo Henrique da Silva Alexandre*

A administração de recursos humanos tem sua origem no desenvolvimento empresarial e na evolução da teoria organizacional nos Estados Unidos. Segundo André Fisher (2002), trata-se de uma produção tipicamente americana, que procura suplantar a visão do departamento de pessoal. Um conceito que reflete a imagem de uma área de trabalho voltada prioritariamente para as transações processuais e os trâmites burocráticos.

O HRM - *Human Resource Management* teve seu início nos EUA a partir do nascimento do setor de departamento de pessoal por volta de 1890, ano em que a NCR Corporation cria seu *personal office*. As inquietações restringiam-se apenas às burocracias, aos procedimentos e processos que fizessem os empregados trabalharem de maneira mais efetiva. O que importava para o departamento eram os custos e cuidavam de toda papelada e dos procedimentos legais. Porém, conforme foi evoluindo a área foi-se percebendo que a mesma era muito mais abrangente e tinha influência em vários aspectos internos e externos da organização. Passou a se ter uma visão muito mais

humanista e que exigiu do gestor uma maior habilidade para lidar com as várias funções incorporadas a ele, deixando de ser operacional para ser estratégico [Springer e Springer, 1990].

Já o termo Gestão de Pessoas é mais recente. A gestão de pessoas vem para mudar as práticas e ideias a cerca da antiga área de recursos humanos. Atualmente até mesmo a relação de trabalho entre empregado e empresa mudou de nome de quadro de funcionários para colaboradores.

As pessoas como recursos precisam ser administradas, o que envolve planejamento, organização, direção e controle de suas atividades, já que passam, nesse caso, a serem considerados sujeitos passivos da ação organizacional, parte do patrimônio físico na contabilidade da organização.

As pessoas como parceiras são fornecedoras de conhecimentos, habilidades, competências e, sobretudo, o mais importante, de inteligência, que proporciona decisões racionais e dá significado e rumo aos objetivos globais da organização. A Tabela 19 faz um paralelo das características das visões das pessoas vistas como Recursos e Parceiros.

**Tabela 19**. Características das Pessoas como Recursos x Parceiros, (Chiavenato, 2008)

| Pessoas como Recursos | Pessoas como Colaboradores/Parceiros |
|---|---|
| • Empregados isolados nos cargos.<br>• Horário rigidamente estabelecido.<br>• Preocupação com normas e regras.<br>• Subordinação ao chefe.<br>• Fidelidade à organização.<br>• Dependência da chefia.<br>• Alienação em relação à organização.<br>• Ênfase na especialização.<br>• Executoras de tarefas.<br>• Ênfase nas destrezas manuais.<br>• Mão de obra. | • Colaboradores agrupados em equipes.<br>• Metas negociadas e compartilhadas.<br>• Preocupação com resultados.<br>• Atendimento e satisfação do cliente.<br>• Vinculação à missão e à visão.<br>• Interdependência entre colegas e equipes.<br>• Participação e comprometimento.<br>• Ênfase na ética e na responsabilidade.<br>• Ênfase no conhecimento.<br>• Inteligência e talento. |

Então, o parceiro da organização é quem deve ser privilegiado, pois está dentro do entendimento das parcerias do negócio da empresa. É quem domina o grupo de interesses que atua interna e externamente na organização, utilizando gestão do conhecimento, ética na responsabilidade, equipes, metas negociadas e compartilhadas, além de resultados com inteligência e talentos.

Segundo Chiavenato (1999), as definições para a Gestão de Pessoas são:

- Conjunto de políticas e práticas necessárias para conduzir os aspectos da posição gerencial relacionados com as pessoas ou recursos humanos, incluindo recrutamento, seleção, treinamento, recompensas e avaliação de desempenho;
- A gestão de pessoas é a função na organização que está relacionada com provisão, treinamento, desenvolvimento, motivação e manutenção dos empregados.

A gestão de pessoas constitui o principal ativo da organização [Chiavenato, 2000b]. Com isso, faz-se necessário transformar as organizações mais atentas aos seus funcionários. As organizações que estão conseguindo grande sucesso em seus negócios já perceberam isso, que somente alcançarão novos patamares de lucro quando investirem em seus parceiros, principalmente no que se refere aos seus colaboradores.

O modelo antigo de gestão não suporta mais as mudanças dos processos atuais que buscam, principalmente, reduzir custos, minimizar etapas de trabalho e agregar valores que são percebidos pelos clientes [Knapik, 2008].

As funções dentro da empresa tornaram-se mais amplas, o desenvolvimento da carreira dentro das organizações ficou transparente e as avaliações de desempenho deixaram de ser tão subjetivas e de ser usada como instrumento de punição

e julgamento dos colaboradores. A ideia de treinamento dos colaboradores deixou de ser visto pelos gestores como um custo e passou a ser vista como investimento real e o processo seletivo passou a se basear nas necessidades da empresa e não na subjetividade do gestor.

Ao lado dos avanços tecnológicos, descobre-se que a forma de lidar com as pessoas realmente mudou. Presencia-se um modelo diferente, objetivo, justo, equilibrado e mais transparente.

## 7.1. Fundamentos da Gestão de Pessoas

Desde a fase de recrutamento e seleção já se tem início ao processo de gestão de pessoas nas empresas. Com isso, as empresas assimilam talentos com um conjunto de políticas e práticas pré-definidas por uma organização com o objetivo de orientar comportamentos, relações e resultados das pessoas de forma coorporativa. A partir da contratação, outras formas de potencialização de talentos passam a vigorar, como o treinamento profissional, planos de carreira, salários e benefícios.

O processo de GP – Gestão de Pessoas ainda é entendido como uma série de atividades que fornecem valor a um cliente. O cliente do processo não é necessariamente um cliente externo da empresa, ele pode estar dentro da empresa – é o chamado cliente interno [Chiavenato, 1999]. Assim, os processos da Gestão de Pessoas são:

- Agregar pessoas: recrutamento e seleção de pessoas;
- Aplicar pessoas: desenhos de cargos, avaliação de desempenho;
- Recompensar pessoas: remuneração, benefícios e serviços;
- Desenvolver pessoas: treinamento, mudanças, comunicação;

- Manter pessoas: disciplina, higiene e segurança, qualidade de vida, relações com sindicatos;
- Monitorar pessoas: banco de dados, sistemas de informações gerenciais.

Essas políticas e práticas foram denominadas como os seis processos da Gestão de Pessoas. A Figura 28 ilustra esses processos.

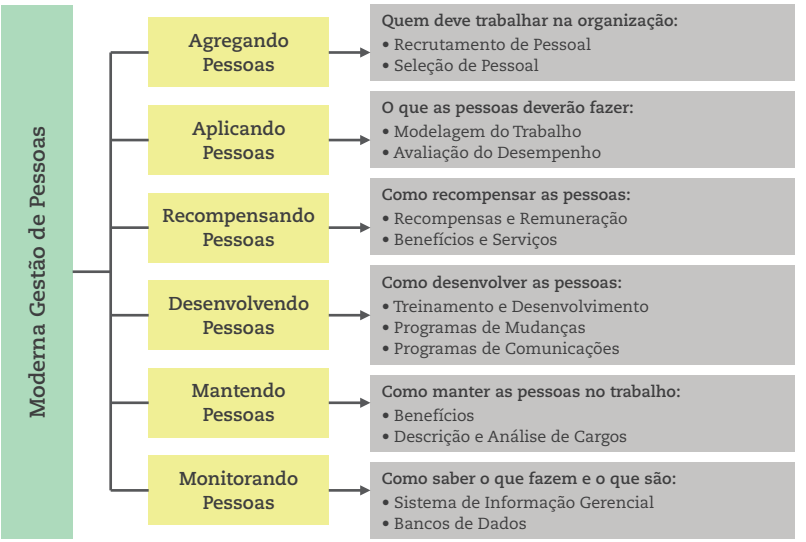



**Figura 28**. Processos de Gestão de Pessoas [Chiavenato, 2008]

### 7.1.1. Agregando pessoas

Chiavenato (1999) define que os processos de Agregar Pessoas são os de incluir novas pessoas, de suprimento de novos funcionários. Nesse processo estão as atividades de recrutamento e seleção de pessoas.

O processo de recrutamento é realizado a partir do surgimento da necessidade de se compor o quadro funcional de uma organização. O gestor é o responsável por levantar se

existe a necessidade de preenchimento de alguma vaga em aberto ou se há a necessidade da criação de algum cargo novo.

Havendo a necessidade de iniciar o processo de recrutamento, será preciso então que as responsabilidades, tarefas e obrigações sejam avaliadas e listadas pelo responsável do processo. Após essa etapa, será preciso definir o perfil do candidato. Para isso será necessário listar as características, experiências, habilidades e conhecimentos que o candidato ao cargo deverá possuir. A gestão do processo de admissão e desligamento é uma das mais importantes atividades da gestão de pessoas dentro da administração de recursos humanos. A responsabilidade do profissional de recursos humanos é grande e cabe a ele cuidar das pessoas que atuam nas empresas que são o principal ativo das mesmas.

Os principais motivos para se possuir uma área de recursos humanos dentro de uma organização são:

- Aumento do quadro de pessoal;
- Promoção ou transferência de colaborador;
- Demissão;
- Reposição temporária.

Independente do motivo, sempre é importante que seja feito um processo de recrutamento criterioso e que siga todas as etapas para garantir uma contratação bem feita.

Já o processo de seleção nada mais é que selecionar, dentre os candidatos recrutados, o(s) melhor(es) candidato(s) ao(s) cargo(s) disponível(eis).

E feito através de um processo de tomada de decisão, em que os candidatos são comparados entre si pelos seus perfis de competência e em relação ao perfil da vaga em aberto.

Chiavenato (1999) define o conceito de seleção como o processo que a organização realiza para escolher, dentre uma

lista de candidatos, as pessoas que melhor preencham os requisitos da vaga disponível, considerando as atuais condições do mercado.

O processo de seleção é realizado pelas pessoas que pertencem à área de recursos humanos da organização. Após se ter a definição das tarefas e responsabilidades a cerca do cargo e do perfil do candidato para a vaga em aberto, realizada na etapa de recrutamento, a seleção é realizada podendo ser utilizado diversos instrumentos tais como: análise curricular, testes de conhecimentos técnicos e de língua, dinâmicas de grupo, entrevistas e outras fontes necessárias.

### 7.1.2. Aplicando pessoas

Os processos de Aplicar Pessoas são aqueles utilizados para modelar as atividades que os funcionários irão realizar na organização, acompanhar e orientar seu desempenho. As atividades encontradas nesse processo são de desenhos de cargos e avaliação de desempenho [Chiavenato, 1999].

Chiavenato (1999) define, ainda, cargo como sendo uma composição de todas as atividades desempenhadas por um funcionário que pode ser agrupada em um todo unificado e que tem uma posição formal no organograma da empresa. O desenho organizacional é condicionado à estrutura de cargos que está condita no desenho.

O desenho de cargos, ou a modelagem do cargo, é feito especificando o que cada cargo deve fazer, os métodos de trabalho e das relações com outros cargos. É um processo que possibilita a organização do trabalho através das tarefas necessárias para o desempenho de um cargo específico [Chiavenato, 1999]

A modelagem de um cargo constitui a forma como cada cargo é estruturado e dimensionado, e que para isso é preciso definir quatro condições básicas que são:

- O conjunto de tarefas ou atribuições que o funcionário deverá desempenhar;
- Como as tarefas ou atribuições deverão ser desempenhadas;
- A quem o funcionário do cargo deverá se reportar;
- Quem o funcionário do cargo deverá supervisionar ou dirigir.

Sobre a avaliação do desempenho, um dos pioneiros na formalização da necessidade de se acompanhar o desempenho do trabalhador foi Taylor (1970). Ele dizia que era importante que fossem registrados os movimentos do trabalhador, os tempos na execução das tarefas, a sua produtividade, etc. Mas a avaliação de desempenho formal só surgiu na chamada escola de relações humanas e seus seguidores ligados à prática de gestão.

As primeiras formas de avaliação consistiam em listas informais de qualidade e alguns itens de desempenho, como assiduidade, pontualidade, zelo, etc., que o gestor utilizava pontuando cada um desses itens.

Com o tempo, as formas de avaliação foram evoluindo e se sofisticando, tornando-se mais abrangentes e apropriadas de forma a ter uma visão mais ampla das competências desejadas pela empresa, além de formas de avaliações que incluíam avaliações coletivas e ascendentes, como as avaliações 360º. Neste tipo de avaliação, subordinados e superiores avaliam-se mutuamente, possibilitando uma avaliação mais completa do ponto de vista da empresa como um todo. Questionários *online* são preenchidos e oferecem um parâmetro mais preciso do desempenho do colaborador, até mesmo para auxiliar no seu auto-desenvolvimento.

Para que o processo de avaliação de desempenho aconteça com êxito é muito importante que os gestores deem apoio ao processo, mesmo que ele não concorde totalmente com o

mesmo. O gestor precisa estar comprometido com o processo a fim de transmitir aos seus subordinados a importância e seriedade do processo de avaliação. Muitas vezes o insucesso e a não aceitação do gestor ao processo de avaliação deve-se a falta de compressão do gestor, por isso é de extrema importância também que o gestor esteja bem informado de como funciona todo o processo.

Os subordinados ao gestor devem conseguir compreender a importância do processo através de seu discurso e prática. O gestor deve ter o cuidado para que o processo de avaliação não seja visto como algo meramente burocrático e sem valor, ou algo que só é feito por desejo, promoção ou aumento salarial.

### 7.1.3. Recompensando pessoas

Conforme Chiavenato (1999), os processos de Recompensar Pessoas são utilizados para motivar e incentivar as pessoas, e satisfazer suas necessidades individuais. Nesse processo podem-se encontrar as atividades de remuneração e benefícios.

Com base na sua cultura, as organizações desenvolvem sistemas de recompensas que visam atender com qualidade o planejamento estratégico empresarial, de forma que venha provocar impacto direto na sua capacidade de atrair, reter e motivar os colaboradores. Essas recompensas colaboram com a satisfação e se tornam um incentivo para as pessoas atingirem os objetivos e a lucratividade da organização.

A remuneração, por sua vez, é o processo que envolve todas as formas de recompensas ou pagamento dadas aos colaboradores e que são decorrentes do emprego.

O salário pode ser nominal ou real, pode ser por unidade de tempo (mês ou hora de trabalho), por resultado ou por tarefa. O salário é importante para a pessoa (define seu padrão de vida) e para a organização (impacta seus custos), e depende

de fatores internos (organizacionais) e externos (ambientais). O desenho do sistema de remuneração deve levar em conta certos critérios para sua construção. A administração de salários é o conjunto de normas e procedimentos para estabelecer e/ou manter estruturas salariais equitativas na organização que tenham equilíbrio interno e externo. O equilíbrio interno é assegurado pela avaliação e classificação de cargos, enquanto o externo é assegurado por pesquisas salariais. Os métodos tradicionais de avaliação de cargos são: escalonamento simples, categorias predominadas, comparação por fatores e avaliação por pontos. As decorrências dos salários são os encargos sociais que alavancam os custos do trabalho nas organizações [Chiavenato, 2008].

O interesse das empresas é investir nas recompensas para os colaboradores de forma que recebam em troca contribuições/produtividade ao alcance dos seus objetivos.

A soma das recompensas diretas e indiretas constitui a remuneração. Nesse sentido, a remuneração abrange tudo quanto o colaborador alcança em consequência do trabalho que realiza em uma organização. No tocante às recompensas não financeiras oferecidas pelas organizações, afetam a satisfação das pessoas com o sistema de remuneração. Desta forma, pode-se afirmar que não é somente a remuneração que motiva os colaboradores na empresa.

No caso das organizações, fica claro comprovar esse tipo de motivação, quando o funcionário atua em uma função que lhe permite auto-realização, proporcionando a elevação de sua auto-estima. Quando isso acontece, a sua eficiência é comprovada e produz resultados, os quais superam a expectativa da empresa.

Ninguém motiva ninguém. Uma gerência competente cria condições e influencia os colaboradores a estarem motivados.

Para que seja possível um ambiente motivador, pessoas integradas, além de produtivas nas organizações, torna-se imprescindível um plano com critérios adequados de Gestão de Pessoas.

#### 7.1.3.1. Incentivo e Benefícios

Muitas empresas já oferecem benefícios flexíveis para todos os funcionários. Montando seu pacote de acordo com as suas necessidades.

Abaixo, segue a lista de benefícios e outros atrativos praticados por empresas brasileiras:

- Saúde: auxílio doença, assistência médica, assistência psiquiátrica e psicológica, cobertura para tratamentos de dependências química (droga/alcoolismo), homeopatia, infertilidade, terapia, acupuntura, *check-up*, plano médico com livre escolha, plano com cobertura também para agregados e aposentados, assistência ou consultório odontológico, cobertura para aparelhos ortodônticos, ambulatório, auxílio para aquisição de óculos e lentes de contato, desconto na compra de medicamentos e entrega no local de trabalho, programas de alongamento e relaxamento para atendentes, ginástica laboral durante o expediente, academia e SPA para executivos;
- Alimentação: tíquete, vale-supermercado, café da manhã grátis, lanchonete na empresa, refeições coletivas, cardápios diferenciados para funcionários que tem problemas como diabetes, colesterol e hipertensão;
- Educação e Desenvolvimento: seguro educação, bolsas de estudo, cursos de idiomas, instrução dos filhos, ensino supletivo, reembolso para cursos de graduação, pós-graduação e MBA, grande oferta de treinamento, inclusive à distância, oportunidades de estágio e carreira em outras

unidades do grupo e no exterior, biblioteca, palestras variadas (planejamento familiar, orçamento doméstico, segurança no trabalho, apoio pré-aposentadoria, etc.);

- Carreira: política de promoção com base em avaliação de desempenho para todos os funcionários, prioridade ao recrutamento interno, aconselhamento de carreira, grandes oportunidades de carreira para mulheres, programas de *trainees*;
- Formas de Remuneração e Auxílios: remuneração por competências e habilidade, remuneração variável, participação nos lucros e resultados (salários extras), 14º salário, programa de reconhecimento, previdência privada, seguro de vida, adicional por tempo de casa, ajuda no aluguel (em caso de transferência), empréstimo para aquisição de casa própria, financiamento para aquisição de automóvel, carro com despesas pagas, estacionamento, transporte da empresa (ônibus), venda de produtos fabricados pela empresa com desconto, auxílio para pais de filhos excepcionais, auxílio para compra de material escolar e assistência jurídica;
- Integração e lazer: incentivo à participação dos funcionários em projetos filantrópicos, forte política de recepção e integração aos novos, clube, colônia de férias, área de lazer na empresa com sala de jogos, leitura e ginástica, atividades de recreação para os filhos de funcionários, promoção de festas Juninas e de Natal;
- Comunicação Interna: segurança e confiança na gestão, sinergia entre chefes e subordinados, avaliação 360 graus para todos os funcionários, valorização das sugestões dos funcionários, linha direta e *ombudsman*[3] para

3 *Ombudsman* é um profissional contratado por um órgão, instituição ou empresa com a função de receber críticas, sugestões e reclamações de usuários e consumidores, devendo agir de forma imparcial no sentido de mediar conflitos entre as partes envolvidas (no caso, a empresa e seus consumidores).

reclamações, pesquisa de clima periódica para medir a satisfação dos funcionários, clareza e abertura na comunicação interna, ambiente de trabalho onde as pessoas sintam-se livres para participar, criar e ter iniciativa;

- Outras práticas: horário flexível de trabalho, jornada reduzida no verão, possibilidade de trabalho em locais remotos (casa, cliente, etc.), informalidade nos trajes, licença não remunerada para projetos pessoais, berçário, creche, sala de aleitamento para mães, loja de conveniência e outros serviços dentro da empresa (salão de beleza, etc.).

### 7.1.4. Desenvolvendo pessoas

Os processos de Desenvolver Pessoas são utilizados para capacitar, treinar e desenvolver pessoas. Aqui encontram-se as atividades de treinamento, mudanças e comunicação [Chiavenato, 1999].

Durante a Segunda Guerra Mundial, uma necessidade significativa de aumento na produção das empresas aconteceu, e, concomitante a isso, novos trabalhadores menos qualificados chegavam aos postos de trabalho. Nesse cenário, nos EUA, os ocupantes de um cargo juntaram-se a professores das áreas de ciências sociais para criar um programa de fácil aplicação para treinamento rápido de operários e técnicos. Desta junção, surgiu um programa bem sucedido chamado TWI – *Training Within Industry*, treinamento no local de trabalho, que, por sua eficácia, acabou sendo adotado em todo o mundo.

O TWI estabelecia que o treinamento do pessoal fosse responsabilidade do supervisor. Uma frase do programa dizia: "Se o subordinado não aprendeu é porque o supervisor não ensinou". Por sua atualidade, essa frase bem pode ser relembrada nos dias de hoje. Continua valendo a ideia de que o treinamento e o desempenho do pessoal são responsabilidade do gestor.

Nos dias atuais muitas vezes percebe-se que essa função é negligenciada, por várias razões:

- Pressa e correria no processo de gestão, com falta de tempo para orientação do pessoal;
- Ideia errada de que as empresas têm amplas escolhas na seleção de pessoal e que por isso devem contratar pessoas que já sabem;
- Ideia errada de que as pessoas, apenas elas, devem ser responsáveis pelo seu próprio desenvolvimento e devem "se virar";
- Falta de percepção do gestor sobre seu papel.

Em síntese, o gestor, com o apoio de seu pessoal, define quais são as carências e como deverão ser supridas. Com esses dados em mãos, estabelece-se um programa, que será posto em execução pelo grupo, de forma participativa e envolvente. Todos contribuem para o crescimento de cada um, cria-se então internamente uma organização que aprende.

### 7.1.5. Mantendo pessoas

Segundo Chiavenato (1999), os processos de Manter Pessoas são aqueles de criação de condições ambientais e psicológicas satisfatórias para o trabalho dos funcionários.

Vive-se numa sociedade que está em constante transformação e em um momento excitante para as organizações. A sociedade percebe que Qualidade de Vida e a Saúde estão tornando-se, cada vez mais, dimensões importantes na vida dos indivíduos, além de envolver outras dimensões tais como: dimensões física, intelectual, emocional, profissional, espiritual e social. Práticas inadequadas no ambiente de trabalho geram impacto negativo na saúde física e emocional dos colaboradores e na saúde financeira das empresas. Fatores

como: baixa motivação, falta de atenção, diminuição de produtividade e alta rotatividade criam uma energia negativa que repercute na família, na sociedade, além do sistema médico e de saúde.

Domenico de Masi (2001) afirma que se vive e trabalha numa sociedade do futuro, mas continua-se a usar os instrumentos do passado. Contudo, esse cenário não faz parte da realidade atual de empresas inovadoras e conscientes. As dez melhores empresas para se trabalhar [Guia Exame, 2014] transformaram o ambiente de trabalho e a saúde emocional e física em vantagem competitiva, com a convicção estratégica de que quanto mais satisfação o colaborador apresentar, mais retorno terá em produtividade, criando assim a visão de uma organização mais equilibrada e competitiva.

Segundo a Organização Mundial da Saúde, qualidade de vida é um conjunto de percepções que o indivíduo tem da vida em si no contexto dos sistemas de cultura e de valores em que vivem, e em relação as suas metas, expectativas, padrões e preocupações.

Programas de saúde são programas com o objetivo de ajudar pessoas a modificar seu estilo de vida em direção a um ótimo estado de saúde, sendo esta compreendida como o balanço entre a saúde física, emocional, mental, social e espiritual.

Os programas de saúde e qualidade de vida têm como objetivo auxiliar nessas mudanças de estilo de vida, combinando ações e campanhas para consciência, comportamento e envolvimento, que suportem suas práticas de saúde e previnam doenças.

A finalidade de um programa de qualidade de vida ou promoção de saúde nas organizações é encorajar e apoiar hábitos e estilos de vida que promovam saúde e bem estar entre todos os colaboradores e suas famílias durante toda a sua

vida profissional. Em outras palavras, pode-se dizer que os indivíduos deverão ser encorajados a serem protagonistas na promoção do programa, na medida em que gerenciam e cuidam da sua própria saúde, ganhando, desta forma, satisfação e crescimento, bem como, aumentando sua produção e reduzindo os custos para a empresa na qual trabalha.

### 7.1.6. Monitorando pessoas

Os processos de Monitorar Pessoas são utilizados para acompanhar e controlar o trabalho dos funcionários e analisar os resultados. Incluem nesse processo os bancos de dados e sistemas de informações gerenciais [Chiavenato, 1999].

Banco de dados é um conjunto de informações relacionadas entre si que se referem ao mesmo assunto, organizadas de uma forma prática para que o usuário possa obter as informações de forma fácil e acessível. Já o sistema de informação é o gerenciamento do fluxo de dados na organização e tem como objetivo principal auxiliar no planejamento, implementação e controle dos processos.

As informações que devem conter em um banco de dados para atender o capital humano são:

- Provisão de pessoal – contém informações sobre a visualização dos currículos cadastrados, triagem de candidatos por perfil estabelecido, agenda de entrevistas e acompanhamento do processo seletivo;
- Avaliação de desempenho – armazenamento das avaliações individual ou geral dos funcionários, bem como do desempenho dos colaboradores;
- Avaliação e pesquisa – informações a respeito da eficácia dos treinamentos, tendo como base pesquisas de opiniões entre os colaboradores, além de demonstrações sobre o clima organizacional;

- Administração de Treinamento – informações acerca das necessidades de treinamentos com base no perfil do cargo.

De acordo com Chiavenato (2008), na área de Recursos Humanos, ou Capital Humano, os vários bancos de dados interligados permitem obter e armazenar informações acerca de diferentes níveis de complexidade, tais como:

- Dados pessoais sobre cada colaborador, formando um cadastro de pessoas;
- Dados sobre os ocupantes de cada cargo, formando um cadastro de cargos;
- Dados sobre os salários e incentivos salariais, formando um cadastro de remunerações;
- Dados sobre os benefícios e serviços sociais, formando um cadastro de benefícios;
- Dados sobre candidatos (cadastro de candidatos), sobre cursos e atividades de treinamento, formando um cadastro de treinamento.

Um sistema de informação de recursos humanos ou capital humano requer análise da organização ou de seus subsistemas e das respectivas necessidades de informação. Um sistema de informação deve identificar e envolver toda a rede de fluxo de informação para ser projetado para cada grupo de decisão. A importância deve ser dada na necessidade de informação e não no uso da informação, como se faz convencionalmente. O sistema de informação é a base do processo decisório da organização. Para Chiavenato (1993), um sistema de informações utiliza como fonte de dados elementos fornecidos por:

- Banco de Dados de Recursos Humanos;
- Recrutamento e Seleção de Pessoal;
- Treinamento e Desenvolvimento de Pessoal;

- Avaliação de Desempenho;
- Administração de Salários;
- Higiene e Segurança;
- Estatísticas de Pessoal;
- Registros e Controles de Pessoal a respeito de faltas, atrasos, disciplina etc.

De acordo com Chiavenato (2008), o ponto de partida de um sistema de informação de recursos humanos é o banco de dados. O objetivo final de um sistema de informação é abastecer as gerências de informações sobre seu capital humano. Um sistema de informação é, por definição, um sistema por meio do qual os dados são obtidos, processados e transformados em informações, de forma esquematizada e ordenada, ou seja, sob a forma de relatórios, documentos, índices, listagens, medidas estatísticas de posição ou tendências, para servirem de subsídio ao processo de tomada de decisão.

#### 7.1.6.1. Auditoria de recursos humanos

Uma auditoria tem por objetivo revisar e controlar o programa em desenvolvimento, analisar as políticas adotadas pela organização e avaliar o seu funcionamento. A auditoria em Recursos Humanos, segundo Chiavenato (1994), pode ser definida como sendo "a análise de todas as políticas e práticas realizadas de pessoal de uma organização, e a avaliação do atual funcionamento e seguida de sugestão para melhoria". A empresa busca conhecer-se melhor para poder, assim, alcançar seus resultados de uma maneira mais eficiente.

Segundo Chiavenato (2000b) "o propósito principal da auditoria de recursos humanos é mostrar como o programa está funcionando, localizando práticas e condições que são prejudiciais

à organização ou que não estão compensando o seu custo, além de práticas e condições que devem ser acrescentadas".

A auditoria indica as falhas e problemas, determina as possíveis soluções e sugestões, além de salientar os aspectos positivos e tentar melhorá-los. Segundo Chiavenato (2000b), a auditoria de recursos humanos, dependendo da política da organização, poderá ser bastante profunda, focando em um ou em todos os seguintes níveis de profundidade:

- Resultados: estão relacionados com as realizações, bem como com os problemas existentes;
- Programas: incluem práticas e procedimentos detalhados que compõem o programa;
- Políticas: podem ser explícitas e implícitas;
- Filosofia da administração, seus valores, objetivos etc.;
- Teoria: explicar a filosofia, as políticas, práticas e contínuos problemas.

Os principais aspectos que a Auditoria de Recursos Humanos permite verificar são:

- Objetivos e expectativas quanto à administração de Recursos Humanos, em relação à quantidade, qualidade, tempo e custo;
- Contribuição do recurso humano aos objetivos e resultados da organização;
- Eficiência e eficácia quanto ao treinamento, desenvolvimento de pessoas, bem como remuneração, benefícios sociais, relação sindical, entre outros;
- Clima organizacional;
- Política de Recursos Humanos.

As organizações preferem contratar consultores externos para desenvolver auditorias. Contudo, as organizações mais bem estruturadas formam comissões de auditoria. Tanto os

auditores externos quanto internos são profissionais capacitados para analisar as operações trabalhistas, direitos e deveres do empregado e do empregador.

Chiavenato (2000b) afirma que as principais mudanças que alteraram o cenário da Auditoria de Recursos Humanos são:

- Mudança nas filosofias e teorias administrativas. O empregado passa a ter influências positivas e significativas no incentivo e êxito da organização;
- Mudança no papel do governo. Aumenta a intervenção deste, a fim de propiciar a segurança econômica e o pleno emprego. O governo passa a proteger mais os empregados;
- Expansão dos sindicatos;
- Elevações salariais, aumento dos custos da mão de obra e também maiores oportunidades de vantagem competitiva na administração de pessoal;
- Competência internacional mais agressiva;

Quanto maior e mais descentralizada à organização, maior a necessidade de uma cobertura sistemática de auditoria, diz Chiavenato (2004). Nesses casos, a auditoria serve como reforço ao treinamento dos executivos que atuam na área de recursos humanos, e não tem um caráter fiscalizador, podendo desenvolver forte impacto educacional, pois permite relacionar a qualidade da administração de recursos humanos com os diversos indicadores de eficiência da organização. Ainda segundo Chiavenato (2004), a auditoria permite verificar: até que ponto a política de recursos humanos baseia-se em uma teoria aceitável; até que ponto a prática e os procedimentos são adequados à política e às teorias adotadas.

## 7.2. Gerenciamento de recursos humanos do projeto Segundo o PMBOK

O gerenciamento de recursos humanos em projetos no PMBOK – *Project Managemente Body of Knowledge* [PMI, 2009] inclui os processos que organizam e gerenciam a equipe do projeto. A equipe do projeto é composta de pessoas com funções e responsabilidades atribuídas para a conclusão do projeto. Embora seja comum falar-se de funções e responsabilidades atribuídas, os membros da equipe devem estar envolvidos em grande parte do planejamento e da tomada de decisões do projeto. O envolvimento dos membros da equipe desde o início acrescenta especialização durante o processo de planejamento e fortalece o compromisso com o projeto. O tipo e o número de membros da equipe do projeto muitas vezes podem mudar conforme o projeto desenvolve-se.

A equipe de gerenciamento de projetos é um subconjunto da equipe do projeto e é responsável pelas atividades de gerenciamento de projetos, como planejamento, controle e encerramento. Esse grupo de pessoas pode ser chamado de equipe principal, executiva ou líder. Em projetos menores, as responsabilidades de gerenciamento de projetos podem ser compartilhadas por toda a equipe ou administradas unicamente pelo gerente de projetos.

Os processos de gerenciamento de recursos humanos do projeto incluem [PMI, 2009]:

- Planejar os recursos humanos
- Mobilizar a equipe do projeto
- Desenvolver a equipe do projeto
- Gerenciar a equipe do projeto

Esses processos interagem entre si e também com processos nas outras áreas de conhecimento. As seguintes situações são exemplos de interações que exigem planejamento adicional [PMI, 2009]:

- Depois que os membros da equipe inicial criam uma estrutura analítica do projeto, talvez seja necessário contratar ou mobilizar outros membros da equipe;
- Conforme outros membros da equipe do projeto são contratados ou mobilizados, seus níveis de experiência podem aumentar ou diminuir o risco do projeto, criando a necessidade de um planejamento de riscos adicional;
- Quando as durações das atividades são estimadas antes que todos os membros da equipe do projeto sejam conhecidos, os níveis reais de competência dos membros da equipe contratados ou mobilizados podem provocar mudanças nas durações das atividades e no cronograma.

## 7.3. Estudos de Caso

Esta seção discutirá a aplicação do gerenciamento de recursos humanos em 3 estudos de caso reais.

### 7.3.1. Remédio para o absenteísmo - A Chesf investe nos funcionários e combate o absenteísmo

Com o intuito de diminuir o absenteísmo, ou ausência do funcionário no trabalho, a CHESF - Companhia Hidroelétrica do São Francisco investiu em ações que buscasse atenuar os fatores que levam os funcionários ao absenteísmo tais como atrasos, problemas pessoais, doenças que resultam em licenças médicas, condições inadequadas do ambiente de trabalho, falta de motivação, entre outros [Bispo, 2006].

O absenteísmo provoca muitos problemas como desorganização, queda na qualidade e produtividade de serviços

oferecidos, e isso causa um transtorno para os gestores e para a própria empresa. Mas a organização tem desenvolvido ações que estimulam a presença dos colaboradores no ambiente de trabalho, através de ações que valorizam o próprio ser humano.

A ação que havia sobre o absenteísmo na CHESF era insipiente e sem uma visão mais holística em relação à saúde e à qualidade de vida no trabalho. Consistia na emissão de um relatório para os gerentes, contendo informações sobre os nomes dos funcionários que estiveram ausentes do trabalho, o motivo e o período de afastamento. O objetivo era de apenas orientar os líderes, para que cumprissem a meta de 1,5% ao ano de absenteísmo-doença, criada em meados dos anos 90.

Então, a CHESF decidiu criar programas e ações preventivas e corretivas que contribuíssem efetivamente na saúde dos funcionários. Dentre esses, destacam-se [Bispo, 2006]:

- Criação do Plano de Assistência Patronal (PAP) e da Fundação Chesf de Assistência e Seguridade Social (FACHESF);
- Realização de exames médicos periódicos, que excedem às exigências da NR-7 como, por exemplo, ultra-sonografia, densiometria óssea, colposcopia, citologia oncótica, tonometria binocular e teste ergométrico;
- Estímulo a consultas médicas em cardiologia, oftalmologia, ginecologia, odontologia e avaliação psicológica;
- Adoção dos programas de Preparação para a Aposentadoria; Monitoramento Biopsicossocial, Prevenção e Dependência Química, Ginástica Laboral, entre outros.

Além dessas ações, a CHESF através da DABT - Divisão de Saúde e Bem-Estar no Trabalho, subordinada ao DAH - Departamento de Administração de RH, selecionou um departamento de nível operacional, onde o índice de absenteísmo era considerado preocupante.

Foi realizado um diagnóstico no departamento selecionado e planejadas as ações a serem realizadas. Decidiu-se pela criação de um projeto piloto chamado "Promovendo a Saúde e a Qualidade de Vida" que teve ao todo 32h de atividades durante o expediente dos funcionários. Assim sendo, a DABT promoveu mini-palestras, teórico-prático com a participação dos funcionários, através da exibição de filmes, utilização de técnicas de dinâmicas de grupo e realização de oficinas.

Como resultado, foi realizada uma análise comparativa dos indicadores do absenteísmo-doença, entre janeiro e junho de 2006, em relação ao mesmo período de 2005. Constatou-se, no setor operacional selecionado, uma redução de 39,49% do absenteísmo-doença, uma diminuição de 87% dos acidentes de trabalho e a participação de 100% dos funcionários nos exames médicos periódicos [Bispo, 2006].

### 7.3.2. Ambiente "zen" faz empresa crescer 80%

Investir no crescimento e na valorização dos funcionários gera resultados. Para comprovar isso, a empresa Apdata investiu só na construção da nova sede cerca de US$ 2 milhões. O ambiente de trabalho foi redecorado utilizando a técnica milenar chinesa de harmonização do ambiente, Feng Shui. Além disso, foi construída uma capela ecumênica, fumodrómo, academia, jardim de inverno, sala de cromoterapia, churrasqueira e sala de massagem [Administradores.com, 2006].

A empresa também permite que seus colaboradores participem das decisões da empresa e possuem um canal aberto para manifestar ideias e sugestões sobre melhorias nos processos de trabalho.

Com isso a empresa, em apenas três anos, aumentou seu faturamento em 80%, aumentando também seu número de funcionários, de 50 para cerca de 200 pessoas. Além disso, a

produtividade aumentou em 85% e o *turnover* (rotatividade) caiu para praticamente zero [Administradores.com, 2006].

### 7.3.3. Equipe NASA

A NASA lançou um desafio para seu engenheiro-chefe Brian Muirhead: construir um protótipo que pousasse em marte um custo menor do que o filme Titanic. O objetivo desse desafio é iniciar a exploração planetária com baixo custo.

Para que fosse possível obter sucesso na missão, foi preciso superar vários desafios simultaneamente, como o de desenvolver tecnologias e equipamentos de ponta, criar uma equipe coesa e com alto desempenho. Brian afirmou que a mensagem para sua equipe era simples: se falhou, tente novamente, mesmo que isso signifique ter que refazer 30 vezes. A equipe é que faz a diferença. As pessoas podem ser treinadas para melhorar e aperfeiçoar seus talentos, mas a confiança entre os membros da equipe e no seu líder é fundamental, principalmente em negócios de alto risco.

Para que uma pessoa compusesse sua equipe, Brian diz que precisa ver a centelha nos olhos do candidato. Mesmo depois de avaliar o currículo, saber onde o candidato estudou, suas experiências profissionais, é preciso observar o desejo vibrante de trabalhar no projeto. Brian refere-se a isso como a inteligência emocional.  E completa dizendo que prefere os candidatos com perfis mais generalistas que os com os perfis mais especialistas. Isso se deve ao fato que na NASA os problemas são muito amplos, abrangem variadas áreas e assim o perfil generalista é mais adequado.

Por fim, o ultimo destaque dado por Brian foi sobre a comunicação entre os membros da equipe, especialmente no que se refere à tomada de decisão de risco. Para isso, Brian sugere a criação de uma linguagem específica para que a equipe

possa avaliar os riscos. A cultura de comunicação é fundamental dentro da equipe, pois um erro ou falha em comunicação pode por tudo a perder no projeto. E cabe ao gestor ser a pessoa responsável por manter o grupo coeso, resolvendo os conflitos internos e as diferenças entre os membros da equipe.

## 7.4. Considerações Finais

O maior valor de uma empresa está nas pessoas que nela trabalham. É no capital humano da empresa que está hoje o principal diferencial competitivo. Ou seja, os talentos individuais que somados e unidos fazem a diferença coletiva para a empresa. Adicionalmente, investir no capital humano é uma forma segura de se obter benefícios para sua organização. Qualificando o capital humano, associado a um bom programa de retenção de talentos, a organização certamente colherá os frutos disso no futuro.

Quanto ao investimento necessário para se obter bons resultados na gestão de pessoas, inicialmente pode ser alto como pôde ser visto no caso da Apdata, que investiu 2 milhões de dólares. Mas existem outras maneiras de se investir no capital humano sem que isso exija tanto dinheiro, um bom exemplo disso são algumas das ações realizadas pela equipe da CHESF.

Por fim, para reconhecer as pessoas de uma organização nem sempre é através de dinheiro, com aumento ou promoção salarial. Dinheiro não é tudo. Muitas vezes o reconhecimento por um bom trabalho é mais gratificante que o dinheiro ou premiação.

08

# GESTÃO DE RISCOS EM SOFTWARE: UM ESTUDO BASEADO EM *SURVEY* SOBRE AS PRÁTICAS NOS PROJETOS DE SOFTWARE DO ESTADO DE PERNAMBUCO

*Raquel Godoi Amaral Ferraz*

Riscos referem-se a acontecimentos futuros. Hoje e ontem já não constituem mais uma preocupação, já que se está colhendo os resultados de nossas ações. A pergunta é, mudando as ações hoje, pode-se, então, criar uma oportunidade para uma situação diferente e, conforme se espera, melhor para amanhã? O risco envolve mudanças, como de opinião, ações ou lugares, envolve escolha e a incerteza que a própria escolha traz. Paradoxalmente, o risco assim como a morte e os impostos, é uma das poucas certezas da vida [Charette, 1989].

Segundo o PMI (2013), o risco do projeto é um evento ou condição incerta que, se ocorrer, provocará um efeito positivo ou negativo em um ou mais objetivos do projeto tais como escopo, cronograma, custo e qualidade.

Conforme Pressman (2011), desenvolver software é uma empreitada difícil, muitas coisas podem acontecer fora do esperado, logo entender riscos e tomar medidas proativas para que não ocorram é fundamental para o sucesso no gerenciamento de projetos de software. Embora seja tolice querer

eliminar o risco, e é questionável tentar minimizá-lo, é essencial que os riscos assumidos sejam os certos [Drucker, 1975].

Este capítulo está inserido no contexto em que gerenciar riscos em projetos de software é uma atividade que representa importância significativa no andamento, controle e entrega dos projetos. O capítulo retrata a apuração de uma pesquisa realizada entre gestores de projetos de software no cenário de desenvolvimento de software pernambucano sobre a aplicação prática da gerência de risos.

Pernambuco representa um dos grandes centros de TI do Brasil, onde surgiu o Porto Digital, em julho de 2000, um projeto de desenvolvimento econômico que agrega investimentos públicos, iniciativa privada e universidades, compondo um sistema local de inovação que tem, atualmente, 250 instituições dos setores de TIC - Tecnologia da Informação e Comunicação e de EC - Economia Criativa.

O conjunto das empresas que fazem parte do Porto Digital faturou nos últimos três anos mais de R$ 1 bilhão. Desse montante, 65% são originados de contratos firmados fora do Estado de Pernambuco. O parque tecnológico reúne mais de 7.100 profissionais, sendo 500 deles empreendedores [Porto Digital, 2015].

O Porto Digital apresenta muitos incentivos para programas de qualidade, como o Programa de Qualidade de Software do Porto Digital. O programa faz parte do Fortalecimento do Sistema Pernambucano de Inovação em Empresas de Base Tecnológica (INOVAPE) e tem parceira com o Porto de Digital e a Financiadora de Estudos e Projetos (FINEP) e a Secretaria Estadual de Ciência e Tecnologia de Pernambuco (SECTEC). O Programa apoia a preparação das empresas embarcadas para obtenção de certificados de qualidade em desenvolvimento de software (MPS.BR e CMMI). O início deste programa ocorreu em outubro de 2010 de acordo com regulamento lançado pelo Porto Digital.

A realização da pesquisa apresentada neste capítulo foi por meio de um questionário, o qual aborda itens como realização do gerenciamento de riscos durante os projetos, etapas de gerenciamento de risco, quantidade de funcionários nas organizações, certificação de qualidade, barreiras que dificultam o gerenciamento dos riscos, riscos que acontecem com frequência e controles que são utilizados para mitigar os riscos. O questionário utilizado foi obtido através da pesquisa de Mendes e Oliveira (2014) sobre o cenário brasileiro.

Os resultados obtidos por meio da pesquisa ambicionam apresentar as dificuldades encontradas pelos gestores de software frente à gestão de riscos e propiciar a reflexão sobre projetos futuros com o objetivo de tornar cada vez mais comum o uso do processo de gerenciamento de riscos.

## 8.1. Processo de Gerenciamento de Riscos

De acordo com o PMI (2013), o gerenciamento dos riscos do projeto inclui os processos de planejamento, identificação, análise, planejamento de respostas e controle de riscos de um projeto. Os objetivos do gerenciamento dos riscos do projeto são aumentar a probabilidade e o impacto dos eventos positivos e reduzir a probabilidade e o impacto dos eventos negativos no projeto.

Reconhecer o que pode dar errado é o primeiro passo, chamado de "identificação do risco". Em seguida, o risco é analisado para determinar a probabilidade de que ocorra e o dano que causará se ocorrer. Uma vez estabelecidas essas informações, os riscos são classificados, por probabilidade e por impacto. Por fim, é desenvolvido um plano para gerenciar os riscos de alta probabilidade e alto impacto [Pressman, 2010].

A Figura 29 apresenta a visão geral dos processos do gerenciamento, conforme o PMBOK [PMI, 2013].

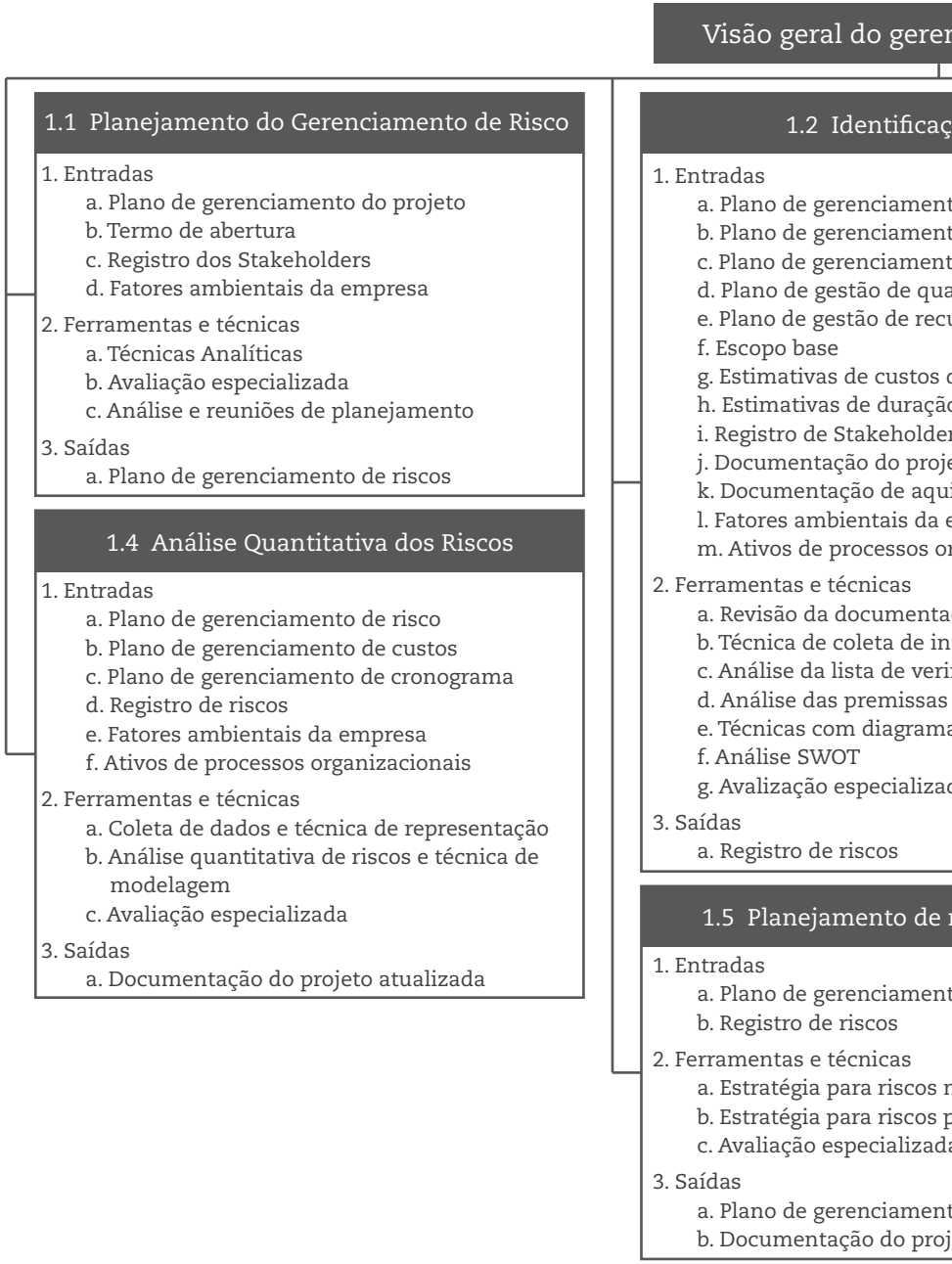



**Figura 29**. Visão geral do gerenciamento de riscos [PMI, 2013]

nciamento de riscos

ão de Riscos

to de risco
to de custos
to de cronograma
alidade
ursos humanos

das atividades
o das atividades
rs
eto
isição
empresa
rganizacionais

ção
formações
ficação

as

da

respostas aos riscos

to de risco

negativos ou ameaças
positivos ou oportunidades
a

to de projeto atualizado
eto atualizada

## 1.3 Análise Qualitativa dos Riscos

1. Entradas
    a. Plano de gerenciamento do projeto
    b. Declaração de escopo
    c. Escopo base
    d. Registro de riscos
    e. Fatores ambientais da empresa
    f. Ativos de processos organizacionais
2. Ferramentas e técnicas
    a. Avaliação do impacto e probabilidade dos riscos
    b. Matriz de impacto e probabilidade
    c. Avaliação da qualidade dos dados dos riscos
    d. Classificação dos riscos
    e. Avaliação de urgência dos riscos
    f. Avaliação especializada
3. Saídas
    a. Documentação do projeto atualizada

## 1.6 Monitoramento e controle de riscos

1. Entradas
    a. Plano de gerenciamento de risco
    b. Registro de riscos
    c. Dados do desempenho do trabalho
    d. Relatório do desempenho do trabalho
2. Ferramentas e técnicas
    a. Reavaliação dos riscos
    b. Auditoria dos riscos
    c. Variância e análise de tendências
    d. Medição do desempenho técnico
    e. Análise das reservas
    f. Reuniões
3. Saídas
    a. Informação de desempenho do trabalho
    b. Solicitação de mudança
    c. Plano de gerenciamento de projeto atualizado
    d. Documentação do projeto atualizado
    e. Avaliação dos processos organizacionais atualizado

Cada um dos processos visualizados na Figura 29 são descritos abaixo:

- Planejamento do gerenciamento do risco: trata do processo de definição de como conduzir as atividades de gerenciamento dos riscos de um projeto;
- Identificação de risco: trata do processo de determinação dos riscos que podem afetar o projeto e de documentação das suas características;
- Análise qualitativa do risco: trata do processo de priorização dos riscos para análise ou ação posterior através da avaliação e combinação de sua probabilidade de ocorrência e impacto;
- Análise quantitativa do risco: trata do processo de analisar numericamente o efeito dos riscos identificados nos objetivos gerais do projeto;
- Planejamento de respostas aos riscos: trata do processo de desenvolvimento de opções e ações para aumentar as oportunidades e reduzir as ameaças aos objetivos do projeto;
- Monitoramento e controle de riscos: trata do processo de implementar planos de respostas aos riscos, acompanhar os riscos identificados, monitorar riscos residuais, identificar novos riscos e avaliar a eficácia do processo de gerenciamento dos riscos durante todo o projeto.

Os processos apresentadas na Figura 29 realizam o caminho para que a área de gerenciamento de risco seja efetivamente realizada segundo o PMBOK. As organizações podem optar por mapear estas fases ou processos usando esta nomenclatura ou apresentar outro nome, e a realização desses passos também pode ser verificada nas organizações de forma mais sucinta. O PMBOK apresenta a forma como a gestão de risco pode ser desenvolvida.

Organizações verificam o risco como eventos que ocorre impactando os objetivos do projeto. As empresas podem aceitar ou não os riscos, por isso é importante manter um processo de riscos e elaborar sua gestão. Esta deve ser validada e estar em contínua melhoria para que estes impactos sejam conduzidos durante o projeto, sem causar grandes modificações no escopo e prazo dos projetos. Desenvolver uma abordagem que seja atrativa e consistente para o projeto ou até mesmo desenvolver um processo diferente para cada projeto da organização, verificando a estratégia adotada para a necessidade e objetivo de cada projeto, também é uma forma de conduzir a gestão de riscos.

Para alcançar sucesso nos projetos é necessário que a criação do processo na organização seja vista de forma comprometida por todos da organização e seus envolvidos. Observa-se que neste mapeamento do PMBOK os processos são apresentados com os itens de entrada, ferramenta e saída de cada etapa do processo geral do gerenciamento de risco. Assim, ao se estabelecer a gestão de risco na organização pessoas serão envolvidas nesses processos, assim o comprometimento com a gestão de riscos é obtida por meio de pessoas que seguem o que foi definido ou melhoram o que estava acordado.

De acordo com Pressman (2010), são consideradas diferentes categorias de riscos: Riscos de Projeto, que ameaçam o plano de projeto, identificam problemas potenciais de orçamento, cronograma, pessoal (equipe e organização), recursos, clientes, e requisitos e seu impacto sobre o projeto de software; Riscos técnicos, que ameaçam a qualidade e a data de entrega do software a ser produzido, identificando problemas potenciais de projeto, implementação, interface, verificação e manutenção; e Riscos de Negócio, que ameaçam a viabilidade do software a ser criado e muitas vezes ameaçam o projeto ou o produto.

Os riscos podem estar presentes em qualquer parte do projeto desde sua concepção, passando por acordos firmados entre cliente e fornecedor, desenvolvimento do plano do projeto, cronograma, requisitos, desenvolvimento, mudanças, teste, implantação, equipe e a própria organização.

## 8.2. Trabalhos Relacionados

Mendes e Oliveria (2014) realizaram uma intensa pesquisa nos repositórios IEEE Xplore, ACM Digital Library e Anais do Simpósio Brasileiro de Qualidade de Software, com a finalidade de apresentar os trabalhos mais relevantes sobre a Gestão de Risco e trazer esta comparação para o trabalho realizado no país.

Além disso, as pesquisas de Odzarly *et al*. (2009), tinham como cerne apresentar as principais barreiras que dificultavam a execução da Gestão de Riscos pelos gerente de software experientes.

Também foram abordados os trabalhos de Addison e Vallabh (2002), que realizaram pesquisa bibliográfica para identificar os principais riscos de software da literatura. Assim como Ropponen e Lyytinen (2000), que exploraram em sua pesquisa a identificação das práticas utilizadas no gerenciamento de risco. Por fim, Lobato *et al*. (2013) analisaram os principais riscos que ocorriam nos projetos de software.

## 8.3. Metodologia Utilizada na Pesquisa

Esta pesquisa teve como objetivo abordar o entendimento por meio dos gestores do estado de Pernambuco sobre a tratativa da Gestão de Risco, usando como comparativo o trabalho realizado no Brasil de Mendes e Oliveira (2014).

O questionário aplicado no trabalho de Mendes e Oliveira (2014), também foi base para a realização deste estudo. As questões enviadas aos gestores de Pernambuco estão apresentadas no Quadro 2.

Foram enviadas mais de 40 solicitações para gestores dos projetos de software de Pernambuco, onde este questionário esteve disponível de 16 a 24 de Abril de 2015, utilizando armazenamento em nuvem.

Conforme estrutura apresentada no trabalho de Mendes e Oliveira (2014), o questionário foi organizado em cinco grupos: (1) perfil do Entrevistado; (2) perfil da Organização que o Entrevistado atua; (3) Barreiras que podem dificultar o gerenciamento dos riscos; (4) principais Riscos identificados; e (5) tipos de Controles utilizados para mitigar riscos.

De acordo também com o trabalho realizado no Brasil, as questões do grupo 3 foram retiradas da pesquisa realizada por *Odzarly et al.* (2009) e as questões dos grupos 4 e 5 foram selecionadas a partir do trabalho apresentado por Addison e Vallabh (2002).

**Quadro 2.** Questões enviadas para realização da pesquisa

| Índice | Itens do Questionário |
|---|---|
| 1 | Perfil Entrevistado |
| 1.1 | Há quanto tempo trabalha com gerenciamento de projetos? |
| 1.2 | Quantos projetos você já gerenciou? |
| 1.3 | Você realiza gerenciamento dos riscos durante os projetos? |
| 1.4 | Quais etapas do gerenciamento de riscos foram realizadas nos projetos que você gerenciou? |

| 2 | Perfil da Organização que o Entrevistado Atua |
|---|---|
| 2.1 | Quantos funcionários têm na organização que você trabalha atualmente? |
| 2.2 | A organização que você trabalha possui certificação ou avaliação em qualidade de software? |
| 2.3 | Caso a organização que você trabalha possua avaliação MPS.BR, qual o nível avaliado? |
| 2.4 | Caso a organização que você trabalha possua avaliação CMMI, qual o nível avaliado? |
| 2.5 | A organização que você trabalha possui processo de Gerência de Riscos definido? |
| 2.6 | Você utiliza algum software de apoio para gerenciar riscos? |
| **3** | **Possíveis barreiras que dificultam o gerenciamento dos riscos** |
| 3.1 | Custos tangíveis e visíveis recebem mais atenção que perdas intangíveis, como perda de lucro, ou redução da produtividade |
| 3.2 | Não há recursos disponíveis |
| 3.3 | Ações de mitigação podem necessitar de mudanças organizacionais ou de processo |
| 3.4 | Gerenciar risco aparenta ser difícil, ou existem muitos riscos para lidar |
| 3.5 | Valor da gerência de riscos não é facilmente provado |
| 3.6 | Equipes e gerentes enxergam recompensas na resolução de problemas, em detrimento da prevenção |
| 3.7 | Excesso de confiança (risco sendo manipulado implicitamente) |
| 3.8 | Falar sobre riscos vai contra normas culturais |
| 3.9 | Fatalismo (softwares sempre terão problemas) |

| 4 | Principais Riscos Identificados |
|---|---|
| 4.1 | Objetivo ou escopo pouco identificados |
| 4.2 | Orçamento e cronogramas pouco realistas |
| 4.3 | Falta de um gerente experiente compromissado com o projeto |
| 4.4 | Falha ao envolver o usuário |
| 4.5 | Conhecimentos ou habilidades inadequadas |
| 4.6 | Falta de uma metodologia efetiva para a gerência do projeto |
| 4.7 | Não entendimento dos requisitos |
| 4.8 | *Gold platting* (inserção de funcionalidades que o cliente não pediu) |
| 4.9 | Mudança contínua nos requisitos |
| 4.10 | Desenvolver funções erradas no software |
| 4.11 | Subcontratação |
| 4.12 | Introdução de uma nova tecnologia |
| 4.13 | Falha ao gerenciar expectativas do usuário final |
| **5** | **Quais tipos de controles são utilizados para mitigar riscos?** |
| 5.1 | Desenvolver e manter um planejamento de software |
| 5.2 | Combinar avaliações internas com revisões externas |
| 5.3 | Gerenciar o envolvimento do usuário durante todo o ciclo de vida |
| 5.4 | Atribuir responsabilidades claras à equipe de desenvolvimento |
| 5.5 | Dividir o projeto em porções controláveis |

| | |
|---|---|
| 5.6 | Estabelecer requisitos e especificações tão cedo quanto possível |
| 5.7 | Avaliar impactos de custo e cronograma a cada mudança de requisitos |
| 5.8 | Educar os usuários sobre o impacto das mudanças de escopo durante o projeto |
| 5.9 | Assegurar que existe um comitê especificando e avaliando diretrizes |
| 5.10 | Incluir avaliação de riscos formal e periódica |
| 5.11 | Não inserir demasiadamente novas funções no projeto do software |
| 5.12 | Desenvolver planos de contingência para lidar com problemas de pessoal |

Visando o entendimento da pesquisa, Mendes e Oliveira (2014) separaram três questões chaves para compreensão do modo como os gerentes de software do Brasil verificam os riscos dos projetos e como conduzem a gestão dos mesmos.

Realizada a divisão destas questões para obter o máximo de conhecimento na pesquisa exploratória aplicada em 2014 no Brasil, esta pesquisa também teve seu foco nas três questões para assimilar os resultados e compará-los com a também pesquisa exploratória realizada no estado de Pernambuco, a saber:

- QP1: Quais os principais problemas relacionados à Gerência de Riscos?
    - » Investiga as causas mais comuns encontradas pelos entrevistados, de acordo com o trabalho realizado por Odzaly *et al*. (2009);
- QP2: Quais os principais riscos identificados durante um projeto de software?

» Verifica quais os riscos comuns aos projetos de software no cenário pernambucano, de acordo com os riscos identificados em projetos de software por Addison e Vallabh (2002);

- QP3: Quais práticas de gerenciamento devem ser prioritárias para redução de riscos?
  - » Visa identificar quais ações os projetos no cenário pernambucano fazem uso, de acordo com pesquisa realizada através do trabalho de Addison e Vallabh (2002), que várias maneiras que os projetos podem atuar para minimizar o impacto dos riscos nos projetos.

A partir do entendimento destas questões, a pesquisa abordará na próxima seção os resultados obtidos e analisará também o perfil dos entrevistados, assim como as características dos projetos que atuam ou já atuaram no cenário pernambucano.

## 8.4. Resultados da Pesquisa

Esta seção apresenta os resultados obtidos na pesquisa realizada em Pernambuco, e será dividida em subseções baseadas no Quadro 2, apresentado na Seção 8.3.

### 8.4.1. Perfil dos Entrevistados

Conforme quadro apresentado na seção da metodologia utilizado na pesquisa, foram expostas quatro perguntas a cerca do perfil do entrevistado. A Figura 30 apresenta os resultados obtidos para a primeira questão, que aborda o tempo de experiência dos entrevistados em gerenciamento de projetos de software.

Foi verificado que 46,7% dos entrevistados possuem mais de 5 anos de atuação como gestores de projetos de software, seguidos de 40% de entrevistados que possuem apenas 2

anos de experiência neste papel. E por fim, 13,3% dos entrevistados atuam como gestores de 2 e 5 anos.

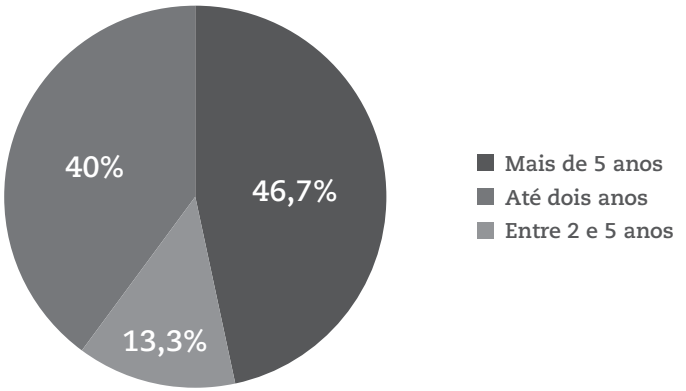



**Figura 30**. Tempo de experiência em gerenciamento de projetos de software

Para a segunda questão, para entender o perfil dos entrevistados, abordou-se a quantidade de projetos gerenciados pelos entrevistados. Conforme Figura 31, 40% dos entrevistados já geriram mais de 7 projetos, 33,3% dos entrevistados já gerenciaram até 3 projetos e 26,7% entre 4 e 7 projetos gerenciados pelos envolvidos nesta pesquisa.

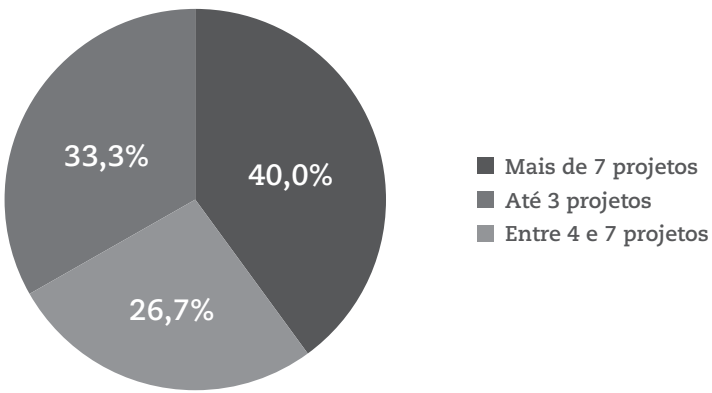



**Figura 31**. Quantidade de projetos gerenciados

A terceira questão apresenta os resultados obtidos em relação à utilização da gestão de risco durante os projetos de softwares gerenciados pelos entrevistados, como visualizado na Figura 32. Assim 93.3% dos entrevistados afirmaram que fazem uso da gestão de risco em seus projetos e apenas 6.7% não gerenciam os riscos em seus projetos de software.

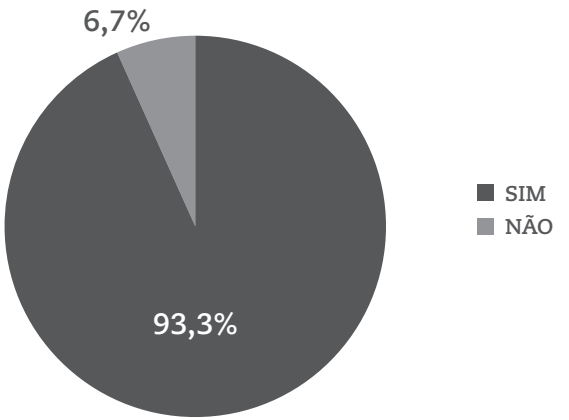



**Figura 3**2. Utilização da gestão de risco nos projetos

A última questão deste mapeamento apresenta quais etapas identificadas pelos entrevistados são utilizadas na gestão de riscos dos projetos que o gestor atua. Com o resultado de: 100% dos entrevistados para a etapa de identificação do risco; para as etapas de planejamento e análise foram obtidos os mesmos percentuais, de 92.9%; seguidos de resultados iguais também para as etapas de mitigação e monitoramento, com 85.7%; a etapa de resolução apresentou o resultado de 71.4%; a etapa de priorização com 57.1%; e, por fim, a etapa de gerência com 42.9%; como pode ser visualizado na Figura 33.

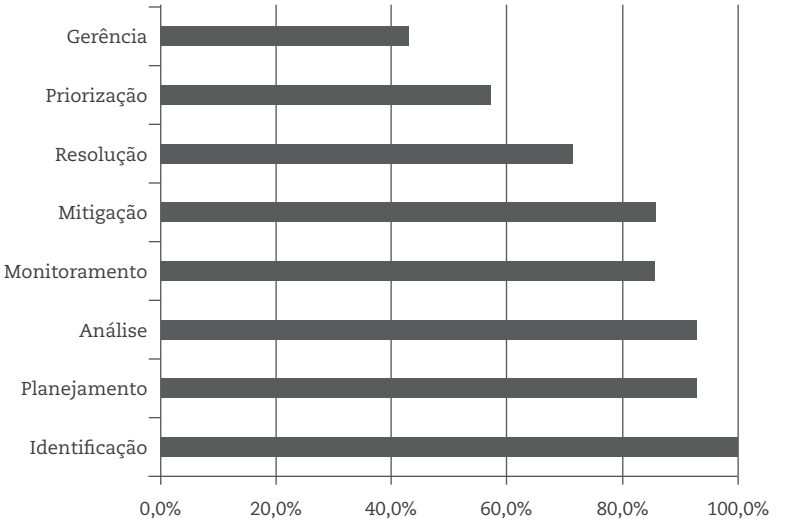



**Figura 33**. Etapa de gerenciamento adotada pelos entrevistados

### 8.4.2. Perfil da Organização que o Entrevistado Atua

Esta seção analisa a organização, na qual os entrevistados estão inseridos. Foram respondidas seis questões para este segundo mapeamento. A Figura 34 apresenta a quantidade de funcionários que a organização do entrevistado atua.

Com o percentual de 80% dos entrevistados apresentando que as organizações que trabalham atualmente apresentam mais de 50 pessoas. Apenas com duas respostas, correspondentes a 13.3% dos entrevistados, onde apresentam entre 10 e 50 pessoas, e apenas 6.7% dos entrevistados estão inseridos em uma organização com menos de 10 pessoas.

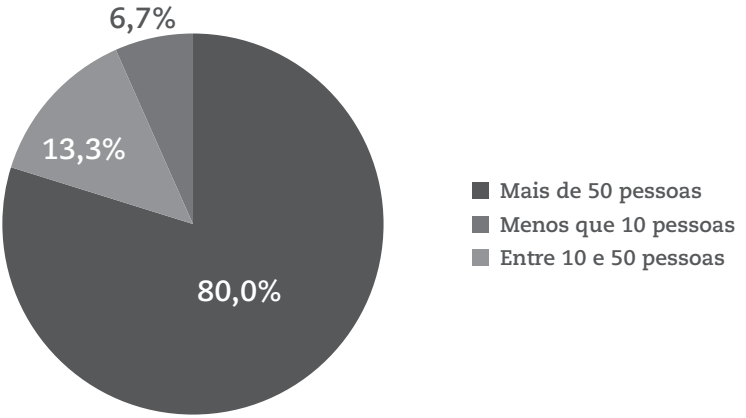



**Figura 34**. Quantidade de funcionários na organização que o entrevistado atua

A segunda questão referente ao perfil da organização que o entrevistado atua, apresenta os resultados sobre a verificação se a instituição apresenta certificação de qualidade. Conforme pode-se verificar na Figura 35, cerca de 46.7% respondeu que a organização não possui certificação, seguidos de 26.7% possuindo o CMMI, as Normas ISO e Outras certificações apresentaram o mesmo percentual de 20%, e, por fim, o MPS.BR com 6.7% das respostas obtidas.

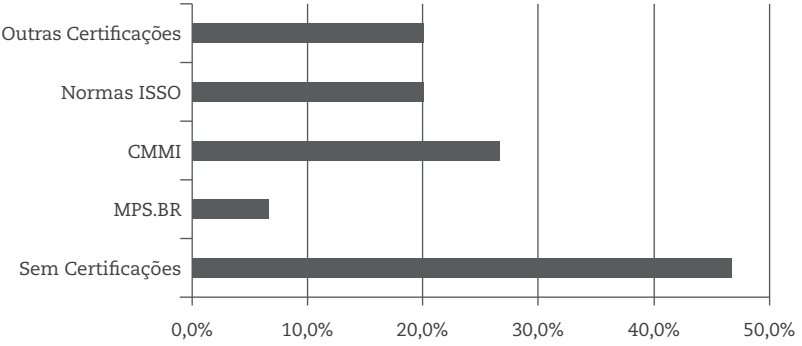



**Figura 35**. Certificação apresentada pelas organizações dos entrevistados

A terceira e a quarta questões, referentes ao perfil da organização, abordam a certificação das empresas para CMMI ou MPS. BR e qual o nível obtido. Foi verificado que apenas 4 respostas foram para o CMMI-DEV nível 3, 1 reposta para o CMMI-Serviços nível 2 e apenas 1 entrevistado apontou o MPS.BR nível F.

Sobre a quinta pergunta do mapeamento do perfil da organização, foi questionada se a instituição apresentava processos definidos de gerenciamento de risco. A Figura 36 apresenta este resultado, onde tem-se: 53.3% dos entrevistados afirmam que existe processo de gerenciamento de risco na entidade que estão inseridos; em contra partida, a um resultado de 46.7% de entrevistados que não apresentam processo de gerenciamento de risco definido na organização.

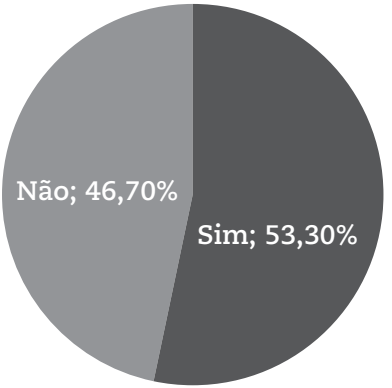



**Figura 36**. Processo de gerenciamento de risco nas organizações

Também foram analisadas se as organizações apresentavam software de apoio ao gerenciamento de risco. Na Figura 37 é possível verificar que com o percentual de 73.3% dos entrevistados não utilizam nas organizações software de apoio à gestão de risco e o restante, correspondente a 26.7%, faz uso de uma ferramenta.

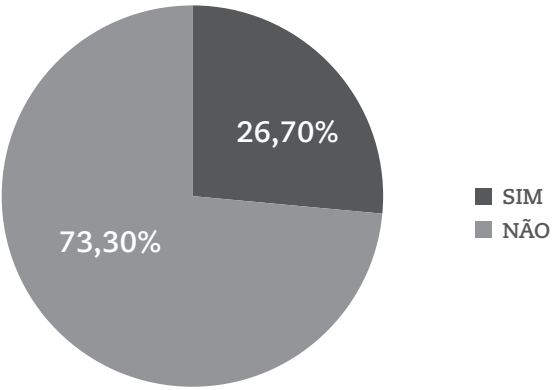



**Figura 37**. Utilização de software para apoiar a gestão de risco na organização

### 8.4.3. Possíveis barreiras que dificultam o gerenciamento dos riscos

Após traçado o mapeamento do perfil do entrevistado e da organização dos participantes, será abordada nesta subseção o início do detalhamento sobre as três questões relevantes para o entendimento e conclusão desta pesquisa.

Foi solicitado aos participantes desta pesquisa que apresentassem seu grau de concordância sobre possíveis barreiras apresentadas no questionário (ver Quadro 2). Conforme pode-se observar na Figura 38, 40% dos entrevistados escolheram o item "Excesso de confiança (risco sendo manipulado implicitamente)". Os itens "Custos tangíveis e visíveis recebem mais atenção que perdas intangíveis, como perda de lucro, ou redução da produtividade" e" Valor da gerência de riscos não é facilmente provado" apareceram empatados com 33,3%, e o terceiro item que aparece como possível barreira indicado pelos gestores de Pernambuco com 20% refere-se ao "Fatalismo (softwares sempre terão problemas)".

Também pode-se observar que os itens 3.2, 3.3 e 3.6 identificados no questionário, correspondentes aos itens "Não há recursos disponíveis", "Ações de mitigação podem necessitar de mudanças organizacionais ou de processo" e "Equipes e gerentes enxergam recompensas na resolução de problemas, em detrimento da prevenção" foram apontados com o mesmo percentual de 13,3%.

Por fim, os itens que apresentaram menor percentual foram o 3.4 e 3.8, com 6.7% dos itens identificados no questionário, "Gerenciar risco aparenta ser difícil, ou existem muitos riscos para lidar" e "Falar sobre riscos vai contra normas culturais".

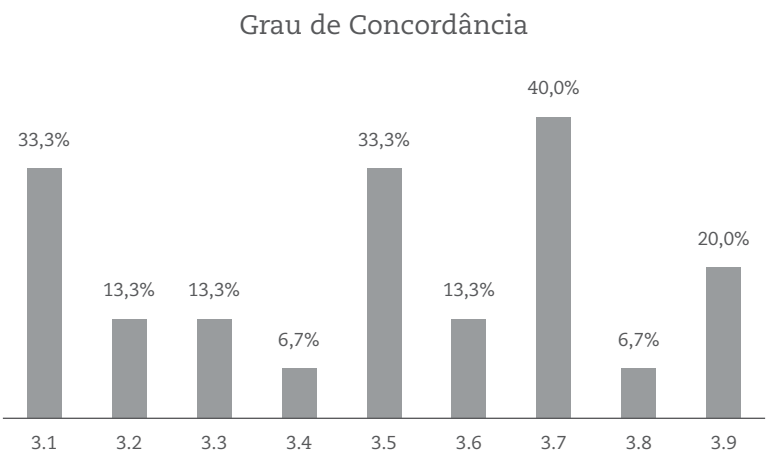



**Figura 38**. Possíveis barreiras que dificultam o gerenciamento dos riscos

### 8.4.4. Principais Riscos Identificados

Para este item, que aborda a segunda questão relevante da pesquisa, foi solicitado aos entrevistados que respondessem com que frequência os riscos aparecem em seus projetos, apresentando assim os itens como frequentemente, algumas vezes e raramente.

Pode-se observar a partir da Figura 39, o resultado obtido, dividido por item e nível de frequência nos projetos.

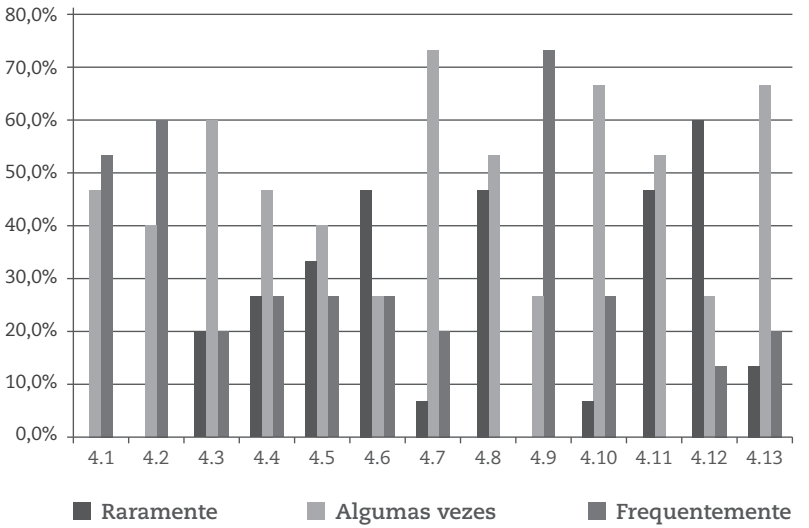



**Figura 39**. Principais Riscos Identificados

O item de maior frequência registrado com 73.3% refere-se "Mudança contínua nos requisitos", seguido com 60% do item "Orçamento e cronogramas pouco realistas" e o último item que aparece frequentemente, de acordo com os entrevistados, com 53.3%, refere-se "Objetivo ou escopo pouco identificados".

Foi observado, também, que os itens que aparecem em destaque para algumas vezes estão: em primeiro lugar, com 73.3%, o item 4.7 "Não entendimento dos requisitos"; apresentando resultados similares os itens 4.10 "Desenvolver funções erradas no software" e 4.13 "Falhas ao gerenciar expectativas do usuário final", com 67%; e com 53.3% para os itens 4.8 "Gold platting (inserção de funcionalidades que o cliente não pediu)" e 4.11 "Subcontratações" desta pesquisa.

Curiosamente os itens 4.1 “Objetivo ou escopo pouco identificados”, 4.2 “Orçamento e cronogramas pouco realistas” e 4.9 “Mudança contínua nos requisitos”, apresentam 0% como itens raramente identificados.

### 8.4.5. Quais tipos de controle são utilizados para mitigar riscos?

Por fim, tem-se a apresentação dos dados encontrados para a terceira questão relevante deste trabalho, a qual aborda o que é realizado para mitigar os riscos encontrados nos projetos de software dos entrevistados no cenário pernambucano.

Claramente pode-se identificar na Figura 40 que concidentemente os itens 5.3 “Gerenciar o envolvimento do usuário durante todo o ciclo de vida”, 5.4 “Atribuir responsabilidades claras à equipe de desenvolvimento” e 5.8 “Educar os usuários sobre o impacto das mudanças de escopo durante o projeto” apresentam percentual de 60%, vistos pelos entrevistados como os mais frequentes. O item 5.7 “Avaliar impactos de custo e cronograma a cada mudança de requisitos” apresentou 53.30%, e seguidos de 46.70% para o item 5.5 “Dividir o projeto em porções controláveis”.

Os itens que apareceram como utilizados algumas vezes para mitigar os riscos com percentual equivalente foram 5.2 “Combinar avaliações internas com revisões externas” e o 5.6 “Estabelecer requisitos e especificações tão cedo quanto possível”, com um total de 53.3%.

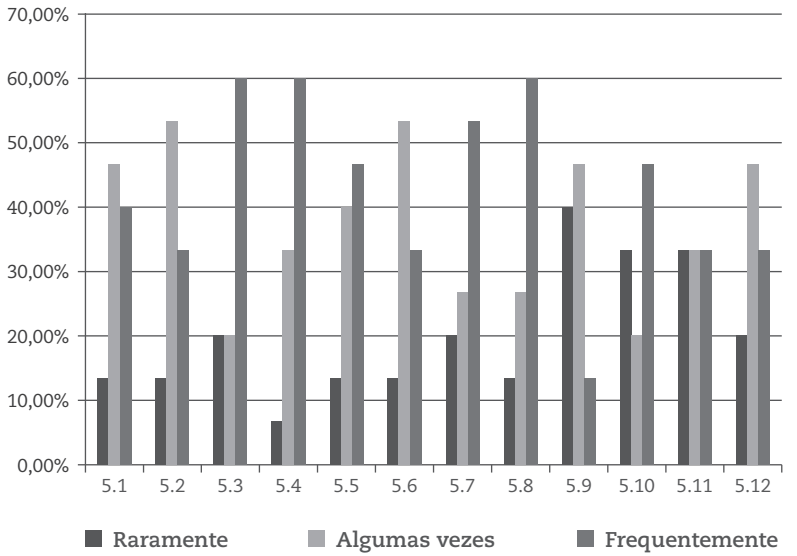



**Figura 40**. Quais tipos de controle são utilizados para mitigar riscos

## 8.5. Análise dos Resultados

Conforme mencionado neste trabalho, foi utilizado como base desta pesquisa o trabalho publicado por Mendes e Oliveira (2014), sobre os resultados obtidos de questionário aplicado a gestores de projetos de software do Brasil.

Com o intuito de realizar uma reflexão e análise sobre a atuação dos gestores dos projetos de software do cenário pernambucano para o gerenciamento de riscos, realizou-se o cruzamento dos itens obtidos a partir das três questões relevantes mapeadas pelos autores Mendes e Oliveira (2014), com os valores obtidos e relatados na seção 8.4 deste capítulo.

Em Pernambuco foi possível identificar que os principais problemas relacionados à Gerência de Riscos foram "Excesso de confiança (risco sendo manipulado implicitamente)",

“Custos tangíveis e visíveis recebem mais atenção que perdas intangíveis, como perda de lucro, ou redução da produtividade” e “Valor da gerência de riscos não é facilmente provado”.

Se for verificar os itens identificados como maior grau de concordância no Brasil, vê-se que existiram dois pontos convergentes que foram “Custos tangíveis e visíveis recebem mais atenção que perdas intangíveis, como perda de lucro, ou redução da produtividade” e “Valor da gerência de riscos não é facilmente provado”. Porém, a primeira posição na pesquisa do Brasil ficou com Custos tangíveis e visíveis.

Apenas dois itens nesta primeira análise foram divergentes, que apareceram na pesquisa de Pernambuco, mas não foi opção no Brasil, o item “Ações de mitigação podem necessitar de mudanças organizacionais ou de processo” e o primeiro item “Excesso de confiança (risco sendo manipulado implicitamente)” também não apareceu na pesquisa realizada no Brasil. Assim, o Quadro 3 apresenta um comparativo entre os dois cenários das pesquisas.

**Quadro 3.** Comparativo dos possíveis barreiras que dificultam o gerenciamento dos riscos

| Brasil | Pernambuco |
|---|---|
| Possíveis barreiras que dificultam o gerenciamento dos riscos | |
| Custos negativos e visíveis recebem mais atenção que perdas intangíveis, como perda de lucro, ou redução da produtividade | Excesso de confiança (risco sendo manipulado implicitamente) |
| Açoes de mitigação podem necessitar de mudanças organizacionais ou de processo | Custos tangíveis e visíveis recebem mais atenção que perdas intangíveis, como perda de lucro, ou redução da produtividade |
| Valor da gerência de riscos não é facilmente provado | Valor da gerência de riscos não é facilmente provado |

Para a segunda questão analisada neste trabalho, os resultados obtidos em Pernambuco sobre os principais riscos identificados foram “Mudança contínua nos requisitos”, “Orçamento e cronogramas pouco realistas” e “Objetivo ou escopo pouco identificados”, em ordem de frequência obtida na pesquisa.

Para o Brasil, concidentemente, o primeiro item apontado também foi “Mudança contínua nos requisitos”, o que mostra claramente que é um problema percebido em ambos os cenários. Outro item que se igualou é o “Objetivo ou escopo pouco identificado”, conforme pode-se observar no Quadro 4.

**Quadro 4.** Comparativo de riscos identificados pelos cenários analisados

| Brasil | Pernambuco |
|---|---|
| Principais Riscos Identificados | |
| Mudança continua nos requisitos | Mudança continua nos requisitos |
| Não entendedimento dos requisitos | Orçamento e cronogramas pouco realistas |
| Falha ao gerenciar expectativas do usuário final | Objetivo ou escopo pouco identificados |
| Objetivo ou escopo pouco identificados | |

Por fim, ao se verificar a terceira questão deste trabalho, sobre os controles utilizados para mitigar os riscos utilizados, foi percebido que “Gerenciar o envolvimento do usuário durante todo o ciclo de vida”, “Atribuir responsabilidades claras à equipe de desenvolvimento” e “Dividir o projeto em porções controláveis” também apareceram na pesquisa realizada no Brasil, porém em ordem diferente como pode-se observar no Quadro 5. Encontrou-se dois itens que não foram iguais ao resultado do cenário brasileiro, sendo os itens “Educar os

usuários sobre o impacto das mudanças de escopo durante o projeto" e "Avaliar impactos de custo e cronograma a cada mudança de requisitos", considerados pelos gestores de projetos de software do estado de Pernambuco.

**Quadro 5.** Comparativo de controle efetuado para mitigar os riscos

| Brasil | Pernambuco |
|---|---|
| Quais tipos de são utilizados para mitigar riscos? | |
| Atribuir responsabilidades claras à equipe de desenvolvimento | Gerenciar o envolvimento do usuário durante todo o ciclo de vida |
| Desenvolver e manter um planejamento de software | Atribuir responsabilidades claras à equipe de desenvolvimento |
| Dividir o projeto em porções controláveis | Educar os usuários sobre o impacto das mudanças de escopo durante o projeto |
| Estabelecer requisitos e especificações tão cedo quanto possível | Avaliar impactos de custo e cronograma a cada mudança de requisitos |
| Gerenciar o envolvimento do usuário durante todo o ciclo de vida | Dividir o projeto em porções contoláveis |

Observa-se, então, que o cenário pernambucano para o atendimento à gestão de riscos, dificuldades, identificação e controle não está distante do percebido no restante do cenário brasileiro.

Contudo, os pontos encontrados apenas nesta pesquisa representam características adotadas e percebidas pelos gestores do estado analisado. Assim, o intuito deste trabalho é realizar uma consideração de que o cenário de software encontrado nestes dois ambientes apresentam pontos de convergência, porém traz-se uma reflexão sobre a cultura encontrada a partir das respostas divergentes obtidas por este questionário que se destacaram conforme verificado.

A partir desta rastreabilidade, advinda de trabalhos internacionais e da pesquisa realizada no Brasil em 2014, é possível atuar e refletir sobre o uso da gestão de risco em projetos de software de forma mais atrativa por meio do entendimento que esta pesquisa proporcionou.

## 8.6. Considerações Finais

A gerência de risco adiciona à gerência de projetos uma abordagem estruturada para identificação e análise de riscos no início do planejamento do projeto e no decorrer das fases de desenvolvimento de software. O planejamento de respostas aos riscos cria a perspectiva de se obter alternativas e planos de contingências para se mitigar os riscos, enquanto as funções de monitoração e controle dos processos de gerência de risco combinam-se com a função de controle da gerência de projetos.

Desta forma, percebe-se que a gerência de risco é fator de grande importância e utilidade no alcance da maturidade organizacional. As atividades de identificação, análise e monitoração de riscos devem ser realizadas de forma sistematizadas e controladas, durante todo o processo. Salienta-se a necessidade de avaliações constantes e contínuas dos fatores e eventos associados a cada risco priorizado, como uma forma de implementar a melhoria e garantir a qualidade do processo de desenvolvimento de software.

09

# ITIL - GESTÃO DE SERVIÇOS DE TECNOLOGIA DA INFORMAÇÃO

*Kamila Nayana Carvalho Serafim*

A Tecnologia da Informação (TI) pode ser definida com várias expressões, mas sempre terão o mesmo objetivo, que é a melhor utilização dos métodos de tratamento para a informação. Os recursos utilizados agregam valor e sentido às atividades, pois as ferramentas de TI devem ser utilizadas de forma correta para que tragam bons resultados e novas soluções, considerando sempre menor custo beneficio. Em uma empresa a utilização da tecnologia da informação pode se expandir em vários campos em diferentes setores, devido ao termo descrito como governança corporativa, ou seja, é o sistema pelo qual as empresas são dirigidas e controladas.

O perfil de uma empresa é atribuída na forma que a gestão controla suas áreas, ou seja, uma empresa pode possuir várias células gerenciáveis e em setores diversos. Com o estudo aplicado ao controle interno e externo de uma empresa, há possibilidades de organização e controle de tarefas e processos. Metodologias foram criadas e são aplicadas para a gestão organizacional e gestão operacional. A área mais comum

abrangida nas empresas é a governança de TI, que lidera os processos tecnológicos da empresa, de forma que qualquer assunto referente a processos, indicadores de ativos e ou metodologias deve ser implementada para melhor rendimento dos processos e evolução da empresa.

No campo de TI, há vários outros aspectos que devem ser considerados, por exemplo: segurança, disponibilidade, utilização de sistemas adequados, tecnologias (qual é a melhor para determinada finalidade), legislação local e assim por diante.

Para aprimorar a governança de TI, existem diversas metodologias e *frameworks* que podem auxiliam no controle de uma empresa, por exemplo: CobiT, BSC, ITIL, CMM/CMMI e PMBOK, entre outros. Assim, este capítulo está relacionado à utilização da ITIL, no alinhamento estratégico baseado na gestão de TI.

A ITIL foi desenvolvida pela CCTA - *Central Computer and Telecommunications Agency* (hoje OGC - *Office for Government Commerce*) na Inglaterra nos anos 80 [The Cabinet Office, 2011]. Surgiu da necessidade do governo britânico de organizar os processos da área de TI, que não estava satisfeito com a qualidade dos serviços de TI prestados a ele. Assim, foi solicitada a construção de uma abordagem de melhores práticas para gerenciar a utilização eficiente e eficaz dos recursos de TI, aplicável a organizações com necessidades técnicas e de negócio

ITIL tem ganhado cada vez mais confiança das empresas em todo mundo por causa da sua abrangência. O modelo é composto das melhores práticas usadas para o gerenciamento de serviços de alta qualidade de TI, conseguidas em acordo após longo tempo de prática e observação, trabalho e pesquisa de profissionais de tecnologia de informação e processamento de dados no mundo inteiro.

Houve muitas publicações da ITIL [Bon, 2012]: a primeira em 1989 denominada v1, com 40 livros; a v2 que era a segunda versão, publicada em 2000 com 7 livros; e a versão 3 da ITIL chamada V3, foi divulgada em maio de 2007, representou uma grande mudança em relação à versão anterior, por usar os processos de gerenciamento de serviços organizados em uma estrutura de ciclo de vida de serviço. Além disso, a ITIL v3 mostrava a maturidade que a matéria de gerenciamento de serviços de TI adquiriu com o passar do tempo, abordando conceitos como portfólios dinâmicos de serviços, integração da TI ao negócio, mensuração do valor do negócio. A partir da versão 3, havia também o fornecimento de uma base robusta para a convergência com outros padrões e modelos de gestão e governança, tais como CMMI, CobiT, PMBOK, IOS/IEC, Escm-SP, etc.

Em julho de 2011, foi publicado pelo TSO - *The Stationery Office*, a editora dos volumes da ITIL, vinculada ao *The Cabinet Office*, uma outra atualização da ITIL v3 denominada de "ITIL 2011", composta por mudanças não muito impactantes, objetivando: a correção de algumas inconsistências e erros identificadas nas figuras, no texto e nos relacionamentos entre os cinco livros; foi revisto o livro de Estratégia de Serviço e o Glossário de Termos para tornar a explicação de alguns conceitos mais concisos, acessíveis e claros; e foram incluídas sugestões de resoluções e melhoria e de problemas observados pelos usuários, fornecedores e instrutores, no sentido de aumentar a consistência  e a clareza, a navegação pelo conteúdo, a abrangência e precisão. Atualmente, a propriedade e a comercialização deste portfólio de melhores práticas pertence a AXELOS.

## 9.1. Serviço

Um serviço é um meio de entregar valor aos clientes, facilitando os resultados que os clientes querem alcançar, trazendo como resultado o aumento da probabilidade do cliente obter os objetivos esperados sem o cliente não assumir os custos e riscos inerentes ao serviço.

A norma ISO/IEC 20000 [ISO/IEC, 2011] fornece um padrão de referência comum para qualquer empresa oferecer serviços de TI para clientes internos ou externos. Esta norma provê a adoção de uma abordagem de processos integrada para a gestão de serviços de TI e alinha-se com as melhores práticas da ITIL para entrega e suporte de serviços.

A ISO/IEC 20000 consiste em cinco partes sob o título geral "Tecnologia da Informação", a saber [ISO/IEC, 2011]:

- ISO/IEC 20000-1: especifica ao provedor de serviços os requisitos para planejar, estabelecer, implementar, operar, monitorar, revisar, manter e melhorar o GSTI - Gerenciamento de Serviços de TI. Os requisitos incluem o projeto, transição, entrega e melhoria dos serviços para atender aos requisitos previamente acordados;
- ISO/IEC 20000-2: representa um consenso do setor sobre padrões de qualidade em processos de GSTI e descreve as melhores práticas para esses processos;
- ISO/IEC TR 20000-3: fornece orientações, explicações e recomendações para a definição do escopo, aplicabilidade e demonstração da conformidade com a ISO/IEC 20000-1 pelo uso de exemplos práticos;
- ISO/IEC 20000-4: tem como objetivo de facilitar o desenvolvimento de um modelo para avaliação de processo de acordo com a norma ISO/IEC 15504. O modelo de referência de processo, previsto nesta norma, é uma representação lógica dos elementos dos processos para o

gerenciamento de serviços que podem ser executados em um nível básico. Cada processo é descrito em termos de um propósito e resultados associados;

- ISO/IEC 20000-5: apresenta um exemplo de plano de implementação no qual são fornecidos guias para os provedores de serviços atenderem aos requisitos da ISO/IEC 20000-1. Também inclui orientações para iniciar o projeto e uma lista de atividades principais para atender cada fase da implementação da ISO/IEC 20000-1.

Na ITIL todos os processos tem que agregar valor para o cliente, para gerar mais interesse para o cliente. Um exemplo de como se vê um serviço é ao estar no aeroporto e pegar um avião, o passageiro não se importa onde o avião vai parar, se a pista estará com frisos ou não, se o radar é da tecnologia A, B ou C, qual a velocidade da esteira, tudo o que se deseja é ser bem atendido e fazer uma viagem segura do ponto de partida ao ponto de destino final.

### 9.1.1. Gerenciamento de Serviço

O gerenciamento do conjunto de habilidades específicas da organização que provêm valor para os clientes na forma de serviços, e a capacidade de transformar recursos em serviços com valor agregado são os principais objetivos do gerenciamento de serviço. Práticas profissionais suportadas por um extensivo corpo de conhecimento, experiências e habilidades.

O gerenciamento de Serviço toma a forma de um conjunto de funções e processos para gerenciar os serviços durante o seu ciclo de vida, ou seja, a prática de acompanhar o serviço desde a concepção da ideia até a sua morte é considerado como Gerenciamento de Serviços.

## 9.2. Visão Geral da ITIL

A ITIL pode ser vista como uma fonte de boas práticas utilizadas pelas organizações para estabelecer e melhorar suas capacitações em gerenciamento de serviços [Bon, 2012]. O núcleo da ITIL é composto por cinco livros (visualizado na Figura 41), cada uma ligada a um estágio do ciclo de vida do serviço, contendo indicações para uma abordagem integrada de gerenciamento de serviços:

- Estratégia de Serviço: serve de base para as outras fases posteriores, mostrando como entender o gerenciamento de serviços como um ativo estratégico descrevendo os princípios da ideia do negócio. A prática desta disciplina é útil para criar políticas, diretrizes e processos de gerenciamento de serviços, não somente como uma capacidade organizacional, e sim como um ativo estratégico;
- Desenho de Serviço: realiza a estratégia anterior, retratando o desenho e desenvolvimento dos serviços, e as práticas de gerenciamento de serviços, desenha sistemas e ferramentas de gerenciamento, para que sejam capazes de apoiar os serviços em todos os momentos do ciclo de vida. Também aborda sobre melhorias necessárias para manter ou agregar valor aos clientes ao longo do ciclo de vida de serviço e mudanças;
- Transição de Serviço: trata sobre como executar a transição de serviços novos e modificados para o ambiente do cliente, a implementação de todas as mudanças no Portfólio ou Catálogo de Serviços através do processo de transição, e aborda também o gerenciamento dos relacionamentos com todas as partes interessadas (*stakeholders*) nos serviços;
- Operação de Serviço: relata a fase do ciclo de vida do gerenciamento de serviços, que é responsável pelas tarefas do

cotidiano, detalhando os processos de gerenciamento de incidentes, eventos, acesso, problemas e de cumprimento de requisições. Também orienta sobre como garantir a entrega e o suporte a serviços de forma eficaz e eficiente em ambientes operacionais controlados por gerenciamento;

- Melhoria Contínua de Serviço: seu escopo contém atividades que suportam o planejamento contínuo da melhoria de processos através de princípios, práticas e métodos de gerenciamento da qualidade. Orienta sobre como realizar melhorias incrementais e de larga escala na qualidade de serviço, nas metas de eficiência operacional, na continuidade do serviço, etc.

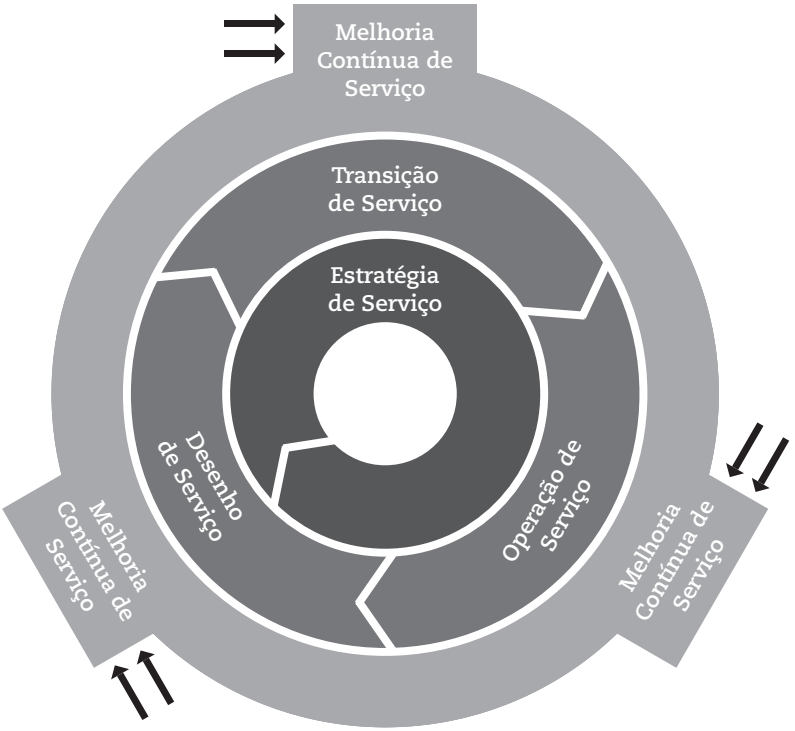



**Figura 41**. O núcleo da ITIL, adaptado de [The Cabinet Office, 2011]

A aquisição das práticas da ITIL pretende influenciar uma organização a um grau de maturidade e qualidade que permita o uso eficiente e eficaz dos seus ativos estratégicos de TI, sempre focando na integração com as necessidades de usuários e clientes. O principal objetivo da ITIL é fazer com que o Mercado teste e comprove o conjunto de práticas de gerenciamento de serviços de TI, que são organizadas usando o modelo de ciclo de vida de serviços. Essas práticas podem servir como apoio, tanto para organizações que já possuem operações de TI em andamento e pretendem empreender melhorias, quanto para criação de novas operações [Bon, 2012].

Um outro objetivo importante sobre a ITIL V3, é mensurar e gerenciar o valor que os serviços de TI efetivamente adicionam ao negócio, através do uso do modelo de ciclo de vida, fazendo com que seja possível ter uma visão do gerenciamento de serviços pela perspectiva do próprio serviço, em vez de focar em um processo ou prática de cada vez.

### 9.2.1 Aplicabilidade da ITIL

O modelo da ITL tem sido usado frequentemente em projetos e operações contínuas envolvendo itens de infraestrutura, como gerenciamento de redes, manutenção de equipamentos, suporte à utilização de aplicações e *outsourcing* de processos de impressão, entre outros. As práticas desse modelo são compatíveis com várias modalidades de prestação de serviços de TI, que necessitem de uma forte gestão de serviços [Bon, 2012].

A ITIL pode ser aplicada a serviços específicos de gerenciamento de serviços relacionados a aplicações, tais como manutenções, operações de fábrica de software, *outsourcing* de desenvolvimento, etc. A ITIL v3 tornou o modelo mais possibilitado para aplicação nas organizações, de empresas de

pequeno até grande porte, das empresas que estão iniciando seus passos até as empresas com alto grau de maturidade em seus processos.

O ciclo de vida de serviços pode ser usado a partir da Estratégia do Serviço. Este caso é mais indicado para quando uma organização está gerando novos serviços para oferecer ao seu Mercado. Para organizações que já estão em operação por muito tempo e possuem serviços consolidados, parece lógico iniciar o ciclo de vida pelas atividades da Melhoria Contínua de Serviço, pois o sucesso das suas iniciativas de TI depende muito de uma gestão efetiva do seu Portfólio de TI, que coordene os investimentos e os esforços entre seus projetos, ativos, processos e serviços de TI.

Segundo Bon (2012), recomenda-se que a implementação do modelo seja feita de forma gradativa, independente de qualquer que seja o ponto de entrada, partindo de um escopo reduzido de operações como piloto e promovendo artefatos sucessivos para as demais operações, respeitando sempre as interdependências existentes entre requisitos de disponibilidade e continuidade dos serviços e os processos de gestão. As questões relacionadas à tecnologia que suportam os serviços e à estrutura organizacional, deve ter uma atenção especial de forma que os seus pontos fortes sejam totalmente aproveitados, e que as mudanças eventuais possam ser realizadas com impacto mínimo na disponibilidade e na continuidade do negócio. Pode ser que a ITIL precise de adaptações em função das características de cada organização de TI.

Pode-se listar os seguintes benefícios da ITIL [Bon, 2012]:

- Serviços de TI entregues com maior qualidade;
- Alinhamento da área de TI com o Negócio;
- Cultura de melhoria contínua e gestão do conhecimento;
- Redução da dependência de pessoas chave;

- Transição da prestação de serviços de reativa para proativa;
- Aumento da produtividade de funcionários;
- Melhoria do Gerenciamento de Serviços e de terceiros e fornecedores;
- Gerenciamento dos níveis de serviço entregue aos clientes;
- Melhoria na satisfação global dos clientes, possibilitando ainda um potencial aumento de clientes e de receitas.

As seções a seguir detalham cada um dos cinco livros constantes na ITIL.

## 9.3. Estratégia do Serviço

O conhecimento da Estratégia de Serviços ajuda a definir quais serviços deve-se oferecer a quem, como se pode diferenciar da concorrência, como agregar valores ao produto, como atingir a qualidade, enfim, como e onde priorizar os investimentos [The Cabinet Office, 2011].

A ITIL considera que são necessários oito passos para que um serviço possa ser concebido e definido em seus aspectos relevantes [Fernandes e Abreu, 2014]:

- Definir o mercado e identificar os clientes: quem são os clientes que têm interesse e condições de comprar o serviço, e para os quais o provedor está preparado legalmente e em termos de infraestrutura logística para entregá-lo?
- Compreender os clientes em termos dos resultados de negócio esperados, das suas preferências, percepções, assim como de seus ativos e suas restrições;
- Quantificar os resultados de forma clara e mensurável;

- Classificar e visualizar o serviço, buscando identificar "arquétipos" de serviços e ativos de clientes que possam revelar padrões de demandas e de comportamento para a prestação dos serviços;
- Compreender as oportunidades (espaços de mercado) que podem existir nas "entrelinhas" da prestação desse serviço;
- Definir os serviços com base nos resultados esperados em termos de utilidade e garantia;
- Criar modelos para os serviços que descrevam a estrutura e a dinâmica dos serviços e que possam ser utilizados como *templates* para futuros serviços;
- Definir unidades e pacotes para os serviços que permitam maior flexibilidade para a combinação das funcionalidades dos serviços.

### 9.3.1. Os processos da Estratégia de Serviço

Na estratégia do serviço existem cinco processos de gerenciamento de serviços:

- Gerenciamento Estratégico para Serviços de TI: oferece um processo e mecanismos que permitam uma organização estudar e decidir sobre quais os serviços a fornecer, com base em uma análise do potencial de retorno e nível de risco aceitável. A responsabilidade deste processo é definir e manter a posição, a perspectiva, dos padrões e planos de uma empresa tratando-se dos serviços e ao gerenciamento de tais serviços;
- Gerenciamento do Portfólio de Serviços: objetiva organizar e gerenciar os investimentos no gerenciamento de serviços e fazer com que este gere valor para a organização. Há neste processo os serviços de negócio, definidos pelo próprio negócio, e os serviços de TI, que

são fornecidos pela TI ao negócio. Esses dois processos podem ser gerenciados separadamente, mas sempre levando em consideração as relações existentes. Este processo tem como fases a análise da visibilidade das iniciativas de serviços, a aprovação dessas iniciativas, a definição do inventário dos serviços, e a abertura do projeto de desenho ou redesenho dos serviços objetivando a sua incorporação ao Catálogo de Serviços;

- Gerenciamento Financeiro: provê a manutenção econômica necessária para a execução dos seus serviços. Visa gerenciar o ciclo financeiro do Portfólio de Serviços de TI de uma organização, de forma a prover a sustentação econômica necessária para a execução dos seus serviços. Neste processo é definido o orçamento de tudo que será gasto no decorrer do projeto, é feita a contabilização por pessoas ou softwares de todos os custos e ganhos e a cobrança, que será escolhida a forma de cobrar o cliente pelo serviço podendo ser mensalmente ou anualmente;
- Gerenciamento da Demanda: objetiva organizar de forma simultânea os ciclos de produção dos serviços (que consomem demanda) e os ciclos de consumo dos serviços (que geram demanda). A análise da dinâmica do negócio poderá permitir o estabelecimento de padrões de atividades de negócio e de perfis de usuários, que podem ser combinados para a composição de pacotes de serviços customizados;
- Gerenciamento do Relacionamento com o Negócio: objetiva permitir um entendimento conciso da necessidade do cliente, fornecendo os serviços que estejam de acordo com a suas expectativas. Existe a preocupação com a medição da satisfação dos clientes, com a escrita de requisitos de negócio para serviços novos ou modificados e com qualquer aspecto que possam influenciar o relacionamento existente.

### 9.3.2. A implantação da Estratégia no ciclo de vida do serviço

A Estratégia de Serviço representa a fonte de requisitos para todas as demais fases que venham a ser implantadas. Através das suas medições e avaliações, a melhoria contínua do serviço fornece a realimentação para todas as demais disciplinas. A estratégia, deve ser desenvolvida na área delimitada pelas limitações determinadas pelo negócio, como garantia, utilidade, capacidade máxima, questões de licenciamento, preço, padrões e regulações, valores morais e éticos, recursos, etc. Aspectos tecnológicos são também importantes no desenho da estratégia, notadamente as necessidades de automação dos serviços, as interfaces entre os serviços habilitadas ou não por TI e as ferramentas a serem utilizadas.

### 9.3.3. Benefícios da Estratégia de Serviço

- Direciona a Organização de TI ao alcance dos resultados adequados aos negócios, definindo estratégias e políticas, objetivando orientar as fases posteriores;
- Prepara a Organização de TI para influenciar as percepções do Cliente, demonstrar valor, e responder às suas preferências;
- Colabora com as necessidades do Negócio para enfrentar a concorrência no mercado em mudança constante.

## 9.4. Desenho do Serviço

O propósito dessa fase é de realizar a estratégia concebida anteriormente, focando no negócio da empresa, baseado nele e na demanda que se elabora toda a arquitetura do novo serviço [The Cabinet Office, 2011]. Toda a ação de projeto de um serviço

é agregado em um único pacote SDP - *Service Design Package*) que, em conjunto com outros serviços, constituem o Catálogo ou Portfólio de Produtos. Desenha métricas e métodos para medição da qualidade do processo de desenho do serviço, em termos do seu progresso, conformidade (com requisitos corporativos, de governança, regulação), eficácia e eficiência.

Há cinco aspectos importantes que devem ser considerados nesta fase [Fernandes e Abreu, 2014]:

- Desenho de um serviço novo ou a alteração de um serviço que já existe deve ser visto como o projeto de uma solução completa, com alto grau de adesão aos requisitos estabelecidos pelo negócio. Devem estar acordados com a estratégia estabelecida pela empresa tais requisitos, como também os recursos e capacitações necessárias para o serviço;
- Desenhar as ferramentas de sistemas de gerenciamento, para que sejam capazes de amparar os serviços em todas as fases do ciclo de vida;
- Desenhar as arquiteturas de gestão e tecnológicas, para que tenham as capacitações necessárias para operar os serviços de forma concisas;
- Desenhar os processos de Gerenciamento de Serviços e TI, também como papéis, habilidades relacionados e responsabilidades, para que sejam capazes de operar, manter e apoiar os serviços, assim como criar ferramentas que permitam a integração entre organizações;
- Desenhar métricas e métodos para medição da qualidade do processo de desenho do serviço, contemplando seu progresso, conformidade, eficácia e eficiência.

Caso haja envolvimento de outros fornecedores, é importante avaliar soluções alternativas e providenciar a obtenção da solução escolhida. O projeto (ou programa) de desenvolvimento

da solução de serviço deve considerar atividades de criação, ajuste ou reutilização dos componentes do serviço.

### 9.4.1. Os processos do Desenho de Serviço

O estágio de Desenho de Serviço é suportado por um conjunto de oito processos de Gerenciamento de Serviços, descritos a seguir:

- Coordenação do Desenho: garante que os objetivos e as metas do estágio de Desenho de Serviço sejam atingidos, fornecendo e mantendo um ponto único de coordenação e controle de todos os demais processos deste estágio do ciclo de vida de serviço. Na prática pode ser gerenciado através de ferramentas de gerenciamento de atividades e ter o *status* das atividades acompanhados para serem controlados por algum gerente;
- Gerenciamento do Catálogo de Serviços: garante que exista apenas uma única fonte de informações confiáveis, consistentes e que esteja sendo constantemente atualizada sobre todos os serviços que estão ocorrendo operacionalmente e sobre os que estão sendo preparados para entrar em operação futuramente;
- Gerenciamento do Nível de Serviço: objetiva melhorar e manter a qualidade dos serviços de TI usando um ciclo contínuo de atividades, envolvendo a coordenação, colaboração, o planejamento, estabelecimento de acordos e metas de responsabilidades mútuas e desempenho, divulgações de níveis de serviços e desempenho, de níveis operacionais e de contratos de apoio com fornecedores de serviços externos. Nesse ciclo é desenvolvido também um programa com priorização de ações de melhoria para os serviços. Pode ser monitorado através de um acordo de nível de serviço (SLA - *Service Level Agreement*) que tem como objetivo garantir que os níveis

de serviços acordados com os clientes sejam entregues no presente e no futuro. O SLA é um acordo formal entre a organização de TI e seus clientes, definindo objetivos do serviço e as responsabilidades de ambas as partes;

- Gerenciamento da Capacidade: balanceia a oferta de serviços relacionada à demanda e melhorando a infraestrutura necessária à prestação dos serviços de TI. É a oportunidade de estudar a necessidade de infraestrutura que o serviço do cliente necessitará e disponibilizar essa infraestrutura, como servidores, capacidades de banco de dados, capacidade de armazenamento nas nuvens, etc.;
- Gerenciamento da Disponibilidade: objetiva manter os níveis de confiabilidade e disponibilidade que o negócio requer, minimizando os riscos de interrupção através de atividades de monitoramento físico, melhoria contínua da infraestrutura e da organização de suporte e solução de incidentes. O fornecedor do serviço pode atender esse processo tendo um plano de mitigação para o risco do serviço parar de funcionar como contrato com dois fornecedores de internet, uma equipe multifuncional, etc.;
- Gerenciamento da Continuidade: assegura que todos os serviços de TI e recursos técnicos necessários sejam recuperados dentro de um limite de tempo;
- Gerenciamento da Segurança da Informação: este processo alinha a segurança do negócio com segurança da TI, e assegura que a segurança da informação seja gerenciada efetivamente durante todo o ciclo de vida dos serviços. Na prática, o fornecedor necessitará de uma equipe especialista no serviço para prestar o serviço da melhor forma, como equipes de testes focados na segurança do serviço;
- Gerenciamento de Fornecedores: gerencia fornecedores e os contratos necessários para suportar os serviços

por eles prestados, visando prover um serviço de TI com qualidade transparente para o negócio, assegurando o valor do investimento feito.

### 9.4.2. A implementação do Desenho de Serviço e suas implicações organizacionais e tecnológicas

Para implementar o estágio de Desenho de Serviço, uma empresa deve levar em conta o impacto no negócio, elicitar os requisitos de nível de serviço, levantar e avaliar os riscos, executar as atividades de implementação e medir o processos. O Desenho de Serviço traz a necessidade da definição de uma matriz de responsabilidade, na qual as atribuições de cada função devem estar bastante claras.

A fase de estágio de Desenho de Serviço poderá envolver atividades relacionadas a disciplinas, devido às necessidades de desenho de arquiteturas tecnológicas para o serviço, como Engenharia de Requisitos, Gerenciamento de Dados e Informações e Gerenciamento de Aplicações.

Existe também o uso de ferramentas para automatizar atividades de processo, infraestrutura, software, etc., e para aumentar a eficiência, eficácia, a segurança e a qualidade do gerenciamento do serviço durante sua constante operação. Não só é suficiente o uso das ferramentas, pois a dependência de processos e principalmente das pessoas é um ponto muito importante para o sucesso da sua implantação.

### 9.4.3. Benefícios do Desenho de Serviço

- Contribui com a entrega de Serviços com qualidade e custos reduzidos;
- Redução do Custo Total de Propriedade;
- Melhoria da qualidade;

- Melhoria da consistência dos Serviços (frente a estratégias, arquiteturas e restrições da Organização);
- Facilidade na implementação;
- Melhor alinhamento dos Serviços com as necessidades do negócio;
- Melhor Performance;
- Governança aprimorada;
- Melhoria nos processos de gestão;
- Melhores informações para tomada de decisão.

## 9.5. Transição de Serviço

O objetivo principal deste estágio é colocar em produção o resultado, produto que saiu do estágio de Desenho de Serviço, garantindo que os requisitos preestabelecidos de custo, prazo e qualidade estão desenvolvidos e que o impacto seja o menor nas operações correntes da organização [The Cabinet Office, 2011]. Os novos serviços podem ser utilizados de forma a maximizar o valor das operações de negócio.

Neste estágio do serviço há várias características importantes que podem ser envolvidas, como: a Definição de um Sistema de Gerenciamento da Configuração, no qual existe uma Base de Dados de Gerenciamento da Configuração integrada, Processos focados na gestão da mudança organizacional, no planejamento e no suporte à transição, na validação, no teste e na avaliação dos serviços a serem liberados para produção e no gerenciamento do conhecimento acerca de todos os aspectos envolvidos na transição; e a Definição de um Sistema de Gestão do Conhecimento sobre Serviços, como uma ferramenta poderosa para a tomada de decisões mais rápidas e precisas acerca dos serviços.

No processo de transição, a ITIL regulamenta políticas formais focadas em importantes aspectos para o processo de transição, como [Fernandes e Abreu, 2014]:

- Implementação de todas as mudanças no Catálogo de Serviços ou Portfólio através do processo de transição;
- Adoção de um *framewiork* comum e de padrões conhecidos para melhorar a integração das partes envolvidas na transição;
- Maximização da reutilização de processos e sistemas já existentes;
- Integração dos planos de transição às necessidades do negócio, visando maximizar o valor das mudanças;
- Gerenciamento dos relacionamentos com todas as partes interessadas (*stakeholders*) nos serviços;
- Desenvolvimento de sistemas e processos para facilitar a transferência de conhecimento e o suporte às decisões;
- Planejamento dos pacotes de liberação e distribuição;
- Antecipação e gerenciamento das correções de desvios identificados na transição;
- Gerenciamento proativo de recursos através de várias instâncias do processo de transição de serviços;
- Detecção antecipada de falhas (no início do ciclo de vida do serviço), visando reduzir custos de correção;
- Garantia da qualidade do processo de transição e do serviço novo ou alterado já em operação.

### 9.5.1. Os processos da Transição de Serviço

O estágio de Transição de Serviço pode ser visto similarmente com o ciclo de vida de um projeto, com produtos bem definidos e previsão de finalização. Um importante estágio desta

fase é o Suporte à Transição e planejamento, que objetiva planejar e coordenar os recursos necessários para colocar um serviço novo ou modificado no ambiente de produção, dentro do prazo, custo, e da qualidade estimados. Para que a expectativa seja real, deve-se avaliar constantemente a ligação dos planos de transição à estratégia, ter sempre planos integrados em conjunto com o cliente, fornecedores, gerenciar mudanças, problemas, progressos, riscos e desvios, prover suporte aos envolvidos para o cumprimento dos objetivos e continuar o trabalho na monitoração e constante melhoria do desempenho do processo de transição.

O estágio de Transição de Serviço é suportado por sete processos de Gerenciamento de Serviços, descritos a seguir:

- Planejamento e Suporte à Transição: visa planejar e coordenar os recursos necessários para colocar um serviço novo ou modificado no ambiente de produção, dentro do custo, do prazo e da qualidade estimados;
- Gerenciamento de Mudanças: visa assegurar o tratamento sistemático e padronizado de todas as mudanças ocorridas no ambiente operacional, minimizando assim os impactos decorrentes de incidentes/problemas relacionados a essas mudanças na qualidade do serviço e melhorando, consequentemente, a rotina operacional da organização. Para estudar o impacto o fornecedor do serviço pode primeiro configurar o ambiente do cliente em sua empresa e usar o serviço com ajuda de uma equipe especialista em testes para verificar as mudanças necessárias e verificar os impactos antes da instalação;
- Gerenciamento de Ativos de Serviço e da Configuração: abrange identificação, registro, controle e verificação de ativos de serviço e itens de configuração (componentes de TI como hardware, software e documentação

relacionada), incluindo suas versões, componentes e interfaces dentro de um repositório centralizado;

- Gerenciamento da Liberação e da Distribuição: abrange o gerenciamento do tratamento de um conjunto de mudanças em um serviço de TI, devidamente autorizadas (incluindo atividades de planejamento, desenho, construção, configuração e teste de itens de software e hardware), visando criar um conjunto de componentes finais e implantá-los em bloco em um ambiente de produção, de forma a adicionar valor ao cliente, em conformidade com os requisitos estabelecidos na estratégia e no desenho de serviço. A liberação pode ser feita primeiro em um ambiente de homologação o mais parecido possível com o ambiente de produção, para que problemas sejam encontrados o mais cedo possível;
- Validação e Teste de Serviço: relacionado à garantia da qualidade de uma liberação, incluindo todos os seus componentes de serviço, os serviços resultantes e a capacitação do serviço por ela viabilizada. Um serviço validado e testado está pronto para o uso dentro dos propósitos para os quais foi desenhado e construído. No ambiente de homologação deve-se fazer todos os testes possíveis, para que seja minimizado o risco de problemas no ambiente de produção;
- Avaliação de Mudança: visa criar meios padronizados e consistentes para avaliar o desempenho de uma mudança no contexto de uma infraestrutura de TI e serviços já existentes, confrontando-o com as metas previstas, registrando e gerenciando os desvios encontrados. Ao testar o serviço em um ambiente de testes o mais parecido possível do ambiente de produção, é necessário avaliar o quão impactante será uma mudança para o cliente;

- Gerenciamento do Conhecimento: visa garantir que a informação correta seja entregue no local apropriado, para uma pessoa que tenha competência para atuar no tempo certo, habilitando a tomada de decisões informadas. Na prática podem ser usados ambientes como uma Wiki para a empresa, e todo conhecimento importante ser compartilhado, assim todos os interessados terão acesso à informação.

### 9.5.2. A implementação da Transição de Serviço e suas implicações organizacionais e tecnológicas

Para a implementação dos processos de Transição de Serviço, é necessário deixar claro para o cliente que é uma importante estratégia para o negócio, passando para ele todas as fases que se passarão, começando pelo desenho de seus padrões, políticas e relacionamentos, e chegando à institucionalização do processo, levando em conta características de mudança cultural, da compreensão dos riscos e da medição de valor adicionado à empresa.

Tecnologicamente, existe muitas ferramentas que poderão ajudar a transição de serviço, como sistema de gerenciamento da configuração (incluindo a integração das bases de dados de gerenciamento da configuração), ferramentas da colaboração e *workflow*, automação de testes, gerenciamento de massas de testes, automação de distribuição de liberações de software e da logística de hardware, ferramentas de gestão do conhecimento, tais como *dashboards*, gerenciamento eletrônico de documentos, gestão de conteúdo, etc.

### 9.5.3. Benefícios da Transição de Serviço

- Garantir que os clientes e usuários possam usar o serviço novo ou alterado de uma maneira que maximize valor para as operações de negócio.

## 9.6. Operação de Serviço

É um estágio bastante crucial dentro do ciclo de vida do serviço, pois se houver erros no controle, na condução e na gestão das atividades do cotidiano operacional pode impactar comprometendo a disponibilidade do serviço. A operação de serviço visa executar e coordenar as atividades para entregar e suportar os serviços dentro dos níveis estabelecidos com o cliente [The Cabinet Office, 2011].

Um das principais responsabilidades desse estágio é encontrar um ponto de equilíbrio entre diferentes prioridades conflitantes, para minimizar os riscos. Um grande desafio desse princípio é seguir processos, atividades e funções que objetivam a entrega regular dos serviços nos níveis preestabelecidos, dentro de um ambiente sujeito a mudanças imprevisíveis e frequentes [Fernandes e Abreu, 2014].

### 9.6.1. Os processos da Operação de Serviço

O estágio de Operação de Serviço é suportado por cinco processos de Gerenciamento de Serviços:

- Gerenciamento de Eventos: monitora todos os eventos que ocorrem na infraestrutura de TI, para comprovar que a operação está normal. Eventos podem ser exceções (problemas, incidentes, mudanças), alertas ou requisições de informação que terão tratamentos diferentes. Há ferramentas usadas pelo cliente que podem ajudar no acompanhamento de incidentes, o cliente cria *tickets* de solicitação, seja de manutenção ou mudança e o fornecedor pode usar uma equipe de atendimento que checa diariamente a ferramenta;
- Gerenciamento de Incidentes: objetiva minimizar os impactos adversos para o negócio, garantindo que os níveis de qualidade e disponibilidade sejam mantidos dentro

dos padrões acordados (trata o efeito e não a causa). Na prática, quando um colaborador da equipe de atendimento encontrar um evento que possa se tornar um incidente, entra em contato com a equipe responsável, para que possam evitar que o incidente se torne um problema;

- Gerenciamento de Problemas: visa minimizar os impactos adversos de incidentes e problemas para o negócio, quando causados por falhas na infraestrutura de TI, assim como prevenir que incidentes relacionados a essas falhas ocorram novamente. Pode ter uma situação reativa (resolução de problemas em resposta a um ou mais incidentes) ou proativa (identificando e resolvendo problemas e falhas conhecidas antes da ocorrência dos incidentes);
- Cumprimento de Requisições: trata requisições dos usuários que foram originadas a partir de uma solicitação de serviço ou de uma simples solicitação de informação. No dia a dia seria o uso da ferramenta de relacionamento do cliente com o fornecedor e a criação de *tickets* pelo cliente, solicitando requisições;
- Gerenciamento de Acesso: controla o acesso de usuários ao direito de utilizar os serviços, garantindo-o àqueles que foram previamente autorizados e restringindo-o a todos os demais.

### 9.6.2. A implementação da Operação de Serviço e suas implicações tecnológicas

Pode-se ver como boas práticas para a implementação da Operação de Serviço: avaliar e gerenciar os riscos da operação; e a alocação antecipada dos colaboradores nos estágios de desenho e transição. A implementação dessa fase em uma empresa pode ser iniciada pelo gerenciamento das mudanças no ambiente atual da operação e pela utilização de processos de gerenciamento de projetos nas ações de melhoria

que representarem mudanças significativas na infraestrutura ou nos processos.

Existem muitas ferramentas que poderão ajudar a Operação de Serviço, como: ferramentas de auto-ajuda para o usuário (para minimizar a quantidade de eventos); sistema de Gerenciamento da Configuração integrado, que deve permitir que todos os itens de configuração fiquem armazenados juntamente com seus atributos relevantes em uma localidade centralizada; Controle remoto de estações, ferramentas de diagnóstico e elaboração de relatórios, e *dashboards* de indicadores.

### 9.6.3. Benefícios da Operação de Serviço

- Todas as fases do ciclo de vida do Serviço geram valor para o Negócio, no entanto, é nesta fase que o valor é percebido e mensurado;
- Valor para o Negócio oferecido por vários Processos e funções:
    - » Gerenciamento de Eventos: detecção antecipada de incidentes, antes que ocorram interrupções nos serviços;
    - » Gerenciamento de Incidentes: detecta e resolve incidentes, resultando na redução do tempo de indisponibilidade;
    - » Gerenciamento de Problemas: junto com o Gerenciamento de Incidentes e de Mudanças, garante que a disponibilidade e a qualidade dos serviços sejam melhoradas, possibilitando maior produtividade;
    - » Gerenciamento de Acesso: garante que a organização esteja apta a manter mais efetivamente a confidencialidade de sua informação.

## 9.7. Melhoria Contínua de Serviço

O principal objetivo da Melhoria Contínua de Serviço é manter os serviços de TI constantemente alinhados e integrados ao que o negócio necessita [The Cabinet Office, 2011]. É formado por tarefas que lidam com: o planejamento contínuo da melhoria de processos, como o gerenciamento do Plano de Melhoria de Serviços; e identificação de oportunidades de melhoria, análise das informações gerenciais e das tendências quanto ao atingimento dos níveis de serviço e dos resultados esperados pelos serviços; avaliações de maturidade e auditorias internas periódicas; pesquisas de satisfação junto aos clientes.

Esta fase deve ser dirigida por uma equipe onde exista papéis e responsabilidades bem definidas e que tenham representatividade e autoridade para promovê-las. Um programa de melhoria sempre deve estar vinculado às oportunidades de mudança organizacional. É importante também estar alerta às influências internas, como as estruturas organizacionais, cultura, capacidade, que podem ser fontes para oportunidades de melhoria, e as fontes externas, que incluem regulações, concorrência, requisitos de clientes, etc.

Os processos de Gerenciamento do Nível de Serviço, Gerenciamento do Conhecimento e Gerenciamento de Problemas são considerados chaves para este processo, pois fornecem uma base importante de conhecimento estruturado e medido sobre os serviços os quais as ações de melhoria podem ser planejadas e conduzidas adequadamente.

Há dois tópicos cuja importância é fundamental para a implementação e a operação da Melhoria Contínua de Serviço [Fernandes e Abreu, 2014]:

- Medição do Serviço: objetiva fornecer informações sobre o serviço dentro de uma visão completa orientada à integração com o negócio. Para isso, é recomendado o desenvolvimento de um modelo de medição de serviço que estabeleça diferentes níveis para a medição e para a visualização através de relatórios:
    - » Componente: medições acerca de aspectos físicos e técnicos, como disponibilidade, desempenho, capacidade, falhas, mudanças; etc.;
    - » Serviço: medições sobre aspectos funcionais e de relacionamento direto com o cliente (pesquisas de satisfação, reclamações, qualidade do serviço);
    - » Processos que suportam os serviços: medições de progresso, conformidade, eficácia, eficiência;
    - » *Scorecards* de Serviços: visões estáticas periódicas de um serviço em particular;
    - » *Dashboard* de Serviços: contém as mesmas medidas dos *scorecards* de serviços, mas disponibilizadas em tempo real para a TI e para os negócios;
    - » *Scorecard* de TI ou *Balanced Scorecard*: visões de alto nível com consolidações das medições, visando refletir as metas e os objetivos táticos e estratégicos;
- Relato do Serviço: envolve a composição de relatórios de serviço a partir dos dados coletados e monitorados durante a entrega do serviço, além da identificação do seu objetivo, do público-alvo e da utilização planejada para as informações contidas nestes relatórios.

Os resultados das melhorias realizadas e das medições devem estar ligados a vantagens para a organização, que poderão ser avaliados de forma quantitativa como valor sobre investimento ou retorno. Assim, torna-se mais fácil a

comunicação, e mesmo a justificativa, para manutenção e crescimento do programa de melhoria contínua.

Muitas técnicas e métodos são utilizados nessa fase, para garantir consistência, tanto nos relatos de informações quanto na execução das medições e ações de melhoria. Entre eles, podem ser relacionadas *brenchmarkings*, avaliações formais, análise SWOT – *Strengths, Weakness, Opportunities, Threats, Balanced Scorecard*, etc.

### 9.7.1. Processos de Melhoria Contínua do Serviço

Um dos princípios mais importantes deste estágio é a medição do serviço, pois os resultados encontrados nas medições objetivam direcionar atividades para o atingimento de metas, justificar direcionamentos necessários e validar decisões tomadas ou identificar pontos onde é necessário intervir com ações corretivas ou mudanças. As fases para criar métrica para algum ativo é a seguinte:

- Identificar;
- Definir o que se vai medir;
- Obter Dados;
- Processar Dados;
- Analisar os dados;
- Apresentar a informação;
- Implementar ação corretiva.

### 9.7.2. A implementação da Melhoria Contínua de Serviços e suas implicações organizacionais e tecnológicas

Em uma organização este estágio pode ser implementado seguindo vários caminhos. Um deles é a abordagem por serviço, no qual um ponto de melhoria de um determinado serviço é

escolhido como piloto; a abordagem do ciclo de vida focaliza as interfaces deste estágio com os demais estágios. Na abordagem por grupo funcional, o piloto pode ser feito sobre uma equipe responsável por ativos com alto grau de falhas. Em qualquer abordagem, é muito importante considerar o grau de maturidade dos processos de gerenciamento de serviços de TI da organização. Quanto menos maduros esses processos, mais difícil é a aplicação do processo de melhoria em passos.

Conforme dito anteriormente, melhorias implicam em mudanças organizacionais, Na maioria dos casos, a introdução da ITIL certamente começará por ações de melhoria em operações ou serviços que já estão em produção. Assim, deve-se pensar em criar mecanismos de gestão de mudanças organizacionais (desde a sua identificação até a institucionalização), além de estratégias e planos de comunicação por toda a organização.

Tecnologiacamente, há uma conjunto de ferramentas que poderão apoiar a Melhoria Contínua de Serviço (além daquelas já mencionadas nos demais estágios), tais como:

- Ferramentas para análise estatística;
- Ferramentas de *business intelligence* e geração de relatórios.

### 9.7.3. Benefícios da Melhoria Contínua

- O maior valor proveniente da aplicação contínua das práticas de Melhoria de Serviço está no estabelecimento de um círculo contínuo de monitoração e *feedback*, encontrando oportunidades de melhoria dentro de todas as fases do Ciclo de Vida do Serviço;
- Aumenta o aprendizado/conhecimento sobre os Serviços;
- *Feedbacks* permitem melhorar os processos (explicitando o conhecimento);

- Alinha às necessidades do Negócio, no que tange a qualidade, custos e prazos.

## 9.8. Estudo de Caso

O Serpro - Serviço Federal de Processamento de Dados foi criado em 1964, com o objetivo de modernizar e dar agilidade a setores estratégicos da administração pública brasileira. É uma empresa pública vinculada ao Ministério da Fazenda, com papel de prestar serviços em Tecnologia da Informação e Comunicações para o setor público. É considerada uma das maiores organizações públicas de TI no mundo, operando com o desenvolvimento de programas e serviços que permitem maior controle e transparência sobre a receita e os gastos públicos, além de facilitar a relação dos cidadãos com o governo brasileiro. Algumas soluções desenvolvidas são: Declaração de Imposto de Renda, Carteira Nacional de Habilitação, Passaporte Brasileiro e SISCOMEX. Tem sua sede em Brasília, está presente em 11 capitais, conta com mais de 11 mil empregados e investe na formação, capacitação e atualização de suas equipes com uma política de gestão de pessoas.

O Serpro sentiu a necessidade da implantação de um processo, pois havia muitos pontos negatives no fornecimento de serviços: como 28% das ligações eram abandonadas pelo cliente ou usuário; o tempo médio de espera era de 2 minutos; as atividades não eram organizadas por processos; existia uma dificuldade na elaboração de relatórios gerenciais e de indicadores de desempenho (KPI); existia dificuldade de comunicação entre as áreas envolvidas; o processo era descentralizado, pois o atendimento não era realizado em um único ponto de contato; e não havia um tratamento

adequado para os chamados críticos, podendo comprometer o funcionamento do próprio negócio do cliente.

Diante de tais dificuldades, o Serpro tomou a iniciativa de abrir um edital em busca de uma solução capaz de automatizar seus processos com base na biblioteca ITIL para substituir as aplicações de gerenciamento de TI existentes por uma aplicação aderente às melhores práticas. O negócio do Serpro é TI, portanto, seus gestores também necessitavam de informações diárias, semanais e mensais do ponto de vista gerencial e não apenas técnico. Essa demanda foi atendida através da implantação de um DW - *Data Warehouse* de informações de processos ITIL. E também para que o projeto tivesse êxito, foi realizado o treinamento de 6000 colaboradores do Serpro, sendo que 2300 foram treinados de forma presencial e os demais receberam treinamento à distância. Todas estas realizações do Serpro serviram para colocar a mesma na vanguarda da comunidade ITIL no país e para tornar este projeto efetivo.

No final do processo de licitação, o Serpro contratou a empresa SPEKX, que é especializada em soluções de governança e gerenciamento de serviços de TI. Apesar da sua magnitude, o projeto durou pouco mais de 3 anos com a implementação da ITIL versões 2 e 3, e superou as expectativas em termos de volumetria e complexidade.

Uma das melhorias da implementação das melhores práticas ITIL no Serpro foi a consolidação de um único ponto de contato dentro do ambiente de TI disponibilizado para os clientes e usuários dos produtos e serviços Serpro. O novo *Service Desk* passou a contar com mais de 150 atendentes, sendo responsável por todos os acionamentos recebidos, desde o registro até o encerramento. Também foram implementados os processos de incidentes, mudança e configuração.

### 9.8.1. Otimização dos serviços do Serpro com a ITIL

O *service desk* do Serpro, depois da adoção da ITIL, passou a ser responsável por todos os chamados recebidos, desde o registro até o fechamento e está atuando como instrumento de garantia de atendimento dentro dos prazos acordados (SLA), interagindo com todos os processos vinculados aos processos de gerenciamento de incidentes, problemas, requisição de serviços e configuração. Segundo Carvalho (2006a), hoje toda ocorrência, tanto uma solicitação do cliente ou uma falha, passa a ser tratada como um incidente e tem que ser registrada. Se a falha é decorrente de um erro conhecido, é resolvida com base em conhecimento registrado. Se for um erro não conhecido, é restabelecido o mais rápido possível e depois tratado como um problema, para ser descoberto e regularizado, para que não torne a ocorrer.

Gerenciar as atividades em processos permitiu melhorias na determinação dos níveis de serviços. Já existem indicadores definidos e metas organizacionais, podendo identificar quem está atendendo, os prazos, o tempo médio, os tipos de acionamento e todo o conjunto de ações referentes àquele serviço. Este fato está permitindo o aprimoramento gradativo de sua forma de prestar serviços, o que possibilita, cada vez mais, oferecer atendimento de suporte de boa qualidade, com alta disponibilidade, antecipando-se às possíveis falhas e evitando imprevistos para o cliente.

Conforme dados da empresa SPEKX (2009), em 2 anos de funcionamento do *service desk*, após a implementação da ITIL, foi possível reduzir em 98% os incidentes recorrentes no Serpro, além de aumentar o índice de resolução do 1 nível para 89% e cortar em 97% as falhas em processos de mudança. A instalação foi a maior em volumetria ITIL do Brasil, com

250 mil usuários catalogados, 800 mil eventos automáticos por ano e 6 milhões de *tickets*.

A Central de Serviços Serpro opera 24 horas por dia, 7 dias por semana, estando permanentemente ativa para atender as dúvidas e necessidades dos usuários e clientes. De acordo com Carvalho (2006b), o Serpro também investe em programas de qualidade de vida dos funcionários, com o objetivo de garantir melhor desempenho. Todos os processos só terão sucesso na medida em que os empregados envolvidos estejam bem. Isso significa ver o ser humano em toda sua dimensão, procurando melhorar seu desempenho no trabalho, quanto sua vida pessoal.

## 9.9. Considerações Finais

O controle organizacional de uma empresa impulsiona o crescimento de mercado e o alinha com as estratégias de TI. O controle operacional administra os recursos utilizados pela empresa.

A gestão da TI dentro de uma empresa, utiliza não somente os *frameworks* existentes na literatura especializada, mas o *framework* que melhor encaixe-se no cotidiano da mesma. Assim, uma organização trabalhada pela ITIL, está bem estruturada e consciente no quesito referenciado à gestão e ao monitoramento do desempenho aplicado.

A ITIL é compacta e focada em infraestrutura organizacional. Quando aplicada, sua abrangência alavanca o controle operacional e minimiza os riscos. Quando aplicado no controle de utilização de recursos, observa-se o nível de planejamento técnico e operacional, proporcionando à empresa uma visão de onde está e de onde quer chegar.

# PROCESSOS DE SOFTWARE

*Sandro Ronaldo Bezerra Oliveira*
*Alexandre Marcos Lins de Vasconcelos*

O processo de software é um conjunto de atividades realizadas para construir software, levando em consideração os produtos sendo construídos, as pessoas envolvidas e as ferramentas com as quais trabalham, tendo, assim, que levar em consideração em sua definição as atividades a serem realizadas, recursos utilizados, artefatos consumidos e gerados, procedimentos adotados, paradigma e tecnologia adotados, e o modelo de ciclo de vida utilizado [Pfleeger, 2001].

Os trabalhos de Travassos (1994) e Falbo (1998) apresentam alguns conceitos relacionados à definição de processos de software:

- Atividades: São as tarefas ou trabalhos a serem realizados. Uma atividade requer recursos e pode consumir ou gerar artefatos. Para sua realização, uma atividade pode adotar um procedimento. É possível, ainda, a sua decomposição em sub-atividades. Além disso, atividades podem depender da finalização de outras atividades, denominadas pré-atividades. Em função de sua natureza,

atividades podem ser classificadas em: atividades de gerência, atividades de construção e atividades de avaliação da qualidade. Como exemplo de atividade realizada no projeto de software pode-se citar "Especificar os Requisitos do Usuário em Alto Nível";

- Artefatos: São produtos de software gerados ou consumidos por atividades durante a sua execução. Os artefatos podem ser classificados em: artefatos de código, componentes de software, documentos, diagramas, modelos, etc. Seguindo o exemplo anterior, pode-se ter para a atividade como artefato consumido o "Relatório de Entrevista com o Usuário" e como artefato gerado o documento "Especificação dos Requisitos";
- Procedimentos: São condutas bem estabelecidas e ordenadas para a realização de atividades. Determina o apoio de como as atividades são realizadas. Quanto à sua natureza, procedimentos podem ser classificados em: métodos, técnicas e diretrizes. Como forma de executar a atividade definida anteriormente, pode-se fazer uso da UML - *Unified Modeling Language*, como procedimento para a definição de cenários das necessidades do usuário;
- Recursos: Qualquer fator necessário à execução de uma atividade, mas que não seja um insumo para a atividade. Os recursos podem ser classificados em: recursos de hardware, recursos de software e recursos humanos. Finalizando a caracterização do exemplo anterior, pode-se usar como recurso humano o "Analista de Sistemas", o qual poderá fazer uso de uma ferramenta CASE para automatizar o procedimento UML e um editor de texto como forma de documentação.
- Pare o SEI - *Software Engineering Institute* deve-se considerar três dimensões críticas da organização (SEI, 2010), de forma similar a Pfleeger (2001), sendo: processos e

> métodos, pessoas e ferramentas e equipamentos. Assim, o processo é o responsável por manter o elo entre estas dimensões, auxiliando a organização a gerenciar seus recursos, a proceder com seu negócio e manter o conhecimento adquirido com a experiência.

Em uma organização ou grupo de desenvolvimento multi-organizacional, diversos projetos podem coexistir possuindo características específicas. Porém, existe um conjunto de elementos fundamentais necessários e que são incorporados em qualquer processo definido. A norma ISO/IEC 15504 [ISO/IEC, 2003] define o conjunto destes elementos fundamentais como o processo padrão, ou seja, o processo básico que guia o estabelecimento de um processo comum na organização. Desta forma, um processo padrão define uma estrutura única a ser seguida por todas as equipes envolvidas em um projeto de software, independente das características do software a ser desenvolvido.

Segundo Falbo (2000), processo padrão é um modelo básico de processo a ser adaptado e instanciado a cada projeto da organização. Poupando, assim, tempo da organização ao iniciar novos projetos, pois a base do processo já é conhecida e não precisa ser novamente elaborada. Para Pressman (2010), isto é chamado de arcabouço de processo.

Melhoria do processo de software é, portanto, todo esforço empreendido por uma organização para que esse processo de software possa ser utilizado com o menor número de problemas advindos do crescimento de um software [Sommerville, 2010]. No entanto, o fato de que incorporar qualidade ao produto após o seu processo de desenvolvimento é algo que dificilmente poderá ser obtido, mas a qualidade do produto de software pode ser obtida através da qualidade dos processos pelos quais ele é desenvolvido.

Neste contexto, as organizações que se motivarem a melhorar seus processos estarão melhorando seu desempenho e fortalecendo a sua presença em um mercado que está sempre mudando. Acompanhar estas mudanças pode ser facilitada quando os processos das organizações são direcionadas por padrões e modelos de referência de processos [de Mello, 2011]. Essa melhoria orientada a modelos e padrões é mais efetiva e eficiente do que quando considera-se apenas demandas eventuais, iniciativas *ad hoc* dentro da organização [Mutafelija e Stromberg, 2003].

No entanto, a busca pela melhoria do processo de software é uma atividade complexa [Minghui *et al.*, 2004], onde os envolvidos devem possuir conhecimento sobre engenharia de software e serem capazes de usá-lo como orientação para a implementação da melhoria, aumentando as chances de concluir a melhoria com sucesso [Niazi *et al.*, 2006].

Assim, esta parte do livro fornecerá:

- os conhecimentos sobre alguns métodos ágeis que possuem práticas que podem compor um processo de software (Capítulo 10);
- os conceitos sobre os modelos Catalysis, para o desenvolvimento baseado em componentes (Capítulo 11), e TDD, com foco no desenvolvimento orientado a testes (Capítulo 12);
- as definições de uso da técnica do Kanban aplicada para o desenvolvimento de software (Capítulo 13).

10

# MÉTODOS ÁGEIS PARA PROCESSOS DE SOFTWARE

*Kamila Nayana Carvalho Serafim*

Segundo Sutherland *et al.* (2011), um processo rígido ou resistente a mudanças produz produtos medíocres. Os clientes geralmente acabam recebendo o que eles solicitaram primeiramente, mas muitas vezes o produto que eles realmente querem é diferente do recebido. Coletando todos os requisitos no início, o produto é condenado a ser tão bom quanto a ideia inicial, ao invés de ser o melhor, uma vez que as pessoas aprendem ou descobrem como fazer melhor.

Uma maneira encontrada de fazer um produto melhor foi usando os métodos ágeis. Eles surgiram devido a crescente necessidade de melhorar a forma como os softwares eram desenvolvidos e ter um foco principal em satisfazer o cliente, levando em consideração o mercado que exige inovação, produtividade (prazos cada vez mais curtos), flexibilidade e melhoria no desempenho/qualidade dos projetos de desenvolvimento de software.

Os métodos ágeis facilitam o desenvolvimento de software, pois usam práticas para minimizar o risco pelo desenvolvimento

do software como o exemplo das iterações, que em um curto período de tempo é possível ter uma parte do sistema usável, podendo ser avaliado e incluído mudanças pelo cliente.

Nos métodos ágeis existem planejamento, análise de requisitos, projeto, codificação, teste e documentação. Enquanto em um processo convencional, cada iteração não está necessariamente focada em adicionar um novo conjunto significativo de funcionalidades, um projeto de software ágil busca a capacidade de implantar uma nova versão do software ao fim de cada iteração, etapa a qual a equipe responsável reavalia as prioridades do projeto.

Os seguintes princípios dos métodos ágeis podem ser destacados [Succi e Marchesi, 2001]:

- Valorizam garantir a satisfação do consumidor, entregando rapidamente e continuamente software funcional;
- Softwares funcionais são entregues frequentemente (semanas, ao invés de meses);
- Softwares funcionais são a principal medida de progresso do projeto;
- Até mesmo mudanças tardias de escopo no projeto são bem-vindas;
- Cooperação constante entre pessoas que entendem do 'negócio' e desenvolvedores;
- Projetos surgem através de indivíduos motivados, entre os quais existe relação de confiança;
- *Design* do software deve prezar pela excelência técnica;
- Simplicidade;
- Rápida adaptação às mudanças;
- Indivíduos e interações mais do que processos e ferramentas;
- Software funcional mais do que documentação extensa;

- Colaboração com clientes mais do que negociação de contratos;
- Responder a mudanças mais do que seguir um plano.

Assim, este capítulo objetiva focar na particularidade de cada método ágil, para que seja compreendido o ponto forte de cada um e fazer uma análise do que combina e contrasta, elevando o real valor que é de se adaptar às necessidades do cliente.

## 10.1. Métodos Ágeis

Nessa sessão, serão abordadas alguns dos métodos ágeis existentes e mais usados na comunidade.

### 10.1.1. Scrum

O Scrum possui seu foco no gerenciamento de projeto da organização onde é difícil planejar à frente. Mecanismos do Controle de Processo Empírico, onde ciclos de *feedback* constituem o núcleo da técnica de gerenciamento que são usadas em oposição ao tradicional gerenciamento de comando e controle. O Scrum não é um processo prescribente, ou seja, ele não descreve o que fazer em cada situação. Ele é usado para trabalhos complexos nos quais é impossível predizer tudo o que irá ocorrer [Sutherland *et al.*, 2011].

Apesar do Scrum ter sido destinado para gerenciamento de projetos de software, ele pode ser utilizado em equipes de manutenção de software ou como uma abordagem geral de gerenciamento de projetos/programas.

Suas principais características são [Sutherland *et al.*, 2011]:

- Clientes tornam-se parte da equipe de desenvolvimento (os clientes devem estar genuinamente interessados na saída);

- Entregas frequentes e intermediárias de funcionalidades 100% desenvolvidas;
- Planos frequentes de mitigação de riscos desenvolvidos pela equipe;
- Discussões diárias de *status* com a equipe;
- Discussão diária, na qual cada membro da equipe responde às seguintes perguntas:
    - » O que fiz desde ontem?
    - » O que estou planejando fazer até amanhã?
    - » Existe algo me impedindo de atingir minha meta?
- Transparência no planejamento e desenvolvimento;
- Reuniões frequentes com os *stakeholders* (todos os envolvidos no processo) para monitorar o progresso;
- Problemas não são ignorados e ninguém é penalizado por reconhecer ou descrever qualquer problema não visto;
- Locais e horas de trabalho devem ser energizados, no sentido de que "trabalhar horas extras" não necessariamente significa "produzir mais".

Há uma iteração que segue um ciclo (PDCA), que são vistas como caixa de tempo e entrega de incremento de software pronto, sendo chamadas de *Sprint*. As *Sprints* constroem histórias que ficam em um *backlog*, que é um conjunto de requisitos, priorizado pelo Product Owner (representando do cliente dentro da empresa).

Há 3 tipos de reuniões no Scrum:

- Breve reunião diária, em que cada participante fala sobre o progresso conseguido, o trabalho a ser realizado e/ou o que o impede de seguir avançando;
- Breve sessão de planejamento, na qual os itens do *backlog* para uma *sprint* (iteração) são definidos;

- Retrospectiva, na qual todos os membros da equipe refletem sobre a *sprint* passada.

Um princípio-chave do Scrum é o reconhecimento de que, durante um projeto, os clientes podem mudar de ideia sobre o que eles querem e precisam, e que os desafios imprevisíveis não podem ser facilmente tratados de uma maneira preditiva ou planejada tradicional. Como tal, o Scrum adota uma abordagem empírica, aceitando que o problema não pode ser totalmente entendido ou definido, focando na maximização da habilidade da equipe para entregar rapidamente e responder às necessidades emergentes.

### 10.1.2. XP

O XP – *eXtreme Programming* (Programação eXtrema) é um método revolucionário de desenvolvimento de software que se opõe a um conjunto de premissas adotadas pelos métodos tradicionais de engenharia de software. XP consiste numa série de práticas e regras que permitem aos programadores desenvolver software de alta qualidade de uma forma dinâmica e muito ágil [Sato, 2015].

Possui como princípios básicos:

- Presumir simplicidade
- Mudanças incrementais
- Abraçar mudanças
- Trabalho de alta qualidade.

O XP é constituído das seguintes características/práticas [Beck, 1999]:

- **Jogo de Planejamento:** o desenvolvimento é feito em iterações semanais. No início da semana, desenvolvedores e cliente reúnem-se para priorizar as funcionalidades.

Essa reunião recebe o nome de Jogo do Planejamento. Nela, o cliente identifica prioridades e os desenvolvedores as estimam. O cliente é essencial neste processo, ele fica sabendo o que está acontecendo e o que vai acontecer no projeto. Como o escopo é reavaliado semanalmente, o projeto é regido por um contrato de escopo negociável, que difere significativamente das formas tradicionais de contratação de projetos de software. Ao final de cada semana, o cliente recebe novas funcionalidades, completamente testadas e prontas para serem postas em produção;

- **Fases Pequenas:** a liberação de pequenas versões funcionais do projeto auxilia muito no processo de aceitação por parte do cliente, que já pode testar uma parte do sistema que está comprando. As versões chegam a ser ainda menores que as produzidas por outras metodologias incrementais, como o RUP;
- **Metáfora:** procura facilitar a comunicação com o cliente, entendendo a realidade dele. O conceito de rápido para um cliente de um sistema jurídico é diferente para um programador experiente em controlar comunicação em sistemas em tempo real, como controle de tráfego aéreo. É preciso traduzir as palavras do cliente para o significado que ele espera dentro do projeto;
- ***Design* Simples:** simplicidade é um princípio do XP. Projeto simples significa dizer que caso o cliente tenha pedido que na primeira versão apenas o usuário "teste" possa entrar no sistema com a senha "123" e assim ter acesso a todo o sistema, deve-se fazer o código exato para que esta funcionalidade seja implementada, sem se preocupar com sistemas de autenticação e restrições de acesso. Um erro comum ao adotar essa prática é a confusão por parte dos programadores de código simples e código fácil. Nem sempre o código mais fácil de

ser desenvolvido levará à solução mais simples por parte de projeto. Esse entendimento é fundamental para o bom andamento do XP. Código fácil deve ser identificado e substituído por código simples;

- **Time Coeso:** a equipe de desenvolvimento é formada por pessoas engajadas e de forma multidisciplinar (no sentido de incluir pessoas com cada uma das habilidades necessárias para o projeto);
- **Testes de Aceitação:** são testes construídos pelo cliente e conjunto de analistas e testadores, para aceitar um determinado requisito do sistema;
- **Semana de 40 Horas:** trabalhar com qualidade, buscando ter ritmo de trabalho saudável (40 horas/semana, 8 horas/dia), sem horas extras. Horas extras são permitidas quando trouxerem produtividade para a execução do projeto. Outra prática que se verifica neste processo é a prática de trabalho energizado, onde se busca trabalho motivado sempre. Para isto o ambiente de trabalho e a motivação da equipe devem estar sempre em harmonia;
- **Propriedade Coletiva:** o código-fonte não tem dono e ninguém precisa solicitar permissão para poder modificar o mesmo. O objetivo com isto é fazer a equipe conhecer todas as partes do sistema;
- **Programação Pareada:** é a programação em par/dupla num único computador. Geralmente a dupla é formada por um iniciante na linguagem e outra pessoa funcionando como um instrutor. Como é apenas um computador, o novato é que fica à frente fazendo a codificação, e o instrutor acompanha ajudando a desenvolver suas habilidades. Desta forma, o programa sempre é revisto por duas pessoas, evitando e diminuindo assim a possibilidade de defeitos. Com isto, busca-se sempre a evolução da equipe, melhorando a qualidade do código-fonte gerado;

- **Padronização do Código:** a equipe de desenvolvimento precisa estabelecer regras para programar e todos devem seguir estas regras. Desta forma, parecerá que todo o código-fonte foi editado pela mesma pessoa, mesmo quando a equipe possui 10 ou 100 membros;
- **Desenvolvimento Orientado a Testes:** primeiro crie os testes unitários, depois crie o código para que os testes funcionem. Esta abordagem é complexa no início, pois vai contra o processo de desenvolvimento de muitos anos. Só que os testes unitários são essenciais para que a qualidade do projeto seja mantida;
- **Refatoração:** é um processo que permite a melhoria continua da programação, com o mínimo de introdução de erros e mantendo a compatibilidade com o código já existente. Refabricar melhora a clareza (leitura) do código, divide-o em módulos mais coesos e de maior reaproveitamento, evitando a duplicação de código-fonte;
- **Integração Contínua:** sempre que produzir uma nova funcionalidade, nunca esperar uma semana para integrar à versão atual do sistema. Isto só aumenta a possibilidade de conflitos e a possibilidade de erros no código-fonte. Integrar de forma contínua permite saber o *status* real da programação.

### 10.1.3. FDD

Entre os métodos ágeis que existiam antes do manifesto ágil, o FDD - *Feature Driven Developement* é um deles. Concebido originalmente por Jeff de Luca, o FDD surgiu em Singapura, entre os anos de 1997 e 1999 com base no método Coad (Criado por Peter Coad – 1980/1990) e nos processos interativos e lean já utilizados por Jeff de Luca [Palmer e Felsing, 2002].

O FDD busca o desenvolvimento por funcionalidade, ou seja, por um requisito funcional do sistema [Palmer e Felsing, 2002]. É pratico para o trabalho com projetos iniciais ou projetos com codificações existentes. Apesar de ter algumas diferenças entre o FDD e o XP, é possível utilizar as melhores práticas de cada método. O FDD atua muito bem em conjunto com o Scrum, pois o Scrum atua no foco do gerenciamento do projeto e o FDD atua no processo de desenvolvimento.

O FDD possui cinco princípios básicos [Palmer e Felsing, 2002]:

- Desenvolvimento de modelo abrangente (Análise orientada por objetos);
- Construção de lista de funcionalidades (Decomposição funcional);
- Planejar por funcionalidade (Planejamento incremental);
- Detalhe por funcionalidade (Desenho orientado a objetos);
- Construção por funcionalidade (Programação e teste orientado a objetos).

Assim como acontece na metodologia XP, o FDD faz uso de teste de software. Desta forma é possível utilizar TDD – *Test-Driven Development*, aliás, é indicada a utilização deste para manter a qualidade do software.

O FDD, assim como a teoria de sistemas afirma, entende que a soma das partes é maior do que o todo. Desta forma, apesar de criar um modelo abrangente baseado no todo que será desenvolvido, esta fase inicia-se com o detalhamento do domínio do negócio com divisão de áreas que serão modeladas. O modelo só está pronto quando todos da equipe estiverem de acordo, sendo assim, o planejamento é feito com base na lista de funcionalidades. Se for trabalhar com FDD em conjunto com o Scrum, a lista de funcionalidades seria

utilizada no *backlog* de produtos, como histórias de usuários a serem desenvolvidas [Palmer e Felsing, 2002].

Com base na lista de funcionalidades, deve-se planejar por funcionalidade, mas este planejamento é incremental. Isto em conjunto com o Scrum, deve ser analisado como etapa de desenvolvimento do incremento, então este planejamento é feito com base no que será desenvolvido naquele incremento.

Após o planejamento, é feito o detalhamento, nesta fase é de extrema importância que o desenho esteja de acordo com o que o cliente deseja, então é mantido contato constante com o cliente, como em todas as metodologias ágeis. Esta documentação é a base para o desenvolvimento. Não deve-se perder tempo com documentação que não será utilizada, mas é necessário documentar. Posteriormente inicia-se a fase de desenvolvimento, sendo a etapa de desenvolvimento e teste real. O desenvolvimento também é incremental, e indica-se a utilização de testes do início ao fim do processo, além da utilização de integração contínua [Palmer e Felsing, 2002].

Um ponto que diverge do XP é que o FDD incentiva o desenvolvedor como único responsável pelo módulo que este desenvolve, já no XP, o código é comunitário.

### 10.1.4. ASD

Jim Highsmith passou diversos anos trabalhando com métodos predeterministas. Ele desenvolveu, instalou, ensinou e concluiu que tais métodos são profundamente falhos: particularmente para empresas modernas.

Há 3 principais fases não-lineares e sobrepostas no ASD – *Adaptative Software Development* (Desenvolvimento de Software Adaptativo): especulação, colaboração e aprendizado [Highsmith, 2002].

No planejamento tradicional, desvios dos planos são enganos e devem ser corrigidos. Em um ambiente adaptativo, entretanto, desvios guiam à solução correta. Highsmith (2002) vê o planejamento como um paradoxo em um ambiente adaptativo, uma vez que os produtos são naturalmente imprevisíveis.

Este ambiente imprevisível precisa de pessoas trabalhando de maneira integrada pra lidar com a incerteza. A atenção da administração é menor a respeito de dizer às pessoas o que fazer, e maior a respeito de estimular a comunicação, para que as pessoas possam ser criativas por conta própria.

Em ambientes predeterministas, o aprendizado normalmente é desestimulado. As coisas são planejadas de antemão e depois seguem o *design* estipulado. Em um ambiente adaptativo, o aprendizado desafia todos os envolvidos no projeto - desenvolvedores e seus clientes - a examinar suas premissas e a utilizar os resultados de cada ciclo de desenvolvimento para adaptar o seguinte.

O benefício principal, poderoso, indivisível, e predominante do ciclo de vida do Desenvolvimento Adaptativo é que ele força a confrontar os modelos mentais, que estão na raiz das próprias ilusões. Ele força a estimar a habilidade de forma mais realista [Highsmith, 2002].

### 10.1.5. Crystal

Inclui grande número de métodos diferentes que são selecionados de acordo com as características do projeto a ser desenvolvido [Cockburn, 2004]. Hoje em dia, apenas dois dos quatro métodos propostos mantêm o desenvolvimento de novas práticas para cobrir diferentes tipos de projetos.

Os membros da família Crystal são identificados por cores que indicam a intensidade do método. Neste caso, quanto mais escura a cor, maior é a complexidade do projeto.

Existem algumas características comuns à família Crystal, tais como: o desenvolvimento incremental com ciclos de no máximo quatro meses; ênfase maior na comunicação e cooperação das pessoas; não limitação de quaisquer práticas de desenvolvimento; ferramentas ou produtos de trabalho; e incorporação de objetivos para reduzir produtos de trabalho intermediários e desenvolvê-los como projetos evoluídos [Cockburn, 2004].

### 10.1.6. Lean Manufaturing

*Lean manufacturing*, traduzível como manufatura enxuta ou manufatura esbelta, e também chamado de Sistema Toyota de Produção, é uma filosofia de gestão focada na redução dos sete tipos de desperdícios (super-produção, tempo de espera, transporte, excesso de processamento, inventário, movimento e defeitos) [Dailey, 2003]. Eliminando esses desperdícios, a qualidade melhora e o tempo e o custo de produção diminuem. As ferramentas "lean" incluem processos contínuos de análise (*kaizen*), produção "*pull*" (no sentido de *kanban*) e elementos/processos à prova de falhas (*Poka-Yoke*) [Dailey, 2003].

Foi baseado no conceito de Manufatura Enxuta (Lean Manufacturing) que Eric Ries criou o conceito de Startup Enxuta (Lean Startup). Ele usou várias metodologias utilizadas pela Toyota e uniu com outras (como o *Design Thinking*) para criar esse conceito. Isso prova que mesmo uma metodologia utilizada para produção carros pode ser adaptada para qualquer outra área de negócio.

Os pontos-chave do lean manufacturing são [Dailey, 2003]:

- Qualidade total imediata - ir em busca do "zero defeito", e detecção e solução dos problemas em sua origem;
- Minimização do desperdício - eliminação de todas as atividades que não têm valor agregado e redes de segurança, otimização do uso dos recursos escassos (capital, pessoas e espaço).;
- Melhoria contínua - redução de custos, melhoria da qualidade, aumento da produtividade e compartilhamento da informação;
- Processos "*pull*" - os produtos são retirados pelo cliente final, e não empurrados para o fim da cadeia de produção;
- Flexibilidade - produzir rapidamente diferentes lotes de grande variedade de produtos, sem comprometer a eficiência devido a volumes menores de produção;
- Construção e manutenção de uma relação a longo prazo com os fornecedores, tomando acordos para compartilhar o risco, os custos e a informação.

Lean é basicamente tudo o que concerne à obtenção de materiais corretos, no local correto, na quantidade correta, minimizando o desperdício, sendo flexível e aberto a mudanças.

### 10.1.7. Kanban

Kanban é um palavra em japonês que significa literalmente registro ou placa visível. Em Administração da Produção significa um cartão de sinalização que controla os fluxos de produção ou transportes em uma indústria. O cartão pode ser substituído por outro sistema de sinalização, como luzes, caixas vazias e até locais vazios demarcados [Reis, 2008].

Coloca-se um Kanban em peças ou partes específicas de uma linha de produção, para indicar a entrega de uma

determinada quantidade. Quando se esgotarem todas as peças, o mesmo aviso é levado ao seu ponto de partida, onde se converte num novo pedido para mais peças. Quando for recebido o cartão ou quando não há nenhuma peça na caixa ou no local definido, então deve-se movimentar, produzir ou solicitar a produção da peça [Reis, 2008].

O sistema Kanban é uma das variantes mais conhecidas do *Just in Time*, um sistema de administração da produção que determina que nada deve ser produzido, transportado ou comprado antes da hora exata. Pode ser aplicado em qualquer organização, para reduzir estoques e os custos decorrentes. O *just in time* é o principal pilar do Sistema Toyota de Produção ou produção enxuta.

Embora o sistema de Kanban físico seja mais conhecido, muitas empresas têm implementado sistemas de Kanban Eletrônico (*e-Kanban*) em substituição ao sistema tradicional. Diversos sistemas ERP - *Enterprise Resource Planning* e alguns sistemas especializados oferecem a possibilidade de utilização integrada do Kanban Eletrônico, permitindo sinalização imediata da demanda real do cliente em toda a Cadeia de fornecimento. O sistema eletrônico tem como um de seus principais objetivos eliminar problemas comuns à utilização do sistema físico de Kanban como a perda de cartões e a atualização dos quadros [Reis, 2008].

### 10.1.8. Comparativo entre os Métodos Ágeis

O Quadro 6 permite uma consolidação das práticas e dos princípios dos métodos discutidos neste capítulo, como forma de prover um comparativo de uso dos métodos ao longo do desenvolvimento de projetos.

**Quadro 6.** Comparativo entre os Métodos Ágeis [Fernandes e Almeida, 2010]

| Características | XP | SCRUM | FDD | ASD | Crystal |
|---|---|---|---|---|---|
| Abordagem | Incrementos interativos | Incrementos interativos | Iterativo | Iterativo | Incremental |
| Tempo de iteração | 1-6 semanas | 2-4 semanas | 2 dias a duas semanas | 4-8 semanas | Depende da necessidade do projeto |
| Tamanho do time | Pequenos < 20 | Não importa o tamanho | < 10 pessoas | 5-9 pessoas | Não há restrição de tamanho |
| Comunicação | Informal | Informal | Documento | Informal | Informal |
| Envolvimento do cliente | Envolvido | Envolvido como PO | Envolvido através de relatórios | Envolvido através de releases | Envolvido através de releases |
| Especialidades | TDD, user stories, refactoring | Sprint, product backlog, Planning poker, Scrum Master | Diagramas UML | Ciclos de aprendizado | Métodos adaptativos |

## 10.2. Considerações Finais

Quase sempre era exigido um levantamento completo de requisitos, seguido por um projeto de alto-nível, de uma implementação, de uma validação e, por fim, de uma manutenção.

De acordo com essa afirmação, é válido afirmar que as metodologias ágeis vieram para simplificar o processo de desenvolvimento de software. Muitas vezes era necessário no processo criar um documento completo de requisitos, seguido pela fase de desenvolvimento, validação e finalmente uma manutenção, que nem sempre são eficientes e agregam valor, pois cada empresa tem sua particularidade que precisa ser atendida. Muitas empresas funcionam sem necessidade de

documentações extensas, assim ter um grande esforço nessa fase pode levar a uma perda de tempo e produtividade, assim como isolar e dividir cada fase do processo pode trazer prejuízos, como o custo de correção de um *bug* se ele só é descoberto tardiamente. Desta forma, quanto mais conhecimento produzido por pessoas de áreas distintas melhor, por isso as metodologias ágeis estão tão em alta, pois ter na mesma fase requisitos, desenvolvimento e testes faz a integração das partes entregarem um produto com mais qualidade e ter uma equipe mais produtiva.

11

# CATALYSIS E O DESENVOLVIMENTO BASEADO EM COMPONENTES

*Leonardo da Silva Leandro*

O desenvolvimento de software é uma atividade que exige grande conhecimento técnico e teórico. Os tipos de produtos solicitados são tão variados que exigem um conhecimento amplo de metodologias e técnicas de desenvolvimento, para que o time de desenvolvedores possa se adaptar ao tipo de software que precisa ser produzido.

Diferentes técnicas podem produzir diferentes produtos e, normalmente, existem *frameworks* e padrões criados para um propósito específico. Este é o caso do desenvolvimento baseado em componentes, uma técnica de desenvolvimento de software voltada para serviços. Dentro desse ramo pode-se destacar o método Catalysis, o qual combina técnicas de análise e projeto, e procura sempre fazer um refinamento contínuo durante as etapas do desenvolvimento do produto.

Este capítulo trata do desenvolvimento baseado em componentes com foco no método Catalysis. Inicialmente é descrito o processo de desenvolvimento baseado em componentes,

especificando o que é um componente de software e quais são as características desse tipo de prática.

Em seguida são apresentados alguns métodos de desenvolvimento em componentes, com maior detalhe no método Catalysis.

Após isso, são descritas vantagens e desvantagens do uso desse tipo de abordagem de desenvolvimento de software e é apresentado um exemplo de aplicação dessa prática por uma linguagem de programação. E então, é descrito um estudo de caso do uso de processo baseado em componentes para a implementação de um software para gerenciamento de laboratórios de informática em uma universidade.

## 11.1. Desenvolvimento de Software baseado em Componentes

A engenharia de software baseada em componentes é um ramo da engenharia de software com foco em decomposição de sistemas em componentes funcionais e lógicos com interfaces bem definidas. Tais interfaces são responsáveis por realizar a comunicação dos componentes entre si [Oliveira e de Paula, 2009].

O conceito de construir software utilizando componentes não é novo. Esta abordagem faz parte da essência do princípio de dividir para conquistar, muito difundido pelas técnicas de modularidade e reuso de software [Miranda e Lages, 2003]. Quebrar um problema em partes menores ajuda o desenvolvedor não só em achar uma solução para o problema, mas também o ajuda no gerenciamento das atividades.

### 11.1.1. Componente de Software

Um componente de software nada mais é do que uma unidade de composição com interfaces contratualmente especificadas e dependências de contextos explícitas. Por ser uma

unidade de composição, um componente de software pode ser instalado independentemente e ser conectado a terceiros através de suas interfaces [Miranda e Lages, 2003].

A primeira vez que se citou componentização de software foi em 1968, durante uma conferência de engenharia de software nos Estados Unidos. Em 1976, DeRemer propõe um paradigma de desenvolvimento baseado na construção de módulos independentes e depois interligá-los. Na década de 80, surge o paradigma orientado a objetos que fortaleceu esta visão pela possibilidade de reutilização [Oliveira e de Paula, 2009].

Em algumas definições, um componente é tido como um *framework*. Contudo, um framework provê um conjunto de classes inter-relacionadas, onde o relacionamento foi definido pelo projeto. Já um componente estabelece suas comunicações por meio de suas interfaces (contrato estabelecido pela interface) [Oliveira e de Paula, 2009]. Muitas vezes o uso de um *framework* define qual tipo de arquitetura de software deve ser contemplado, mas componentes são módulos independentes desse contexto.

Existem alguns princípios na definição de componentes de software [Eler, 2006]:

- Unificação de dados e funções: um objeto de software encapsula dados e funções que processam esses dados, o que aumenta a coesão do objeto;
- Encapsulamento: os clientes dos objetos de software não precisam conhecer como os dados são armazenados e como as funções processam esses dados, ou seja, conhecer a especificação, não a implementação. Com esse princípio bem aderido, pode-se fazer um melhor gerenciamento das dependências e a redução do acoplamento no software por meio da separação de interesses;

- Identidade: cada componente de software tem uma única identidade, sem considerar o estado do objeto.

Os componentes são autocontidos, o que significa que podem ser reusáveis sem a dependência de outros componentes. Também têm como característica a identificação, que fala sobre estar contido em um único lugar, ao invés de espalhados e misturados com outros artefatos de software ou documentação [Melo, 2005].

No tocante às interfaces, estas podem ser classificadas da seguinte forma [Melo, 2005]:

- Interfaces fornecidas (*provided interfaces*): define os serviços oferecidos pelo componente por meio de operações. Durante a implementação, essas interfaces devem ser identificadas e definidas separadamente;
- Interfaces requeridas (*required interfaces*): define os serviços que o componente necessita de outros componentes. Componentes conectam-se por meio da interface requerida de um com a interface fornecida de outro. Se um componente for autocontido, então não existirão interfaces requeridas. Elas devem ser definidas explicitamente.

A Figura 42 mostra um exemplo simples de componentes modelados através da notação UML – *Unified Modeling Language*. Neste exemplo, o componente é responsável por processar endereços de diferentes países. Esse componente pode ser utilizado para criar e manter endereços postais em diferentes países. É possível visualizar na imagem as interfaces requeridas e fornecidas pelo Componente Endereço.

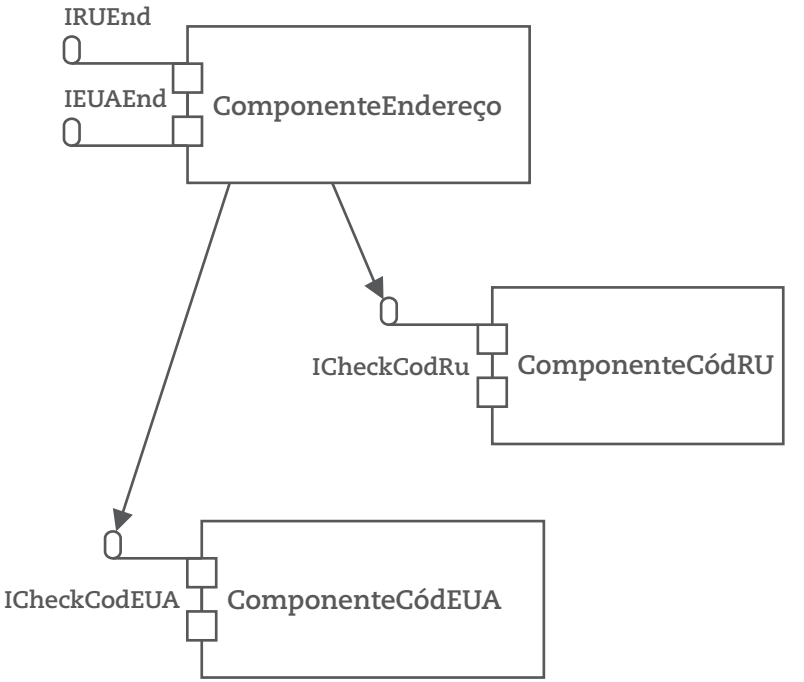



**Figura 42**. Ilustração do ComponenteEndereço

Já a Figura 43 mostra como o uso de interfaces permite aos componentes serem facilmente substituídos sem que a essência da funcionalidade sofra algum impacto. No exemplo, o cliente utiliza interfaces estáveis e não é afetado pela substituição do componente utilizado (existente), pois o novo componente oferece a interface antiga utilizada (IX). Além disso, o novo componente pode oferecer novas funcionalidades (IX+) para os clientes antigos ou novos.

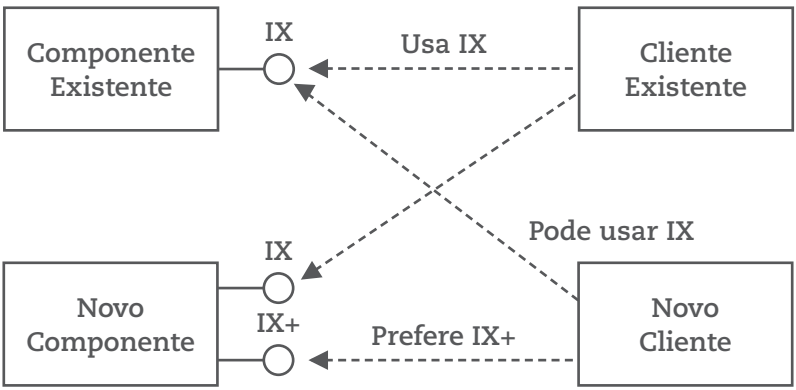



**Figura 43**. Substituição de Componentes com Impacto Baixo

### 11.1.2. Contratos

Contratos relacionados ao comportamento de componentes de software são criados para garantir que os módulos de um sistema irão manter a forma como interagem ao longo do processo de desenvolvimento. Esta abordagem é também conhecida como *Design By Contract* [Miranda e Lages, 2003]. Uma vez que um contrato é um acordo formal entre duas ou mais partes, no tocante aos componentes funcionais da mesma forma, tendo em vista que suas interfaces são as partes envolvidas no "acordo". Os contratos podem ser de três tipos [Miranda e Lages, 2003]:

- Pré-condição;
- Pós-condição;
- Invariante.

A pré-condição funciona como uma forma de garantir ao cliente que ele terá sucesso na chamada de um método de uma determinada interface. Ao final da chamada do método,

são asseguradas algumas propriedades do componente que implementa a interface, isso por meio de uma pós-condição.

Além destes, ainda existem componentes que possuem um estado que deverá ser mantido ao longo de toda sua existência. Estes estados são assegurados através de invariantes [Miranda e Lages, 2003].

Os contratos podem ser escritos em linguagem natural ou OCL - *Object Constraint Language*. A Figura 44 ilustra a realização de interfaces a partir de modelos UML. Desta forma, é possível visualizar os métodos que cada interface (seja ela requerida ou fornecida) provê para os componentes que as realizam.

### 11.1.3. Processo de Desenvolvimento

Considerando-se que todo desenvolvimento do sistema está centrado nos componentes envolvidos e na sua arquitetura, podem-se considerar dois processos distintos [Miranda e Lages, 2003]:

- Composição de componentes;
- Desenvolvimento de componentes.

Inicialmente, deve-se fazer uma análise de quais componentes podem ser utilizados para satisfazerem os requisitos levantados. Esta busca não está limitada apenas a uma biblioteca de componentes da equipe ou da empresa. Ainda existe a possibilidade de adaptação de componentes para atenderem aos requisitos.

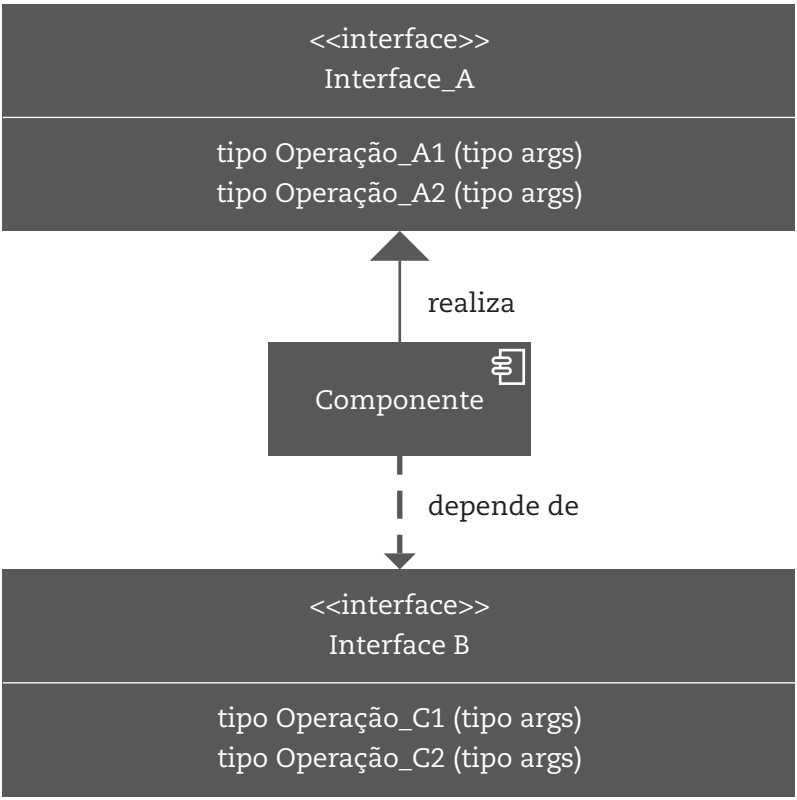



**Figura 44**. Esquema UML mostrando a Realização de uma Interface por Meio de um Componente e a Dependência deste com outra Interface B, por meio dos Métodos Requeridos.

Já a construção (desenvolvimento) dos componentes é considerada como o processo menos atraente, uma vez que requer mais esforço e tempo. Todavia, oferece a oportunidade de desenvolver funcionalidades do sistema que vão ser, provavelmente, as de maior valor para a empresa [Miranda e Lages, 2003].

Para definir um processo de software baseado em componentes, algumas atividades devem ser seguidas, a saber [Oliveira e de Paula, 2009]:

- Especificação de requisitos;
- Especificação do sistema;
- Projeto de arquitetura;
- Projeto interno dos componentes;
- Implementação e teste do sistema;
- Armazenamento dos componentes em bibliotecas.

A Figura 45 apresenta de forma resumida as etapas do desenvolvimento baseado em componentes, através de um fluxo direcionado de ações.

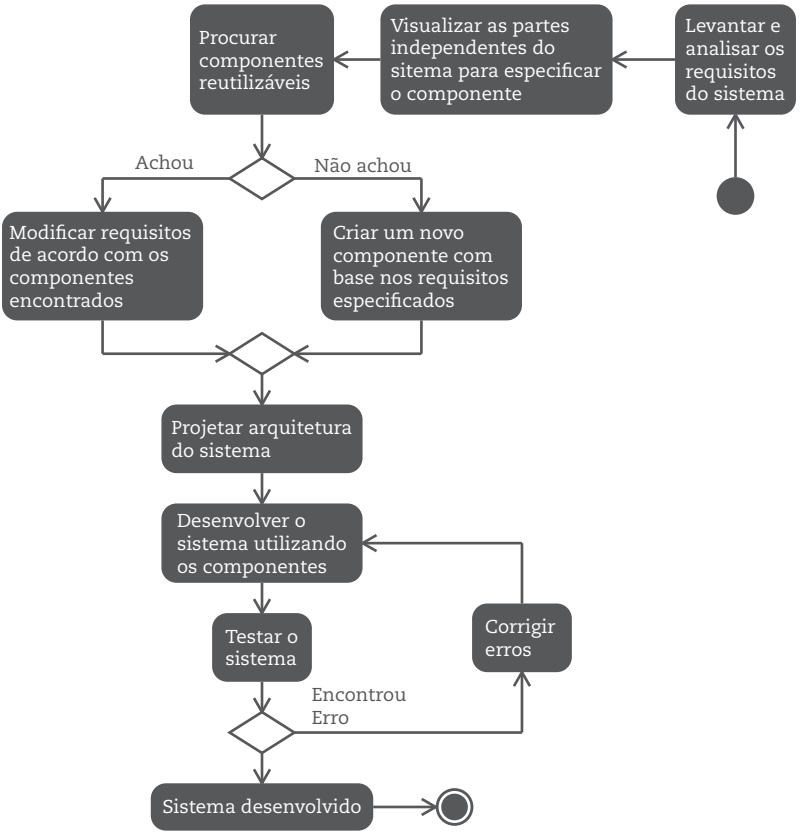



**Figura 45**. Processo de Desenvolvimento baseado em Componentes

Inicialmente, tem-se a etapa de especificação de requisitos. Esta fase diz respeito ao entendimento do negócio, de acordo com as informações que são adquiridas do cliente após as reuniões de planejamento. Esta fase do processo de desenvolvimento é muito importante, pois é a partir desta que as demais fases serão baseadas.

As técnicas usadas para prover o volume de informações que será usado pela equipe de desenvolvimento não apresentam nenhuma surpresa quando em comparação com métodos tradicionais. Sendo assim, tradicionalmente criam-se dois artefatos, o Modelo de Casos de Uso e o Modelo Conceitual.

O modelo de caso de uso projeta os requisitos funcionais esperados no sistema em um diagrama que serve como base para o modelo conceitual. Os casos de uso envolvem atores, os quais podem ser inerentes ao sistema ou serem externos.

Já o modelo conceitual permite que os requisitos e serviços requisitados fiquem representados de uma forma mais próxima do código. Os diagramas de classes e componentes detalham atributos e métodos, que já deixam claro uma "carcaça" do programa que será implementado. Ao final dessa etapa, uma representação da arquitetura está disponível.

As Figuras 46 e 47 mostram, respectivamente, exemplos de um modelo de casos de uso e um modelo conceitual (diagrama de classe). Enquanto que um caso de uso descreve uma funcionalidade de forma alto nível, em linguagem natural, o diagrama de classe detalha, em nível técnico, o caso de uso que será implementado, definindo as classes que irão compor o projeto.

#### 11.1.3.1. Projeto Interno dos Componentes

O desenvolvimento de software baseado em componentes pode considerar o desenvolvimento de componentes e o desenvolvimento com componentes [Melo, 2005].

Primeiramente, busca-se por componentes que já estejam prontos e sejam adequados à especificação do sistema [Oliveira e de Paula, 2009]. A ideia por trás disso é justamente reaproveitar o que já foi feito e que já está funcionando corretamente, de forma a ganhar tempo e manter a qualidade.

Porém, nem sempre é possível aproveitar tudo, ou mesmo pode-se não aproveitar nada. Sendo assim, em muitos casos os componentes acabam sendo construídos do zero. Consequentemente, trata-se de uma etapa mais demorada, onde toda uma análise das funcionalidades necessárias para o componente deve ser feita, bem como uma validação do que foi pensado e testes para garantir que está tudo nos conformes.

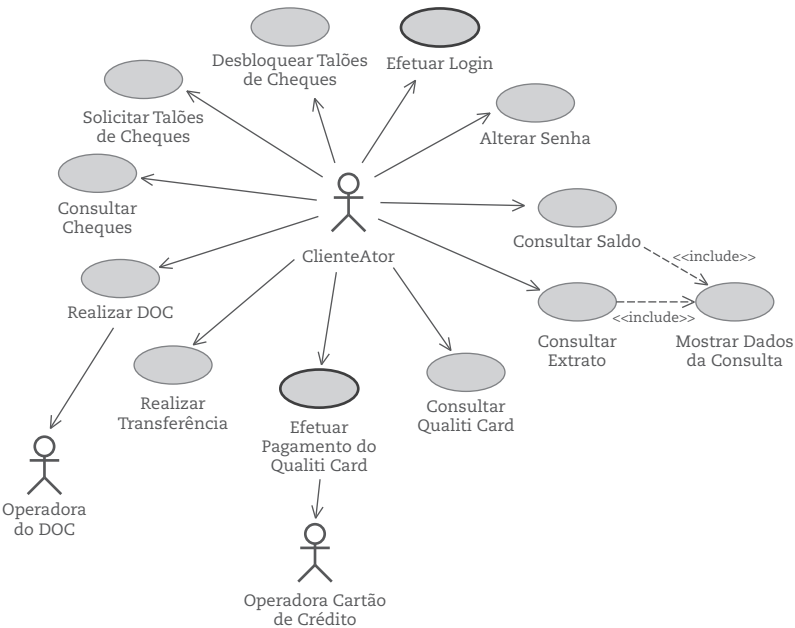



**Figura 46**. Modelo de Caso de Uso para um Software de Operações Bancárias

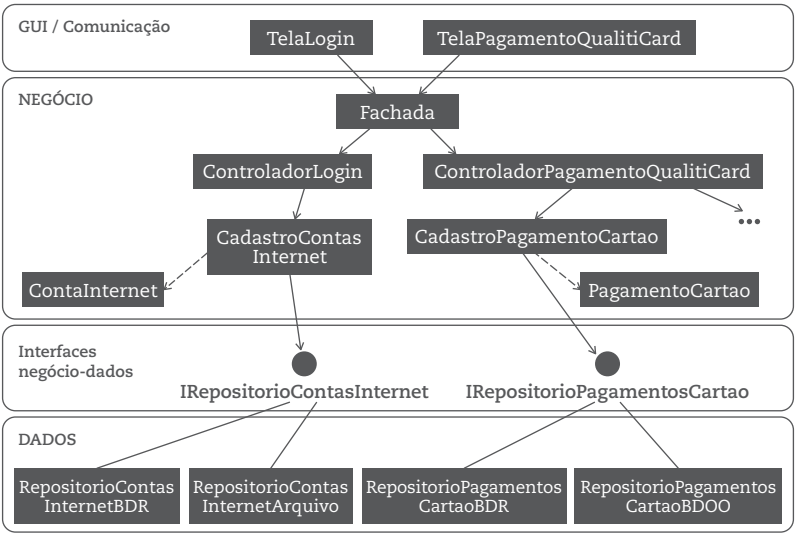



**Figura 47**. Diagrama de Classes para um Software de Operações Bancárias

Tendo em mãos o projeto dos componentes, é a vez de se modelar a arquitetura do sistema. Os componentes que foram desenvolvidos passam a serem organizados de forma a facilitar a integração do sistema, definindo estruturas de controle, protocolos de comunicação, padrões de projeto e as interfaces [Oliveira e de Paula, 2009].

Na fase final, os componentes são apresentados em nível de código. Testes de corretude e completude dos requisitos funcionais e não funcionais são realizados, e então o projeto chega a sua fase final, com a documentação e a catalogação dos componentes criados.

### 11.1.4. Métodos de Modelagem de Componentes

A modelagem de componentes é feita através de diferentes métodos. Estes oferecem um conjunto de ferramentas e regras que permitem ao engenheiro de software construir modelos arquiteturais complexos, com um nível de detalhamento muito grande, aproximando-se bastante do nível de código. Nesta seção serão descritos dois métodos: UML e o KobrA.

#### 11.1.4.1. UML

A UML - Unified Modeling Language é a base da grande maioria (se não de todos) dos métodos de modelagem de classes e objetos. Ela não é um processo de desenvolvimento completo, uma vez que não inclui atividades relacionadas com o processo gerencial, e enfatiza as etapas de análise, projeto e, em menor grau, implementação.

Outra definição para UML é de uma família de notações gráficas, apoiadas por um metamodelo único, que ajuda na descrição e no projeto de sistemas de software, particularmente os que são construídos a partir do paradigma orientado a objetos (OO) [Fowler, 2004].

Uma das questões interessantes sobre a UML como linguagem de programação é como modelar lógica comportamental. A UML 2 oferece três espécies de modelagem comportamental: diagramas de interação, diagramas de estado e diagramas de atividade. As Figuras 48, 49 e 50 mostram, respectivamente, exemplos dos diagramas de interação, de estado e de atividade.

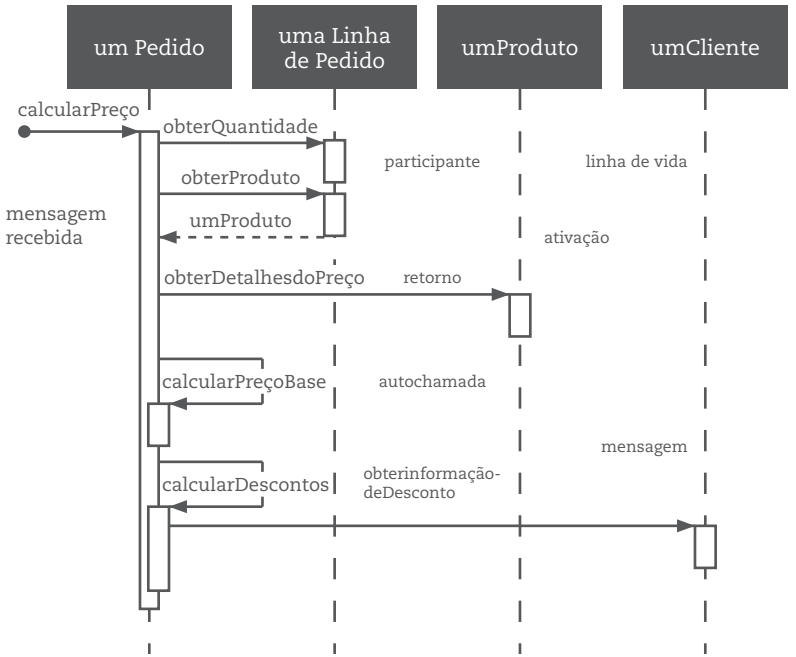



**Figura 48**. Diagrama UML de Interação Simples

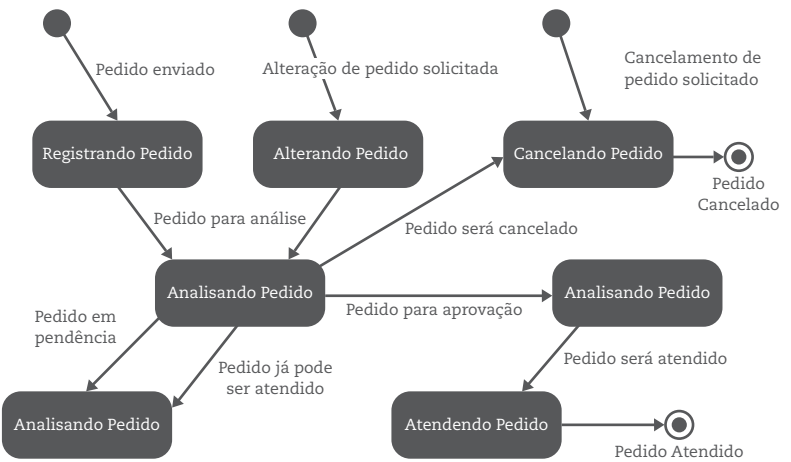



**Figura 49**. Diagrama UML de Estados

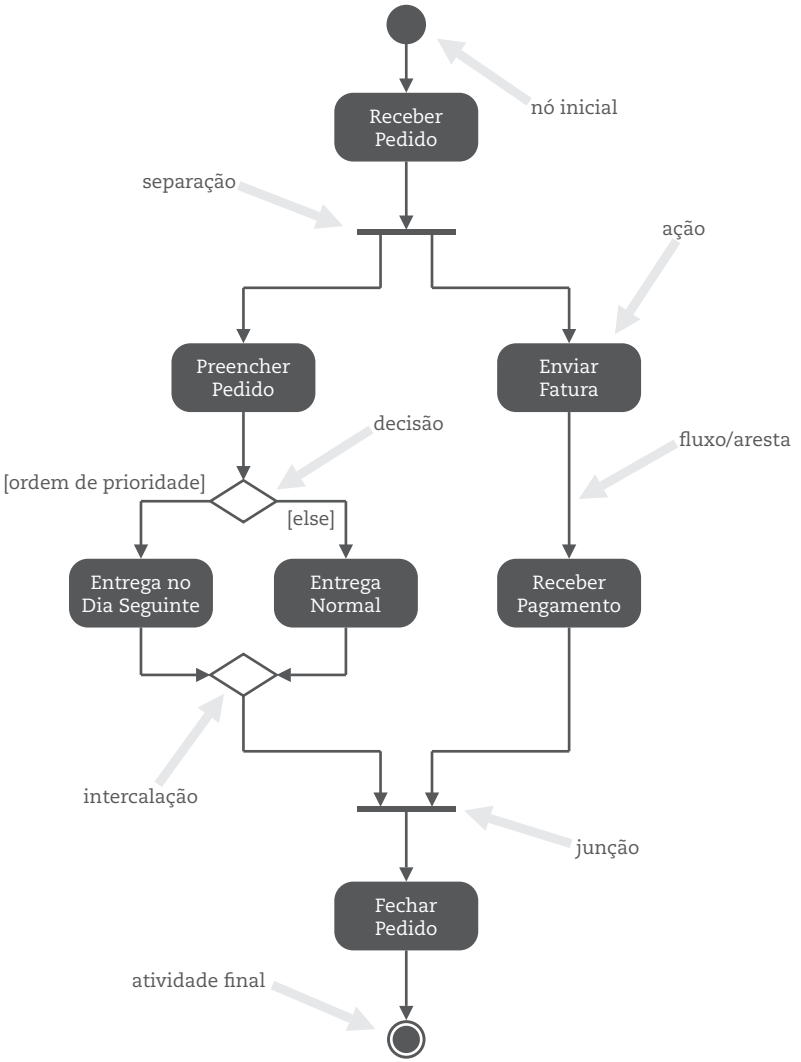



**Figura 50**. Diagrama UML de Atividades

A UML também tem uma variação na forma de modelar. Pode ser feita uma modelagem conceitual ou uma modelagem de software. A maioria dos usuários está familiarizada

com a modelagem de software. Nesta perspectiva, os elementos da UML são mapeados diretamente nos elementos de um sistema de software.

Por outro lado, na perspectiva conceitual a UML representa uma descrição dos conceitos de um domínio de estudo [Fowler, 2004].

Em suma, não existem regras rígidas e diretas sobre perspectiva. Algumas ferramentas são construídas no intuito de transformar, de forma automática, código-fonte em diagramas UML, oferecendo outra forma de visualizar o projeto que não somente em código-fonte. Esse tipo de aplicação parece-se muito com uma perspectiva de software. Mas, se diagramas UML são usados para tentar entender os vários significados do termo fundo de bens com vários contadores, deve-se estar com uma disposição de espírito muito mais conceitual [Fowler, 2004].

O diagrama de classes é um dos mais utilizados pelos engenheiros de software, para apresentar a arquitetura do software que irá ser produzido. Este tipo de diagrama mostra cada classe do projeto sendo representada por blocos. Além disso, é possível criar estereótipos para cada classe, descrevendo que tipo de entidade ela representa. Os estereótipos são opcionais e o usuário pode criar quantos quiser. Nos conceitos estudados em análise e projeto de software, existe uma convenção de se anotar certos estereótipos para determinadas classes do projeto orientado a objetos. Alguns exemplos desses estereótipos são:

- <<*entity collection*>>: para identificar que determinada classe representa uma coleção de certa entidade;
- <<*interface*>>: destacar uma interface do sistema;
- <<*entity*>>: representa uma entidade persistente do sistema.

A Figura 51 apresenta um diagrama de classes simples anotado com alguns estereótipos. O diagrama representa uma modelagem para um sistema bancário.

### 11.1.4.2. KobrA

O método KobrA foi desenvolvido no Insituto Fraunhofer e patrocinado pelo ministério alemão de pesquisa e tecnologia, com o objetivo de dar apoio ao método ágil de desenvolvimento dirigido por modelos (MDD - *Model-Driven Development*) [Eler, 2006]. Assim como muitos outros métodos de "componentização", o KobrA também tem sua base na UML, usando este para descrever a estrutura e comportamento essencial dos componentes. Desta forma, é possível garantir que os benefícios do desenvolvimento baseado em componentes possam ser obtidos em todo o ciclo de vida do software, aumentando a reusabilidade dos componentes produzidos.

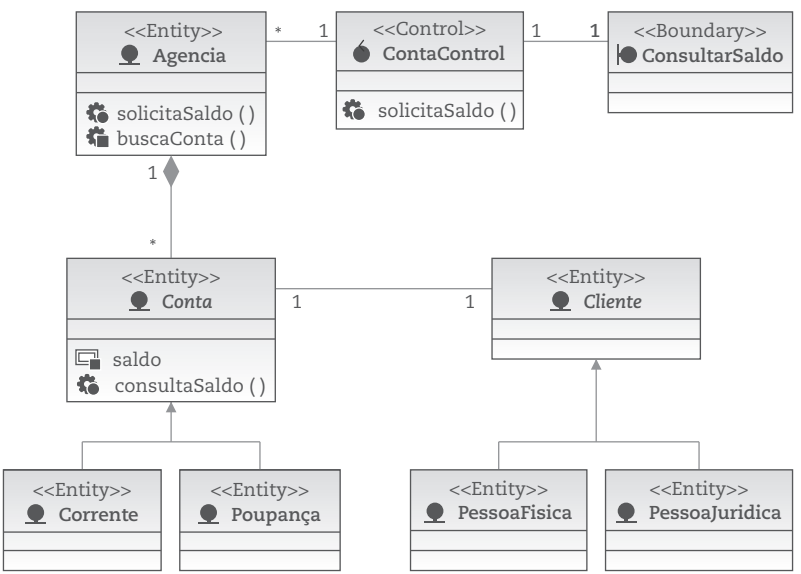



**Figura 51**. Digrama de Classes com alguns Estereótipos Anotados

No tocante ao MDD, este busca desenvolver uma arquitetura dirigida por modelos em que as funcionalidades principais do sistema sejam descritas independentes de plataformas específicas de implementação. Esta é a essência da reusabilidade aplicada por este método, sempre visando o aumento da qualidade dos componentes e dos sistemas formados por eles.

O método KobrA tem por objetivo principal fornecer apoio concreto para o desenvolvimento e aplicação de *frameworks* de domínio específico baseado em componentes com objetivo de desenvolver linhas de produto de software. O método procura desenvolver um *framework* genérico, tido como o artefato principal, e este é instanciado pela tomada de decisões sobre quais funcionalidades farão parte da aplicação a ser gerada [Eler, 2006].

No KobrA seus componentes tem um nome particular: Komponente [Eler, 2006]. Cada um destes componentes do *framework* é descrito por um diagrama UML adequado. A ideia é que essas estruturas sejam vistas como se fossem sistemas independentes e essa descrição é realizada em duas etapas: a especificação, a qual descreve as características externas e visíveis do componente, bem como especifica os requisitos; e a realização, que define como o componente satisfaz os requisitos elicitados.

Como já foi dito antes, a influência da UML neste e em outros métodos é notória, seja direta ou indiretamente. Sendo assim, os Komponentes do KobrA são especificados através de quatro diagramas: estrutural, comportamental, funcional e de decisão; que descrevem as diferentes variações do Komponente [Eler, 2006].

A Figura 52 ilustra a estrutura de componentes do método KobrA. Nela é possível visualizar que sua organização estrutural é baseada em árvore.

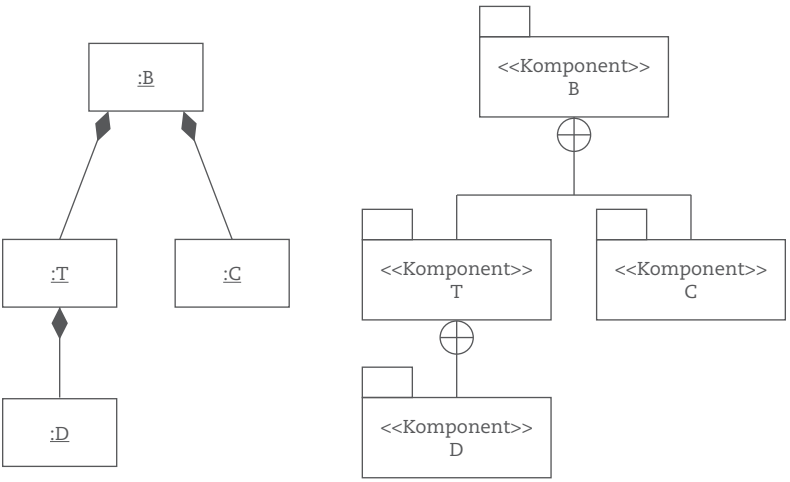



**Figura 52**. Estrutura em Árvore dos Komponetes KobrA

## 11.2. Catalysis

Catalysis tem origem na Química por conta do conceito de catalisadores químicos, responsáveis por acelerar reações químicas sem serem consumidos no processo. Um catalisador permite que reações ou processos sejam acelerados ou retardados com condições bem controladas [Melo, 2005].

O método Catalysis incorporou a UML, que, como foi visto anteriormente, é um método aberto usado para especificar, visualizar, construir e documentar os artefatos de um sistema de software. Contudo, o Catalysis possui modificações no conceito geral da UML em virtude de seus diferentes diagramas que compõem o processo. Desta forma, ele permite que o processo de desenvolvimento seja adaptado às características do projeto, possibilitando assim a definição da melhor sequência de atividades e artefatos para cada caso.

Fundamenta-se nos princípios de abstração, precisão e componentes "*plug-in*". Com relação ao princípio de abstração, ele

procura orientar o desenvolvedor na busca dos aspectos essenciais do sistema, dispensando detalhes que não são relevantes para o contexto do sistema. Já o princípio precisão tem como objetivo descobrir erros e inconsistências na modelagem, tendo em vista que o artefato final esteja coerente e pronto para ser usado como base na codificação. Por fim, o princípio de componentes "*plug-in*", que dão suporte ao reuso de componentes para construir outros sistemas [Melo, 2005]. O termo "*plug-in*" nesse contexto significa que os componentes são facilmente retirados e adicionados em outros sistemas, sem que essa ação necessite de grandes manutenções no código.

Uma característica do Catalysis é a sua capacidade de eliminar ambiguidades ao longo da modelagem. Particularmente, isso é importante para a modelagem baseada em componentes, onde leitores e escritores de uma especificação de interface podem não ter contato entre si [Miranda e Lages, 2003].

O Catalysis começa mostrar suas diferenças com relação a outros métodos de modelagem de componentes já na especificação dos requisitos. Nesta etapa inicial, além do modelo de casos de uso e de diagramas de estado, o método também possui os *snapshots*, que, de forma resumida, são como "fotos" tiradas em diferentes momentos da modelagem com o intuito de identificar algum problema (ou mesmo uma solução) por conta de mudanças feitas em diferentes momentos do processo.

### 11.2.1. Casos de Uso

A visão que o Catalysis tem dos casos de uso é de uma especificação de uma sequência de ações, incluindo variantes, que um sistema (ou outra entidade) pode realizar, interagindo com os atores do sistema. Todavia, não é apenas desta forma que os casos de uso são visto por esse método, mas também como uma ação conjunta [Miranda e Lages, 2003].

Bem como ocorre numa ação conjunta, um caso de uso reflete a interação entre vários objetos e também qual será o efeito em cascata nesses objetos.

### 11.2.2. Estados

Na Figura 53 é possível observar o diagrama de estados do Catalysis [Miranda e Lages, 2003]. Ele serve para ilustrar os estados possíveis de um objeto e suas transições. Um objeto pode ter estados sob perspectivas distintas, de forma a ser possível criar diagramas também distintos para cada perspectiva.

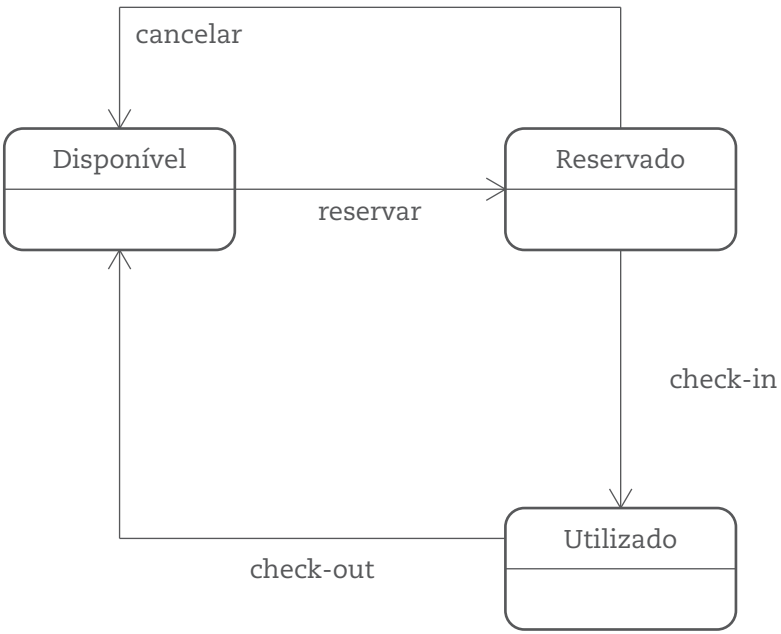



**Figura 53**. Diagrama de Estados do Método Catalysis

Os diagramas de estados também definem ações. Estas são as transições entre estados, uma vez que para um objeto ir de um estado A para um estado B é necessária uma ação.

Sendo assim, a Figura 54 apresenta um diagrama de estados, no qual pode-se definir pré e pós-condições para as transições [Miranda e Lages, 2003].

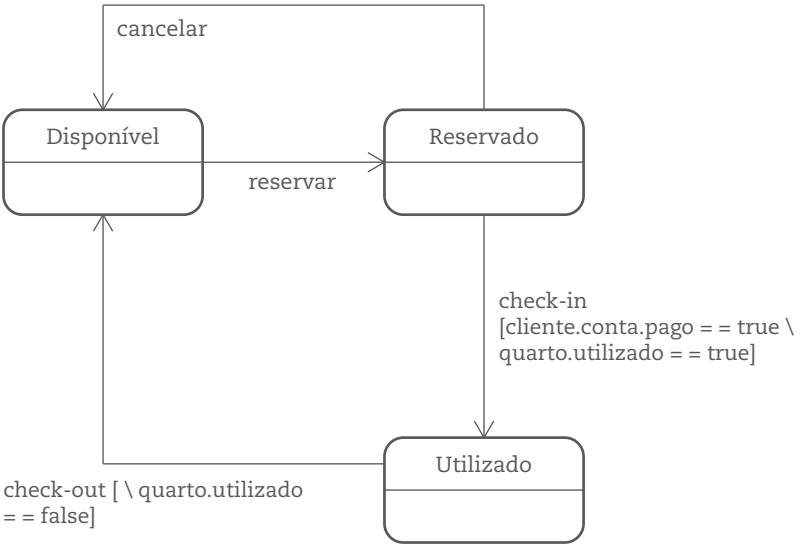



**Figura 54**. Diagrama de Estados com Pré e Pós-condições para as Transições

### 11.2.3. *Snapshots*

Um ponto diferencial do Catalysis para realizar a modelagem comportamental é usando os *snapshots*. Um snapshot é como uma "foto" dos objetos de um componente, que é tirada em determinados momentos do processo [Miranda e Lages, 2003].

É muito comum durante a modelagem de uma determinada ação usar o esquema de *snapshots* antes e depois da realização da ação. Desta forma, é possível destacar as alterações que ocorreram em virtude da aplicação da ação.

A Figura 55 esclarece o papel dos *snapshots* [Miranda e Lages, 2003]. Nela é possível ver como estava o modelo antes e depois da aplicação de uma determinada ação.

### 11.2.4. Identificação de Componentes e Fatoração de Modelos

O Catalysis não tem um modelo sistemático para a identificação de componentes [Miranda e Lages, 2003]. Este tipo de atividade deve ser feita com base nas necessidades que o componente deverá suportar. No entanto, em geral, cada colaboração deverá representar um modelo de interface do componente. Uma colaboração nada mais é do que um conjunto de ações que têm como resultado uma alteração nos estados dos objetos relacionados.

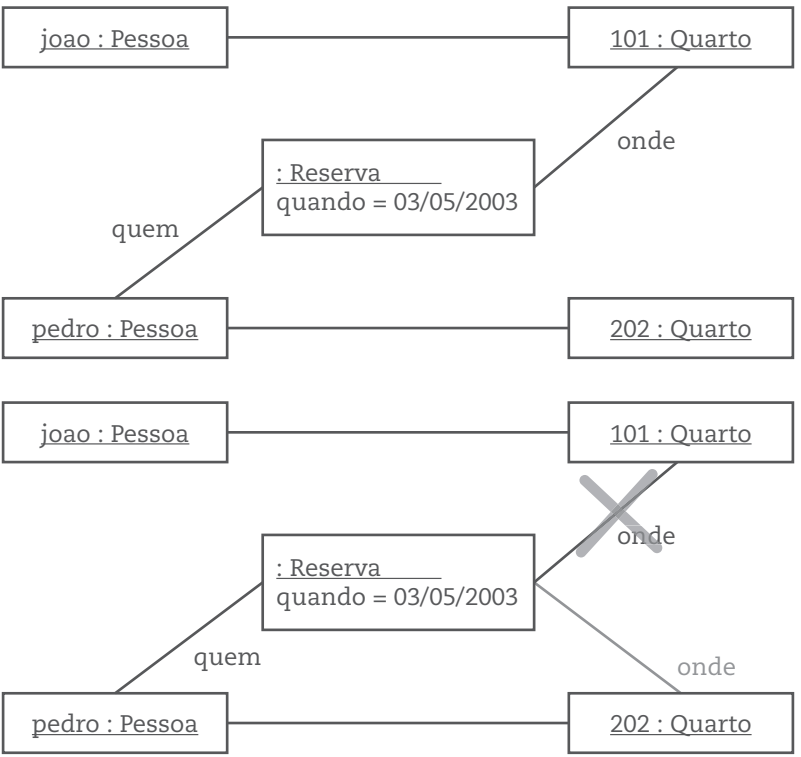



**Figura 55**. *Snapshots* registrando as Mudanças ocorridas após a Aplicação de uma Ação

No tocante à fatoração de modelos, a ideia é que para projetar sistemas de grande porte é necessário dividi-lo em partes menores (dividir para conquistar!). Uma vez que as partes estejam prontas, é preciso montar o sistema, realizando a composição dos submodelos. Para qualquer tipo de divisão fazem-se necessários a abstração e o refinamento dos modelos [Miranda e Lages, 2003].

Quando se fala em abstração durante uma modelagem, quer dizer que se deve deixar visível apenas aquilo que é importante para um propósito particular. No caso do Catalysis, é interessante deixar o conjunto de objetos, as ações e as colaborações da maneira mais genérica possível [Miranda e Lages, 2003]. Vale salientar que a modelagem no Catalysis não tem foco na implementação, de forma que a sua própria modelagem, no nível mais abstrato, é uma abstração da implementação.

Para cada refinamento e/ou abstração, é indispensável deixar claro como isso foi feito, mantendo-se uma espécie de rastreabilidade (*traceability*) [Miranda e Lages, 2003].

## 11.3. Vantagens do Desenvolvimento baseado em Componentes

O processo de desenvolvimento baseado em componentes traz vantagens que vão além do simples fato de facilitar o desenvolvimento por conta da divisão em unidades funcionais (componentes).

O software baseado em componentes é extensível por definição e novos componentes podem ser acrescentados a qualquer momento [Melo, 2005].

É seguro, pois a instalação de novos componentes no ambiente do usuário não deve invalidar os componentes já instalados [Melo, 2005]. O termo "*plug-in*" diz respeito justamente a essa facilidade de retirar/acrescentar componentes, sem que ocorram impactos na estrutura do que já existe no sistema.

A ideia de existir um *garbage collection* (coletor de lixo) é de suma relevância, uma vez que um componente nunca pode saber quando outro componente pode ser liberado [Melo, 2005].

O desenvolvimento em componentes permite a diminuição de tempo de desenvolvimento de sistemas de software em razão do reuso dos componentes e facilita a gestão das mudanças que os sistemas sofrem no decorrer do tempo [Eler, 2006].

## 11.4. Suporte das Linguagens

Uma vez que um componente é uma unidade de composição, ele precisa do suporte das linguagens de programação para que sua funcionalidade seja aplicada ao sistema da forma esperada.

É necessário um mecanismo para definir interfaces: "O que ofereço" (interface de cima); "O que preciso" (interface de baixo) [Melo; 2005]. Na Figura 56 é possível visualizar essas necessidades através das interfaces de cima e de baixo em um sistema de locadora de vídeo. É possível ver como a interface encaixa-se com as necessidades do cliente.

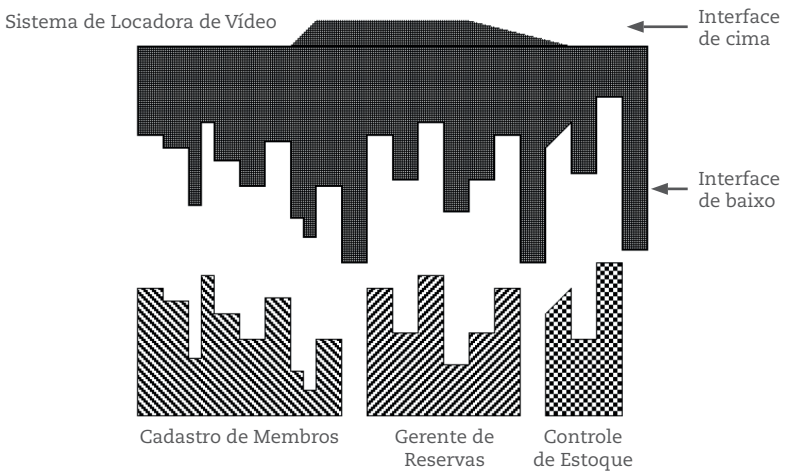



**Figura 56**. Mecanismo para definir Interfaces

O Enterprise JavaBeans (EJB) é uma tecnologia usada na linguagem Java que dá suporte ao desenvolvimento em componentes. Os EJBs foram criados em março de 1998, na primeira especificação da Sun, com o intuito de prover uma arquitetura de objetos distribuídos pela internet [Melo, 2005].

O EJB é mais do que uma API - *Application Programming Interface*, ele se caracteriza como uma arquitetura de componentes para o desenvolvimento e utilização de aplicações corporativas baseadas em componentes distribuídos [Melo, 2005].

No EJB, existem três entidades principais da sua arquitetura:

- *Enterprise Bean* ou simplesmente *Bean*: são os componentes do EJB que ficam hospedados nos *containers* EJB;
- *Containers* EJB: tem a função de gerenciar todos os aspectos do *bean* durante a execução, incluindo acesso remoto, segurança, persistência, transações, concorrência e acesso a recursos;
- Servidor EJB: embora não exista uma clara distinção em relação aos *containers* EJB, o papel do servidor é apresentar diferentes *containers*, uma para cada tipo de *bean*, e prover serviços que permitam que os *containers* gerenciem os *beans*.

A Figura 57 mostra como é o relacionamento entre essas três entidades.

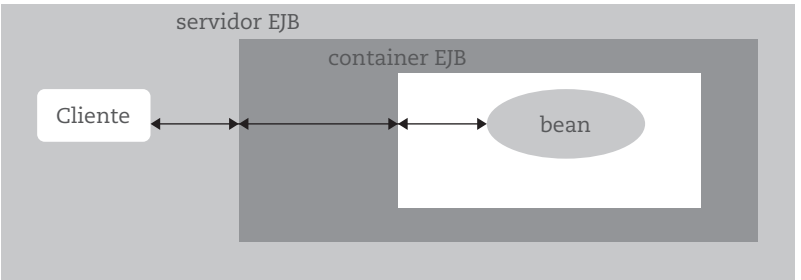



**Figura 57**. Interação entre as Entidades do EJB

Em virtude dos diferentes tipos de *beans*, existe a possibilidade de se ter diferentes cenários no desenvolvimento com componentes EJB, onde diferentes classes e interfaces devem se codificadas [Melo, 2005].

Um componente EJB possui três tipos fundamentais de *beans*: os de entidade; os de sessão; e os dirigido por mensagem.

Os *Beans* de Entidade (*Entity Beans*) modelam conceitos de negócio que podem ser expressos por nomes. Estes tipos de *beans* modelam realmente um dado no bando de dados onde uma instância representa uma linha na tabela do banco. Eles são responsáveis pelos dados de negócio (*Core Business Data*) [Melo, 2005].

No caso dos *Beans* de Sessão (*Session Beans*) são responsáveis por representar as ações de negócio. Para tanto, eles atendem mais o lado da aplicação cliente que necessita acionar um serviço. São divididos em dois tipos básicos: *stateless* e *stateful*. Um *bean stateless* é uma coleção de serviços cada qual sendo representado por um método específico. Já os *beans stateful* são uma extensão da aplicação cliente. Quando um cliente obtém uma conexão com o EJB *stateful*, o seu estado é mantido entre as chamadas do mesmo cliente até o mesmo remover esta conexão [Melo, 2005].

Por fim, os *Beans* Orientados a Mensagens (*Message-Driven Beans*) processam alguma lógica de negócios usando mensagens JMS - *Java Message Service* enviadas para uma destinação particular, ou seja, consomem mensagens JMS através da tecnologia EJB [Melo, 2005]. Diferente dos *beans* de sessão, os orientados a mensagens são completamente escondidos do cliente, não possuem interfaces *home* nem *remote*. O único meio dos clientes estabelecerem uma comunicação com os *beans* orientados a mensagens seria enviando uma mensagem para um destinatário JMS.

## 11.5. Estudo de Caso: Sistema de Alocação de Salas

A seguir será apresentado um exemplo da aplicação do desenvolvimento baseado em componentes para a construção de um sistema de alocação de salas. Esta situação real foi levantada por [Oliveira e de Paula, 2009] e mostra de forma detalhada várias etapas do processo baseado em componentes, culminando na implementação do sistema final.

Na Universidade de Brasília (UnB) havia um sério problema de gestão do laboratório de informática. Isso porque o processo de alocação do mesmo exige o preenchimento manual de formulários para a reserva do ambiente para os professores, bem como a requisição de instalação de softwares para o uso em aulas práticas.

A ideia de criar um sistema que automatize o processo de alocação de salas e laboratórios busca solucionar problemas de conflitos entre os requerentes na reserva de salas no mesmo horário e minimizar a espera pela instalação de software em determinada sala.

Para tanto, começou-se com a identificação dos principais atores neste sistema. Estes foram enquadrados em dois tipos:

- Os requerentes da sala;
- Os gestores do laboratório.

Os requerentes são qualquer tipo de usuário capaz de alocar salas no laboratório: professores, palestrantes externos, empresas Junior, etc. Os gestores do laboratório correspondem a três atores:

- Secretária;
- Bolsista;
- Coordenador.

A Secretária é responsável por receber todos os pedidos de alocação de sala e de instalação de software, realizando o primeiro filtro na avaliação dos pedidos. O Bolsista é incluído no processo quando a instalação de software é aceita. Ele provê manutenção do hardware e software nas máquinas do laboratório.

Já o Coordenador do laboratório acompanha todas as atividades que ocorrem no ambiente e pode emitir parecer sobre a ocorrência de alguma ilegalidade. Ele também é responsável máximo para solução de pedidos de maior dificuldade ou urgência.

A Figura 58 mostra uma representação hierárquica dos atores no nível de caso de uso.

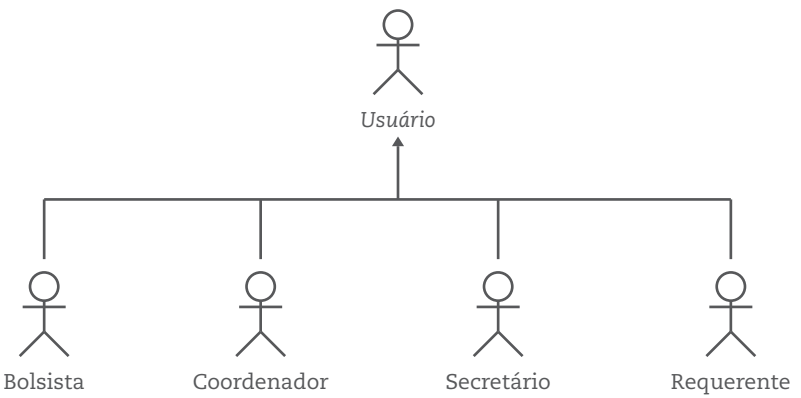



**Figura 58**. Hierarquia dos Atores do Sistema no Modelo de Caso de Uso

De forma a extinguir o processo manual de preenchimento de formulários e planilhas por meio dos requerentes, o sistema oferece um conjunto de recursos Web para a alocação de salas, sendo flexível para inserção de: novas salas, novos softwares e novos usuários. Sendo assim, proverá uma flexibilidade necessária para alocação de horários de aulas e gerando

o mínimo de transtorno para todos os usuários envolvidos neste sistema.

Algumas questões não funcionais que devem estar presentes no sistema são: permitir o mínimo de cinco usuários concorrentemente; a geração das páginas deverá ser no máximo de dez segundos; e o tempo de resposta médio dos serviços deverá ser de três segundos. Também será provido tratamento de erros, disponibilização de mensagens na execução das operações e o sistema deverá ser modular e extensível, sendo utilizada a linguagem Java para o seu desenvolvimento e o banco de dados MySQL.

### 11.5.1. Principais Funcionalidades

De forma a separar as responsabilidades do sistema, foram listadas as principais funcionalidades:

- Manter Requerentes: responsável pela manutenção dos requerentes na base do sistema;
- Manter Sala: responsável pela manutenção das salas, que disponibiliza primeiramente a inclusão das salas no laboratório por parte dos responsáveis pelo mesmo;
- Manter Software: operações voltadas ao Bolsista do laboratório. Nesta parte, o Bolsista pode ter o controle sobre todos os fluxos. Ele pode incluir informações de sistema operacional, descrição dos softwares, etc.;
- Solicitar Software: busca oferecer ao requerente uma forma de requisitar um software para ser instalado em uma determinada sala para ministrar uma aula ou palestra;
- Reservar Sala: esta poderia ser dita como a principal funcionalidade do sistema. O procedimento deverá acontecer com o Requerente escolhendo o horário e a sala que ele deseja reservar;

- Autenticar Usuário: regras internas de negócio para realizar a autenticação do usuário.

A Figura 59 mostra o diagrama de casos de uso para a solicitação de software.

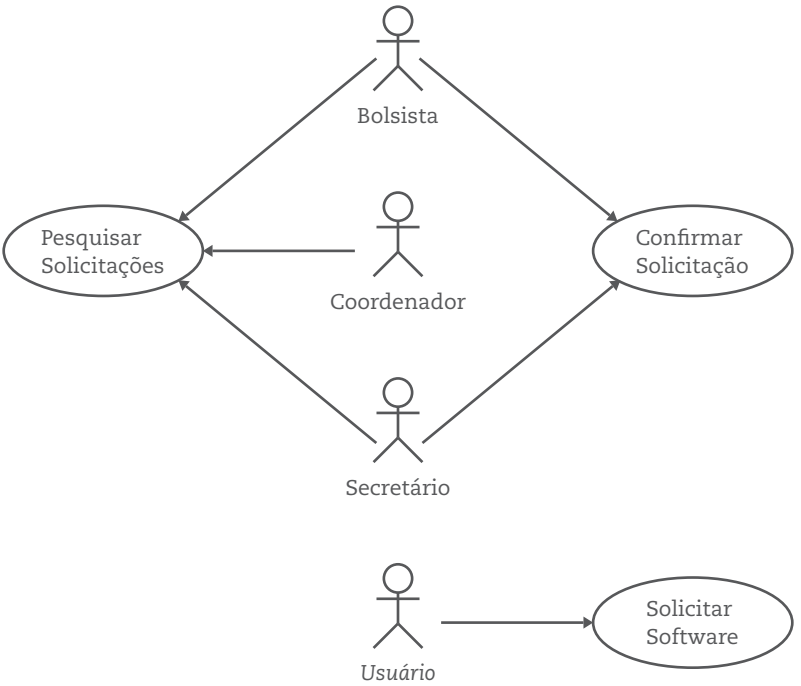



**Figura 59**. Diagrama de Casos de Uso para Solicitação de Software

Após a identificação das principais funcionalidades e da elaboração dos modelos de casos de uso, segue-se com os processos de modelagem conceitual, identificação das interfaces, modelagem de tipo de negócio e arquitetura de componentes.

### 11.5.2. Modelagem Conceitual

Para alcançar este objetivo, foram consideradas as funcionalidades providas e identificaram-se as seguintes entidades:

- Software;
- Sala;
- Usuário;
- Instalação de Software;
- Reserva de Sala;
- Horário.

A Figura 60 apresenta a modelagem conceitual do negócio. Tomou-se por base que a principal funcionalidade do sistema é a reserva de sala.

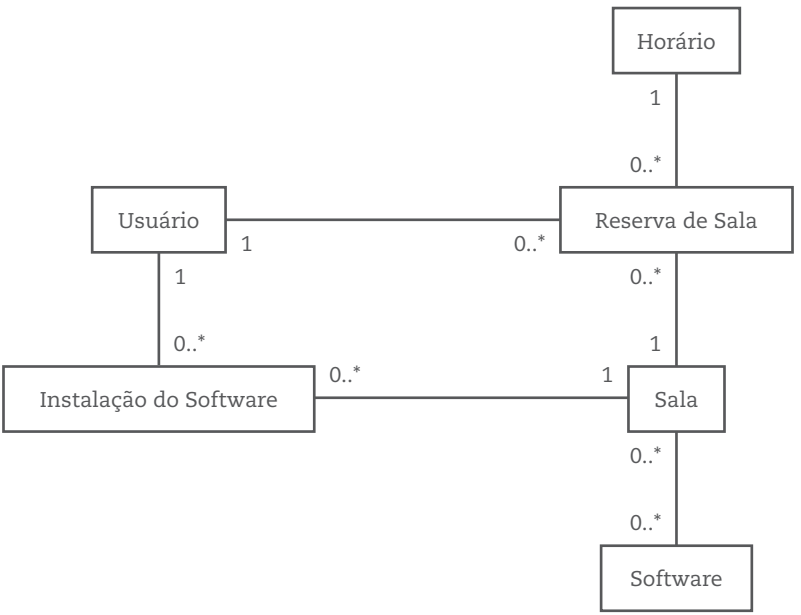



**Figura 60**. Modelagem Conceitual do Negócio

### 11.5.3. Identificação das Interfaces do Sistema

O padrão de nomenclatura para as interfaces dos componentes de negócio é o sufixo “Service”, já para as interfaces dos componentes de persistência é o sufixo “Dao”. Para ambos os casos, suas implementações devem ter o mesmo nome, seguido pelo sufixo “Impl”. Elas foram divididas em três tipos:

- Interfaces de negócio: responsáveis por prover os métodos que trabalham com dados de entrada e produzem resultados seguindo as regras de negócio do sistema;
- Interfaces de sistema: provêm do estudo de casos de uso focando como o sistema interagirá para manter informações sem envolver regras de negócio;
- Interfaces de persistência: visam fornecer aos componentes uma interface com o banco de dados para que o componente não perca a flexibilidade e portabilidade com uma possível mudança da camada de persistência.

A partir dessas informações identificaram-se as seguintes interfaces de negócio:

- ManterRequerenteService;
- ManterSalaService.

A Figura 61 mostra o mapeamento das funcionalidades especificadas nos requisitos para os métodos que a interface proverá, no exemplo apresenta-se a interface ManterRequerente.

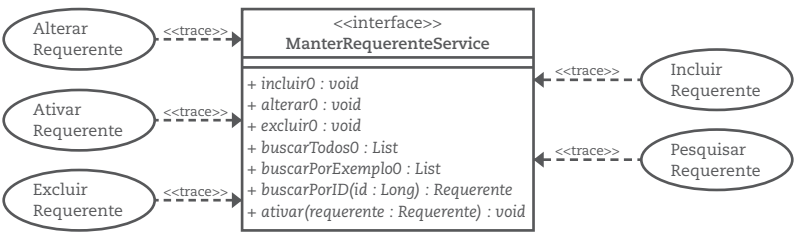



**Figura 61**. Mapeamento do ManterRequerente

No tocante às interfaces de sistema, foram identificadas as seguintes interfaces:

- AutenticarUsuarioService;
- ManterSoftwareService;
- ReservarSalaService;
- SolicitarSoftwareService.

A Figura 62 mostra o mapeamento das funcionalidades para os métodos que a interface ManterSoftwareService proverá.

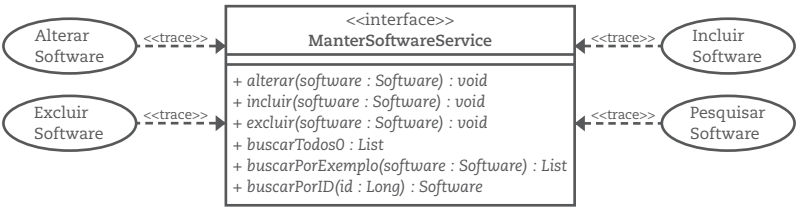



**Figura 62**. Mapeamento do ManterSoftware

Finalmente, para a camada de persistência foram identificadas as seguintes interfaces:

- AutenticarUsuarioDao;
- ManterRequerenteDao;
- MaterSalaDao;
- ManterSoftwareDao;
- ReservarSalaDao;
- SolicitarSoftwareDao.

Estas interfaces foram criadas para separar a camada de negócio da camada de persistência, visando maior flexibilidade às mudanças na lógica de persistência.

### 11.5.4. Modelagem de Tipo de Negócio

O modelo de tipo de negócio é o mapeamento do modelo de entidade da parte de análise para a parte de projeto. Neste modelo são definidos os atributos de cada classe e as classes que independem da existência de outras são consideradas como principais para o projeto, recebendo o estereótipo <<*core*>>.

A Figura 63 ilustra o modelo de tipo de negócio com os atributos de cada classe, suas dependências e cardinalidades.

### 11.5.5. Arquitetura de Componentes

Por fim, na construção da arquitetura de componentes deve-se mostrar como a interface de cada componente relacionar-se-á com outros componentes para formar o sistema.

Nos sistemas SAS[4] - *Statiscal Analysis System* os componentes identificados, além de implementarem as suas respectivas interfaces de negócio, deverão consumir serviços das interfaces de persistência para prover menos dependências com a parte de persistência dos dados.

A Figura 64 mostra a arquitetura dos componentes do sistema. Esta é a composição do sistema de alocação de salas, contendo os seus fluxos de manutenção e os processos de reserva de sala e instalação de software.

**4** O SAS é um sistema integrado de aplicações para a análise de dados, que consiste de: Recuperação de dados, Gerenciamento de arquivos, Análise estatística, Acesso a Banco de Dados, Geração de gráficos, Geração de relatórios. Trabalha com quatro ações básicas sobre o dado: Acessar, Manipular, Analisar e Apresentar.

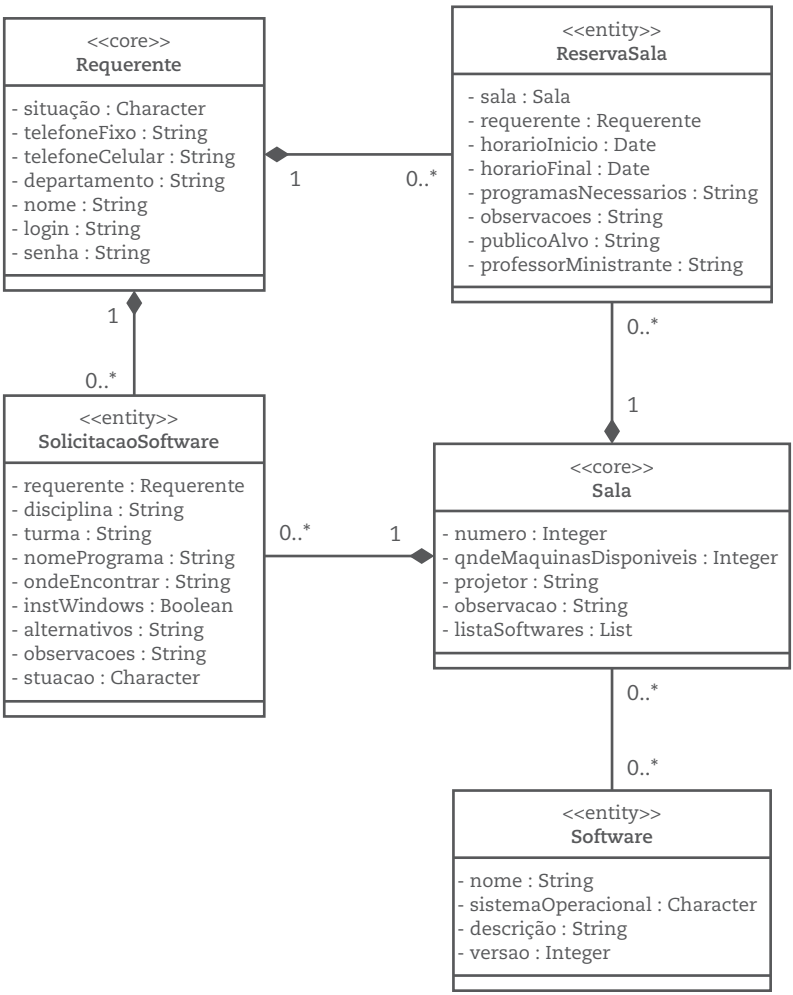



**Figura 63**. Modelagem de Tipo de Negócio

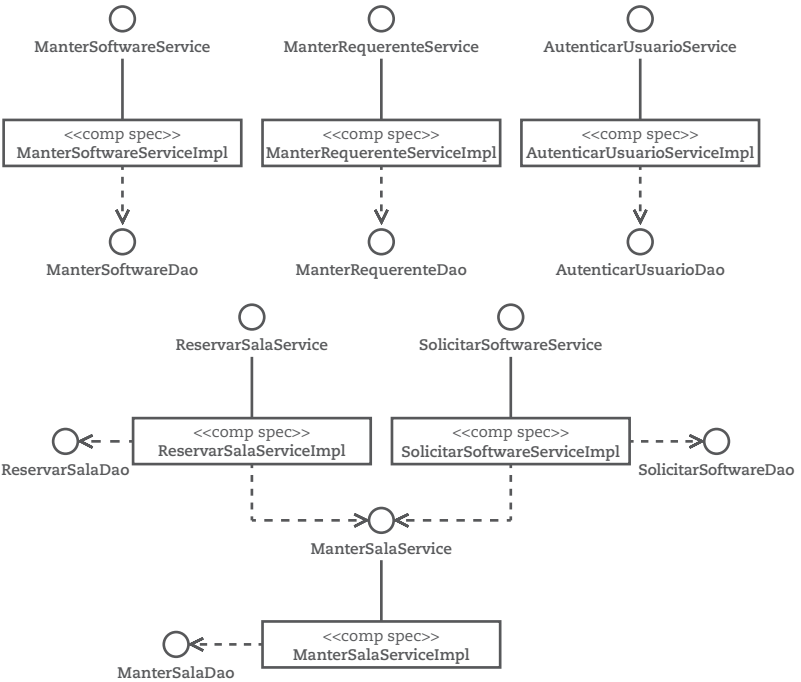



**Figura 64**. Modelo Arquitetural de Componentes

## 11.6. Considerações Finais

Tendo em vista a necessidade de desenvolver software de boa qualidade, para um mercado exigente, percebe-se a necessidade de utilizar algum método que garanta o controle do processo. O processo de software baseado em componente representa um conjunto de ferramentas poderosas para auxiliar as equipes de desenvolvimento, independente da metodologia utilizada, como o Catalysis. Os benefícios conquistados com a reusabilidade garantem, na maioria das vezes, prazos e custos reduzidos.

A adoção de processos dessa área, visando a produção de componentes com características como: encapsulamento, confiabilidade, portabilidade, desempenho e, principalmente, capacidade de reuso; é bastante promissora.

Uma receita ideal para a resolução de qualquer problema de desenvolvimento de software dificilmente será encontrada, a complexidade e inclusão de novos conceitos e possibilidades acontecem a cada dia. Resta aos técnicos buscar estas tecnologias e de posse deste conhecimento, aplicarem o método que melhor atenda os requisitos de qualidade e produtividade.

12

# TDD - DESENVOLVIMENTO ORIENTADO A TESTES

*Gustavo Henrique da Silva Alexandre*

Testes de software são elementos críticos da garantia de qualidade de software e representam uma revisão final da especificação, projeto e geração de código. Podem ser vistos como "o processo que visa a sua execução de forma controlada, com o objetivo de avaliar seu comportamento baseado no que foi especificado" [Pressman, 2010].

As falhas de software são grandes responsáveis por custos e tempo no processo de desenvolvimento de software. Embora não seja possível remover todos os erros existentes em certa aplicação, é possível reduzir consideravelmente o número dos mesmos utilizando uma infraestrutura de testes mais elaborada, que permita identificar e remover defeitos mais cedo e de forma mais eficaz [Borges, 2015]. Segundo o CHAOS Report [the Standish Group, 2013], estudo envolvendo milhares de projetos na área de Tecnologia da Informação, revela que apenas 37% dos projetos são bem sucedidos, enquanto que, 42% encontram-se em situação de risco e por fim, 21% dos projetos foram cancelados.

Um dos fatores que determinam a qualidade de um software são os comportamentos inesperados que acabam manifestando-se na forma de defeitos [Andrade *et al*., 2011]. As metodologias ágeis tratam o processo de incremento da qualidade como um exercício na eliminação do desperdício. Qualidade não pode ser alcançada através da avaliação de um produto já feito. O objetivo, portanto, é prevenir defeitos de qualidade ou deficiências em primeiro lugar, tornando os produtos avaliáveis através de medidas de garantia de qualidade [Lewis, 2004].

Segundo [Beck, 1999], baseado em exemplos, alguns dos riscos que estão ligados à qualidade de software são tratados através da utilização do TDD - *Test-Driven Development*, são eles: taxa de defeito e deterioração do sistema.

O TDD é o método ágil que pode ser implementado com o objetivo de alcançar a qualidade necessária na construção do código e na integração dos sistemas. Apesar do nome sugerir que TDD esteja relacionado apenas a testes, TDD não é sobre testes, é sobre como usar testes para criar software de maneira simples e incremental. Além de melhorar a qualidade e o projeto do software, ele também simplifica o processo de desenvolvimento [Andrade *et al*., 2011].

Com o objetivo de reduzir o tempo utilizado em testes a cada mudança realizada, e aumentar a qualidade, o TDD criado para antecipar e corrigir falhas durante o desenvolvimento do software [Borges, 2015]. Essa antecipação não garante a falta de erro no desenvolvimento, ela minimiza o impacto para área de testes.

No entanto, de acordo com Giorgi e Siqueira (2003), culturalmente os desenvolvedores têm resistência à adoção da prática TDD, pois eles precisam produzir códigos extras e, com isso, acreditam que será perda de tempo. Contudo,

desenvolvedores mais experientes entendem que a médio e longo prazo ter-se-á um ganho na produtividade e na qualidade do produto. Desenvolvedores mais maduros obtêm mais benefícios de TDD, escrevendo classes mais simples. Além disso, desenvolvedores maduros que experimentam a prática tendem a optar por TDD mais do que desenvolvedores menos experientes.

Alguns dos métodos do XP fornecem um conjunto bastante rigoroso de práticas no processo de construção da aplicação que visam fornecer o produto com qualidade suficiente e com o objetivo de antecipar defeitos encontrados no momento dos testes pelo próprio programador, reduzindo, assim, o tempo alocado, sendo o cliente responsável apenas pelos testes de aceitação. O TDD é uma prática sugerida pelo XP [Beck, 2002], onde a mecânica é baseada em um pequeno ciclo, onde o desenvolvedor escreve o teste antes de implementar a funcionalidade esperada e, em seguida, com o código apto ao teste recém-criado, refatora o código para remover possíveis duplicidades.

Introduzir o TDD em um ambiente onde não há um processo definido pode trazer benefícios suficientes para evidenciar o ganho no uso de um processo de desenvolvimento ágil de software. Avaliar se o projeto está apto para o uso dessa prática pode ser o ponto de partida na adoção de um processo de desenvolvimento ágil.

## 12.1. *Test-Driven Development*

O TDD é um método que usa testes para dar suporte no projeto de software durante a sua implementação e força o desenvolvimento acontecer de forma incremental. Prega o desenvolvimento do teste antes da codificação e orienta que,

durante a implementação, o código deve ser escrito apenas com o objetivo de fazer algum teste resultar em sucesso [Koskela, 2007].

O TDD também é chamado de TFD - *Test-First Development*, mas não é uma técnica de teste, é uma técnica de projeto. É contrário ao modo tradicional de desenvolvimento de software, onde se inicia criando um projeto que irá nortear a implementação. No TDD inicia-se criando um teste que determina como uma parte do sistema deve funcionar. Escrever os testes primeiro auxilia a entender o comportamento do sistema, e é o contrário da prática tradicional de se fazer um projeto detalhado antes de se iniciar o desenvolvimento. Adeptos de metodologias ágeis criticam essa forma de projetar software por ser pouco suscetível a mudanças e por resultar em projetos desnecessariamente grandes ou complicados.

Esta metodologia de "testar primeiro" do TDD é bem mais antiga que o termo TDD. Os primeiros registros indicando o uso desta técnica são da década de 60, foi utilizado pela NASA [Larman e Basili, 2003]. O TDD é um método que ganhou destaque com o surgimento de metodologias ágeis, como uma das práticas XP, e cada dia mais vem ganhando corpo à medida que mais e mais artigos são escritos especificamente sobre o assunto.

A redução do número de defeitos que o uso de TDD traz é fundamental para que se tenha um maior retorno de investimento (ROI - *Return On Investment*). O custo da correção de defeitos cresce de forma exponencial com o passar do tempo. A correção de um erro na fase de especificação pode ter valor inferior a 1% do custo que teria esta correção sendo feita durante a fase de implantação [Viega e McManus, 2000].

Apesar dos resultados serem animadores, TDD pode ser difícil de ser implantado por equipes de desenvolvimento e

entender como utilizá-lo de forma adequada é fundamental para se obter um maior retorno de investimento [Dubinsky e Hazzan, 2007].

Para usar TDD de forma correta é necessário implementar testes de uma maneira diferente. Um ponto relevante, que é preciso compreender para o uso do TDD de forma carreta, é a forma de escrever testes efetivos que possam ser usados para guiar o desenvolvimento do software. A maior parte dos desenvolvedores sabe como fazer testes para testar código que já está escrito, mas escrever testes antes do código pode necessitar uma abordagem diferente. Para escrever testes que guiem o desenvolvimento do código, é preciso se concentrar em como testar a funcionalidade de um código ao invés de sua implementação [Gold *et al.*, 2004].

Essa maneira diferente de escrever testes é tão importante que existem referências que usam uma nomenclatura diferenciada para os testes de unidade feitos com o uso do TDD: testes do programador [Astels, 2003]. Testes do programador são os testes implementados objetivando guiar o desenvolvimento do software, enquanto testes de unidade têm por objetivo testar a implementação já codificada de uma unidade. São chamados de teste do programador em oposição aos testes do cliente, que são testes realizados pelo cliente para verificar o comportamento do sistema do ponto de vista do usuário.

TDD é definido como um método de desenvolvimento onde [Astels, 2003]:

- Código somente entra em produção se houver um teste associado a ele;
- Mantém-se um conjunto exaustivo de testes do programador;
- Testes são escritos antes da implementação;
- Testes determinam o código que será escrito.

## 12.2. Ciclo do TDD

O desenvolvimento em TDD é constituído de etapas. Cada iteração corresponde à implementação de uma funcionalidade e é constituída dos seguintes passos [Beck, 2002], como pode ser vista na Figura 65:

- Escrever um teste para a funcionalidade: testes devem ser escritos, independentes de outros testes, de uma forma que não dependam de outros testes para passarem. Desta forma, um teste falho não implicará na falha de outros testes e sua execução não afetará o teste posterior;
- Garantir que esse teste falhe: o teste recém escrito deve falhar, uma vez que, não há código desenvolvido que implemente o comportamento esperado. Se o teste falhar indica que ele é válido léxica e sintaticamente, e que ele não irá passar sempre, ou seja, ele tem valor e realmente está testando alguma funcionalidade do sistema [Santos, 2010];
- Implementar o código, apenas o necessário para fazer o teste passar: código deve ser escrito com um único objetivo, de fazer o teste passar. Não se deve escrever código que não seja por esse motivo, ou seja, o desenvolvedor não deve se preocupar nesse momento com a qualidade código como, por exemplo, legibilidade e modularidade;
- Executar novamente o teste: uma vez implementado o código, todos os testes devem ser executados. Isso garante não só que o código escrito atende aos critérios do teste, mas que também ele não introduz nenhuma falha no sistema fazendo falhar testes que antes passavam. Em geral, *frameworks* de teste de unidade são usados para auxiliar na criação de testes automatizados;

- Refatorar o código removendo as duplicidades: refatoração é o processo de mudar o código com o objetivo único de melhorar sua estrutura interna, sem mudar seu funcionamento. Refatoração é basicamente melhoria de mau código. A maioria dos desenvolvedores refatora código diariamente usando o bom senso, contudo um catálogo de métodos de refatoração foi definido e explicado em [Fowler *et al.*, 1999]. Como refatoração é uma parte importante de TDD, é preciso desenvolver um entendimento desses métodos para que se possa rapidamente identificar padrões de mau código e os refatorar [Gold *et al.*, 2004].

Primeiro escreve-se o teste que falhe, depois o código é desenvolvido até que o teste seja executado com sucesso e, após estas etapas, deve-se refatorar o código. Quando então o projeto de software vai ganhando corpo, observa-se que esse mantra de "teste-codifique-refatore" é o oposto do tradicional "projete-codifique-teste" [Koskela, 2007].

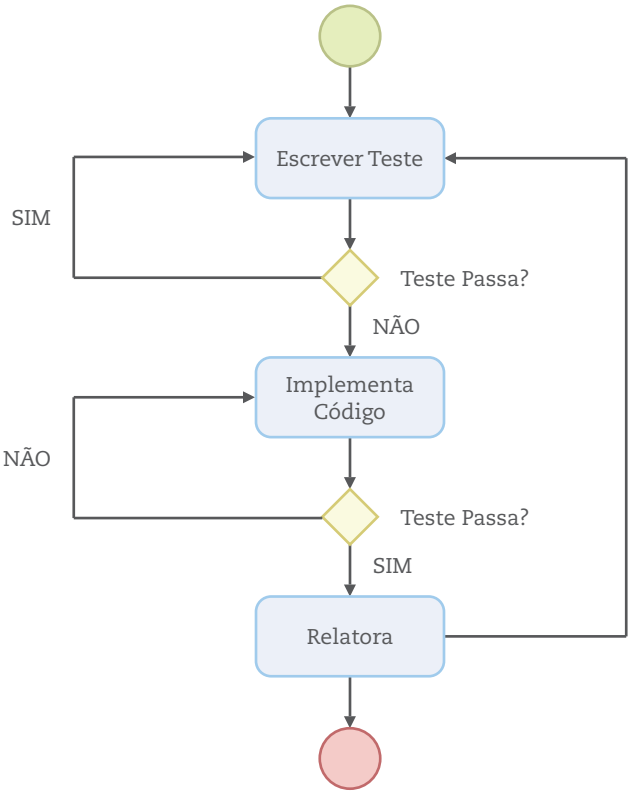



**Figura 65**. Ciclo de Vida do TDD

Uma variação do "teste – codifique - refatore" é o "vermelho – verde - refatore" devido ao uso comum de *frameworks* xUnit em TDD, que normalmente mostram uma barra verde para testes que passam e uma barra vermelha para testes que falham. De acordo com o ciclo TDD, quando o teste é escrito pela primeira vez ele deve falhar, aparecendo uma barra vermelha, e após o código ter sido implementado e estar de forma correta ele vai passar, mostrando uma barra verde. Uma vez que o teste esteja passando o código é refatorado.

As iterações devem ser pequenas. Com iterações pequenas o sistema nunca se afasta muito de um estado funcional no qual sua utilização não é prejudicada devido à quantidade de erros presentes. Pequenos incrementos também evitam projetos de software demasiadamente grandes, tornando o projeto de software algo contínuo e gradual, realizado a cada iteração. Isso permite tomar decisões de projeto baseadas em conhecimento adquirido durante o desenvolvimento, ao invés de premissas e previsões [Koskela, 2007].

Pequenos incrementos são proporcionais à facilidade no raciocínio do desenvolvedor, que tem que lidar com uma complexidade menor por se preocupar com pequenas partes do sistema e que envolvem poucas variáveis.

## 12.3. Variações do TDD

Desde seu surgimento, algumas variações do TDD foram criadas, entre elas *Acceptance Test-Driven Development* (ATDD), *Need-Driven Development* (NDD) e *Behavior-Driven Development* (BDD).

### 12.3.1. *Acceptance Test-Driven Development*

A forma tradicional de adicionar funcionalidades ao sistema é: primeiro escrever os documentos de requisito, proceder com a implementação e então o cliente realiza testes de aceitação. ATDD move a fase de testes para antes da implementação. É um processo similar ao TDD, apenas tratado em um nível diferente: enquanto TDD trata de comportamento de unidades, ATDD trata de comportamento de sistema [Koskela, 2007].

ATDD é a prática que estende o uso de TDD ao nível de testes de aceitação. É uma técnica orientada a testes, então um teste deve ser aplicado em primeiro lugar. Ela combina

a especificação de requisitos com a execução automática de testes desses requisitos [Koudelia, 2011]. Ainda segundo Koudelia (2011), ATDD possui três objetivos principais:

- Fornece meio de comunicação para melhorar a troca de informações entre todas as partes. ATDD não só tenta fazer com que o fluxo de informação trabalhe mais fácil entre esses grupos, mas também dentro dos grupos;
- Fornece instrumentos para armazenar documentação de software funcional que se mantém em toda fase de desenvolvimento conjunto;
- Implica no sistema em construção atender constantemente seus requisitos através de testes automáticos e permanece em boa forma através de *refactoring* constante.

### 12.3.2. *Need-Driven Development*

NDD é uma configuração particular de TDD que utiliza *mocks* no lugar das dependências da unidade, sendo testada e usa a estratégia de integração *top-down* [Freeman *et al*., 2004].

Seguindo essa técnica, primeiro trabalha-se nas unidades de mais alto nível, tomando-se uma delas para realização de testes de unidade. Unidades dependentes são identificadas à medida que novas funcionalidades são implementadas. Para essas unidades, devem ser criados *mocks* [Santos, 2010]. *Mocks* são objetos controlados dentro dos testes que simulam os objetos reais. [Andrade *et al*., 2011].

Essa forma de desenvolver torna mais fácil a delimitação de responsabilidade das unidades, uma vez que novas responsabilidades são externalizadas através de *mocks*, que posteriormente serão implementados [Freeman *et al*., 2004]. Faz o desenvolvedor pensar, em termos de interfaces. Como resultado, interfaces representam interações mais bem definidas entre os componentes.

### 12.3.3. *Behavior-Driven Development*

O BDD surgiu da necessidade de se mudar o foco que o TDD possui em testes. Como foi dito anteriormente, TDD não é sobre testes, é sobre projeto, embora a nomenclatura e a forma de trabalho com *frameworks* de teste de unidade remetam o desenvolvedor constantemente ao contexto de testes. Isso dificulta a assimilação dos conceitos de TDD e traz confusão no uso correto da técnica [North, 2006].

Em última instância, BDD é considerado um refinamento de TDD, com a principal diferença sendo a mudança do foco: de teste para comportamento. Em vez de escrever testes, o desenvolvedor passa a escrever especificações, descrevendo comportamentos que o sistema deve possuir. Essa mudança na linguagem muda a visão do desenvolvedor, fazendo-o perceber que ele está especificando comportamentos e não escrevendo testes.

## 12.4. Diferenças entre TDD e Testes Clássicos

No desenvolvimento tradicional, testes de unidade tendem a ser usados para validar funcionalidades de um único módulo do sistema. Por outro lado, no TDD, são usados para projetar-se um objeto que ainda não foi implementado nas primeiras fases do desenvolvimento. Escrevendo testes de unidade, desenvolvedores podem decidir as funcionalidades de uma classe e as interações dela com outras classes. Assim, os testes ajudam a descrever o sistema. O teste de unidade assume um papel no projeto do sistema [Kim *et al.*, 2006].

Realizar testes no desenvolvimento do software nem sempre é uma tarefa fácil. Sistemas em sua maioria necessitam de toda uma infraestrutura de amparo para realização de seu desenvolvimento, consequentemente, de seus testes, como

por exemplo: banco de dados, *web services*, serviços de terceiros, entre outros. Isso proporciona o aumento à complexidade e dificultando ainda mais a tarefa de testar. Testes podem simplesmente falhar, não por a funcionalidade desenvolvida estar com defeito e simplesmente porque uma dessas dependências está comprometida [Santos, 2010].

Criar objetos que simulem outros surgiu através da comunidade XP, onde uma de suas principais características é o desenvolvimento guiado por teste, onde um projeto de software é desenvolvido por meio de iteração guiado pela escrita de testes [Fowler, 2007].

Um dos desafios é desenvolver estratégias que isolem a complexidade do código e facilite a escrita dos testes. Uma das principais estratégias, oriundas do TDD, que vem cobrir essa dificuldade bem como remover as dependências no desenvolvimento e nos testes é o conceito de *Test Double*. O *Test Double* é um termo genérico para qualquer tipo de fingimento de objeto utilizado no lugar de um objeto real com o propósito de realizar testes [Meszaros, 2007]. Este mesmo autor definiu quatro tipos de *Test Double*:

- *Dummy*: objetos são passados, ao redor, mas nunca realmente utilizados. Normalmente utilizados para preencher listas;
- *Fake*: Objetos realmente têm implementação, mas usualmente tomam algum atalho fazendo com que não seja adequada à produção;
- *Stubs*: fornecem respostas fixas para chamadas feitas durante os testes. Geralmente não respondem a todas as chamadas, apenas as programadas para o teste;
- *Mocks*: são objetos pré-programados com o objetivo de preencher a especificação das chamadas que eles esperam receber.

## 12.5. Vantagens

As vantagens trazidas pelo TDD, além das melhorias de projeto, incluem:

- Um processo de desenvolvimento mais simples com o código testado de forma automática, pois existe uma base para testes de regressão. Os requisitos de software têm uma melhor compreensão, a cobertura do código tem um percentual alto e o nível de confiança dos desenvolvedores na aplicação também;
- O processo de desenvolvimento torna-se mais simples, pois após o desenvolvedor escrever um teste de unidade ele somente precisa focar em fazer o teste passar. Com isso o programador pode apenas concentrar seus esforços em uma parte do sistema. Desta forma, as decisões são adiadas e não necessitam serem tomadas previamente, quando ainda não há nada codificado e possivelmente não há informações suficientes [Gold *et al.*, 2004];
- O código produzido pode ser testado de forma automática em nível de unidades, pois o código em TDD é escrito apenas para fazer passar algum teste. O que não garante que será testável em níveis mais altos como testes de sistema [Maximilien e Williams, 2003];
- Durante o processo uma conjunto de testes de unidade são feitos. Esse conjunto de testes pode ser rodado após a adição de novas funcionalidades ao sistema para verificar se algum "*bug*" foi inserido, servindo como testes de regressão;
- Tem-se um melhor entendimento dos requisitos do software, pois, ao escrever os testes de unidade, uma boa compreensão de como a unidade deve se comportar é construída à medida que os critérios de sucesso vão sendo estabelecidos. Assim, um teste de unidade define bem quais comportamentos a unidade deve ter,

possuindo um nível de detalhamento mais concreto do que possivelmente um documento de requisitos [Gold *et al.*, 2004];

- Há uma diminuição da necessidade de depuradores. Seguir uma abordagem de testar primeiro em pequenas interações assegura que o uso de depuradores vai diminuir. É possível localizar com exatidão quais linhas adicionadas provocaram a falha do teste e é possível identificar à fonte do problema de forma quase imediata. Isso evita demoradas sessões de depuração, fato que ocorre com frequência entre desenvolvedores. Assim, é capaz de consertar defeitos antecipadamente, diminuindo o custo do projeto [Koskela, 2007]. O uso de depuradores coloca remendos no código e esses remendos e pequenas alterações podem ser até 40% mais propensos a erros do que desenvolver um novo código [Humphrey, 1989];
- Em razão do conceito de que somente deve-se implementar código para fazer com que algum teste passe, os testes tem uma tendência a adquirir um percentual muito elevado de cobertura [Astels, 2003]. Isso diminui o número de defeitos. Possivelmente o primeiro motivo para um defeito entrar em produção é não ter havido teste que fizesse uma verificação se um caminho de execução particular do código de fato funciona adequadamente. TDD corrige essa situação assegurando que praticamente não há código no sistema que não é necessário e consequentemente executado pelos testes [Koskela, 2007];
- O desenvolvedor ganha no seu código, uma vez que tem uma base de testes que se certificam do funcionamento esperado do código desenvolvido [Koskela, 2007]. Com uma maior confiança o desenvolvedor tem mais segurança em fazer modificações, o que melhora a estrutura

interna do sistema. Um sistema com uma estrutura melhor torna mais simples a tarefa de escrever testes.

## 12.6. Desvantagens

Requisitos tendem a sofrer mudanças, e nesse cenário, mudanças em requisitos requerem mudanças também em seus testes. Um requisito mal interpretado pode refletir em seus testes. Embora o teste esteja passando, o objetivo do requisito não foi atingido, ou seja, tanto o teste quanto o seu código estão errados. Com isso, o desenvolvedor pode ter a falsa sensação de que tudo esta funcionando. A técnica do TDD pode não ser aplicável a todos os domínios de aplicações.

A maior desvantagem na implantação de TDD está na mudança de cultura pela qual a equipe de desenvolvimento e a empresa têm de passar: é necessário o apoio gerencial, caso contrário o tempo necessário para escrever os códigos será considerado desperdício. Para um desenvolvedor já é difícil mudar sua mente em favor dos testes e TDD. Deverá haver esforço necessário para que um gerente o entenda [Andrade *et al.*, 2011].

Projetos nos quais atuem mais de uma empresa também têm o uso de TDD comprometido, principalmente porque o contato entre gestores, cliente e desenvolvedores torna-se complicado e demorado.

Quem cria os testes é o próprio desenvolvedor. Portanto, erros conceituais poderão continuar existindo na aplicação [Andrade *et al.*, 2011]. Correções desses erros também geram custos nas correções dos testes.

Por fim, os próprios testes tornam-se parte da manutenção do sistema.

## 12.7. Influência em Projeto de Software

Como em TDD o código apenas deve ser escrito a fim de fazer passar um teste que está falhando, a unidade é desenvolvida da maneira mais simples possível, sem tratar casos raros e complexos. Isso torna o código mais simples e legível. Estudos comprovam que o uso de TDD cria classes mais simples, com menos métodos, e métodos mais simples, com menor quantidade de linhas de código. Seu uso também diminui a complexidade do código [Janzen e Saiedian, 2005].

Seguir o método TDD força a focar somente em unidades que estão colaborando diretamente com a unidade que está sendo implementada, diminuindo assim o problema da complexidade de criação de *mocks* que pode ter redes de grande complexidade de dependências [Fowler, 2007].

Um objeto que necessita navegar uma rede de objetos na sua implementação é disposto a ser quebradiço por possuir muitas dependências. Um indicativo disso são testes com muita complexidade de se configurar e pouco legíveis, porque devem construir uma rede de *mock objects* similar. Testes de unidade têm um funcionamento melhor quando focam em um ponto de cada vez e apenas estabelecem expectativas para objetos que são vizinhos próximos [Freeman *et al.*, 2004].

Em estudos feitos pela IBM, após a adoção de TDD, foi observada uma diminuição de 50% na quantidade de defeitos encontrados em seus produtos. Em outros estudos feitos pela Microsoft foi identificado um aumento na qualidade do código, no entanto estes estudos mostram que com o uso do TDD gastou-se 15% a mais de tempo [Bhat e Nagappan, 2006].

Muller e Hagner (2002) conduziram um experimento estruturado comparando TDD com programação tradicional. Este experimento mediu a eficiência do TDD em termos do time

de desenvolvimento, qualidade do código resultante e nível de entendimento do mesmo. Dada a especificação e a declaração de alguns métodos, os times deveriam completar o corpo destes todos. O grupo que desenvolveu com TDD escreveu os casos de teste antes de começar a implementação. Já o grupo da programação tradicional escreveu os testes depois de completar o código.

A experiência ocorreu em duas fases, uma de implementação e outra de aceitação de teste. Ambos os grupos tiveram oportunidade de corrigir o código após a implementação. Apesar do tempo total de desenvolvimento dos métodos ter sido o mesmo para as duas equipes, o código do grupo TDD teve menos erros significantes quando reusado. Baseado nesse fato, os pesquisadores concluíram que a abordagem *test-first* aumenta substancialmente a qualidade do software e proporciona maior entendimento do código.

Outro experimento realizado por George e Williams (2004) mede a qualidade do software através do número de casos de teste caixa preta realizados com sucesso em ambas as abordagens de desenvolvimento. Todos os participantes implementam em duplas (*pair-programming*). São alocados de forma que cada dupla contenha um programador experiente na abordagem de desenvolvimento e outro iniciante. Apesar do tamanho do código desenvolvido ser relativamente pequeno (cerca de 200 linhas), e cada grupo ser formado por 6 duplas, os resultados foram bastante expressivos. A prática TDD forneceu código com qualidade superior quando comparada à praática de desenvolvimento tradicional. Além disso, programadores que praticam TDD codificaram mais rápido (16%) do que desenvolvedores que não a utilizaram. Este tempo foi medido em horas de trabalho considerando todo o processo de desenvolvimento.

## 12.8. Considerações Finais

Utilizar a técnica TDD para desenvolver software não é trivial. Seguir os passos corretos de maneira a obter os benefícios que esta técnica pode proporcionar exige disciplina, pois altera a maneira tradicional de desenvolver software. Esta mudança brusca no estilo de programar é o principal desafio no início da programação usando essa técnica.

Porém, seus benefícios são rapidamente percebidos. A cada passo durante a construção do código-fonte é possível visualizar e perceber o software assumindo um *design* enxuto e objetivo, com um código-fonte legível e fácil de entender. Dessa forma, a manuteniblidade do software é melhorada, tendo em vista que a fácil legibilidade do código-fonte ajuda na sua compreensão. O desenvolvimento com a técnica TDD pode proporcionar um código-fonte com maior nível abstração, fazendo com que trechos de código-fonte (componentes/módulos) com lógica mais complexa fiquem encapsulados e escondidos de trechos de código-fonte que os utilizam.

13

# KANBAN APLICADO AO DESENVOLVIMENTO DE SOFTWARE

*Jussara Adolfo Moreira*

As metodologias ágeis possuem métodos, práticas e técnicas que podem aumentar a satisfação do cliente [Boehm e Turner, 2003], além de produzir um sistema com maior qualidade e em menor tempo [Anderson, 2003]. Elas também podem prover maior comunicação entre os membros da equipe, melhorando o processo de desenvolvimento como um todo, possibilitando que as respostas aos requisitos e às mudanças sejam mais rápidas.

Assim, este capítulo apresenta o conhecimento obtido na experiência da aplicação do método Kanban como ferramenta para melhoria da gestão de processos de software, utilizando uma abordagem ágil baseada em Scrum, no contexto do IF-SERTÃO-PE - Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano. O relato dá-se por meio do detalhamento das atividades realizadas pelas equipes da disciplina Projeto de desenvolvimento de software do curso Licenciatura em Computação entre os semestres 2013.2 e 2015.1.

A experiência foi realizada no IF-SERTÃO-PE, no curso de Licenciatura em Computação, e foi motivada pela necessidade

de acompanhamento pela docente das atividades realizadas pelas equipes de desenvolvimento que devem elaborar software para um cliente interno da instituição.

Desta forma, com o intuito de buscar melhores soluções para o desenvolvimento de softwares, e facilitar o gerenciamento dos membros das equipes foi experimentado o uso do método Kanban [Anderson, 2010].

Assim, inicialmente é feita uma descrição do método Kanban, as principais características deste método para o desenvolvimento de software e discutida algumas diferenças entre o Kanban e o quadro de tarefas do Scrum. Por fim, são descritas algumas experiências com a utilização do método nas equipes formadas pelos alunos de Licenciatura em Computação, relatando os benefícios e dificuldades encontradas, além dos fatores de sucesso.

## 13.1. O Método Kanban

O método Kanban surgiu no Japão com o *Toyota Production System*, ou simplesmente TPS [Ohno, 1997], para controlar a fabricação de automóveis. O método tinha como objetivo produzir apenas o necessário de acordo com a demanda da fábrica. Foram usados cartões Kanban que eram acionados quando as peças precisavam ser repostas.

Diferente das indústrias americanas, os japoneses optaram por implementar um sistema de produção diferente, onde a demanda sinalizava através de cartões quando se deve produzir mais (*pull system*). Assim, o ritmo da produção é determinado pela demanda da fábrica, fazendo com que a indústria adapte sua velocidade de produção de acordo com o consumo dos clientes.

O Kanban é um termo japonês que significa sinal visual e traz como grande inovação o conceito de eliminar estoques,

os materiais e componentes agregados ao produto chegam no momento de sua produção e execução (*just in time*).

O uso do método em desenvolvimento de software registrou aumento em 2007 quando Rick Garber e David J. Anderson publicaram nas conferências "Lean New Product Development" e "Agile 2007" sobre os resultados obtidos no uso do método [Silva *et al.*, 2012]. Desde então, o uso deste método vem sendo estudado e experimentado por equipes de desenvolvimento.

Uma das grandes características deste método é evidenciar os problemas existentes no processo, pois é utilizado um painel de cartões Kanban e nestes cartões são colocadas todas as tarefas, possibilitando ter uma visão de todas as tarefas e a situação de cada uma. O quadro possui colunas que correspondem às fases, onde cada fase possui um número de cartões que representa a capacidade limite acordada em cada fase de um sistema que são colocadas em circulação. Cada cartão funciona como um mecanismo de sinalização e o sistema só permite iniciar uma nova tarefa quando um cartão está disponível. Esta regra é importante e precisa ser respeitada para que qualquer novo trabalho espere em uma fila até que um cartão torne-se disponível.

## 13.2. Kanban no Desenvolvimento de Software

O método Kanban para desenvolvimento de software e processos ágeis tem como ênfase não sobrecarregar os membros que compõe a equipe de criação do produto. Por isso, o método contem princípios básicos como [Silva *et al.*, 2012]: a equipe ou membro deve iniciar uma nova tarefa apenas quando é capaz de realizá-la; a equipe deve aceitar mudanças incrementais e evolutivas estimuladas pelo método Kanban e respeitar os atuais processos, papéis e responsabilidades.

Kanban, basicamente, tem como principal objetivo transformar o trabalho em andamento visível para toda equipe através de um painel, criar um sinal visual que indica que o novo trabalho pode ou não ser iniciado e estabelecer um limite para cada fase e acordar para que cada fase respeite essa premissa. Para que seja um painel Kanban é necessário que as atividades em andamento sejam limitadas em cada fase. Um novo item só pode ser iniciado quando o item em andamento é finalizado.

O Kanban não é uma metodologia ou um processo, não descreve papéis e fases para serem seguidos. Pode-se dizer que o Kanban é uma abordagem para mudança gerencial do projeto, um conceito para introduzir alterações em um ciclo de desenvolvimento de software ou gerenciamento de projetos. A equipe pode adaptar o processo, papéis e ferramentas e utilizar o método para gerenciar as atividades e controlar a produtividade.

Os métodos ágeis fornecem transparência sobre as atividades em andamento e concluídas, e reportam métricas com velocidade. O Kanban dá transparência ao processo e seu fluxo, expondo gargalos, filas, e desperdícios. Restringe o progresso do trabalho de modo que a equipe não se sobrecarregue. Portanto, tudo que impacta no desempenho da equipe de produção e para entrega de valor, fica explícito no modelo Kanban.

Dos métodos para desenvolvimento de software, o Kanban é o menos prescritivo, característica essa que estimula ainda mais as equipes a adotarem esse método. Segundo Kniberg (2009) o Kanban tem apenas três prescrições:

- Visualize o fluxo de trabalho atual;
- Limite o fluxo de trabalho;
- Acompanhe e gerencie o fluxo de trabalho.

O Kanban acaba tornando-se muito adaptativo por ser um método pouco prescritivo. As equipes que adotam o método devem buscar locais de melhoria, observando o processo

aplicado e realizar adaptações para que o método aconteça de forma satisfatória.

A primeira prescrição do Kanban ressalta a importância de visualizar o fluxo de trabalho. As atividades a serem fixadas no painel devem ser aquelas que de fato ocorrem, e não o que é formalmente definido. Desta forma, o painel vai retratar a realidade da equipe.

A segunda prescrição orienta limitar o fluxo de trabalho, essa limitação também é chamada de WIP - *Work in Progress*. É necessário explicitar quantos itens de trabalho devem estar em cada uma das fases do processo (*pull system*). Cada coluna do painel representa uma fase do processo, e cada fase deve possuir uma limitação de itens de trabalho para evitar gargalos no processo e focar em mais qualidade para as soluções. O WIP pode ser definido de acordo com a velocidade do time, ou ainda de acordo com o número de pessoas que trabalham na fase. Apenas quando um item sair de uma fase é que esta fase poderá receber outro item [Kniberg, 2009].

Na terceira prescrição, é definida uma forma de medir e controlar o fluxo de trabalho. Os times realizam fortes adaptações para que, de acordo com os problemas evidenciados, sejam propostas formas de controlar e contornar cada dificuldade. O Kanban controla as entradas de itens de trabalho e a vazão que é dada de acordo com o WIP definido. A esta vazão dos itens de trabalho dá-se o nome de *leadtime*, que representa o tempo de um item de trabalho desde a sua entrada no fluxo de trabalho mapeado até a sua saída.

De acordo com Kniberg (2009), aumentar o número de atividades em andamento leva diretamente ao aumento do tempo de espera. Por isso, faz-se necessário medir e controlar o número de cartões em uma coluna (WIP) ao longo do tempo. Após entender a proposta de um Sistema Kanban, fica fácil

perceber que o uso deste limita o trabalho em andamento para uma capacidade suportada pela equipe.

Após entender a proposta do método Kanban, torna-se simples perceber que o método retrata o uso de um sistema que prepara e limita o trabalho em andamento para uma capacidade suportada pela equipe. O método proporciona o equilíbrio da demanda de uma equipe, controlando o seu rendimento e acelerando sua produção. É simples perceber que todas as pessoas produzem mais quando conseguem chegar a um equilíbrio na vida.

O foco do Kanban é conduzir mudanças evolucionárias, e estes passos simples tem-se provado extremamente úteis para esse objetivo. Alguns autores referem-se ao Kanban como "Sistema Puxado Kanban", pois ao visualizar o fluxo e estabelecer os limites de WIP, garante-se que nunca se pode introduzir mais trabalho no sistema que a capacidade do sistema de processar esse trabalho. Está disponível apenas uma quantidade limitada de permissões de trabalho. Então, é necessário que se complete o trabalho existente antes que um novo trabalho possa ser iniciado.

O conceito de limitar o que deve ser feito é aplicado em todas as colunas do quadro. Essa é uma maneira rápida de reduzir o *leadtime*. No exemplo da Figura 66 itens de *backlog* compõem as atividades do Kanban.

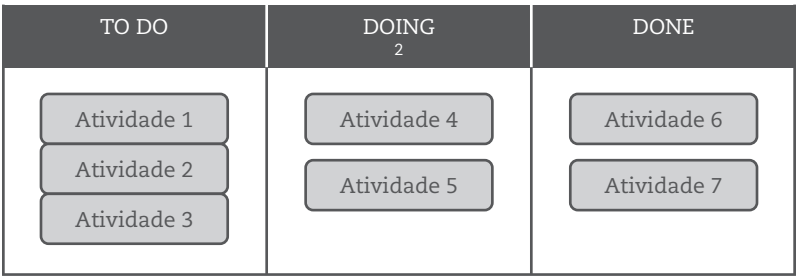



**Figura 66**. Painel Kanban

Para o desenvolvimento de software, é comum o uso de um sistema Kanban digital. Mas, pode-se manter o conceito de painel físico e digital, isso é reconhecido como boa prática, uma vez que ele mantém o princípio de sinalização visual.

Uma das ferramentas que podem ser utilizadas é o Trello [Silva *et al.*, 2012], ferramenta colaborativa para gerenciamento de projetos, que organiza as tarefas e eventos de forma funcional e que pode ser ajustada conforme a necessidade do projeto. A Figura 67 mostra a ferramenta sendo utilizada em um projeto. A ferramenta usa paradigma do Kanban e foi desenvolvida pela empresa Fog Creek [Silva *et al.*, 2014].

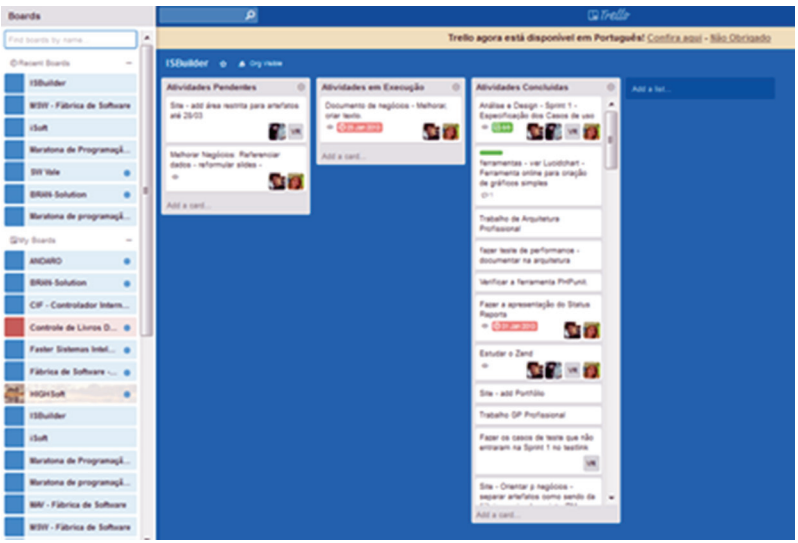



**Figura 67**. Ferramenta Trello

## 13.3. Kanban e o Quadro de Tarefas do Scrum

O Kanban baseia-se em um processo onde o fluxo de trabalho é contínuo, ou seja, diferente de Scrum, não há um ciclo de trabalho definido onde, conhecida uma atividade, a mesma é estimada e colocada neste ciclo.

O Kanban controla as entradas de itens de trabalho e a vazão que é dada de acordo com o WIP definido. O quadro de tarefas do Kanban não possui regras com relação à vazão. O WIP determina os limites de cada fase.

Para ser um sistema Kanban é necessário existir a ideia de puxar tarefas, conforme o limite acordado em cada fase. É necessário aplicar as três etapas cruciais que são:

- Criar o painel de visualização;
- Limitar os processos WIP; e
- Gerenciar o *leadtime*, aplicando o conceito de puxar uma nova tarefa quando um cartão está disponível.

Tanto no quadro de tarefas do Scrum quanto no Kanban as falhas tornam-se visíveis em tempo real. Ambos fornecem uma evolução gradual do processo e trazem como benefícios encontrar "gargalos". O método visual faz com que as pessoas passem a colaborar ainda mais durante o processo. Apenas o Kanban controla o *leadtime* e limita os processos WIP. Tanto o quadro de tarefas quanto o Kanban exibem as atividades a serem realizadas.

### 13.3.1. Calculando o WIP

Construir um consenso em torno da necessidade de equilibrar a demanda contra a capacidade da equipe é crucial. Para isso, é preciso resolver problemas como a disfunção entre os papéis e as responsabilidades dos membros da equipe, de modo que a equipe compreenda bem as atividades e responsabilidades de cada integrante.

Essa etapa implica definir a taxa em que a equipe aceita novos requisitos no processo de desenvolvimento de software para corresponder com a capacidade em que a equipe pode entregar código de qualidade.

Para definir o *Work In Progress* pode ser realizado um estudo com relação à capacidade da equipe em desenvolver soluções, ou ainda poder ser definido um WIP para cada integrante da equipe alocado na fase do processo.

## 13.4. Utilização do Método Kanban

No IF-SERTÃO-PE, no curso de Licenciatura em Computação, na disciplina Projeto de desenvolvimento de software, a docente foi motivada pela necessidade do acompanhamento das atividades realizadas pelas equipes que devem elaborar software para um cliente interno da instituição. Com o uso do Trello foi possível perceber o andamento de cada equipe nas atividades de desenvolvimento de software. A ferramenta foi utilizada para ser um painel *online* das atividades de cada projeto.

As equipes devem criar um site para demostrar o processo das equipes, métodos, técnicas e ferramentas utilizadas, possibilitando que de forma visual, todos os integrantes saibam as atividades que estão sendo realizadas e que possam ter controle das ações realizadas por cada um. O processo utilizado por cada equipe foi escolhido pelos integrantes, que em maioria seguem Scrum ou Kanban.

A Figura 68 exibe o painel da equipe HighSoft que optou por utilizar o método Kanban. A equipe é formada por quatro integrantes, que inicialmente optaram por trabalhar em duplas para equilibrar os conhecimentos dos integrantes. O WIP foi calculado com base na velocidade do time que inicialmente era 2, deveria existir apenas um cartão com cada dupla por fase no processo de desenvolvimento, ou melhor nas fases *Doing* ou *Doing - hold*. Foram definidas cores para os cartões de modo a classificar as atividades a serem realizadas.

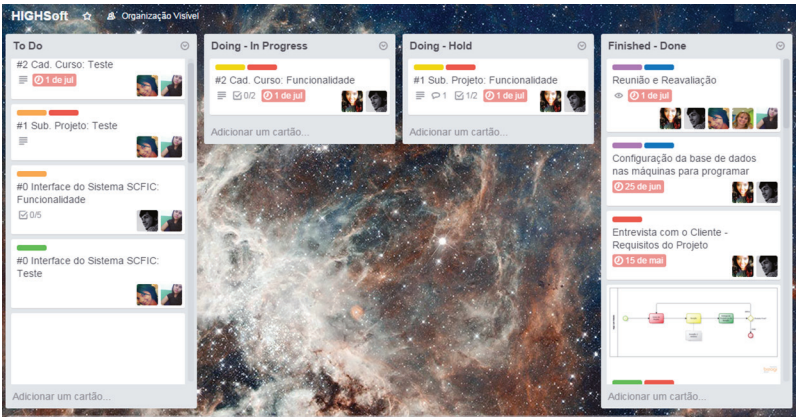



**Figura 68**. Painel equipe HighSoft

O Trello é uma ferramenta que permite classificar os cartões por cores, onde cada etiqueta pode ser inserida em um cartão Kanban. As cores e as categorias foram definidas pela equipe de desenvolvimento. A ferramenta permite ainda adicionar os membros responsáveis pelas cor de cada cartão, bem como definir um *checklist* de passos para conseguir concluir a atividade. A Figura 69 mostra um cartão Kanban no Trello, com a possibilidade de utilizar uma ou duas das etiquetas em visualização.

Cada cartão Kanban pode ainda ter detalhes da atividade, como adicionar um anexo, definir uma data para conclusão, ou ainda utilizar a opção de comentários para adicionar informações do cartão.


Figura 69. Cartão Kanban da Equipe HighSoft

### 13.4.1. Análise do Uso do Kanban

Durante o uso da ferramenta Trello, as equipes que se empenharam a realizar as atividades tiveram sucesso no uso da ferramenta e no resultado do projeto. O cálculo do sistema *just in time*, foi usado por apenas algumas equipes que optaram polo modelo Kanban.

Vários benefícios foram percebidos devido a seu uso. O Kanban permite visualização de todas as tarefas, bem como o controle de atividades de acordo com o fluxo do processo. As equipes poderiam definir datas para cada atividade de acordo com o cronograma inicial e alterá-las caso fosse necessário replanejar. Com isso o painel estaria sempre demonstrando a

situação atual do projeto de cada equipe, possibilitando ainda consultar a evolução do projeto.

Através do uso da ferramenta é possível perceber que o interesse aumenta quando os integrantes das equipes percebem as atividades no painel e a equipe alocada que realiza suas tarefas. As que possuem atrasos de acordo com o planejado, recebem cobranças para realizar a utilização da ferramenta Trello, e ajustar as atividades com o cronograma, atualizando todas as atividades relacionadas.

O Kanban atua fornecendo visibilidade nos processos, deixando explícito os problemas e prendendo o foco da equipe em qualidade. Reflete os defeitos, pontos de sobrecarga, custos econômicos sobre o fluxo de rendimento e a variabilidade.

A simples regra de limitar os trabalhos em andamento no sistema Kanban estimula maior qualidade e maior desempenho na execução de cada tarefa.

## 13.5. Considerações Finais

A dificuldade de desenvolver um software que seja utilizável tem sido amenizada com os novos conceitos e soluções propostas pelas metodologias ágeis. O Kanban, alvo desse capítulo, tem esse foco. Quando o cliente pode definir a prioridade das atividades e pode mudá-la a qualquer momento, sua satisfação é maior. Quando a equipe pode acompanhar o fluxo de todas as atividades do processo, podendo também controlar o trabalho que está em progresso, esta trabalha de forma mais produtiva e com qualidade. Objetivos são traçados e a simplicidade torna-se lei no desenvolvimento do projeto, o que traz como retorno grandes facilidades para todos. Além disso, o Kanban pode ser combinado com outros métodos, bem como pode ser adaptado de acordo com

a realidade da empresa. Não existe uma metodologia padrão a ser adotada em um projeto, existe uma que seja mais favorável a sua necessidade.

O sistema Kanban apresenta um conjunto de vantagens para os seus usuários, a saber:

- O Kanban é um sistema autocontrolado e extremamente simples de ser implementado;
- O Kanban elimina a necessidade de controles por meio de documentos formais, ele contribui para a desburocratização;
- O Kanban valoriza o colaborador, fazendo com que ele possa contribuir com sua experiência para o sucesso do sistema;
- O Kanban é um processo controlado pela produção;
- O Kanban limita e permite reduzir os estoques;
- O Kanban reduz os custos de fabricação;
- O Kanban tem baixo custo de implantação.

14

# CONCLUSÕES

*Sandro Ronaldo Bezerra Oliveira*
*Alexandre Marcos Lins de Vasconcelos*

A tecnologia de processo de software é uma área da Engenharia de Software que vem atraindo cada vez mais atenção da comunidade científica e absorvendo recursos das indústrias envolvidas no desenvolvimento de software de alta complexidade. Dentro deste contexto, muitas teorias, formalismos e ferramentas estão sendo divulgados para apoiar a evolução do processo de desenvolvimento de software, objetivando melhorar o gerenciamento e a qualidade da produção de software de uma forma economicamente viável.

Assim sendo, este livro apresentou uma revisão bibliográfica sobre alguns tópicos relevantes da área de Qualidade, Gestão e Processos de Software. A conceituação básica de Engenharia de Software (envolvendo os seus princípios, métodos, metodologias, entre outros) aliada às principais técnicas de gerenciamento de projetos, qualidade e processos de software estabeleceram a base teórica necessária para a discussão dos principais problemas relacionados à definição, execução e melhoria do processo de software.

Apesar da quantidade de propostas que tem surgido para aperfeiçoar os mecanismos de implantação de processos, uma área que tem se destacado bastante como auxiliar nesta implantação de processos é a definição e a melhoria contínua de processos de software, a partir da adoção dos mais variados modelos e normas de qualidade do processo e do produto de software. Esta melhoria recai no fato de prover um aperfeiçoamento constante dos ativos que compõem um processo de software a partir do aprendizado e análise de experiências da execução de processos de software.

A necessidade da especificação e melhoria contínua de um processo de software aumenta a possibilidade de se atingir um processo de software mais bem definido e continuamente aperfeiçoado às necessidades de uma organização, tornando estas áreas bastante promissoras.

## 14.1. Principais Contribuições do Livro

A seguir são apresentadas algumas contribuições obtidas durante o desenvolvimento deste livro e que levam a uma reflexão maior acerca de opções de novas versões do mesmo:

- O livro oferece de uma forma geral uma abordagem sobre todas as características que compõem um processo de software, levando em consideração uma análise de todos os princípios norteadores para a definição de um processo de software, bem como a sua relação com a área de gerência de projetos;
- A partir da abordagem do item anterior, surgem aspectos relevantes no tocante à importância de modelos que melhor especificam a estrutura de definição de um processo de software a fim de propiciar melhoria contínua do mesmo. Dentro do foco de melhoria, observou-se a presença de um ciclo de vida capaz de abordar os aspectos-chave

para o aperfeiçoamento de um processo de software dado a coleta de experiências na sua execução e o planejamento estratégico desta mudança. Observou-se, também, a presença de uma grande quantidade de modelos/normas de qualidade para o processo e o produto de software a fim de possibilitar a sua definição, avaliação e melhoria a partir de padrões internacionais;

- A modelagem de processos de software é uma tarefa de extrema complexidade, pois se envolve com diferentes aspectos tecnológicos, psicológicos e sociais, exigindo, como consequência, muita experiência do projetista;
- As organizações enfrentam constantes mudanças para aumentar o controle, eficiência e agilidade dos seus processos de negócio. No entanto, essas mudanças muitas vezes não trazem os resultados esperados devido ao pouco conhecimento da organização sobre como seus processos de negócio são executados. Neste contexto, o uso de métodos e técnicas para a gestão e definição do processo pode ajudar a organização a compor seu processo de negócio.

# REFERÊNCIAS BIBLIOGRÁFICAS


Aalst, W. P. M. V. del *et al*., 2003, Business Process Management: A Survey, BPM 2003, LNCS 2678, 2003, Springer-Verlag Berlin Heidelberg, p. 1–12.

ABNT - Associação Brasileira de Normas Técnicas, 1994, NBR ISO 8402/1994 – Gestão da Qualidade e Garantia da Qualidade – Terminologia, Brasil.

ABPMP - Association of Business Process Management Professionals, 2009, Guide to the Business Process Management Body of Knowledge (BPM CBOK®), Edition 2009.

Addison, T., Vallabh, S., 2002, Controlling Software Project Risks – an Empirical Study of Methods used by Experienced Project Managers, Proceedings of SAICSIT 2002, p. 128 – 140, África do Sul.

Administradores.com, 2006, Ambiente "zen" faz empresa crescer 80%. Disponível em: *http://www.administradores.com.br/noticias/negocios /ambiente-zen-faz-empresa-crescer-80/7658/*. Último Acesso: Julho/2015.

Anderson, D. J., 2010, Kanban: Successful Evolutionary Change for Your Technology Business, Blue Hole Press.

Anderson D. J., 2003, Agile management for software engineering, applying the theory and constraints for business results, Prentice Hall, Upper Saddle River, NJ.

Andrade, B., Fonseca, J. V. B., Nascimento, M., Funghi, T., 2011. TDD-Test Driven Development - Desenvolvimento guiado por testes, Artigo Acadêmico.

Andrade, A., Ribeiro, A., Borges, E., Neves, W., 2004, Um estudo de aplicação de modelagem de processo de negócio para apoiar a especificação de requisitos de um sistema, VI Simpósio Internacional de Melhoria de Processos de Software, São Paulo.

Araújo, L. C. G., 2001, Organização, sistemas e métodos e as modernas ferramentas de gestão organizacional, Atlas, São Paulo.

Astels, D., 2003, Test-Driven Development: A Practical Guide (Coad Series), Prentice Hall PTR.

Bhat, T., Nagappan, N., 2006, Evaluating the ecacy of test-driven development: industrial case studies, In ISESE '06: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, pages 356-363, New York, NY, USA, ACM.

Baldam, R. L., Valle, R. A. B., Pereira, H. R. M., Hilst, S. M., Abreu, M. P., Sobral, V. S., 2007, Gerenciamento de Processos de Negócio, 1. Ed, Érica, São Paulo.

Basili, V. R., 1992, Software Modelling and Measurement: The Goal/Question/Metric Paradigm, Technical Report CS-TR-2956, Department of Computer Science, Universityof Maryland, MD 20742.

Beck, K., 1999, Extreme Programming Explained: Embrace Change, Ed. 1, Addison-Wesley Professional.

Beck, K., 2002, Test Driven Development: By Example, Ed. 1, Addison-Wesley Professional.

Bispo, P., 2006, Remédio para o absenteísmo. Disponível em: *http://www.rh.com.br/Portal/Desempenho/Materia/4524/remedio-para-o-absenteismo.html*, Último Acesso: Julho/2015.

Boehm, B., Turner, R., 2003, Balancing Agility and Discipline: A Guide for the Perplexed, Ed. 1, Addison-Wesley/Pearson Education.

Bon, J. V., 2012, Guia de Referência ITIL, Edição 2011, Campus, Rio de Janeiro.

Borges, E. N., 2015, Conceitos e Benefícios do Test Driven Development, Instituto de Informática – Universidade Federal do Rio Grande do Sul, Porto Alegre, Disponível em: *http://www.inf.ufrgs.br/~cesantin/TDD-Eduardo.pdf*, Último Acesso: Julho/2015.

Boria, J. L., Rubinstein, V. L. Rubinstein, A., 2013, A História de Tahini-Tahini: Melhoria de Processos de Software com Métodos Ágeis e Modelos MPS, PBQP Software, Brasil.

Braconi, J., Oliveira, S. B., 2009, Business Process Modeling Notation (BPMN) In: Valle, R., Oliveira, S. B. (org.) Análise e Modelagem de Processos de Negócio: Foco na Notação BPMN (Business Process Modeling Notation), Atlas, São Paulo, p. 77-93.

Britto, R. S., 2015, Uma introdução ao Scrum, Disponível em: *http://www.ufpi.br/subsiteFiles/pasn/arquivos/files/aula11_Scrum.pdf,* Último Acesso: Julho/2015.

Bueno, C., Campelo, G., 2010, Qualidade de Software, Universidade Federal de Pernambuco, Recife.

Capuano, E. A., 2008, Construtos para modelagem de organizações fundamentadas na informação e no conhecimento no serviço público brasileiro, Ciência da Informação, Brasília, v. 37, n. 3, p. 18-37.

Carvalho, A. L., 2006a, Foco é o cliente, Disponível em: *http://www4.serpro.gov.br/imprensa/publicacoes/tema-1/antigas%20tem as/tema_184/materias/o-foco-e-o-cliente,* Último Acesso: Julho/2015.

Carvalho, A. L., 2006b, Otimização nos Serviços, Disponível em: *http://www4.serpro.gov.br/imprensa/publicacoes/tema-1/antigas%20tem as/tema_184/materias/otimizacao-nos-servicos*, Último Acesso: Julho/2015.

Chang, J. F., 2006, Business Process Management Systems: Strategy and Implementation, Boca Raton: Auerbach Publications.

Charette, R., 1989, Software Engineering Risk Analysis and Management, McGraw-Hill (MultiScience Press), New York, NY, USA.

Chiavenato, I., 1993, Introdução à teoria geral da administração, 4. ed., Makron, São Paulo.

Chiavenato, I., 1994, Gerenciando Pessoas, 2. ed., Makron Books, São Paulo.

Chiavenato, I., 1999, Gestão de pessoas, Campus, Rio de Janeiro.

Chiavenato, I., 2000a, Administração: Teoria, Processo e Prática, MakronBooks, São Paulo.

Chiavenato, I., 2000b, Recursos humanos, 6. ed., Compacta, Atlas, São Paulo.

Chiavenato, I., 2004, Gestão de Pessoas: o novo papel dos recursos humanos nas organizações, Elsevier, Rio de Janeiro.

Chiavenato, I., 2008, Gestão de pessoas: o novo papel dos recursos humanos nas organizações, 3. ed., Elsevier/Campus, Rio de Janeiro.

CMMI Institute, 2010, Improving processes for providing better services - CMMI for Services, Versão 1.3.

Cockburn, A., 2004, Crystal Clear: A Human-Powered Methodology for Small Teams, Ed. 1, Addison-Wesley Professional.

Coriat, B., 1994, Pensar pelo Avesso, Editora UFRJ/REVAN, Rio de Janeiro.

Dailey, K. W., 2003, The Lean Manufacturing Pocket Handbook, DW Publishing.

Dal Moro, R., Falbo, R. A., 2008, Uma Ontologia para o Domínio de Qualidade de Software com Foco em Produtos e Processos de Software, 3rd Workshop on Ontologies and Metamodeling Software and Data Engineering – WOMSDE´2008, XXII Simpósio Brasileiro de Engenharia de Software – SBES´2008.

Dale, B. G., Elkjaer, M. B. F., van der Wiele, A., Williams, A. R. T., 2001, Fad, fashion and fit: An examination of quality circles, business process re-engineering and statistical process control, International Journal of Production Economics, 73(2), 137- 152.

Dankbaar, B., 1993, Economic Crisis and Institutional Change: the crisis of Fordism from the perspective of the automobile industry, Universitaire pers Maastricht.

Davenport, T., 1994, Reengenharia de Processos, Campus, Rio de Janeiro.

Davies, R., Sedley L., 2015, Agile Coaching, Disponível em: *http://www.it-ebooks.info*, Último Acesso: Julho/2015.

De la Vara, J. L., 2011, Business process-based requirements specification and object-oriented conceptual modeling of information systems, PhD Thesis, Universidad Politécnica de Valencia.

Demarco, T., 1986, Controlling Software Projects: Management, Measurement, and Estimates, 1 edition, Prentice Hall.

Drucker, Peter. F., 1975, Administração: tarefas, responsabilidades, práticas, Vol. 1, Pioneira, São Paulo.

Dubinsky, Y., Hazzan, O., 2007, Measured test-driven development: Using measures to monitor and control the unit development, Journal of Computer Science.

Eder, S., Conforto, E. C., Amaral, D. C., Silva, S. L., 2015, Diferenciando as abordagens tradicional e ágil de gerenciamento de projetos, Disponível em: *http://www.prod.org.br/doi/10.1590/S0103-65132014005000021*, Último Acesso: Julho/2015.

Eler, M. M., 2006, Um método para o desenvolvimento de software baseado em componentes e aspectos, Universidade de São Paulo (USP).

Eriksson, H. E., Penker, M., 2000, Business Modeling with UML: Business Patterns at Work, 1 ed., John Wiley & Sons, Inc..

Falbo, R. A., 1998, Integração de Conhecimento em um Ambiente de Desenvolvimento de Software, Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro.

Falbo, R. A., 2000, A Experiência na Definição de um Processo Padrão Baseado no Processo Unificado, II Simpósio Internacional de Melhoria de Processo de Software, São Paulo.

Fernandes, A. A., 1995, A Gerência de Software através de Métricas, São Paulo, Editora Atlas S/A.

Fernandes, J. M., Almeida, M., 2010, Classification and Comparison of Agile Methods, In: Seventh International Conference on the Quality of Information and Communications Technology - QUATIC, Oporto - Portugal.

Fernandes, A., Abreu, V., 2014, Implantando a Governança de TI, Brasport, São Paulo.

Ferrari E., 2011, Métricas e indicadores são mais que instrumentos estatísticos para uma organização, MONDO Strategies.

Fleury, A., 1994, Qualidade, produtividade e competitividade: abordagem comparativa entre França e Brasil, In: Revista de Administração,v.29, n.2, p.20-31, São Paulo.

Fischer, A. L., 2002, Um Resgate Conceitual e histórico dos modelos de gestão de pessoas, In: Fleury, M. T. (Org.). As Pessoas na Organização, Gente, Rio de Janeiro.

Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D., 1999, Refactoring: Improving the Design of Existing Code, Addison-Wesley Professional.

Fowler, M., 2004, UML Distilled: A Brief Guide to the Standard Object Modeling Language, Ed. 3, Addison-Wesley.

Fowler, M., 2007, Mocks aren't stubs, Disponível em: *http://martinfowler.com /articles/mocksArentStubs.html*, Último Acesso: Julho/2015.

FREEMAN et al., 2004. Freeman, S., Mackinnon, T., Pryce, N., and Walnes, J. (2004). Mock roles, not objects. In OOPSLA '04: Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, 236–246, New York, NY, USA. ACM.

Garvin, D., 1992, Operations strategy: text and cases, Englewood Cliffs, Prentice-Hall.

George, B., Williams, L., 2004, A structured experiment of test-driven development, Information and Software Technology, 46(5):337-342.

Gibson, D., Goldenson, D., Kost, K., 2006, Performance Results of CMMI-Based Process Improvement, Technical Report CMU/SEI-2006-TR-004.

Gold, R., Hammell, T., Snyder, T., 2004, Test-Driven Development: A J2EE Example (Expert's Voice), Apress.

Gomes, A. F., 2014, Agile: Desenvolvimento de Software com entregas frequentes e foco no valor de negócio, Casa do Código.

Gonçalves, , J. E. L., 2000, As empresas são grandes coleções de processos, RAE – Revista de Administração de Empresas, V. 40, n. 1, p. 6-19.

Graham, M., Lebaron, M., 1994, The horizontal revolution, Jossey-Bass, San Francisco.

Guerra, A. C., Colombo, R. M. T., 2009 Qualidade de Produto de Software, Ministério da Ciência, Tecnologia e Inovação, Secretaria de Política de Informática, Brasília, Brasil.

Guia Exame, 2014, Melhores Empresas para Trabalhar 2014, Editora Abril, São Pualo, Suplemento, Julho/2014.

Hackman, R., Wageman, R., 1995, Total quality management: empirical, conceptual and practical issues, Administrative Science Quarterly, June, p. 309-342;

Hammer, M., Champy, J., 1995, Reengenharia: o caminho para a mudança, 29. ed., Campus, Rio de Janeiro.

Harrington, H. J., 1991, Business process improvement, McGraw Hill, New York.

Havey, M., 2005, Essential Business Process Modeling, O'Reilly Media, Sebastopol.

Highsmith, J. A., 2002, Agile Software Development Ecosystems, Ed. 1, Addison-Wesley Professional.

Hirata, H. 1993, Sobre o "modelo" japonês, EDUSP, São Paulo.

Houy, C. *et al*., 2009, Empirical Research in Business Process Management –Analysis of an Emerging Feld of Research, Business Process Management Journal, v. 16 n. 4, p. 619-661.

Humphrey, W. S., 1989, Managing the Software Process, Addison-Wesley Professional.

IDC, 2007, IDC Predicts Rapid Growth for Business Process Management Software Market, Reaching $5.5 Billion by 2011.

the International Organization for Standardization and the International Electrotechnical Comission, *1998, Information Technology – Software Product Evaluation - Part 5: Process for Evaluators, Geneve: ISO*.

the International Organization for Standardization and the International Electrotechnical Comission, 1999a, Information Technology – Software /Product Evaluation - Part 1: General Overview, Geneve: ISO.

the International Organization for Standardization and the International Electrotechnical Comission, 1999b, Information Technology – Software Product Evaluation - Part 4: Process for Acquirers, Geneve: ISO.

the International Organization for Standardization and the International Electrotechnical Comission, 2000a, Information Technology – Software Product Evaluation - Part 2: Planning and Management, Geneve: ISO.

the International Organization for Standardization and the International Electrotechnical Comission, 2000b, Information Technology – Software Product Evaluation - Part 3: Process for Developers, Geneve: ISO.

the International Organization for Standardization and the International Electrotechnical Comission, 2001a, Information Technology – Software engineering – Product quality - Part 1: Quality model, Geneve: ISO.

the International Organization for Standardization and the International Electrotechnical Comission, 2001b, Information Technology – Software Product Evaluation - Part 6: Evaluation Modules, Geneve: ISO.

the International Organization for Standardization and the International Electrotechnical Comission, 2003a, ISO/IEC 15504-2: Information Technology - Process Assessment – Part 2 - Performing an Assessment, Geneve: ISO.

the International Organization for Standardization and the International Electrotechnical Comission, 2003b, Information Technology – Software engineering – Product quality - Part 2: External metrics, Geneve: ISO.

the International Organization for Standardization and the International Electrotechnical Comission, 2003c, Information Technology – Software engineering – Product quality - Part 3: Internal metrics, Geneve: ISO.

the International Organization for Standardization and the International Electrotechnical Comission, 2004, Information Technology – Software engineering – Product quality - Part 4: Quality in use metrics, Geneve: ISO.

the International Organization for Standardization and the International Electrotechnical Comission, 2005, ISO 9000: Quality management systems – Fundamentals and vocabulary, Geneve: ISO.

the International Organization for Standardization and the International Electrotechnical Comission, 2008, ISO/IEC 12207 Systems and Software Engineering – Software Life Cycle Processes, Geneve: ISO.

the International Organization for Standardization and the International Electrotechnical Comission, 2011, ISO/IEC 20000 Information Technology– Service Management, Geneve: ISO.

Isaksson, R., 2006, Total quality management for sustainable development: Process based system models, Business Process Management Journal, Vol. 12, Nº. 5, p. 632-645.

Ishikawa, K., 1990, Introduction to quality control, 3A Corporation, Tokyo.

Janzen, D., Saiedian, H., 2005, Test-Driven Development: Concepts, Taxonomy, and Future Direction. IEEE Computer, 38(9):43–50.

Jedd, M., 2007, BPM: Transforming the Organization, AIIM E-Doc Magazine, v. 21, n. 2, p. 25-29.

Jeston, J., Nelis, J., 2008, Business Process Management: Pratical Guidelines to Successful Implementations, 2 ed., Elsevier, Oxford.

Jones, C., 2000, Software Assessments, Benchmarks, and Best Practices, Addison-Wesley Longman Publishing Co., Inc.

Jung, J., Choi, I., Sung, M., 2007, An Integration Architecture for Knowledge Management Systems and Business Process Management Systems, Computers in Industry: An International, Application Oriented Research Journal, v. 58, p. 21–34.

Kaplan, R. S., Norton, D. P., 1997, A estratégia em ação, Campus, Rio de Janeiro.

Kaplan, R. S., Norton, D. P., 2001, Organização orientada para a estratégia: como as empresas que adotam o balanced score-card prosperam no novo ambiente de negócios, Campus, Rio de Janeiro.

Kim, K., Chang, D., 1995, Global quality management: a re-search focus, Decision Sciences, vol. 26, n. 5, p. 561- 568.

Kim, T., Park, C., Wu, C., 2006, Mock object models for test driven development, In SERA '06: Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications, pages 221–228, Washington, DC, USA, IEEE Computer Society.

Knapik, J., 2008, Gestão de Pessoas e Talentos, Ed. 2, Ibpex, Curitiba.

Kniberg, H., 2009, Kanban vs. Scrum: Making the most of both, Disponível em: *http://www.crisp.se/henrik.kniberg/Kanban-vs-Scrum.pdf*, Último Acesso: Julho/2015.

Kor, K. L. *et al.*, 2009, Business Process Management (BPM) Standards: A Survey, Business Process Management Journal, v. 15, n. 5, p. 744-791.

Korhonen, J., 2007, On the Lookout for Organizational Effectiveness – Requisite Control Structure in BPM Governance, 1st International Workshop on BPM Governance – WoGo'2007.

Koscianski, A., Soares, M. S., 2007, Qualidade de Software, Segunda Edição, Editora Novatec.

Koskela, L., 2007, Test Driven: TDD and Acceptance TDD for Java Developers, Manning Publications.

Koudelia, N., 2011, Acceptance Test-Driven Development, Dissertação de Mestrado, DEPARTMENT OF MATHEMATICAL INFORMATION TECHNOLOGY. UNIVERSITY OF JYVA SKYLA, JYVA SKYLA .

Kotler, P., 1993, Administração de marketing: análise, plaejamento, implementação e controle, Atlas, São Paulo.

Kruchten, P., 2000, The Rational Unified Process – an Introduction, Adisson-Wesley, ISBN: 0-201-60459-0.

Larman, C., Basili, V. R., 2003, Iterative and incremental development: A brief history, Computer, 36(6): 47-56.

Leite, L. O., Rezende, D. A., 2010, Modelo de gestão municipal baseado na utilização estratégica de recursos da tecnologia da informação para a gestão governamental: formatação do modelo e avaliação em um município, Rev. Adm. Pública, vol.44, n.2, p. 459-493.

Lewis, W. E., 2004, Software Testing and Continuous Quality Improvement, Ed. 2, Auerbach.

Lima, M. E. A., 1996, Programas de Qualidade Total e seus impactos na subjetividade, Anais Seminário Interinstitucional Qualidade da Produção, Produção dos Homens, EEUFMG, Belo Horizonte.

Lobato L. L., Machado, I. C., Silveira Neto, P., Bittar, T. J., Almeida, E., Meira, S., 2013, Risk Management in Software Product Lines: An Expert Opinion Survey, Proceedings of SBQS213 - Simpósio Brasileiro de Qualidade de Software, Brasil.

Loumos, V., Christonakis, G., Mpardis, G., Tziova, P., 2010, Change Management and Quality of Service through Business Process Modeling: The N-VIS, a Public Sector Project.

de Masi, D., 2001, O futuro do trabalho: fadiga e ócio na sociedade pós-industrial José Olympio, Rio de Janeiro.

Maximilien, E. M., Williams, L., 2003, Assessing test-driven development at ibm, In ICSE '03: Proceedings of the 25th International Conference on Software Engineering, pages 564–569, Washington, DC, USA. IEEE Computer Society.

Mears, P., 1993, How to stop talking about, and begin progress toward total quality management, In: Business Horizons, v.36, p.66-68, Greenwich.

Melão, N., Pidd M., 2000, A conceptual framework for understanding business processes and business process modeling, Information Systems Journal, v.10, p. 105–129.

de Mello, M. S., 2011, Melhoria de Processos de Software Multimodelos Baseada nos Modelos MPS e CMMI-DEV, Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro.

Melo, S. A., 2005, Uma abordagem sobre o panorama atual do desenvolvimento baseado em componentes, Universidade de Presidente Antônio Carlos (UNIPAC-MG).

Mendes, H. J. P. S., Oliveira, S. R. B., 2014, Um Estudo Baseado em Survey Sobre a Aplicação das Práticas de Gerência de Riscos em Projetos de Software no Cenário Brasileiro, Programa de Pós-Graduação em Ciência da Computação (PPGCC) – Instituto de Ciências Exatas e Naturais (ICEN) – Universidade Federal do Pará (UFPA).

Merriam, S. B., 2009, Qualitative Research – A Guide to Design and Implementation, Jossey-Bass.

Meszaros, G., 2007, Xunit Test Patterns: Refactoring Test Code, Addison-Wesley Professional, Reading.

Minghui, W., Jing, Y., Chunyan, Y., 2004, A methodology and its support environment for benchmark-based adaptable software process improvement, v. 6, pp. 5183-5188, The Hague, Netherlands.

Miranda, B. F., Lages, D. D., 2003, Uma comparação de dois métodos de desenvolvimento de software baseado em componentes: Catalysis e UML Components, Universidade Federal do Rio de Janeiro (UFRJ).

Muller, M. M., Hagner, O., 2002, Experiment about test-first programming, Empirical Assessment In Software Engineering EASE'02, Keele,.

Mutafelija, B., Stromberg, H., 2003, Systematic Process Improvement Using ISO 9001:2000 and CMMI, Artech House.

Nascimento, T., 2010, Avaliação da Qualidade de um Produto de Software, Monografia (Graduação), Universidade Federal de Pernambuco, Recife.

Niazi, M., Wilson, D., Zowghi, D., 2006, Critical success factors for software process improvement implementation: An empirical study, Software Process Improvement and Practice, v. 11, n. 2, pp. 193-211.

North, D., 2006, Introducing bdd, Disponível em: *http://dannorth.net/ introducing-bdd*, Último Acesso: Julho/2015.

Oakland, J., 1994, Gerenciamento da qualidade total, Nobel, São Paulo.

Odzaly, E. E, Greer, D., Sage, P., 2009, Software risk management barriers: an empirical study, In: ESEM '09: Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, IEEE Computer Society, Washington, DC, USA.

Ohno, T., 1997, O Sistema Toyota de Produção, Bookman, Porto Alegre.

Oliveira, S. R. B., 2007, ProDefiner: Uma Abordagem Progressiva para a Definição de Processos de Software no Contexto de um Ambiente Centrado no Processo, Tese de Doutorado sob orientação do Prof. Dr. Alexandre Marcos Lins de Vasconcelos, Centro de Informática, Universidade Federal de Pernambuco, Recife – Brasil.

Oliveira, F. C., de Paula, L. L., 2009, Engenharia de Software baseada em componentes: uma abordagem prática em ambientes web, Universidade de Brasília (UnB).

OMG - Object Management Group, 2009, Business Process Maturity Model (BPMM) Version 1.0, USA.

Overby, E., 2008, Process Virtualization Theory and the Impact of Information Technology, Organization Science, Vol. 19(2), p. 277-291.

Pádua, S. I. D., Cazarini, E. W., 2004, Modelagem organizacional, facilitador do desenvolvimento de sistemas de informação, ABEPRO.

Paladini, E. P., 2012, Gestão da Qualidade, Atlas, São Paulo.

Palmer, S. R., Felsing, J. M., 2002, A Practical Guide to Feature-Driven Development, Ed. 1, Prentice Hall.

Paulk, M. C., Weber, C. V., Curtis, B. Chrissis, M. B., 1994, The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley Professional.

Pfleeger, S. L., 2001, Software Engineering: theory and practice, 2nd edition, Prentice-Hall, Inc., ISBN 0-13-029049-1.

PMI - Project Management Institute, 2009, Guia PMBOK, Quarta Edição.

PMI - Project Management Institute, 2013, A Guide to the Project Management Body of Knowledge (PMBOK® Guide), Fifth Edition.

Porto Digital. Disponível em:< www.portodigital.org>. Acesso em 16/04/2015.

Pressman, R. S., 2010, Software Engineering: A Practioner's Approach, 7th edition, McGraw-Hill.

Rasmusson, J., 2015, The Agile Samurai – How Agile Masters Deliver Great Software, Disponível em: *http://www.wowebook.com*, Último Acesso: Julho/2015.

Reis, R. Q., Reis, C. A. L., 1995, Métricas para Estimativa de Custo de Software, Notas de Aula da Especialização em Análise de Sistemas, DI-UFPA.

Reis, R. L., 2008, Manual da gestão de stocks: teoria e prática, Editorial Presença, Lisboal.

Resende, D. K., Grego, J. B., Pimentel, N., Gonçalves, C. A., Junior, E. N. V., Ferreira, A. C., Kruel, F., Batista, P. R., Neto, O. C. T., Cavalcanti, W., Godinho, H., Montoni, M., Nunes, E., Barreto, A., Rocha, A. R., 2009, Implementação do MPS.BR Nível F e CMMI-DEV Nível 2 na Red & White IT Solutions, WAMPS – Worshop Anual do MPS, Campinas – SP.

Revere, L., Black, K., 2003, Integrating six sigma with total quality management: a case example for measuring medication errors, Journal of Healthcare Management, vol. 48, n. 6, p. 377-391.

Rezende, D. A., Abreu, A. F., 2001, Tecnologia da Informação Aplicada a Sistemas de Informação Empresariais: O papel estratégico da informação e dos Sistemas de Informação nas empresas, 2 ed., Atlas, São Paulo.

Ribeiro, M. C. F., 2009, Uma análise sobre os processos de trabalho do setor de projetos e obras de um Instituto de P&D em saúde no Brasil: um olhar sobre a informação como apoio à inovação em saúde, Dissertação de Mestrado, ARCA – Escola Nacional de Saúde Pública Sergio Aroucha, Fundação Oswaldo Cruz, Rio de Janeiro.

Rigby, D., 1998, What´s today special at the consultants´café?, Fortune, September, p. 162-163.

Ropponen, J., Lyytinen, K., 2000, Components of Software Development Risk: How to Address Them?, IEEE Transactions on Software Engineering, v. 26, p. 98-111.

Runeson, P., Höst, M., 2008, Guidelines for Conducting and Reporting Case Study Research in Software Engineering, Empirical Software Engineering, vol. 14, n. 2, p. 131-164.

Salerno, M. S., 1992, Reestruturação Industrial e Novos Padrões de Produção, In: São Paulo em Perspectiva. vol. 6, n. 3, p. 100-108.

Santos, R. L., 2010, Emprego de Test Driven Development no desenvolvimento de aplicações, Departamento de Ciência da Computação, Brasília.

Sato, D., 2015, Introdução à Programação Extrema (XP), Revista Engenharia de Software, Edição 10, Devmedia. Disponível em: *http://www.devmedia. com.br/artigo-engenharia-de-software-10-introducao-a-programacao-extrema-xp/11907#ixzz3esA0rPt3*. Último Acesso: Julho/2015

Schulmeyer, G., Mcmanus, J., 1999, The Handbook of Software Quality Assurance, Prentice Hall PTR, 3rd edition.

Schwaber, K., 2009, Scrum Guide, ScrumAlliance.

SEI - Software Engineering Institute, 2010 Capability Maturity Model Integration (CMMI) for Development, Version 1.3, Carnegie Mellon, USA.

Significados, 2015, Disponível em: *http://www.significados.com.br/gestao/*, Último Acesso: Julho/2015.

Silva, D. V. D. S., Santos, F. A. D. O., Neto, P. S., 2012, Os benefícios do uso de Kanban na gerência de projetos de manutenção de software, The Standish Group: CHAOS Summary for, VIII Simpósio Brasileiro de Sistemas de Informação (SBSI 2012), Trilhas Técnicas.

Silva, S. V., da Silva, L. B., Sales, M. S., Arantes, F, 2014, Uma Ferramenta para Gestão Integrada de Projetos, Congresso Brasileiro de Software: Teoria e Prática, Trilha da Indústria.

Siqueira, F. L., Giorgi, R. P., 2003, Qualidade de software: conceitos e paradigmas. Análise do Test-First, Disponível em: *http://www.levysiqueira.com.br/ artigos/test-first_2003.pdf*. Último Acesso: Julho/2015.

SOFTEX - Associação para Promoção da Excelência do Software Brasileiro, 2012a, MPS.BR - Guia Geral do MR-MPS-SW, versão 2012 – MPS.BR:2012, Brasil.

SOFTEX - Associação para Promoção da Excelência do Software Brasileiro, 2012b, Guia de Implementação – Parte 11: Implementação e Avaliação do MR-MPS-SW:2012 em Conjunto com o CMMI-DEV v1.3, Brasil.

SOFTEX - Associação para Promoção da Excelência do Software Brasileiro, 2013, iMPS 2012 - Evidências Sobre o Desempenho das Empresas que Adotaram o Modelo MPS-SW desde 2008, Brasil.

Sommerville, I., 2010, Software Engineering, 9th edition, Addison-Wesley.

Sordi, J. O., 2005, Gestão por processos: uma abordagem da moderna administração, Saraiva, São Paulo.

SPEKX, 2009, Spekx for IT Governance completa dois anos em funcionamento no Serpro, Disponível em: *http://www.spekx.com.br/press_releases/013.html*, Último Acesso: Julho/2015.

Springer, B., Springer, S., 1990, Human resource management in the U.S.: celebration of its century, In Human resource management: an international comparison, Walter de Gruyter, Berlin and New York.

Srark, J., 1998, A few words about tqm, John Stark Associates.

the Standish Group International, 2009, CHAOS Summary 2009 – The 10 Laws of CHAOS, Report.

the Standish Group International, 2013, CHAOS Report 2013, Report.

Stergiou, M., Johnson, L., 1998, The Importance of business rules in the organizational transformation process, In: International Conference on Information Systems, Analysis And Synthesis, 4., International Institute of Informatics and Systemics, Orlando.

Succi, G., Marchesi, M., 2001, Extreme Programming Examined, Ed. 1, Pearson Education.

Sutherland, J. V., Solingen, D. M. van, Rustenberg, E. 2011, The Power of SCRUM, CreateSpace Independent Publishing Platform, USA.

Taylor, F. W., 1970, Princípios de administração científica, Atlas, São Paulo.

The Cabinet Office, 2011, ITIL - Service Strategy - 2001 edition, The Stationey Office.

The Economist, 1992, The cracks in quality, The Economist, 18 April, p. 67-68.

Travassos, G. H., 1994, O Modelo de Integração de Ferramentas da Estação TABA, Orientadora Ana Regina Cavalcanti da Rocha, Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, Brasil.

Trkman, P., 2010, The Critical Success Factors of Business Process Management, International Journal of Information Management, v. 30, p. 125-134.

Turner, M. S., 2006, Microsoft Solutions Framework Essentials (Pro-Developer), Microsoft Press.

Unterkalmsteiner, M., *et al.*, 2012, Evaluation and Measurement of Software Process Improvement - A Systematic Literature Review, IEEE TSE 38(2): 398-424.

USA Today, 1995, Is TQM dead?, USA Today, 17 October, B1-B2.

Vara, J. L., Sánchez, J., Pastor, O., 2008, Business process modelling and purpose analysis for requirements analysis of information systems, In Proceedings of the 20th international conference on Advanced Information Systems Engineering, CAiSE '08, p. 213–227, Berlin, Heidelberg, Springer-Verlag.

Vasconcellos, F. P., 2012, Gestão de Processos de Negócio e Governança de TI: Um Estudo em Instituições Financeiras, Dissertação de Mestrado, Mestrado Profissional em Sistemas de Informação e Gestão do Conhecimento, Universidade FUMEC, Belo Horizonte - MG.

Verner, L., 2004, BPM: The Promise and Challenge, ACM Queue, v. 2, n. 1.

Viega, J., McManus, J., 2000, The importance of software testing, Disponível em: *http://www.cutter.com/research/2000/crb000111.html*, Último Acesso: Julho/2015.

Wall Street Journal, 1995, Is TQM yesterday´s news or does it still shine?, Wall Street Journal, 20 September, A14.

Watanabe, S., 1995, The 'Japanese Model':its evolution and transferability, Anais Simpósio Internacional Gestão, Economia e Tecnologia, USP, São Paulo.

Weber, K. C. *et al*., 2004a, Modelo de Referência para Melhoria de Processo de Software: uma abordagem brasileira, In: XXX Conferencia Latinoamericana de Informatica (CLEI2004), Sesión. 2004a. p. 20-10.

Weber, K. C. *et al*., 2004b, Uma estratégia para melhoria de processo de software nas empresas brasileiras, Proceedings of QUATIC, p. 73-78, 2004b.

Yoshidome, E. Y. C., 2014, Uma Ontologia que Estabelece os Relacionamentos de Dependência entre as Práticas de Gerência de Requisitos e Gerência de Projetos constantes nos Modelos MR-MPS-SW e CMMI-DEV, Dissertação de Mestrado – PGGC-UFPA.

Zhu, Z., Scheuermann, L., 1999, A comparison of quality programmes: total quality management and ISO 9000, Total Quality Management, vol. 10, n. 2, p. 291-297.

# AUTORES


## Alexandre Marcos Lins de Vasconcelos

Mestre em Ciência da Computação pela UFPE, Doutor em Ciência da Computação pela University of York e Pós-Doutor em Engenharia de Software pela Universidad Politécnica de Valencia. Professor Associado do Centro de Informática da Universidade Federal de Pernambuco. Implementador e Avaliador Líder MPS.

## Sandro Ronaldo Bezerra Oliveira

Mestre, Doutor e Pós-Doutor em Ciência da Computação pela UFPE. Professor Adjunto do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Pará. Implementador e Avaliador Líder MPS.

### Gustavo Henrique da Silva Alexandre

Mestre em Engenharia de Software pelo CESAR e Aluno de Doutorado em Ciência da Computação pela UFPE. Professor no MPES (Mestrado Profissional em Engenharia de Software) do CESAR. Atua na concepção e criação de cursos de ensino à distância no CESAR.EDU.

### Jussara Adolfo Moreira

Mestre em Engenharia de Software pelo CESAR. Professora do Instituto Federal Pernambucano - Campus Petrolina.

### Kamila Nayana Carvalho Serafim

Graduada em Análise e Desenvolvimento de Sistemas pela FAFICA e Aluna de Mestrado em Ciência da Computação pela UFPE. Certificada CTFL pelo BSTQB. Atualmente é Analista de Testes Pleno da Accenture - Brasil.

### Leonardo da Silva Leandro

Aluno de Mestrado em Ciência da Computação pela UFPE. Engenheiro de Software Pleno no Projeto Motorola Mobility no Centro de Informática da UFPE. Atua na automação de testes para smartphones Motorola. Experiência em desenvolvimento de software, metodologias ágeis de gerenciamento e desenvolvimento e modelagem arquitetural de software.

### Marcela da Conceição Oliveira de Souza

Mestre em Engenharia de Software pelo CESAR e Aluna de Doutorado em Ciência da Computação pela UFPE. Atua como Gestora de Requisitos/Negócios pela INFOX Tecnologia lotado no TRF 5ª Região.

## Raquel Godoi do Amaral

Graduada em Sistemas de Informação pela Faculdade Estácio de Sá - Recife. Atua na área de qualidade da Accenture - Brasil, sendo responsável pelas auditorias de projeto e produto, realizando suporte a gerentes, líderes técnicos e equipe dos projetos da organização. Possui formação nos cursos oficiais de Introdução ao CMMI-DEV V1.3 e Introdução ao MPS. BR (C1-MPS.BR).

*Título* Qualidade, Gestão e Processos de Software
*Autores* Alexandre Marcos Lins de Vasconcelos
Sandro Ronaldo Bezerra Oliveira
Gustavo Henrique da Silva Alexandre
Jussara Adolfo Moreira
Kamila Nayana Carvalho Serafim
Leonardo da Silva Leandro
Marcela da Conceição Oliveira de Souza
Raquel Godoi do Amaral
*Projeto Gráfico e diagramação* Tamyres Siqueira
*Capa* Editora EDUFPE

*Revisão de Texto* Alexandre Marcos Lins de Vasconcelos
Sandro Ronaldo Bezerra Oliveira

*Formato* eletrônico
*Fontes* Caecilia LT Std, Aero Matics

*Editoração eletrônica* TIC Editora EDUFPE


9 788541 507332