CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
MESTRADO EM TECNOLOGIA

FRANCISCO XAVIER FREIRE NETO

PROPOSTA DE MÉTODO DE AUDITORIA AOS PROJETOS DE SOFTWARE BASEADOS NO PROCESSO UNIFICADO

SÃO PAULO
MAIO/2012

FRANCISCO XAVIER FREIRE NETO

PROPOSTA DE MÉTODO DE AUDITORIA AOS PROJETOS DE SOFTWARE BASEADOS NO PROCESSO UNIFICADO

Dissertação apresentada como exigência parcial para obtenção do título de Mestre em Tecnologia no Centro Estadual de Educação Tecnológica Paula Souza, no Programa de Mestrado em Tecnologia: Gestão, Desenvolvimento e Formação, sob orientação do Prof. Dr. Aristides Novelli Filho.

SÃO PAULO
MAIO/2012

FICHA ELABORADA PELA BIBLIOTECA NELSON ALVES VIANA
FATEC-SP / CEETEPS

Freire Neto, Francisco Xavier
F866p Proposta de método de auditoria aos projetos de software baseados no processo unificado / Francisco Xavier Freire Neto. – São Paulo : CEETEPS, 2012.
189 f. : il.

Orientador: Prof. Dr. Aristides Novelli Filho.
Dissertação (Mestrado) – Centro Estadual de Educação Tecnológica Paula Souza, 2012.

1. Software. 2. Engenharia de sistemas. 3. Engenharia de software. 4. Projeto de software. 5. Auditoria de sistemas. 6. Qualidade de software. 7. Método de auditoria de sistemas. I. Novelli Filho, Aristides. II. Centro Estadual de Educação Tecnológica Paula Souza. III. Título.

FRANCISCO XAVIER FREIRE NETO

PROPOSTA DE UM MÉTODO DE AUDITORIA AOS PROJETOS DE SOFTWARE BASEADOS NO PROCESSO UNIFICADO

PROF. DR. ARISTIDES NOVELLI FILHO

PROF. DR. CARLOS HIDEO ARIMA

PROF. DR. MARCELO DUDUCHI FEITOSA

São Paulo, 11 de maio de 2012

Dedico esse trabalho à minha família, que me apoiou e incentivou em todos os momentos, até a conclusão deste projeto.

Dedico-o também aos meus amigos e a todos que me auxiliaram em seu desenvolvimento.

Agradecimentos

Em primeiro lugar a Deus, pela graça atingida.

Ao Professor Dr. Aristides Novelli Filho, pela orientação, apoio, paciência, contribuição à elaboração desta dissertação e, acima de tudo, por ter acreditado em meu projeto.

Aos Professores Dr. Marcelo Duduchi Feitosa e Dr. Carlos Hideo Arima, que compuseram a banca examinadora, pelas orientações e contribuições para esse trabalho, enriquecendo-o com seus conhecimentos singulares.

A todos os professores que fazem parte do Programa de Mestrado em Tecnologia, por terem transmitido conhecimentos necessários ao desenvolvimento da minha dissertação, e a todos os demais colaboradores, em especial os colaboradores da secretaria, por terem fornecido todo o respaldo administrativo para cumprimento das obrigações do curso.

A todos os profissionais que contribuíram com o preenchimento do questionário, principal fonte de informação para a realização desse trabalho.

A minha esposa Vânia, por toda a contribuição com meus estudos, em especial pelas revisões desse trabalho.

Aos meus pais, Regina e Roque, por nunca terem medido esforços em prover os recursos necessários para que eu conseguisse ter uma formação acadêmica completa.

A minha família e amigos, pela tolerância às ausências e compreensão.

Enfim agradeço a todas as pessoas que direta ou indiretamente tornaram possível a elaboração deste trabalho.

Resumo

NETO, Francisco Xavier Freire. **Proposta de Método de Auditoria aos Projetos de Software Baseados no Processo Unificado**. 2011. 177 f. Dissertação (Mestrado em Tecnologia) - Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2011.

Este trabalho propõe um método de auditoria para acompanhamento de projetos de software, composto de um conjunto de atividades baseadas nos principais padrões de referência, para ser utilizado em avaliações realizadas por equipes independentes, visando identificar falhas que possam afetar o produto entregue e definir ações buscando a entrega do software dentro do prazo e orçamento, com todas as características e funções previstas. Este método foi validado por meio de uma pesquisa de campo com especialistas em auditoria de sistemas e no processo de software, considerando a importância das atividades de auditoria apresentadas aos projetos de software, tendo os resultados sido utilizados para definir as atividades que fazem parte do método. Além da contribuição dos especialistas, foi considerada na composição do método uma pesquisa bibliográfica de livros e artigos que abordam o tema desse trabalho.

Palavras-chave: software; engenharia de sistemas; engenharia de software; projeto de software; auditoria de sistemas; qualidade; qualidade de software; método de auditoria de sistemas.

Abstract

NETO, Francisco Xavier Freire. **Proposta de Um Método de Auditoria ao Processo Unificado de Desenvolvimento de Software**. 2011. 177 f. Dissertação (Mestrado em Tecnologia) - Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2011.

This essay proposes an audit method for project tracking software, which is composed of a set of activities based on key standards to be used in assessments by independent teams, in order to identify failures that may affect the product delivered and set actions seeking delivering software on time and within budget, with all the features and functions provided. This method was validated through a field research with experts in auditing systems and software process, considering the importance of audit activities presented to software projects, and the results were used to define the activities that are part of the method. Besides the contribution of experts, it was considered in the composition of the method a literature search of books and articles that addresses the theme of this essay.

Keywords: software; systems engineering; software engineering; software design; system audit; quality; software quality; systems audit method.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AICPA	*American Institute of Certified Public Accountants*
ABNT	Associação Brasileira de Normas Técnicas
CMMI	*Capability Maturity Model Integration*
COBIT	*Control Objectives for Information and related Technology*
COSO	*Committee of Sponsoring Organizations*
lSACA	*Information Systems Audit and Control Association*
ISO	*International Organization for Standardization*
PRM	*Process Reference Model*
PU	Processo Unificado
RFP	Solicitação de Pedido de Proposta
SLA	*Service Level Agreement*
SEI	*Software Engineering Institute*
SGQ	Sistema de Gerenciamento da Qualidade
TI	Tecnologia da Informação
TOE	*Target of Evaluation*
TSF	*TOE Security Functionality*
UML	*Unified Modeling Language*
WEB	*Worldwide Web*

Sumário

Introdução

O desenvolvimento de software é uma atividade complexa, uma vez que os softwares devem atender as necessidades das organizações, para que consigam atingir suas metas. Nos últimos anos, houve um crescimento considerável na indústria de softwares em todo mundo, em virtude do avanço e disseminação da informática, o que levou empresas e desenvolvedores a profissionalizarem suas atividades de produção de softwares buscando entregar um produto de qualidade, obter uma melhor distribuição dos recursos empregados e, consequentemente, o aumento de vendas e a maximização dos seus lucros. Segundo Sabbag (2009), os processos foram informatizados nas organizações a tal ponto que projetos de infraestrutura, desenvolvimento de sistemas e melhoria de operações tornaram-se urgentes e cruciais.

Com base nas informações da Associação Brasileira das Empresas de Software (ABES, 2011), em 2010 o mercado brasileiro de software e serviços atingiu a 11ª posição no cenário mundial, tendo movimentado 19,4 bilhões de dólares, equivalente a 1,00% do produto interno bruto (PIB) brasileiro daquele ano. Deste total, foram movimentados 5,51 bilhões em software, o que representou perto de 2,2% do mercado mundial e 13,53 bilhões em serviços relacionados. A participação de programas de computador desenvolvidos no país atingiu 35% do total do mercado brasileiro de software, reforçando a tendência de crescimento que vem sendo apontada desde 2004, quando a participação era de 27%. Este mercado é explorado por cerca de 8.520 empresas, dedicadas ao desenvolvimento, produção e distribuição de software e de prestação de serviços. Daquelas que atuam no desenvolvimento e produção de software, 94% são classificadas como micro e pequenas empresas.

Apesar do crescimento desse mercado, empresas especializadas no desenvolvimento de software têm reportado problemas constantes quanto à qualidade dos produtos, cumprimento de prazos e dos custos dos projetos. Com base no relatório divulgado pelo The Standish Group (2011), apenas 37% dos projetos de desenvolvimento de software podem ser considerados de sucesso ao

final de sua execução e os demais 63% são encerrados antes do planejado ou terminam com resultados considerados insatisfatórios. As principais causas do insucesso desses projetos não recaem sobre a falta de domínio técnico, mas sim, são atribuídos à ausência ou à inadequação de métodos, técnicas ou práticas de gerenciamento de projetos. Segundo Pressman (2011), apesar da evolução do software nos últimos 50 anos, ainda ocorrem problemas na construção de software de alta qualidade no prazo e dentro do orçamento.

Embora existam metodologias de desenvolvimento de software e de gerenciamento de projetos, que são conjuntos sistematizados de tarefas necessárias para executar e gerenciar um projeto de software, ainda é frequente a ocorrência de falhas no desenvolvimento de software. Conforme Chin (2004), usualmente, esses projetos estão inseridos em ambientes de negócio bastante dinâmicos, sujeitos a mudanças constantes, o que foge aos padrões do gerenciamento clássico de projetos. Com isso, há espaço para se avaliar a viabilidade da inclusão de atividades de auditoria nos projetos de desenvolvimento de software como proposta para reduzir as falhas nesse tipo de projeto. Segundo Schmidt, et al. (2006), a auditoria de sistemas durante o desenvolvimento consiste em revisar e avaliar o processo de construção de sistemas de informação, desde o levantamento e estudo de viabilidade do sistema a ser informatizado até o teste e implantação do mesmo.

Questão-Problema

Um desafio predominante na gestão de projetos de desenvolvimento de software é prover o desenvolvimento e a entrega do produto de acordo com o planejamento de três principais aspectos: tempo (urgência da empresa), custo (limite orçamentário) e a qualidade do produto (ou capacidade). Apesar das tentativas mais rigorosas de fazer a entrega do projeto de acordo com o planejado, uma considerável proporção dos projetos de desenvolvimento de software resulta em sistemas que não atendem às expectativas do usuário. Com isso, atividades de monitoração são necessárias para prevenir falhas ou situações de risco, mas para isso são necessários processos e ferramentas que possam aperfeiçoar esta atividade.

Considerando esse cenário, o presente estudo considera a seguinte questão-problema:

- Como estabelecer o acompanhamento padronizado de atividades de controle nos projetos de software que seguem o processo unificado de desenvolvimento de software com o intuito de reduzir falhas no produto entregue?

Ciente da complexidade que envolve o processo de software, visando tornar esta pesquisa exequível, uma vez estabelecida a questão-problema, foram definidas as seguintes delimitações:

- Foi estabelecido como cenário de estudo e pesquisa o contexto de sistemas de informações comerciais e o processo unificado de desenvolvimento de software, por ter fases de processo relacionadas mais estritamente aos negócios do que a assuntos técnicos (SOMMERVILLE, 2011);
- Foram definidos como elegíveis para a pesquisa de campo realizada especialistas em auditoria de sistemas e no processo de software.

Objetivos da Pesquisa

Este estudo tem como objetivo principal:

- Propor um método de auditoria identificando as atividades mínimas que devem ser consideradas no acompanhamento dos projetos de software baseados no processo unificado, o qual possa auxiliar na gestão do desenvolvimento de software, sendo este definido a partir do levantamento e análise dos principais padrões de referência para engenharia de software e auditoria de sistemas.

Contempla, ainda, os seguintes objetivos específicos:

- Identificar, por meio de pesquisa bibliográfica, as potenciais atividades de controle do processo unificado de desenvolvimento de software com base na bibliografia especializada, normas técnicas e guias de boas práticas.

- Validar, por meio de pesquisa de campo, a aplicabilidade das atividades de controle identificadas de acordo com a avaliação de especialistas em auditoria de sistemas e desenvolvimento de software.

Projetos de desenvolvimento de software são em sua maioria complexos e envolvem diversos *stakeholders*, o que torna necessário um acompanhamento contínuo para se prevenirem e corrigirem eventuais falhas, antes que impactem no produto. No entanto, o dia a dia das equipes responsáveis pelos projetos nem sempre permite um acompanhamento rigoroso de todas as atividades, portanto, são necessárias ferramentas de apoio. Dessa forma, o método de auditoria proposto indicará pontos de controle para o acompanhamento independente sobre este tipo de projeto, com o intuito de evitar as principais falhas que abarcam o seu andamento, contribuindo para a melhoria da qualidade do produto entregue.

Justificativas e Contribuições da Pesquisa

Para Sabbag (2009), os processos foram informatizados nas organizações a tal ponto que projetos de infraestrutura, desenvolvimento de sistemas e melhoria de operações tornaram-se urgentes e cruciais.

A necessidade de se evitarem falhas nos projetos de software exige que profissionais de desenvolvimento aperfeiçoem seus processos e que os gerenciadores mantenham mais controles sobre os projetos. Isso promoveu a evolução da engenharia de software e exigiu que se criassem modelos e normas internacionais, voltadas à melhoria e qualidade do processo de desenvolvimento e de produtos de software. Contudo, apesar desta evolução e do investimento feito em processos de desenvolvimento, ainda não há garantias de que os softwares sejam desenvolvidos conforme o projeto e satisfaçam as necessidades dos usuários, o que possibilita a realização de estudos voltados à proposição de processos e ferramentas de apoio ao processo de software no contexto atual, em que projetos de software podem contemplar volumosas equipes de colaboradores com conhecimentos multidisciplinares, havendo então a necessidade de mais controles para garantir o cumprimento das suas atribuições.

Como resposta às crescentes pressões do mercado e ao desempenho insatisfatório dos projetos de desenvolvimento de software conduzidos com o uso de métodos clássicos de desenvolvimento e gerenciados de acordo com os princípios do gerenciamento clássico de projetos, há espaço para o estudo de um conjunto sistematizado de tarefas necessárias para completar um projeto de desenvolvimento de software. De acordo com o IT Governance Institute (ITGI, 2009), muitas implementações de sistemas em organizações fracassaram em função de controles inadequados e, consequentemente, em função de os sistemas gerarem informações não confiáveis. Muitas destas falhas poderiam ter sido evitadas por um foco mais diligente na concepção e ensaio de controles antes da implementação.

A análise do relatório “CHAOS Manifesto”, publicado pelo Standish Group (2011) e da literatura correlata (CHIN, 2004; THOMSETT, 2002) sugere a existência de divergências entre as práticas do gerenciamento clássico de projetos adotadas pelas organizações e suas necessidades propriamente ditas. Conforme Chin (2004), apesar dos ganhos obtidos, o gerenciamento clássico de projetos não se mostrou plenamente efetivo para projetos de desenvolvimento de software, o que indica a necessidade de se repensar a prática de gerenciamento de projetos nas organizações. Esta diferença se explica pelo fato de que, usualmente, os projetos de desenvolvimento de software estão inseridos em ambientes de negócio bastante dinâmicos, sujeitos a mudanças constantes, o que foge aos padrões do gerenciamento clássico de projetos.

Há soluções desenvolvidas para melhoria da qualidade do desenvolvimento de softwares nas organizações, porém, ainda que direcionem parcialmente as necessidades deste processo, estas soluções requerem que as organizações estejam apoiadas em um modelo de processo organizacional eficiente. Os modelos tecnológicos e de gestão podem levar as organizações a uma necessidade maior de responsabilização pela segurança dos softwares desenvolvidos, tomando uma postura proativa na forma como concebem, desenvolvem, operam ou terceirizam os softwares que contemplam a solução técnica dos seus processos de negócio.

De acordo com o relatório do seminário Grandes Desafios da Pesquisa em Computação no Brasil, promovido pela Sociedade Brasileira de Computação

(SBC, 2006), um dos cinco grandes desafios da computação identificados pela comunidade científica é o “Desenvolvimento Tecnológico de Qualidade: sistemas disponíveis, corretos, seguros, escaláveis, persistentes e ubíquos”, o qual aborda a construção de sistemas corretos e seguros, uma premissa do desenvolvimento tecnológico de qualidade. É abordada também a necessidade de se assegurar a fidedignidade dos softwares, em virtude da sua crescente participação na sociedade. Em particular, um dos tópicos levantados na pesquisa é o projeto de sistemas fidedignos por construção. A engenharia de software é um processo complexo, em que são gerados artefatos em diferentes níveis de abstração e de complexidade e é de suma importância garantir a correspondência entre esses artefatos. O relatório aponta que cerca de 50% do software tornado disponível contém falhas não triviais. Além disso, o relatório indica a necessidade de se desenvolverem softwares que possam evoluir sem comprometer a sua qualidade.

Em linha com esse contexto, há diversas pesquisas desenvolvidas buscando contribuir com o tópico de pesquisa “Desenvolvimento de ferramentas de apoio ao processo de implementação e de evolução de software”, associado ao desafio “Desenvolvimento Tecnológico de Qualidade: sistemas disponíveis, corretos, seguros, escaláveis, persistentes e ubíquos”. Como exemplo, a de Moreira, et al. (2007), visando o desenvolvimento de um modelo de qualidade no desenvolvimento de software e a de Dantas, et al. (2008), buscando desenvolver uma ferramenta de apoio ao processo de software.

Além disso, foram identificados dissertações e artigos publicados em periódicos, que abordam qualidade de software. Navas (2006), apresentou na sua dissertação um método para a melhoria do processo de desenvolvimento de software aplicando conceitos de CMM, SPICE e da norma ISO 12207. Etxeberria, et al. (2008), apresentou em seu artigo conclusões, requisitos e diretrizes para lidar com aspectos de qualidade em linhas de produtos de software. Gomes, et al. (2001), Meneses, et al. (2001), Gomes, et al. (2001), Oliveira, et al. (2002), Campos, et al. (2006), Bezerra (2004), Andrade, et al. (2004), Cerdeiral, et al. (2009), Santos, et al. (2010), Carvalho, et al. (2010) e Martinez, et al. (2011) apresentaram em seus artigos diferentes abordagens para avaliação e melhoria de processos e projetos de software. Almeida, et al. (2009) e Guedes, et al. (2009) apresentaram abordagens de ferramentas de apoio à melhoria de qualidade de software. Gusmão, et al. (2004) e

Pereira, et al. (2006) apresentaram abordagem de gerência de risco no processo de software. Também foram identificadas abordagens de melhoria de processo de software baseada em medição (MOREIRA, et al., 2009), *benchmarking* em iniciativas de melhorias em processos de software (ZANETTI, et al., 2009).

Este cenário demonstra a necessidade de se estabelecerem atividades de controle no processo de desenvolvimento de software, com o objetivo de evitar falhas no produto entregue, sendo também justificativa dessa pesquisa. De acordo com Thomsett (2002), os projetos de desenvolvimento de software têm basicamente duas vertentes – uma técnica e outra gerencial. O autor afirma que, por determinado período, muita atenção foi dada ao aprimoramento dos modelos de desenvolvimento de software (ênfase técnica), ficando o componente gerencial relegado ao segundo plano.

Segundo Imoniana (2005), para obter êxito nos projetos de desenvolvimento de sistemas, é necessário envolver a alta direção da corporação, principalmente na aprovação ou recomendação de alterações nos projetos e no acompanhamento destes. É necessário também realizar um estudo da viabilidade econômica, operacional e técnica para definição da aquisição de aplicações novas e definição de padrões para o sistema. Segundo Schmidt, et al. (2006) a auditoria de sistemas em desenvolvimento deve compreender atividades de revisão e avaliação do processo de desenvolvimento de sistemas de informação em todo o ciclo de desenvolvimento, até a implantação do sistema de informação.

Assim, o objetivo do método de auditoria proposto é o de contribuir para avaliações independentes, que possam apontar inclusive falhas relacionadas a fatores subjetivos como de conflitos de interesses, entre outros. Diferentemente de auditorias realizadas com o fim de certificar a maturidade dos processos de desenvolvimento de software, como CMMI ou MPS.BR, o método tem como foco o acompanhamento pontual de projetos em tempo de execução, sendo o resultado esperado um parecer tempestivo, apresentando riscos que necessitam de ações corretivas para evitar falhas no produto final.

A preocupação em contribuir ao tema “Qualidade de Software”, deveu-se basicamente a três razões principais:

a) Os modelos existentes, a exemplo do CMMI e SPICE, são bastante abrangentes e completos, portanto, salvo alguma inovação de paradigma, qualquer tentativa de um novo modelo possivelmente pouco acrescentaria;

b) Qualquer novo modelo exigiria um estudo exaustivo pelo menos do CMMI e SPICE como também um longo prazo de desenvolvimento, haja vista que os modelos CMM / CMMI vêm sendo aperfeiçoados há mais de quinze anos;

c) Por outro lado, pouco se tem desenvolvido em relação a um método (não um *framework*) dirigido a empresas de qualquer porte, focando inicialmente, porém, as de pequeno e médio porte, onde é maior a dificuldade de adoção dos modelos consagrados.

No aspecto pessoal sempre houve uma inquietude em relação ao assunto, em função das recorrentes falhas observadas no dia-a-dia de trabalho, em empresas de pequeno, médio e grande porte. Além disso, esse trabalho pretende contribuir com os estudos relacionados à engenharia de software.

Metodologia

A construção do método proposto na presente dissertação teve como base um estudo feito a partir de conhecimentos práticos, a partir das necessidades identificadas na literatura, sendo assim, optou-se pela pesquisa explicativa. Segundo Gil (2007), a pesquisa explicativa tem como objetivo primordial identificar fatores que determinam ou que contribuem para a ocorrência de fenômenos. Este tipo de pesquisa é a que mais aprofunda o conhecimento da realidade, e por isso mesmo, de grande utilidade, pois geralmente possui aplicação prática.

Como procedimentos técnicos, foram realizadas:

- pesquisa de campo, para validar as atividades do método proposto;
- pesquisa bibliográfica, buscando-se fontes especializadas nas áreas de Sistemas de Informação e Tecnologia da Informação, nas quais foram abordados aspectos de qualidade no desenvolvimento

de software, auditoria de sistemas e boas práticas de gerenciamento de projetos.

A criação de software inclui uma série de processos, atividades e metodologias para trazer eficiência nas operações de uma entidade empresarial. Para definir o conjunto de atividades contempladas no método de auditoria proposto, foi necessário estudar o processo de software e as metodologias existentes, visando os principais conceitos, boas práticas e ferramentas que apresentam recomendações para se planejar corretamente o desenvolvimento de um software, implementá-lo de forma adequada, verificar se atende às necessidades dos usuários e garantir formalmente ao cliente a qualidade do software.

Além disso, foi necessário aprofundar o conhecimento sobre as características dos principais padrões de referência que indicam atividades de controle para a área de Tecnologia da Informação, sendo eles:

- Cobit 4.1 (ITGI, 2007), modelo de Governança de TI, que define o que deve ser controlado nos processos de TI de uma empresa;
- CMMI for Development, Version 1.3 (SEI, 2011), que é um modelo de qualidade de software que contém práticas necessárias à maturidade das organizações que desenvolvem softwares;
- ISO/IEC 15408 (ISO/IEC/JTC, 2009), que fornece um conjunto de critérios fixos que permitem especificar a segurança de uma aplicação de forma não ambígua a partir de características do ambiente da aplicação, e definir formas de garantir a segurança da aplicação para o cliente final;
- MPS.BR - Melhoria de Processos do Software Brasileiro (SOFTEX, 2012), que é simultaneamente um movimento para a melhoria da qualidade (Programa MPS.BR) e um modelo de qualidade de processo (Modelo MPS) voltado para a realidade do mercado de pequenas e médias empresas de desenvolvimento de software no Brasil. É compatível com o CMMI e foi baseado nas normas ISO/IEC 12207, ISO/IEC 15504 e na realidade do mercado brasileiro.

- NBR ISO/IEC 12207 (ABNT, 2009), que define o ciclo de vida do processo de software.
- NBR ISO/IEC 15504 (ABNT, 2008), que define uma estrutura para a realização de avaliações de processos em organizações;
- NBR ISO/IEC 27002 (ABNT, 2005), que é um código de prática para a gestão da segurança da informação;
- NBR ISO/IEC 9126-1 (ABNT, 2003), que descreve um modelo de qualidade do produto software.

Em complemento a esta pesquisa, realizou-se a participação no Seminário Engenharia de Software 2010 (Tempo Real Eventos, 2010), que abordou diversos assuntos sobre engenharia de software e no Congresso CNASI Latino Americano de Auditoria de TI (IDETI, 2011) buscando identificar os principais problemas relacionados ao processo de software e que afetam diretamente seu ciclo de vida. Partindo desse pressuposto, o estudo foi focado nas principais causas desses problemas e a que eles estão relacionados.

Estrutura do Trabalho

Além desta introdução, que contemplou a relevância, o interesse do trabalho, o problema, os objetivos, as justificativas e as contribuições do estudo, a presente dissertação apresenta mais cinco capítulos:

O capítulo um, “Fundamentos de Auditoria e Referências Técnicas para Avaliação do Processo de Software” define de forma geral o processo de auditoria e de forma específica as atividades de auditoria aplicáveis aos projetos de softwares e apresenta os padrões de referência para a realização de auditorias, que são normas, *frameworks* e boas práticas ao processo de software.

O capítulo dois, “Projetos de Software”, define o processo de software de forma geral e descreve de forma específica o processo unificado, que é base para a definição do método de auditoria proposto neste trabalho.

No capítulo três, “Pesquisa de Campo: Avaliação das Atividades de Controle do Método de Auditoria Proposto”, é apresentada a pesquisa realizada para

validar o método proposto, a partir da avaliação da relevância das atividades de controle do método por profissionais, das áreas de desenvolvimento de software ou auditoria de sistemas, com base na sua experiência profissional ou percepção. Por fim, é apresentada a avaliação final e os resultados da pesquisa.

No capítulo quatro, “Proposta de Método de Auditoria para Projetos de Software Baseados no Processo Unificado”, é apresentada a descrição detalhada do método proposto, principal objetivo desta dissertação. O método apresenta as atividades previstas, os princípios e as diretrizes para a realização de auditorias em projetos de software baseados no processo unificado e as considerações quanto ao método proposto.

Na Conclusão, são apresentadas as considerações gerais sobre esta dissertação e são indicadas as perspectivas de trabalhos futuros proporcionados por esta pesquisa.

No Apêndice A, “Mapeamento dos padrões de referência para o processo de software”, é apresentado o resultado do mapeamento de atividades de controle nos padrões de referência considerados nessa pesquisa.

No Apêndice B, “Mapeamento de atividades para o método de auditoria em projetos de software baseados no Processo Unificado”, é apresentado o resultado do mapeamento de atividades que integram o método proposto.

No Apêndice C, “Aspectos que podem influenciar auditorias em projetos de software baseados no Processo Unificado”, é apresentado o instrumento aplicado na pesquisa de campo.

1. Fundamentos de Auditoria e Referências Técnicas para Avaliação do Processo de Software

Esse estudo tem como base conhecimentos específicos sobre processos de auditoria, além de padrões de referência que apresentam as melhores práticas a serem observadas nesses trabalhos, de acordo com o ambiente a ser auditado, sendo nessa pesquisa, o processo de software. Dessa forma, o presente capítulo apresenta a primeira parte da fundamentação teórica desse trabalho, abrangendo conceitos de auditoria e padrões de referência considerados nessa pesquisa.

1.1. Conceitos de Auditoria

O termo auditoria pode ser usado para descrever uma grande variedade de atividades em nossa sociedade (BOYNTON, et al., 2006). A definição geral de auditoria a seguir identifica um número de atributos comuns das mais modernas atividades de auditoria, como descrito na Figura 1.1. Segundo eles, na definição mais abrangente, constante do Relatório do Comitê de Conceitos Básicos de Auditoria da Associação Americana de Contabilidade, auditoria compreende o processo sistemático de obtenção e avaliação de evidências com foco em afirmações sobre ações e eventos econômicos para avaliar o grau de correspondência entre estas afirmações e os critérios estabelecidos, para comunicação dos resultados aos usuários interessados.

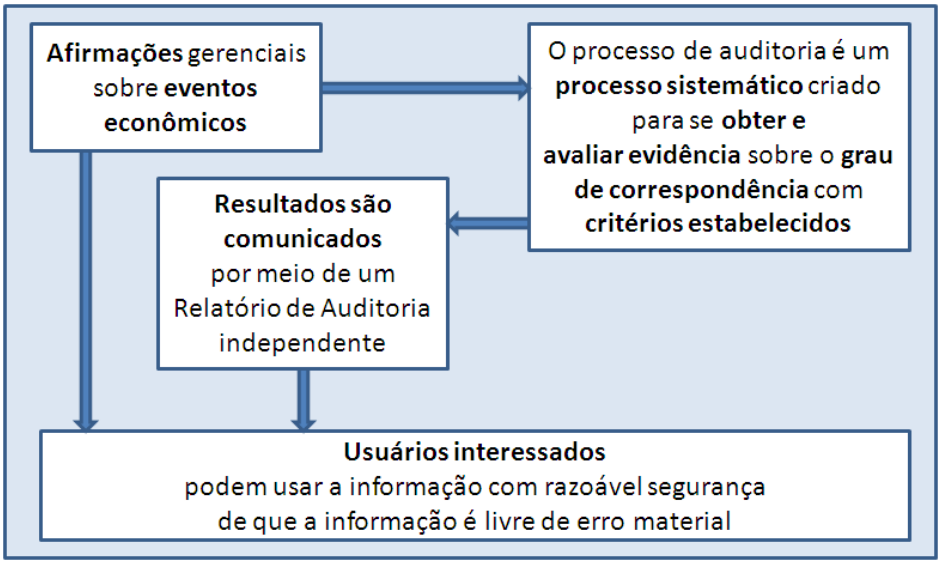

Figura 1.1 – Visão geral do processo de auditoria (BOYNTON, et al., 2006)

Esta definição compreende os seguintes termos, detalhados abaixo:

- processo sistemático: envolve uma série de passos e procedimentos organizados de maneira lógica e estruturada;
- obtenção e avaliação objetiva das evidências: significa examinar as bases das afirmações e avaliá-las de maneira justa sem viés ou preconceito;
- afirmações sobre ações e eventos econômicos: são as declarações ou informações fornecidas pelo indivíduo ou entidade correspondentes aos assuntos sujeitos à auditoria. Afirmações incluem informações contidas nas demonstrações financeiras, relatórios de operações internas, ações executadas etc.;
- grau de correspondência: refere-se à afinidade com que as afirmações podem ser associadas com os critérios estabelecidos;
- critérios estabelecidos: são os padrões contra os quais as afirmações ou representações são julgadas. Podem ser determinados internamente (normas da administração) ou partir de fontes externas (leis e regulamentos). No contexto deste trabalho, pode-se considerar como critérios estabelecidos as boas práticas aplicáveis ao processo de desenvolvimento de software;
- comunicação dos resultados: obtêm-se através de um relatório formal que indique o grau de correspondência entre as afirmações e os critérios definidos. A comunicação de resultados tanto pode fortalecer como enfraquecer a credibilidade das representações feitas por terceiros;
- usuários interessados: são indivíduos que usam (ou confiam) nas constatações dos auditores. Incluem-se alta administração, acionistas, credores, órgãos governamentais e o público em geral.

Ainda Boynton, et al. (2006) definem que há três tipos de auditoria:

- auditoria das demonstrações financeiras: envolve a obtenção e avaliação das evidências sobre as demonstrações financeiras de uma entidade, com o propósito de opinar se tais demonstrações

foram elaboradas obedecendo a um critério estabelecido (normalmente os princípios de contabilidade geralmente aceitos). Este tipo de auditoria é conduzido por auditores externos nomeados pela organização cujas demonstrações financeiras estejam sendo auditadas. O resultado da auditoria nas demonstrações financeiras é destinado a um leque amplo de usuários, como acionistas, credores, órgãos reguladores e o público em geral.

- auditoria de conformidade (*compliance*): a auditoria de *compliance* compreende a obtenção e avaliação de evidências para determinar se certas atividades financeiras ou operacionais de uma entidade estão em conformidade com as condições, regras e regulamentos específicos. O critério estabelecido neste tipo de auditoria pode ser derivado de várias fontes. Como exemplos destas fontes, e de critérios estabelecidos por cada uma, podem-se destacar:
    - a administração, definindo políticas de segurança das informações e de operação de correio eletrônico;
    - os credores (ou um tipo especial de credor) definindo que as operações da empresa sigam determinados padrões para que conceda crédito a taxas subsidiadas;
    - o governo ou órgãos de tutela (Banco Central, por exemplo), definindo os tipos de operação autorizadas para a organização.

  Os relatórios da auditoria de *compliance* são geralmente direcionados para a autoridade que estabeleceu os critérios e podem incluir um sumário das constatações e um parecer que assegure o grau de aderência aos critérios estabelecidos.

- auditoria operacional: uma auditoria operacional envolve a obtenção e avaliação de evidências sobre a eficiência e a eficácia das atividades operacionais de uma entidade em relação a objetivos específicos. Este tipo de auditoria normalmente é chamado de auditoria de desempenho ou auditoria administrativa. Os relatórios destas auditorias geralmente incluem não somente avaliação da eficiência e eficácia das operações, mas também recomendações para tornar tais operações mais eficientes e eficazes.

Os profissionais que executam as atividades associadas à auditoria estão classificados em três grupos:

- auditores independentes: são profissionais que, isoladamente ou como membros de uma empresa de auditoria, prestam serviços de auditoria contábil, de *compliance* ou operacional a clientes aos quais não estão ligados funcionalmente. A remuneração pela prestação destes serviços é feita por meio de honorários.
- auditores internos: são colaboradores da empresa em que efetuam auditoria. Efetuam uma atividade de avaliação independente, chamada auditoria interna, dentro da organização e para ela. O objetivo da auditoria interna é auxiliar a administração da organização no exercício de suas responsabilidades. A abrangência da função auditoria interna estende-se a todas as fases das atividades organizacionais. Envolve principalmente auditoria operacional e de *compliance*, mas pode também auxiliar os auditores independentes na auditoria das demonstrações financeiras.
- auditores governamentais: são funcionários de órgãos municipais, estaduais e federais que executam auditorias de demonstrações financeiras, de *compliance* e operacionais, prestando contas de seu trabalho aos órgãos que os empregam e ao público.

Como o objetivo deste trabalho é tratar a auditoria em projeto de software baseados no processo unificado, o foco é a auditoria executada tanto por auditores independentes quanto por auditores internos.

1.2. A Auditoria e o Controle Interno

Qualquer atividade humana, para cumprir sua finalidade, necessita ser conduzida dentro de determinados padrões. A importância destes padrões é ampliada quando várias pessoas se unem para conduzir em comum determinado empreendimento, seja uma organização comercial, uma associação recreativa, um condomínio etc. Este conjunto de padrões que rege a vida de uma organização no sentido de ela atingir seus objetivos pode ser chamado de controle interno.

Schmidt, et al. (2006) cita que, segundo o Instituto Americano dos Contadores Públicos Certificados, através do Relatório Especial da Comissão de Procedimento de Auditoria, controle interno é o plano de organização e todos os métodos e medidas coordenados, aplicados em uma empresa, a fim de proteger seus bens, conferir a exatidão e a fidelidade de seus dados contábeis, promover a eficiência e estimular a obediência às diretrizes administrativas estabelecidas pela gestão.

A importância do controle interno e sua relevância para o trabalho dos auditores é reconhecida há longo tempo. Ilustrando tal reconhecimento, Boynton e Johnson (2006) citam que uma publicação de 1947 do AICPA (*American Institute of Certified Public Accountants*) chamada *Internal Control* indicava que os seguintes fatores contribuíam para a expansão do reconhecimento da importância do controle interno:

- o âmbito e o tamanho das entidades comerciais se tornou tão complexo e comum que a administração tem que confiar em numerosos relatórios e análises para controlar as operações de maneira eficaz;
- a verificação e revisão inerente a um bom sistema de controle interno garante proteção contra fraquezas humanas e reduz a possibilidade de que erros ou irregularidades ocorram;
- é impraticável para os auditores auditar a maioria das empresas obedecendo limitações econômicas dos honorários sem confiar no sistema de controle interno do cliente.

A preocupação com controle interno ao longo do tempo ultrapassou as fronteiras de auditoria e contabilidade. Em outubro de 1987, ainda segundo Boynton e Johnson (2006), o relatório final da *National Commission on Fraudulent Financial Reporting* (*Treadway Commission*), formada pelo Congresso dos Estados Unidos, indicava que:

- a importância na prevenção de relatórios financeiros fraudulentos deve ser definida pela alta administração, que influencia o ambiente organizacional dentro do qual os relatórios financeiros ocorrem;

- todas as empresas públicas devem manter controles internos que forneçam razoável segurança de prevenção ou detecção de relatórios financeiros fraudulentos;
- as organizações patrocinadoras da Comissão, inclusive o Conselho de Padrões de Auditoria (*Auditing Standards Board* - ASB), devem cooperar para o desenvolvimento de instruções adicionais de um sistema de controles internos.

Em 1992, seguindo a última recomendação da *Treadway Commission*, o Comitê de Organizações Patrocinadoras (COSO - *Committee of Sponsoring Organizations*) divulgou um relatório chamado *Internal Control - Integrated Framework*. Este Comitê (COSO) congregava representantes do AICPA, da *American Accounting Association*, do *Institute of Internal Auditors*, do *Institute of Management Accountants* e do *Financial Executive Institute*. O propósito do trabalho era:

- estabelecer uma definição comum de controle interno atendendo as necessidades de diferentes parceiros;
- fornecer um padrão contra o qual as empresas e outras entidades poderiam avaliar seu sistema de controle e determinar como incrementá-lo.

O relatório do COSO (1994) define controle interno como um processo, executado pelo conselho de administração da organização, pela gerência e por todos os funcionários, projetado para fornecer razoável segurança de que os objetivos da organização sejam atingidos, nas seguintes categorias:

- confiabilidade dos relatórios financeiros;
- conformidade às leis e regulamentos aplicáveis;
- eficácia e eficiência das operações.

A existência de controle interno na organização pressupõe a existência de uma estrutura de controle. Segundo o relatório do COSO (1994) esta estrutura está baseada em cinco componentes inter-relacionados:

- ambiente de controle: é a base de todos os outros componentes, porque rege a organização, influenciando a consciência de controle

de seus integrantes. Compreende integridade, valores éticos e competência das pessoas da organização, filosofia gerencial e estilo operacional, o modo como a administração delega autoridade e responsabilidade, organiza e desenvolve as pessoas e a atenção e direção fornecida pelo conselho de administração.

- avaliação de riscos: todas as instituições enfrentam uma variedade de riscos provenientes de fontes externas e internas que têm que ser avaliados. Um pré-requisito para avaliação de riscos é o estabelecimento de objetivos interligados e consistentes. A avaliação de riscos envolve a identificação e análise de ameaças que possam comprometer o alcance dos objetivos, formando uma base para determinar como o risco deve ser administrado. Como as condições mudam, são necessários mecanismos para tratar estas mudanças.
- atividades de controle: são políticas e procedimentos que ajudam a assegurar que as decisões gerenciais estão sendo seguidas. Também ajudam a assegurar que ações necessárias são tomadas para enfocar e mitigar riscos para que a organização alcance seus objetivos. Podem ocorrer em todos os níveis hierárquicos e inclui atividades como aprovações, autorizações, revisões de desempenho operacional e segregação de tarefas.
- informação e comunicação: a informação adequada precisa ser identificada, obtida e comunicada no formato e tempo oportuno de maneira que permita às pessoas executarem suas responsabilidades. As informações compreendem não apenas os dados gerados internamente, mas também dados e eventos externos. A comunicação para ser eficaz deve fluir por toda a organização e todo pessoal deve ter claramente definido seu papel no sistema de controle interno e como sua atividade se relaciona com o trabalho dos outros. Também deve fluir com os parceiros externos, clientes, fornecedores, órgãos de tutela e acionistas.

- monitoração: o sistema de controle interno deve ter sua qualidade avaliada ao longo do tempo. Esta avaliação pode tanto ser permanente (o acompanhamento ser feito durante o curso normal das operações), como em avaliações separadas do processo normal ou ainda uma combinação dos dois critérios.

Ainda de acordo com o relatório COSO (1994), existe uma sinergia entre os componentes formando um sistema integrado que reage dinamicamente às mudanças. Este sistema de controle interno está entrelaçado com as operações da organização e se torna mais efetivo quando está construído sobre a infraestrutura organizacional e faz parte da filosofia da empresa.

De acordo com o *International Fedeреation of Accountants* (IFAC, 2011), o auditor deve obter um entendimento de controle interno suficiente para planejar a auditoria e executá-la de maneira eficaz. Para isso, deve avaliar o risco de auditoria e definir procedimentos para garantir que esse risco foi reduzido a um nível aceitável. "Risco de auditoria", de acordo com esta norma, significa o risco de que o auditor expresse uma opinião incorreta ou dê um parecer errado, no caso das demonstrações contábeis possuírem distorções significativas. Para a geração do risco de auditoria cooperam três tipos de riscos:

- Risco Inerente - é a suscetibilidade do saldo de uma conta ou classe de transações a uma distorção que poderia ser relevante, individualmente ou quando considerada em conjunto com distorções em outros saldos ou classes, desde que não houvesse controles internos correlatos;
- Risco de Controle - é o risco de que os sistemas contábeis e de controle interno deixem de prevenir ou detectar, e corrigir em tempo hábil, uma distorção no saldo de uma conta ou classe de transações, que poderia ser relevante, individualmente ou quando agregada em outros saldos ou classes;
- Risco de Detecção - é o risco de que os procedimentos de comprovação de um auditor não detectem uma distorção existente no saldo de uma conta ou classe de transações que possa ser

relevante, individualmente ou quando considerada em conjunto com distorções em outros saldos ou classes.

Ao executar a avaliação do controle interno procura-se verificar se objetivos de controle estão sendo atendidos, isto é, se a finalidade para a qual um procedimento de controle existe está sendo atingida. Como existem vários fatores que influenciam o atendimento destes objetivos de controle, procura-se garantir, com razoável segurança, tal atendimento, considerando-se que a segurança integral dificilmente pode ser atingida porque:

- é impossível obter-se total segurança do atendimento;
- o volume de controles para garantir segurança integral torna este processo muito dispendioso, não compensando o benefício dele decorrente; ou
- a implementação de todos os controles necessários em um processo podem torná-lo inviável operacionalmente.

1.3. A Auditoria em Ambientes Computadorizados

A avaliação da estrutura de controle interno existe independentemente da ferramenta utilizada para tratamento e divulgação das informações. Obviamente quando estas são tratadas através de processamento eletrônico, os mecanismos para avaliar o ambiente de controle são diferentes. Não necessariamente existe uma trilha visível de todos os tratamentos que um dado recebeu para se transformar em uma informação relevante para o processo organizacional.

Boynton e Johnson (2006) traçam as seguintes diferenças que afetam o controle interno ao comparar o processamento manual e o computadorizado:

- o sistema computadorizado pode produzir uma trilha da transação que esteja disponível para auditoria apenas por um curto espaço de tempo;
- frequentemente existe menos evidência dos procedimentos de controle nos sistemas computadorizados do que nos manuais;

- as informações nos sistemas manuais são visíveis, ao contrário dos arquivos e registros nos sistemas computadorizados, que não podem ser lidos sem a ajuda de um computador;
- a diminuição do envolvimento humano nos sistemas computadorizados pode ocultar erros que poderiam ser identificados em sistemas manuais;
- a informação em sistemas computadorizados pode ser mais vulnerável a desastres físicos, manipulação não autorizada e defeitos mecânicos que as informações em sistemas manuais;
- várias funções podem ser concentradas nos sistemas computadorizados, reduzindo assim a segregação de funções seguida nos sistemas manuais;
- as mudanças nos sistemas computadorizados são mais difíceis de implementar e controlar que nos sistemas manuais;
- os sistemas computadorizados podem fornecer um nível de consistência mais confiável que os manuais porque sujeita todas as transações aos mesmos controles;
- relatórios que subsidiem a administração podem ser fornecidos de maneira mais oportuna por sistemas computadorizados que por sistemas manuais.

Embora as duas últimas diferenças favoreçam o ambiente computadorizado, as outras sete representam problemas ou riscos maiores, que devem ser tratados na estrutura de controles internos. O tratamento dos riscos trazidos pelo sistema de informação computadorizado, de acordo com Boynton e Johnson (2006), pode ser feito por controles gerais e controles de aplicação. Os controles gerais compreendem:

- Controles organizacionais e operacionais: estão associados à filosofia gerencial, estilo operacional e estrutura organizacional. Estes controles estão associados à segregação de tarefas dentro da área de processamento de dados e entre a área de processamento e os departamentos usuários.

- Controles de documentação e de desenvolvimento de sistemas: os controles no desenvolvimento de sistemas compreendem revisão, teste e aprovação de novos sistemas, controles de alteração de programas e procedimentos documentados. Os controles de desenvolvimento estão relacionados à documentação e registros mantidos pela organização para descrever as atividades de processamento computacionais. No contexto deste trabalho, esse é o principal ambiente de controle a ser explorado.
- Controles de equipamentos e de sistemas operacionais: são os controles para detectar eventuais falhas nos equipamentos (hardware) ou sistemas operacionais (*system software*) onde o processamento é efetuado.
- Controles de acesso: estes controles devem prevenir uso não autorizado dos equipamentos de processamento, arquivos de dados e programas. Incluem salvaguardas físicas (acesso restrito à área de processamento, chaves especiais etc.) e procedimentos de revisão, como avaliação da utilização dos computadores.
- Controles de dados e processamentos: fornecem uma estrutura para controlar diariamente as operações dos computadores, diminuindo a probabilidade de ocorrência de erros e assegurando a continuidade das operações no caso de um desastre físico ou falha dos equipamentos.

Os controles de aplicações compreendem:

- Controles de entrada: estão projetados para fornecer razoável segurança de que os dados recebidos para processamento estão adequadamente autorizados e que os dados inicialmente incorretos tenham sido rejeitados, corrigidos e incluídos novamente.
- Controles de processamentos: estes controles são projetados para fornecer razoável segurança de que o processamento foi executado como planejado sem perder, adicionar, duplicar ou alterar dados indevidamente.

- Controles de saídas: são projetados para assegurar que o resultado do processamento (relatórios ou arquivos magnéticos) seja correto e que somente o pessoal autorizado tenha acesso a ele.

O gerenciamento dos recursos de tecnologia de informações é feito por meio de processos de TI, que consolidam as diversas atividades relacionadas à tecnologia da informação, para atender as exigências do negócio. A execução destas atividades deve ser feita de forma que se atinjam objetivos de controle. Alcançando-se estes objetivos, existe uma razoável segurança de que as exigências do negócio estejam sendo atendidas. A função da auditoria é avaliar a execução destas atividades e se os objetivos de controle correspondentes a elas estão sendo atingidos.

A seguir, serão apresentados os principais padrões de referência que abordam em parte ou no todo atividades de controle que se aplicam ao processo de software, possibilitando a identificação das atividades que serão base ao método de auditoria proposto.

1.4. Objetivos e Papel da Auditoria de Sistemas

Segundo Schmidt, et al. (2006), a função da auditoria de sistemas é promover adequação, revisão, avaliação e recomendações para o aprimoramento dos controles internos nos sistemas de informação da empresa, bem como avaliar a utilização dos recursos humanos, materiais e tecnológicos envolvidos no processamento dos mesmos.

A auditoria de sistemas deve atuar em qualquer sistema de informação, seja no nível estratégico, gerencial ou operacional, sendo os trabalhos agrupados para alcançar os seguintes objetivos, divididos em quatro linhas mestras de atuação:

- auditoria de sistemas em produção: abrange os procedimentos e resultados dos sistemas de informação já implantados (características preventiva, detectiva e corretiva);
- auditoria durante o desenvolvimento de sistemas: abrange todo o processo de construção de sistemas de informação, desde a fase de levantamento do sistema a ser informatizado até o teste e implantação (característica preventiva);

- auditoria do ambiente de tecnologia da informação: abrange a análise do ambiente de informática em termos de estrutura orgânica, contratos de software e hardware, normas técnicas e operacionais, custos, nível de utilização dos equipamentos e planos de segurança e de contingência;
- auditoria de eventos específicos: abrange a análise da causa, da consequência e da ação corretiva cabível, de eventos localizados que não se encontram sob auditoria, detectados por outros órgãos e levados ao seu conhecimento (característica corretiva).

Nessa pesquisa, o objetivo considerado é a auditoria durante o desenvolvimento de sistemas, abrangendo acompanhamento das atividades em tempo de projeto.

Entende-se que a equipe de auditoria de sistemas pode realizar um acompanhamento independente sobre projetos de software a partir da avaliação das atividades desses projetos. Diferentemente da equipe de *Quality Assurance*, que pertence ao *staff* da equipe de TI, a equipe de auditoria de sistemas deve manter sua independência, portanto, não responde funcionalmente à equipe de TI, mas pode realizar apontamentos que irão impactar nos projetos.

O objetivo da função de *Quality Assurance* é o de identificar pontos falhos na execução dos processos de TI, bem como apontar oportunidades de melhoria na gestão desses processos e prover aos gestores subsídio para tomada de decisões que assegurem a conformidade e otimização da cadeia produtiva de TI. A função realizada pela auditoria de sistemas é mais abrangente, pois em sua avaliação, que deve ser técnica, deve garantir os interesses da direção e acionistas. No contexto de projetos de software, a auditoria de sistemas deve manter sua independência no acompanhamento do projeto, buscando identificar riscos aos interesses da empresa, mesmo que em detrimento dos interesses da equipe de TI ou do próprio cliente, tendo, portanto, que divulgar à direção e acionistas quaisquer violações, se vier a ocorrer.

Outro aspecto importante a ser considerado é o de que a auditoria de sistemas, como órgão de linha da organização, realiza trabalhos pontuais, com base

em análise dos principais riscos ao negócio, ao passo que a função de *Quality Assurance* é realizada de forma permanente, a partir do acompanhamento de indicadores da evolução da maturidade dos processos de TI da empresa.

1.5. Padrões de Referência para o Processo de Software

Para a realização de exames de auditoria, os auditores se baseiam em boas práticas para avaliar controles e direcionar recomendações às fraquezas identificadas, uma vez que as boas práticas são definidas por especialistas das respectivas áreas de atuação. Por exemplo, o *framework* Cobit indica boas práticas que direcionam a avaliação do nível da gestão de TI e do ambiente de controle interno da organização.

A seguir são apresentados os principais padrões de referência para avaliação do processo de software, os quais foram considerados nessa pesquisa.

1.5.1. Cobit

A associação lSACA *(Information Systems Audit and Control Association)*, de âmbito mundial, que congrega auditores de informática de mais de 100 países, com o apoio de instituições como Unisys e Coopers & Lybrand, desenvolveu, em 1996 uma estrutura para facilitar a condução de auditoria em ambientes computadorizados. Tal estrutura, baseando-se nos componentes de controle definidos no relatório COSO, procura avaliar se as exigências do negócio estão sendo atendidas pela tecnologia de informação.

A orientação para negócios é o principal tema do Cobit (ITGI, 2007), o qual foi desenvolvido não somente para ser utilizado por provedores de serviços, usuários e auditores, mas também, e mais importante, para fornecer um guia abrangente para os executivos e donos de processos de negócios. A estrutura Cobit é baseada nos seguintes princípios (Figura 1.2):

> Prover a informação de que a organização precisa para atingir os seus objetivos, as necessidades para investir, gerenciar e controlar os recursos de TI usando um conjunto estruturado de processos para prover os serviços que disponibilizam as informações necessárias para a organização (ITGI, 2007, p. 12).

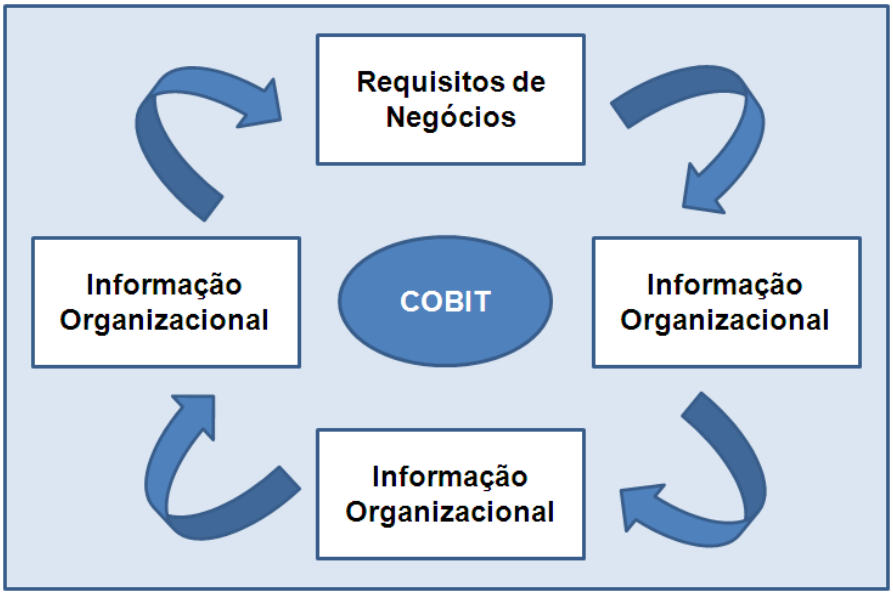

Figura 1.2 – Entrelaçamento da TI com as exigências do negócio (ITGI, 2007)

A estrutura Cobit (ITGI, 2007) define controle como sendo:

> Políticas, procedimentos, práticas e estruturas organizacionais criadas para prover uma razoável garantia de que os objetivos de negócios serão atingidos e que eventos indesejáveis serão evitados ou detectados e corrigidos (ITGI, 2007, p. 15).

Objetivo de controle em tecnologia de informações é definido como:

> Um conjunto completo de requisitos de alto nível a serem considerados pelos executivos para um controle efetivo de cada processo de TI (ITGI, 2007, p. 15).

Segundo a estrutura Cobit (ITGI, 2007), para atender aos objetivos de negócios, as informações precisam se adequar a certos critérios de controles, aos quais o Cobit denomina necessidades de informação da empresa. Baseado em abrangentes requisitos de qualidade, guarda e segurança, sete critérios de informação distintos e sobrepostos são definidos, como segue:

- **Efetividade** lida com a informação relevante e pertinente para o processo de negócio bem como a mesma sendo entregue em tempo, de maneira correta, consistente e utilizável;
- **Eficiência** relaciona-se com a entrega da informação através do melhor (mais produtivo e econômico) uso dos recursos;

- **Confidencialidade** está relacionada com a proteção de informações confidenciais para evitar a divulgação indevida;
- **Integridade** relaciona-se com a fidedignidade e totalidade da informação bem como sua validade de acordo os valores de negócios e expectativas;
- **Disponibilidade** relaciona-se com a disponibilidade da informação quando exigida pelo processo de negócio hoje e no futuro. Também está ligada à salvaguarda dos recursos necessários e capacidades associadas;
- **Conformidade** lida com a aderência a leis, regulamentos e obrigações contratuais aos quais os processos de negócios estão sujeitos, isto é, critérios de negócios impostos externamente e políticas internas;
- **Confiabilidade** relaciona-se com a entrega da informação apropriada para os executivos, para que possam administrar a entidade e exercer suas responsabilidades fiduciárias e de governança.

A estrutura define que a geração das informações para o negócio é feita utilizando os seguintes recursos de tecnologia de informações:

- **Aplicativos** são os sistemas automatizados para usuários e os procedimentos manuais que processam as informações;
- **Informações** são os dados em todas as suas formas, a entrada, o processamento e a saída fornecida pelo sistema de informação em qualquer formato a ser utilizado pelos negócios;
- **Infraestrutura** refere-se à tecnologia e aos recursos (ou seja, hardware, sistemas operacionais, sistemas de gerenciamento de bases de dados, redes, multimídia e os ambientes que abrigam e dão suporte a eles) que possibilitam o processamento dos aplicativos;
- **Pessoas** são os funcionários requeridos para planejar, organizar, adquirir, implementar, entregar, suportar, monitorar e avaliar os sistemas de informação e serviços. Eles podem ser internos, terceirizados ou contratados, conforme necessário.

Segundo a estrutura Cobit (ITGI, 2007), processos de TI são um conjunto de atividades relacionadas. Na execução destas atividades busca-se atingir objetivos de controle para assim assegurar-se que os objetivos do negócio também estejam sendo atingidos.

Os processos de TI são divididos em quatro domínios, cada um tem um conjunto de processos que são subdivididos em atividades para que os objetivos do domínio sejam alcançados, sendo que o Cobit é um modelo orientado a processos. Na estrutura Cobit há também informações sobre modelos de maturidade para cada um desses processos.

Visando verificar se as atividades ligadas à tecnologia de informação estão atendendo seus objetivos de controle, as atividades de natureza semelhante foram agrupadas em 34 processos, os quais também foram agrupados, obedecendo as semelhanças entre eles, em quatro domínios:

- **Planejar e Organizar (PO)** - Provê direção para entrega de soluções (AI) e entrega de serviços (DS);
- **Adquirir e Implementar (AI)** - Provê as soluções e as transferem para tornarem-se serviços;
- **Entregar e Suportar (DS)** - Recebe as soluções e as tornam passíveis de uso pelos usuários finais;
- **Monitorar e Avaliar (ME)** - Monitora todos os processos para garantir que a direção definida seja seguida.

a) Domínio (PO): Planejar e Organizar

Este domínio engloba as estratégias e táticas, diz respeito à identificação de como a TI pode melhor contribuir para alcançar os objetivos do negócio por meio do planejamento e da organização. Segundo o ITGI (2007), este domínio é normalmente dirigido às seguintes questões gerenciais:

- TI e o negócio estão alinhados estrategicamente?
- A empresa está usando seus recursos de forma otimizada?
- Todos na empresa conhecem os objetivos da TI?

- Os riscos da TI são conhecidos e gerenciados?
- A qualidade dos sistemas de TI é adequada para as necessidades da empresa?

A seguir, um resumo das atividades desse domínio:

- PO1 - Definir um plano estratégico de TI: O planejamento estratégico é necessário para gerenciar e direcionar todos os recursos de TI alinhados com as estratégias e prioridades do negócio. O plano estratégico deve aumentar a compreensão dos *stakeholders* em relação às oportunidades e limites da TI, avaliar o desempenho e deixar claros os níveis de investimentos necessários.
- PO2 - Definir a arquitetura de informação: A função dos sistemas de informação é criar e atualizar regularmente um modelo de informação para o negócio e definir sistemas apropriados para aperfeiçoar o uso dessas informações.
- PO3 - Determinar a direção tecnológica: A função dos serviços de informação é determinar a direção tecnológica para suportar o negócio. Isso requer a criação de um plano de infraestrutura tecnológica e um comitê que determina e gerencia de forma clara e realista as expectativas do que a tecnologia pode oferecer em termos de produtos, serviços e mecanismos de entrega.
- PO4 - Definir processos de TI, organização e relacionamentos: A organização de TI precisa ser definida, considerando os requisitos de pessoas, habilidades, funções, responsabilidades, autoridades, regras, responsabilidades e supervisão. Esta organização deve estar dentro de um modelo de processos de TI que assegure transparência e controle, além de envolver os principais executivos e gerentes de negócio. Processos, políticas e procedimentos administrativos devem ser implementados para todas as funções, com atenção especial para controles, garantia de qualidade, gerenciamento de riscos, segurança da informação, propriedade de dados e sistemas, e separação de funções. Para assegurar tempo

hábil aos requisitos de negócio, TI deve estar envolvida nos processos de decisão mais relevantes.

- PO5 - Gerenciar os investimentos em TI: Estabelecer e manter um padrão para gerenciar programas que tenham investimentos em TI e que abrangem custos, benefícios, priorização no orçamento, um processo formal de orçamento e gerenciamento em relação aos orçamentos. Consultar os *stakeholders* para identificar e controlar os custos totais e benefícios no contexto do plano estratégico e tático da TI e iniciar ações corretivas, quando necessárias.
- PO6 - Comunicar metas e diretivas gerenciais: A administração deve desenvolver uma estrutura de controle empresarial de TI, definir e comunicar suas políticas. Um programa contínuo de comunicação deve ser implementado para articular a missão, objetivos de serviço, políticas e procedimentos etc., aprovados e apoiados pela administração. A comunicação assegura a realização dos objetivos de TI, a conscientização e a compreensão entre o negócio e os riscos da TI, objetivos e direção. O processo deve assegurar a conformidade com leis e regulamentos.
- PO7 - Gerenciar recursos humanos: Uma competente equipe de trabalho é contratada e mantida para a criação e entrega dos serviços de TI para o negócio. Isso é alcançado seguindo-se práticas definidas e acordadas que garantam o recrutamento, treinamento, avaliação de desempenho, promoção e demissão. Esse processo é crítico, as pessoas são um importante trunfo, pois a governança e os controles internos dependem muito da motivação e competência dos funcionários.
- PO8 - Gerenciar qualidade: Um sistema de gerenciamento da qualidade (SGQ) desenvolvido e mantido, no qual inclua a prover desenvolvimento, processos de aquisição e padronizações. Isso é alcançado através do planejamento, implementação e manutenção do SGQ provendo requisitos de qualidade claros, procedimentos e políticas. Requisitos de qualidade devem ser determinados e

comunicados, com indicadores quantificáveis e atingíveis. Melhorias contínuas são atingidas através do monitoramento, análise e agindo em caso de desvios e comunicando os resultados para os *stakeholders*. O gerenciamento da qualidade é essencial para assegurar que a TI está entregando valor ao negócio, melhorias contínuas e transparência para os *stakeholders*.

- PO9 - Avaliar e gerenciar riscos de TI: Criar e manter uma estrutura de gerenciamento de riscos. A estrutura deve documentar um acordo comum sobre os níveis de riscos da TI, estratégias de mitigação e riscos residuais. Qualquer impacto potencial sobre as metas da organização, causada por eventos não planejados devem ser identificados, analisados e avaliados. Estratégias de mitigação de riscos devem ser adotadas para minimizar os riscos residuais para um nível aceitável. O resultado da avaliação deve ser compreensível para os *stakeholders* e expresso em termos financeiros, para permitir os *stakeholders* alinharem os riscos em um nível aceitável de tolerância.
- PO10 - Gerenciar projetos: Estabelecer um modelo de gerenciamento de programas e projetos para gerenciar todos os projetos de TI. Este modelo deve assegurar a correta priorização e coordenação de todos os projetos. O modelo deve incluir um plano mestre, atribuição de recursos, definição de entregáveis, aprovações pelos usuários, uma divisão em fases para as entregas, garantia de qualidade, um plano formal de testes, testes e revisões pós-implementação após a instalação para assegurar o gerenciamento de riscos e a entrega de valor para o negócio.

b) Domínio (AI) Adquirir e Implementar

As soluções de TI precisam ser identificadas, desenvolvidas ou adquiridas, implementadas e mantidas de forma integrada aos processos do negócio. Segundo o ITGI (2007), este domínio normalmente é dirigido às seguintes questões gerenciais:

- Novos projetos são capazes de oferecer soluções que atendam as necessidades da empresa?
- Novos projetos são capazes de serem entregues dentro do prazo e do orçamento?
- Novos sistemas trabalharão satisfatoriamente depois de implementados?
- As mudanças serão executadas sem impactar as operações atuais?

A seguir, um resumo das atividades desse domínio:

- AI1 - Identificar soluções automatizadas: A necessidade por uma nova aplicação ou função requer análises antes da aquisição ou criação para assegurar que os requisitos do negócio sejam satisfeitos de forma efetiva e eficiente. Este processo cobre a definição das necessidades, consideração de alternativas, revisão de viabilidades tecnológica e econômica, execução da análise de risco e custo-benefício e a conclusão da decisão final de adquirir ou criar. Todos estes passos possibilitam a organização, tanto minimizar os custos de adquirir e implementar soluções quanto assegurar que eles permitam ao negócio atingir seus objetivos.
- AI2 - Adquirir e manter aplicativos de software: Aplicações devem estar disponíveis e alinhadas com os requisitos do negócio. Este processo cobre o desenho das aplicações, a inclusão apropriada dos requisitos de controle e segurança, o desenvolvimento e configurações alinhadas aos padrões.
- AI3 - Adquirir e manter a infraestrutura tecnológica: A organização deve ter um processo para aquisição, implementação e atualização da infraestrutura tecnológica. Isso requer um planejamento alinhado com as estratégias tecnológicas além de ambientes para desenvolvimento e testes.
- AI4 - Permitir operação e uso: Conhecimentos sobre novos sistemas são disponibilizados. Este processo requer a produção de

documentação e manuais para usuários e TI, prover treinamento para assegurar o apropriado uso e operação das aplicações e infraestrutura.

- AI5 - Obter recursos de TI: Recursos de TI incluindo pessoas, hardware, software e serviços precisam ser obtidos. Isso requer uma definição de procedimentos para aquisições, a seleção de fornecedores e definição de termos contratuais. Dessa maneira assegura-se que a organização tenha todos os recursos necessários de TI, de forma eficiente em tempo e custo.
- AI6 - Gerenciar mudanças: Todas as mudanças, incluindo manutenções de emergência e correções relacionadas à infraestrutura e aplicações dentro do ambiente de produção, precisam ser gerenciadas de uma forma controlada. As mudanças precisam ser registradas, avaliadas e autorizadas antes da implementação e os resultados devem ser revisados logo em seguida. Isso assegura a mitigação de riscos e impactos negativos sobre a estabilidade ou integridade do ambiente de produção.
- AI7 - Instalar e certificar soluções e mudanças: Novos sistemas precisam ser operacionalizados após a conclusão do desenvolvimento. Isso requer testes apropriados em um ambiente dedicado com uma base de dados relevante para testes, definições de restauração e instruções de migração, planejamento de liberação e implantação na produção, revisões pós-implementação. Isso assegura que sistemas operacionais estarão alinhados com as expectativas e resultados acordados.

c) Domínio (DS) Entregar e Suportar

Este domínio preocupa-se com a prestação de serviços, gestão da segurança e continuidade, suporte aos usuários, gerenciamento de dados e instalações operacionais. Segundo o ITGI (2007), este domínio normalmente é dirigido às seguintes questões gerenciais:

- Os serviços de TI são entregues de acordo com as prioridades do negócio?

- Os custos de TI estão otimizados?
- A força de trabalho é capaz de utilizar os sistemas de forma produtiva e segura?
- A segurança da informação é adequada quanto à confidencialidade, integridade e disponibilidade?

A seguir, um resumo das atividades desse domínio:

- DS1 - Definir e gerenciar níveis de serviço: uma comunicação eficaz entre a gerência de TI e os clientes do negócio, em relação aos serviços requeridos, é determinada por meio de um documento de serviços e níveis de serviços (SLAs). Este processo inclui o monitoramento e relatórios atualizados para os *stakeholders* sobre o cumprimento dos SLAs. Este processo permite o alinhamento entre os serviços de TI e os requisitos relacionados ao negócio.

- DS2 - Gerenciar serviços terceirizados: A necessidade de garantir que os serviços prestados por terceiros atendam aos requisitos de negócio, requer um efetivo processo de gestão de terceiros. Este processo é realizado através da definição clara de papéis, responsabilidades e expectativas em acordos com terceiros, bem como revisão e acompanhamento desses acordos. Uma gestão eficaz dos serviços de terceiros minimiza os riscos associados a terceiros não qualificados.
- DS3 - Gerenciar desempenho e capacidade: A necessidade de gerenciar o desempenho e a capacidade dos recursos de TI requer um processo de revisão periódico de desempenho e capacidade dos recursos da TI. Este processo inclui a previsão de necessidades futuras baseada na carga de trabalho, armazenamento e requisitos de contingência. Este processo assegura que os recursos de informação que sustentam os requisitos do negócio estejam sempre disponíveis.
- DS4 - Garantir a continuidade dos serviços: A necessidade de prover serviços de TI de forma contínua requer o desenvolvimento,

manutenção e teste de planos de continuidade de TI, utilizando *backup* de dados em outra localidade e realizar treinamentos periódicos para o plano de continuidade. Um eficiente processo de continuidade de serviço minimiza a probabilidade e o impacto de grandes interrupções de serviço sobre as principais funções e processos de negócio.

- DS5 - Garantir a segurança dos Sistemas: A necessidade de manter a integridade das informações e proteger os bens de TI exige um processo de gestão da segurança. Este processo inclui o estabelecimento e a manutenção da segurança da TI com papéis e responsabilidades, políticas, normas e procedimentos. A gestão de segurança inclui também monitorar a segurança, realizar testes periódicos e implementar ações corretivas para as deficiências de segurança ou incidentes. A gestão da segurança eficaz protege todos os ativos de TI a fim de minimizar o impacto das vulnerabilidades e incidentes no negócio.
- DS6 - Identificar e Alocar Custos: A necessidade por uma justa e equitativa alocação dos custos de TI para o negócio requer medição precisa dos custos de TI e de um acordo com os usuários para uma divisão justa. Este processo inclui a construção e operação de um sistema para captar, alocar e relatar os custos de TI para os usuários. Um sistema justo de distribuição permite ao negócio tomar decisões mais orientadas em relação à utilização dos serviços de TI.
- DS7 - Educar e treinar usuários: Educação eficaz de todos os usuários de TI, incluindo o pessoal de TI, requer identificar as necessidades de treinamento para cada grupo de usuários. Além de identificar as necessidades, este processo inclui a definição e execução de uma estratégia para treinamento e avaliação de resultados. Um programa eficaz de formação melhora a utilização da tecnologia pelos usuários reduzindo erros, aumentando a produtividade e a conformidade com os principais controles.

- DS8 - Gerenciar central de serviços (*service desk*) e incidentes: Respostas rápidas e eficazes às consultas e problemas dos usuários de TI exigem uma central de serviços e um processo de gestão de incidentes bem concebido e executado. Este processo inclui a criação de uma central de serviços com a função de registrar incidentes, tendências, análise das causas e resoluções. Os benefícios incluem o aumento da produtividade das empresas por meio da rápida resolução das demandas dos usuários.
- DS9 - Gerenciar a configuração: Assegurar a integridade das configurações de hardware e software requer o estabelecimento e a manutenção de um completo repositório de dados da configuração. Este processo inclui a coleta de informações da configuração inicial, que estabelece a linha base, verificação e auditoria do repositório de informações da configuração e atualização do repositório conforme necessário. Um sistema de gerenciamento de configuração eficaz possibilita maior disponibilidade dos sistemas, minimiza problemas na produção e os resolve mais rapidamente.
- DS10 - Gerenciar problemas: Uma Gestão de problemas eficaz requer a identificação e classificação dos problemas, análise das causas e a resolução dos problemas. O processo de gestão de problemas inclui a formulação de recomendações de melhorias, manutenção de registro dos problemas e avaliação das ações corretivas. Um processo eficaz da gestão de problemas maximiza a disponibilidade dos sistemas, melhora os níveis de serviço, reduz custos e melhora a satisfação dos clientes.
- DS11 - Gerenciar Dados: Uma gestão eficaz de dados exige identificar os requisitos de dados. O processo de gerenciamento de dados inclui a criação de procedimentos eficazes para a biblioteca de mídias, *backup* e recuperação de dados. A Gestão eficaz dos dados ajuda garantir qualidade, oportunidade e disponibilidade dos dados para o negócio.

- DS12 - Gerenciar os ambientes físicos: A proteção para equipamentos e pessoal de informática requer instalações físicas bem concebidas e gerenciadas. O processo de gestão do ambiente físico inclui a definição dos requisitos para o lugar físico, escolha de instalações adequadas, conceber processos eficazes para monitoração do ambiente e gestão do acesso físico. Uma gestão eficaz do ambiente físico reduz interrupções nos negócios devido a danos nos equipamentos e pessoal de informática.
- DS13 - Gerenciar as operações: Processamento de dados completos e precisos requer uma gestão eficaz dos processos de tratamento de dados e uma cuidadosa manutenção do hardware. Este processo inclui a definição de políticas e procedimentos para o funcionamento eficaz do gerenciamento da agenda de processamento, protegendo processamentos importantes, acompanhar o desempenho da infraestrutura e garantir a manutenção preventiva do hardware. A gestão eficaz das operações ajuda a manter a integridade dos dados e reduzir os atrasos nas operações e os custos operacionais de TI.

d) Domínio (ME) Monitorar e Avaliar

Todos os processos de TI precisam ser avaliados continuamente pelas suas qualidades e cumprimento dos requisitos de controle. Este domínio visa o desempenho da gestão, monitoramento de controles internos, conformidades regulatórias e de governança. Segundo o ITGI (2007), este domínio normalmente é dirigido às seguintes questões gerenciais:

- O desempenho de TI é medido para detectar problemas antes que seja muito tarde?
- A gerência está segura que os controles internos são eficazes e eficientes?
- O desempenho de TI pode ser vinculado ao do negócio?
- Os controles para confidencialidade, integridade e disponibilidade são adequados para a segurança da informação?

A seguir, um resumo das atividades desse domínio (ME):

- ME1 - Monitorar e avaliar o desempenho de TI: A gestão eficaz do desempenho de TI requer um processo de acompanhamento. Este processo inclui a definição de indicadores relevantes, sistemática e relatórios periódicos de desempenho, e ações rápidas em caso de desvios. O acompanhamento é necessário para assegurar que as coisas certas são feitas e estão alinhadas com as direções e políticas definidas.
- ME2 - Monitorar e avaliar controles internos: Estabelecer um efetivo programa de controle interno para TI requer processos de monitoração bem definidos. Estes processos incluem a monitoração, relatórios para controle de exceções, resultados de autoavaliação e revisões de terceirizados. Um dos principais benefícios da monitoração de controles internos é fornecer segurança relacionada à eficiência e eficácia das operações e conformidade com leis e regulamentos.
- ME3 - Garantir conformidade com requisitos externos: Uma fiscalização eficiente requer um processo de revisão estabelecido para garantir conformidade com leis, regulamentos e exigências contratuais. Este processo inclui identificar requisitos de conformidade, otimização e avaliação da resposta, obtendo segurança de que os requisitos foram cumpridos e integrando os relatórios de conformidade de TI com o restante do negócio.
- ME4 - Prover governança de TI: Estabelecer uma estrutura efetiva de governança inclui definir estruturas organizacionais, processos, liderança, papeis e responsabilidades para assegurar que os investimentos em TI estejam alinhados e entregues de acordo com as estratégias e objetivos empresariais.

1.5.2. CMMI

O CMMI (*Capability Maturity Model Integration*) é um modelo de referência que contém práticas (Genéricas ou Específicas) necessárias à maturidade em

disciplinas específicas (*Systems Engineering* - SE, *Software Engineering* - SW, *Integrated Product and Process Development* - IPPD, *Supplier Sourcing* - SS). Desenvolvido pelo Software Engineering Institute (SEI) da Universidade Carnegie Mellon, o CMMI é uma evolução do CMM e procura estabelecer um modelo único para o processo de melhoria corporativo, integrando diferentes modelos e disciplinas.

O CMMI foi baseado nas melhores práticas para desenvolvimento e manutenção de produtos. Há uma ênfase tanto em engenharia de sistemas quanto em engenharia de software, e há uma integração necessária para o desenvolvimento e a manutenção.

A versão atual do CMMI (versão 1.3) foi publicada em 27 de outubro de 2010 e apresenta três modelos:

- *CMMI for Development* (CMMI-DEV), voltado ao processo de desenvolvimento de produtos e serviços;
- *CMMI for Acquisition* (CMMI-ACQ), voltado aos processos de aquisição e terceirização de bens e serviços;
- *CMMI for Services* (CMMI-SVC), voltado aos processos de empresas prestadoras de serviços.

Uma das premissas do modelo é "A qualidade é influenciada pelo processo", e seu foco é "Melhorar processo de uma empresa".

1.5.3. ISO/IEC 15408-1

Segundo Albuquerque e Ribeiro (2002), o objetivo da norma ISO/IEC 15.408, também chamada de *Common Criteria* devido ao padrão de mercado que deu origem a essa norma (*Common Criteria for Information Technology Security Evaluation*), é fornecer um conjunto de critérios fixos que permitem especificar a segurança de uma aplicação de forma não ambígua a partir de características do ambiente da aplicação, e definir formas de garantir a segurança da aplicação para o cliente final.

O *Common Criteria* é uma norma ou padrão de indústria criado a partir da união dos diversos padrões anteriores com o objetivo de gerar uma norma internacional no assunto. A primeira versão do *Common Criteria* foi lançada em janeiro de 1996. Em maio de 1998 foi liberada uma grande revisão, denominada *Common Criteria* 2.0. Finalmente, em dezembro de 1999 a versão 2.1 do *Common Criteria* foi homologada como a norma internacional ISO/IEC 15408. Até o momento da realização desse trabalho, ainda não estava disponível a versão brasileira (NBR) da norma ISO 15408.

Segundo Albuquerque e Ribeiro (2002), a ideia do *Common Criteria* é a de que se pode desenvolver um sistema, seguindo a norma, testá-lo em um laboratório credenciado e obter um selo de certificação de que o sistema atende aos requisitos de segurança eliminando-se ou mitigando-se, com isso, o risco para o cliente na compra do sistema.

O *Common Criteria* estabelece que qualquer sistema, para ser considerado seguro, precisa ter seu objetivo de segurança (*Security Target*) elaborado. O *Security Target* é a especificação de segurança, ou seja, indica quais aspectos de segurança foram considerados importantes e por que o foram para aquele sistema em particular.

Há ainda a figura do perfil de proteção (*Protection Profile*), que é um documento semelhante ao *Security Target*, porém sem especificar uma única aplicação, podendo ser aplicado a toda uma classe de sistemas.

O *Common Criteria* define também sete níveis de garantia de segurança. A cada nível, há um número maior de testes e, portanto, maior garantia de que o sistema atende aos requisitos de segurança. Esses níveis são denominados EAL (*Evaluation Assurance Level*, ou nível de garantia da avaliação), e podem ser de EAL-1 a EAL-7, conforme o *Common Criteria*. Na ISO/IEC 15408, os níveis EAL-5 a EAL-7 foram considerados extremamente rígidos e, na prática, inviáveis. Assim, apenas os níveis EAL-1 a EAL-4 são reconhecidos pela ISO.

A ISO/IEC 15408 e o *Common Criteria* definem e mantêm uma rede de laboratórios credenciados a avaliar a segurança das aplicações em determinado

nível de garantia (EALs 1 a 4 na ISO/IEC, EALs 1 a 7 no *Common Criteria*), os mais acessíveis se encontram nos Estados Unidos.

Para evitar a confusão com o termo sistema, que tanto pode indicar o sistema que está sendo avaliado quanto outro sistema qualquer, o *Common Criteria* emprega o termo TOE (*Target of Evaluation*, ou alvo da avaliação). O TOE é o sistema que estamos desenvolvendo ou avaliando. Esse sistema tem funções de segurança, que são o conjunto de rotinas responsáveis por garantir a política de segurança do TOE. Algumas vezes as funções de segurança são designadas pela sigla TSF (*TOE Security Functionality*, ou alvo da avaliação da funcionalidade de segurança). A política de segurança do TOE são as regras que definem a política de acesso e controle dos ativos do sistema.

Além de ajudar na definição da especificação da segurança, a ISO/IEC 15408 também define meios para o desenvolvedor garantir a segurança da aplicação gerada, e para o cliente avaliar a segurança deste.

1.5.4. MPS.BR

O MPS.BR – Melhoria de Processo do Software Brasileiro (SOFTEX, 2012) foi criado por pesquisadores brasileiros para a melhoria do processo de desenvolvimento de software em empresas brasileiras. Trata-se de um modelo em contínuo desenvolvimento, que começou a ser desenvolvido em 2003, pelas instituições SOFTEX, Riosoft, COPPE/UFRJ, CESAR, CenPRA e CELEPAR.

O foco principal são as micro, pequenas e médias empresas de software brasileiras que possuem poucos recursos para melhoria de processos, mas que estão diante de fazê-lo. O MPS.BR atende à necessidade de implantar os princípios de engenharia de software de forma adequada ao contexto das empresas brasileiras, seguindo as principais abordagens internacionais para definição, avaliação e melhoria de processos de software.

A descrição do modelo baseia-se em três guias:

- Guia geral: contém a descrição geral do MPS.BR e detalha o modelo de referência (MR-MPS), seus componentes e as

definições comuns necessárias para seu entendimento e aplicação;

- Guia de aquisição: contém recomendações para a condução de compras de software e serviços correlatos. Foi elaborado para guiar as instituições que irão adquirir produtos de software e serviços;
- Guia de avaliação: contém a descrição do processo de avaliação, os requisitos para o avaliador e para a avaliação, o método e os formulários para guiar a avaliação.

O modelo tem algumas diferenças em relação a outras possibilidades. Segundo o MPS.BR (SOFTEX, 2012), esses diferenciais são:

- sete níveis de maturidade, o que possibilita uma implantação mais gradual e adequada a pequenas empresas;
- compatibilidade plena com CMMI e com a norma ISO/IEC 15504;
- o modeo foi criado para o Brasil, onde a maioria das empresas que desenvolvem software são micro, pequenas e médias;
- custo acessível;
- avaliação bienal das empresas;
- forte interação Universidade-Empresa.

A base utilizada na construção do MPS.BR são as normas NBR ISO/IEC 12207, ISO/IEC 15504 e o CMMI (Figura 1.3).

O MPS.BR está dividido em três componentes:

- Modelo de Referência (MR-MPS), que contém os requisitos a serem cumpridos pelas organizações que desejam estar em conformidade com o MR-MPS. Tem definições dos níveis de maturidade, da capacitação dos processos e dos processos, com base nas normas NBR ISO/IEC 12207 e suas emendas 1 e 2, na ISO/IEC 15504 e é adequado ao CMMI-SE/SW. O Guia de

Aquisição é um documento complementar para organizações que pretendem adquirir software e serviços correlatos;

- Método de Avaliação (MA-MPS) contém o processo de avaliação, os requisitos para os avaliadores e os requisitos para averiguação da conformidade ao modelo MR-MPS. Ele está descrito de forma detalhada no Guia de Avaliação e foi baseado no documento ISO/IEC 15504;
- Modelo de Negócio (MN-MPS) contém uma descrição das regras para a implementação do MR-MPS pelas empresas de consultoria, de software e de avaliação.

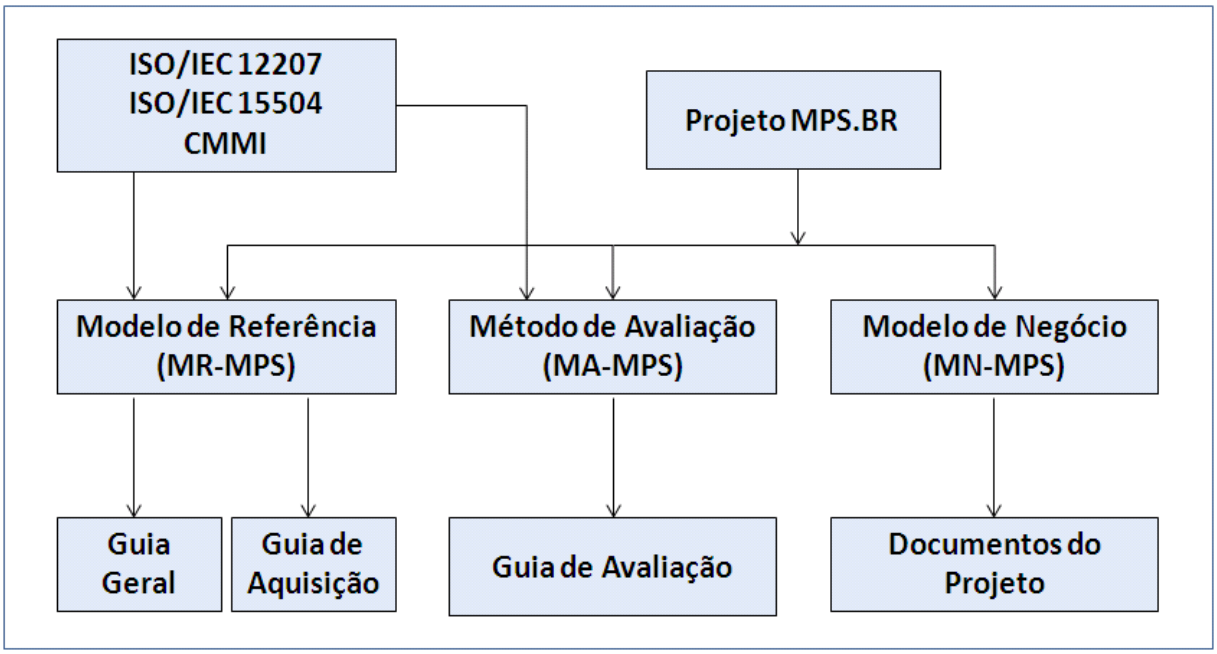

Figura 1.3 – Estrutura do MPS.BR (KOSCIANSKI, et al., 2007)

Segundo Koscianski, et al. (2007), o Modelo de Referência MR-MPS define níveis de maturidade que são uma combinação de processos e capacitação de processos. O nível de maturidade em que se encontra uma organização permite prever seu desempenho futuro. Os objetivos de melhoria foram divididos em sete níveis de maturidade, de A (melhor) a G (pior), e para cada nível, definiu-se um perfil de processos e um perfil de capacitação de processos, vide Tabela 1.1.

Tabela 1.1 – Níveis de maturidade e processos (KOSCIANSKI, et al., 2007)

A	Em otimização	Inovação e implantação na organização
		Análise de causas e resolução
B	Gerenciado Quantitativamente	Desempenho do processo organizacional
		Gerência quantitativa do projeto
C	Definido	Análise de decisão e resolução
		Gerência de riscos
D	Largamente definido	Desenvolvimento de requisitos
		Solução técnica
		Integração de produto
		Instalação de produto
		Liberação de produto
		Verificação
		Validação
E	Parcialmente definido	Treinamento
		Avaliação e melhoria do processo organizacional
		Definição do processo organizacional
		Adaptação do processo para gerência de projeto
F	Gerenciado	Medição
		Gerência de configuração
		Aquisição
		Garantia da qualidade
G	Parcialmente gerenciado	Gerência de requisitos
		Gerência de projeto

1.5.5. NBR ISO/IEC 12207

A norma internacional ISO/IEC 12207 (ABNT, 2009) tem como objetivo principal estabelecer uma estrutura comum para os processos de ciclo de vida e de desenvolvimento de softwares visando ajudar as organizações a compreenderem todos os componentes presentes na aquisição e fornecimento de software e, assim, conseguirem firmar contratos e executarem projetos de forma mais eficaz.

De acordo com esta norma, o processo de software envolve métodos, técnicas, ferramentas e pessoas. Um processo pode ser descrito de duas formas: por propósito ou resultado e por atividade.

A descrição por propósito ou resultado é utilizada quando não há necessidade de detalhar o processo, apenas indicar o objetivo e o resultado. Essa abordagem poderá ser utilizada na avaliação do processo em relação aos modelos de maturidade de software como, por exemplo, o modelo CMMI e o modelo da ISO/IEC 15504.

A descrição por atividade é a abordagem mais conhecida e intuitiva. Nela são descritas as atividades com as inter-relações e o algoritmo de execução de cada atividade. As atividades devem atingir o propósito do processo. Para isso deve adotar as premissas:

- Que procedimentos e métodos serão usados para a execução das atividades;
- Que ferramentas e equipamentos suportarão a realização das atividades, de forma a simplificar e automatizar o trabalho;
- Qual o perfil adequado de quem irá executar as atividades e qual o treinamento requerido nos procedimentos, métodos, ferramentas para que se possam realizar as atividades de forma adequada;
- Quais as métricas de processo que poderão ser empregadas para que a execução do processo possa ter a qualidade avaliada.

A norma NBR ISO/IEC 12207 estabelece uma arquitetura de alto nível do ciclo de vida de software que é construída a partir de um conjunto de processos e seus inter-relacionamentos. Os processos são descritos tanto em nível de propósito/saídas como em termos de atividades. Essa norma não tem nenhuma ligação com métodos, ferramentas, treinamentos, métricas ou tecnologias empregadas. Esta determinação é útil para permitir que a norma seja utilizada mundialmente e possa acompanhar a evolução da engenharia de software nas diversas culturas organizacionais. Ela pode ser utilizada com qualquer modelo de ciclo de vida, método ou técnica de engenharia de software e linguagem de

programação. Sua flexibilidade é uma característica importante, as atividades e tarefas do processo de ciclo de vida do software especificam "o que fazer" e não "como fazer".

Os processos da NBR ISO/IEC 12207 são modulares, ou seja, são fortemente coesos e fracamente acoplados. Isto significa que todas as partes de um processo são fortemente relacionadas, mas a quantidade de interfaces entre os processos é mínima.

As regras listadas a seguir são importantes para identificação, escopo e estruturação dos processos e devem ser seguidas.

- Um processo deve ser modular, isto é, convém que um processo execute uma e somente uma função dentro do ciclo de vida e é conveniente que as interfaces entre dois processos quaisquer sejam mínimas;
- Cada processo é invocado na arquitetura;
- Se um processo A é invocado por um processo B e somente por ele, então A pertence a B;
- Se uma função é invocada por mais de um processo, então esta função torna-se um processo;
- Deve ser possível verificar qualquer função dentro do modelo de ciclo de vida;
- Convém que cada processo tenha uma estrutura interna suficientemente definida para que possa ser executável.

Os processos na NBR ISO/IEC 12207 são de responsabilidade de uma organização, mas não são exclusivos desta, ou seja, uma organização pode executar um ou mais processos e um processo pode ser executado por uma ou mais organizações.

Neste caso, uma das organizações será a responsável pelo processo total, mesmo que tarefas individuais sejam realizadas por pessoas diferentes. Os processos são agrupados, por uma questão de organização, de acordo com

a sua natureza, ou seja, o seu objetivo principal no ciclo de vida de software. Esse agrupamento resultou em três categorias de processos, que são:

a) Processos primários

São os processos básicos que se relacionam aos produtos de software. Abrangem, por exemplo, o desenvolvimento, operação e manutenção. São eles:

- Aquisição: tem o propósito de obter o produto e/ou serviço que satisfaça as necessidades do cliente;
- Fornecimento: tem o propósito de prover um produto e/ou serviço;
- Desenvolvimento: tem o propósito de transformar um conjunto de requisitos em um produto ou sistema de software;
- Operação: tem o propósito de operar o produto no seu ambiente e prover suporte aos usuários;
- Manutenção: tem o propósito de modificar o produto de software e depois dar liberação para o uso.

b) Processos de apoio

Os processos dessa categoria tem lugar, em geral, depois que um processo primário é iniciado. São eles:

- Documentação: tem o propósito de prover e manter um registro de informações de software;
- Gerência de configuração: tem o propósito de estabelecer e manter a integridade de todos os produtos de trabalho (artefatos) de um processo do projeto;
- Garantia da qualidade: tem o propósito de prover garantia de que os produtos e processos estão em conformidade com os requisitos (padrões/normas) pré-definidos;
- Verificação: tem o propósito de confirmar que os produtos e/ou serviços refletem os requisitos especificados;
- Validação: tem o propósito de confirmar que os requisitos para o uso específico de um produto e/ou serviço são atendidos;

- Revisão conjunta: tem o propósito de avaliar as atividades de um processo do ciclo de vida, examinando-se tanto as atividades em si quanto eventuais artefatos por ela produzidos;
- Auditoria: tem o propósito de determinar independentemente a conformidade dos produtos e processos contra os requisitos definidos;
- Resolução de problemas: tem o propósito de assegurar que todos os problemas levantados sejam analisados e resolvidos.

c) Processos organizacionais

Auxiliam a organização e gerência geral dos processos e podem ser empregados fora do domínio de projetos e contratos específicos, servindo para toda a organização. São eles:

- Gerência: tem o propósito de organizar, monitorar e controlar a iniciação e o desempenho dos processos;
- Infraestrutura: tem o propósito de manter uma infraestrutura estável e confiável;
- Melhoria: tem o propósito de estabelecer, avaliar, controlar e melhorar um processo de ciclo de vida de software;
- Treinamento: tem o propósito de apontar as atividades para preparação e realização de treinamentos.

1.5.6. NBR ISO/IEC 15504

O SPICE, acrônimo para *Software Process Improvement and Capability Determination*, é uma norma elaborada pela ISO/IEC objetivando tornar-se um padrão para a avaliação do processo de software e para determinar a capacitação de uma organização. A norma apresenta uma estrutura para avaliações de processos em organizações, e pode ser aplicada em circunstâncias como:

- uma empresa que busca melhorias internas;
- avaliação de terceiros ao realizarem contratos de prestação de serviços ou fornecimento de produtos.

Em junho de 1991 a ISO/IEC aprovou um período de estudo, por sua resolução ISO/IEC JTC1/SC7 n. 144, para a investigação das necessidades e requisitos exigidos à padronização da avaliação dos processos relacionados à criação e uso de software.

Em 1993 constituiu-se e teve início o projeto SPICE (*Software Process Improvement and Capability Determination*), com três objetivos básicos: auxiliar o início do projeto de norma, executar testes de campo, para obter dados de experiências práticas e despertar o mercado para o surgimento da futura norma.

O projeto SPICE acabou associando definitivamente o acrônimo à norma ISO/IEC 15504. Esse projeto foi oficialmente encerrado em março de 2003, sendo substituído pelo *SPICE Network*. Segundo Koscianski, et al. (2007), a versão inicial da “norma SPICE” foi publicada em 1998 e era apenas um relatório técnico, um estágio anterior ao de norma internacional. Essa versão era focada exclusivamente em software, baseando-se nos processos da ISO/IEC 12207, mas acrescentando outros como integração de software, e era dividida em um total de nove partes.

A versão atual apresenta várias mudanças: é uma norma internacional, é composta de sete partes, é genérica, não sendo mais exclusivamente dedicada a software e introduziu o conceito de modelo de referência de processo.

Conforme Koscianski, et al. (2007), para ser aplicada a software, a norma ISO/IEC 15504 deve ser complementada pela ISO/IEC 12207 e, mais especificamente, pelas extensões (*amendment*) AMD1 e AMD2. A ISO/IEC 15504 está dividida da forma apresentada na Tabela 1.2.

Tabela 1.2 – Divisão da norma ISO/IEC 15504 (ABNT, 2008)

Parte	Descrição
1	Conceitos e vocabulário
2	Realização de uma avaliação
3	Orientações para realização de uma avaliação
4	Orientação no uso para melhoria do processo e determinação da potencialidade do processo
5	Um exemplo de Modelo de Avaliação de Processo
6	Exemplo de modelo de avaliação de processo de ciclo de vida de sistema
7	Avaliação da maturidade de uma organização

A ISO/IEC 15504 pode ser aplicada no melhoramento ou na avaliação dos processos. Em ambos os casos, três elementos devem ser precisamente definidos: os processos, uma escala de medida e um método de medição.

Diferentemente da versão antiga, a atual ISO/IEC 15504 não mais define os processos, mas sim um conceito chamado modelo de referência de processo (*Process Reference Model* - PRM), o qual contém uma descrição de escopo e uma descrição de requisitos. Tais requisitos estabelecem os resultados esperados da execução de cada processo e permitem avaliar se os objetivos do processo serão alcançados. A extensão um da norma ISO/IEC 12207 traz um PRM aplicável com a ISO/IEC 15504.

Para realizar uma medição, define-se um modelo de medição (PAM – *Process Assessment Model*), que identifica elementos da organização a serem examinados. Para cada processo ele define dois indicadores: práticas base (BP – *Base Practice*) e artefatos produzidos (WP – *Working Products*).

Tais elementos são utilizados na chamada dimensão de processo e permitem verificar se os processos são ou não executados (*assessment of process performance*). Outra dimensão de avaliação é a dimensão de capacidade (*assessment of process capability*). O PAM define seis níveis de capacidade para os processos, conforme apresenta a Tabela 1.3.

Tabela 1.3 – Níveis de capacitação da ISO/IEC 15504 (ABNT, 2008)

Nível	Nome	Descrição
0	Incompleto	O processo não é implementado ou falha em atingir seus objetivos
1	Executado	O processo essencialmente atinge os objetivos, mesmo se de forma pouco planejada ou rigorosa
2	Gerenciado	O processo é implementado de forma controlada (planejado, monitorado e ajustado); os produtos por ele criados são controlados e mantidos de forma apropriada
3	Estabelecido	O processo é implementado de forma sistemática e consistente
4	Previsível	O processo é executado e existe um controle que permite verificar se ele se encontra dentro dos limites estabelecidos para atingir os resultados
5	Otimizado	O processo é adaptado continuamente para, de uma forma mais eficiente, atingir os objetivos de negócio definidos e projetados.

A dimensão de execução de processo relaciona-se apenas ao nível 1, enquanto a dimensão de capacidade engloba todos os níveis.

a) Dimensão de processo

Ao realizar uma avaliação sob a dimensão de processo, estes são divididos em cinco categorias:

- **CON**: consumidor e fornecedor;
- **ENG**: engenharia;
- **SUP**: suporte;
- **MAN**: administração;
- **ORG**: organização.

Os processos da categoria CON têm um impacto direto sobre os consumidores. Incluem, por exemplo, o levantamento de requisitos e os processos ligados à operação e uso do produto ou do serviço fornecido.

Na categoria ENG são agrupados os processos que levam à implementação do produto. Incluem, entre outros itens: a análise de requisitos, o projeto da arquitetura, construção, integração e testes do produto.

Os processos da categoria SUP dão apoio (*support*) aos demais processos da organização. Exemplos são revisões, auditorias e processos de solução de problemas.

Na categoria de gerência MAN estão incluídos os processos que, de forma genérica, podem ser usados na administração de todo outro processo ou do projeto em si.

Finalmente, os processos organizacionais ORG dizem respeito ao funcionamento da empresa como um todo e incluem, entre outros itens, infraestrutura, gerência de recursos humanos e treinamentos.

b) Dimensão de capacidade

A dimensão de capacidade permite uma avaliação mais detalhada dos processos executados por uma organização. Enquanto a dimensão de processo se

limita à verificação da execução ou não dos processos, a dimensão de capacidade leva a uma avaliação de níveis semelhantes aos do CMMI.

Para avaliar sob essa ótica, o PAM inclui a definição de atributos de processo, que permitem avaliá-los em uma escala. Há um total de nove atributos agrupados em níveis de capacidade, que são aplicáveis a todos os processos. Os atributos são apresentados na Tabela 1.4.

Tabela 1.4 – Atributos de processos (ABNT, 2008)

Nível	Atributo
1	1.1 Execução
2	2.1 Administração do processo 2.2 Administração dos produtos obtidos do processo
3	3.1 Definição 3.2 Implementação
4	4.1 Medição 4.2 Controle
5	5.1 Inovação 5.2 Otimização

Os atributos são brevemente descritos na Tabela 1.5 a seguir, sendo destacados apenas os aspectos principais.

Tabela 1.5 – Principais aspectos dos atributos de processos (ABNT, 2008)

Atributo	Descrição
1.1 – Execução	O processo atinge os objetivos dele esperados.
2.1 – Administração	Quando plenamente atingido, este atributo indica que os objetivos do processo são identificados e sua execução planejada. Responsabilidades são atribuídas, a infraestrutura é fornecida e a comunicação entre os envolvidos é gerenciada.
2.2 – Administração de produto	Requisitos para os produtos do processo são definidos. Tais produtos são identificados e documentados. Revisões e ajustes são efetuados conforme necessário.
3.1 – Definição	Um processo padronizado é definido pela organização. Deve incluir, entre outros elementos, a identificação de competências e papéis, da infraestrutura e das interações com outros processos.

Atributo	Descrição
3.2 – Implementação	Os elementos identificados em 3.1 são postos em prática. Por exemplo, a infraestrutura é implementada. Todos os recursos necessários são alocados e, se necessário, treinamentos são realizados.
4.1 – Medição	Estabelecem-se objetivos quantitativos, bem como as medições a serem realizadas e a frequência de sua aplicação. Os resultados são coletados, analisados e publicados na organização.
4.2 – Controle	Estabelecem-se limites de variação para as medidas, bem como ações corretivas que tratem as causas de tais variações.
5.1 – Inovação	Objetivos de melhoria são definidos. Oportunidades de melhoria são identificadas nos dados da organização e na aplicação de novas tecnologias.
5.2 – Otimização	O desempenho dos processos é medido e o impacto das melhorias propostas é comparado aos objetivos esperados. A implementação de mudanças é gerenciada.

Cada atributo de um processo possui associado um conjunto de indicadores de quatro tipos diferentes:

- Os indicadores de práticas genéricas (GPI) relacionam-se a atividades genéricas, que guiam a implementação de um atributo.
- Os indicadores de recursos genéricos (GRI) mostram recursos que podem ser utilizados pelo processo. Exemplos são recursos humanos, ferramentas ou outros elementos de infraestrutura.
- Os indicadores de produto genérico (GWI) referem-se aos produtos da execução de um processo, como documentos, código, componentes ou outros artefatos.
- Os indicadores de processos relacionados (RPI) fazem ligação com outros processos da organização que são necessários para alcançar os objetivos do atributo considerado.

Os atributos de um processo podem ser avaliados em quatro níveis qualitativos. Cada nível é mapeado a uma faixa de valores porcentuais variando de 0%, quando o atributo não é alcançado, a 100%, quando é completamente satisfeito.

A Tabela 1.6 apresenta os níveis e as faixas porcentuais correspondentes.

Tabela 1.6 – Escala de capacidade dos atributos de processos (ABNT, 2008)

Porcentagem	Nível	Descrição
0% a 15%	N *(not)* Não atingido	Existe pouca ou nenhuma evidência de que o atributo do processo seja alcançado
16% a 50%	P (*partially*) Parcialmente atingido	Existe evidência de uma abordagem sistemática significativa para atingir o atributo. Alguns aspectos, como resultados, podem ser imprevisíveis
51% a 85%	L (*largely*) Largamente atingido	O desempenho do processo pode variar em algumas áreas
86% a 100%	F (*fully*) Totalmente atingido	Não há nenhuma falta ou falha significativa

c) Níveis de capacidade

Os nove atributos da Tabela 1.4 devem ser avaliados segundo os quatro indicadores, GPI, GRI, GWI e RPI, utilizando os níveis apresentados na Tabela 1.6. A partir daí, pode-se estabelecer para cada processo um nível de capacidade.

Os níveis de capacidade são determinados segundo um mapeamento a partir das escalas de capacidade apresentadas na Tabela 8.5. Esse mapeamento é apresentado na Tabela 1.7.

Tabela 1.7 – Níveis exigidos de capacidade de processo (KOSCIANSKI, et al., 2007)

Atributos	Níveis de capacidade				
	1	2	3	4	5
1.1	L ou F	F	F	F	F
2.1		L ou F	F	F	F
2.2		L ou F	F	F	F
3.1			L ou F	F	F
3.2			L ou F	F	F
4.1				L ou F	F
4.2				L ou F	F
5.1					L ou F
5.2					L ou F

1.5.7. NBR ISO/IEC 27002

A ISO/IEC 27002 (ABNT, 2005), que é uma atualização da ISO/IEC 17799, é uma norma de Segurança da Informação revisada em 2005 pela ISO e pela IEC. A versão original foi publicada em 2000, que por sua vez era uma cópia fiel do padrão britânico (BS) 7799-1:1999.

O padrão é um conjunto de recomendações para práticas na gestão de Segurança da Informação, ideal para aqueles que querem criar, implementar e manter um sistema.

A ISO/IEC 17799 tem como objetivos: confidencialidade, integridade e disponibilidade das informações, os quais são fatores muito importantes para a segurança da informação.

A norma NBR ISO/IEC 27002 está organizada em 11 seções que correspondem a controles de segurança da informação. As boas práticas para o processo de desenvolvimento de software estão na seção 12 – Aquisição, Desenvolvimento e Manutenção de Sistemas de Informação.

Segundo a norma (ABNT, 2005), “Sistemas de informação incluem sistemas operacionais, infraestrutura, aplicações de negócios, produtos de prateleira, serviços e aplicações desenvolvidas pelo usuário”. Por essa razão, os requisitos de segurança de sistemas de informação devem ser identificados e acordados antes do seu desenvolvimento e/ou de sua implementação.

As informações devem ser protegidas visando à manutenção de sua confidencialidade, autenticidade ou integridade por meios criptográficos.

1.5.8. NBR ISO/IEC 9126

A qualidade de um sistema de software pode ser entendida de diversas formas e utilizando diferentes abordagens. A norma ISO/IEC 9126 (ABNT, 2003) estabelece um modelo de qualidade com foco em produtos de software, propondo Atributos de Qualidade, distribuídos em seis características principais, com cada uma delas em subcaracterísticas, vide Figura 1.4:

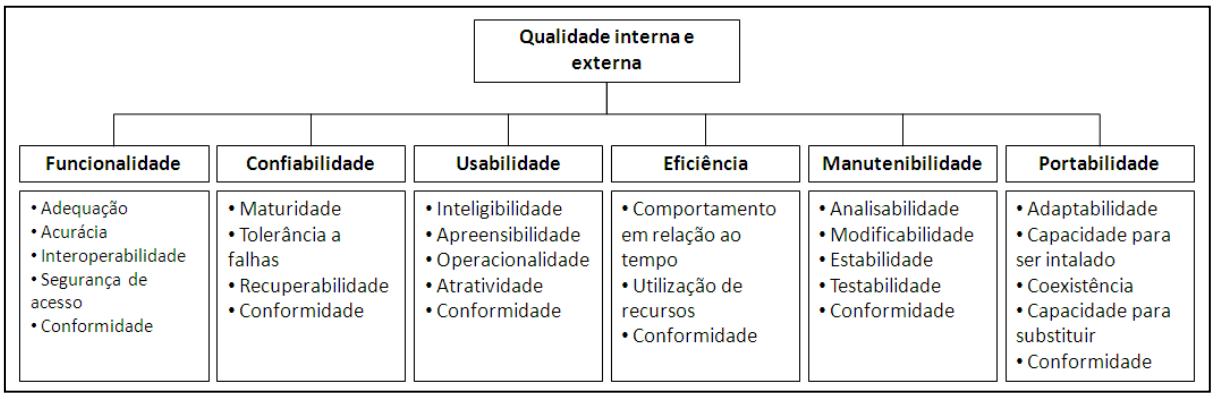

Figura 1.4 – Modelo de qualidade para qualidade externa e interna (ABNT, 2003)

No nível mais alto estão definidas as características de qualidade e nos quadros abaixo as suas subcaracterísticas. Cada característica ou subcaracterística compõe um Atributo de Qualidade do software.

Em todas as características há uma subcategoria com o nome de Conformidade. A conformidade é utilizada para avaliar o quanto o software obedece aos requisitos de legislação e todo o tipo de padronização ou normalização aplicável ao contexto.

a) Funcionalidade

Capacidade de um software prover funcionalidades que satisfaçam o usuário em suas necessidades declaradas e implícitas, dentro de um determinado contexto de uso.

Suas subcaracterísticas são:

- Adequação, que mede o quanto o conjunto de funcionalidades é adequado às necessidades do usuário;
- Acurácia (ou precisão), que representa a capacidade do software de fornecer resultados precisos ou com a precisão dentro do que foi acordado/solicitado;
- Interoperabilidade, que trata da maneira como o software interage com outro(s) sistema(s) especificado(s);
- Segurança, que mede a capacidade do sistema de proteger as informações do usuário e fornecê-las apenas (e sempre) às pessoas autorizadas.

b) Confiabilidade

O produto se mantém no nível de desempenho nas condições estabelecidas. Suas subcaracterísticas são:

- Maturidade, entendida como sendo a capacidade do software em evitar falhas decorrentes de defeitos no software;
- Tolerância a falhas, representando a capacidade do software em manter o funcionamento adequado mesmo quando ocorrem defeitos nele ou nas suas interfaces externas;
- Recuperabilidade, que foca na capacidade de um software se recuperar após uma falha, restabelecendo seus níveis de desempenho e recuperando os seus dados.

c) Usabilidade

Capacidade do produto de software ser compreendido, seu funcionamento aprendido, ser operado e ser atraente ao usuário. Suas subcaracterísticas são:

- Inteligibilidade que representa a facilidade com que o usuário pode compreender as suas funcionalidades e avaliar se o mesmo pode ser usado para satisfazer as suas necessidades específicas;
- Apreensibilidade identifica a facilidade de aprendizado do sistema para os seus potenciais usuários;
- Operacionalidade é como o produto facilita a sua operação por parte do usuário, incluindo a maneira como ele tolera erros de operação;
- Atratividade envolve características que possam atrair um potencial usuário para o sistema, o que pode incluir desde a adequação das informações prestadas para o usuário até os requintes visuais utilizados na sua interface gráfica.

d) Eficiência

O tempo de execução e os recursos envolvidos são compatíveis com o nível de desempenho do software. Suas subcaracterísticas são:

- Comportamento, em relação ao tempo, que avalia se os tempos de resposta (ou de processamento) estão dentro das especificações;
- Utilização de Recursos que mede tanto os recursos consumidos quanto a capacidade do sistema em utilizar os recursos disponíveis.

e) Manutenibilidade

Capacidade (ou facilidade) do produto de software ser modificado, incluindo tanto as melhorias ou extensões de funcionalidade quanto às correções de defeitos, falhas ou erros. Suas subcaracterísticas são:

- Analisabilidade, que identifica a facilidade em se diagnosticar eventuais problemas e identificar as causas das deficiências ou falhas;
- Modificabilidade, que caracteriza a facilidade com que o comportamento do software pode ser modificado;
- Estabilidade, que avalia a capacidade do software de evitar efeitos colaterais decorrentes de modificações introduzidas;
- Testabilidade, que representa a capacidade de se testar o sistema modificado, tanto quanto as novas funcionalidades quanto as não afetadas diretamente pela modificação.

f) Portabilidade

A capacidade de o sistema ser transferido de um ambiente para outro. Suas subcaracterísticas são:

- Adaptabilidade, representando a capacidade do software de se a adaptar a diferentes ambientes sem a necessidade de ações adicionais (configurações);
- Capacidade para ser Instalado identifica a facilidade com que se pode instalar o sistema em um novo ambiente;

- Coexistência mede o quão facilmente um software convive com outros instalados no mesmo ambiente;
- Capacidade para Substituir representa a capacidade que o sistema tem de substituir outro sistema especificado, em um contexto de uso e ambiente específicos. Este atributo interage tanto com adaptabilidade quanto com a capacidade para ser instalado.

1.5.9. Síntese do Capítulo

A auditoria significa a realização de uma atividade sistemática, logicamente ordenada, para avaliação de um processo. Esta avaliação envolve comparar o processo sob exame com outro já conhecido ou com alguma diretriz estabelecida, procurando identificar eventuais desvios e propor, se for o caso, medidas que melhorem o processo analisado.

O ambiente de auditoria compreende avaliação dos controles que permitam à administração a gestão adequada das operações da organização. Quando estes controles estão suportados em uma plataforma computadorizada, a avaliação executada pela auditoria deve envolver também o ambiente de tecnologia de informações.

No contexto deste trabalho, o objetivo considerado é a auditoria durante o desenvolvimento de sistemas, sendo a sua influência exercida pelo estabelecimento de padrões de atuação para acompanhamento das atividades de controles associadas ao processo de desenvolvimento de software. Esta avaliação visa fornecer à administração subsídios para julgar se as atividades de controles existentes são adequadas e se possibilitam acompanhar o andamento do projeto com o fim de mitigar o risco de falhas na entrega do produto.

Há de se considerar que a atuação da auditoria de sistemas pode ser realizada como órgão de linha nas organizações, realizando-se acompanhamento independente sobre projetos de software a partir da avaliação das atividades desses projetos, diferentemente da equipe de *Quality Assurance*, que pertence ao *staff* da equipe de TI. A auditoria de sistemas deve manter sua independência, podendo impactar nos projetos de software identificando pontos falhos e apontando oportunidades de melhoria pontualmente, sem perder sua independência.

2. Projetos de Software

Para melhor compreensão de um projeto de software, que é o tema dessa pesquisa, faz-se necessário inicialmente conceituar software, diferenciá-lo do hardware que o executa e apresentar uma visão geral do processo de software. Além disso, conceituar o Processo Unificado (PU) de desenvolvimento de software, a abordagem considerada na formulação do método de auditoria proposto.

2.1. Conceitos de Software

A origem da palavra software tem várias curiosidades e hoje em dia possui um contexto bem diferente ao qual foi empregada inicialmente. Segundo alguns historiadores, o primeiro uso da palavra software foi encontrado em 1850 e era utilizado por profissionais que tratavam de lixo. A palavra software, nesse contexto, se referia a algo que viria a se decompor com o tempo (KRUSZELNICKI, 2003). Entretanto, nos dias de hoje, esse conceito não passa de uma mera característica do software.

Segundo Pressman (2011), apesar de haver inúmeras formas mais completas de descrever esse significado, podemos classificar o software como instruções (programas de computador) que quando são executadas fornecem as características, função e desempenho desejados, estruturas de dados que permitem aos programas manipular adequadamente a informação e documentos que descrevem a operação e o uso dos programas.

Sommerville (2011) apresenta uma definição mais abrangente, a de que software consiste em uma série de programas separados, arquivos de configuração que são utilizados para configurar esses programas, documentação do sistema, que descreve a estrutura desse sistema, e documentação do usuário, que explica como utilizar o sistema e, no caso de produtos de software, sites WEB para os usuários realizarem o *download* das informações recentes sobre o produto.

Software é definitivamente diferente de hardware, sendo este último tangível e com processo de fabricação semelhante aos produtos industrializados.

Para Pressman (2011), software é mais um elemento de sistema lógico do que físico e tem características que são consideravelmente diferentes daquelas do hardware, pois software é elaborado e não manufaturado.

Por ser um sistema lógico de fácil disseminação, diferencia-se consideravelmente na forma de produção e distribuição, sendo, segundo Pressman (2011), o produto, e ao mesmo tempo, o veículo para entrega do produto.

Sua base é a informação e, sendo assim, pode usar facilmente a maioria dos meios de comunicação, para sua própria distribuição, tornando praticamente infinita a sua forma de armazenamento e disseminação.

Outro ponto importante que diferencia o software de outros produtos é que ele “não se desgasta, mas sim se deteriora” (PRESSMAN, 2011), ou seja, ele não “quebra”, mas apresenta falhas que precisam ser corrigidas. A cada correção, são introduzidos novos pontos de falhas que precisam ser testados para serem descobertos. Apesar da existência de técnicas avançadas de testes, não há garantias de que todo erro será descoberto antes de sua real utilização. Pode-se dizer então que um software se deteriora à medida que sofre alterações, corretivas ou evolutivas, em sua vida útil.

2.2. Engenharia de Software

A Engenharia de Software tem várias definições, além da possibilidade de que cada indivíduo faça um incremento pessoal nesse conceito, considerando desde aspectos concentrados na concepção do software até temas mais focados atualmente, como a qualidade do mesmo.

A etimologia da palavra engenharia, que é um substantivo feminino, vem do latim engenho + ária e, segundo o dicionário Aurélio (FERREIRA, 2006), pode ser vista como a arte de aplicar conhecimentos científicos e empíricos e certas habilitações específicas à criação de estruturas, dispositivos e processos que utilizados para converter recursos naturais em formas adequadas ao atendimento das necessidades humanas. Esse conceito é bem generalista e, até por isso, pode ser considerado também na área computacional.

A primeira definição da Engenharia de Software remonta de 1969, segundo Fritz Bauer (apud PRESSMAN, 2011, p.31) “é a criação e a utilização de sólidos princípios de engenharia a fim de obter softwares econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais”. Seguindo as definições históricas, o IEEE (apud PRESSMAN, 2011, p.31) desenvolveu uma definição mais abrangente que é “(1) aplicação de uma abordagem sistemática, disciplinada e quantificável, para desenvolvimento, operação e manutenção do software, isto é a aplicação da engenharia ao software, (2) Os estudos de abordagens como as de (1)”. Todas essas definições estão inseridas no livro de Pressman (2011), que ressalva que a engenharia de software é uma tecnologia em camadas (Ferramentas, Métodos, Processo e Foco na qualidade).

Sommerville (2011, p.5) tem uma descrição complementar: “A Engenharia de Software é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção deste sistema, depois que ele entrou em operação”.

Embora a Engenharia de Software ofereça técnicas de sucesso para projetos de software, ainda há problemas em produzir software complexo, que atenda às expectativas dos usuários e que seja entregue dentro do prazo e do orçamento estabelecido. Muitos projetos de software ainda têm problemas, e isso levou alguns críticos, como Pressman (2011), a sugerirem que a Engenharia de Software está em um estado de aflição crônica.

À medida que a capacidade do engenheiro de produzir software aumentou, também cresceu a complexidade dos sistemas requeridos. Novas tecnologias que resultam da convergência de sistema de computadores e de comunicação trazem novas questões para os engenheiros de software. Por essa razão e pelo fato de ocorrerem falhas na aplicação das técnicas de Engenharia de Software, são identificados diversos problemas nos projetos de software, o que demonstra haver espaço para o estudo de melhorias na implementação dos projetos de software.

2.3. Processo de Software

O processo de software é o conjunto de áreas que formam a base para o gerenciamento do projeto e estabelecem o contexto em que:

- Os métodos são aplicados;
- O produto é feito;
- Metas são estabelecidas;
- A qualidade é garantida;
- Mudanças podem ser feitas.

Na definição de Sommerville (2011), o processo de desenvolvimento de software é um conjunto de atividades e resultados associados que produz um produto de software. Segundo Pressman (2011), o processo de engenharia de software é o adesivo que mantém unidas as camadas de tecnologia e permite o desenvolvimento racional e oportuno de softwares de computador.

A boa gerência de um projeto de software depende do entendimento do processo em questão, da escolha do método que atenda aos requisitos do produto e das ferramentas a serem utilizadas no projeto.

Os chamados processos de desenvolvimento de software não são constituídos apenas de certas etapas ou tecnologias inovadoras, mas sim de uma série ordenada de fases, que tem como objetivo final fornecer um produto ou até mesmo um serviço, de tal forma que os mesmos sejam menos dispendiosos, desgastantes, entre outros fatores que ajudam e dão suporte desde os clientes (usuários) até mesmo à própria empresa desenvolvedora. Esse apoio à empresa diz respeito às suas questões organizacionais.

Para Sommerville (2011), um processo de desenvolvimento de software é uma forma estruturada de se desenvolver software com o objetivo de facilitar a sua produção com alta qualidade e de maneira rentável. Segundo Pressman (2011), há de se considerar um importante aspecto do processo de software, chamado fluxo de processo, que descreve como são organizadas as atividades metodológicas, bem como as ações e tarefas que ocorrem dentro de cada atividade em relação à

sequência e ao tempo, de acordo com o tipo de fluxo selecionado, descritos a seguir:

- Um fluxo de processo linear executa cada uma das cinco atividades metodológicas em sequência, começando com a de comunicação e culminando com a da entrega (Figura 2.1);

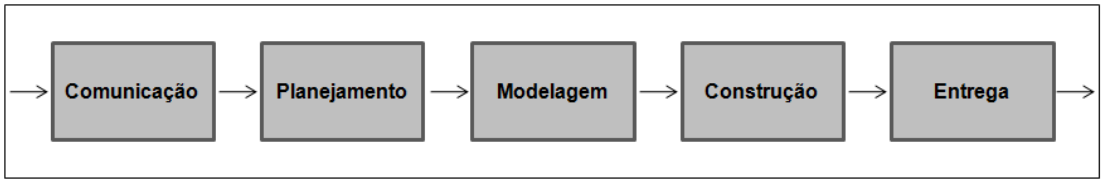

Figura 2.1 – Fluxo de processo linear (PRESSMAN, 2011)

- Um fluxo de processo iterativo repete uma ou mais das atividades antes de prosseguir para a seguinte (Figura 2.2);

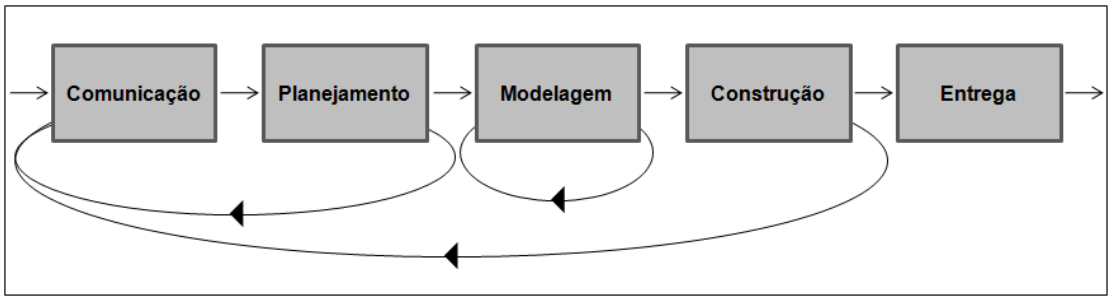

Figura 2.2 – Fluxo de processo iterativo (PRESSMAN, 2011)

- Um fluxo de processo evolucionário executa as atividades de uma forma "circular", em que cada volta pelas cinco atividades conduz a uma versão mais completa do software (Figura 2.3);

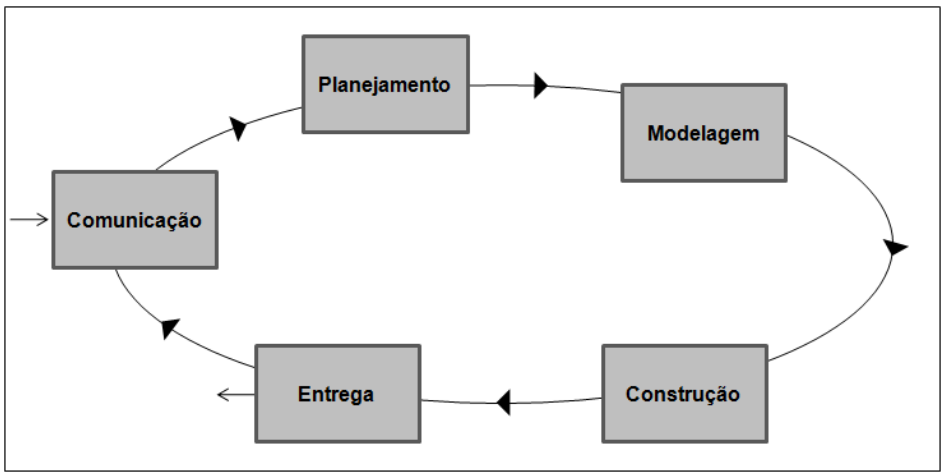

Figura 2.3 – Fluxo de processo evolucionário (PRESSMAN, 2011)

- Um fluxo de processo paralelo (Figura 2.4) executa uma ou mais atividades em paralelo com outras atividades (por exemplo, a modelagem para um aspecto do software poderia ser executada em paralelo com a construção de outro aspecto do software).

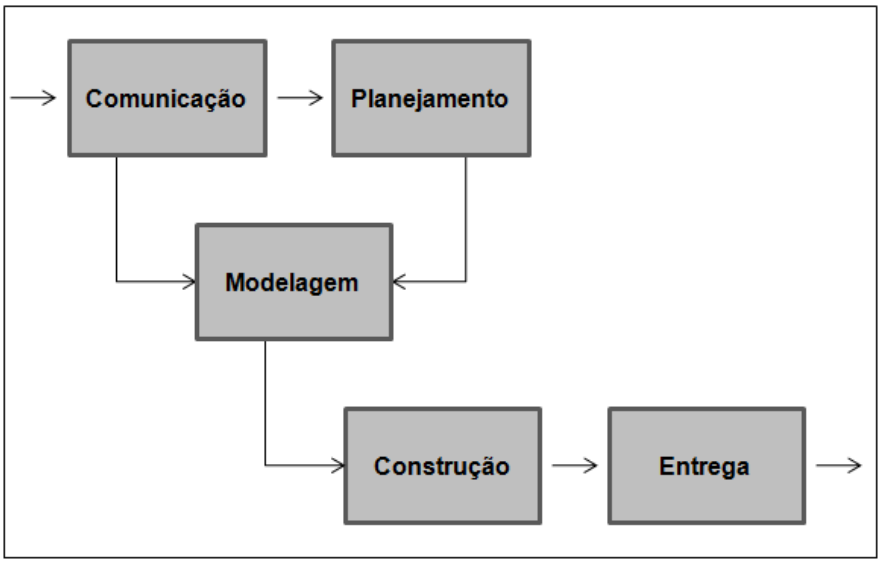

Figura 2.4 – Fluxo de processo paralelo (PRESSMAN, 2011)

Por fim, um processo é considerado como uma evolução na história do desenvolvimento de software.

2.4. Métodos

Os métodos fornecem as formas descritivas de como desenvolver o produto, abrangendo um grande número de tarefas, e devem seguir um conjunto de princípios básicos definido em cada método utilizado, um conjunto de atividades e resultados associados que geram um produto de software (SOMMERVILLE, 2011). Para Pressman (2011), métodos de engenharia de software fornecem a técnica de como fazer para construir software.

O desenvolvimento de um software depende do método utilizado e ocorre dentro de um ciclo de vida próprio, sendo que suas fases podem variar, dependendo da área de aplicação, do tamanho do projeto ou de sua complexidade. Para Pressman (2011), há três fases genéricas: a fase de definição, a fase de desenvolvimento e a fase de manutenção, que estão presentes em qualquer método

de desenvolvimento escolhido. Para a fase de manutenção há quatro tipos distintos: correção, adaptação, aperfeiçoamento e prevenção.

2.5. Ferramentas

As ferramentas proporcionam a possibilidade de um suporte ao controle do processo e métodos, podendo ser o apoio automatizado ou semiautomatizado.

As CASES (*Computer Aided Software Engineering*) criam um ambiente em engenharia de software análogo ao projeto auxiliado por computador (CAD) bastante conhecido atualmente.

2.6. Modelos de Processo

Um modelo de processo de software é uma representação abstrata de um processo de software. Cada modelo de processo representa um processo sob determinada perspectiva, fornecendo apenas informações parciais sobre esse processo (SOMMERVILLE, 2011).

Entre os modelos de processos de softwares mais conhecidos estão:

- O modelo em cascata;
- Desenvolvimento evolucionário;
- Desenvolvimento baseado em componentes;
- Desenvolvimento iterativo:
    - Modelo de desenvolvimento incremental;
    - Modelo de desenvolvimento espiral.

Há outros modelos além dos citados acima, que Pressman (2011) menciona em seu livro:

- Modelos concorrentes;
- Modelo de métodos formais;
- Desenvolvimento de software orientado a aspectos;
- Processo unificado.

2.6.1. Modelo Cascata

O modelo cascata (Figura 2.5) descreve um método de desenvolvimento que é linear e sequencial. Na primeira vez que uma fase de desenvolvimento é completada, o desenvolvimento prossegue para a próxima fase e não há retorno. A vantagem do desenvolvimento cascata é que ele permite controle departamental e gerencial. Um planejamento pode ser atribuído com prazo final para cada estágio de desenvolvimento e um produto pode prosseguir no processo de desenvolvimento e, teoricamente, ser entregue no prazo. O desenvolvimento parte do conceito, por meio do projeto (design), implementação, teste, instalação, descoberta de defeitos e termina com a operação e manutenção. Cada fase de desenvolvimento prossegue em uma ordem estrita, sem qualquer sobreposição ou passos iterativos (PRESSMAN, 2011).

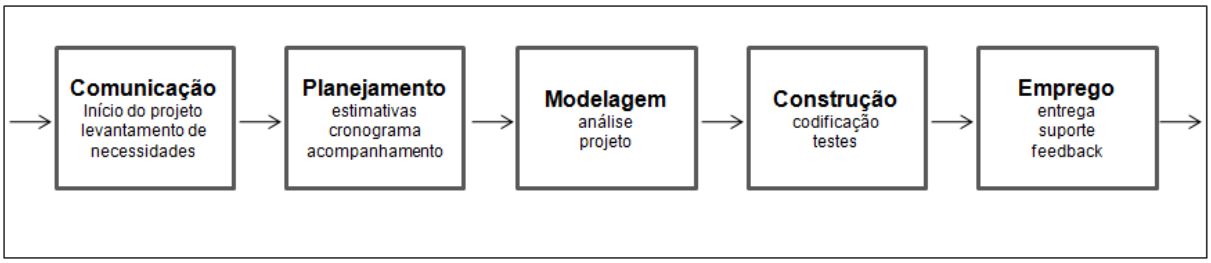

Figura 2.5 – Modelo de Processo em Cascata (PRESSMAN, 2011)

A desvantagem do desenvolvimento em cascata é que ele não permite muita flexibilidade ou revisão. A primeira vez que uma aplicação está em estágio de teste é muito difícil retornar e mudar alguma coisa que não foi bem pensada no estágio conceitual.

O modelo cascata inicia de uma abordagem orientada a projetos para a Engenharia de Software. O projeto é considerado uma tarefa claramente delineada para a qual os resultados desejados podem ser determinados completamente e sem ambiguidade. Pressman (2011) aponta alguns problemas identificados neste ciclo de vida:

- Os projetos reais raramente seguem o fluxo sequencial que o modelo propõe. Alguma iteração sempre ocorre e traz problemas na aplicação do paradigma;

- Muitas vezes é difícil para o cliente declarar todas as exigências explicitamente. O ciclo de vida clássico exige isso e tem dificuldade de acomodar a incerteza natural que existe no começo de muitos projetos;
- O cliente deve ter paciência. Uma versão de trabalho dos programas não estará disponível até um ponto tardio do cronograma do projeto. Um erro crasso, se não for detectado até que o programa de trabalho seja revisto, pode ser desastroso.

2.6.2. Desenvolvimento Evolucionário

Estudos mostraram que o software, como todos os sistemas complexos, evolui durante um período de tempo e os requisitos do negócio e do produto mudam frequentemente à medida que o desenvolvimento prossegue, dificultando um caminho direto para um produto final (PRESSMAN, 2011).

Segundo Sommerville (2011), o desenvolvimento evolucionário baseia-se na idéia de desenvolvimento de uma implementação inicial, expondo o resultado aos comentários do usuário e refinando esse resultado por meio de várias versões até que seja desenvolvido um sistema adequado. As atividades de especificação, desenvolvimento e validação são intercaladas, em vez de serem separadas, com feedback rápido que permeia as atividades.

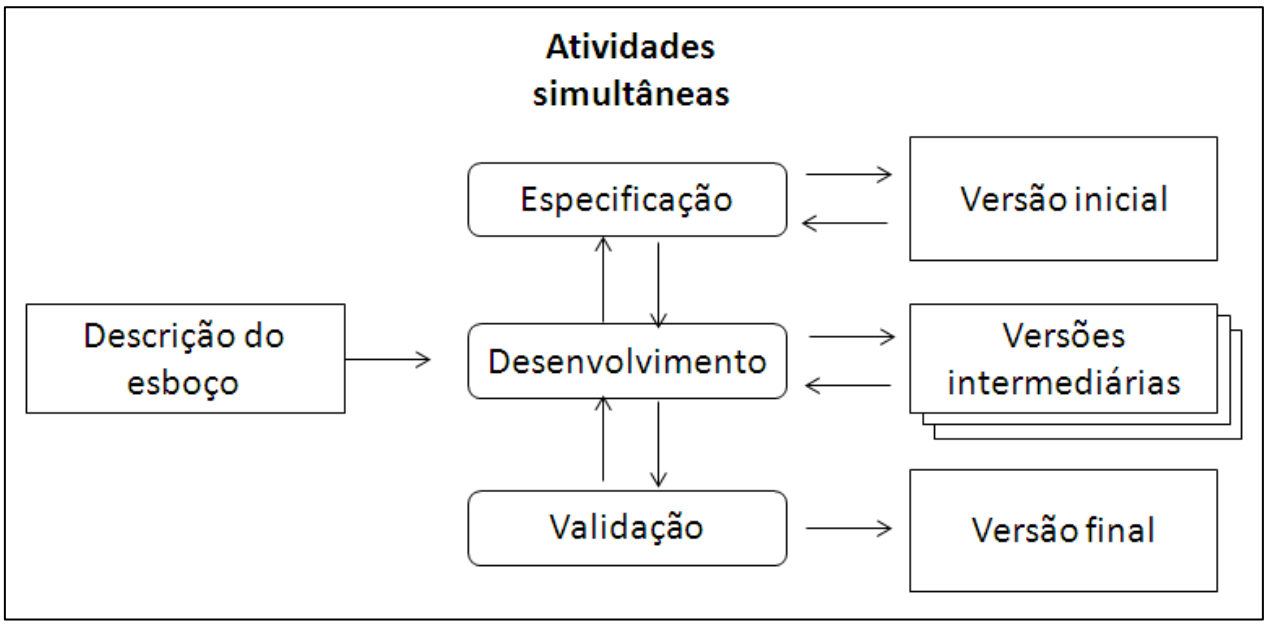

Figura 2.6 – Desenvolvimento Evolucionário (SOMMERVILLE, 2011)

2.6.3. Engenharia de Software Baseada em Componentes

Segundo Sommerville (2011), na maioria dos projetos de software, há algum reuso de software geralmente de maneira informal, quando as pessoas que trabalham no projeto conhecem projetos ou códigos similares aos necessários elas procuram por esses produtos, os modificam e os incorporam ao sistema. Na abordagem evolucionária, o reuso é frequentemente essencial para o desenvolvimento rápido do sistema. Inclusive, o reuso informal ocorre independentemente do processo de desenvolvimento usado. Um modelo genérico para esse processo é mostrado na Figura 2.7.

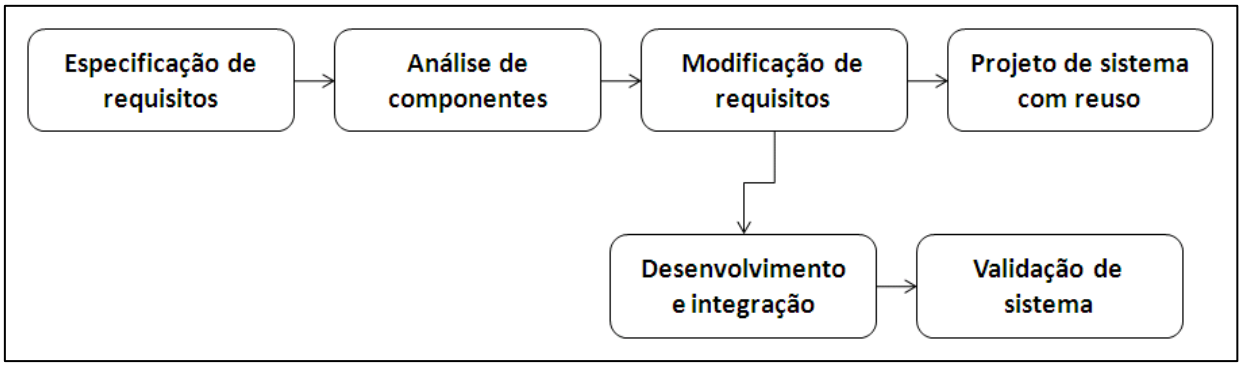

Figura 2.7 – Engenharia de Software Baseada em Componentes (SOMMERVILLE, 2011)

2.7. Iteração de Processo

Segundo Sommerville (2011), a mudança é inevitável em todos os projetos de grande porte, uma vez que os requisitos mudam, à medida que a empresa que está adquirindo o sistema responde às pressões externas, as prioridades de gerenciamento mudam. Isso significa que o processo de software não é de execução única, pelo contrário, as atividades de processo são repetidas regularmente à medida que o sistema é retrabalhado em resposta às solicitações de mudança.

Os modelos que serão citados a seguir são iterativos e caracterizam-se pela forma como se desenvolve versões cada vez mais completas do software.

2.7.1. Desenvolvimento em Espiral

Esse modelo foi desenvolvido para abranger as melhores características tanto do ciclo de vida clássico como da prototipação, acrescentando, ao mesmo tempo, um novo elemento: a análise de riscos que falta a esses paradigmas. Um modelo espiral é dividido em um conjunto de atividades metodológicas definidas pela equipe de engenharia de software, como exemplo, as atividades genéricas citadas na Figura 2.8.

Assim que esse processo evolucionário começa, a equipe de software realiza atividades indicadas por um circuito em torno da espiral no sentido horário, começando pelo seu centro.

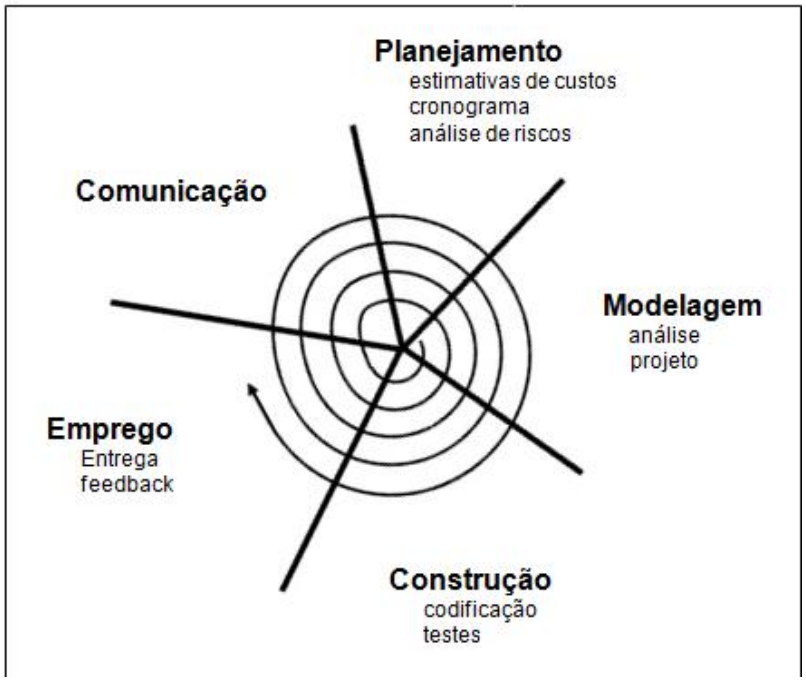

Figura 2.8 – Modelo Espiral (PRESSMAN, 2011)

Esse modelo usa uma abordagem “evolucionária” à engenharia de software, capacitando o desenvolvedor e o cliente a entender e reagir aos riscos em cada fase evolutiva. O modelo espiral usa a prototipação como um mecanismo de redução de riscos, mas possibilita que o desenvolvedor aplique a abordagem de prototipação em qualquer etapa da evolução do produto. Ele mantém a abordagem de passos sistemáticos sugerida pelo ciclo de vida clássico, mas incorpora-a numa estrutura iterativa que reflete mais realisticamente o mundo real. O modelo espiral exige uma consideração direta dos riscos técnicos em todas as etapas do projeto e,

se adequadamente aplicado, deve reduzir os riscos antes que eles se tornem problemáticos. Esse modelo considera que a forma do desenvolvimento de software não pode ser completamente determinada de antemão (PRESSMAN, 2011).

2.7.2. Entrega Incremental

O modelo incremental combina elementos dos fluxos de processos lineares e paralelos (PRESSMAN, 2011). O modelo de processo incremental é iterativo como a prototipagem, mas diferente da prototipagem, o incremental tem como objetivo apresentar um produto operacional a cada incremento realizado.

Na Figura 2.9, o modelo incremental aplica sequências lineares, de forma escalonada, à medida que o tempo vai avançando. Cada sequência linear gera “incrementais” (entregáveis/aprovados/liberados) do software (PRESSMAN, 2011).

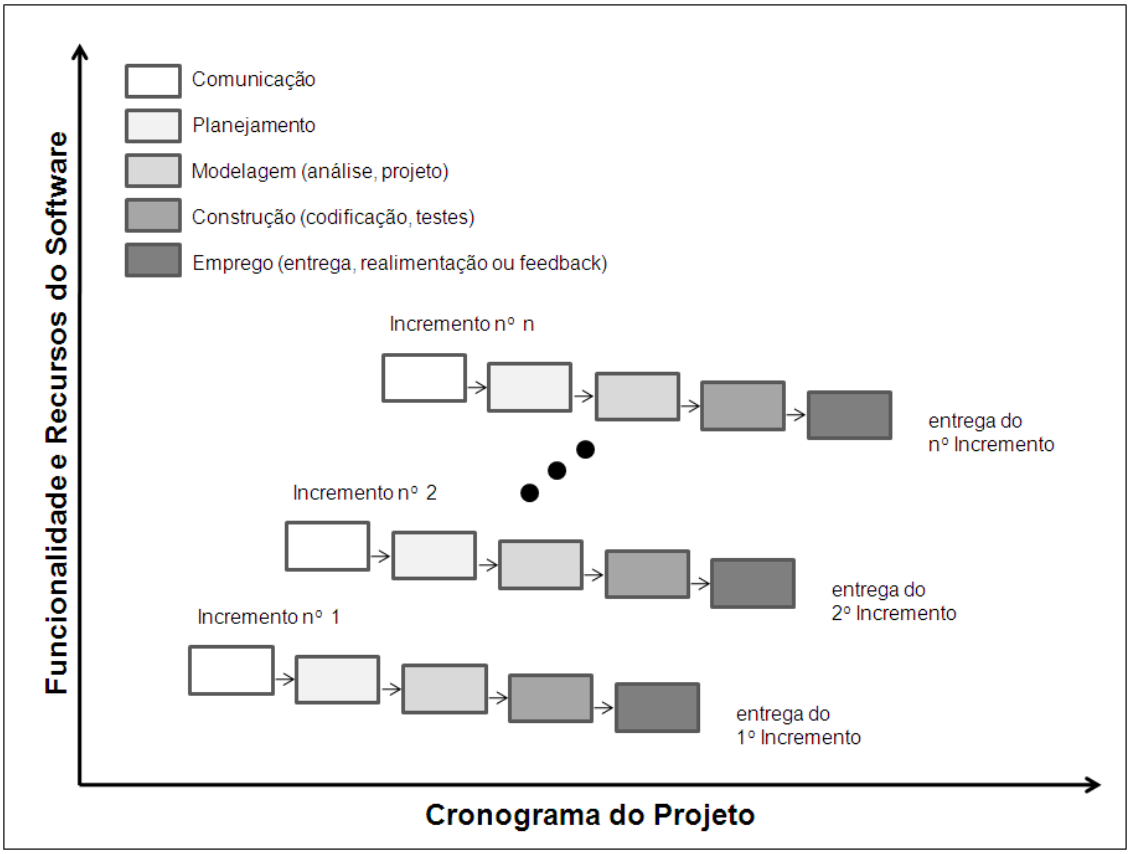

Figura 2.9 – Fases do processo unificado (PRESSMAN, 2011)

Esse modelo é muito útil quando a empresa não possui mão de obra disponível no momento para uma implementação completa, dentro do prazo estipulado.

2.8. Processo Unificado (PU)

O processo unificado de desenvolvimento de software é o conjunto de atividades necessárias para transformar requisitos do usuário em um sistema de software (JACOBSON, et al., 1999). Ele é baseado em componentes, o que significa o sistema ser construído a partir de componentes de software interconectados por meio de interfaces muito bem definidas. O processo unificado utiliza a Linguagem de Modelagem Unificada (*Unified Modeling Language* – UML) no preparo de todos os artefatos do sistema. Segundo Jacobson, et al. (1999), os aspectos que distinguem o processo unificado são determinados em três conceitos chave: direcionado a casos de uso, centrado na arquitetura e iterativo e incremental.

2.8.1. Direcionado a Casos de Uso

Um sistema de software é feito para servir seus usuários. Portanto, para construir um sistema de sucesso devemos saber quem são seus usuários potenciais e o que eles querem e precisam. O termo usuário representa alguém ou alguma coisa (como outro sistema) que interage com o sistema que está sendo desenvolvido.

Um caso de uso é um pedaço de funcionalidade do sistema que dá ao usuário um resultado de valor (JACOBSON, et al., 1999). Casos de uso capturam requisitos funcionais e todos juntos resultam no modelo de casos de uso, o qual descreve a funcionalidade completa do sistema.

Este modelo substitui a especificação funcional tradicional, cujo papel é responder à seguinte questão: o que o sistema faz? A estratégia de casos de uso pode ser caracterizada pela adição de três palavras no final dessa pergunta: para cada usuário? Estas palavras têm uma implicação muito importante. Forçam-nos a pensar em termos dos valores dos usuários, não apenas em funções que poderiam ser interessantes.

Os casos de uso direcionam o processo de desenvolvimento, já que, baseados no modelo de casos de uso os desenvolvedores criam uma série de modelos de projeto e implementação que os realizam efetivamente. Os responsáveis pelos testes realizam seu trabalho com o propósito de garantir que os componentes

do modelo de implementação cumpram corretamente os objetivos estabelecidos nos casos de uso. Desta forma, os casos de uso não somente iniciam o processo de desenvolvimento, mas também o mantém coeso. Direcionado a casos de uso significa que o processo de desenvolvimento executa uma sequencia de tarefas derivadas dos casos de uso. Eles são especificados, projetados e servem de base para a construção dos casos de teste.

Embora seja verdade que os casos de uso dirigem o processo, eles não são selecionados isoladamente. São desenvolvidos juntamente com a arquitetura do sistema. Ou seja, os casos de uso direcionam a arquitetura do sistema, que por sua vez influencia a seleção dos casos de uso. Portanto, ambos amadurecem no decorrer do ciclo de vida do sistema.

2.8.2. Centrado na Arquitetura

O papel da arquitetura no software (JACOBSON, et al., 1999) é de natureza similar ao papel da arquitetura na construção civil. As construções são observadas sob vários pontos de vista: estrutura, serviços, condução de calor, encanamento, eletricidade etc. Da mesma forma, a arquitetura em um sistema de software é descrita como sendo as diferentes visões desse sistema.

O conceito de arquitetura de software incorpora os aspectos estáticos e dinâmicos mais importantes do sistema. A arquitetura é influenciada por muitos fatores, tais como a plataforma de software sobre a qual o sistema vai rodar (sistema operacional, sistema gerenciador de banco de dados, protocolos para comunicação em rede etc.), blocos de construção reusáveis disponíveis (por exemplo, um *framework* para construção de interface gráfica com o usuário), considerações de distribuição, sistemas legado e requisitos não funcionais (desempenho, confiabilidade etc.). Ela representa uma visão do projeto como um todo, na qual as características mais importantes são colocadas em destaque, deixando os detalhes de lado.

Todo produto tem função e forma e nenhum desses elementos sozinho é suficiente. Essas duas forças devem ser balanceadas para se obter um produto de sucesso. Neste caso, a função corresponde aos casos de uso e a forma à arquitetura. Por um lado, os casos de uso devem, quando construídos, encaixar-se

na arquitetura. Por outro, a arquitetura deve fornecer espaço para a construção de todos os casos de uso necessários, agora e no futuro. Na realidade, ambos devem ser desenvolvidos em paralelo.

Para encontrar essa forma, os arquitetos devem trabalhar a partir de uma compreensão geral das funções chave do sistema, isto é, dos casos de uso chave. Estes devem ficar em torno de 5 a 10% de todos os casos de uso, mas são os mais significativos, aqueles que constituem o núcleo das funções do sistema. Em termos simplificados, os arquitetos criam um esboço da arquitetura iniciando com a parte que não é específica dos casos de uso (por exemplo, a plataforma). Embora seja uma parte independente dos casos de uso, o arquiteto deve ter uma compreensão geral destes antes da criação do esboço.

Depois, o arquiteto trabalha com um subconjunto dos casos de uso identificados, aqueles que representam as funções chave do sistema em desenvolvimento. Cada caso de uso selecionado é especificado em detalhes e construído em termos de subsistemas, classes e componentes.

À medida que os casos de uso são especificados e atingem maturidade, mais detalhes da arquitetura são descobertos. Isto, por sua vez, leva ao surgimento de mais casos de uso.

Este processo continua até que a arquitetura seja considerada estável.

2.8.3. Iterativo e Incremental

O desenvolvimento de um produto comercial de software é uma grande tarefa que pode ser estendida por vários meses, possivelmente um ano ou mais. É mais prático dividir o trabalho em pedaços menores ou miniprojetos. Cada miniprojeto é uma iteração que resulta em um incremento. Iterações são passos em um fluxo de trabalho e incrementos são crescimentos do produto.

Os desenvolvedores selecionam o que deve ser feito em cada iteração baseados em dois fatores. Primeiro, a iteração deve trabalhar com um grupo de casos de uso que juntos estendam a usabilidade do produto em desenvolvimento. Segundo, a iteração deve tratar os riscos mais importantes.

Um incremento não é necessariamente a adição do código executável correspondente aos casos de uso que pertencem à iteração em andamento. Especialmente nas primeiras fases do ciclo de desenvolvimento, os desenvolvedores podem substituir um projeto superficial por um mais detalhado ou sofisticado. Em fases avançadas os incrementos são tipicamente aditivos.

Em cada iteração, os desenvolvedores identificam e especificam os casos de uso relevantes, criam um projeto utilizando a arquitetura escolhida como guia, implementam o projeto em componentes e verificam se esses componentes satisfazem os casos de uso. Se uma iteração atinge seus objetivos, e isso normalmente ocorre, o desenvolvimento prossegue com a próxima iteração, caso contrário, os desenvolvedores devem rever suas decisões e tentar uma nova abordagem.

Segundo Jacobson, et al. (1999), há vários benefícios em se adotar um processo iterativo controlado, entre os quais podemos destacar:

- Redução dos riscos envolvendo custos a um único incremento. Se os desenvolvedores precisarem repetir a iteração, a organização perde somente o esforço mal direcionado de uma iteração, não o valor de um produto inteiro.
- Redução do risco de lançar o projeto no mercado fora da data planejada. Identificando os riscos numa fase inicial o esforço despendido para gerenciá-los ocorre cedo, quando as pessoas estão sob menos pressão do que numa fase final de projeto.
- Aceleração do tempo de desenvolvimento do projeto como um todo, porque os desenvolvedores trabalham de maneira mais eficiente quando buscam resultados de escopo pequeno e claro;
- Reconhecimento de uma realidade frequentemente ignorada: as necessidades dos usuários e os requisitos correspondentes não podem ser totalmente definidos no início do processo. Eles são tipicamente refinados em sucessivas iterações. Este modelo de operação facilita a adaptação a mudanças de requisitos.

Os três conceitos apresentados (dirigido a casos de uso, centrado em arquitetura, iterativo e incremental) são igualmente importantes. Segundo Jacobson (1999), remover um deles poderia reduzir drasticamente o valor do processo unificado.

2.8.4. O ciclo de vida do Processo Unificado

Segundo Jacobson, et al. (1999), o processo unificado consiste da repetição de uma série de ciclos durante a vida de um sistema, como mostrado na Figura 2.10. Cada ciclo é concluído com uma versão do produto pronta para distribuição. Essa versão é um conjunto relativamente completo e consistente de artefatos, possivelmente incluindo manuais e um módulo executável do sistema, que podem ser distribuídos para usuários internos ou externos.

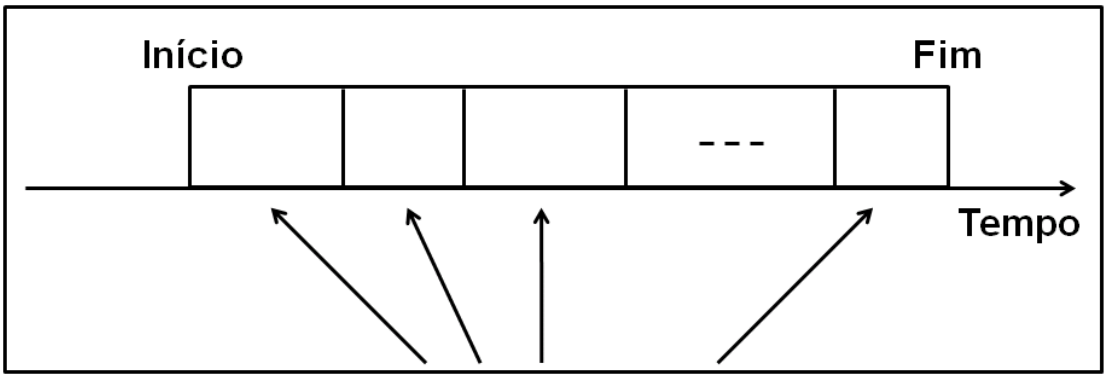

Figura 2.10 – A vida de um processo em ciclos (JACOBSON, 1999)

Cada ciclo consiste de quatro fases: concepção, elaboração, construção e transição. Cada fase é também subdividida em iterações, vide Figura 2.11.

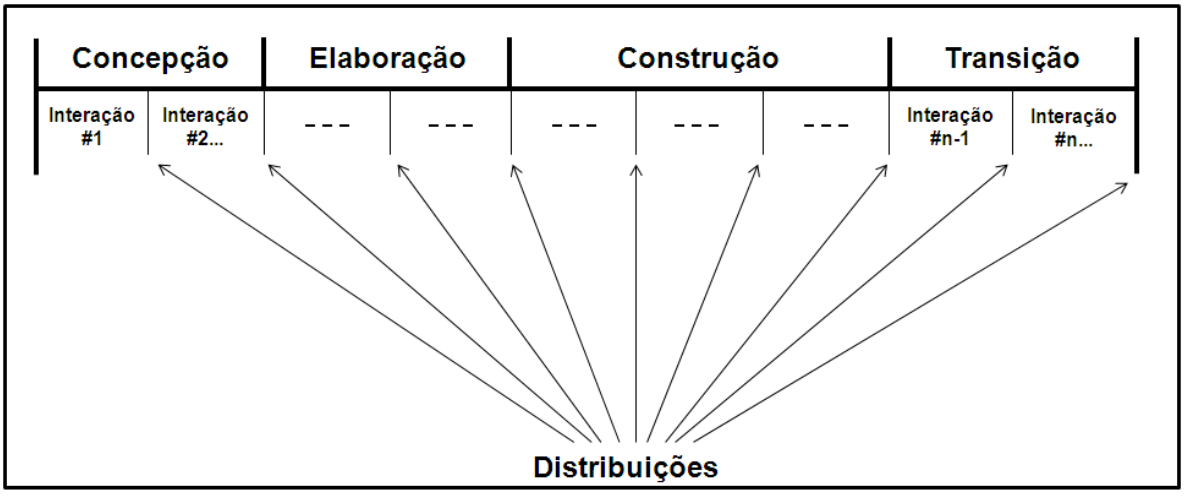

Figura 2.11 – A vida de um processo em ciclos (JACOBSON, 1999)

2.8.5. O Produto

O produto final inclui artefatos de interesse do usuário, tais como manuais e código fonte incorporado em componentes que podem ser compilados e executados, e artefatos que interessam a outras pessoas, como os requisitos, os casos de uso, as especificações não funcionais e os casos de teste. Inclui também modelos da arquitetura. Na realidade, inclui todos os elementos já mencionados, os quais permitem especificar, projetar, implementar, testar e utilizar o sistema.

Mesmo que componentes executáveis sejam os produtos mais importantes do ponto de vista dos usuários, sozinhos eles não são suficientes. Isto acontece porque o ambiente muda. Sistemas operacionais, sistemas de bancos de dados e máquinas evoluem. Os próprios requisitos podem mudar à medida que compreendermos melhor a missão do sistema. Na realidade, esta é uma das constantes no desenvolvimento de software: requisitos irão mudar.

Para executar um novo ciclo eficientemente, os desenvolvedores precisam de todas as representações do produto (Figura 2.12):

- um modelo de casos de uso, contendo todos dos casos de uso e seus relacionamentos com os usuários;
- um modelo de análise, que tem dois propósitos: refinar os casos de uso em mais detalhes e fazer uma alocação inicial do comportamento do sistema a um conjunto de objetos;
- um modelo de projeto que define a estrutura estática do sistema em termos de subsistemas, classes e interfaces, e a realização dos casos de uso como sendo colaborações entre subsistemas, classes e interfaces;
- um modelo de implementação, o qual inclui componentes representando código fonte e o mapeamento entre classes e componentes;
- um modelo de distribuição que define os nós físicos de computadores e o mapeamento entre os componentes e esses nós;

- um modelo de teste, o qual descreve os casos de teste que serão usados para verificar os casos de uso;
- uma representação da arquitetura.

O sistema pode ter também um modelo de domínio ou um modelo de negócios que descreva o contexto no qual ele está inserido.

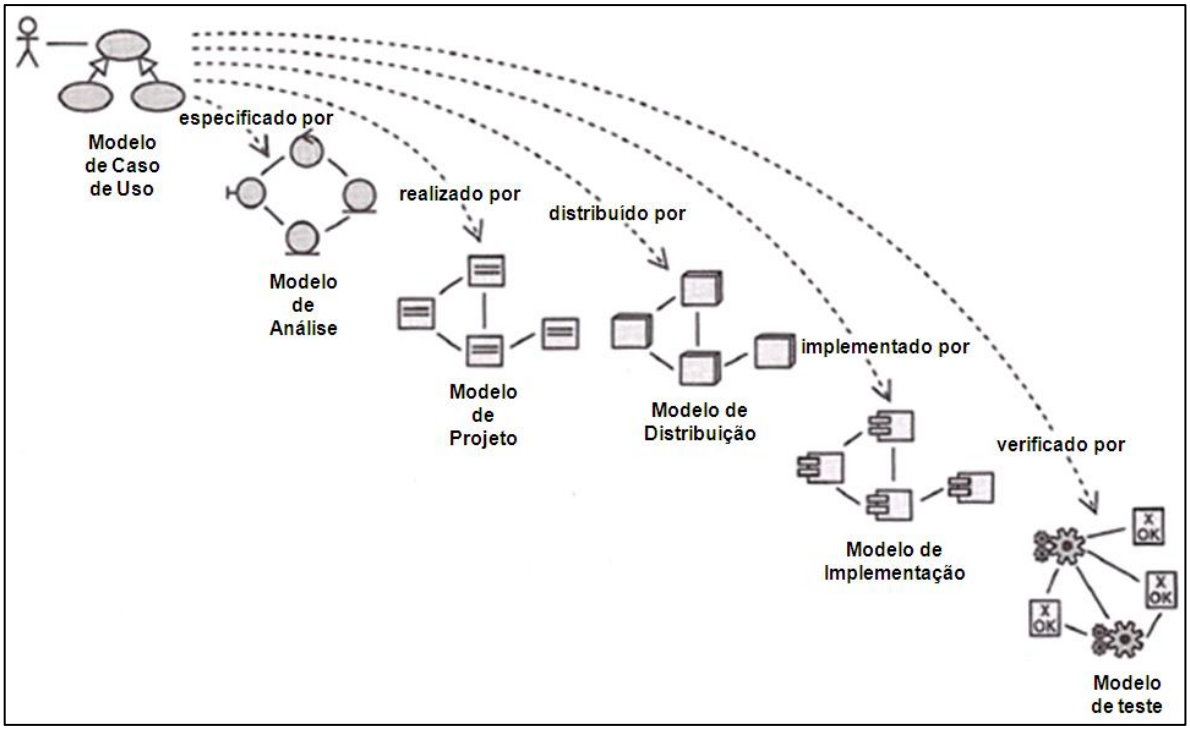

Figura 2.12 – Modelos do Processo Unificado (JACOBSON, et al., 1999)

Todos esses modelos estão relacionados e juntos representam o sistema como um todo. Segundo Jacobson, et al. (1999), elementos em um modelo têm uma ligação com elementos de outro modelo. Por exemplo, um caso de uso pode estar relacionado à sua realização no modelo de projeto e a um caso de teste no modelo de teste. A rastreabilidade facilita a compreensão e a modificação.

2.8.6. Fases em um ciclo

Um ciclo está dividido em quatro fases, cada qual podendo ser subdividida em iterações e consequentes incrementos (figura 2.13). O final de uma fase é marcado por um ponto de verificação, isto é, pela disponibilidade de um conjunto de artefatos que possibilitem a avaliação do projeto, tais como modelos e outros documentos.

Os pontos de verificação servem a diversos propósitos:

- Gerentes devem tomar certas decisões cruciais antes de o trabalho continuar;
- Permitem a monitoração do progresso dos trabalhos;
- Observando o tempo e o esforço despendidos em cada fase, é possível reunir dados úteis para estimar os requisitos de tempo e *staff* de outros projetos.

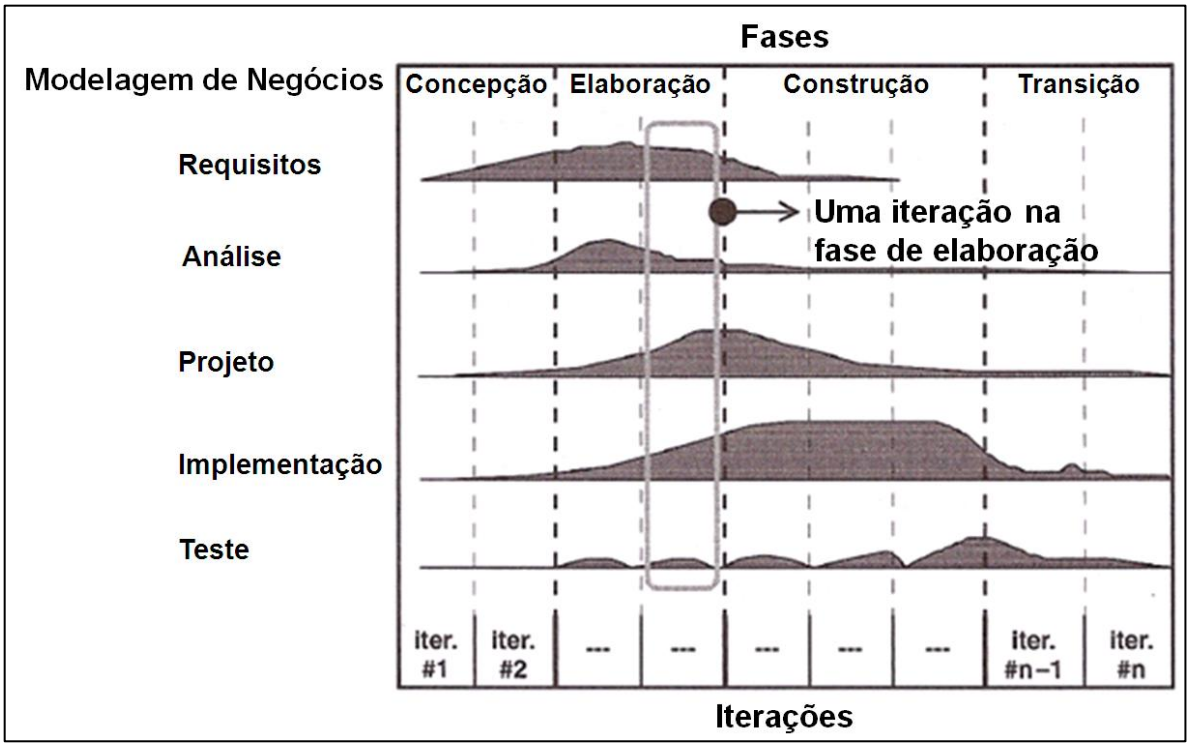

Figura 2.13 – As cinco atividades ocorrem nas quatro fases (JACOBSON, et al., 1999)

A figura 2.13 lista na coluna à esquerda as atividades que devem ser realizadas em cada fase (requisitos, análise, projeto, implementação e teste). As curvas não devem ser interpretadas literalmente, mas representam uma aproximação do esforço despendido com cada atividade em cada fase. Relembre que uma fase normalmente está subdividida em iterações, ou miniprojetos. Uma iteração típica executa as cinco atividades, como mostra a figura 2.13 na fase de elaboração.

Durante a fase de concepção é possível responder às seguintes perguntas:

- O que o sistema fará, principalmente para cada um dos principais usuários?
- Como poderia ser a arquitetura desse sistema?
- Qual é o plano e quanto custará desenvolver o produto?

Um modelo de casos de uso simplificado, que contenha os casos de uso mais críticos, poderá responder à primeira questão. Nesta fase a arquitetura é experimental, tipicamente apenas um esboço contendo os subsistemas mais cruciais. Os riscos mais importantes são identificados e priorizados, o planejamento é detalhado e o projeto como um todo é estimado de forma parcial.

Durante a fase de elaboração, a maioria dos casos de uso do produto é especificada em detalhes e a arquitetura do sistema é projetada.

O relacionamento entre arquitetura do sistema e o sistema propriamente dito é o que existe de mais importante. Uma maneira simples de entender isso é imaginar que a arquitetura é um esqueleto coberto de pele, mas com muito pouco músculo (software) entre os ossos e a pele, apenas o suficiente para permitir que o esqueleto faça movimentos básicos. O sistema é o corpo inteiro, composto de esqueleto, pele e músculos.

Portanto, a arquitetura é expressa em forma de visões de todos os modelos do sistema. Isto implica a existência de visões arquiteturais do modelo de casos de uso, do modelo de análise, do modelo de projeto, do modelo de implementação e do modelo de distribuição. Durante esta fase, os casos de uso mais críticos são realizados. O resultado desta fase é uma arquitetura básica, isto é, um sistema pequeno, "magro", "descarnado" (com pouco software).

No final da fase de elaboração, o gerente do projeto está em condição de planejar as atividades e estimar os recursos necessários para completar o projeto. A questão chave aqui é a seguinte: os casos de uso, a arquitetura e os planos são estáveis o suficiente e os riscos estão sob controle, a ponto de nos comprometermos em contrato com o desenvolvimento do trabalho como um todo?

Durante a fase de construção o produto é efetivamente construído, músculos (software completo) são adicionados ao esqueleto (arquitetura). Embora a

arquitetura do sistema encontre-se estável, os desenvolvedores podem descobrir maneiras melhores de estruturá-lo e podem sugerir pequenas mudanças aos arquitetos.

No final desta fase, o produto contém todos os casos de uso que gerentes e clientes acordaram desenvolver nessa versão. Entretanto, é possível que ele não esteja totalmente livre de defeitos. Mais defeitos poderão ser descobertos e corrigidos durante a fase de transição.

A fase de transição cobre o período durante o qual o produto fica em versão beta. Nessa versão, um pequeno número de usuários experientes utiliza o produto e relata defeitos e deficiências. Desenvolvedores os corrigem e incorporam algumas das melhorias sugeridas. Esta fase envolve atividades como, treinamento de clientes, fornecimento de assistência online e correção de defeitos encontrados depois da distribuição. A equipe de manutenção frequentemente divide os defeitos em duas categorias: aqueles com efeitos operacionais que justificam correção imediata e aqueles que podem ser corrigidos na próxima versão regular.

No contexto deste trabalho, pode-se considerar o processo unificado como base para o desenvolvimento do método de auditoria a ser proposto, pois se trata de um processo iterativo e incremental. Em seguida será apresentada a pesquisa de campo que foi base para a formulação do método de auditoria proposto.

3. Pesquisa de Campo: Avaliação das Atividades de Controle do Método de Auditoria Proposto

Este trabalho propõe um método de auditoria para apoiar o desenvolvimento de software, visando garantir a entrega do produto de acordo com o planejado. O conjunto inicial de atividades de controle que compões este método foi validado por meio de uma pesquisa de campo com especialistas em auditoria de sistemas e no processo de software, considerando a importância destas atividades para os projetos de software, tendo os resultados sido utilizados para definir as atividades que fazem parte do método.

Esse capítulo apresenta os critérios quantitativos da amostra, como foram definidas as atividades de controle e o resultado da pesquisa. Com os resultados obtidos nessa pesquisa de campo, as atividades avaliadas foram refinadas e geraram as atividades do Método de Auditoria para Apoio ao Desenvolvimento de Software, proposto no capítulo 6.

3.1. Escopo e Critérios Amostrais da Pesquisa de Campo

O conjunto inicial de atividades de controle foi avaliado, por meio de uma pesquisa de campo, por 50 especialistas, em auditoria de sistemas ou no processo de software, sendo colaboradores de instituições públicas e privadas distribuídas entre os estados São Paulo, Paraná e Santa Catarina. A amostra foi totalmente aleatória, tendo sido submetidos inicialmente 100 questionários para especialistas que pertenciam à rede de colegas especialistas em uma das duas áreas, os quais também encaminharam a pesquisa para sua rede de colegas, o que resultou no total de 120 questionários enviados, obtendo-se a porcentagem de 42% de respostas.

A condução desta pesquisa de campo ocorreu a partir da aplicação de um questionário desenvolvido a partir de boas práticas extraídas dos seguintes padrões de referência, por serem os mais citados nas consultas do tema qualidade de software:

- Cobit 4.1 (ITGI, 2007) - boas práticas de governança de TI relacionadas ao desenvolvimento de sistemas;

- *CMMI for Development* (SEI, 2011) – boas práticas identificadas no modelo de qualidade de software;
- ISO/IEC 15408 (ISO/IEC/JTC, 2009) - boas práticas de segurança no desenvolvimento de sistemas;
- MPS.BR - Melhoria de Processos do Software Brasileiro (SOFTEX, 2012) - boas práticas identificadas no modelo de qualidade de software;
- NBR ISO/IEC 12207 (ABNT, 2009) - boas práticas identificadas no processo de desenvolvimento de software;
- NBR ISO/IEC 15504 (ABNT, 2008) - boas práticas identificadas no processo de desenvolvimento de software;
- NBR ISO/IEC 27002 (ABNT, 2005) - boas práticas de segurança;
- NBR ISO/IEC 9126-1 (ABNT, 2003) - boas práticas identificadas no modelo de qualidade de software.

Além disso, foram consideradas as seguintes publicações, principalmente para a definição da estrutura e organização dos tópicos alinhados ao processo unificado de desenvolvimento de software:

- Pressman (2011);
- Sommerville (2011).

Em seguida, será descrito o procedimento da pesquisa para definição das atividades de controle com base nos padrões de referência citados.

3.2. Procedimento de Identificação e Definição das Atividades de Controle do Método Proposto

Para se definir as atividades de controle do método de auditoria, foram seguidos os seguintes passos, detalhados a seguir:

- Definição da estrutura do método e dos dados necessários;
- Mapeamento das boas práticas aplicáveis ao objetivo do método, a partir de consulta em padrões de referência, ou base de conhecimento;

- Identificação de atividades de controle que poderiam ser formadas a partir das boas práticas consideradas;
- Definição do conjunto inicial de atividades do método, agrupando-as por objetivos de controle similares ou por equivalência com outras atividades de controle, eliminando-se as duplicidades.

Para se definir como o método seria estruturado, optou-se por uma divisão pelas etapas que compõem o ciclo do processo unificado, uma vez que o método é orientado por este processo: Concepção, Elaboração, Construção e Transição. Foram consideradas como base as definições de Pressman (2011), Sommerville (2011) e Jacobson, et. al. (1999). O detalhamento desse lavantamento pode ser observado no APÊNDICE A.

Em seguida, foi realizado um mapeamento sobre os padrões de referência para identificar as boas práticas do processo de software consideradas importantes no contexto de qualidade, principalmente aquelas que contemplam definição ou análise de requisitos e que envolvem usuários ou clientes. Uma vez reunidas, as atividades foram agrupadas em um segundo nível, chamado “Macro Atividade”, para facilitar o agrupamento. Um terceiro e último nível, que é a atividade propriamente dita, contempla a descrição completa da atividade a ser avaliada segundo o método de auditoria. Nessa fase de pesquisa, optou-se por manter a referência completa da origem da boa prática considerada, para que, no desenvolvimento do método, fosse possível consultar a descrição completa da boa prática para descrever os possíveis testes ou evidências a serem observadas (APÊNDICE A).

A última etapa da definição das atividades consistiu na normalização das diversas tabelas de atividades geradas anteriormente, que foram geradas particularmente observando-se o padrão de referência ou publicação. Portanto, foi observada a correlação das atividades e realizou-se a consolidação daquelas que se equivalem ou se complementam. Então, o mapeamento completo de atividades do método de auditoria proposto foi definido com os seguintes dados, cujo detalhamento pode ser observado no APÊNDICE B:

- Macro Atividade: as etapas que compõem o ciclo do processo unificado;
- Objetivo de Controle: uma orientação do objetivo da avaliação a ser realizada (o que se pretende garantir com o teste);
- Grupo de Atividades: um agrupamento macro das atividades quanto ao seu enquadramento nas fases ou etapas do projeto de software;
- Atividade: qual artefato ou resultado deve ser verificado no projeto, ou evidências a serem avaliadas.

Por fim, como pode ser visto no APÊNDICE C, este mapeamento originou o questionário aplicado na pesquisa de campo, o qual foi formado pelos seguintes dados:

- Macro Atividade: as etapas que compõem o ciclo do processo unificado;
- Atividade: foi expressa no nível de agrupamento macro das atividades quanto ao seu enquadramento nas fases ou etapas do projeto de software.

O objetivo da pesquisa de campo foi validar as atividades selecionadas para o método por meio da opinião dos especialistas, os quais em seu dia a dia têm experiências para avaliar se fazem sentido e se são de fato importantes. Com isso, foi definido um método de auditoria pautado em conclusões de experiências reais, não apenas teóricas. O resultado da pesquisa é apresentado a seguir.

3.3. Pesquisa de Campo

Dos questionários respondidos nenhum foi descartado, apenas as atividades que não receberam uma nota foram desconsideradas das avaliações. Essa pesquisa de campo foi divida em três blocos. O primeiro bloco traçou o perfil dos especialistas em:

- Processo de Software:

    - Como você classificaria seu conhecimento em processos de software?
    - Como você classificaria sua experiência prática em processos de software?

- Auditoria de Sistemas:
    - Como você classificaria seu conhecimento em auditoria de sistemas?
    - Como você classificaria sua experiência prática em auditoria de sistemas?

O segundo bloco engloba questões relativas ao grau de importância de cada atividade desse método de auditoria, a partir da escala abaixo:

- 0: atividade não é importante para o processo de desenvolvimento de software.
- 1: atividade pouco importante para o processo de desenvolvimento de software.
- 2: atividade com média importância para o processo de desenvolvimento de software.
- 3: atividade importante para o processo de desenvolvimento de software.
- 4: atividade muito importante para o processo de desenvolvimento de software.

O terceiro bloco elenca um conjunto de aspectos que podem influenciar o apoio da auditoria de sistemas num processo de desenvolvimento de software a partir da escala abaixo:

- 0: aspecto não é influente na realização de auditoria de sistemas no processo de desenvolvimento de software.
- 1: aspecto com pouca influência na realização de auditoria de sistemas no processo de desenvolvimento de software.

- 2: aspecto com média influência na realização de auditoria de sistemas no processo de desenvolvimento de software.
- 3: aspecto influente na realização de auditoria de sistemas no processo de desenvolvimento de software.
- 4: aspecto com muita influência na realização de auditoria de sistemas no processo de desenvolvimento de software.

Os resultados obtidos nesta pesquisa serão apresentados, enfocando a:

(i) avaliação consolidada dos especialistas em relação à importância das atividades de desenvolvimento de software a um método de auditoria;

(ii) avaliação dos especialistas em processo de software em relação à importância das atividades de desenvolvimento de software a um método de auditoria;

(iii) avaliação dos especialistas em auditoria de sistemas em relação à importância das atividades de desenvolvimento de software a um método de auditoria;

(iv) avaliação consolidada dos especialistas sobre os aspectos que podem influenciar o apoio da auditoria de sistemas num processo de desenvolvimento de software.

3.4. Avaliação Consolidada das Respostas dos Especialistas

A Tabela 3.1 mostra, em ordem decrescente do grau de importância (média aritmética e desvio padrão), todas as atividades analisadas pelos especialistas desta pesquisa de campo. A Figura 3.1 mostra, em ordem decrescente do grau de importância, as seis atividades melhor avaliadas do método de auditoria proposto, por todos os especialistas desta pesquisa. Elas representam as atividades que obtiveram avaliações iguais ou superiores a 3,50. Nas demais representações procurou-se seguir o padrão de seis atividades para aquelas melhor avaliadas.

A atividade “Estabelecer o escopo e os limites do projeto” foi avaliada com o valor mais alto, 3,86, tendendo para muito importante em um método de auditoria

no processo de desenvolvimento de software. Isto evidencia a relevância de se definir o escopo do software e os limites do projeto, ou seja, as funções e características que deverão ser entregues aos usuários finais, os dados de entrada e saída, as informações que serão apresentadas aos usuários como consequência do uso do software e o desempenho, as restrições, as interfaces e a confiabilidade que limitarão o sistema.

Tabela 3.1 – Atividades do método de auditoria avaliadas pelos especialistas (Fonte: O Autor)

Atividades presentes no processo de desenvolvimento de software	Média	Desvio Padrão
Estabelecer o escopo e os limites do projeto	3,86	0,35
Implementar o software no ambiente de produção	3,74	0,48
Realizar o planejamento inicial	3,70	0,50
Definir os requisitos do software	3,62	0,56
Realizar a validação do software	3,55	0,67
Analisar e validar Requisitos	3,52	0,61
Desenvolver o software conforme o projeto	3,46	0,61
Gerenciar os riscos do projeto de desenvolvimento de software	3,46	0,73
Realizar a verificação do software	3,44	0,78
Desenvolver o plano de testes do software	3,40	0,66
Educar e treinar os usuários envolvidos no desenvolvimento do software	3,39	0,66
Definir o planejamento do projeto de desenvolvimento de software	3,38	0,77
Gerenciar o projeto de software	3,36	0,59
Gerenciar mudanças no software	3,36	0,66
Gerenciar os requisitos do software	3,36	0,69
Definir o projeto de testes de software	3,36	0,74
Definir processos e procedimentos para definição dos requisitos do software	3,34	0,65
Desenvolver o software conforme as mudanças requeridas	3,31	0,71
Definir especificação para sistemas críticos	3,28	0,85
Gerenciar a qualidade do software	3,28	0,85
Definir políticas, processos e planos para orientar o desenvolvimento do software	3,24	0,74
Definir o projeto de interface com o usuário	3,24	0,76
Definir o projeto de desenvolvimento do software	3,23	0,65
Definir o projeto de arquitetura do software	3,22	0,61
Definir a especificação formal do software	3,20	0,76
Atualizar a documentação do usuário	3,20	0,86
Definir o planejamento para verificação do software	3,20	0,72

Tabela 3.1 – Atividades do método de auditoria avaliadas pelos especialistas (continuação)

Atividades presentes no processo de desenvolvimento de software	Média	Desvio Padrão
Adquirir e manter a infraestrutura de tecnologia	3,20	0,77
Gerenciar configurações do software	3,18	0,71
Definir o planejamento para validação do software	3,14	0,64
Definir controles a serem implementados no software	3,12	0,77
Definir uma estratégia de integração	3,12	0,79
Definir estratégias de testes de software	3,12	0,84
Definir a modelagem de análise do software	3,08	0,72
Definir o modelo do sistema	3,06	0,81
Definir técnicas para o desenvolvimento de sistemas críticos	3,06	0,90
Definir uma estratégia de validação	3,04	0,75
Definir o modelo de projeto de software	3,00	0,82
Realizar auditoria técnica do software	2,92	0,72
Definir o planejamento para a evolução do software	2,86	0,72
Definir o projeto no nível de componentes	2,69	0,81
Definir uma estratégia de testes adicionais	2,67	0,77
Realizar revisão por pares nos produtos do projeto	2,65	0,83

A atividade “Implementar o software no ambiente de produção” foi a segunda em importância na avaliação com o valor de 3,74, convergindo para o patamar de muito importante em um método de auditoria no processo de desenvolvimento de software. Neste contexto, é recomendável que sejam definidas atividades de controle para verificar se o software foi disponibilizado no ambiente de produção em alinhamento com os requisitos do negócio, no prazo desejado e com um custo razoável.

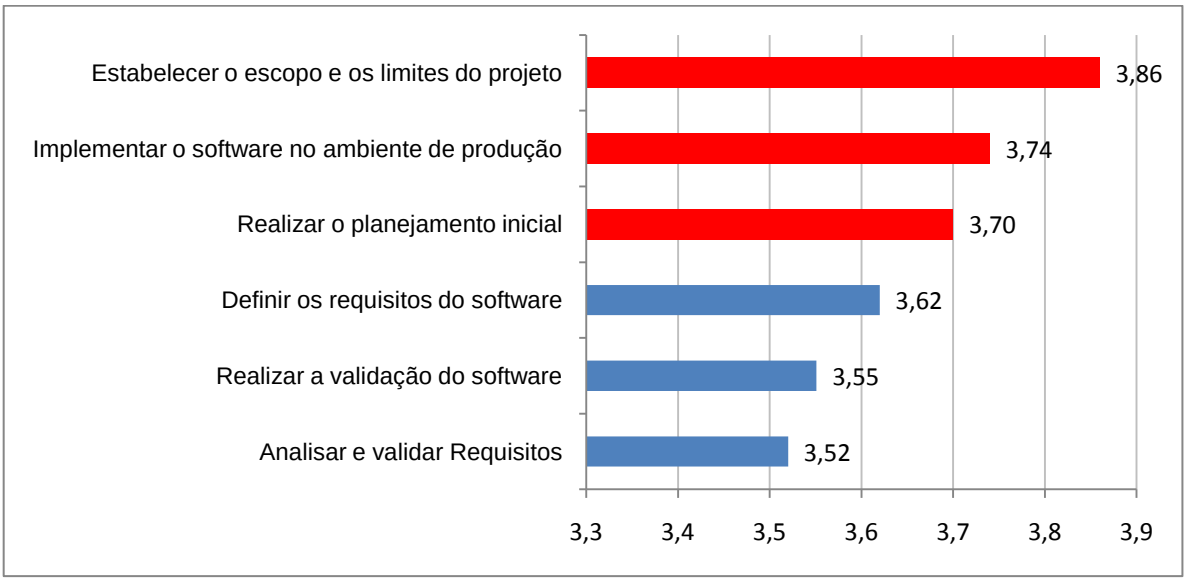

Figura 3.1 – As seis atividades melhor avaliadas para o método de auditoria no processo de desenvolvimento de software (Fonte: O Autor)

A atividade “Realizar o planejamento inicial” foi a terceira na avaliação, com o valor de 3,70, próximo do valor de muito importante estabelecido para um método de auditoria no processo de desenvolvimento de software. Isto reforça a importância de um planejamento inicial abrangente o suficiente para traduzir os requisitos funcionais de negócio e de controle em um projeto eficiente e eficaz de desenvolvimento de software, inclusive prevendo a avaliação dos riscos relacionados com a informação e os riscos de processamento das informações relacionadas.

A atividade “Definir os requisitos do software” foi a quarta com maior valor médio de avaliação, 3,62, posicionando-se entre a escala de valor importante e valor muito importante para um método de auditoria no processo de desenvolvimento de software. Isto ressalta a necessidade de se verificar a definição de um conjunto consistente de requisitos, que são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais, que reflitam as necessidades dos clientes do sistema a ser desenvolvido.

A atividade “Realizar a validação do software” foi a quinta pontuada e obteve um valor de 3,55. Isto mostra que devem ser realizados testes para validar se o software desenvolvido atende ao propósito pretendido e se está livre de erros, prevendo as correções necessárias dos erros identificados, antes de planejar a implementação e a migração para a produção.

A atividade “Analisar e validar Requisitos” obteve o valor de 3,52, evidenciando sua posição entre importante e muito importante. Portanto, devem ser realizadas validações dos requisitos elicitados no projeto de desenvolvimento de software para identificar se definem o que o sistema deve fazer, suas propriedades emergentes desejáveis e essenciais e as restrições quanto à operação do sistema e quanto aos processos de desenvolvimento de software.

Comparando-se os resultados das avaliações realizadas por especialistas no processo de desenvolvimento de software e em auditoria de sistemas, é percebido que três das seis atividades melhor pontuadas são as mesmas tanto na avaliação dos especialistas em processo de software quanto na avaliação dos especialistas em auditoria de sistemas, destacado em vermelho na Figura 3.1. Essas atividades de desenvolvimento de software foram: “Estabelecer o escopo e os limites do projeto”, “Implementar o software no ambiente de produção” e “Realizar o planejamento inicial”. Essas três atividades obtiveram um valor maior ou igual a 3,55. Portanto, essas atividades foram consideradas de muita importância na estruturação de um método de auditoria para o processo de desenvolvimento de software.

A Figura 3.2 apresenta as três atividades que obtiveram os menores valores, em relação ao seu grau de importância, na avaliação de um método de auditoria no processo de desenvolvimento de software, segundo os especialistas desta pesquisa. Elas representam as atividades que obtiveram avaliações iguais ou menores do que 2,85. Nas demais representações procurou-se seguir o padrão de três atividades para aquelas pior avaliadas.

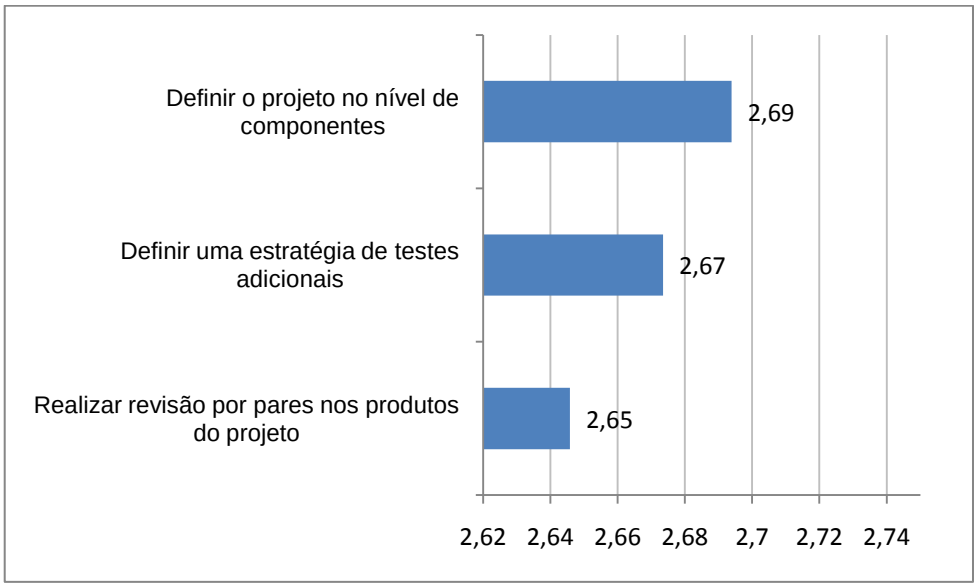

Figura 3.2 – As três atividades de menor grau de avaliação para um método de auditoria no processo de desenvolvimento de software (Fonte: O Autor)

A atividade “Realizar revisão por pares nos produtos do projeto” foi considerada de menor importância em processo para apoiar a segurança de software, obtendo o valor de 2,65. Isto significa que os especialistas deram importância mediana, tendendo para importante, à questão de prever revisões por pares, que constituem um exame metódico dos produtos de trabalho pelos pares dos responsáveis pela construção do produto com o intuito de identificar defeitos a serem removidos e recomendar outras modificações que sejam necessárias.

As atividades “Definir uma estratégia de testes adicionais” e “Definir o projeto no nível de componentes” também tiveram uma avaliação de importância mediana, contudo mais próximas de importante, com valores de 2,67 e 2,69, respectivamente. Assim sendo, é considerada de alguma relevância a definição de uma estratégia de testes adicionais para validação de softwares como, por exemplo, de sistemas críticos, que ratifiquem a segurança do software. Também é considerada de alguma relevância a criação de modelos para se obter um melhor entendimento da entidade real a ser construída, pois ela precisa ser capaz de representar a informação que o software transforma, a arquitetura e funções que permitem que essa transformação ocorra, as características que os usuários desejam e o comportamento do sistema à medida que a transformação ocorre.

Em relação às atividades com a menor pontuação, nenhuma atividade foi comum em ambas as avaliações dos especialistas.

3.5. Avaliação dos Especialistas em Processo de Desenvolvimento de Software

A Figura 3.3 mostra, em ordem decrescente de seu grau de importância, as seis atividades do método de auditoria proposto melhor avaliadas pelos especialistas em processos de software.

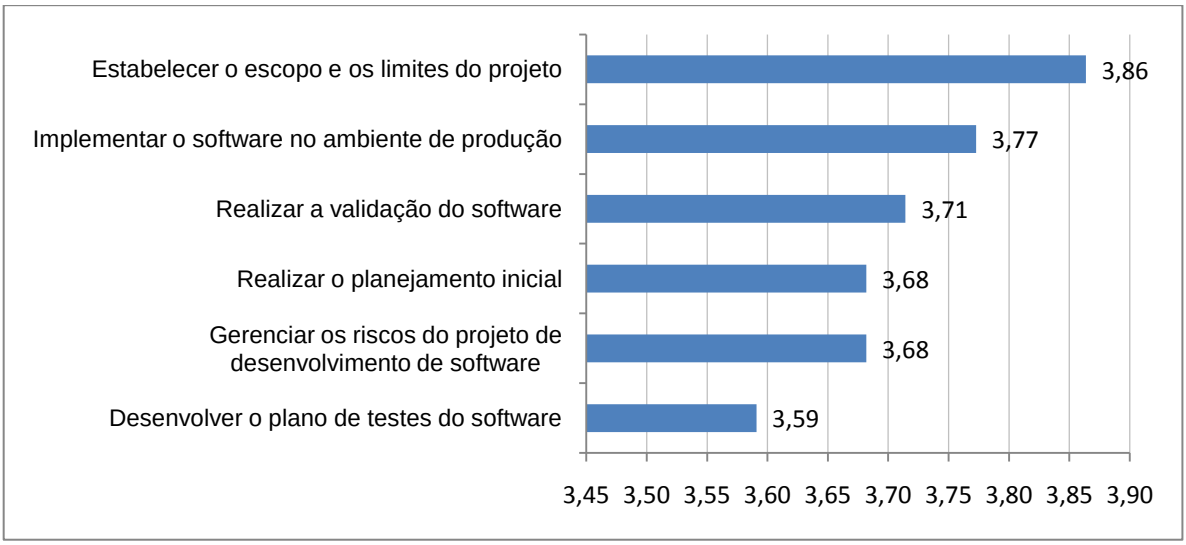

Figura 3.3 – As atividades melhor avaliadas por especialistas em processos de software (Fonte: O Autor)

A atividade “Estabelecer o escopo e os limites do projeto” foi a avaliada com o valor mais alto, 3,86, tendendo para o valor máximo da escala de avaliação. Isto evidencia a relevância de se definir o escopo do software e os limites do projeto, ou seja, as funções e características que deverão ser entregues aos usuários finais, os dados de entrada e saída, as informações que serão apresentadas aos usuários como consequência do uso do software e o desempenho, as restrições, as interfaces e a confiabilidade que limitarão o sistema.

A atividade “Implementar o software no ambiente de produção” a segunda em importância na avaliação, com o valor 3,77, também tendendo para o valor máximo da escala de avaliação. Isto evidencia que é altamente recomendável que sejam definidas atividades de controle para verificar se o software foi disponibilizado no ambiente de produção em alinhamento com os requisitos do negócio, no prazo desejado e com um custo razoável.

A terceira na avaliação foi a “Realizar a validação do software” com o valor de 3,71. Isto evidencia a relevância da realização dos testes para validar se o software desenvolvido atende ao propósito pretendido e se está livre de erros, prevendo as correções necessárias aos erros identificados, antes de planejar a implementação e a migração para a produção.

As atividades “Realizar o planejamento inicial” e “Gerenciar os riscos do projeto de desenvolvimento de software” foram todas avaliadas com o valor de 3,68, convergindo também para o patamar de muito importante em um método de auditoria.

Neste contexto, é altamente recomendável que seja realizada:

(i) um planejamento inicial abrangente o suficiente para traduzir os requisitos funcionais de negócio e de controle em um projeto eficiente e eficaz de desenvolvimento de software, inclusive prevendo a avaliação dos riscos relacionados com a informação e os riscos de processamento das informações relacionadas;

(ii) a definição dos riscos do projeto, a identificação dos riscos de grande impacto e o planejamento de contingências, que estabeleçam o nível de segurança para que o projeto seja realizado e o produto entregue ao cliente.

A atividade “Desenvolver o plano de testes do software” foi bem pontuada e obteve um valor de 3,59. Isto mostra que se deve estabelecer um plano de teste baseado nos padrões organizacionais que definem papéis, responsabilidades e critérios de sucesso de entrada e saída e assegurar que o plano seja aprovado pelas partes relevantes.

A figura 3.4 apresenta as três atividades que obtiveram os menores valores, em relação ao seu grau de importância, na avaliação de um método de auditoria, segundo os especialistas em processo de software.

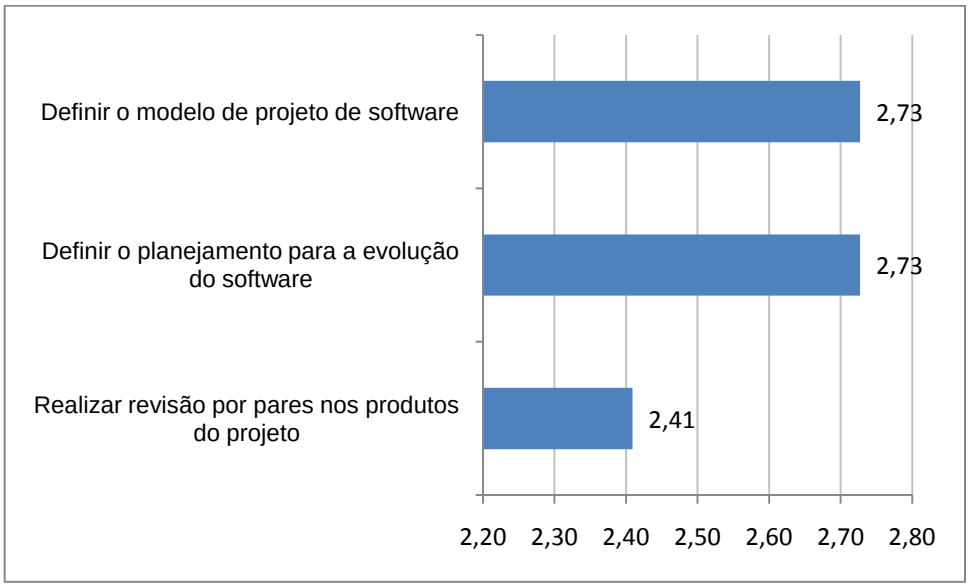

Figura 3.4 – As atividades de menor grau de avaliação por especialistas em processos de software (Fonte: O Autor)

A atividade “Realizar revisão por pares nos produtos do projeto” foi a considerada de menor importância em um método de auditoria, obtendo o valor de 2,41, em que os especialistas em processo de software deram importância mediana à questão de prever revisões por pares, com o intuito de identificar defeitos a serem removidos e recomendar outras modificações que sejam necessárias.

As atividades “Definir o planejamento para a evolução do software” e “Definir o modelo de projeto de software” também tiveram uma avaliação de importância mediana, contudo mais próximas de importante, com valores de 2,73. Assim sendo, é considerada de razoável importância definir um processo formal para evolução de software que inclua as atividades fundamentais de análise de mudanças, planejamento de *releases* e implementação de mudanças. É também considerada de média importância, com tendência para importante, a definição do modelo de projeto para se obter uma visão mais completa do software a ser desenvolvido.

3.6. Avaliação dos Especialistas em Auditoria de Sistemas

A Figura 3.5 apresenta, em ordem decrescente do grau de importância, as seis atividades do método de auditoria proposto melhor avaliadas pelos especialistas em auditoria de sistemas.

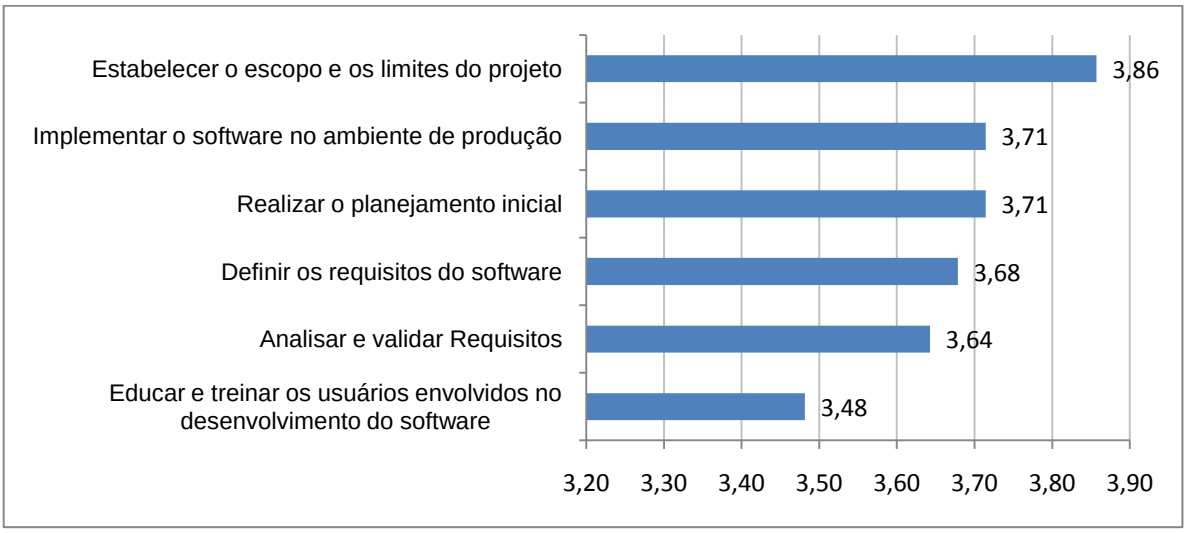

Figura 3.5 – As atividades melhor avaliadas por especialistas em auditoria de sistemas (Fonte: O Autor)

A atividade “Estabelecer o escopo e os limites do projeto” foi também a avaliada pelos especialistas em auditoria de sistemas com o valor mais alto, 3,86, tendendo para muito importante em um método de auditoria. Isto evidencia a relevância de se definir o escopo do software e os limites do projeto, ou seja, as funções e características que deverão ser entregues aos usuários finais, os dados de entrada e saída, as informações que serão apresentadas aos usuários como consequência do uso do software e o desempenho, as restrições, as interfaces e a confiabilidade que limitarão o sistema.

As atividades “Implementar o software no ambiente de produção” e “Realizar o planejamento inicial” foram a segunda e terceira avaliadas com o valor mais alto, 3,71. Neste contexto, é altamente recomendável que:

(i) sejam definidas atividades de controle para verificar se o software foi disponibilizado no ambiente de produção em alinhamento com os requisitos do negócio, no prazo desejado e com um custo razoável;

(ii) seja realizado um planejamento inicial abrangente o suficiente para traduzir os requisitos funcionais de negócio e de controle em um projeto eficiente e eficaz de desenvolvimento de software, inclusive prevendo a avaliação dos riscos relacionados com a informação e os riscos de processamento das informações relacionadas.

A atividade “Definir os requisitos do software” foi a quarta avaliada com o valor mais alto, 3,68, próximo do valor de muito importante estabelecido em um método de auditoria. Neste contexto, é altamente recomendável que seja verificada a definição de um conjunto consistente de requisitos, que são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais, que reflitam as necessidades dos clientes do sistema a ser desenvolvido.

A atividade “Analisar e validar Requisitos” foi avaliada com o valor de 3,64, sendo muito bem avaliada. Neste contexto, é altamente recomendável que sejam realizadas validações dos requisitos elicitados no projeto de desenvolvimento de software para identificar se definem o que o sistema deve fazer, suas propriedades emergentes desejáveis e essenciais e as restrições quanto à operação do sistema e quanto aos processos de desenvolvimento de software.

A atividade “Educar e treinar os usuários envolvidos no desenvolvimento do software” obteve o valor de 3,48, evidenciando sua posição entre importante e muito importante. Portanto, devem-se identificar as necessidades de ensino e treinamento para a equipe envolvida na implementação do software.

A Figura 3.6 apresenta as três atividades que obtiveram os menores valores, em relação ao seu grau de importância, na avaliação de um método de auditoria, segundo os especialistas em auditoria de sistemas.

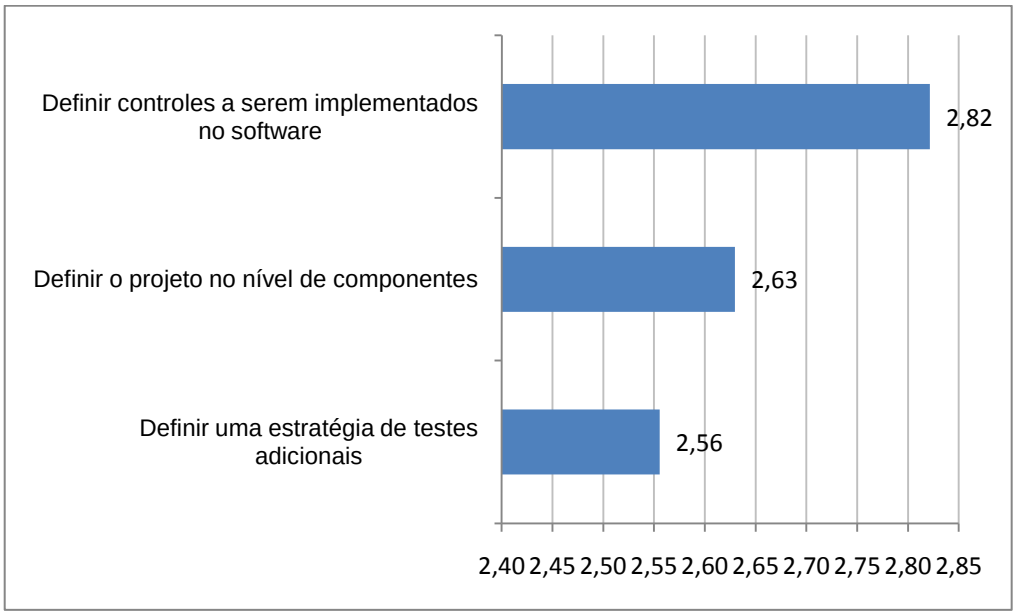

Figura 3.6 – As atividades de menor grau de avaliação por especialistas em auditoria de sistemas (Fonte: O Autor)

As atividades “Definir o projeto no nível de componentes” e “Definir uma estratégia de testes adicionais” foram consideradas de menor grau de importância em um método de auditoria no desenvolvimento de software baseado no processo unificado, obtendo o valor de 2,63 e 2,56 respectivamente, na opinião dos especialistas em auditoria de sistemas. Isto significa que esses especialistas deram importância mediana à questão da criação de modelos para se obter um melhor entendimento da entidade real a ser construída e de uma estratégia de testes adicionais para validação de softwares como, por exemplo, de sistemas críticos, que ratifiquem a segurança do software.

A atividade “Definir controles a serem implementados no software” também teve uma avaliação de importância mediana com valor de 2,82. Assim sendo, é considerada de média importância, com tendência para importante, assegurar que os controles de negócio sejam expressos adequadamente nos controles dos aplicativos de forma que o processamento ocorra no prazo correto e seja exato, completo, autorizado e auditável.

3.7. Aspectos que Podem Influenciar a Auditoria no Processo Unificado de Desenvolvimento de Software

Este trabalho propõe um conjunto de aspectos que poderiam influenciar a utilização de um método de auditoria no processo unificado de desenvolvimento de software. A Tabela 3.2 mostra, em ordem decrescente do grau de importância (média aritmética e desvio padrão), aspectos que foram analisados pelos especialistas na pesquisa de campo. Esses aspectos representam ações estratégicas e organizacionais que podem promover um método de auditoria para apoio ao desenvolvimento de software com o objetivo de garantir a entrega do produto.

O aspecto “Apoio da direção da empresa” foi o melhor avaliado com o valor de 3,72, tendendo para muito influente à realização de auditorias no desenvolvimento de softwares. Isto reforça a relevância do comprometimento dos gestores, nesse caso a alta direção, com o projeto onde será implantada a prática de auditoria no desenvolvimento de software.

Tabela 3.2 – Aspectos que podem influenciar o apoio da auditoria de sistemas num processo de desenvolvimento de software (Fonte: O Autor)

Aspectos que podem influenciar o apoio da auditoria de sistemas num processo de desenvolvimento de software	Média	Desvio Padrão
Apoio da direção da empresa	3,72	0,49
Alto comprometimento da gerência com o projeto	3,68	0,51
Responsabilidades claramente definidas	3,54	0,64
O processo de desenvolvimento de software estar alinhado aos objetivos de negócio da organização	3,50	0,67
Disciplina no atendimento às recomendações geradas pela equipe de auditoria de sistemas	3,28	0,66
O processo de desenvolvimento de software estar baseado em expectativas realistas	3,28	0,78
Treinamento formal da equipe no processo de desenvolvimento de software	3,22	0,76
Equipe com experiência em auditoria no desenvolvimento de software	3,14	0,85
Gerência com experiência em auditoria no desenvolvimento de software	3,10	0,96
Existência de orientações para a realização da auditoria no desenvolvimento de software	3,06	0,61
Participação do cliente nas etapas do processo	2,98	0,88
Existência de auditorias da aderência ao processo de desenvolvimento de software	2,94	0,84
Apoio à utilização do conhecimento de experiências em projetos anteriores	2,86	0,69
Iniciar a implantação da auditoria no desenvolvimento de softwares por um conjunto específico de atividades	2,78	0,78
Existência de um grupo de Auditoria de Sistemas na empresa	2,78	1,02
Iniciar a implantação da auditoria no desenvolvimento de softwares de maneira uniforme	2,76	0,81
Ambiente físico de trabalho adequado	2,76	0,95
Existência de apoio automatizado	2,50	0,88

O aspecto “Alto comprometimento da gerência com o projeto” foi o segundo avaliado com o valor mais alto, 3,68, convergindo para o patamar de muito influente à implantação de auditoria no desenvolvimento de software. Isto evidencia a relevância do comprometimento dos gestores de negócio e gestores de tecnologia com o projeto onde será implantada a prática de auditoria no desenvolvimento de software.

O aspecto “Responsabilidades claramente definidas” foi o terceiro avaliado com o valor mais alto, 3,54, posicionando-se entre a escala de valor influente e valor muito influente para a realização de auditoria no desenvolvimento de software. Isto ressalta a necessidade de definir responsabilidades no projeto para obter sucesso no desenvolvimento de software e entrega do produto.

O aspecto “O processo de desenvolvimento de software estar alinhado aos objetivos de negócio da organização” foi o quarto avaliado com o valor mais alto, 3,50, próximo do valor de muito influente estabelecido para se obter bons resultados na realização de auditoria no desenvolvimento de software. Isto ratifica a necessidade do envolvimento da área de negócio no projeto de desenvolvimento de software para que as decisões sejam alinhadas com os objetivos de negócio da organização. Esse aspecto oferece maiores garantias de que o software satisfará as estratégias de negócio.

O aspecto “Existência de apoio automatizado” foi considerado de média influência na utilização de um método de auditoria e o que obteve menor avaliação, obtendo o valor de 2,50. Isto significa que os especialistas deram pouca influência à questão de possuir ferramentas automatizadas para apoiar a realização de auditoria no desenvolvimento de software.

O aspecto “Ambiente físico de trabalho adequado” também recebeu uma avaliação de influência mediana, contudo mais próxima de influente, com valor de 2,76. Assim sendo, a ausência de recursos satisfatórios para o projeto de desenvolvimento de software pode dificultar a participação dos *stakeholders*, porém não é visto como um fator que irá diminuir as chances de sucesso na entrega do produto.

3.8. Estruturação do Método de Auditoria ao Desenvolvimento de Software Baseado no Processo Unificado

A partir dos resultados da pesquisa de campo realizada, o método de auditoria proposto foi finalmente estruturado. Houve redução de 55 atividades propostas inicialmente para 36, pois algumas atividades foram refinadas e integradas em seu conteúdo ou abrangência, como apresentado a seguir:

1. “Definir processos e procedimentos para definição dos requisitos do software” e “Definir políticas, processos e planos para orientar o desenvolvimento do software” foram agrupadas na atividade “Definir políticas, processos e procedimentos para as etapas de desenvolvimento do software”.

2. “Definir o modelo do sistema” foi integrada à atividade “Definir os requisitos do software”.

3. “Definir o projeto de desenvolvimento do software” e “Definir o projeto no nível de componentes” foram integradas à atividade “Definir o planejamento do projeto de desenvolvimento de software”.

4. “Gerenciar o projeto de software”, “Gerenciar os requisitos do software” e “Gerenciar os riscos do projeto de desenvolvimento de software” foram integradas à atividade “Gerenciar o projeto de desenvolvimento do software”.

5. “Desenvolver o software conforme as mudanças requeridas” foi integrada à atividade “Desenvolver o software conforme o projeto”.

6. “Gerenciar a configuração do software” foi integrada à atividade “Gerenciar configurações do software”. Estas atividades apareceram em partes distintas (Concepção, Elaboração, Construção ou Transição) de mais de um padrão de referência, porém não haviam sido identificadas como equivalentes.

7. “Implementar o software no ambiente de produção” foi renomeada para “Gerenciar a implantação do software no ambiente de produção”.

Após esta adequação, foram eliminadas 19 atividades que apareceram mais de uma vez na pesquisa, em fases diferentes do PU, com o intuito de destacar que, em alguns casos, podem demandar esforços de teste em fases distintas.

As seguintes atividades foram mantidas:

- Adquirir e Manter a Infraestrutura de Tecnologia;
- Analisar e validar Requisitos;
- Atualizar a documentação do usuário;
- Definir a especificação formal do software;
- Definir a modelagem de análise do software;
- Definir controles a serem implementados no software;
- Definir especificação para sistemas críticos;

- Definir estratégias de testes de software;
- Definir o modelo de projeto de software;
- Definir o planejamento para a evolução do software;
- Definir o planejamento para validação do software;
- Definir o planejamento para verificação do software;
- Definir o projeto de arquitetura do software;
- Definir o projeto de interface com o usuário;
- Definir o projeto de testes de software;
- Definir os requisitos do software;
- Definir técnicas para o desenvolvimento de sistemas críticos;
- Definir uma estratégia de integração;
- Definir uma estratégia de testes adicionais;
- Definir uma estratégia de validação;
- Desenvolver o plano de testes do software;
- Estabelecer o escopo e os limites do projeto;
- Gerenciar a qualidade do software;
- Gerenciar mudanças no software;
- Realizar a validação do software;
- Realizar a verificação do software;
- Realizar auditoria técnica do software;
- Realizar o planejamento inicial;
- Realizar revisão por pares nos produtos do projeto.

3.9. Síntese da Pesquisa de Campo

Este capítulo apresentou e analisou os resultados de uma pesquisa de campo realizada com especialistas em processo de desenvolvimento de software e

especialistas em auditoria de sistemas, considerando um conjunto de atividades para o método de auditoria para projetos de software baseados no Processo Unificado.

A avaliação das atividades pelos especialistas possibilitou identificar as mais importantes, e consequentemente, elaborar o método com base nessa avaliação. Além disso, foi possível identificar atividades repetidas e corrigir esta inconsistência.

Destacam-se dois pontos importantes percebidos com o resultado da pesquisa e que forneceram maior garantia na adequada condução da pesquisa:

- As diversas coincidências no resultado das avaliações dos especialistas em processo de software e dos especialistas em auditoria de sistemas;
- Apesar de haver atividades consideradas menos e mais importantes, as diferenças identificadas em todos os resultados das avaliações não são muito significativas entre a atividade pior pontuada e a atividade melhor pontuada. A maior diferença de valores, de 2,65 a 3,86, obtida na avaliação conjunta de todos os especialistas, conforme Tabela 3.1, demonstra que mesmo as “piores” atividades ainda têm certa importância para o processo unificado de desenvolvimento de software e, consequentemente, para o método de auditoria proposto. Devido a essa constatação, foram mantidas as 36 atividades no processo final.

No capítulo seguinte, com base na pesquisa de campo realizada, será apresentado o método de auditoria proposto.

4. Proposta de Método de Auditoria para Projetos de Software Baseados no Processo Unificado

O método de auditoria proposto trata do acompanhamento sobre o processo de desenvolvimento de software, a partir de um conjunto de atividades, relacionadas no capítulo 3, a serem executadas ao longo do ciclo de vida, visando obter melhores resultados na entrega do produto nos projetos de software.

4.1. Objetivo

O objetivo deste método é obter e apresentar os requisitos mínimos de controles padronizados para avaliação dos projetos de software que seguem a abordagem iterativa e incremental, buscando desta forma auxiliar na identificação de vulnerabilidades que possam afetar a qualidade do produto.

4.2. Alcance

O alcance deste método é o de estabelecer avaliação independente das atividades aplicáveis aos projetos de desenvolvimento de software, de forma a identificar potenciais falhas que possam comprometer a qualidade, prazos, custos, e aderência aos requisitos dos *stakeholders*.

4.3. Referências

Este método foi definido com base nos seguintes padrões de referência:

- Cobit 4.1 (ITGI, 2007), do qual se aproveitaram os objetivos de controle de Governança de TI relacionados ao processo de desenvolvimento de software;
- *CMMI for Development* (SEI, 2011), da qual se aproveitaram as boas práticas relacionadas à maturidade da organização no processo de desenvolvimento de software;
- ISO/IEC 15408 (ISO/IEC/JTC, 2009), da qual se observaram conceitos e se aproveitaram boas práticas que visam requisitos de segurança no desenvolvimento de software;

- MPS.BR - Melhoria de Processos do Software Brasileiro (SOFTEX, 2012), da qual se observaram boas práticas relacionadas à maturidade da organização no processo de desenvolvimento de software;
- NBR ISO/IEC 12207 (ABNT, 2008), da qual se aproveitaram a estruturação em que se classificam os processos e boas práticas do ciclo de desenvolvimento e toda a terminologia em vigor;
- NBR ISO/IEC 15504 (ABNT, 2008), da qual se observaram a estruturação evoluída da NBR ISO/IEC 12207, a abordagem “contínua” para a melhoria da capacitação do processo e toda a terminologia em vigor;
- NBR ISO/IEC 27002 (ABNT, 2005), da qual se aproveitam conceitos e boas práticas que visam requisitos de segurança no desenvolvimento de sistemas;
- NBR ISO/IEC 9126-1 (ABNT, 2003), da qual se aproveitaram boas práticas e conceitos relacionados à qualidade de software.

Há diversas normas técnicas relacionadas com software além das já citadas. Entre as demais normas existentes, podem-se citar:

- ISO/IEC 12119:1994 – Pacotes de software – Requisitos de qualidade e testes;
- ISO/IEC 14598-1:1999 – Avaliação de qualidade de produtos de software;
- ISO/IEC 25000:2005 – Modelo de qualidade de software, nova versão das séries 14.598 e 9.126;
- ISO/IEC 9241:1998 – Ergonomia de software;
- ISO/IEC 20926:2003 – Medida de software por pontos de função;
- ISO/IEC 90000-3:2004 – Diretivas para aplicação da ISO 9001 ao software;
- ISO/IEC 9001:2000 – Requisitos para sistemas de gerenciamento de qualidade (aplicável a qualquer empresa, de software ou não);

- SQuaRE:ISO/IEC 25000 – Requisitos de Qualidade e Avaliação de Produtos de Software.

4.4. Características do processo unificado de desenvolvimento de software

O processo unificado (*Unified Process* - UP) de desenvolvimento de software é o conjunto de atividades necessárias para transformar requisitos do usuário em um sistema de software. O UP de desenvolvimento de sistemas combina os ciclos iterativo e incremental para a construção de softwares. É fundamental na visão de que o avanço de um projeto deve estar baseado na construção de artefatos de software, e não apenas em documentação.

O *Rational Unified Process* (RUP) é um exemplo de modelo de processo que foi derivado do trabalho sobre a UML e do Processo Unificado de Desenvolvimento de Software associado, sendo um modelo híbrido de processo, pois traz elementos de todos os modelos genéricos de processo, apoia a iteração e ilustra boas práticas de especificação e projeto.

O RUP é um modelo constituído por fases que identifica quatro fases discretas no processo de software. No entanto, ao contrário do modelo em cascata, no qual as fases coincidem com as atividades do processo, as fases do RUP estão relacionadas mais estritamente aos negócios do que a assuntos técnicos. São elas (Figura 4.1):

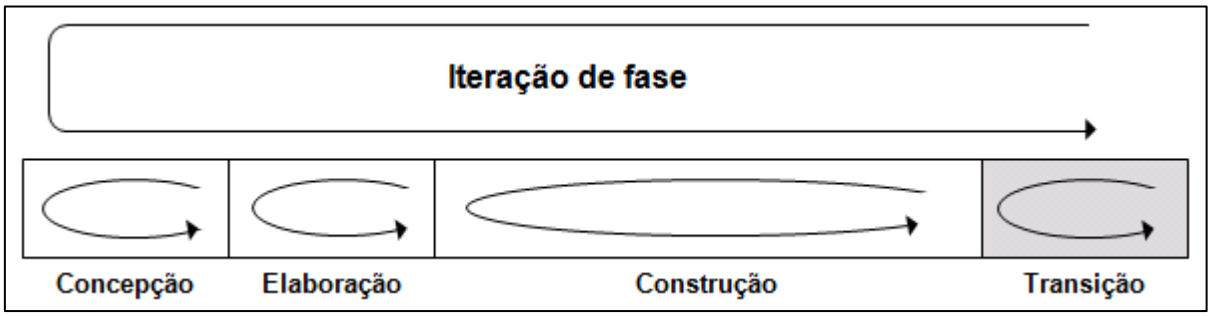

Figura 4.1 – As fases do RUP (PRESSMAN, 2011)

a) Concepção

O objetivo dessa fase é estabelecer um *business case* para o sistema. Devem-se identificar todas as entidades externas (pessoas e sistemas), que irão

interagir com o sistema, e definir essas interações. Depois você usa essas informações para avaliar a contribuição do sistema com o negócio. Se a contribuição for de pouca importância, o projeto pode ser cancelado depois dessa fase.

b) Elaboração

Os objetivos da fase de elaboração são desenvolver um entendimento do domínio do problema, estabelecer um *framework* de arquitetura para o sistema, desenvolver o plano de projeto e identificar os riscos principais do projeto. Ao concluir essa fase, deve-se ter um modelo de requisitos para o sistema (os casos de uso da UML são especificados), uma descrição de arquitetura e um plano de desenvolvimento para o software.

c) Construção

A fase de construção está essencialmente relacionada ao projeto, programação e teste de sistema. As partes do sistema são desenvolvidas paralelamente e integradas durante esta fase. Ao concluir esta fase, deve-se ter um sistema de software em funcionamento e a documentação associada pronta para ser liberada para os usuários.

d) Transição

A fase final do RUP está relacionada à transferência do sistema da comunidade de desenvolvimento para a comunidade dos usuários e com a entrada do sistema em funcionamento no ambiente real. Isso é algo ignorado na maioria dos modelos de processo de software, mas é, de fato, uma atividade onerosa e, às vezes, problemática. Ao concluir esta fase, deve-se ter um sistema de software documentado, funcionando corretamente em seu ambiente operacional.

A iteração no RUP é considerada em duas formas, como apresentado na Figura 4.1. Cada fase pode ser realizada de forma iterativa, com os resultados desenvolvidos incrementalmente. Além disso, o conjunto total das fases pode também ser realizado de forma incremental, conforme apresentado pela seta retornando da Transição para a Concepção na Figura 4.1

4.5. Atividades de Controle para Avaliação de Projetos de Software Baseados no Processo Unificado

Cada uma das atividades de controle definidas nesse método deve ser avaliada pelo condutor do trabalho, de acordo com o ambiente em avaliação e suas particularidades. A Figura 4.2 apresenta uma matriz, desenvolvida em software Microsoft Excel, sugerida como modelo para o registro dos exames, avaliação das atividades e repositório para armazenamento de evidências e comentários pertinentes.

Nº OC	Objetivo de Controle	Nº AC	Atividade de Controle	Avaliação	Cód. Evidência	Comentários
Concepção						
1	Garantir que sejam estabelecidos o escopo e viabilidade econômica do projeto	1	Estabelecer o escopo e os limites do projeto			
		2	Realizar o planejamento inicial			
Elaboração						
2	Garantir que sejam definidas a funcionalidade do software, seus requisitos e restrições sobre sua operação	3	Analisar e validar Requisitos			
		4	Definir a especificação formal do software			
		5	Definir especificação para sistemas críticos			
		6	Definir o planejamento do projeto de desenvolvimento			
		7	Definir o projeto de arquitetura do software			
		8	Definir o projeto de interface com o usuário			
		9	Definir os requisitos do software			
		10	Definir políticas, processos e procedimentos para as etapas de desenvolvimento do software			
		11	Educar e treinar os usuários envolvidos no desenvolvimento do software			
		12	Gerenciar a qualidade do software			
		13	Gerenciar o projeto de desenvolvimento do software			
Construção						
3	Garantir a entrega de um sistema de software em funcionamento, de acordo com sua especificação, e da documentação associada pronta para ser liberada para os usuários	14	Adquirir e Manter a Infraestrutura de Tecnologia			
		15	Atualizar a documentação do usuário			
		16	Definir a modelagem de análise do software			
		17	Definir controles a serem implementados no software			
		18	Definir estratégias de testes de software			
		19	Definir o modelo de projeto de software			
		20	Definir o planejamento para a evolução do software			
		21	Definir o planejamento para validação do software			
		22	Definir uma estratégia de validação			
		23	Definir uma estratégia de integração			
		24	Definir o planejamento para verificação do software			
		25	Definir técnicas para o desenvolvimento de sistemas críticos			
		26	Definir o projeto de testes de software			
		27	Definir uma estratégia de testes adicionais			
		28	Desenvolver o plano de testes do software			
		29	Desenvolver o software conforme o projeto			
		30	Gerenciar mudanças no software			
		31	Realizar auditoria técnica do software			
		32	Realizar revisão por pares nos produtos do projeto			
Transição						
4	Garantir que seja entregue um sistema de software documentado, funcionando corretamente em seu ambiente operacional e que seja validado pelo cliente para comprovar que atende aos seus requisitos	33	Gerenciar a implantação do software no ambiente de produção			
		34	Gerenciar configurações do software			
		35	Realizar a validação do software			
		36	Realizar a verificação do software			

Figura 4.2 – Modelo de planilha para trabalho de auditoria (Fonte: O Autor)

A planilha contempla o primeiro nível de descrição das atividades de controle a serem examinadas, seguindo o layout abaixo:

- Macro Atividade: as etapas que compõem o ciclo do processo unificado;
- Nº OC / Objetivo de Controle: uma orientação do objetivo da avaliação a ser realizada (o que se pretende garantir com o teste); a seguir, a descrição detalhada das atividades e exames ou evidências a serem observadas serão iniciadas por este dado;
- Nº AC / Atividade de Controle: um agrupamento macro das atividades quanto ao seu enquadramento nas fases ou etapas do projeto de software; é indicado também que tipo de teste/evidência deve ser observado, bem como possíveis testes/evidências a serem realizados/verificados.

A seguir são apresentadas as atividades de controle consideradas nesse método de auditoria.

4.5.1. OC 1 - Escopo e Viabilidade Econômica do Projeto

O objetivo é garantir que sejam realizadas as atividades necessárias para estabelecer o escopo e avaliar a viabilidade econômica do projeto. Os procedimentos a serem seguidos são:

a) AC 1 - Obter os documentos em que foram estabelecidos o escopo e os limites do projeto, verificando se:

- estão identificados os principais *stakeholders* que têm um interesse legítimo no sistema ao longo do seu ciclo de vida e suas responsabilidades;
- contemplam as necessidades do software e foram especificadas as interfaces com outros sistemas;
- foi realizado um estudo de viabilidade e foi devidamente validado pelo cliente;

- foram estabelecidos conceitos operacionais e cenários, bem como uma definição da funcionalidade requerida;
- foram realizadas as estimativas de esforço e custo;
- os documentos estão de acordo com as práticas e padrões de qualidade definidas (metodologia, políticas e procedimentos para o desenvolvimento de software);
- foi documentada a natureza e o escopo do projeto, visando confirmar e desenvolver um entendimento comum do escopo do projeto com os *stakeholders* e quanto ao relacionamento com outros projetos de um programa de investimento em TI.

b) AC 2 - Obter o planejamento inicial do projeto de software, verificando se:

- foram realizados e validados os resultados de estudos de viabilidade;
- foram determinadas as estimativas de esforço e custo;
- foram identificados e especificados os riscos do projeto.

4.5.2. OC 2 - Requisitos do Software

Garantir que sejam definidas a funcionalidade do software, seus requisitos e restrições sobre sua operação. Os procedimentos a serem seguidos são:

a) AC 3 - Obter os documentos de requisitos do software, verificando se:

- foram validados os requisitos do software (funcionais e não funcionais, de cliente, de usuário, de *stakeholders*, de sistema, de software, técnicos e de negócio, de interface (IHC), para controles de segurança, ambiente e critérios de segurança, de produto, de componente de produto e de qualificação);
- foram validados os requisitos de qualidade externa e interna do software, compreendendo os seguintes aspectos (se aplicável): funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.

b) AC 4 - Obter a especificação formal do software, verificando se:

- foram definidos: processo de software, interface de subsistema e comportamento.

c) AC 5 - Obter a especificação para sistemas críticos, verificando se:

- foram definidas: especificações dirigidas a riscos, segurança, proteção e confiabilidade de software.

d) AC 6 - Obter os documentos de planejamento do projeto de desenvolvimento de software, verificando se:

- foram estabelecidos o orçamento, cronograma, plano do projeto e um processo definido para o projeto, bem como definido e validado junto ao cliente o escopo do projeto, considerando: necessidade, valores e restrições do negócio, características/necessidades dos usuários finais, saídas visíveis requeridas ao usuário, cenários de uso visíveis ao cliente usando o formato padrão, saídas e entradas resultantes, características, funções e comportamentos importantes do software e riscos de negócios definidos pelo cliente;
- foram definidas as atividades de planejamento de gestão dos dados e dos recursos do projeto, bem como habilidades e conhecimentos necessários ao projeto e envolvimento dos *stakeholders*;
- foi definida atividade de revisão dos planos que afetam o projeto;
- foram definidas as características técnicas que formam a infraestrutura do software, agrupadas de acordo com a prioridade do cliente;
- foi definido o projeto no nível de componentes a partir de notações no contexto da engenharia de software orientada a objetos;
- foi definido o ciclo de vida do projeto e as tarefas de trabalho, a responsabilidade por cada tarefa e produtos de trabalho a serem produzidos;
- foi criado o projeto de software, abrangendo a arquitetura do software, definição de subsistemas e estabelecimento de

mecanismos de comunicação entre subsistemas, identificação e descrição detalhada de cada um dos componentes e projeto de interfaces externas, internas e com o usuário.

e) AC 7 - Obter a documentação do projeto de arquitetura do software, verificando se:

- foram traduzidos os requisitos de negócio nas especificações de projeto;
- o projeto prevê atividades de avaliação para garantir que as funcionalidades automatizadas foram desenvolvidas em conformidade com as especificações de projeto, padrões de desenvolvimento e documentação e requisitos de qualidade e de autorização;
- o projeto contempla:
    - sistema representado no contexto, definindo as entidades externas com as quais o software interage e a natureza da interação;
    - identificação de um conjunto de abstrações de mais alto nível chamado de arquétipos que representam os elementos-chave do comportamento ou função do sistema;
    - identificação e representação dos componentes no contexto de uma arquitetura que lhes dá suporte;
    - desenvolvimento de instanciações específicas da arquitetura para provar o projeto em um contexto do mundo real.

f) AC 8 - Obter a documentação do projeto de interface com o usuário, verificando se:

- contempla a identificação dos usuários finais do sistema, análise de tarefas e ações do usuário e análise ambiental, identificando as estruturas físicas e sociais nas quais a interface deve operar.

g) AC 9 - Obter a documentação de requisitos do software, verificando se:

- foram definidos os requisitos do software (requisitos funcionais e não funcionais, requisitos de cliente, usuário e *stakeholders*, requisitos de sistema, requisitos técnicos e de negócio, requisitos de interface (IHC), requisitos para controles de segurança, requisitos de produto e de componente de produto e requisitos de qualificação);
- os requisitos do software foram definidos e validados pelo cliente, bem como foram negociados a prioridade, a disponibilidade, e o custo relativos a cada requisito;
- foram definidos responsáveis por gerenciar os requisitos;
- foram definidos os modelos de sistema.

h) AC 10 - Obter as políticas, processos e procedimentos para as etapas de desenvolvimento do software, verificando se:

- foram definidos procedimentos para definição dos requisitos (estudo de viabilidade, elicitação e análise de requisitos, especificação de requisitos e validação de requisitos);
- os documentos seguem as regras da organização.

i) AC 11 - Obter a documentação de cursos ou treinamentos aplicados aos usuários envolvidos com o fim de educá-los e treiná-los no desenvolvimento do software, verificando se:

- foi realizado um levantamento das necessidades de ensino e treinamento para a implementação do software;
- o treinamento contemplou usuários, gestores de negócio, equipes de suporte e equipes de operação;
- os materiais de treinamento do usuário são atualizados e oferecem uma formação de acordo com o exigido ao pessoal afetado.

j) AC 12 - Obter uma descrição do processo de gerenciamento da qualidade do software, verificando se:

- contempla um plano de gestão de qualidade, que descreva o sistema de qualidade de projeto e como será implementado;
- contempla um plano de garantia de qualidade de software, definindo uma estratégia de *Software Quality Assurance* (SQA) de uma equipe de software.

k) AC 13 - Verificar se foi estabelecido processo prevendo o gerenciamento do projeto de desenvolvimento do software, verificando se:

- contempla uma estratégia de gestão de riscos específicos associados ao projeto, tendo realizado a avaliação, categorização e priorização dos riscos identificados e elaboração de planos para mitigar os riscos;
- prevê atividades de gestão dos requisitos, contemplando: garantir o entendimento dos requisitos, comprometimento das partes interessadas e gerenciar mudanças nos requisitos;
- define etapa de avaliação para identificar inconsistências entre produtos de trabalho, planos de projeto e requisitos;
- prevê que a fase de início do projeto deverá ser formalmente aprovada e comunicada a todas as partes interessadas, bem como prevê a aprovação das fases subsequentes com base na revisão e na aceitação dos resultados entregues da fase anterior e na aprovação de um estudo de caso atualizado na próxima revisão geral do programa;
- define responsabilidades, relacionamentos, autoridades e critérios de desempenho para os membros da equipe de projeto e especifica a base de aquisição e atribuição de funcionários e/ou prestadores de serviço competentes para o projeto;
- estabelece um plano integrado de projeto formalizado e aprovado (que abranja recursos de negócio e de sistemas de informação) para orientar a execução e o controle em todas as etapas do projeto;

- define o acompanhamento dos resultados do patrocinador do programa, patrocinador do projeto, comitê diretor, coordenador e gerente do projeto e os mecanismos pelos quais eles podem cumprir com essas responsabilidades (como relatórios e revisões de estágios do projeto);
- prevê atividade de avaliação do desempenho do projeto em comparação com critérios-chave (como escopo, cronograma, qualidade, custo e risco) e de revisões de progresso;
- define que, ao final do projeto, as partes interessadas apurem se o projeto gerou os resultados e benefícios planejados;
- prevê meios de monitorar e controlar o processo, parâmetros de planejamento do projeto, compromissos, riscos do projeto, gestão de dados e o envolvimento das partes interessadas.

4.5.3. OC 3 - Construção do Software e Adequação às Especificações

Garantir a entrega de um sistema de software em funcionamento, de acordo com sua especificação, e da documentação associada pronta para ser liberada para os usuários.

Os procedimentos a serem seguidos são:

a) AC 14 - Verificar se foi adquirida a infraestrutura de tecnologia requerida ao software e se foram definidos controles para sua operação, observando se:

- foram estabelecidos ambientes segregados de desenvolvimento e de teste para proporcionar eficiência e eficácia nos testes de viabilidade e integração dos componentes da infraestrutura;
- foram definidos controles prevendo segregação entre as atividades de desenvolvimento, teste e operação.

b) AC 15 - Verificar se está prevista a atualização da documentação do usuário caso haja mudança no projeto do software;

c) AC 16 - Obter os documentos de modelagem de análise do software, observando se:

- foram definidas as representações que mostram os requisitos de software quanto aos aspectos de informação, função e comportamento (modelos baseados em cenário, modelos de fluxo, modelos baseados em classe e modelos comportamentais);
- foi definida a modelagem de análise, contemplando o domínio da informação, domínio funcional, domínio comportamental e a interface do usuário;
- foram definidos os modelos de projeto e de implantação, completos e consistentes, contemplando a arquitetura do software, a interface do usuário e detalhes em nível de componentes.

d) AC 17 - Verificar se foram definidos os controles que deverão ser implementados no software, observando se:

- o projeto de controles de aplicação é suficiente aos requisitos dos *stakeholders*;
- está previsto incorporar no software checagens de validação com o objetivo de detectar qualquer corrupção de informações, por erros ou por ações deliberadas;
- está previsto implementar validações dos dados de saída para garantir que o processamento das informações armazenadas está correto e é apropriado às circunstâncias;
- está previsto o desenvolvimento e implementação de uma política para o uso de controles criptográficos para a proteção da informação;
- está prevista a implementação de validações de dados de entrada para garantir que são corretos e apropriados.

e) AC 18 - Verificar se foi definida uma especificação de teste para documentar as estratégias de testes de software e a abordagem da equipe de software para os testes, prevendo um plano que descreve uma estratégia global e

um procedimento que define passos específicos de teste e os testes que serão conduzidos (softwares convencionais ou orientados a objetos).

f) AC 19 - Obter o modelo de projeto de software e observar se contempla: elemento de projeto de dados, elemento do projeto arquitetural, elementos de projeto de interface, elementos de projeto no nível de componente e elementos de projeto no nível de implantação.

g) AC 20 - Verificar se foi definido o planejamento para a evolução do software, observando se:

- foi definido um processo formal para evolução de software, incluindo as atividades fundamentais de análise de mudanças, planejamento de *releases* e implementação de mudanças;
- o processo de evolução contempla a modificação da especificação do software, do projeto e dos requisitos (se necessário);
- foi definida a estratégia e o plano de manutenção de software.

h) AC 21, 22, 23 - Verificar se foi definido um planejamento para validação do software (o sistema realiza o que o usuário necessita), observando se:

- foi definida uma estratégia de validação;
- foi definido um processo de teste: de componente (ou unidade), de sistema e de aceitação;
- foi desenvolvido e documentado um plano de validação do software;
- foram estabelecidos procedimentos e critérios de validação;
- está previsto testar a conformidade do software às expectativas do usuário;
- foram definidas estratégias para a integração do software.

i) AC 24 - Verificar se foi definido um planejamento para verificação do software, observando se:

- define os produtos de trabalho para verificação;
- estabelece o ambiente, procedimentos e critérios de verificação;

- está previsto testar a conformidade do software à sua função;
- está prevista a validação do código-fonte do sistema (inspeção e análise estática);
- prevê o desenvolvimento, documentação e aprovação de um plano de teste do software em desenvolvimento.

j) AC 25 - Verificar se foram definidas técnicas para o desenvolvimento de sistemas críticos, observando se:

- está prevista a verificação da confiabilidade do processo de software, contemplando atividades de prevenção de defeitos, detecção de defeitos e tolerância a defeitos;
- está prevista a atividade de verificação da programação com base na especificação do sistema;
- deverá ser verificado se foi adotada uma linguagem de programação com tipos fortes de dados (como Java ou ADA) para o desenvolvimento;
- deverão ser utilizadas técnicas de programação segura no desenvolvimento do software;
- está prevista a utilização de técnicas que garantam que as informações sejam protegidas (deve ser usada uma abordagem para projeto e implementação de software baseada em ocultamento e encapsulamento de informações).

k) AC 26, 27, 28 - Obter o plano de testes do software e observar se:

- foi definida uma estratégia de testes adicionais para validação de sistemas críticos;
- foram projetados testes de unidade para cada componente de software;
- foram projetados e documentados os casos de teste para exercitar a lógica interna, interfaces, colaborações de componentes e os requisitos externos do software com o intuito de detectar erros.

l) AC 29 - Verificar se estão previstas atividades para verificar a conformidade do software ao projeto, observando se:

- foram construídos protótipos do sistema e da interface com o usuário, avaliados e validados pelo usuário;
- o código-fonte do sistema ou componente de software reusável está disponível aos desenvolvedores;
- novos requisitos ou requisitos modificados são revalidados pelos clientes antes que as mudanças sejam implementadas;
- foram estabelecidos critérios com relação aos quais o *design* possa ser avaliado;
- o projeto de interfaces foi realizado utilizando os critérios previamente estabelecidos;
- estão previstos testes unitários do sistema;
- estão previstas atividades de gerenciamento de interfaces, inclusive revisão das descrições de interfaces para assegurar completude;
- foi definida etapa de avaliação para confirmar se os componentes do produto estão prontos para serem integrados;
- foram estabelecidos procedimentos e critérios para integração do produto, bem como uma sequência de integração;
- foram seguidas boas práticas de programação (segurança);
- foi elaborada a documentação de suporte ao produto.

m) AC 30 - Verificar se foram estabelecidas atividades de gerenciamento de mudanças no software, observando se:

- há um processo de controle de modificações no software contemplando as seguintes atividades:
    - pedido formal de modificação;
    - análise (aceitação ou rejeição) do pedido de modificação;

    - atualização controlada do item de configuração de software que deve ser modificado.

- há um banco de dados ou repositório para manter os itens de configuração do software;
- foram estabelecidos procedimentos formais de gerenciamento de mudanças para lidar de modo padronizado com todas as solicitações de mudança na aplicação e parâmetros de sistema;
- está previsto que todas as mudanças feitas no escopo original do projeto (como custo, cronograma, escopo e qualidade) sejam devidamente revisadas, aprovadas e incorporadas ao plano de projeto integrado em alinhamento com a estrutura de governança de programa e projeto;
- as mudanças realizadas no software foram devidamente controladas, considerando os seguintes itens:
    - o software original foi mantido e as mudanças aplicadas numa cópia claramente identificada;
    - todas as mudanças foram completamente testadas e documentadas para que possam ser reaplicadas, se necessário, em atualizações futuras do software;
    - se requerido, as modificações foram testadas e validadas por um grupo de avaliação independente.

n) AC 31 - Está prevista a realização de auditoria técnica do software, verificando se:

- os produtos de software (como um item de software) refletem a documentação do projeto;
- os requisitos de revisão de aceitação e testes prescritos pela documentação estão adequados para a aceitação dos produtos de software;
- os dados de teste são compatíveis com a especificação;

- os produtos de software foram testados com sucesso e satisfizeram as suas especificações;
- os relatórios dos testes estão corretos e discrepâncias entre os resultados reais e esperadas foram resolvidas;
- a documentação do usuário está em conformidade com as normas especificadas;
- as atividades foram realizadas de acordo com os requisitos, planos e contrato;
- os custos e cronogramas aderem aos planos estabelecidos;
- os resultados da auditoria foram documentados.

o) AC 32 - Está prevista uma revisão pelos pares dos responsáveis pela construção do produto para identificar defeitos e outras mudanças que sejam necessárias.

4.5.4. OC 4 - Entrega do Software

Garantir que seja entregue um sistema de software documentado, funcionando corretamente em seu ambiente operacional e que seja validado pelo cliente para comprovar que atende aos seus requisitos.

Os procedimentos a serem seguidos são:

a) AC 33 - Verificar se foram previstas atividades de gerenciamento da implantação do software no ambiente de produção, observando se:

- foi definido responsável por acompanhar e validar a instalação e configuração da aplicação, incluindo os controles configuráveis;
- foi definido um plano de implantação para que, após a conclusão dos testes, seja controlada a transferência dos sistemas alterados para operação, de acordo com o plano, e com a aprovação dos *stakeholders*;
- está prevista a execução do sistema em paralelo com o sistema antigo durante um período para comparação do

comportamento/resultados e decisão de mudança definitiva do software;

- foram estabelecidos procedimentos em linha com o gerenciamento de mudanças organizacionais para garantir a realização da revisão pós-implantação, conforme definido no plano de implantação;
- foi verificado se o software foi instalado corretamente no ambiente alvo, se atendeu aos requisitos do cliente e se foi confirmada a operação correta do produto.

b) AC 34 - Verificar se foram definidos controles para o gerenciamento das configurações do software, observando se:

- estão previstas auditorias de configuração para manter a integridade dos *baselines*;
- foi definido um plano de gestão de configuração de software estabelecendo a estratégia do projeto para modificação;
- são necessárias aprovações formais para as solicitações de mudança dos itens de configuração;
- foram estabelecidas identificação e registro dos itens de configuração, componentes e produtos de trabalho relacionados a serem colocados sob gestão de configuração;
- foi estabelecido um sistema de gestão de configuração e de gestão de mudanças para controlar os produtos de trabalho;
- foram estabelecidos controles para garantir a completude funcional dos itens de software aos seus requisitos e integridade física dos itens de software (se o seu design e o código refletem uma descrição técnica atualizada);
- foi definida uma estratégia de gestão de configuração para o projeto;
- foi definido um plano para o gerenciamento de configurações, prevendo: itens de configuração, responsável pelos procedimentos de gerenciamento de configuração, políticas de gerenciamento de

configuração, ferramentas a serem usadas no gerenciamento de configurações e estrutura do banco de dados de configuração;

- foi implantado um processo de gestão de configuração de softwares que identifique, controle, audite e relate modificações realizadas no software, contemplando a identificação, controle de versão, controle de modificação, auditoria de configuração e preparação de relatórios;
- foi definido repositório para salvaguarda das informações sobre configurações com um nível adequado de integridade e segurança;
- foi definida uma ferramenta para automatizar o processo de construção de softwares (compilação e ligação de componentes de software num programa que executa determinada configuração definida);
- alterações ao *baseline* de configuração estão devidamente identificadas e registradas, avaliadas, aprovadas, incorporadas e verificadas;
- há identificação e registro dos pedidos de mudança, análise e avaliação das alterações e aprovação ou reprovação do pedido; e execução, verificação e liberação do item de software modificado;
- há uma trilha de auditoria para que, em cada alteração, o motivo da modificação e de autorização da modificação pode ser rastreado.

c) AC 35 - Obter os resultados da validação do software realizada, observando se:

- foram definidos os requisitos de teste, casos de teste e especificações de testes para analisar os resultados do teste com base nas exigências específicas para o uso específico pretendido;
- foram realizados os testes de integração e as correções necessárias aos erros identificados;

- o sistema foi avaliado considerando os seguintes critérios:
    - verificar a cobertura de teste dos requisitos do sistema;
    - verificar a conformidade com os resultados esperados;
    - avaliar a viabilidade da operação e manutenção.
- foram realizados testes dos controles da aplicação em conformidade com o plano de testes;
- foi estabelecido um plano de implementação e de retorno à configuração anterior, aprovado por todas as partes relevantes;
- foi estabelecido um ambiente de testes seguro que reflita o ambiente de operações planejado no que diz respeito à segurança, controles internos, práticas operacionais, exigências de qualidade e confidencialidade e cargas de trabalho;
- os dados de teste foram selecionados com aprovação do dono da informação;
- os testes de qualificação foram realizados em conformidade com os requisitos de qualificação para o item de software e os resultados foram documentados;
- foram identificados, registrados e resolvidos os erros e os problemas identificados durante os testes;
- as mudanças foram testadas de maneira independente e de acordo com o plano de testes definido antes da migração para o ambiente de produção (se aplicável);
- foram executados todos os testes listados no plano de testes, bem como testes de regressão necessários, e corrigidos erros significativos identificados no processo de testes;
- a promoção para a produção ocorreu somente após a avaliação e aprovação do usuário e da área de TI;
- foi validado se o produto de software satisfaz seu uso pretendido.

d) AC 36 - Obter os resultados da verificação do software realizada, observando se:

- foi realizada verificação independente por uma organização qualificada, se o projeto justificar o esforço;
- foram utilizados métodos de ensaio e padrões adequados;
- foram obtidos resultados conforme esperado;
- foi identificada a viabilidade do teste de qualificação do sistema;
- foi identificada a viabilidade da operação e manutenção do software;
- foram realizadas revisões técnicas no software;
- foram avaliados: o projeto, plano de integração, *design*, código, testes, os resultados dos testes e a documentação do usuário, observando a conformidade com os resultados esperados;
- foram avaliados os requisitos de verificação, observando se são consistentes, viáveis e testáveis;
- os requisitos do software foram apropriadamente alocados aos itens de hardware, software e operações manuais de acordo com critérios do projeto;
- os requisitos de software são consistentes, viáveis e testáveis;
- os requisitos de software relacionados com a segurança estão corretos, conforme demonstrado, pelos métodos de rigor;
- foi verificado o *design* do projeto, considerando:
    - se o projeto está correto e coerente com as exigências;
    - se o projeto implementa a segurança e outros requisitos críticos corretamente.
- foi verificado o código do software, observando:
    - se o código é rastreável relativamente ao *design* e requisitos, testável, correto, e em conformidade com os requisitos e padrões de codificação;

  - se o código selecionado pode ser derivado do projeto ou requisitos;
  - se o código implementa segurança, e outros requisitos críticos corretamente, conforme os métodos de rigor.
- foi verificada a integração do software considerando:
  - se os componentes de software e unidades de cada item de software estão completamente e corretamente integrados no item de software;
  - se os itens de hardware, software e manual de operações do sistema estão completamente e corretamente integrados no sistema;
  - se as tarefas de integração foram realizadas em conformidade com um plano de integração.
- foi verificada a documentação do software considerando:
  - se está adequada, completa e consistente;
  - se a preparação da documentação é oportuna;
- foram realizadas análises técnicas para avaliar os produtos de software e fornecer provas de que:
  - eles são completos.
  - estão em conformidade com seus padrões e especificações;
  - mudanças são devidamente implementadas e afetam apenas as áreas identificadas;
  - estão aderindo aos cronogramas aplicáveis;
  - estão prontos para a próxima atividade prevista;
  - o desenvolvimento, operação ou manutenção está sendo realizada de acordo com os planos, programas, normas e diretrizes do projeto.
- foram analisados os resultados da verificação e realizadas as adequações necessárias.

4.6. Pontuação das Atividades de Controle

Para pontuação das atividades de controle deve-se utilizar o seguinte critério para cada uma das atividades testadas:

- 0: Não existe controle;
- 1: Existe controle com deficiência importante e impacto relevante;
- 2: Existe controle com algumas deficiências importantes e médio impacto;
- 3: Existe controle com deficiência pouco importante e baixo impacto;
- 4: Existe controle com um nível aceitável.

O resultado final mostrará a avaliação específica de cada uma das atividades, conforme o critério supracitado, e também a avaliação de cada objetivo de controle a partir da média ponderada da pontuação das atividades contidas no objetivo de controle avaliado. A Tabela 4.1 apresenta um modelo de planilha para pontuação das atividades e objetivos de controle:

Tabela 4.1 – Planilha para pontuação das atividades e objetivos de controle (Fonte: O Autor)

Objetivos de Controle (OC)	Atividades de Controle (AC)	Pontuação por AC	Importância por OC (Peso: 1 – Baixo; 2 – Médio; 3 – Alto)
Garantir que sejam estabelecidos o escopo e viabilidade econômica do projeto	Atividade 1	**(0 a 4)**	**(1 a 3)**
	Atividade 2	**(0 a 4)**	**(1 a 3)**
	Atividade n	**(0 a 4)**	**(1 a 3)**
	Resultado (OC)	**(0 a 100%)**	
Garantir que sejam definidas a funcionalidade do software, seus requisitos e restrições sobre sua operação	Atividade 1	**(0 a 4)**	**(1 a 3)**
	Atividade 2	**(0 a 4)**	**(1 a 3)**
	Atividade n	**(0 a 4)**	**(1 a 3)**
	Resultado (OC)	**(0 a 100%)**	
Garantir a entrega de um sistema de software em funcionamento, de acordo com sua especificação, e da documentação associada pronta para ser liberada para os usuários	Atividade 1	**(0 a 4)**	**(1 a 3)**
	Atividade 2	**(0 a 4)**	**(1 a 3)**
	Atividade n	**(0 a 4)**	**(1 a 3)**
	Resultado (OC)	**(0 a 100%)**	
Garantir que seja entregue um sistema de software documentado, funcionando corretamente em seu ambiente operacional e que seja validado pelo cliente para comprovar que atende aos seus requisitos	Atividade 1	**(0 a 4)**	**(1 a 3)**
	Atividade 2	**(0 a 4)**	**(1 a 3)**
	Atividade n	**(0 a 4)**	**(1 a 3)**
	Resultado (OC)	**(0 a 100%)**	

Depois de avaliadas todas as atividades, a avaliação final e consolidada dos objetivos de controle pode ser gerada a partir do modelo a seguir (Tabela 4.2).

Tabela 4.2 – Planilha para avaliação final dos objetivos de controle

Objetivos de Controle (OC)	Pontuação	Avaliação	Qualificação	Importância por Objetivo de Controle (Peso: 1 – Baixo; 2 – Médio; 3 – Alto)
Garantir que sejam estabelecidos o escopo e viabilidade econômica do projeto	**(0 a 4)**	**(0 a 100%)**	**(Inaceitável, Deficiente, Regular, Bom e Excelente)**	**(1 a 3)**
Garantir que sejam definidas a funcionalidade do software, seus requisitos e restrições sobre sua operação	**(0 a 4)**	**(0 a 100%)**	**(Inaceitável, Deficiente, Regular, Bom e Excelente)**	**(1 a 3)**
Garantir a entrega de um sistema de software em funcionamento, de acordo com sua especificação, e da documentação associada pronta para ser liberada para os usuários	**(0 a 4)**	**(0 a 100%)**	**(Inaceitável, Deficiente, Regular, Bom e Excelente)**	**(1 a 3)**
Garantir que seja entregue um sistema de software documentado, funcionando corretamente em seu ambiente operacional e que seja validado pelo cliente para comprovar que atende aos seus requisitos	**(0 a 4)**	**(0 a 100%)**	**(Inaceitável, Deficiente, Regular, Bom e Excelente)**	**(1 a 3)**
Resultado Final		**(média da avaliação)**	**(Qualificação)**	
Resultado Global do Trabalho		**(média ponderada da avaliação)**	**(Qualificação)**	

Ao final do trabalho, consideram-se os seguintes valores para aceitação do resultado global do trabalho:

- < 40 : Inaceitável
- 40 ~ 60 : Deficiente
- 60 ~ 75 : Regular
- 75 ~ 90 : Bom
- 90 ~ 100 : Excelente

4.7. Acompanhamento do Grau de Avanço

Durante a realização da auditoria, deve-se medir e acompanhar a evolução do trabalho e reportar aos responsáveis, conforme modelo a seguir:

Tabela 4.3 – Acompanhamento de Grau de Avanço (Fonte: O Autor)

Objetivo de Controle	Atividades de Controle Identificadas	Atividades de Controle Revisadas	% Revisado
Garantir que sejam estabelecidos o escopo e viabilidade econômica do projeto			
Garantir que sejam definidas a funcionalidade do software, seus requisitos e restrições sobre sua operação			
Garantir a entrega de um sistema de software em funcionamento, de acordo com sua especificação, e da documentação associada pronta para ser liberada para os usuários			
Garantir que seja entregue um sistema de software documentado, funcionando corretamente em seu ambiente operacional e que seja validado pelo cliente para comprovar que atende aos seus requisitos			

4.8. Resultados

Como resultado do trabalho espera-se a elaboração dos seguintes documentos:

- Elaboração e apresentação do Relatório de Auditoria, indicando o resultado da avaliação, riscos identificados, planos de ação acordados/realizados, considerações dos responsáveis dos ambientes auditados e estratégia de acompanhamento após a auditoria;
- Matriz com a qualificação dos objetivos de controle;
- Planilha de acompanhamento finalizada.

A avaliação indicará a aderência do projeto aos padrões de referência. Caso o resultado indique baixa aderência às atividades de controle indicadas nesse método, o auditor deverá apresentar recomendações para adequação do projeto de modo a contemplar as atividades de controle não previstas ou realizadas de forma não satisfatória, o que contribuirá para mitigar o risco de falha no produto final (software).

Conclusão

Esta pesquisa apresentou uma proposta de método de auditoria para o acompanhamento de projetos de desenvolvimento de software baseados no Processo Unificado, a qual foi desenvolvida com base nos principais padrões de referência para processos de software, tendo sido as atividades de controle selecionadas do:

- Cobit (Governança de TI);
- *CMMI for Development*, MPS.BR, NBR ISO/IEC 9126 (Modelos de Qualidade de Software);
- NBR ISO/IEC 12207, NBR ISO/IEC 15504 (Processo de Desenvolvimento de Software e sua avaliação);
- NBR ISO/IEC 27002, ISO/IEC 15408 (Segurança).

Para validar a viabilidade do uso desse método de auditoria, as atividades que fazem parte do mesmo foram validadas junto a especialistas em auditoria de sistemas ou no processo de desenvolvimento de software por meio de uma pesquisa de campo, na qual foi possível verificar que as atividades foram bem avaliadas pelos especialistas, o que as qualificou a fazer parte do método proposto. Inclusive, mesmo havendo atividades consideradas menos importantes, todas as selecionadas para o método foram consideradas significativas pelos especialistas, o que indica que são pertinentes aos projetos de software e aplicáveis ao método proposto.

Sendo assim, o principal objetivo da pesquisa foi atingido, uma vez que foi definido um método de auditoria estruturado para direcionar o acompanhamento padronizado de atividades de controle em projetos de desenvolvimento de software baseados no processo unificado, o qual poderá ser utilizado de forma independente por equipes de auditores ou mesmo como linha base para uma abordagem de gerenciamento a ser realizado pela própria equipe de TI, com o intuito de reduzir falhas no produto entregue.

Como conclusões mais importantes deste trabalho, podem-se destacar:

- A viabilidade e importância de se implantar práticas de auditoria no processo de desenvolvimento de software a fim de produzir produtos de software mais aderentes às necessidades dos usuários;

- A definição do escopo e os limites do projeto como uma das atividades mais importantes do processo para melhor dimensionar e direcionar os esforços e possibilitar a identificação de riscos que causariam maior impacto ao projeto. Isto evidencia a relevância de se definir as funções e características que deverão ser entregues aos usuários finais, os dados de entrada e saída, as informações que serão apresentadas aos usuários como consequência do uso do software e o desempenho, as restrições, as interfaces e a confiabilidade que limitarão o sistema;

- A especificação das necessidades de informação, as quais culminam na definição de requisitos como uma atividade que conclui a identificação e seleção das informações mais críticas por meio da aprovação do cliente e dos *stakeholders*;

- Que apesar de haver padrões de referência ao desenvolvimento de software, nem sempre são aplicados na sua íntegra, porém a decisão de limitação do uso dos padrões de referência não garante o resultado satisfatório.

Podem-se destacar como principais contribuições deste trabalho:

- Uma proposta de um método de auditoria ao desenvolvimento de software baseado no processo unificado formado por atividades de controle, permitindo uma visão mais detalhada do processo de desenvolvimento de software aos profissionais de auditoria de sistemas, pois o conhecimento do processo por todos os envolvidos pode influenciar fortemente a produção de software mais aderente às necessidades dos clientes;

- Ressaltar a importância de se aplicar boas práticas de desenvolvimento no ciclo de vida de desenvolvimento de software;

- Proposição de ferramentas de apoio à execução de auditorias no processo de desenvolvimento de software.

Dessa forma, o trabalho realizado buscou a organização e o refinamento de atividades ligadas à Engenharia de Software e à Auditoria de Sistemas para se montar um método de auditoria para apoio ao desenvolvimento de software baseados no processo unificado. A proposição desse método tem o objetivo de ajudar equipes de gerenciamento da área de TI a obterem uma visão mais aproximada da situação dos projetos de software e equipes de desenvolvimento de software a produzir soluções e sistemas de software com a menor quantidade de falhas e mais aderentes às necessidades dos usuários.

Sugestões para Trabalhos Futuros

Esse trabalho abre espaço para realização de novos trabalhos na área de auditoria de sistemas, uma área ainda nova e que necessita de pesquisa e incentivo, com o intuito de fornecer maior suporte para o desenvolvimento de software de mais qualidade. Um método que formalize a utilização de um conjunto de atividades de controle pode ser de grande interesse para empresas que necessitam de maior acompanhamento nos seus produtos de software e estão dispostas a investir para obter melhores resultados. Não obstante, destaca-se que apesar da aplicação de técnicas e métodos de desenvolvimento de software não se pode garantir a prevenção de um software sem erro, portanto todas as ações e esforços têm o fim de mitigar esse risco.

Dentre os trabalhos futuros que podem dar continuidade à pesquisa aqui apresentada, sugere-se:

- Aplicar o método de auditoria proposto em um projeto de desenvolvimento de software, com a possibilidade de envolver o maior número possível das atividades distribuídas entre os quatro objetivos de controle, de acordo com o escopo do projeto;

- Agregar a este método padrões de segurança, com o objetivo de contribuir para o desenvolvimento de softwares mais seguros;
- Realizar um estudo para integrar no método a identificação, organização e utilização de um conjunto de métricas de desempenho para medir o progresso e a adequação do software aos requisitos prospectados;
- Estabelecer o resultado da aplicação do método em projetos reais de software e medir o grau de contribuição para o sucesso destes projetos.

Referências Bibliográficas

ABES - Associação Brasileira das Empresas de Software. ***Mercado Brasileiro de Software - Panorama e Tendências***, http://www.abes.org.br/UserFiles/Image/PDFs/Mercado_BR2010.pdf, acesso em 28 de Janeiro de 2012.

ABNT - Associação Brasileira de Normas Técnicas. ***NBR ISO/IEC 12207 - Tecnologia da informação - Processos de ciclo de vida de software*** *(correspondente à ISO/IEC 12207, 1995).* Rio de Janeiro : s.n., 2009.

—. ***NBR ISO/IEC 9126-1 - Engenharia de software - Qualidade de produto - Parte 1: Modelo de qualidade****,* 2003.

—. ***NBR ISO/IEC 15504 – Tecnologia de informação - Avaliação de processo.*** **Rio de Janeiro**, 2008.

—. ***NBR ISO/IEC 27002:2005 – Tecnologia da Informação – Técnicas de Segurança – Código de Prática para a Gestão de Segurança da Informação***. Rio de Janeiro, 2005.

ALBUQUERQUE, R. e RIBEIRO, B. ***Segurança no Desenvolvimento de Software****.* Rio de Janeiro: Campus, 2002.

ALMEIDA, A. C., BOFF, G., OLIVEIRA, J. L. ***A Framework for Modeling, Building and Maintaining Enterprise Information Systems Software***. *XXIII Brazilian Symposium on Software Engineering.* Fortaleza, 2009.

ANDRADE, J. M., ALBUQUERQUE, A. B., CAMPOS, F. B. e ROCHA, A. R. C. ***Consequências e Características de um Processo de Desenvolvimento de Software de Qualidade e Aspectos que o influenciam: uma avaliação de especialistas*****.** *III Simpósio Brasileiro de Qualidade de Software.* Brasília, 2004.

BEZERRA, C. ***A qualidade do processo de desenvolvimento de software a partir da gestão de projetos: um estudo de caso***. *III Simpósio Brasileiro de Qualidade de Software.* Brasília, 2004.

BOYNTON, W. C. e JOHNSON, R. N. ***Modern Auditing: Assurance Services And the Integrity of Financial Reporting****, 8th Edition.* John Wiley & Sons, Inc., 2006.

CAMPOS, F. B., ALBUQUERQUE, A. B., ANDRADE, J. M., SILVA FILHO, R. C. e ROCHA, A. R. C. ***Abordagem em Níveis para Avaliação e Melhoria de Processos de Software***. *V Simpósio Brasileiro de Qualidade de Software – SBQS.* Vila Velha, 2006.

CARVALHO, G. H. P., SILVA, F. Q. B. e FRANÇA, A. C. C. ***Management Function Deployment: um Método para o Alinhamento Estratégico da Melhoria dos Processos de Gerenciamento de Projetos de Software***. *IX Simpósio Brasileiro de Qualidade de Software.* Pará, 2010.

CERDEIRAL, C. e ROCHA, A. R. ***Uma Abordagem para Gerência e Avaliação de Projetos de Melhoria de Processos de Software do Ponto de Vista da Instituição de Consultoria***. *VIII Simpósio Brasileiro de Qualidade de Software.* Ouro Preto, 2009.

CHIN, G. ***Agile management: how to succeed in the face of changing project requirements****.* Amacon, 2004.

CÔRTES, L. M. e CHIOSSI, T. C. ***Modelos de Qualidade de Software****.* Campinas: Editora da Unicamp, 2001.

COSO - Comittee of Sponsoring Organizations of the Treadway Commission. ***Internal Control - Integrated Framework****.* 1994.

DANTAS, B., DÉHARBE, D., GALVÃO, S., MOREIRA, A. M. e MEDEIROS JÚNIOR, V. ***Proposta e Avaliação de uma Abordagem de Desenvolvimento de Software Fidedigno por Construção com o Método B****.* *SEMISH - Seminário Integrado de Software e Hardware.* 2008.

ENGHOLM JR**,** H. ***Engenharia de Software na Prática****.* São Paulo: Novatec, 2010.

ETXEBERRIA, L., SAGARDUI, G. e BELATEGIET, L. ***Quality aware Software Product.*** *Journal of the Brazilian Computer Society (JBCS).* Campinas, 2008. ISSN 0104-6500.

FERREIRA, A. B. H. ***Novo dicionário Aurelio versão 5.0 edição revista e atualizada****.* São Paulo: Positivo, 2006. Vol. 5.

GIL, A. C. ***Como elaborar projetos de pesquisa****.* São Paulo: Atlas, 2007.

GOMES, A., OLIVEIRA, K. e ROCHA, A. R. ***Avaliação de Processos de Software Baseada em Medições***. *XV Simpósio Brasileiro de Engenharia de Software.* Rio de Janeiro, 2001.

GOMES, A., MAFRA, S., OLIVEIRA, K. e ROCHA, A. R. ***Avaliação de Processos de Software na Estação Taba***. *XV Simpósio Brasileiro de Engenharia de Software.* Rio de Janeiro, 2001.

GUEDES, F. C. e PIETROBON, C. A. M. ***Glifo para Verificação da Utilização de Processos de Requisitos por meio da Rastreabilidade como apoio à Melhoria de Qualidade de Software***. *VIII Simpósio Brasileiro de Qualidade de Software.* Ouro Preto, 2009.

GUSMÃO, C. M. G. e MOURA, H. P. ***Gerência de Risco em Processos de Qualidade de Software: uma Análise Comparativa***. *III Simpósio Brasileiro de Qualidade de Software.* Brasília, 2004.

IDETI. ***Congresso Latinoamericano de Auditoria de TI, Segurança da Informação e Governança***. São Paulo, 2011.

IFAC - *International Federation of Accountants*. ***International Federation of Accountants****.* http://ifac.org, acesso em 20 de outubro de 2011.

IMONIANA, J. O. ***Auditoria de Sistemas de Informação****.* Atlas, 2005.

ITGI - IT *Governance Institute*. ***CobiT 4.1.*** Unites States of America: ITGI, 2007.

—. ***CobiT and Application Controls: A Management Guide****.* Unites States of America: ITGI, 2009.

ISO/IEC/JTC 1 *Information Technology*. ***ISO/IEC 15408-1:2009 Information technology -- Security techniques -- Evaluation criteria for IT security-- Part 1: Introduction and general model****.* 2009.

JACOBSON, I., BOOCH, G. e RUMBAUGH, J. ***The Unified Software Development Process****.* Addison-Wesley Professional, 1999.

KOSCIANSKI, A. e SOARES, M. ***Qualidade de Software****.* São Paulo: Novatec, 2007.

KRUSZELNICKI, K. S. ***Software sucks***. *ABC Science,* http://www.abc.net.au/science/articles/2003/02/19/726391.htm?site=science/greatmomentsinscience, acesso em 23 de Maio de 2011.

MARTINEZ, M. R. M. e SALVIANO, C. F. ***Método para Estabelecimento de Referências em Ciclos de Melhoria de Processo*****.** *X SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE.* Curitiba, 2011.

MENESES, J. B. e MOURA, H. P. ***Um Processo de Avaliação de Progresso para Projetos de Software***. *XV Simpósio Brasileiro de Engenharia de Software.* Rio de Janeiro, 2001.

MOREIRA, A., COTA, E., RIBEIRO, L., GASPARY, L., CARRO, L., RITT, M. e WEBER, T. ***Em Direção a um Modelo para Desenvolvimento de Sistemas Computacionais de Qualidade para Aplicações Onivalentes***. *SEMISH - Seminário Integrado de Software e Hardware.* 2007, p. 2265.

MOREIRA, R. T, LIMA, G. N, MACHADO, B. B., MARINHO, W. T., VASCONCELOS, A. e Rouiller, A. C. ***Uma Abordagem para Melhoria do Processo de Software baseada em Medição***. *VIII Simpósio Brasileiro de Qualidade de Software.* Ouro Preto, 2009.

SOFTEX. ***MPS.BR – Melhorias de Processos do Software Brasileiro***. *www.softex.br*, acesso em 24 de janeiro de 2012.

NAVAS, J. ***Um Método para a Melhoria do Processo de Desenvolvimento de Software Aplicando Conceitos de CMM, SPICE e da Norma ISO 12207***. Piracicaba, 2006.

OLIVEIRA, K. R. e BELCHIOR, A. D. ***AdeQuaS: Ferramenta Fuzzy para Avaliação da Qualidade de Software***. *I Simpósio Brasileiro de Qualidade de Software.* Gramado, 2002.

PEREIRA, P. C. R., GONÇALVES, F. M. G. S., PIRES, C. G. e BELCHIOR, A. D. ***Um Processo para o Gerenciamento de Riscos em Projetos de Software***. *V Simpósio Brasileiro de Qualidade de Software – SBQS.* Vila Velha, 2006.

PRESSMAN, R. S. ***Engenharia de Software****.* São Paulo: McGraw-Hill, 2011. Vol. Sétima Edição.

SABBAG, P. Y. ***Gerenciamento de Projetos e Empreendedorismo****.* São Paulo: Saraiva, 2009.

SANTOS, L. B. e PRETZ, E. ***Framework para Especialização de Modelos de Qualidade de Produtos de Software***. *IX Simpósio Brasileiro de Qualidade de Software.* Belém, 2010.

SBC - Sociedade Brasileira de Computação. ***Grandes Desafios da Pesquisa em Computação no Brasil: 2006–2016***. http://www.sbc.org.br, Acesso em 14 de janeiro de 2012.

SCHMIDT, P., SANTOS, J. L. e ARIMA, C. H. ***Fundamentos de auditoria de sistemas****.* São Paulo: Atlas, 2006.

SEI - Software Engineering Institute. ***CMMI for Development, Version 1.3****.* http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm, acessado em 22 de Maio de 2011.

SOMMERVILLE, I. ***Engenharia de Software****.* 9ª Edição. São Paulo: Pearson Addison Wesley, 2011.

TEMPO REAL EVENTOS. ***Seminário Engenharia de Software 2010***, 2010.

THE STANDISH GROUP. ***CHAOS Manifesto 2011***. https://secure.standishgroup.com/reports/reports.php#reports, acesso em 28 de Janeiro de 2012.

THOMSETT, R. ***Radical Project Management****.* Prentice Hall, 2002.

ZANETTI, D., MONTONI, M. e ROCHA, A. R. ***Benchmarking em Iniciativas de Melhorias em Processos de Software***. *VIII Simpósio Brasileiro de Qualidade de Software.* Ouro Preto, 2009.

Apêndice A - Mapeamento dos padrões de referência para o processo de software

Fase	Capítulo	Macro Atividade	Atividade
Sommerville			
Concepção	2. Requisitos	Requisitos de Software	Especificar interfaces.
		Processos de Engenharia de Requisitos	Realizar estudos de viabilidade.
		Especificação de sistemas críticos	Definir especificação dirigida a riscos.
Elaboração	1. Visão Geral	Processos de software	Verificar se foi definido um processo para definição dos requisitos, contemplando: Estudo de viabilidade, Elicitação e análise de requisitos, Especificação de requisitos e Validação de Requisitos.
	2. Requisitos	Requisitos de Software	Definir os requisitos funcionais e não funcionais.
			Definir os requisitos de usuário.
			Definir os requisitos de sistema.
			Documentar os requisitos de software.
		Processos de Engenharia de Requisitos	Elicitar e analisar os requisitos.
			Validar os requisitos.
			Gerenciar os requisitos.
		Modelos do sistema	Definir modelos de sistema.
		Especificação de sistemas críticos	Definir especificação de segurança, proteção, confiabilidade de software.
		Especificação formal	Definir especificação formal (processo de software, interface de subsistema, comportamento).
	3. Projeto	Projeto de arquitetura	Definir o projeto de arquitetura (decisões de projeto de arquitetura, organização do sistema, decomposição modular, controle, modelo de arquitetura). Ver viabilidade de: projeto orientado a objetos, projeto de software de tempo real e projeto de interface com o usuário.
	4. Desenvolvimento	Desenvolvimento rápido de software - Métodos ágeis	Envolver o cliente (processo de desenvolvimento, especificação dos requisitos).
		Desenvolvimento rápido de software - *Extreme programming*	Registrar os requisitos.
			Realizar um projeto.
			Definir responsáveis por realizar a programação.
			Definir responsável (cliente) para apoiar a equipe de XP e por trazer os requisitos de sistema à equipe para implementação.
		Desenvolvimento rápido de software - Prototipação de software	Estabelecer os objetivos do protótipo.
			Definir responsabilidade do protótipo.
			Desenvolver protótipo.
			Avaliar protótipo.
		Engenharia de software baseada em componentes	Registrar os requisitos do usuário. Avaliar a viabilidade dos requisitos x componentes disponíveis. Realizar busca de componente adicional. Realizar o desenvolvimento (integração de componentes com a infraestrutura de modelo de componente e, muitas vezes, desenvolvimento de "código colante" para reconciliar as interfaces de componentes incompatíveis).

Fase	Capítulo	Macro Atividade	Atividade
		Desenvolvimento de sistemas críticos	Verificar se o processo de software é confiável, contemplando: - atividades de prevenção de defeitos - o processo de projeto e de implementação do sistema deve usar abordagens de desenvolvimento que ajudem a evitar erros de programação e, assim, minimizar o número de defeitos de um programa; - detecção de defeitos - os processos de verificação e de validação são projetados para descobrir e remover defeitos de um programa antes que este seja implantado para uso operacional; - tolerância a defeitos - o sistema é projetado de forma que os defeitos ou o comportamento inesperado do sistema, durante a execução, sejam detectados e gerenciados de modo que a falha do sistema não ocorra (Detecção de Defeitos - Preventiva ou *backward*), Avaliação de Danos, Recuperação de Defeitos e Reparação de Defeitos) e implementação de arquiteturas tolerantes a defeitos (Programação em N-versões e Blocos de Recuperação).
			Verificar se há especificação formal que defina o sistema a ser implementado.
		Evolução de software	Verificar se o processo de evolução contempla a modificação da especificação do software, do projeto e dos requisitos (se necessário).
			Verificar se novos requisitos ou requisitos modificados são revalidados pelos clientes antes que as mudanças sejam implementadas.
	6. Gerenciamento	Gerenciamento de Configurações	Verificar se foi definido o plano para o gerenciamento de configurações, prevendo: itens de configuração, responsável pelos procedimentos de gerenciamento de configuração, políticas de gerenciamento de configuração, ferramentas a serem usadas no gerenciamento de configurações e estrutura do banco de dados de configuração.
			Verificar se foram definidos procedimentos de gerenciamento de mudanças, prevendo o fluxo para aprovação das mudanças solicitadas e a rastreabilidade de quais componentes do sistema foram alterados.
			Verificar se foram definidos procedimentos para o gerenciamento de versões e releases.
Construção	4. Desenvolvimento	Desenvolvimento rápido de software - *Extreme programming*	Definir o plano de testes às funcionalidades.
			Realizar os testes unitários do sistema.
		Reuso de software	Verificar se o código-fonte do sistema ou componente de software reusável está disponível
			Verificar se há apoio de ferramenta (CASE) ao desenvolvimento com reuso .
			Criar e realizar manutenção em uma biblioteca de componentes.
			Procurar, compreender e adaptar os componentes reusáveis.
		Desenvolvimento de sistemas críticos	Verificar se é realizada atividade de verificação da programação com base na especificação do sistema.
			Verificar se foi adotada linguagem de programação com tipos fortes de dados (como Java ou ADA) para o desenvolvimento.

Fase	Capítulo	Macro Atividade	Atividade
		Desenvolvimento de sistemas críticos	Verificar se são utilizadas técnicas de programação segura no desenvolvimento do sistema.
			Verificar se são utilizadas técnicas de que garantam que as informações sejam protegidas - deve ser usada uma abordagem para projeto e implementação de software baseada em ocultamento e encapsulamento de informações.
	1. Visão Geral	Processos de software	Verificar se foi definido o processo de teste, contemplando testes de componente (ou unidade), teste de sistema e teste de aceitação.
	5. Verificação e Validação	Verificação e Validação	Verificar se foi definido o planejamento para verificação (o sistema atende à sua especificação) e validação (o sistema realiza o que o usuário necessita) do sistema em desenvolvimento, contemplando sua conformidade à função do software, às expectativas do usuário e ao ambiente de mercado.
		Verificação e Validação	Verificar se foi definido o plano de teste do software em desenvolvimento, contemplando: cronograma de teste, procedimentos para gerenciamento do processo de teste, o hardware e os requisitos de software, além de quaisquer outros problemas de teste que possam surgir.
		Verificação e Validação	Verificar se foi definido o planejamento para validar o código-fonte do sistema (inspeção, análise estática).
		Teste de Software	Verificar se o processo de teste contempla: - testes de componente (ou unidade): teste dos componentes individuais do sistema, inclusive teste de interface; - teste de sistema: integração de dois ou mais componentes que implementam funções ou características do sistema e depois o teste desse sistema integrado, inclusive teste de integração, teste de release e teste de desempenho; - teste de aceitação: o sistema é testado com os dados fornecidos pelo cliente do sistema, em vez de dados simulados de teste, antes que o sistema seja aceito para o uso operacional.
		Validação de sistemas críticos	Verificar se foi definida uma estratégia de testes adicionais para validação de sistemas críticos, como validação de confiabilidade (métricas para requisitos de confiabilidade), garantia de segurança (nível de confiabilidade do sistema) e avaliação de proteção (requisitos que demonstram o que não deve ocorrer no sistema).
	4. Desenvolvimento	Evolução de software	Verificar se há um processo formal para evolução de software que inclua as atividades fundamentais de análise de mudanças, planejamento de releases e implementação de mudanças.
	6. Gerenciamento	Gerenciamento de Configurações	Verificar se há ferramenta para automatizar o processo de construção de sistemas (compilação e ligação de componentes de software num programa que executa determinada configuração definida).
Transição	5. Verificação e Validação	Verificação e Validação	Verificar se foi definido o planejamento para verificação (o sistema atende à sua especificação) e validação (o sistema realiza o que o usuário necessita) do sistema em desenvolvimento, contemplando sua conformidade à função do software, às expectativas do usuário e ao ambiente de mercado.

Fase	Capítulo	Macro Atividade	Atividade
Pressman			
Concepção	5. Prática: Uma visão Genérica	5.2. Práticas de Comunicação	Verificar se foi definido e validado junto ao cliente o escopo do projeto, considerando: - necessidade do negócio e valores do negócio; - características/necessidades dos usuários finais; - Saídas visíveis ao usuário requeridas; - Restrições do negócio.
		5.3. Práticas de Planejamento	Verificar se foram definidas as características técnicas que forma a infraestrutura do software e se foram agrupadas de acordo com a prioridade do cliente.
			Verificar se foram definidas as tarefas de trabalho, responsabilidade por cada tarefa e produtos de trabalho a serem produzidos.
Elaboração	5. Prática: Uma visão Genérica	5.2. Práticas de Comunicação	Verificar se foram definidas e validadas pelo cliente: - Cenários de uso visíveis ao cliente usando o formato padrão; - Saídas e entradas resultantes; - Características, funções e comportamentos importantes do software; - Riscos de negócios definidos pelo cliente.
		5.3. Práticas de Planejamento	Verificar se foi realizada a reavaliação do escopo do projeto.
			Verificar se foi realizada a avaliação dos riscos do projeto.
			Verificar se foi definido o plano do projeto, contemplando: número de incrementos de software projetados, cronograma geral do projeto e datas de entregas projetadas para cada incremento.
		5.4. Práticas de Modelagem	Verificar se a modelagem de análise foi concluída, contemplando o domínio da informação, domínio funcional, domínio comportamental e a interface do usuário e se todos os modelos estão completos e consistentes.
			Verificar se foram concluídos os modelos de projeto, contemplando a arquitetura do software, a interface do usuário e detalhes em nível de componentes e se todos os modelos estão completos e consistentes.
	7. Engenharia de Requisitos		Verificar se foram definidos e validados pelo cliente os requisitos do software, bem como foram negociados a prioridade, a disponibilidade, e o custo relativos a cada requisito.
	8. Modelagem de Análise		Verificar se foram definidas as representações que mostram os requisitos de software quanto à informação, função e comportamento (modelos baseados em cenário, modelos de fluxo, modelos baseados em classe e modelos comportamentais).
	12. Projeto de Interface com o Usuário		Verificar se foi definida a interface do usuário, considerando: - identificação dos usuários finais do sistema; - análise de tarefas e ações do usuário; - análise ambiental, identificando as estruturas físicas e sociais nas quais a interface deve operar.
			Verificar se foi construído o protótipo operacional da interface com o usuário.
			Verificar se o usuário avaliou e validou o protótipo.

Fase	Capítulo	Macro Atividade	Atividade
	Parte 4 - Gestão de Projetos de Software	21. Conceitos de Gestão de Projetos 22. Métricas de Processo e Projeto 23. Estimativa de Projetos de Software 24. Cronogramação de Projeto de Software 25. Gestão de Risco	Verificar se foi definido um plano para o projeto e se contempla: - medidas e métricas de processo e projeto (medidas quantitativas que permitem aos engenheiros de software ter ideia da eficácia do processo de software do projeto); - estimativa (delineamento das tarefas a serem realizadas, as funções a serem implementadas, e o custo, o esforço e o tempo envolvidos para cada uma delas); - análise de risco (plano de atenuação, monitoração e gestão de riscos); - cronograma do projeto; - atividades de acompanhamento; - controles.
	Parte 4 - Gestão de Projetos de Software	26. Gestão de Qualidade	Verificar se foi criado um plano de garantia de qualidade de software para definir a estratégia de *Software Quality Assurance* - SQA de uma equipe de software.
	Parte 4 - Gestão de Projetos de Software	27. Gestão de Modificações	Verificar se foi criado um plano de gestão de configuração de software definindo a estratégia do projeto para modificação.
Construção	9. Engenharia de Projeto	9.6 Resumo	Verificar se o projeto desenvolvido abrange: - arquitetura do software; - definição de subsistemas e estabelecimento de mecanismos de comunicação entre subsistemas; - identificação e descrição detalhada de cada um dos componentes; - projeto de interfaces externas, internas e com o usuário.
	9. Engenharia de Projeto		Verificar se o modelo de projeto inclui os seguintes elementos: - Elemento de projeto de dados (cria um modelo de dados e/ou informação que é representado no nível mais alto de abstração - a visão dos dados do cliente/usuário). - Elemento do projeto arquitetural (usa informação derivada do domínio de aplicação, o modelo de análise, catálogos de padrões e estilos disponíveis para derivar uma representação estrutural completa do software, seus subsistemas e componentes); - Elementos de projeto de interface (modelam as interfaces externas e internas e a interface com o usuário); - Elementos de projeto no nível de componente (definem cada um dos módulos - componentes - que compõem a arquitetura); - Elementos de projeto no nível de implantação (alocam a arquitetura, seus componentes e interfaces à configuração física que vai alojar o software).
	10. Projeto Arquitetural		Verificar se o projeto arquitetural contempla: - sistema representado no contexto (o projetista deve definir as entidades externas com as quais o software interage e a natureza da interação); - identificação de um conjunto de abstrações de mais alto nível chamado de arquétipos que representam os elementos-chave do comportamento ou função do sistema; - identificação e representação dos componentes no contexto de uma arquitetura que lhes dá suporte; - desenvolvimento de instanciações específicas da arquitetura para provar o projeto em um contexto do mundo real.

Fase	Capítulo	Macro Atividade	Atividade
	11. Projeto no Nível de Componentes		Verificar se foi desenvolvido o projeto no nível de componentes a partir de notações no contexto da engenharia de software orientada a objetos ou convencional (programação estruturada).
	5. Prática: Uma visão Genérica		Verificar se foram projetados testes de unidade para cada componente de software.
			Verificar se foi desenvolvida uma estratégia de integração.
	13. Estratégias de Teste de Software		Verificar se foi desenvolvida uma especificação de Teste para documentar a abordagem da equipe de software para o teste, definindo um plano que descreve uma estratégia global e um procedimento que define passos específicos de teste e os testes que serão conduzidos.
		13.1 Uma Abordagem Estratégica ao Teste de Software	Para softwares convencionais, verificar se a especificação de Teste de Software contempla os como: - Teste de unidade (concentra-se em cada unidade, por exemplo, componente); - Teste de integração (o foco fica no projeto e na construção da arquitetura do software); - Teste de validação (os requisitos estabelecidos como parte da análise dos requisitos do software são validados com o software que acabou de ser construído); - Teste de sistema (o software e os outros elementos do sistema são testados como um todo).
		13.4 Estratégias de Teste para Software Orientado a Objetos	Para softwares orientados a objetos, verificar se a especificação de Teste de Software contempla os itens: - Teste de unidade no contexto OO (teste guiado pelas operações encapsuladas na classe e pelo estado de comportamento da classe); - Teste de integração no contexto OO (teste baseado no caminho de execução - thread-based testing, teste baseado no uso - use-based testing e teste agregado).
	14. Técnicas de Teste de Software		Verificar se foram projetados e documentados os casos de teste para exercitar a lógica interna, interfaces, colaborações de componentes e os requisitos externos do software com o intuito de detectar erros no software.
			Verificar se os projetos de casos de testes contemplam testes como: - teste caixa-branca (enfoca a estrutura de controle do programa); - teste caixa-preta (projetados para validar os requisitos funcionais sem se prender ao funcionamento interno de um programa); - testes para sistemas orientados a objetos (revisão do modelo de análise e de projeto e testes para classes, utilizando os métodos: teste baseado em erro, teste aleatório e teste de partição); - testes de integração.
	Parte 4 - Gestão de Projetos de Software	27. Gestão de Modificações	Verificar se foi implantado um processo de gestão de configuração de softwares e se este processo identifica, controla, audita e relata modificações realizadas no software.
			Verificar se há um banco de dados ou repositório para manter os itens de configuração do software.
			Verificar se o processo de gestão de configurações contempla a identificação, controle de versão, controle de modificação, auditoria de configuração e preparação de relatórios.

Fase	Capítulo	Macro Atividade	Atividade
			Verificar se o processo de controle de modificações no software contempla as seguintes atividades: - pedido formal de modificação; - análise (aceitação ou rejeição) do pedido de modificação; - atualização controlada do item de configuração de software que deve ser modificado.
			Verificar se são realizadas auditorias de configuração de software, prevendo as seguintes atividades: - validar se as modificações especificadas na ordem de modificação de engenharia foram realizadas em conformidade e se foram incorporadas mudanças adicionais; - validar se foi realizada uma revisão técnica formal para avaliar a correção técnica aplicada; - verificar se o processo de software foi seguido e as normas de engenharia de software foram adequadamente aplicadas; - verificar se a modificação foi destacada no item de configuração de software, se a data de modificação e o autor foram especificados e se os atributos do objeto de configuração refletem a modificação; - verificar se os procedimentos de gestão de configuração de software para anotar a modificação, registrá-la e relatá-la foram seguidos; - verificar se todos os itens de configuração relacionados foram adequadamente atualizados.
Transição	5. Prática: Uma visão Genérica	5.4. Práticas de Modelagem	Verificar se foi desenvolvido um modelo de implantação e se está completo e consistente.
			Verificar se foi desenvolvida uma estratégia de validação.
			Verificar o resultado dos testes de integração e validação e se foram realizadas as correções necessárias aos erros identificados.
			Verificar se foram realizados os testes de aceitação junto ao cliente, se foram aprovados e se foram realizadas as correções necessárias aos erros identificados.
	13. Estratégias de Teste de Software	13.5 Teste de Validação	Verificar se todos os testes de validação foram realizados e validados pelos clientes antes da implementação do software.
Cobit			
Concepção	AI1	Identificar Soluções Automatizadas	Verificar se foram identificados os principais *stakeholders* e atribuídas funções relacionadas com a aplicação de controle e responsabilidades. (AI1.1)
	AI2	Identificar Soluções Automatizadas	Identificar e documentar os objetivos de controle relevantes. (AI1.1)
	AI3	Identificar Soluções Automatizadas	• Definição dos requisitos técnicos e de negócio.
	AI4	Identificar Soluções Automatizadas	• Realização de estudos de viabilidade conforme definido nos padrões de desenvolvimento.
	AI5	Identificar Soluções Automatizadas	• Aprovação (ou rejeição) de requisitos e resultados de estudos de viabilidade.
	AI2	Adquirir e Manter Software Aplicativo	Realizar uma avaliação dos riscos relacionados com a informação e os riscos de processamento das informações relacionadas. (AI1.2)
	AI5	Adquirir Recursos de TI	Para software de terceiros, incluir requisitos de controle na RFP. (AI5.2, 5.3, 5.4)

Fase	Capítulo	Macro Atividade	Atividade
	P10	Gerenciar Projetos	Obter comprometimento e participação das partes interessadas afetadas na definição e na execução do projeto dentro do contexto do programa de investimento geral de TI.
			Definir responsabilidades, relacionamentos, autoridades e critérios de desempenho para os membros da equipe de projeto e especificar a base de aquisição e atribuição de funcionários e/ou prestadores de serviço competentes para o projeto.
Elaboração	P08	Gerenciar a Qualidade	Definição de práticas e padrões de qualidade.
	P10	Gerenciar Projetos	Estabelecer e manter uma estrutura de gestão de programas de projetos (relacionados ao portfólio de programas de investimentos em TI) e projetos que defina o escopo e a abrangência dos projetos gerenciados, bem como os métodos a serem adotados e aplicados a cada projeto iniciado.
			Definir e documentar a natureza e o escopo do projeto, visando confirmar e desenvolver um entendimento comum do escopo do projeto com as partes interessadas e quanto ao relacionamento com outros projetos de um programa de investimento em TI. A definição deve ser formalmente aprovada pelo patrocinador do programa e pelo patrocinador do projeto antes de seu início.
			Eliminar ou minimizar riscos específicos associados a cada projeto através de um processo sistemático de planejamento, identificação, análise, resposta, monitoramento e controle de áreas ou eventos com potencial para causar mudanças indesejadas. Os riscos identificados pelo processo de gestão de projeto e os resultados esperados do projeto devem ser estabelecidos e centralmente registrados.
			Preparar um plano de gestão de qualidade que descreva o sistema de qualidade de projeto e como será implementado. O plano deve ser formalmente revisado e aceito por todas as partes envolvidas e então incorporado ao plano integrado de projeto.
	AI2	Adquirir e Manter Software Aplicativo	Tradução dos requisitos de negócio nas especificações de projeto.
Construção	P10	Gerenciar Projetos	A estrutura de governança de projeto deve incluir os papéis, as responsabilidades e o acompanhamento dos resultados do patrocinador do programa, patrocinador do projeto, comitê diretor, coordenador e gerente do projeto e os mecanismos pelos quais eles podem cumprir com essas responsabilidades (como relatórios e revisões de estágios do projeto).
			Assegurar que a fase de início do projeto seja formalmente aprovada e comunicada a todas as partes interessadas. A aprovação da fase de início deve ser baseada nas decisões da governança do programa. A aprovação das fases subsequentes deve ser baseada na revisão e na aceitação dos resultados entregues da fase anterior e na aprovação de um estudo de caso atualizado na próxima revisão geral do programa. No caso de uma sobreposição de fases, deve ser estabelecido um ponto de aprovação pelos patrocinadores do programa e do projeto para autorizar a continuidade.

Fase	Capítulo	Macro Atividade	Atividade
			Estabelecer um plano integrado de projeto formalizado e aprovado (que abranja recursos de negócio e de sistemas de informação) para orientar a execução e o controle em todas as etapas do projeto.
			Avaliar o desempenho do projeto em comparação com critérios-chave (como escopo, cronograma, qualidade, custo e risco). Identificar qualquer desvio do plano. Avaliar o impacto dos desvios sobre o projeto e o programa. Reportar os resultados às principais partes interessadas. Recomendar, implementar e monitorar ações corretivas quando necessárias, em alinhamento com a estrutura de governança de projeto e programa.
	AI2	Adquirir e Manter Software Aplicativo	Adesão aos padrões de desenvolvimento em todas as modificações.
			Segregação entre as atividades de desenvolvimento, teste e operação.
			Revisar detalhadamente o projeto de especificação da aplicação (incluindo controles de aplicação automática). (AI2.1, 2.3)
			Avaliar e concluir sobre a suficiência do projeto de controles de aplicação (AI2.1, 2.3, 2.4, 2.5).
			Assegurar que as funcionalidades automatizadas sejam desenvolvidas em conformidade com as especificações de projeto, padrões de desenvolvimento e documentação e requisitos de qualidade e de autorização (AI2.7).
	AI3	Adquirir e Manter Infraestrutura de Tecnologia	Estabelecer um ambiente de desenvolvimento e de teste para proporcionar eficiência e eficácia nos testes de viabilidade e integração dos componentes da infraestrutura.
	AI4	Habilitar operação e uso	Comunicação e treinamento de usuários, gestores de negócio, equipes de suporte e equipes de operação.
	DS13	Gerenciar as Operações	Verificar se foram definidos procedimentos padronizados para a operação de TI, abrangendo: - mudança de turnos (passagem formal das atividades, atualização de informações, problemas operacionais, procedimentos de escalação e relatórios das responsabilidades atuais para assegurar o nível de serviço acordado e a continuidade das operações); - monitorar a infraestrutura de TI e eventos relacionados; - assegurar a manutenção da infraestrutura em tempo hábil para reduzir a frequência e o impacto de falhas ou degradação de desempenho.
	AI7	Instalar e Homologar Soluções e Mudanças	Desenvolver, documentar e aprovar o plano de teste (AI7.2).
			Realizar testes dos controles da aplicação em conformidade com o plano de teste (AI7.6).
			Identificar, registrar, priorizar e resolver os erros e os problemas identificados durante os testes (AI7.6).
			Documentar e interpretar os resultados finais do teste, incluindo questões não resolvidas (AI7.7).
			Aprovar os resultados dos testes e determinar a execução das atividades de controle do aplicativo (AI7.7).

Fase	Capítulo	Macro Atividade	Atividade
			Estabelecer um plano de implementação e de retorno à configuração anterior. Obter aprovação de todas as partes relevantes (AI7.3).
			Estabelecer um ambiente de testes seguro que reflita o ambiente de operações planejado no que diz respeito a segurança, controles internos, práticas operacionais, exigências de qualidade e confidencialidade e cargas de trabalho (AI7.4).
			Assegurar que as mudanças sejam testadas de maneira independente e de acordo com o plano de testes definido antes da migração para o ambiente de produção (AI7.6).
			Assegurar que o gerenciamento do departamento usuário e da área de TI avalie o resultado do processo de testes como determinado no plano de testes. Corrigir erros significativos identificados no processo de testes, executar todos os testes listados no plano de testes, bem como qualquer teste de regressão necessário. Após a avaliação, aprovar a promoção para a produção (AI7.7).
			Após a conclusão dos testes, controlar a transferência dos sistemas alterados para operação, de acordo com o plano de implementação. Obter a aprovação das partes interessadas, como usuários, proprietário do sistema e gerência operacional. Quando apropriado, executar o sistema em paralelo com o sistema antigo durante um período e comparar comportamento/resultados (AI7.8).
			Estabelecer procedimentos em linha com o gerenciamento de mudanças organizacionais para garantir a realização da revisão pós-implementação, conforme definido no plano de implementação (AI7.9).
	AI2	Adquirir e Manter Software Aplicativo	Desenvolver a estratégia e o plano de manutenção de software aplicativo.
	AI6	Gerenciar Mudanças	Estabelecer procedimentos formais de gerenciamento de mudanças para lidar de modo padronizado com todas as solicitações de mudança em aplicações, procedimentos, processos, parâmetros de sistema, parâmetros de serviço e plataformas subjacentes (inclusive solicitações de manutenção e reparo).
			Atualizar a documentação os procedimentos do sistema e de usuários sempre que forem implementadas mudanças no sistema.
Transição	P10	Gerenciar Projetos	Estabelecer um sistema de controle de mudança para cada projeto, de forma que todas as mudanças feitas no escopo original do projeto (como custo, cronograma, escopo e qualidade) sejam devidamente revisadas, aprovadas e incorporadas ao plano de projeto integrado em alinhamento com a estrutura de governança de programa e projeto.
			Exigir que, ao final de cada projeto, as partes interessadas apurem se o projeto gerou os resultados e benefícios planejados. Identificar e comunicar quaisquer atividades de destaque necessárias para obter os resultados esperados do projeto e os benefícios do programa.
	AI2	Adquirir e Manter Software Aplicativo	Avaliar a solução do fornecedor (incluindo o ajuste com as exigências de controle) (AI2.2).
			Instalar e configurar a aplicação, incluindo os controles configuráveis (AI2.5).

Fase	Capítulo	Macro Atividade	Atividade
	AI4	Habilitar operação e uso	Desenvolvimento e disponibilização de documentação de transferência de conhecimento.
			Produção de materiais de treinamento.
			Desenvolver / atualizar os materiais de treinamento do usuário e oferecer uma formação tal como exigido ao pessoal afetado (AI4.2, 4.3, 4.4).
			Para novos funcionários, fornecem treinamento completo sobre a aplicação e controles relevantes da aplicação (AI4.2, 4.3, 4.4).
	DS7	Educar e Treinar os Usuários	Identificar as necessidades de ensino e treinamento para a implementação do software.
ISO/IEC 27002:2005 / ISO/IEC 15408-1:2009			
Elaboração	(27002) 12.1 - Requisitos de segurança de sistemas de informação	A.12.1.1 - Análise e especificação dos requisitos de segurança	Verificar se foram especificados os requisitos para controles de segurança nas especificações de requisitos de negócios, para novos sistemas de informação ou melhorias em sistemas existentes.
	(15408) 12.6 – Critérios de Garantia de Segurança		Verificar se foram definidos os critérios de garantia de segurança no projeto de software, contemplando: Gerência de Configuração, Entrega e Operação, Desenvolvimento, Documentação e Suporte ao Ciclo de Vida do Software.
Construção	(27002) 12.2 - Processamento correto nas aplicações	A.12.2.1 - Validação dos dados de entrada	Verificar se foram implementadas validações de dados de entrada para garantir que são corretos e apropriados.
		A.12.2.2 - Controle de processamento interno	Verificar se foram incorporadas no software checagens de validação com o objetivo de detectar qualquer corrupção de informações, por erros ou por ações deliberadas.
		A.12.2.4 - Validação de dados de saída	Verificar se foram implementadas validações dos dados de saída para garantir que o processamento das informações armazenadas está correto e é apropriado às circunstâncias.
	(27002) 12.3 - Controles criptográficos	A.12.3.1 - Política para o uso de controles criptográficos	Verificar se foi desenvolvida e implementada uma política para o uso de controles criptográficos para a proteção da informação.
	(27002) 12.4 - Segurança dos arquivos do sistema	A.12.4.2 - Proteção dos dados para teste de sistema	Verificar se os dados de teste foram selecionados com cuidado, protegidos e controlados, considerando: - os procedimentos de controle de acesso, aplicáveis aos aplicativos de sistema em ambiente operacional, sejam também aplicados aos aplicativos de sistema em ambiente de teste; - seja obtida autorização cada vez que for utilizada uma cópia da informação operacional para uso de um aplicativo em teste; - a informação operacional seja apagada do aplicativo em teste imediatamente após completar o teste; - a cópia e o uso de informação operacional sejam registrados de forma a prover uma trilha para auditoria.
	(27002) 12.5 - Segurança em processos de desenvolvimento e de suporte	A.12.5.1 - Procedimentos para controle de mudanças	Verificar se foram definidos procedimentos formais para o controle de mudanças.
		A.12.5.3 - Procedimentos para controle de mudanças	Verificar se as mudanças realizadas no software foram devidamente controladas, considerando os seguintes itens: - o software original foi mantido e as mudanças aplicadas numa cópia claramente identificada; - todas as mudanças foram completamente testadas e documentadas para que possam ser reaplicadas, se necessário, em atualizações futuras do software; - se requerido, as modificações foram testadas e validadas por um grupo de avaliação independente.

Fase	Capítulo	Macro Atividade	Atividade
	(15408) 12.1 – Boas práticas de programação para desenvolvimento de aplicações seguras		Verificar se foram seguidas boas práticas de programação.
CMMi for Development (Version 1.3) / MPS.BR / NBR ISO/IEC 9126-1			
Concepção	(CMMi) Metas e Práticas Genéricas	GP 2.7 Identificar e Envolver as Partes Interessadas Relevantes	Verificar se foram identificadas e envolvidas as partes interessadas relevantes do software conforme planejado.
	(CMMi) Desenvolvimento de Requisitos	SG 1 Desenvolver Requisitos de Cliente	SP 1.1 Levantar Necessidades. SP 1.2 Desenvolver Requisitos de Cliente.
	(CMMi) Planejamento de Projeto	SG 1 Estabelecer Estimativas	SP 1.1 Estimar o Escopo do Projeto. SP 1.2 Estabelecer Estimativas para Atributos de Produtos de Trabalho e de Tarefas. SP 1.3 Definir Ciclo de Vida do Projeto. SP 1.4 Determinar Estimativas de Esforço e Custo.
	(CMMi) Gestão de Contrato com Fornecedores	SG 1 Estabelecer Contratos com Fornecedores	SP 1.1 Determinar Tipo de Aquisição. SP 1.2 Selecionar Fornecedores. SP 1.3 Estabelecer Contratos com Fornecedores.
Elaboração	(CMMi) Desenvolvimento de Requisitos	SG 2 Desenvolver Requisitos de Produto	SP 2.1 Estabelecer Requisitos de Produto e de Componente de Produto. SP 2.2 Alocar Requisitos de Componente de Produto. SP 2.3 Identificar Requisitos de Interface.
		SG 3 Analisar e Validar Requisitos	SP 3.1 Estabelecer conceitos operacionais e cenários. SP 3.2 Estabelecer uma Definição da Funcionalidade Requerida. SP 3.3 Analisar Requisitos. SP 3.4 Analisar Requisitos Visando ao Balanceamento. SP 3.5 Validar Requisitos.
	(9126) 6. Modelo de qualidade para qualidade externa e interna		Verificar se foram definidos os requisitos de qualidade externa e interna, compreendendo os seguintes aspectos: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.
	(CMMi) Gestão de Configuração	SG 1 Estabelecer *Baselines*	(SP 1.3) Criar ou liberar *baselines* para uso interno e para entrega ao cliente. Um *baseline* de software pode ser um conjunto formado por requisitos, design, arquivos de código fonte e código executável associado, arquivos de build e documentação de usuário (entidades associadas) ao qual tenha sido atribuído um identificador único.
	(CMMi) Definição dos Processos da Organização +IPPD	SG 1 Estabelecer Ativos de Processo da Organização	(SP 1.2) Estabelecer e manter as descrições dos modelos de ciclo de vida aprovados para uso na organização. Exemplos de modelos de ciclo de vida de projeto: cascata, espiral, evolutivo, incremental e iterativo.
			(SP 1.5) Estabelecer e manter a biblioteca de ativos de processo da organização (políticas, processos, planos etc.).
	(CMMi) Integração de Produto	SG 2 Assegurar Compatibilidade das *Interfaces*	SP 2.1 Revisar Descrições de *Interfaces* para Assegurar Completude. SP 2.2 Gerenciar *Interfaces.*
	(CMMi) Planejamento de Projeto	SG 2 Elaborar um Plano de Projeto	SP 2.1 Estabelecer Orçamento e Cronograma. SP 2.2 Identificar Riscos do Projeto. SP 2.3 Planejar Gestão de Dados. SP 2.4 Planejar Recursos do Projeto. SP 2.5 Planejar Habilidades e Conhecimento Necessários. SP 2.6 Planejar o Envolvimento das Partes Interessadas. SP 2.7 Estabelecer o Plano do Projeto.

Fase	Capítulo	Macro Atividade	Atividade
		SG 3 Obter Comprometimento com o Plano	SP 3.1 Revisar Planos que Afetam o Projeto. SP 3.2 Conciliar Carga de Trabalho e Recursos. SP 3.3 Obter Comprometimento com o Plano.
	(CMMi) Gestão de Requisitos	SG 1 Gerenciar Requisitos	SP 1.1 Obter Entendimento dos Requisitos. SP 1.2 Obter Comprometimento com os Requisitos. SP 1.3 Gerenciar Mudanças nos Requisitos. SP 1.4 Manter Rastreabilidade Bidirecional dos Requisitos. SP 1.5 Identificar Inconsistências entre Produtos de Trabalho, Planos de Projeto e Requisitos.
	(CMMi) Gestão de Riscos	SG 1 Preparar-se para Gestão de Riscos	SP 1.1 Determinar Fontes e Categorias de Riscos. SP 1.2 Definir Parâmetros para Riscos. SP 1.3 Estabelecer uma Estratégia para Gestão de Riscos.
		SG 2 Identificar e Analisar Riscos	SP 2.1 Identificar Riscos. SP 2.2 Avaliar, Categorizar e Priorizar Riscos.
		SG 3 Mitigar Riscos	SP 3.1 Elaborar Planos de Mitigação de Riscos. SP 3.2 Executar Planos de Mitigação de Riscos.
	(CMMi) Solução Técnica	SG 1 Selecionar Soluções de Componentes de Produto	SP 1.1 Desenvolver Soluções Alternativas e Critérios de Seleção. SP 1.2 Selecionar Soluções de Componentes de Produto.
	(CMMi) Planejamento de Projeto	SG 1 Estabelecer Estimativas	SP 1.1 Estimar o Escopo do Projeto. SP 1.2 Estabelecer Estimativas para Atributos de Produtos de Trabalho e de Tarefas. SP 1.3 Definir Ciclo de Vida do Projeto. SP 1.4 Determinar Estimativas de Esforço e Custo.
		SG 2 Elaborar um Plano de Projeto	SP 2.1 Estabelecer Orçamento e Cronograma. SP 2.2 Identificar Riscos do Projeto. SP 2.3 Planejar Gestão de Dados. SP 2.4 Planejar Recursos do Projeto. SP 2.5 Planejar Habilidades e Conhecimento Necessários. SP 2.6 Planejar o Envolvimento das Partes Interessadas. SP 2.7 Estabelecer o Plano do Projeto.
		SG 3 Obter Comprometimento com o Plano	SP 3.1 Revisar Planos que Afetam o Projeto. SP 3.2 Conciliar Carga de Trabalho e Recursos. SP 3.3 Obter Comprometimento com o Plano.
Construção	(CMMi) Metas e Práticas Genéricas	GG 2 Institucionalizar um Processo Gerenciado	(GP 2.8) Monitorar e controlar o processo em relação ao estabelecido no plano para sua execução e implementar ações corretivas apropriadas. 1. Medir o desempenho observado em relação ao previsto no plano para execução do processo. 2. Revisar as realizações e resultados do processo em relação ao previsto no plano para execução do processo. 3. Revisar atividades, status e resultados do processo com o nível gerencial imediatamente superior à gerência responsável pelo processo e identificar questões críticas. As revisões são realizadas para fornecer ao nível imediato de gerência a visibilidade necessária do processo. As revisões podem ser periódicas e por eventos. 4. Identificar e avaliar os efeitos de desvios significativos com relação ao plano para execução do processo. 5. Identificar problemas no plano para execução do processo e na execução do processo. 6. Implementar ações corretivas quando os requisitos e os objetivos não forem satisfeitos, quando forem identificados problemas, ou quando o progresso do projeto divergir significativamente do plano para execução do processo. 7. Acompanhar a ação corretiva até sua conclusão.

Fase	Capítulo	Macro Atividade	Atividade
	(CMMi) Gestão Integrada de Projeto +IPPD	SG 1 Utilizar o Processo Definido para o Projeto	(SP 1.1) Estabelecer o Processo Definido para o Projeto. O projeto é conduzido com a utilização de um processo definido que é adaptado a partir do conjunto de processos-padrão da organização. - O processo definido para o projeto deve incluir todos os processos pertencentes ao conjunto de processos-padrão da organização necessários para aquisição, desenvolvimento e manutenção do produto. Os processos de ciclo de vida relacionados ao produto, tais como os de manufatura e de suporte, são desenvolvidos em paralelo com o produto. - Recomenda-se que o processo definido para o projeto satisfaça às necessidades contratuais e operacionais, às oportunidades e às restrições do projeto. Um processo definido é elaborado visando a melhor adequação às necessidades do projeto e baseia-se em: · Requisitos do cliente. · Requisitos de produto e de componente de produto. · Compromissos. · Necessidades e objetivos do processo da organização. · Conjunto de processos-padrão da organização e diretrizes para adaptação. · Ambiente operacional. · Ambiente de negócio.
	(CMMi) Implantação de Inovações na Organização	GG 3 Institucionalizar um Processo Definido	(GP 2.2) Estabelecer e manter o plano para a execução do processo de gestão integrada de projeto.
	(CMMi) Definição dos Processos da Organização +IPPD	SG 1 Estabelecer Ativos de Processo da Organização	(SP 1.1) Estabelecer e manter o conjunto de processos-padrão da organização. - Decompor cada processo-padrão em elementos de processo, com detalhes suficientes para facilitar a compreensão e a descrição do processo (por exemplo, metodologia de design de produto de trabalho).
	(CMMi) Integração de Produto	SG 1 Preparar-se para Integração de Produto	SP 1.1 Determinar Sequência de Integração. SP 1.2 Estabelecer Ambiente de Integração do Produto. SP 1.3 Estabelecer Procedimentos e Critérios para Integração do Produto.
		SG 3 Montar Componentes do Produto e Entregar Produto	SP 3.1 Confirmar se os Componentes do Produto estão Prontos para serem Integrados. SP 3.2 Montar Componentes do Produto. SP 3.3 Avaliar Componentes de Produto Montados. SP 3.4 Empacotar e Entregar Produto ou Componente de Produto.
	(CMMi) Monitoramento e Controle de Projeto	SG 1 Monitorar o Projeto em Relação ao Plano	SP 1.1 Monitorar os Parâmetros de Planejamento do Projeto. SP 1.2 Monitorar Compromissos. SP 1.3 Monitorar Riscos do Projeto. SP 1.4 Monitorar a Gestão de Dados. SP 1.5 Monitorar o Envolvimento das Partes Interessadas. SP 1.6 Conduzir Revisões de Progresso. SP 1.7 Conduzir Revisões de Marco.
	(CMMi) Solução Técnica	SG 2 Desenvolver Design	SP 2.1 Desenvolver o Design do Produto ou dos Componentes de Produto. SP 2.2 Estabelecer Pacote de Dados Técnicos. SP 2.3 Projetar Interfaces Utilizando Critérios. SP 2.4 Analisar Alternativas: Desenvolver, Comprar ou Reusar.

Fase	Capítulo	Macro Atividade	Atividade
		SG 3 Implementar Design do Produto	SP 3.1 Implementar Design. SP 3.2 Elaborar Documentação de Suporte ao Produto.
	(CMMi) Verificação	SG 1 Preparar-se para Verificação	SP 1.1 Selecionar Produtos de Trabalho para Verificação. SP 1.2 Estabelecer Ambiente de Verificação. SP 1.3 Estabelecer Procedimentos e Critérios de Verificação.
		SG 2 Realizar Revisão por Pares	SP 2.1 Preparar-se para Revisão por Pares. SP 2.2 Conduzir Revisão por Pares. SP 2.3 Analisar Dados de Revisão por Pares.
		SG 3 Verificar Produtos de Trabalho Selecionados	SP 3.1 Realizar Verificação. SP 3.2 Analisar Resultados da Verificação.
	(CMMi) Solução Técnica	SG 3 Implementar Design do Produto	SP 3.2 Elaborar Documentação de Suporte ao Produto.
Transição	(CMMi) Gestão de Configuração	SG 1 Estabelecer *Baselines*	(SP 1.1) Identificar os itens de configuração, componentes e produtos de trabalho relacionados a serem colocados sob gestão de configuração. - Selecionar os itens de configuração e os produtos de trabalho que os compõem, com base em critérios documentados; - Atribuir identificadores únicos para os itens de configuração; - Especificar as características importantes de cada item de configuração; - Especificar quando cada item de configuração é colocado sob gestão de configuração; - Identificar o responsável por cada item de configuração.
			(SP 1.2) Estabelecer e manter um sistema de gestão de configuração e de gestão de mudanças para controlar os produtos de trabalho. - Um sistema de gestão de configuração inclui o meio de armazenamento, os procedimentos e as ferramentas para acesso ao sistema de configuração. - Um sistema de gestão de mudanças inclui o meio de armazenamento, os procedimentos e as ferramentas para registro e acesso às solicitações de mudança.
		SG 2 Acompanhar e Controlar Mudanças	(SP 2.1) Acompanhar as solicitações de mudança dos itens de configuração. 1. Iniciar e registrar as solicitações de mudança no banco de dados de solicitações de mudança. 2. Analisar o impacto das mudanças e das correções propostas pelas solicitações de mudança. 3. Revisar as solicitações de mudança que serão tratadas no próximo baseline com as partes interessadas relevantes e obter sua anuência. 4. Acompanhar o status das solicitações de mudança até sua conclusão.
		SG 2 Acompanhar e Controlar Mudanças	(SP 2.2) Controlar mudanças nos itens de configuração. 1. Controlar mudanças nos itens de configuração ao longo da vida do produto. 2. Obter autorização adequada antes de incorporar itens de configuração alterados ao sistema de gestão de configuração. 3. Realizar atividades de *check-in* e *check-out* de itens de configuração no sistema de gestão de configuração para incorporar as mudanças, de maneira que a correção e integridade dos itens de configuração sejam mantidas. 4. Realizar revisões para assegurar que as mudanças não causaram efeitos indesejáveis nos

Fase	Capítulo	Macro Atividade	Atividade
			baselines (por exemplo, assegurar que as mudanças não comprometeram a segurança física ou a segurança lógica do sistema). 5. Registrar as mudanças nos itens de configuração e os motivos das mudanças, conforme apropriado.
		SG 3 Estabelecer Integridade	(SP 3.1) Estabelecer e manter registros que descrevem os itens de configuração. 1. Registrar ações de gestão de configuração com nível suficiente de detalhe, de forma que o conteúdo e o status de cada item de configuração seja conhecido e que versões anteriores possam ser recuperadas. 2. Assegurar que as partes interessadas relevantes tenham acesso ao status dos itens de configuração e conhecimento dele. 3. Especificar a última versão dos baselines. 4. Identificar a versão dos itens de configuração que constituem um baseline específico. 5. Descrever as diferenças entre baselines sucessivos. 6. Atualizar o status e o histórico (isto é, mudanças e outras ações) de cada item de configuração, conforme necessário.
			(SP 3.2) Executar auditorias de configuração para manter a integridade dos *baselines*. 1. Avaliar a integridade dos *baselines*. 2. Confirmar se os registros de gestão de configuração identificam corretamente dos itens de configuração. 3. Revisar a estrutura e a integridade dos itens no sistema de gestão de configuração. 4. Confirmar a completude e correção dos itens no sistema de gestão de configuração. 5. Confirmar a conformidade com padrões e procedimentos aplicáveis de gestão de configuração. 6. Acompanhar os itens de ação da auditoria até sua conclusão.
	(CMMi) Treinamento na Organização	SG 2 Proporcionar Treinamento Necessário	SP 2.1 Fornecer Treinamentos.
	(CMMi) Gestão de Contrato com Fornecedores	SG 2 Cumprir Contratos com Fornecedor	SP 2.1 Executar Contrato com Fornecedor. SP 2.2 Monitorar Processos Selecionados do Fornecedor. SP 2.3 Avaliar Produtos de Trabalho Selecionados do Fornecedor. SP 2.4 Aceitar Produto Adquirido. SP 2.5 Transferir Produtos.
	(CMMi) Validação	SG 1 Preparar-se para Validação	SP 1.1 Selecionar Produtos para Validação. SP 1.2 Estabelecer Ambiente de Validação. SP 1.3 Estabelecer Procedimentos e Critérios de Validação.
		SG 2 Validar Produto ou Componentes de Produto	SP 2.1 Realizar Validação. SP 2.2 Analisar Resultados de Validação.
	(MPS.BR) Nível D	Instalação do produto	Verificar se o software foi instalado no ambiente alvo e se a documentação associada foi entregue.
		Liberação do produto	Verificar se o software atendeu aos requisitos acordados e se foi confirmada a operação correta do produto.

Fase	Capítulo	Macro Atividade	Atividade
ISO/IEC 12207 / NBR ISO/IEC 15504			
Concepção	6 Ciclo de Vida de Processos do Sistema	6.4.1 Processo de Definição de Requisitos das Partes Interessadas	6.4.1.3.1.1 O projeto deve identificar os participantes individuais ou classes de interessados que têm um interesse legítimo no sistema ao longo do seu ciclo de vida.
			6.4.1.3.2.4 O projeto deve identificar a interação entre os usuários e o sistema, tendo em conta as capacidades e limitações humanas habilidades.
			6.4.1.3.4.3 O projeto deve estabelecer com as partes interessadas que as suas necessidades estão escritas corretamente.
		6.3.4 Processo de Gestão de Risco	6.3.4.3.1.3 O responsável pelo gerenciamento de risco e executar seus papéis e responsabilidades devem ser identificados.
Elaboração	6 Ciclo de Vida de Processos do Sistema	6.3.1 Processo de Planejamento de Projetos	6.3.1.3.2.1 O gerente deve elaborar os planos para a execução do projeto. Os planos associados à execução do projeto devem conter a descrição das atividades e tarefas associadas e a identificação dos produtos de software que serão fornecidos. Estes planos devem incluir, mas não estão limitados a, o seguinte: a) Cronograma para a conclusão tempestiva das tarefas. b) Estimativa de esforço. c) os recursos adequados necessários para executar as tarefas. d) Atribuição de tarefas. e) Atribuição de responsabilidades. f) Quantificação dos riscos associados com as tarefas ou o próprio processo. g) As medidas de garantia da qualidade a ser utilizado durante o projeto. h) Os custos associados à execução do processo. i) Provisão de ambiente e infraestrutura. j) Definição e manutenção de um modelo de ciclo de vida, que é composto de fases usando os modelos de ciclo de vida definido para os projetos da organização.
		6.4.1 Processo de Definição de Requisitos das Partes Interessadas	6.4.1.3.2.1 O projeto deve contemplar requisitos das partes interessadas.
		6.4.2 Processo de Análise de Requisitos do Sistema	6.4.2.3.1.1 O uso específico pretendido do sistema a ser desenvolvido deve ser analisado para especificar os requisitos do sistema. O sistema de especificação de requisitos deve descrever: funções e capacidades do sistema, requisitos de negócio, organizacionais e de usuários, segurança, segurança, engenharia de fatores humanos (ergonomia), interface, operações e os requisitos de manutenção, restrições de projeto e requisitos de qualificação. O sistema de especificação de requisitos deve ser documentado.
		6.4.2 Processo de Análise de Requisitos do Sistema	6.4.2.3.2.1 Os requisitos de sistema devem ser avaliados considerando os critérios listados abaixo. Os resultados das avaliações devem ser documentados. a) Rastreabilidade para as necessidades de aquisição; b) Coerência com as necessidades de aquisição; c) Testabilidade; d) Viabilidade do projeto do sistema de arquitetura; e) Viabilidade da operação e manutenção.

Fase	Capítulo	Macro Atividade	Atividade
		6.4.3 Processo de projeto de arquitetura de sistema	6.4.3.3.1.1 A arquitetura de alto nível do sistema deve ser estabelecida. A arquitetura deve identificar itens de hardware, software e operações manuais. Deve assegurar-se que todos os requisitos do sistema são alocados entre os itens. Itens de configuração de Hardware, itens de configuração de software e operações manuais serão posteriormente identificados a partir desses itens. A arquitetura do sistema e os requisitos do sistema alocados para os itens devem ser documentados.
			6.4.3.3.2.1 A arquitetura do sistema e os requisitos para os itens devem ser avaliados, considerando os critérios listados abaixo. Os resultados das avaliações devem ser documentados. a) Rastreabilidade para os requisitos do sistema. b) Consistência com os requisitos do sistema. c) Adequação das normas de concepção e métodos utilizados. d) Viabilidade dos itens de software atenderem seus requisitos alocados. e) Viabilidade da operação e manutenção.
	7 Processos Específicos de Software	7.1.2 Processo de Análise de Requisitos de Software	7.1.2.3.1.1 O implementador deve estabelecer e documentar requisitos de software (incluindo as características das especificações de qualidade), descrito abaixo: a) Especificações funcionais e de capacidade, incluindo o desempenho, as características físicas e condições ambientais em que o item de software é executar. b) Interfaces externas ao item de software. c) As necessidades de qualificação. d) As especificações de segurança, incluindo aquelas relacionadas aos métodos de operação e manutenção, influências ambientais, e lesões corporais. e) Especificações de segurança, incluindo os relacionados com a fuga de informações confidenciais. f) A engenharia de fatores humanos (ergonomia), especificações, incluindo os relacionados com operações manuais, equipamentos de interações humanas, limitações de pessoal, e as áreas que necessitam de atenção concentrada humanos, que são sensíveis a erros humanos e treinamento. g) Os requisitos para definição de dados e banco de dados. h) Instalação e requisitos de aceitação do produto de software entregue na operação e manutenção do site(s). i) As exigências de documentação do usuário. j) Os requisitos de operação e execução de instruções. k) os requisitos de manutenção do usuário.
		7.1.2 Processo de Análise de Requisitos de Software	7.1.2.3.1.2 implementador deve avaliar os requisitos de software, considerando os critérios listados abaixo. Os resultados das avaliações devem ser documentados. a) Rastreabilidade para os requisitos do sistema e projeto do sistema. b) Consistência externa com os requisitos do sistema. c) A consistência interna. d) Testabilidade. e) Viabilidade do projeto de software. f) Viabilidade da operação e manutenção.

Fase	Capítulo	Macro Atividade	Atividade
		7.1.3 Processo de Projeto de Arquitetura de Software	7.1.3.3.1.1 implementador deve transformar os requisitos do item de software em uma arquitetura que descreve a sua estrutura de alto nível e identifica os componentes de software. Deve ser assegurado que todos os requisitos para o item de software são atribuídos aos seus componentes de software e aperfeiçoados para facilitar o projeto detalhado. A arquitetura do item de software deve ser documentada.
		7.2.1 Processo de Gestão de Documentação de Software	7.2.1.3.1.1 Um plano, identificando os documentos a serem produzidos durante o ciclo de vida do produto de software deve ser desenvolvido, documentado e implementado. Para obter a documentação identificada, deve-se considerar: a) Título ou nome. b) Objetivo e conteúdo. c) público-alvo. d) Os procedimentos e responsabilidades para os insumos, desenvolvimento, revisão, modificação, aprovação, produção, armazenamento, distribuição, manutenção e gerenciamento de configuração. e) Cronograma das versões intermediárias e finais.
		7.2.5 Processo de Validação de Software	7.2.5.3.1.4 plano de validação deve ser desenvolvido e documentado. O plano deve incluir, mas não está limitado a, o seguinte: a) Itens sujeitos a validação. b) as tarefas de validação para ser executada. c) Os recursos, responsabilidades e cronograma para validação. d) Os procedimentos para encaminhamento de relatórios de validação para o adquirente e outras partes.
			7.2.5.3.2.1 Preparar os requisitos de teste, casos de teste e especificações de testes para analisar os resultados do teste.
			7.2.5.3.2.2 Certifique-se que estes requisitos de teste, casos de teste e testes das especificações refletem as exigências específicas para o uso específico pretendido.
			7.2.5.3.2.1 Preparar os requisitos de teste, casos de teste especificações de testes para analisar os resultados do teste.
			7.2.5.3.2.2 Certificar-se que estes requisitos de teste, casos de teste e especificações de testes refletem as exigências específicas para o uso específico pretendido.
Construção	6 Ciclo de Vida de Processos do Sistema	6.3.2 Processo de Controle e Avaliação de Projeto	6.3.2.3.1.1 O gerente deve acompanhar a execução global do projeto, fornecendo relatórios internos do andamento do projeto e relatórios externos para o adquirente, tal como definido no contrato.
		6.3.3 Processo de Gestão de decisão	6.3.3.3.3.2 O projeto deve manter registros de problemas e oportunidades e sua disposição, tal como estipulado nos acordos ou procedimentos organizacionais e de uma forma que permite a auditoria e aprender com a experiência.
		6.3.5 Processo de Gerenciamento de Configuração	6.3.5.3.1.1 O projeto deve definir uma estratégia de gerenciamento de configuração.
			6.3.5.3.1.2 O projeto deve identificar itens que são sujeitos a um controlo de configuração.
			6.3.5.3.2.1 O projeto deve manter as informações sobre as configurações com um adequado nível de integridade e segurança.

Fase	Capítulo	Macro Atividade	Atividade
			6.3.5.3.2.2 O projeto deve garantir que as alterações às linhas de base de configuração estão devidamente identificados e registrados, avaliados, aprovados, incorporados e verificada.
		6.3.6 Processo de Gestão da Informação	6.3.6.3.1.1 O projeto deve definir os elementos de informação que serão gerenciados durante o ciclo de vida do sistema e, de acordo com a política organizacional ou de legislação, mantida por um período definido para além dela.
		6.4.5 Processo de Integração de Sistemas	6.4.5.3.1.1 Os itens de configuração de software devem ser integrados, com itens de configuração de hardware, operações manuais, e outros sistemas como necessário, para o sistema. Os agregados devem ser testados, como eles são desenvolvidos, contra as suas exigências. A integração e os resultados dos testes devem ser documentados.
			6.4.5.3.2.1 Para cada requisito de qualificação do sistema, um conjunto de testes, casos de teste (entradas, saídas e critérios de teste) e procedimentos de ensaio para a realização do Sistema de Qualificação de teste devem ser desenvolvidos e documentados. O desenvolvedor deve garantir que o sistema integrado está pronto para o sistema de teste de qualificação.
			6.4.5.3.2.2 O sistema integrado será avaliado considerando os critérios listados abaixo. Os resultados das avaliações devem ser documentados. a) Cobertura de teste dos requisitos do sistema. b) Adequação dos métodos de ensaio e os padrões utilizados. c) Conformidade com os resultados esperados. d) Viabilidade do teste de qualificação do sistema. e) Viabilidade da operação e manutenção.
		6.4.6 Processo de Teste de Qualificação de Sistema	6.4.6.3.1.1 Teste de qualificação do sistema deve ser realizado em conformidade com os requisitos de qualificação especificados para o sistema. Deve assegurar-se que a implementação de cada requisito do sistema é testada para conformidade e que o sistema está pronto para entrega. Os resultados dos testes de qualificação devem ser documentados.
			6.4.6.3.1.2 O sistema deve ser avaliado considerando os critérios listados abaixo. Os resultados das avaliações devem ser documentados. a) Cobertura de teste dos requisitos do sistema; b) Conformidade com os resultados esperados; c) Viabilidade da operação e manutenção.
		6.4.10 Processo de Manutenção de Software	6.4.10.3.2.5 O mantenedor deve obter aprovação para a opção de modificação selecionado, conforme especificado no contrato.
			6.4.10.3.4.1 O mantenedor deve realizar avaliação (s) com a organização, que autoriza a modificação para determinar a integridade do sistema modificado.
			6.4.10.3.5.1 Se um sistema ou produto de software (incluindo dados) é migrado de um velho a um novo ambiente operacional, deve ser assegurado que qualquer produto de software ou dados produzidos ou modificados durante a migração está em conformidade com esta norma.

Fase	Capítulo	Macro Atividade	Atividade
			6.4.10.3.5.2 Um plano de migração deve ser desenvolvido, documentado e executado. As atividades de planejamento devem incluir os usuários. Itens incluídos no plano: a) análise de requisitos e definição de migração. b) Desenvolvimento de ferramentas de migração. c) Conversão de produto de software e dados. d) execução de Migração. e) Verificação da migração. f) Suporte para o antigo ambiente no futuro.
	7 Processos Específicos de Software	7.1.3 Processo de Projeto de Arquitetura de Software	7.1.3.3.1.2 implementador deve desenvolver e documentar um projeto de alto nível para as interfaces externas ao item de software e entre os componentes de software do item de software.
			7.1.3.3.1.3 implementador deve desenvolver e documentar um projeto de nível superior para o banco de dados.
			7.1.3.3.1.4 O implementador deve desenvolver e documentar as versões preliminares da documentação do usuário.
			7.1.3.3.1.5 O implementador deve definir e documentar os requisitos preliminares de teste e o cronograma para a integração do software.
			7.1.3.3.1.6 implementador deve avaliar a arquitetura do item de software e os projetos de interface e de banco de dados, considerando os critérios listados abaixo. Os resultados das avaliações devem ser documentados. a) Rastreabilidade para os requisitos do item de software. b) Consistência externa com os requisitos do item de software. c) A consistência interna entre os componentes de software. d) Adequação dos métodos e padrões de projeto utilizados. e) Viabilidade do projeto detalhado. f) Viabilidade da operação e manutenção.
		7.1.4.3.1 Projeto detalhado do software.	7.1.4.3.1.1 O implementador deve desenvolver um projeto detalhado para cada componente de software do item de software. Os componentes de software devem ser refinados em níveis mais baixos contendo unidades de software que pode ser codificado, compilado e testado. Deve assegurar-se que todos os requisitos de software são alocados a partir do software componentes para unidades de software. O projeto detalhado deve ser documentado.
		7.1.4.3.1 Projeto detalhado do software.	7.1.4.3.1.2 O implementador deve desenvolver e documentar um projeto detalhado das interfaces externas ao item de software, entre os componentes e as unidades de software. O projeto detalhado das interfaces deve permitir a codificação sem a necessidade de mais informações.
			7.1.4.3.1.3 O implementador deve desenvolver e documentar um projeto detalhado para o banco de dados.
			7.1.4.3.1.4 implementador deve atualizar a documentação do usuário, conforme necessário.
			7.1.4.3.1.5 O implementador deve definir e documentar os requisitos de teste e o cronograma para testar unidades de software. Os requisitos de ensaio devem incluir a unidade de software nos limites das suas necessidades.

Fase	Capítulo	Macro Atividade	Atividade
			7.1.4.3.1.6 O implementador deve atualizar os requisitos de teste e o cronograma para a integração do software.
			7.1.4.3.1.7 O implementador deve avaliar o projeto detalhado de software e requisitos de teste, considerando os critérios listados abaixo. Os resultados das avaliações devem ser documentados. a) Rastreabilidade para os requisitos do item de software; b) Consistência externa com o projeto arquitetônico; c) A consistência interna entre os componentes e unidades de software; d) Adequação dos métodos e padrões de projeto utilizados; e) Viabilidade do teste; f) Viabilidade da operação e manutenção.
		7.1.5 Processo de Construção de Software	7.1.5.3.1.1 O implementador deve desenvolver e documentar o seguinte: a) Cada unidade de software e banco de dados. b) Os procedimentos de teste e dados para cada unidade de teste de software e banco de dados.
			7.1.5.3.1.2 O implementador deve testar cada unidade de software e banco de dados garantindo que ela satisfaça suas necessidades. Devem-se documentar os resultados dos testes.
			7.1.5.3.1.3 O implementador deve atualizar a documentação do usuário, conforme necessário.
			7.1.5.3.1.4 O implementador deve atualizar os requisitos de teste e o cronograma para a integração do software.
			7.1.5.3.1.5 O implementador deve avaliar o código do software e testar os resultados, considerando os critérios listados abaixo. Os resultados das avaliações devem ser documentados. a) Rastreabilidade para os requisitos e projeto do item de software. b) Consistência externa com os requisitos e projeto do item de software. c) A consistência interna entre as necessidades da unidade. d) A cobertura de teste das unidades. e) Adequação dos métodos de codificação e padrões utilizados. f) Viabilidade da integração e teste de software. g) Viabilidade da operação e manutenção.
		7.1.6 Processos de Integração de Software	7.1.6.3.1.1 O implementador deve desenvolver um plano de integração para integrar as unidades de software e componentes de software para o item de software. O plano deve incluir requisitos de teste, procedimentos, dados, responsabilidades e cronograma. O plano deve ser documentado.
			7.1.6.3.1.2 O implementador deve integrar as unidades de software e componentes de software e testar como os agregados são desenvolvidos em conformidade com o plano de integração. Deve assegurar-se que cada agregado satisfaz os requisitos do item de software e que o item de software é integrado na conclusão da atividade de integração. Os resultados da integração e testes devem ser documentados.
			7.1.6.3.1.4 O implementador deve desenvolver e documentar para cada requisito de qualificação do item de software um conjunto de testes, casos de teste (entradas, saídas e critérios de teste) e procedimentos de ensaio para a realização de

Fase	Capítulo	Macro Atividade	Atividade
			Teste de Qualificação de Software. O desenvolvedor deve garantir que o item de software integrado está pronto para o Teste de Qualificação de Software.
			7.1.6.3.1.5 O implementador deve avaliar o plano de integração, design, código, testes, resultados dos testes e a documentação do usuário, considerando os critérios listados abaixo. Os resultados das avaliações devem ser documentados. a) Rastreabilidade para os requisitos do sistema. b) Consistência externa com os requisitos do sistema. c) A consistência interna. d) Cobertura de teste dos requisitos do item de software. e) Adequação dos padrões de teste e métodos utilizados. f) Conformidade com os resultados esperados. g) Viabilidade de testes de qualificação de software. h) Viabilidade da operação e manutenção.
			7.1.6.3.1.6 O implementador deve conduzir revisões no projeto, sendo (7.2.6): 7.2.6.3.1 Processo de implementação; 7.2.6.3.2 Revisões do Gerenciamento do Projeto; 7.2.6.3.3 Revisões Técnicas.
		7.1.7 Processo de Teste de Qualificação de Software	7.1.7.3.1.1 implementador deve conduzir o teste de qualificação, em conformidade com os requisitos de qualificação para o item de software. Deve-se assegurar que a implementação de cada requisito de software é testada para cumprimento. Os resultados dos testes de qualificação devem ser documentados.
			7.1.7.3.1.2 O implementador deve atualizar a documentação do usuário, conforme necessário.
			7.1.7.3.1.3 O implementador deve avaliar o projeto, código, testes, resultados dos testes e a documentação do usuário, considerando os critérios listados abaixo. Os resultados das avaliações devem ser documentados. a) Cobertura de teste dos requisitos do item de software. b) Conformidade com os resultados esperados. c) Viabilidade da integração de sistemas e testes, se realizados. d) Viabilidade da operação e manutenção.
			7.1.7.3.1.4 O implementador deve apoiar auditorias (s) para garantir que: a) À medida que são codificados, os produtos de software (como um item de software) refletem a documentação do projeto. b) Os requisitos de revisão de aceitação e testes prescritos pela documentação estão adequados para a aceitação dos produtos de software. c) Os dados de teste são conformes à especificação. d) Os produtos de software foram testados com sucesso e satisfazem as suas especificações. e) Os relatórios dos testes estão corretos e discrepâncias entre os resultados reais e esperadas foram resolvidos. f) A documentação do usuário está em conformidade com as normas especificadas. g) As atividades foram realizadas de acordo com os requisitos, planos e contrato.

Fase	Capítulo	Macro Atividade	Atividade
			h) Os custos e cronogramas aderem aos planos estabelecidos. Os resultados das auditorias devem ser documentados. Se ambos, hardware e software, estiverem em desenvolvimento ou integração, as auditorias podem ser adiadas até os Testes de Qualificação do Sistema.
			7.1.7.3.1.5 Após a conclusão das auditorias, se realizado, o implementador deve atualizar e preparar o produto de software a ser entregue para a Integração de Sistemas, Sistema de Qualificação de ensaios, instalação de software, ou software de aceitação, conforme aplicável.
		7.2.2 *Software Configuration Management Process*	7.2.2.3.3.1 A seguir, devem ser realizados: identificação e registro dos pedidos de mudança, análise e avaliação das alterações e aprovação ou reprovação do pedido; e execução, verificação e liberação do item de software modificado. Uma trilha de auditoria deve existir, segundo o qual cada alteração, o motivo da modificação, e de autorização da modificação pode ser rastreado. Controle e auditoria de todos os acessos aos itens controlados por software que tratam de segurança ou de funções críticas de segurança devem ser realizados.
		7.2.4 Processo de Verificação de Software	7.2.4.3.2.2 verificação Design. O projeto deve ser verificado considerando os critérios abaixo: a) O projeto está correto e coerente com as exigências e rastreáveis. b) O projeto implementa uma boa sequencia de eventos, entradas, saídas, interfaces, fluxo de lógica, a alocação de tempo e orçamentos de dimensionamento e definição de erro, isolamento e recuperação. c) O projeto selecionado pode ser derivado de exigências. d) O projeto implementa a segurança, e outros requisitos críticos corretamente, conforme os métodos de rigor.
			7.2.4.3.2.3 código de verificação. O código deve ser verificado considerando os critérios listados abaixo: a) O código é rastreavel relativamente ao design e requisitos, testável, correto, e em conformidade com os requisitos e padrões de codificação. b) O código implementa a sequencia de eventos apropriada, interfaces consistentes, dados e fluxo de controle, integridade, alocação de tempo e de orçamentos apropriada, e definição, isolamento e recuperação. c) O código selecionado pode ser derivado do projeto ou requisitos. d) O código implementa segurança, e outros requisitos críticos corretamente, conforme os métodos de rigor.
			7.2.4.3.2.4 verificação da integração. A integração deve ser verificada considerando os critérios listados abaixo: a) Os componentes de software e unidades de cada item de software ter sido completa e corretamente integrados no item de software. b) Os itens de hardware, software e manual de operações do sistema ter sido completa e corretamente integrados no sistema. c) As tarefas de integração têm sido realizadas em conformidade com um plano de integração.

Fase	Capítulo	Macro Atividade	Atividade
			7.2.4.3.2.5 verificação de documentação. A documentação deve ser verificada considerando os critérios listados abaixo: a) A documentação está adequada, completa e consistente. b) A preparação da documentação é oportuna. c) gerência de configuração dos seguintes documentos especificados os procedimentos.
		7.2.5 Processo de Validação de Software	7.2.5.3.2.3 Conduzir os testes nos subitens 7.2.5.3.2.1 e 7.2.5.3.2.2, incluindo: a) Teste de estresse, limites e insumos singular; b) Testar o produto de software para a sua capacidade de isolar e minimizar o efeito de erros, isto é, uma degradação suave em caso de falha, o pedido de assistência de telefonista a fronteira, estresse e condições singular; c) Testes de que os usuários representantes podem alcançar com êxito as suas tarefas destinadas a utilização do produto de software.
			7.2.5.3.2.3 Conduzir os testes nos subitens 7.2.5.3.2.1 e 7.2.5.3.2.2, incluindo: a) Teste de estresse, limites e insumos singular; b) Testar o produto de software para a sua capacidade de isolar e minimizar o efeito de erros, isto é, uma degradação suave em caso de falha, o pedido de assistência de telefonista a fronteira, estresse e condições singular; c) Testes de que os usuários representantes podem alcançar com êxito as suas tarefas destinadas a utilização do produto de software.
		7.2.6 Processos de Revisão de Software	7.2.6.3.1.1 revisões periódicas devem ser realizadas em marcos predeterminados, tal como especificado no plano de projeto (s). Os interessados devem determinar a necessidade de eventuais revisões *ad hoc* em que partes convenentes poderão participar.
			7.2.6.3.2.1 situação do projeto deve ser avaliada de acordo com os planos de projeto aplicável, horários, normas e orientações. O resultado do exame deve ser considerado por uma gestão adequada e deve prever o seguinte: a) Fazer progredir de acordo com o plano de atividades, com base numa avaliação da atividade ou do estatuto de produto de software. b) Manter o controle global do projeto através da alocação adequada de recursos. c) Alterar a direção do projeto ou determinar a necessidade de um planejamento alternativo. d) Avaliar e gerenciar as questões de risco que podem comprometer o sucesso do projeto.
Transição	6 Ciclo de Vida de Processos do Sistema	6.4.10 Processo de Manutenção de Software	6.4.10.3.1.2 O mantenedor deve estabelecer procedimentos para recebimento, registro e acompanhamento de relatórios de problemas e pedidos de modificação dos usuários e fornecer *feedback* para os usuários. Sempre que forem encontrados problemas, eles devem ser registrados e inseridos no Processo de Resolução de problema de software (subitem 7.2.8).
			6.4.10.3.2.1 O mantenedor deve analisar o relatório de problema ou pedido de modificação de seu impacto sobre a organização, o sistema existente, e os sistemas de interface para o seguinte: a) Tipo, por exemplo, corretivo, melhoria, preventivas ou de adaptação ao novo ambiente; b) Âmbito, por exemplo, o tamanho da alteração,

Fase	Capítulo	Macro Atividade	Atividade
			os custos envolvidos, tempo para modificar; c) Criticidade: por exemplo, o impacto sobre o desempenho, a segurança ou a segurança.
	7 Processos Específicos de Software	7.2.2 Processo de Gerenciamento de Configuração de Software	7.2.2.3.5.1 Deve ser determinado e garantido: a completude funcional dos itens de software aos seus requisitos e integridade física dos itens de software (se o seu design e o código refletem uma descrição técnica atualizada).
		7.2.4 Processo de Verificação de Software	7.2.4.3.2.1 requisitos de verificação. Os requisitos devem ser verificados considerando os critérios listados abaixo: a) Os requisitos do sistema são consistentes, viáveis, e testáveis. b) Os requisitos do sistema foram apropriadamente alocados aos itens de hardware, software e operações manuais de acordo com critérios do projeto. c) Os requisitos de software são consistentes, viáveis, testáveis e refletem com precisão os requisitos do sistema. d) Os requisitos de software relacionados com a segurança, a segurança e a criticidade estão corretas, conforme demonstrado, pelos métodos de rigor.
		7.2.5 Processo de Validação de Software	7.2.5.3.1.5 O plano de validação deve ser implementado. Todos os problemas e as não conformidades devem ser resolvidos. Resultados das atividades de validação devem ser colocados à disposição do adquirente e outras organizações envolvidas.
			7.2.5.3.2.4 Validar que o produto de software satisfaz seu uso pretendido.
			7.2.5.3.2.5 Teste o produto de software, conforme apropriado, em áreas selecionadas do ambiente de destino.
			7.2.5.3.1.1 A determinação deve ser feita se o projeto justifica um esforço de validação e o grau de autonomia de organização de que o esforço necessário.
			7.2.5.3.1.2 Se o projeto justifica um esforço de validação, um processo de validação deve ser estabelecido para validar o sistema ou produto de software. Validação de tarefas definidas abaixo, incluindo os métodos associados, técnicas e ferramentas para executar as tarefas, devem ser selecionados.
			7.2.5.3.1.3 Se o projeto justifica um esforço independente, uma organização qualificada e responsável para conduzir o esforço deve ser selecionada. Ao condutor deve ser assegurada a independência e autoridade para desempenhar as funções de validação.
			7.2.5.3.1.4 plano de validação deve ser desenvolvido e documentado. O plano deve incluir, mas não está limitado a, o seguinte: a) Itens sujeitos a validação. b) as tarefas de validação para ser executada. c) Os recursos, responsabilidades e cronograma para validação. d) Os procedimentos para encaminhamento de relatórios de validação para o adquirente e outras partes.

Fase	Capítulo	Macro Atividade	Atividade
			7.2.5.3.1.5 O plano de validação deve ser implementado. Problemas e não conformidades detectadas pelo esforço de validação devem ser inseridos no Processo de Resolução de problema de software (subitem 7.2.8). Todos os problemas e as não conformidades devem ser resolvidos. Resultados das atividades de validação devem ser colocados à disposição do adquirente e outras organizações envolvidas.
			7.2.5.3.2.4 Validar que o produto de software satisfaz seu uso pretendido.
			7.2.5.3.2.5 Teste o produto de software, conforme apropriado, em áreas selecionadas do ambiente de destino.
		7.2.6 Processos de Revisão de Software	7.2.6.3.3.1 análises técnicas devem ser realizadas para avaliar os produtos de software ou serviços em questão e fornecer provas de que: a) Eles são completos. b) Estão em conformidade com seus padrões e especificações. c) Mudanças neles são devidamente implementadas e afetam apenas as áreas identificadas pelo *Configuration Management Process* (subitem 7.2.2). d) Eles estão aderindo aos cronogramas aplicáveis. e) Eles estão prontos para a próxima atividade prevista. f) O desenvolvimento, operação ou manutenção está sendo realizada de acordo com os planos, programas, normas e diretrizes do projeto.

Apêndice B - Mapeamento de atividades para o método de auditoria aos projetos de software baseados no Processo Unificado

Macro Atividade	Objetivo de Controle	Grupo de Atividades	Atividade
Concepção	Garantir que sejam estabelecidos o escopo e viabilidade econômica do projeto	Estabelecer o escopo e os limites do projeto	Identificar os principais *stakeholders* que têm um interesse legítimo no sistema ao longo do seu ciclo de vida e atribuir funções relacionadas com a aplicação de controles e responsabilidades.
			Levantar necessidades do software.
			Especificar interfaces com outros sistemas.
			Estabelecer conceitos operacionais e cenários.
			Estabelecer uma definição da funcionalidade requerida.
			Estimar o Escopo do Projeto.
			Definir práticas e padrões de qualidade (metodologia, políticas e procedimentos para o desenvolvimento de softwares).
			Definir e documentar a natureza e o escopo do projeto, visando confirmar e desenvolver um entendimento comum do escopo do projeto com as partes interessadas e quanto ao relacionamento com outros projetos de um programa de investimento em TI. A definição deve ser formalmente aprovada pelo patrocinador do programa e pelo patrocinador do projeto antes de seu início.
		Realizar o planejamento inicial	Realizar uma avaliação dos riscos relacionados com a informação e com os riscos de processamento das informações relacionadas.
			Realizar e validar os resultados de estudos de viabilidade.
			Determinar Estimativas de Esforço e Custo.
			Identificar Riscos do projeto.
Elaboração	Garantir que sejam definidas a funcionalidade do software, seus requisitos e restrições sobre sua operação	Analisar e validar Requisitos	Analisar os requisitos.
			Validar os requisitos (funcionais e não funcionais, de cliente, de usuário, de *stakeholders*, de sistema, de software, técnicos e de negócio, de interface (IHC), para controles de segurança, de produto, de componente de produto e de qualificação) e objetivos de controle.
			Verificar se foram definidos os critérios de garantia de segurança no projeto de software, contemplando: Gerência de Configuração, Entrega e Operação, Desenvolvimento, Documentação e Suporte ao Ciclo de Vida do Software.
			Verificar se foram definidos os requisitos de qualidade externa e interna, compreendendo os seguintes aspectos: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.
		Definir a especificação formal do software	Definir especificação formal (processo de software, interface de subsistema e comportamento).
		Definir especificação para sistemas críticos	Definir especificação dirigida a riscos.
			Definir especificação de segurança, proteção e confiabilidade de software.
		Definir o planejamento do projeto de desenvolvimento de software	Estabelecer Orçamento e Cronograma.
			Planejar Gestão de Dados.
			Planejar Recursos do Projeto.
			Planejar Habilidades e Conhecimento Necessários.
			Planejar o Envolvimento das Partes Interessadas.
			Estabelecer o Plano do Projeto.
			Revisar Planos que Afetam o Projeto.
			Definir as características técnicas que formam a infraestrutura do software e agrupa-las de acordo com a prioridade do cliente.
			Definir e validar junto ao cliente o escopo do projeto, considerando: necessidade do negócio e valores do negócio,

Macro Atividade	Objetivo de Controle	Grupo de Atividades	Atividade
			características/necessidades dos usuários finais, saídas visíveis ao usuário requeridas, restrições do negócio, cenários de uso visíveis ao cliente usando o formato padrão, saídas e entradas resultantes, características, funções e comportamentos importantes do software e riscos de negócios definidos pelo cliente.
			Definir o projeto no nível de componentes a partir de notações no contexto da engenharia de software orientada a objetos ou convencional (programação estruturada).
		Definir o projeto de arquitetura do software	Traduzir os requisitos de negócio nas especificações de projeto.
		Definir o projeto de interface com o usuário	Definir o projeto de interface do usuário, considerando: - identificação dos usuários finais do sistema; - análise de tarefas e ações do usuário; - análise ambiental, identificando as estruturas físicas e sociais nas quais a interface deve operar.
		Definir os requisitos do software	Definir os requisitos funcionais e não funcionais.
			Definir os requisitos de cliente.
			Definir os requisitos de usuário.
			Definir os requisitos de *stakeholders.*
			Definir os requisitos de sistema.
			Definir os requisitos de software.
			Definir os requisitos técnicos e de negócio.
			Definir os requisitos de *interface* (IHC).
			Definir os requisitos para controles de segurança.
			Definir os objetivos de controle relevantes.
			Definir os requisitos de produto e de componente de produto.
			Definir os requisitos de qualificação.
			Alocar Requisitos de componente de produto.
			Verificar se foram definidos e validados pelo cliente os requisitos do software, bem como foram negociados a prioridade, a disponibilidade e o custo relativos a cada requisito.
			Documentar os requisitos de software.
			Gerenciar os requisitos.
			Definir modelos de sistema.
		Definir políticas, processos e procedimentos para as etapas de desenvolvimento do software	Definir procedimentos para definição dos requisitos (estudo de viabilidade, elicitação e análise de requisitos, especificação de requisitos e validação de requisitos).
			Estabelecer e manter a biblioteca de ativos de processo da organização (políticas, processos, planos etc.).
		Desenvolver o software conforme o projeto	Verificar se o usuário avaliou e validou o protótipo.
			Verificar se o código-fonte do sistema ou componente de software reusável está disponível.
			Verificar se novos requisitos ou requisitos modificados são revalidados pelos clientes antes que as mudanças sejam implementadas.
		Educar e treinar os usuários envolvidos no desenvolvimento do software	Identificar as necessidades de ensino e treinamento para a implementação do software.
		Gerenciar a qualidade do software	Preparar um plano de gestão de qualidade que descreva o sistema de qualidade de projeto e como será implementado.
			Criar um plano de garantia de qualidade de software para definir a estratégia de *Software Quality Assurance* - SQA de uma equipe de software.
		Gerenciar o projeto de desenvolvimento do software	Eliminar ou minimizar riscos específicos associados ao projeto através de um processo sistemático de planejamento, identificação, análise, resposta, monitoramento e controle de áreas ou eventos com potencial para causar mudanças indesejadas.

Macro Atividade	Objetivo de Controle	Grupo de Atividades	Atividade
			Obter Entendimento dos Requisitos.
			Obter Comprometimento com os Requisitos.
			Gerenciar Mudanças nos Requisitos.
		Gerenciar o projeto de desenvolvimento do software	Manter Rastreabilidade Bidirecional dos Requisitos.
			Identificar Inconsistências entre Produtos de Trabalho, Planos de Projeto e Requisitos.
			Avaliar, Categorizar e Priorizar Riscos do projeto.
			Elaborar Planos de Mitigação de Riscos.
			Executar Planos de Mitigação de Riscos.
			Determinar Fontes e Categorias de Riscos.
			Definir Parâmetros para Riscos.
			Estabelecer uma Estratégia para Gestão de Riscos.
Construção	Garantir a entrega de um sistema de software em funcionamento, de acordo com sua especificação, e da documentação associada pronta para ser liberada para os usuários.	Adquirir e Manter a Infraestrutura de Tecnologia	Estabelecer um ambiente de desenvolvimento e de teste para proporcionar eficiência e eficácia nos testes de viabilidade e integração dos componentes da infraestrutura.
			Prever segregação entre as atividades de desenvolvimento, teste e operação.
		Atualizar a documentação do usuário	Atualizar a documentação do usuário, conforme necessário.
		Definir a modelagem de análise do software	Definir as representações que mostram os requisitos de software quanto à informação, função e comportamento (modelos baseados em cenário, modelos de fluxo, modelos baseados em classe e modelos comportamentais).
			Definir a modelagem de análise, contemplando o domínio da informação, domínio funcional, domínio comportamental e a interface do usuário.
			Definir os modelos de projeto, contemplando a arquitetura do software, a interface do usuário e detalhes em nível de componentes. Verificar se todos os modelos estão completos e consistentes.
			Desenvolver um modelo de implantação completo e consistente.
		Definir controles a serem implementados no software	Avaliar e concluir sobre a suficiência do projeto de controles de aplicação.
			Incorporar no software checagens de validação com o objetivo de detectar qualquer corrupção de informações, por erros ou por ações deliberadas.
			Implementar validações dos dados de saída para garantir que o processamento das informações armazenadas está correto e é apropriado às circunstâncias.
			Desenvolver e implementar uma política para o uso de controles criptográficos para a proteção da informação.
			Implementar validações de dados de entrada para garantir que são corretos e apropriados.
		Definir estratégias de testes de software	Desenvolver uma especificação de teste para documentar a abordagem da equipe de software para o teste, definindo um plano que descreve uma estratégia global e um procedimento que define passos específicos de teste e os testes que serão conduzidos (softwares convencionais ou orientados a objetos).
		Definir o modelo de projeto de software	Definir o modelo de projeto e verificar se inclui elemento de projeto de dados, elemento do projeto arquitetural, elementos de projeto de interface, elementos de projeto no nível de componente e elementos de projeto no nível de implantação.
		Definir o planejamento do projeto de desenvolvimento de software	Estabelecer Estimativas para Atributos de Produtos de Trabalho e de Tarefas.
			Definir Ciclo de Vida do Projeto.
			Conciliar Carga de Trabalho e Recursos.
			Obter Comprometimento com o Plano.
			Definir as tarefas de trabalho, responsabilidade por cada tarefa e produtos de trabalho a serem produzidos.

Macro Atividade	Objetivo de Controle	Grupo de Atividades	Atividade
		Definir o planejamento do projeto de desenvolvimento de software	Definir o projeto de software abrangendo: - arquitetura do software; - definição de subsistemas e estabelecimento de mecanismos de comunicação entre subsistemas; - identificação e descrição detalhada de cada um dos componentes; - projeto de interfaces externas, internas e com o usuário.
			Estabelecer o Processo Definido para o Projeto.
		Definir o planejamento para a evolução do software	Definir um processo formal para evolução de software que inclua as atividades fundamentais de análise de mudanças, planejamento de releases e implementação de mudanças.
			Desenvolver a estratégia e o plano de manutenção de software aplicativo.
		Definir o planejamento para validação do software	Definir o processo de teste (testes de componente (ou unidade), de sistema e de aceitação).
			Desenvolver e documentar um plano de validação do software.
		Definir o planejamento para verificação do software	Selecionar Produtos de Trabalho para Verificação.
			Estabelecer Ambiente de Verificação.
			Estabelecer Procedimentos e Critérios de Verificação.
			Definir o planejamento para verificação do software, contemplando sua conformidade à função do software, às expectativas do usuário e ao ambiente de mercado.
			Definir o planejamento para validar o código-fonte do sistema (inspeção e análise estática).
			Desenvolver, documentar e aprovar um plano de teste do software em desenvolvimento.
		Definir o projeto de arquitetura do software	Revisar detalhadamente o projeto de especificação da aplicação (incluindo controles de aplicação automática).
			Assegurar que as funcionalidades automatizadas sejam desenvolvidas em conformidade com as especificações de projeto, padrões de desenvolvimento e documentação e requisitos de qualidade e de autorização.
			Definir o projeto arquitetural e verificar se contempla: - sistema representado no contexto (o projetista deve definir as entidades externas com as quais o software interage e a natureza da interação); - identificação de um conjunto de abstrações de mais alto nível chamado de arquétipos que representam os elementos-chave do comportamento ou função do sistema; - identificação e representação dos componentes no contexto de uma arquitetura que lhes dá suporte; - desenvolvimento de instanciações específicas da arquitetura para provar o projeto em um contexto do mundo real.
		Definir o projeto de testes de software	Projetar testes de unidade para cada componente de software.
			Projetar e documentar os casos de teste para exercitar a lógica interna, interfaces, colaborações de componentes e os requisitos externos do software com o intuito de detectar erros no software.
		Definir técnicas para o desenvolvimento de sistemas críticos	Verificar se o processo de software é confiável, contemplando atividades de prevenção de defeitos, detecção de defeitos e tolerância a defeitos.
			Verificar se é realizada atividade de verificação da programação com base na especificação do sistema.
			Verificar se foi adotada linguagem de programação com tipos fortes de dados (como Java ou ADA) para o desenvolvimento.
			Utilizar técnicas de programação segura no desenvolvimento do sistema.
			Utilizar técnicas de que garantam que as informações sejam protegidas - deve ser usada uma abordagem para projeto e implementação de software baseada em ocultamento e encapsulamento de informações.

Macro Atividade	Objetivo de Controle	Grupo de Atividades	Atividade
		Definir uma estratégia de integração	Desenvolver uma estratégia de integração.
		Definir uma estratégia de testes adicionais	Definir uma estratégia de testes adicionais para validação de sistemas críticos.
		Definir uma estratégia de validação	Desenvolver uma estratégia de validação.
		Desenvolver o plano de testes do software	Definir o plano de testes às funcionalidades.
		Desenvolver o software conforme o projeto	Desenvolver o Design do Produto ou dos Componentes de Produto.
			Estabelecer Pacote de Dados Técnicos.
			Projetar Interfaces Utilizando Critérios.
			Definir responsáveis por realizar a programação.
			Desenvolver e avaliar um protótipo.
			Realizar os testes unitários do sistema.
			Revisar Descrições de Interfaces para Assegurar Completude.
			Gerenciar Interfaces.
			Confirmar se os Componentes do Produto estão Prontos para serem Integrados.
			Montar Componentes do Produto.
			Avaliar Componentes de Produto Montados.
			Empacotar e Entregar Produto ou Componente de Produto.
			Determinar Sequência de Integração.
			Estabelecer Procedimentos e Critérios para Integração do Produto.
			Construir o protótipo operacional da interface com o usuário.
			Procurar, compreender e adaptar os componentes reusáveis.
			Implementar *Design.*
			Elaborar Documentação de Suporte ao Produto.
			Verificar se o processo de evolução contempla a modificação da especificação do software, do projeto e dos requisitos (se necessário).
			Seguir boas práticas de programação (segurança).
		Gerenciar configurações do software	Definir um plano de gestão de configuração de software definindo a estratégia do projeto para modificação.
			Acompanhar as solicitações de mudança dos itens de configuração.
			Controlar mudanças nos itens de configuração.
			Identificar os itens de configuração, componentes e produtos de trabalho relacionados a serem colocados sob gestão de configuração.
			Estabelecer e manter registros que descrevem os itens de configuração.
			Estabelecer e manter um sistema de gestão de configuração e de gestão de mudanças para controlar os produtos de trabalho.
			Determinar e garantir: a completude funcional dos itens de software aos seus requisitos e integridade física dos itens de software (se o seu design e o código refletem uma descrição técnica atualizada).
			Definir uma estratégia de gestão de configuração para o projeto.
			Definir o plano para o gerenciamento de configurações, prevendo: itens de configuração, responsável pelos procedimentos de gerenciamento de configuração, políticas de gerenciamento de configuração, ferramentas a serem usadas no gerenciamento de configurações e estrutura do banco de dados de configuração.
			Implantar um processo de gestão de configuração de softwares que identifique, controle, audite e relate modificações realizadas no software.

Macro Atividade	Objetivo de Controle	Grupo de Atividades	Atividade
			Verificar se o processo de gestão de configurações contempla a identificação, controle de versão, controle de modificação, auditoria de configuração e preparação de relatórios.
			Manter informações sobre configurações com um nível adequado de integridade e segurança.
			Verificar se são realizadas auditorias de configuração de software.
			Verificar se há ferramenta para automatizar o processo de construção de sistemas (compilação e ligação de componentes de software num programa que executa determinada configuração definida).
			Verificar se as alterações às linhas de base de configuração estão devidamente identificados e registrados, avaliados, aprovados, incorporados e verificados.
			Realizar a identificação e registro dos pedidos de mudança, análise e avaliação das alterações e aprovação ou reprovação do pedido; e execução, verificação e liberação do item de software modificado. Uma trilha de auditoria deve existir, segundo o qual cada alteração, o motivo da modificação, e de autorização da modificação pode ser rastreado. Controle e auditoria de todos os acessos aos itens controlados por software que tratam de segurança ou de funções críticas de segurança devem ser realizados.
		Gerenciar mudanças no software	Verificar se o processo de controle de modificações no software contempla as seguintes atividades: - pedido formal de modificação; - análise (aceitação ou rejeição) do pedido de modificação; - atualização controlada do item de configuração de software que deve ser modificado.
			Verificar se há um banco de dados ou repositório para manter os itens de configuração do software.
			Estabelecer procedimentos formais de gerenciamento de mudanças para lidar de modo padronizado com todas as solicitações de mudança em aplicações, procedimentos, processos, parâmetros de sistema, parâmetros de serviço e plataformas subjacentes (inclusive solicitações de manutenção e reparo).
			Atualizar a documentação dos procedimentos do sistema e de usuários sempre que forem implementadas mudanças no sistema.
			Estabelecer um sistema de controle de mudança para cada projeto, de forma que todas as mudanças feitas no escopo original do projeto (como custo, cronograma, escopo e qualidade) sejam devidamente revisadas, aprovadas e incorporadas ao plano de projeto integrado em alinhamento com a estrutura de governança de programa e projeto.
			Verificar se as mudanças realizadas no software foram devidamente controladas, considerando os seguintes itens: - o software original foi mantido e as mudanças aplicadas numa cópia claramente identificada; - todas as mudanças foram completamente testadas e documentadas para que possam ser reaplicadas, se necessário, em atualizações futuras do software; - se requerido, as modificações foram testadas e validadas por um grupo de avaliação independente.
			Aderir aos padrões de desenvolvimento em todas as modificações.
		Gerenciar o projeto de desenvolvimento do software	Assegurar que a fase de início do projeto seja formalmente aprovada e comunicada a todas as partes interessadas. Prever a aprovação das fases subsequentes com base na revisão e na aceitação dos resultados entregues da fase anterior e na aprovação de um estudo de caso atualizado na próxima revisão geral do programa.

Macro Atividade	Objetivo de Controle	Grupo de Atividades	Atividade
			Definir responsabilidades, relacionamentos, autoridades e critérios de desempenho para os membros da equipe de projeto e especificar a base de aquisição e atribuição de funcionários e/ou prestadores de serviço competentes para o projeto.
			Estabelecer um plano integrado de projeto formalizado e aprovado (que abranja recursos de negócio e de sistemas de informação) para orientar a execução e o controle em todas as etapas do projeto.
			Definir o acompanhamento dos resultados do patrocinador do programa, patrocinador do projeto, comitê diretor, coordenador e gerente do projeto e os mecanismos pelos quais eles podem cumprir com essas responsabilidades (como relatórios e revisões de estágios do projeto).
			Avaliar o desempenho do projeto em comparação com critérios-chave (como escopo, cronograma, qualidade, custo e risco).
			Exigir que, ao final do projeto, as partes interessadas apurem se o projeto gerou os resultados e benefícios planejados. Identificar e comunicar quaisquer atividades de destaque necessárias para obter os resultados esperados do projeto e os benefícios do programa.
			Monitorar e controlar o processo em relação ao estabelecido no plano para execução do processo, e implementar ações corretivas apropriadas.
			Monitorar os Parâmetros de Planejamento do Projeto.
			Monitorar Compromissos.
			Monitorar Riscos do Projeto.
			Monitorar a Gestão de Dados.
			Monitorar o Envolvimento das Partes Interessadas.
			Conduzir Revisões de Progresso.
			Conduzir Revisões de Marco.
		Realizar a validação do software	Realizar os testes de integração e realizar as correções necessárias aos erros identificados.
			Avaliar o sistema considerando os critérios listados abaixo e documentar os resultados das avaliações: a) Cobertura de teste dos requisitos do sistema; b) Conformidade com os resultados esperados; c) Viabilidade da operação e manutenção.
			Realizar testes dos controles da aplicação em conformidade com o plano de teste.
			Preparar os requisitos de teste, casos de teste e especificações de testes para analisar os resultados do teste com base nas exigências específicas para o uso específico pretendido.
		Realizar a verificação do software	Avaliar o sistema integrado considerando os critérios listados abaixo e documentar os resultados: a) Cobertura de teste dos requisitos do sistema. b) Adequação dos métodos de ensaio e os padrões utilizados. c) Conformidade com os resultados esperados. d) Viabilidade do teste de qualificação do sistema. e) Viabilidade da operação e manutenção.
			Conduzir revisões no projeto, sendo: Processo de implementação; Revisões do Gerenciamento do Projeto; e Revisões Técnicas.
			Avaliar o projeto, código, testes, resultados dos testes e a documentação do usuário, considerando os critérios listados abaixo. Os resultados das avaliações devem ser documentados. a) Cobertura de teste dos requisitos do item de software. b) Conformidade com os resultados esperados. c) Viabilidade da integração de sistemas e testes, se realizados. d) Viabilidade da operação e manutenção.
			Avaliar o plano de integração, design, código, testes, resultados dos testes e a documentação do usuário, considerando os critérios listados abaixo. Os resultados das avaliações devem ser documentados.

Macro Atividade	Objetivo de Controle	Grupo de Atividades	Atividade
			Os requisitos de verificação devem ser verificados considerando os critérios listados abaixo: a) Os requisitos do sistema são consistentes, viáveis e testáveis. b) Os requisitos do sistema foram apropriadamente alocados aos itens de hardware, software e operações manuais de acordo com critérios do projeto. c) Os requisitos de software são consistentes, viáveis, testáveis e refletem com precisão os requisitos do sistema. d) Os requisitos de software relacionados com a segurança, a segurança e a criticidade estão corretas, conforme demonstrado, pelos métodos de rigor.
			O design do projeto deve ser verificado considerando os critérios listados abaixo: a) O projeto está correto e coerente com as exigências e rastreáveis. b) O projeto implementa uma boa sequencia de eventos, entradas, saídas, interfaces, fluxo de lógica, a alocação de tempo e orçamentos de dimensionamento e definição de erro, isolamento e recuperação. c) O projeto selecionado pode ser derivada de exigências. d) O projeto implementa a segurança, e outros requisitos críticos corretamente, conforme os métodos de rigor.
			O código do software deve ser verificado considerando os critérios listados abaixo: a) O código é rastreável relativamente ao design e requisitos, testável, correto, e em conformidade com os requisitos e padrões de codificação. b) O código implementa a sequencia de eventos apropriada, interfaces consistentes, dados e fluxo de controle, integridade, alocação de tempo e de orçamentos apropriada, e definição, isolamento e recuperação. c) O código selecionado pode ser derivado do projeto ou requisitos. d) O código implementa segurança, e outros requisitos críticos corretamente, conforme os métodos de rigor.
			Verificar a integração considerando os seguintes critérios: a) Os componentes de software e unidades de cada item de software ter sido completa e corretamente integrados no item de software. b) Os itens de hardware, software e manual de operações do sistema ter sido completa e corretamente integrados no sistema. c) As tarefas de integração têm sido realizadas em conformidade com um plano de integração.
			Verificar a documentação considerando os critérios listados abaixo: a) A documentação está adequada, completa e consistente. b) A preparação da documentação é oportuna. c) gerência de configuração dos seguintes documentos especificados os procedimentos.
			Realizar análises técnicas para avaliar os produtos de software ou serviços em questão e fornecer provas de que: a) Eles são completos. b) Estão em conformidade com seus padrões e especificações. c) Mudanças neles são devidamente implementadas e afetam apenas as áreas identificadas pelo *Configuration Management Process* (subitem 7.2.2). d) Eles estão aderindo aos cronogramas aplicáveis. e) Eles estão prontos para a próxima atividade prevista. f) O desenvolvimento, operação ou manutenção está sendo realizada de acordo com os planos, programas, normas e diretrizes do projeto.
			Realizar Verificação.
			Analisar Resultados da Verificação.

Macro Atividade	Objetivo de Controle	Grupo de Atividades	Atividade
		Realizar auditoria técnica do software	Apoiar auditorias (s) técnicas para garantir que: a) Como codificada, os produtos de software (como um item de software) refletem a documentação do projeto. b) Os requisitos de revisão de aceitação e testes prescritos pela documentação está adequada para a aceitação dos produtos de software. c) Os dados de teste compatíveis com a especificação. d) Os produtos de software foram testados com sucesso e satisfazer as suas especificações. e) Os relatórios dos testes estão corretos e discrepâncias entre os resultados reais e esperadas foram resolvidos. f) A documentação do usuário está em conformidade com as normas especificadas. g) As atividades foram realizadas de acordo com os requisitos, planos e contrato. h) Os custos e cronogramas aderir aos planos estabelecidos. Os resultados das auditorias devem ser documentados. Se ambos, hardware e software estão em desenvolvimento ou de integração, as auditorias podem ser adiados até que o Sistema de Qualificação de teste.
			Após a conclusão das auditorias, se realizado, o implementador deve atualizar e preparar o produto de software a ser entregue para a Integração de Sistemas, Sistema de Qualificação de ensaios, instalação de software, ou software de aceitação, conforme aplicável.
		Realizar revisão por pares nos produtos do projeto	Preparar-se para Revisão por Pares.
			Conduzir Revisão por Pares.
			Analisar Dados de Revisão por Pares.
Transição	Garantir que seja entregue um sistema de software documentado, funcionando corretamente em seu ambiente operacional e que seja validado pelo cliente para comprovar que atende aos seus requisitos	Gerenciar a implantação do software no ambiente de produção	Instalar e configurar a aplicação, incluindo os controles configuráveis.
			Instalar o produto no ambiente alvo e verificar se atendeu aos requisitos do cliente e se foi confirmada a operação correta do produto.
		Educar e treinar os usuários envolvidos no desenvolvimento do software	Definir a comunicação e treinamento de usuários, gestores de negócio, equipes de suporte e equipes de operação.
			Desenvolver / atualizar os materiais de treinamento do usuário e oferecer uma formação tal como exigido ao pessoal afetado.
			Fornecer Treinamentos.
		Gerenciar configurações do software	Executar auditorias de configuração para manter a integridade dos baselines.
		Definir o planejamento para validação do software	Estabelecer procedimentos e critérios de validação.
			Definir o planejamento para validação (o sistema realiza o que o usuário necessita) do software, contemplando sua conformidade à função do software, às expectativas do usuário e ao ambiente de mercado.
		Gerenciar a implantação do software no ambiente de produção	Após a conclusão dos testes, controlar a transferência dos sistemas alterados para operação, de acordo com o plano de implementação. Obter a aprovação das partes interessadas, como usuários, proprietário do sistema e gerência operacional. Quando apropriado, executar o sistema em paralelo com o sistema antigo durante um período e comparar comportamento/resultados.
			Estabelecer procedimentos em linha com o gerenciamento de mudanças organizacionais para garantir a realização da revisão pós-implementação, conforme definido no plano de implementação.
		Realizar a validação do software	Estabelecer um plano de implementação e de retorno à configuração anterior. Obter aprovação de todas as partes relevantes.

Macro Atividade	Objetivo de Controle	Grupo de Atividades	Atividade
			Estabelecer um ambiente de testes seguro que reflita o ambiente de operações planejado no que diz respeito à segurança, controles internos, práticas operacionais, exigências de qualidade e confidencialidade e cargas de trabalho.
			Selecionar os dados de teste com cuidado, proteção e controle.
			Realizar o teste de qualificação do sistema em conformidade com os requisitos de qualificação especificados para o sistema e documentar os resultados.
			Testar o produto de software, conforme apropriado, em áreas selecionadas do ambiente de destino.
			Conduzir o teste de qualificação, em conformidade com os requisitos de qualificação para o item de software e documentar os resultados dos testes de qualificação.
			Identificar, registrar, priorizar e resolver os erros e os problemas identificados durante os testes.
			Analisar resultados de validação.
			Documentar e interpretar os resultados finais do teste, incluindo questões não resolvidas.
			Assegurar que as mudanças sejam testadas de maneira independente e de acordo com o plano de testes definido antes da migração para o ambiente de produção.
			Aprovar os resultados dos testes e determinar a execução das atividades de controle do aplicativo.
			Assegurar que o gerenciamento do departamento usuário e da área de TI avalie o resultado do processo de testes como determinado no plano de testes. Corrigir erros significativos identificados no processo de testes, executar todos os testes listados no plano de testes, bem como qualquer teste de regressão necessário. Após a avaliação, aprovar a promoção para a produção (AI7.7).
			Estabelecer um ambiente de validação seguro que reflita o ambiente de operações planejado no que diz respeito à segurança, controles internos, práticas operacionais, exigências de qualidade e confidencialidade e cargas de trabalho.
			Realizar os testes de validação e realizar as correções necessárias aos erros identificados.
			Realizar os testes de aceitação junto ao cliente, obter aprovação e realizar as correções necessárias aos erros identificados.
			Validar que o produto de software satisfaz seu uso pretendido.
		Realizar a verificação do software	Se o projeto justifica um esforço independente, uma organização qualificada e responsável para conduzir o esforço deve ser selecionada. Ao condutor deve ser assegurada a independência e autoridade para desempenhar as funções de validação.

Apêndice C - Aspectos que podem influenciar auditorias em projetos de software baseados no Processo Unificado

1. INTRODUÇÃO

O objetivo deste questionário é colher a opinião de especialistas no processo de engenharia de software e auditoria de sistemas sobre as atividades pertinentes a um método de auditoria para apoio ao desenvolvimento de sistemas, bem como os aspectos que podem influenciar a utilização desse método.

2. CARACTERIZAÇÃO DO ESPECIALISTA

Nome (opcional):		E-mail (opcional):	
ÁREA DE ATUAÇÃO			
Empresa:		Universidade:	
	Auditor de Sistemas/TI		Professor de Engenharia de Software/Computação/Informática
	Gerente de Auditoria de Sistemas		Pesquisador doutor em Engenharia de Software (ou área relacionada)
	Gerente de TI/Qualidade		Pesquisador mestre em Engenharia de Software (ou área relacionada)
	Gerente de Projeto		Pesquisador doutorando em Engenharia de Software (ou área relacionada)
	Analista de Sistemas		Pesquisador mestrando em Engenharia de Software (ou área relacionada)
	Outro:		Outro:
Tempo de atuação na área: _____ anos		Número de projetos (software) que já participou: _____	

FORMAÇÃO: *Nível e Área*			
	Doutorado	**( )** Eng de Software/Computação/Informática	**( )** Outro:
	Mestrado	**( )** Eng de Software/Computação/Informática	**( )** Outro:
	Especialização	**( )** Eng de Software/Computação/Informática	**( )** Outro:
	Graduação	**( )** Eng de Software/Computação/Informática	**( )** Outro:

EXPERIÊNCIA EM PROCESSO DE SOFTWARE
Como você classificaria o seu conhecimento da área de processos de software? **( )** Excelente **( )** Alto **( )** Médio **( )** Baixo **( )** Nenhum
Como você classificaria a sua experiência prática em processos de software? **( )** Excelente **( )** Alto **( )** Médio **( )** Baixo **( )** Nenhum
EXPERIÊNCIA EM AUDITORIA DE SISTEMAS
Como você classificaria o seu conhecimento da área de auditoria de sistemas? **( )** Excelente **( )** Alto **( )** Médio **( )** Baixo **( )** Nenhum
Como você classificaria a sua experiência prática em auditoria de sistemas? **( )** Excelente **( )** Alto **( )** Médio **( )** Baixo **( )** Nenhum

3. IMPORTÂNCIA DAS ATIVIDADES PARA UM MÉTODO DE AUDITORIA NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE BASEADO NO PROCESSO UNIFICADO

INSTRUÇÕES

Considere o conjunto de atividades listadas abaixo. Marque com um "X" a coluna que representa a sua opinião a respeito do grau de importância de cada atividade para o processo de desenvolvimento de softwares, considerando a seguinte escala:

0. Sem importância
1. Pouca importância
2. Média importância
3. Importante
4. Muito importante

O conjunto de atividades é completo? Caso você encontre alguma que não tenha sido incluída no conjunto listado abaixo, acrescente sua descrição no final da tabela, nas linhas em branco e avalie segundo a escala.

Atividades presentes no processo de desenvolvimento de software		**0**	**1**	**2**	**3**	**4**
Concepção	1. Estabelecer o escopo e os limites do projeto					
	2. Realizar o planejamento inicial					
Elaboração	3. Definir políticas, processos e planos para orientar o desenvolvimento do software					
	4. Definir processos e procedimentos para definição dos requisitos do software					
	5. Definir os requisitos do software					
	6. Analisar e validar Requisitos					
	7. Definir a especificação formal do software					
	8. Definir especificação para sistemas críticos					
	9. Definir o modelo do sistema					
	10. Definir o planejamento do projeto de desenvolvimento de software					
	11. Definir o projeto de arquitetura do software					
	12. Definir o projeto de desenvolvimento do software					
	13. Definir o projeto de interface com o usuário					
	14. Definir o projeto no nível de componentes					
	15. Desenvolver o software conforme as mudanças requeridas					
	16. Desenvolver o software conforme o projeto					
	17. Educar e treinar os usuários envolvidos no desenvolvimento do software					
	18. Gerenciar a qualidade do software					
	19. Gerenciar o projeto de software					
	20. Gerenciar os requisitos do software					
	21. Gerenciar os riscos do projeto de desenvolvimento de software					

Construção	22. Definir a modelagem de análise do software					
	23. Definir o modelo de projeto de software					
	24. Definir o planejamento do projeto de desenvolvimento de software					
	25. Definir o planejamento para a evolução do software					
	26. Definir o planejamento para validação do software					
	27. Definir o planejamento para verificação do software					
	28. Definir controles a serem implementados no software					
	29. Definir o projeto de arquitetura do software					
	30. Definir o projeto de desenvolvimento do software					
	31. Definir o projeto de testes de software					
	32. Definir técnicas para o desenvolvimento de sistemas críticos					
	33. Desenvolver o plano de testes do software					
	34. Desenvolver o software conforme o projeto					
	35. Desenvolver o software conforme as mudanças requeridas					
	36. Gerenciar a configuração do software					
	37. Gerenciar configurações do software					
	38. Gerenciar mudanças no software					
	39. Gerenciar o projeto de software					
	40. Definir estratégias de testes de software					
	41. Definir uma estratégia de integração					
	42. Definir uma estratégia de testes adicionais					
	43. Definir uma estratégia de validação					
	44. Realizar a verificação do software					
	45. Realizar a validação do software					
	46. Realizar auditoria técnica do software					
	47. Realizar revisão por pares nos produtos do projeto					
	48. Adquirir e Manter a Infraestrutura de Tecnologia					
	49. Atualizar a documentação do usuário					
Transição	50. Definir o planejamento para validação do software					
	51. Educar e treinar os usuários envolvidos no desenvolvimento do software					
	52. Gerenciar configurações do software					
	53. Implementar o software no ambiente de produção					
	54. Realizar a verificação do software					
	55. Realizar a validação do software					

4. ASPECTOS QUE PODEM INFLUENCIAR A AUDITORIA NO PROCESSO UNIFICADO DE DESENVOLVIMENTO DE SOFTWARE

INSTRUÇÕES

Considere o conjunto de itens listados abaixo. Marque com um “X” a coluna que representa a sua opinião a respeito do grau no qual o processo de desenvolvimento de softwares é influenciado pelos itens listados, considerando a seguinte escala:

0. Sem influência
1. Pouca influência
2. Média influência
3. Influente
4. Muita influência

O conjunto de aspectos é completo? Caso você encontre algum aspecto que não tenha sido incluído no conjunto listado abaixo, acrescente sua descrição no final da tabela, nas linhas em branco e avalie segundo a escala.

Aspectos que podem influenciar o apoio da auditoria de sistemas num processo de desenvolvimento de software	**0**	**1**	**2**	**3**	**4**
1. Equipe com experiência em auditoria no desenvolvimento de software					
2. Gerência com experiência em auditoria no desenvolvimento de software					
3. Participação do cliente nas etapas do processo					
4. Existência de apoio automatizado					
5. Alto comprometimento da gerência com o projeto					
6. Apoio da direção da empresa					
7. Treinamento formal da equipe no processo de desenvolvimento de software					
8. Responsabilidades claramente definidas					
9. Existência de orientações para a realização da auditoria no desenvolvimento de software					
10. Apoio à utilização do conhecimento de experiências em projetos anteriores					
11. Ambiente físico de trabalho adequado					
12. Existência de um grupo de Auditoria de Sistemas na empresa					
13. Existência de auditorias da aderência ao processo de desenvolvimento de software					
14. Iniciar a implantação da auditoria no desenvolvimento de softwares por um conjunto específico de atividades					
15. Iniciar a implantação da auditoria no desenvolvimento de softwares de maneira uniforme					
16. Disciplina no atendimento às recomendações geradas pela equipe de auditoria de sistemas					
17. O processo de desenvolvimento de software estar alinhado aos objetivos de negócio da organização					
18. O processo de desenvolvimento de software estar baseado em expectativas realistas					