
# AVALIAÇÃO DA USABILIDADE E MANUTENIBILIDADE DE MODELOS DE PROCESSO DE SOFTWARE EM BPMN

André Luis Nogueira Campos

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Toacy Cavalcante de Oliveira

Rio de Janeiro

Março de 2017

AVALIAÇÃO DA USABILIDADE E MANUTENIBILIDADE DE MODELOS DE PROCESSO DE SOFTWARE EM BPMN

André Luis Nogueira Campos

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Toacy Cavalcante de Oliveira, D.Sc.

Prof. Ana Regina Cavalcanti da Rocha, D.Sc.

Prof. Marcos Roberto da Silva Borges, Ph.D.

Prof. Geraldo Bonorino Xexéo, D.Sc.

Prof. Adriano Bessa Albuquerque, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2017

Campos, André Luis Nogueira

Avaliação da usabilidade e manutenibilidade de modelos de processo de software em BPMN / André Luis Nogueira Campos. – Rio de Janeiro: UFRJ/COPPE, 2017.

XVIII, 270 p.: il.; 29,7 cm.

Orientador: Toacy Cavalcante Oliveira

Tese (doutorado) – UFRJ/COPPE, Programa de Engenharia de Sistemas e Computação, 2017.

Referências Bibliográficas: p. 154-164.

1. Processo de software. 2. Qualidade. 3. Modelo de avaliação. 4. ISO 25.000. 5. MPS.BR e CMMI. I. Oliveira, Toacy Cavalcante. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

*Até onde eu possa lembrar, minha primeira incursão no mundo da engenharia foi um convite para ir à bancada e construir meus próprios brinquedos. Bem orientado, fiz boneco trapezista de madeira e cartolina, bolinha catapultada na cestinha de basquete, carrinho de mão de madeira que funcionava igual ao de gente grande, caleidoscópio de cartolina, rodelas de vidro e tiras de espelho, balanço de pendurar em árvore, e tantos outros artefatos que ainda existem em minha memória. Eu não tinha 7 anos de idade, e já acreditava ser capaz de construir qualquer coisa que eu quisesse. A vida foi ficando mais complicada, mas eu continuei acreditando, mesmo nas situações mais improváveis. Acreditar fez diferença para mim. Por isso, dedico esse trabalho à esta pessoa especial que me ensinou a acreditar – meu avô Arthur.*

*Em memória.*

# AGRADECIMENTOS

Essas duas pessoas me apoiam, me equilibram e me suportam. E foram fundamentais durante todo o meu doutorado. Contribuíram muito, e de diversas formas, inclusive me tolerando nos piores dias. Por isso sou muito grato. Obrigado minha querida esposa, e obrigado meu filho e grande amigo.

Agradeço aos professores de maneira geral, essas pessoas especiais que continuam construindo o nosso futuro, mesmo vivendo em um presente que não os valoriza adequadamente, e que não lhes retribui por tudo o que fizeram e fazem. Agradeço mais especificamente aos professores do PESC, em especial à Ana Regina, Cláudia, Guilherme e, claro, ao Toacy, que aceitou o desafio de me orientar nesta jornada. Agradeço ainda aos professores que se doaram ao participar da minha banca de defesa, Marcos, Xexéo e Adriano, além de Ana Regina e Toacy, mais uma vez.

Quero agradecer também aos colegas do grupo de pesquisa PRISMA. As discussões com todos foram sempre enriquecedoras, em especial as com o Fábio e Raquel; embora tenham sido poucas, foram fundamentais. Obrigado Raquel, pelo apoio e diversos "salvamentos" ao longo do curso. Edson, Renata, Alcileia, Ulisses – vocês são um grande time. Obrigado.

Agradeço à Fiocruz por todo o apoio, direto e indireto, ao longo de todo o curso. Quero agradecer em especial ao Alvaro Funcia, por sua parceria, por seu ideário positivista e construtivista, que me serviram, por vezes, de farol de milha. Agradeço também ao Pedro Barbosa e ao Maurício Zuma, que de formas diferentes contribuíram muito neste estirão acadêmico, mesmo que talvez não saibam exatamente como. Mas eu sei, e agradeço.

Agradeço ao Criador dos bilhões de estrelas, das partículas subatômicas, da vasta capacidade humana de aprender, e da nossa insaciável sede de saber. Afinal, 'o sábio escuta e assimila mais instrução; o homem que tem entendimento obtém orientação perita' (*Provérbios 1:5*).

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc)

# AVALIAÇÃO DA USABILIDADE E MANUTENIBILIDADE DE MODELOS DE PROCESSO DE SOFTWARE EM BPMN

André Luis Nogueira Campos

Março/2017

Orientador: Toacy Cavalcante de Oliveira

Programa: Engenharia de Sistemas e Computação

O modelo de processo de software é um elemento fundamental para as iniciativas de melhoria de software, porque ele materializa os conceitos de Engenharia de Software e as melhores práticas que devem ser adotadas no ciclo de vida de desenvolvimento de software. Embora iniciativas como CMMI-DEV e MR-MPS-SW tenham sido propostas para melhorar os processos de software, relativamente poucos trabalhos foram identificados na literatura abordando a perspectiva do modelo de processo. Além disso, essas iniciativas de melhoria da qualidade do software não se concentram em propor uma abordagem detalhada e sistemática sobre como definir um conjunto de conhecimentos para avaliar a qualidade dos modelos de processo com base em padrões internacionais. Neste trabalho, apresentamos uma abordagem sistemática para avaliar a qualidade dos modelos de processo de software em relação às características de usabilidade e manutenibilidade. A solução proposta define processos para a definição de Medidas de Qualidade (QMs) e Elementos de Medição de Qualidade (QMEs) para as características escolhidas, com base em uma revisão de literatura e com base na família de normas ISO/IEC 25.000, conhecido como SQuaRE (Requisitos e Avaliação de Qualidade de Sistemas e Software). O trabalho também define um processo sobre como usar QMs e QMEs para avaliar um modelo de processo de software específico e executa esse processo de avaliação em um cenário do mundo real.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc)

# EVALUATION OF USABILITY AND MAINTTAINABILITY OF SOFTWRE PROCESS MODELS IN BPMN

André Luis Nogueira Campos

March/2017

Advisor: Toacy Cavalcante de Oliveira

Department: System Engineering and Computing

A Software Process Model is a key element for the software improvement initiatives because Software Process Models materialize Software Engineering concepts and best practices that should be adopted throughout the software development lifecycle. Although initiatives such as CMMI-DEV and MR-MPS-SW have been proposed to improve Software Processes, only a few were found in the literature addressing the process modelling perspective. Moreover, those initiatives do not provide a detailed and systematic approach on how to define a body of knowledge for assessing the quality of process models based on international standards. In this work, we present a systematic approach to assess the quality of process models with respect to the usability and maintainability characteristics. The proposed solution defines processes for capturing Quality Measures(QMs) and Quality Measurement Elements(QMEs) for the chosen characteristics, based on a Literature Review and the ISO/IEC 25.000 standard, known as SQuaRE (Systems and software Quality Requirements and Evaluation). The work also defines a process on how to use QMs and QMEs to assess a given software process model, and executes this assessment process in a real-world scenario.

**Sumário**

**Lista de figuras**

**Lista de tabelas**

# 1 Introdução

## 1.1 Introdução do capítulo

Neste capítulo é apresentado o contexto da pesquisa, a motivação para sua realização, a definição do problema e as contribuições esperadas. São apresentadas ainda as limitações do trabalho e a metodologia de pesquisa.

## 1.2 Contexto

A capacidade computacional vem aumentando de maneira importante nas últimas décadas, implicando no uso cada vez mais intenso dessa tecnologia por pessoas e empresas, promovendo uma relação simbiótica entre Tecnologia e Sociedade, de modo que uma produz impacto e gera demandas para a outra. Como consequência, mais software é construído para controlar essa capacidade computacional e entregar para as pessoas e organizações o resultado que elas esperam. E este software precisa estar cada vez mais disponível, desde grandes instalações centrais até estruturas distribuídas e móveis (SAHA e MUKHERJEE, 2003) (SURRY e BAKER, 2016).

Diante desta expectativa as organizações que produzem software, tanto para uso próprio quanto para uso de terceiros, buscam melhorar continuamente a qualidade de seus produtos. Além da elevada expectativa dos usuários, outros desafios para estas organizações têm sido a rápida evolução tecnológica, o surgimento de um número crescente de concorrentes, e a necessidade de distribuição em larga escala. (MAGDALENO, WERNER e ARAUJO, 2011) (PRAHALAD e KRISHNAN, 2008).

O caminho adotado por muitas organizações para produzir software com qualidade tem sido investir na melhoria do processo de desenvolvimento de software. Deste modo, espera-se que bons processos resultem em software de boa qualidade (UNTERKALMSTEINER, GORSCHEK, *et al.*, 2012). E, de fato, os estudos demonstram que a melhoria dos modelos e padrões no processo de desenvolvimento de software (PDS) pode melhorar tanto a qualidade do software produzido quanto a própria produtividade desse desenvolvimento (BUTTLER, 1995) (YAMAMURA, 1999) (PITTERMAN, 2000) (GIBSON, GOLDENSON e KOST, 2006) (MOHD, AHMAD e HASSAN, 2008) (HANI, 2009).

Segundo (NIAZI, 2015), com o objetivo de melhorar efetivamente o processo de software, diferentes abordagens têm sido propostas, das quais a SPI (*Software Process Improvement*) é a mais amplamente adotada. Nesta abordagem, as organizações utilizam

padrões e modelos existentes, como por exemplo a família de normas ISO/IEC 9.000 (JIANG, KLEIN, *et al.*, 2004), o CMMI-DEV (*Capability Maturity Model Integration for Development*) (COLEMAN e O'CONNOR, 2008), e o MR MPS-SW (*Modelo de Referência MPS para Software*) (WEBER, ARAÚJO, *et al.*, 2005). Tanto o CMMI-DEV quanto o MR-MPS-SW estimulam a adoção de boas práticas nos processos de desenvolvimento de software e também nos processos gerenciais relacionados ao desenvolvimento do software, tais como gestão de projetos, gestão da qualidade, entre outros.

Por exemplo, o MR MPS-SW, focado principalmente nas pequenas e médias empresas, define processos e práticas que as organizações devem implementar em seus processos de software para obterem conformidade com o modelo. Estes processos são organizados em 7 níveis de maturidade e estão fortemente alinhados com as normas ISO/IEC 12.207 e ISO/IEC 33.000 (SOFTEX, 2016) (WEBER, ARAÚJO, *et al.*, 2005). Este programa tem produzido importantes benefícios, integrando a Indústria, o Governo e a Academia, contribuindo para a construção de software de melhor qualidade para usuários e clientes, apoiando as políticas de governo para a implementação de boas práticas em engenharia de software, e proporcionando um canal para a transferência do conhecimento produzido nas Universidades para a Indústria (MONTONI, ROCHA e WEBER, 2009) (KALINOWSKI, WEBER, *et al.*, 2014);

De modo similar, O CMM *(Capability Maturity Model*) é um modelo de maturidade e capacidade que começou a ser construído nos anos 1980 (HUMPHREY, 1989). O CMM, originalmente composto por partes, foi integrado e publicado em 2002 como CMMI (*Capability Maturity Model Integration*), como um modelo com 5 níveis de maturidade. O objetivo do CMMI é propor um conjunto de boas práticas a serem implementados pelas organizações que buscam aderência ao modelo em disciplinas específicas, disponibilizando modelos tais como o CMMI-DEV, para desenvolvedores de produtos, o CMMI-ACQ, para aquisição e terceirização de bens e serviços, e o CMMI-SVC, para a avaliação da qualidade de processos das organizações que prestam serviços (LIOU, 2011). Há evidências de que a implementação das boas práticas propostas pelo CMMI-DEV traz benefícios importantes para as organizações, como por exemplo o aumento de lucro por funcionário, a redução de erros nos produtos de software, e o aumento de funcionalidades entregues ao cliente/usuário (FALESSI, SHAW e MULLEN, 2014).

Tanto o MR-MPS-SW quanto o CMMI-DEV propõem a adoção de boas práticas de engenharia de software pelas organizações. A base para a implementação dessas boas práticas é o processo de software. Em outras palavras, o processo de software precisa ser ajustado à luz dessas boas práticas para que os potenciais resultados proporcionados pelos modelos de maturidade e capacidade possam realmente ser alcançados.

Mas, como esses modelos são explicitados e percebidos pelas pessoas? De acordo com (SERRANO e WOLFF, 2000) toda atividade cognitiva humana pode ser descrita em termos de símbolos, esquemas, imagens, ideias e outras formas que são as próprias representações mentais. Desse modo, a partir de representações externas, tais como estruturas linguísticas ou diagramáticas, o indivíduo constrói suas representações internas, ou seja, a própria compreensão da realidade que ele observa. Sendo assim, os processos de software precisam ser materializados em representações externas, seja por uma estrutura escrita, por uma estrutura diagramática, ou por uma estrutura que mescle a escrita e a diagramática. A partir dessa representação externa o indivíduo poderá construir sua representação mental (ou interna) e então compreender a realidade que ele observa.

Considerando estes aspectos, o modelo do processo de software, ou sua representação, se constitui em ferramenta fundamental para as iniciativas de melhoria da qualidade do software, uma vez que é esta representação que viabiliza a compreensão desse processo pelos indivíduos. Ainda, é por meio da representação do processo de software que poderão ser descritos e implementadas as práticas propostas pelo MR-MPS-SW e pelo CMMI-DEV, por exemplo.

## 1.3 Motivação

O processo aplicado para a produção de um documento formal que explicite um problema ou aspecto de um domínio, com o objetivo de permitir a compreensão e comunicação desse problema ou aspecto é chamado de modelagem conceitual (MOODY, 2005). Deste modo, a representação de um processo de software é um modelo conceitual, pois ele explicita um conhecimento, ou aspecto, do domínio da Engenharia de Software, que se refere a como desenvolver um software.

Existem diversos estudos apontando a necessidade dessa modelagem conceitual que resulta na representação do processo de software, e sobre a essencialidade desse processo para as iniciativas de melhoria da qualidade do software. No entanto, apesar de vasta publicação sobre processos de software, relativamente pouco se tem publicado sobre o

modelo de processo de software, especialmente sobre a qualidade deste modelo. Por exemplo, (HUMPHREY, SNYDER e WILIS, 1991) destacaram a importância de se realizar avaliações no processo de software da organização e reafirmaram sobre como esse tipo de iniciativa pode ajudar as organizações a elevarem seu nível de maturidade do processo de software.

(HERBSLEB, ZUBROW, *et al.*, 1997) defenderam que o CMM teve enorme impacto na prática de engenharia de software, e que os processos de software baseados nas propostas do CMM foram beneficiados com maior facilidade de compreensão e aumento de desempenho. Adicionalmente, (DIAZ e SLIGO, 1997) propuseram medidas e dados que podiam demonstrar os resultados positivos da aplicação das boas práticas de CMM no processo de software.

(DIBA, 2000) fez um estudo em 120 organizações para coletar práticas em iniciativas de melhoria do processo de software que se mostraram bem sucedidas. (RAVICHANDRAM e RAI, 2000) observaram que os objetivos de qualidade do software são atingidos de maneira mais eficiente quando a alta direção da organização estabelece uma infraestrutura capaz de promover a melhoria do processo de software e encorajar os envolvidos a participarem na construção do processo.

O trabalho de (CONRADI e FUGGETA, 2002) trouxe a atenção para o fato de que as organizações precisam ser mais flexíveis ao implementar seus programas de melhoria do processo de software, pois uma implementação fria e estrita das recomendações dos modelos de melhoria poderia implicar no fracasso de projetos nessa área. Os autores (BADDOO e HALL, 2003) apresentaram possíveis motivos para o insucesso de projetos de melhoria do processo de software, destacando principalmente a resistência à mudança, a ausência de evidências dos resultados desse tipo de iniciativa, os projetos baseados em imposição para os colaboradores, a insuficiência de recursos e as pressões comerciais.

(IVERSEN, MATHIASSEN e NIELSEN, 2004) afirmaram que muitas organizações que desenvolvem iniciativas de melhoria de seus processos de software aumentaram sua capacidade de desenvolvimento e a qualidade de suas soluções. No entanto, os autores destacaram a complexidade desse tipo de iniciativa e apontaram a necessidade de se gerenciar os riscos envolvidos. Um aspecto da complexidade nas iniciativas de melhoria do processo de software é apresentado por (DYBA, KITCHENHAM e JORGENSEN, 2005), e se refere ao risco da tomada de decisões incorretas ao se adotar novas técnicas que desconsideram evidências científicas de sua adequação e efetividade.

(NIAZI, WILSON e ZOWGHI, 2005) afirmaram que melhorar a qualidade do processo de software é em si mesmo um processo, e de alta complexidade. Por isso, é preciso definir e adotar uma estratégia clara e eficiente para a adoção de modelos e padrões de melhoria do processo de software, tal como o CMMI-DEV. Já (STAPLES, NIAZI, *et al.*, 2007) alertaram para o fato de que muitas organizações de pequeno e médio porte entendem que a adoção das práticas recomendadas pelo CMMI-DEV seria benéfica para seus processos de software. Apesar disso, muitas dessas organizações não adotam o CMMI-DEV porque se consideram muito pequenas, porque a implementação dessas boas práticas teria custo elevado, porque essas organizações não dispõem de tempo, ou porque elas optaram por outra forma de melhorar seus processos de software.

Em (MONTONI e ROCHA, 2010) os autores afirmaram que muitas organizações enfrentam dificuldades para implementar seus programas de melhoria do processo de software em função de obstáculos tais como a falta de motivação e o baixo apoio da alta administração da organização. O estudo propõe uma compreensão precisa do contexto sociocultural da organização para facilitar a implementação das estratégias de melhoria do processo de software. No trabalho de (BARRETO, MURTA e ROCHA, 2011) se defende que o modelo do processo de software possui similaridade com o produto de software, e como o produto de software pode ser componentizado e reutilizado, de forma similar o processo de software também poderia ser componentizado e reutilizado.

(UNTERKALMSTEINER, GORSCHEK, *et al.*, 2012) fizeram uma revisão da literatura sobre processos de avaliação e medição do processo de melhoria do processo de software, considerando 148 artigos publicados entre 1991 e 2008. Eles investigaram o impacto das iniciativas de melhoria da qualidade do software, e em como esse impacto poderia ser medido.

Os autores (NIAZI, BABAR e VERNER, 2010) realizaram uma pesquisa exploratória para entender os obstáculos que podem prejudicar os projetos de melhoria do processo de software, sob a perspectiva do mercado. O trabalho identificou diversos obstáculos, como a ausência de gestão do projeto, ausência de recursos, ausência de um patrocinador, e inexperiência da equipe, como fatores que prejudicam ou impedem as iniciativas de melhoria do processo de software. Em (CLARKE e O'CONNOR, 2013) os autores estudaram a melhoria do processo de software nas pequenas e médias empresas, apontando para uma dicotomia entre os modelos de maturidade e padrões de qualidade, focados em aspectos técnicos, e o processo de software que é uma atividade fortemente

executada por humanos. Na opinião dos autores, os modelos de maturidade e padrões de qualidade deveriam explorar melhor o capital humano das organizações como vantagem competitiva para as mesmas.

O estudo realizado por (NIAZI, 2015) destacou que os projetos de melhoria do processo de software, apesar de serem iniciativas relativamente antigas, ainda apresentam importantes desafios de implantação. O autor pesquisou os fatores de sucesso para projetos desse tipo, e identificou fatores tais como o trabalho de conscientização das pessoas envolvidas, o trabalho de revisão dos processos, e a realização de treinamentos específicos sobre melhoria do processo de software. Em (MAGDALENO, DE OLIVEIRA BARROS, *et al.*, 2015) os autores apresentaram uma abordagem de otimização para potencializar a colaboração na composição de processos de software. O objetivo é viabilizar a composição do processo de software para um projeto de desenvolvimento específico, de maneira colaborativa.

Ainda, diversos trabalhos têm sido publicados sobre a medição do processo de software. Aspectos tais como o tempo empregado em cada atividade, os custos dessas atividades, o número de defeitos, os atrasos e os resultados obtidos são alguns dos itens utilizados nessas medições, e diversos trabalhos têm sido publicados nessa área, tanto com aplicações ao processo de software quanto ao próprio processo de melhoria do processo de software (BASILI e GREEN, 1994) (KHOSHGOFTAAR, 1998) (LEHMAN, PERRY e RAMIL, 1998) (COOK e WOLF, 1999) (JALOTE e SAXENA, 2002) (DENARO e PEZZÈ, 2002) (SILLITTI, JANES, *et al.*, 2003) (RAINER e HALL, 2003) (BERANDER e JONSSON, 2006) (MOHAGHEGHI e CONRADI, 2007) (ALSHAYEB e LI, 2005) (UNTERKALMSTEINER, GORSCHEK, *et al.*, 2012).

Como apresentado, há trabalhos sobre qualidade do software tratando de diferentes dimensões dessa qualidade. No entanto, segundo (GARCÍA, RUIZ e PIATTINI, 2004b), os estudos sobre qualidade em processo de software têm sido focados em projetos e em produtos, e não em modelos de processo de software; especialmente aqueles que trataram de medições de qualidade.

Como mencionado anteriormente, um modelo de processo de software é na verdade um modelo conceitual, ou seja, é uma representação explícita de parte da realidade que suporta a modelagem de atividades humana complexas (KROGSTIE, SINDER e JORGENSEN, 2006). (MOODY, 2005) afirmou que não existem padrões amplamente aceitos para a a avaliação da qualidade de modelos conceituais. Segundo o autor,

enquanto há padrões internacionais para avaliação de produtos de software, como a família ISO/IEC 25.000, não existem padrões equivalentes para a avaliação da qualidade do modelo de processo de software.

Portanto, se o processo de software é fundamental para as iniciativas de melhoria da qualidade do software, então o modelo do processo de software ocupa importância central nas iniciativas de melhoria do processo de software (HJALMARSSON e LIND, 2004).

De acordo com (NELSON, POELS, *et al.*, 2005) e (KROGSTIE, SINDER e JORGENSEN, 2006), uma representação deve ser precisa, completa, de uso (e reuso) viável, para suportar a execução adequada do processo. Eles afirmaram ainda que quando o modelo inicial está inadequado, então é provável que as atividades baseadas nesse modelo não alcancem os objetivos desejados. Por outro lado, de acordo com (LINDLAND, SINDRE e SOLVBERG, 1994) nem mesmo a mais brilhante solução para um problema seria de alguma utilidade se ninguém a conseguisse entender.

Definir modelos de processo de software é uma tarefa complexa e que demanda por considerável conhecimento e experiência. Em diversos casos as organizações contam com o apoio de consultores externos para a construção de seus modelos de processo de software (FERREIRA, CERQUEIRA, *et al.*, 2006) (GUERRA, TRAVASSOS, *et al.*, 2006). Considerando tudo isso, entendemos que é importante estabelecer mecanismos para avaliação da qualidade dos modelos de processo de software, tanto para aferir a qualidade dos modelos construídos quanto para estabelecer um referencial de qualidade para a construção de novos modelos. De acordo com (WAND e WEBER, 2002) os modelos conceituais com qualidade elevada tendem a facilitar a detecção e correção e erros, e para (MOODY, 2005) o custo e complexidade da correção de erros em modelos conceituais são muito menores do que a correção de erros durante a execução desses modelos.

Alguns estudos têm buscado, de alguma forma, avaliar a qualidade do modelo conceitual do processo de software, ou seja, do modelo de processo de software. Falando de maneira geral, esses estudos buscam identificar elementos da representação do processo de software que de alguma forma possam ser medidos com o objetivo de identificar oportunidades de melhoria da qualidade desse processo (GARCIA, RUIZ, *et al.*, 2003) (GARCÍA, RUIZ e PIATTINI, 2004) (GARCÍA, RUIZ e PIATTINI, 2004b) (CANFORA, GARCÍA, *et al.*, 2005) (GARCÍA, PIATTINI, *et al.*, 2005) (CANFORA,

GARCIA, *et al.*, 2006) (DA SILVA, MACIEL e RAMALHO, 2013)[1]. Mas, no entanto, percebemos que esses estudos não abordam a questão da avaliação da qualidade do modelo de processo de software de maneira estruturada e alinhada com padrões de qualidade existentes, deixando assim uma lacuna no que se refere à avaliação da qualidade de modelos de processo de software.

Tratando dessa lacuna, (MOODY, 2005) se preocupa em abordar os aspectos práticos e teóricos no que se refere a avaliação da qualidade em modelos conceituais. Em seu trabalho, (MOODY) propõe um conjunto de 5 princípios para garantir um método estruturado e sistêmico de avaliação da qualidade de modelos conceituais:

1. Os requisitos de qualidade para modelos conceituais devem ser decompostos em uma hierarquia de características, subcaracterísticas e medidas;
2. As características e subcaracterística devem ser rotuladas com uma única palavra fundamental para cada uma, e que essas palavras sejam de fácil compreensão;
3. Cada característica e subcaracterística deve ter uma definição única, em uma sentença concisa;
4. Medidas devem ser definidas para cada uma das subcaracterísticas;
5. Deve ser definido um processo claro e suficiente com procedimentos a serem realizados para a avaliação da qualidade do modelo conceitual.

Ainda, de acordo com o autor, a ausência de um mecanismo estruturado para avaliação de um modelo conceitual implica em que esses modelos são avaliados com base meramente na opinião dos indivíduos que o avaliam, de maneira pontual, e suportadas apenas no "senso comum". O autor destaca que esta forma de avaliação está mais próxima da disciplina da Arte do que da disciplina da Engenharia. O trabalho deste autor aponta a ausência de padrões estruturados para a avaliação de modelos como uma lacuna entre essas disciplinas (Arte e Engenharia).

Os trabalhos apresentados na corrente seção indicam a possibilidade de maiores estudos sobre avaliação da qualidade de modelos de processo de software, como apoio de medições relacionadas a características e subcaracterísticas de qualidade estabelecidas por padrões reconhecidos internacionalmente.

[1] Estes trabalhos podem ser observados em mais detalhes na seção 3.4.

## 1.4 Questão de pesquisa

Considerando possibilidade identificada na seção1.3, o objetivo principal do presente trabalho pode ser sintetizado na questão a seguir:

> Q: A qualidade do modelo de processo de software pode ser avaliada com base em medições relacionadas a um modelo de qualidade de referência.

A partir desta questão principal, concordamos com (MOODY, 2005), que traz à atenção questões que se subordinam à questão principal deste trabalho, que seriam:

> $Q_1$: Que medidas podem ser definidas e como elas podem ser relacionadas a um modelo de qualidade de referência, compondo um modelo de medição?
>
> $Q_2$: Que atividades compõem um processo de avaliação do modelo de processo de software?
>
> $Q_3$: É viável aplicar um modelo de medição e um processo de avaliação a modelos de processos de software reais?

## 1.5 Objetivos

O objetivo principal deste trabalho, portanto, pode ser descrito da seguinte maneira;

> O: Propor um processo de avaliação da qualidade para modelos de processo de software, baseado em medições associadas a um modelo de qualidade de referência.

Assim como ocorre com a questão de pesquisa, esse objetivo principal pode ser descrito em partes menores, como segue:

> $O_1$: Propor um modelo de medição composto de medidas de qualidade associadas a características de qualidade, aplicável a modelos de processo de software;
>
> $O_2$: Propor um processo para avaliação de modelos de processo de software utilizando o modelo de medição definido;
>
> $O_3$: Avaliar a viabilidade e resultados da aplicação do processo de avaliação e modelo de medição definidos, considerando para isso um processo de software real.

## 1.6 Delimitação do escopo

A Gestão de Processos de Negócio (BPM, na sigla em inglês) tem sido reconhecida como uma área do conhecimento, ou uma abordagem, há pelo menos cerca de três décadas, e

aplicada a diversas áreas do conhecimento (MCCABE, 1987). BPM possui dimensões tais como como descoberta de processos, modelagem de processos, análise de processos, implementação de processos e monitoramento de processos (FREUND e RÜCKER, 2012).

Os modelos de processo tem sido uma ferramenta útil para a área de Engenharia de Software desde a construção dos primeiros sistemas de informação, pelo menos desde o fim da década de 1970 (VAN DER AALST, 2013). Portanto, considerando a necessidade de se realizar medições que possam ser repetidas e comparadas, decidiu-se pela definição de um escopo específico para a o presente trabalho: modelos de processo de software representados em notação BPMN.

No que se refere ao modelo de qualidade a norma ISO/IEC 25.010 foi considerada como referência, que apresenta um conjunto de 8 características principais de qualidade, e mais 31 subcaracterísticas dessas características maiores. Deste modo, em função do elevando número de características e subcaracterísticas, foi necessário delimitar também as características que seriam consideradas neste estudo. Com base nas pesquisas realizadas, inclusive em revisão de literatura, compreendeu-se que as características de Usabilidade e Manutenibilidade são consideras fundamentais para a avaliação dos modelos de processo de software, uma vez que tanto a compreensão desses modelos quanto a garantia de manutenção dos mesmos são fundamentais para seu uso nas organizações.

## 1.7 Contribuições esperadas

A contribuição desse trabalho é a proposição de um processo de avaliação da qualidade para modelos de processo de software pela realização de medições que estejam associadas a um modelo de qualidade de referência.

Para tanto, uma contribuição inicial é a proposição do processo de elaboração de um modelo de medição, relacionado medidas a características e subcaracterísticas de qualidade, e também a construção deste modelo de medição, aplicável a modelos de processo de software.

Outra contribuição é a proposição do processo de avaliação da qualidade de modelos de processo de software e a verificação da viabilidade e dos resultados obtidos pela aplicação deste processo em um modelo de processo de software real.

## 1.8 Metodologia de pesquisa

Lakatos e Marconi (LAKATOS e MARCONI, 2010), bem como Wazlawic (WAZLAWICK, 2014) sugerem metodologias de pesquisa que incluem fases genéricas, tais como:

1. Escolher o tema;
2. Fazer uma revisão da literatura / Levantamento de dados;
3. Definir o problema / Definir os objetivos da pesquisa;
4. Propor uma solução / Propor hipóteses;
5. Analisar viabilidade da solução / Teste de instrumentos e procedimentos;

Entendemos, contudo, que o método de pesquisa é um processo dinâmico, iterativo, que evolui durante toda a pesquisa. Falando de outra maneira, é possível que a partir dos resultados obtidos em determinada fase, se torne necessário voltar à fase anterior para realizar pequenos ajustes. Considerando isso, a Figura 1 explicita de maneira diagramática as principais etapas de um método de pesquisa.

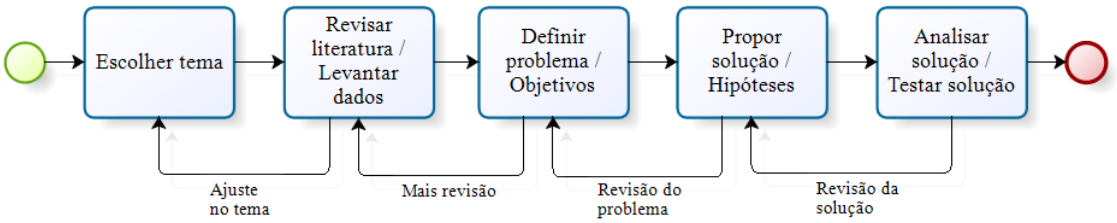



*Figura 1. Modelo do método de pesquisa, a partir de (LAKATOS e MARCONI, 2010) e (WAZLAWICK, 2014).*

O tema, de maneira geral, surgiu a partir da observação de que a definição e manutenção de modelos de processo de software podem representar um desafio para muitas organizações. Por exemplo, os levantamentos de governança de TI do Tribunal de Contas da União (TCU, 2010) (TCU, 2012) indicaram que as organizações do governo federal tinham dificuldades para definir e evoluir seus processos de desenvolvimento de software. Essa percepção foi reforçada por nossas pesquisas iniciais, que indicavam uma ausência de recomendações de qualidade para a construção desse tipo de modelo, bem como uma lacuna em termos de método e medidas para avaliação da qualidade de modelos conceituais do processo de software, ou modelos de processo de software (GARCIA, RUIZ, *et al.*, 2003) (CANFORA, GARCÍA, *et al.*, 2005) (GARCÍA, PIATTINI, *et al.*, 2005).

Em um primeiro momento realizamos uma revisão informal da literatura, de maneira ampla, sobre qualidade em modelos conceituais, o que ajudou a definir melhor o foco do

problema e os objetivos da pesquisa. A partir desse foco, foi realizada uma revisão da literatura de acordo com as recomendações de (KITCHENHAM e CHARTERS, 2007) e de (KITCHENHAM e BRERETON, 2013), definindo o objetivo principal do trabalho como: Propor um processo de avaliação da qualidade para modelos de processo de software, baseado em medições associadas a um modelo de qualidade de referência.

O próximo passo foi a proposição de uma solução, ou seja, das ações necessárias para se atingir o objetivo do trabalho. Neste sentido, a solução proposta foi definida em quatro partes: 1) elaborar um processo para construção de um modelo de medição; 2) construir um modelo de medição com base neste processo; 3) elaborar um processo de avaliação da qualidade de modelos de processo de software; 4) executar o processo de avaliação em um modelo real.

Um processo para elaboração de um modelo de medição para avaliação da qualidade de modelos de processo de software foi proposto. Este processo foi baseado na família de normas ISO/IEC 25.000. Como base para este processo, um modelo de referência para medição da qualidade de modelos de processo de software foi adaptado a partir da ISO/IEC 25.020. Foram apresentadas duas estruturas de informações, uma para as medidas básicas, também chamadas de QME (*Quality Measure Element*) e outra para as medidas derivadas, também chamadas de QM (*Quality Measure*), a serem produzidas pelo processo.

A seguir, este processo foi executado. O primeiro passo foi a definição dos elementos de medida de qualidade, que resultou em um conjunto de QMEs aplicáveis ao modelo de processo de software. Neste passo, as QMEs foram definidas a partir da revisão de literatura e a partir de uma proposta inicial de QMEs da norma ISO/IEC 25.021. Em seguida, foram definidas as medidas de qualidade, resultando em um conjunto de QMs aplicáveis ao modelo de processo de software. Estas QMs também foram obtidas da revisão de literatura e de uma proposta inicial QMs da norma ISO/IEC 25.023. Após isso, foi definido um modelo de medição, explicitando as relações entre características de qualidade, subcaracterísticas de qualidade, QMs e QMEs.

Também foi definido um processo para a avaliação de modelos de processo de software, baseado na norma ISO/IEC 25.040. Em seguida, este processo foi executado para a avaliação do processo de software MDS-Fiocruz, produzindo recomendações de ajustes. Estes ajustes foram implementados, e nova avaliação foi realizada sobre esta MDS-

Fiocruz ajustada. Ao final, foi realizada a análise da solução proposta e dos resultados obtidos.

## 1.9 Estrutura da tese

Além do presente capítulo de introdução, que trata do contexto, motivação, questão de pesquisa, objetivos, delimitações e contribuições esperadas, o restante do trabalho está organizado da seguinte maneira:

**2 - Base conceitual.** Apresenta os conceitos fundamentas referenciados no trabalho, tais como qualidade em modelos conceituais, modelo conceitual do processo, gramáticas para o modelo conceitual de processo de software, ciclo de vida do modelo conceitual de processo de software, modelos de qualidade em Engenharia de Software, e família ISO/IEC 25.000.

**3 - Trabalhos relacionados**. Trata dos trabalhos relacionados nas seguintes dimensões: modelos conceituais, modelos conceituas de processo de negócio e modelos conceituais de processo de software.

**4 - Processo de elaboração de um modelo de medição para avaliação de modelos de processo de software**. Propõe um processo para a elaboração do modelo de medição aplicável a modelos de processo de software.

**5 - Execução do processo de elaboração do modelo de medição.** Executa o processo de construção do modelo de medição, produzindo um modelo de medição a ser utilizado na avaliação de modelos de processo de software.

**6 - Proposta de um processo de avaliação para modelos de processo de software**. Propõe um processo para avaliação de qualidade aplicável a modelos de processo de software.

**0 -**

**Execução do processo de avaliação para modelos de processo de** software**.** Execução do processo de avaliação, utilizando o modelo de medição por meio de uma ferramenta especificamente construída para este fim, aplicado a um modelo real de processo software.

**8 - Conclusão.** Este Capítulo apresenta a conclusão do trabalho.

## 1.10 Conclusão do capítulo

Neste capítulo foi apresentada a importância do software para a Sociedade moderna, e consequentemente a necessidade de qualidade para esse software. A importância dessa qualidade foi demonstrada pelas diversas iniciativas existentes para a melhoria da qualidade do software, que em geral é produzida a partir da melhoria do processo de desenvolvimento desse software, que por sua vez é percebido através de sua representação. Foram destacados dois importantes modelos de maturidade e capacidade para a melhoria do processo de software, o MR-MPS-SW e o CMMI-DEV.

A importância do processo de software e de sua representação para a melhoria da qualidade do software foi apresentado como motivação para este trabalho, considerando os indícios de que há espaço para pesquisas relacionadas à melhoria da representação, ou do modelo do processo de software. Neste sentido, a questão da pesquisa apresentada foi a de que a qualidade do modelo de processo de software pode ser avaliada com base em medições relacionadas a um modelo de qualidade de referência. Sendo assim, o objetivo principal do trabalho foi estabelecido como “propor um processo de avaliação da qualidade para modelos de processo de software, baseado em medições associadas a um modelo de qualidade de referência”.

O escopo do trabalho foi delimitado para modelos de processo de software representados em notação BPMN, considerando as características de “Usabilidade” e “Manutenibilidade”. A partir disso, as contribuições esperadas propostas foram a de definição de um processo de elaboração do modelo de medição e a subsequente execução deste processo, e a definição de um processo de avaliação de modelos de processo de software e sua subsequente execução, aplicado a um processo real. Maiores detalhes sobre a forma de construção dessas contribuições foram apresentados na metodologia de pesquisa.

# 2 Base conceitual

## 2.1 Introdução do capítulo

Há um conjunto de conceitos que precisam ser estabelecidos para a melhor compreensão deste trabalho, tais como o de modelo conceitual, qualidade em modelos conceituais, gramáticas para modelagem, ciclo de vida de um modelo conceitual, e modelos de qualidade em engenharia de software. Estes conceitos serão apresentados neste capítulo.

## 2.2 Qualidade em modelos conceituais

Um modelo conceitual pode ser visto como um conjunto de especificações relevantes para compreender determinado problema (LINDLAND, SINDRE e SOLVBERG, 1994). Por exemplo, uma representação explícita de parte de determinada realidade, que descreva um conjunto de atividades humanas complexas, pode ser chamado de modelo conceitual (KROGSTIE, SINDER e JORGENSEN, 2006). Sendo assim, modelos conceituais são aplicados a diversas áreas do conhecimento, tais como modelagem de dados, documentação e melhoria de processos, desenvolvimento de software, gestão de workflow (fluxo de processos), gestão do conhecimento, entre outras. Como resultado, podem ser produzidos artefatos diversos. Para (DAVIES, GREEN, *et al.*, 2006), os seis principais modelos conceituais produzidos são: Entidade e Relacionamento, Fluxo de Dados, Fluxograma, Fluxo de Trabalho (*processos*), Diagramas UML e Diagramas Estruturados (*Structured Charts*).

Desde os anos 1960 diversos trabalhos foram publicados sobre modelos conceituais. Na década de 1990 houve um interesse renovado em relação aos modelos conceituais por conta de determinados eventos, como o surgimento da abordagem de orientação a objetos, a necessidade de se elicitar requisitos de sistemas mais complexos, a necessidade crescente de se modelar dados e processos, e a reutilização de componentes de software (WAND e WEBER, 2002).

(LINDLAND, SINDRE e SOLVBERG, 1994) analisaram os seguintes modelos de avaliação da qualidade de modelos conceituais daquela época: o de David Kung (KUNG, 1983), o de Gruia-Catalin Roman (ROMAN, 1985), o de Raymond Yeh (YEH, ZAVE, *et al.*, 1984) e o de Alan Davis (DAVIS, 1990). A partir desta análise, eles concluíram que os modelos de avaliação apresentavam limitações, tais como definições vagas, complicadas e insuficientes, listas não estruturadas de propriedades, com especificações confusas e propriedades incompatíveis ou conflitantes, entre outras limitações. Então, os

autores propuseram um novo modelo para avaliação da qualidade de modelos conceituais. Porém, embora esta proposta tenha se constituído em um modelo de qualidade com a definição de características de qualidade, ela não preconizava um processo de avaliação.

Segundo os autores, essas propostas eram meras listas de requisitos desejados ou propostos, mas de maneira muito vaga, complicada, desestruturada, confusa, irrealística e muitas vezes inexequível. O novo modelo de avaliação proposto por estes autores se baseava nas seguintes perspectivas:

- **Sintaxe**. Relaciona o modelo à linguagem de modelagem, apresentando o relacionamento entre seus construtos.
- **Semântica**. Relaciona o modelo ao domínio a partir do qual o modelo é construído, considerando não apenas a sintaxe, mas também a relação entre as declarações (construtos) e seus significados para aquele domínio.
- **Pragmatismo**. Relaciona o modelo ao público alvo deste modelo, por considerar não apenas a sintaxe e a semântica, mas também como esse público irá interpretar o modelo.

Segundo os autores, essas três dimensões deveriam ser avaliadas em quatro conjuntos:

- **Modelo**. O conjunto de todas as declarações explícitas ou explicitas que compõe o modelo;
- **Linguagem**. O conjunto de declarações disponíveis para composição do modelo, de acordo com o vocabulário ou gramática da linguagem de modelagem utilizada;
- **Domínio**. O conjunto de declarações que seriam corretas e relevantes sobre o problema em questão, a ser modelado;
- **Audiência**. O conjunto de declarações que representa a compreensão da audiência (público alvo) sobre o modelo.

Uma representação desse conceito de modelo de qualidade para modelos conceituais é proposta por (KROGSTIE, LINDLAND e SINDRE, 1995), conforme Figura 2.

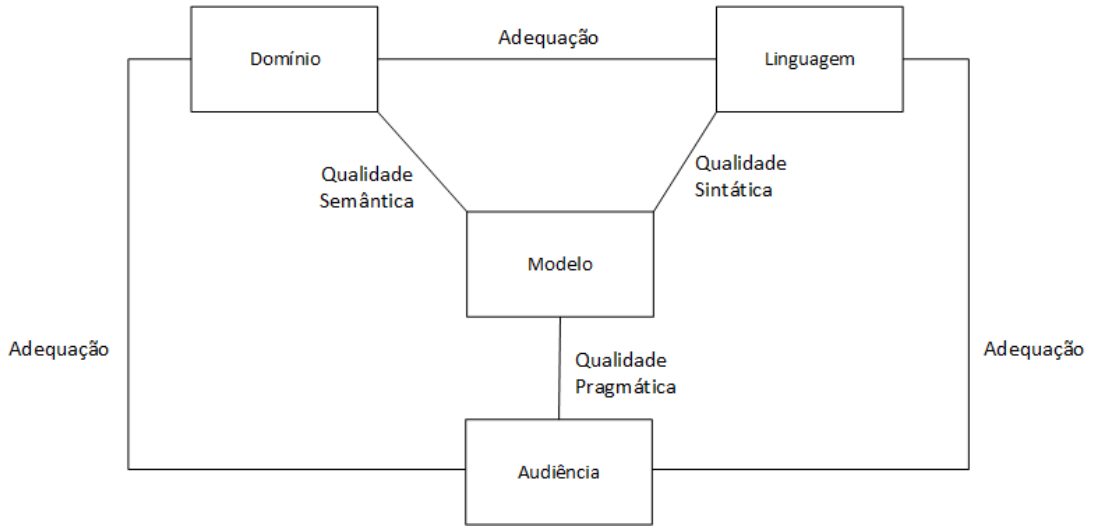



*Figura 2. Modelo de qualidade para modelos conceituais, conforme (KROGSTIE, LINDLAND e SINDRE, 1995).*

Mais tarde, (WAND e WEBER, 2002) concordaram que os modelos conceituais são importantes porque facilitam a detecção e a correção dos erros de maneira antecipada, em um ponto onde os custos de correção são exponencialmente menores. Apesar disso, eles não propuseram um modelo de medição ou um processo de avaliação. Segundo esses mesmos autores, um modelo conceitual pode ser representado de acordo com a estrutura apresentada na Figura 3. Segundo essa estrutura há quatro elementos fundamentais em um modelo conceitual:

- **Gramática**. Conjunto de construtos e regras para a combinação desses construtos de maneira a viabilizar a modelagem de um determinado domínio do mundo real;
- **Método**. Procedimentos por meio dos quais a gramática pode ser utilizada, prescrevendo como mapear a observação do mundo real em um modelo desse mundo real;
- **Script**. É um produto da aplicação do método sobre a gramática, ou seja, o resultado dessa aplicação. Em outras palavras, o Script é o próprio diagrama que representa determinada realidade;
- **Contexto**. É o ambiente no qual a modelagem conceitual ocorre. Pode ser observado sob três dimensões: a) Fatores de diferenças individuais, ou seja, a experiência dos envolvidos, o treinamento e a capacidade cognitiva dos mesmos; b) Fatores de tarefas, ou seja, a aplicação da gramática e scripts para diferentes tarefas relacionadas ao desenvolvimento de sistemas; c) Fatores de agenda social, ou seja, gramática e scripts utilizados em um contexto mais amplo que envolve

mudança organizacional. De maneira geral, o contexto pode afetar o uso da gramática e a geração do script, sua aplicação e sua compreensão.

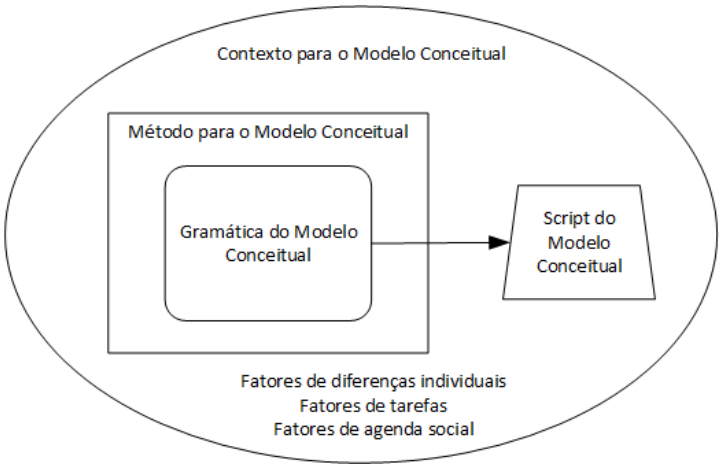



*Figura 3. Estrutura de um modelo conceitual, segundo (WAND e WEBER, 2002).*

No que se refere ao Script do modelo conceitual, ou seja, ao próprio produto da modelagem, ou ao diagrama resultante da modelagem, (WAND e WEBER, 2002) afirmaram que as pesquisas nessa área são no sentido de determinar como produzir o melhor Script por meio de determinada gramática. Nessa linha, os autores citam a proposta de (LINDLAND, SINDRE e SOLVBERG, 1994) para avaliação da qualidade do modelo conceitual, com vistas a produzir o melhor modelo conceitual possível.

Posteriormente, o modelo de (WAND e WEBER, 2002) recebeu evoluções do ponto de vista da semiótica, com outras perspectivas de qualidade, como qualidade semântica percebida, qualidade social, qualidade física, qualidade do conhecimento, qualidade empírica, e qualidade ferramental, além das qualidades sintática, semântica e pragmática previstas no modelo semiótico original (KROGSTIE, SINDER e JORGENSEN, 2006). Pesquisas foram realizadas no sentido de avaliar a qualidade de alguns desses elementos específicos. Por exemplo, (MAES e POELS, 2007) propuseram um framework para avaliação da qualidade de um modelo conceitual a partir da perspectiva do usuário, sugerindo dimensões da qualidade tais como facilidade de uso, compreensão do modelo, usabilidade percebida e satisfação do usuário com o modelo, sempre sob a perspectiva do Script, ou seja, do modelo construído (WAND e WEBER, 2002). Já o trabalho de (GENERO, POELS e PIATTINI, 2008) propôs um conjunto de medidas para avaliar a compreensibilidade de um modelo conceitual específico, o Diagrama de Entidades e Relacionamentos (DER). Outro framework de avaliação da qualidade de modelos conceituais foi proposto por (NELSON, POELS, *et al.*, 2012) para avaliar a qualidade de

modelos conceituais a partir de oito características de qualidade – sintática, semântica, intencional, significância, empírica, de domínio, de conhecimento e de linguagem.

Como apresentado, os trabalhos iniciais sobre a construção modelos conceituais e sobre análise de qualidade desses modelos aplicavam-se a modelos conceituais gerais (LINDLAND, SINDRE e SOLVBERG, 1994). Estes trabalhos se tornaram referência para a aplicação de conceitos de qualidade em modelos conceituais para domínios específicos.

De fato, a concepção de "modelo conceitual" foi aplicada a diversos domínios. Por exemplo, considerando apenas a área de software é possível citar modelos conceituais como os de Entidade e Relacionamentos, Fluxo de Dados, e Fluxo de Trabalho (ou Processos) (DAVIES, GREEN, *et al.*, 2006).

## 2.3 Modelo conceitual de processos

(MENDLING, REIJERS e VAN DER AALST, 2010) trataram do modelo conceitual de processos, e destacaram que muitos trabalhos relacionados à qualidade da representação do processo têm sido conduzidos, mas que estruturas como a apresentada por (LINDLAND, SINDRE e SOLVBERG, 1994) tendem a ser muito genéricas para serem aplicadas, de fato, ao passo que outras propostas têm sido apenas um conjunto de recomendações pragmáticas sem a devida estrutura científica.

(GUIZZARDI, FALBO e GUIZZARDI, 2008) trataram do modelo conceitual do processo de software, e afirmaram que este processo é composto de atividades e de outros processos, ou seja, de subprocessos. Cada atividade se refere a uma certa medida de trabalho a ser realizado a fim de produzir um determinado resultado. Quando essas atividades são realizadas, recursos são consumidos, e informações são utilizadas e produzidas por elas.

Portanto, o modelo do processo de software é o resultado do esforço em explicitar esse processo, que é composto de um conjunto de ferramentas, métodos e práticas (YU e MYLOPOULOS, 1994). Os processos de software são complexos em função das muitas atividades envolvidas, dos diversos atores que realizaram as atividades e das diversas informações que são produzidas e consumidas ao longo de todo o processo de produção do software. Por conta dessa complexidade, é necessário estabelecer um entendimento comum do processo de software e isto é feito pela construção de um modelo conceitual que o represente.

(ANDRADE, ARES, *et al.*, 2004) destacaram que a construção do modelo conceitual é fundamental, especialmente por se tratar de um processo com múltiplos atores. Complementarmente (BERKI, GEROGIADOU e HOLCOMBE, 2004) afirmaram que o processo de software é uma abordagem disciplinada para o desenvolvimento de software, com o objetivo de garantir a qualidade desse software. Esse processo ajuda aos desenvolvedores a resolver problemas complexos, aumenta a confiabilidade da produção de software, e contribui para o atingimento dos objetivos dentro de prazos razoáveis.

A qualidade do modelo conceitual do processo de software é um tema relevante, e a complexidade desse processo tem motivado pesquisas no sentido de avaliar a qualidade desses modelos. Estas pesquisas destacam a importância desses modelos e da necessidade de precisão, boa definição semântica e de mecanismos de medição e avaliação da qualidade (GARCIA, RUIZ, *et al.*, 2003) (GARCÍA, RUIZ e PIATTINI, 2004) (DA SILVA, MACIEL e RAMALHO, 2013). Para (CANFORA, GARCÍA, *et al.*, 2005) o modelo do processo de software é um importante artefato para a qualidade do processo de software, pois é o ponto inicial para a análise, melhoria e execução desse processo. Os autores destacam a necessidade de que este modelo seja compreensível por humanos e manutenível. Com o objetivo de avaliar e promover a qualidade do modelo conceitual do processo de software, outros estudos foram conduzidos no sentido de se estabelecer medidas e também formas de avaliação da qualidade (GARCÍA, RUIZ e PIATTINI, 2004b) (GARCÍA, PIATTINI, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006).

## 2.4 Gramáticas para modelo conceitual de processos de software

Como visto, para (KROGSTIE, LINDLAND e SINDRE, 1995) uma ‘linguagem é um conjunto de declarações disponíveis para composição do modelo’. Mas (WAND e WEBER, 2002) ampliam este conceito para o de um ‘conjunto de construtos e regras para a combinação desses construtos de maneira a viabilizar a modelagem de um determinado domínio do mundo real’, e chamam a este conceito de “gramática”. Considerando assim, as gramáticas, ou notações, têm sido parte do desenvolvimento de software desde o início. Em determinadas fases do projeto o conhecimento tácito precisa ser explicitado de forma estruturada, de maneira que suporte sua interpretação posterior pelo próprio indivíduo que fez o registro, por outros envolvidos no projeto, ou por uma ferramenta computacional (TAYLOR, 2003).

Uma das linguagens utilizadas para a construção de modelos conceituais do processo de software é a linguagem natural. Há importantes diferenças entre as linguagens naturais e

as linguagens visuais, ou diagramáticas. Por exemplo, (MOODY, 2009) entende que as linguagens visuais, que produzem diagramas, são mais concisas e precisas. Ainda, na opinião do mesmo autor, as informações registradas nesses diagramas são mais fáceis de serem lembradas. Ele destaca ainda as diferenças apresentadas na Tabela 1.

*Tabela 1. Diferenças entre linguagem textual e linguagem visual, segundo (MOODY, 2009).*

| **Linguagem textual** | **Linguagem visual** |
|---|---|
| Codifica informações usando sequências de letras | Codifica informações usando estruturas espaciais com elementos gráficos e textuais |
| São unidimensionais e lineares | São bidimensionais e espaciais |
| É processada de maneira serial pela mente humana (uma informação de cada vez) | É processada em paralelo pela mente humana (mais de uma informação de cada vez) |

De fato, se estamos falando de modelos conceituais que precisam ser compreendidos por humanos e que se constituam em artefatos fundamentais para a melhoria da qualidade do processo de software, é preciso considerar a gramática mais adequada para este fim, e isso se relaciona com a capacidade da mente humana em processar essa gramática. Essa era uma preocupação já na década de 1950, quando Miller (1956) publicou o artigo seminal "*The magical number seven, plus or minus two: some limits on our capacity for processing information*" (O mágico número sete, mais ou menos dois: alguns limites em nossa capacidade de processar informação [*tradução do autor*]). Nesse trabalho Miller destaca que a mente humana consegue realizar processamento paralelo, porém limitado. Segundo os estudos conduzidos por Miller, a mente humana consegue processar algo entre 5 e 9 elementos principais de cada vez (7±2).

A partir do trabalho de Miller, muitos outros foram realizados, e mais tarde estruturados por Sweller (2004) em um conjunto de princípios que passou a ser conhecido como Teoria da Carga Cognitiva. Alguns desses princípios se relacionam diretamente com nosso trabalho, como os seguintes:

- **Princípio da representação múltipla**. O indivíduo compreende melhor quando se combinam imagens e palavras do que quando se utilizam apenas palavras;
- **Princípio da proximidade espacial**. Imagens e palavras correspondentes ou relacionadas devem estar próximas entre si;
- **Princípio da proximidade temporal.** A apresentação de imagens e palavras devem ser feitas de maneira simultânea (paralela) e não sucessiva (serial);

- **Princípio das diferenças individuais.** Os indivíduos possuem diferentes graus de orientação espacial, e consequentemente diferentes capacidades de processar seu próprio conhecimento ao interagir com o objeto em questão;
- **Princípio da coerência.** Deve-se excluir todos os elementos (sons, palavras e imagens) que não sejam essenciais para o objeto em questão. Elementos desnecessários sobrecarregarão a capacidade cognitiva do indivíduo sem adicionar nenhum benefício para a compreensão do objeto em questão.

Entendemos, a partir dos princípios da representação múltipla, da proximidade espacial e da proximidade temporal, que as linguagens visuais tendem a representar uma gramática mais adequada para a representação do modelo conceitual do processo de software do que as linguagens estritamente textuais. Adicionalmente, (FANTECHI, GNESI, *et al.*, 2002) ao trabalharem com a análise de linguagem natural, concluíram que apesar de flexível e expressiva, a linguagem natural tende para a ambiguidade, redundância, omissões e outros defeitos que podem representar obstáculos à precisão e clareza.

Em contrapartida, durante a nossa pesquisa encontramos publicações abordando algumas gramáticas visuais (ou diagramáticas) para a modelagem de processos aplicáveis à representação do processo de software, tais como YAWL e Petri Nets (VAN DER AALST e TER HOFSTEDE, 2005). A partir disso, realizamos uma primeira pesquisa em bases de dados para artigos científicos, e a partir de palavras de busca relacionadas aos nomes das notações (gramáticas) utilizadas para a modelagem de processos que encontramos durante nossas pesquisas. Obtivemos os seguintes resultados conforme Tabela 2.

*Tabela 2. Primeira pesquisa de notações em bases de artigos.*

| Palavra chave | Scopus | ACM Digital Library | IEEE Xplore | Web of Knowledge | Média |
|---|---|---|---|---|---|
| "Petri Nets" | 18.155 | 3.304 | 8.372 | 10.162 | 9.998 |
| BPMN | 888 | 552 | 232 | 488 | 540 |
| SPEM | 617 | 101 | 81 | 534 | 333 |
| IDEF | 268 | 118 | 77 | 22 | 121 |
| YAWL | 107 | 117 | 16 | 110 | 88 |
| Little-JIL | 22 | 65 | 7 | 17 | 28 |
| "Aris EPC" | 2 | 4 | 0 | 1 | 2 |

Notamos uma grande diferença da busca pelas palavras chave “Petri Nets” em relação às demais palavras chave, o que nos levou a pesquisar o assunto em mais detalhes.

Entendemos que a notação Petri Nets é uma ferramenta gráfica de visualização, mas também uma ferramenta de modelagem matemática, aplicada em diversas áreas do conhecimento e a diversos tipos de sistema. Uma das linhas de aplicação, no entanto, seria a modelagem de workflows de trabalho, ou processos de trabalho (MURATA, 1989). Então realizamos a pesquisa novamente, ajustando o termo de busca de “Petri nets” para “Petri nets workflow”, produzindo o resultado apresentado na Tabela 3.

*Tabela 3. Segunda pesquisa de notações em bases de artigos.*

| Palavra chave | Scopus | ACM Digital Library | IEEE Xplore | Web of Knowledge | Média |
|---|---|---|---|---|---|
| BPMN | 888 | 552 | 232 | 488 | 540 |
| SPEM | 617 | 101 | 81 | 534 | 333 |
| IDEF | 268 | 118 | 77 | 22 | 121 |
| YAWL | 107 | 117 | 16 | 110 | 88 |
| "Petri Nets Workflow" | 179 | 6 | 3 | 2 | 48 |
| Little-JIL | 22 | 65 | 7 | 17 | 28 |
| "Aris EPC" | 2 | 4 | 0 | 1 | 2 |

Considerando que estas foram as notações identificadas em utilização para a modelagem de processos, segue-se uma visão geral de cada uma dessas notações.

### 2.4.1 BPMN

A BPMN (*Business Process Modeling and Notation*) é uma notação aberta, cujo objetivo é ser de fácil compreensão por todos os profissionais da área de negócios, desde o analista de negócios, passando pelos desenvolvedores técnicos que implementam as tecnologias de suporte aos processos, até as pessoas que gerenciam e monitoram esses processos (OMG, 2013). A notação foi proposta pelo *Business Process Management Initiative* (BPMI) em 2002, e passou a contar com o apoio da *Workflow Management Coalition* (WfMC), que inclusive contribuiu com uma linguagem formal para a notação, a *XML Process Definition Language* (XPDL) (SWENSON e SHAPIRO, 2008). Atualmente, a BPMN é mantida pela *Object Management Group* (OMG), uma organização sem fins lucrativos que mantém padrões da indústria de computação e que conta com o apoio de fabricantes de software, usuários finais, agências de governo e da academia (OMG, 2013), e é considerada o estado da arte em termos de notação para a modelagem de processos (CHINOSI e TROMBETTA, 2011). A objetividade dessa notação, bem como a clareza e facilidade de uso (CAMPOS e OLIVERIA, 2011) são, provavelmente, as características

que tornaram a notação BPMN a mais amplamente utilizada para a modelagem de processos de trabalho (WIL M. P., 2011).

### 2.4.2 SPEM

A notação SPEM (*Software & Systems Process Engineering Metamodel Specification*) foi criada com o objetivo de se tornar o padrão para modelagem de processos de software, que seria a base para o desenvolvimento de software orientado a objetos, e adotado pela OMG (*Object Managent Group*) a partir de 2002 (GONZALES-PEREZ e HENDERSON-SELLERS, 2007) (HENDERSON-SELLERS e GONZALEZ-PEREZ, 2004). Trata-se tanto de um metamodelo quanto de um *profile* da linguagem UML (*Unified Modeling Language*), que em sua versão 1.1 estava focado apenas na representação do processo, mas que na versão seguinte, a 2.0, passou a se preocupar também com a execução desse processo.

### 2.4.3 IDEF

Quando se fala em notação IDEF para processos, estamos falando na verdade das notações IDEF0 (*Integration Definition 0)*, uma vez que IDEF é um conjunto de notações, cada uma para um fim específico. A notação IDEF0 é utilizada para a modelagem de processos em um nível mais alto, identificando as entradas e saídas, bem como os recursos necessários ao processo e os mecanismos de controle dos mesmos (CARNAGHAN, 2006). Esta notação permite representar atividades, fluxos, recursos de suporte, e controles que atuam sobre a atividade. O método é composto basicamente por setas e caixas, sendo a posição da caixa onde a seta se conecta indicativa do seu tipo, ou seja: entrada, saída, controle ou recurso (SOUNG-HIE e KI-JIN, 2000).

### 2.4.4 YAWL

A notação, ou linguagem YAWL (*yet another workflow language*) foi desenvolvida a partir de um conjunto de *patterns* identificados em processos modelados anteriormente, principalmente em notação Petri Nets. O objetivo desta nova notação seria facilitar a modelagem de processos em alto nível resolvendo problemas que teoricamente existiam nos sistemas de gestão de workflow anteriores. Contudo, apesar de ser uma linguagem poderosa para a modelagem de processos, a YAWL é uma notação para uso científico, e não de uso amplo ou comercial (VAN DER AALST e TER HOFSTEDE, 2005).

### 2.4.5 Petri Nets

Como já mencionado, a notação Petri Nets é aplicada em uma diversidade de áreas, inclusive na área de modelagem de processos. Provavelmente o autor mais proficiente na publicação de artigos sobre Petri Nets para workflow é W. M. P. van der Aalst. No entanto, esse mesmo autor reconhece limitações da notação Petri Nets como notação para workflow, ou processos. Entre as questões mais destacadas estão a baixa expressividade na notação e o elevado nível de esforço (trabalho) para produzir resultados comparáveis aos obtidos pelas demais notações para modelagens de processos (VAN DER AALST e TER HOFSTEDE, 2005).

### 2.4.6 Little-JIL

A Little-JIL é uma notação gráfica para a definição de processos que coordena as atividades de agentes autônomos e seus recursos durante a execução de uma tarefa. Programas Little-JIL são executáveis que coordenam agentes, garantindo que as ações desses agentes sejam aderentes ao processo (CASS, LERNER, *et al.*, 2000). Contudo, apesar de possuir uma apresentação gráfica, a Little-JIL aproxima-se mais de uma notação para a abordagem de agentes autônomos (JENNINGS, 2000) e, portanto, mais próxima da camada de execução. Assim, uma possibilidade seria modelar o processo para efeito de representação em uma notação como BPMN ou SPEM, e fazer a transição da Representação para a Execução pela notação Little-JIL (FERREIRA, MACHADO e PAULK, 2011).

### 2.4.7 ARIS

A notação Aris (*ARchitecture for integrated Information Systems*) EPC consiste de uma árvore de elementos incluindo funções (atividades), eventos e conectores, ligados por setas que indicam o fluxo temporal da execução. Isso possibilita descrever atividades, ou processos (GRUHN e LAUE, 2007). No entanto, para (MEYER, SPERNER, *et al.*, 2011) a notação apresenta alguns pontos de preocupação, tais como dificuldade para modelar processos que terão execução distribuída, impossibilidade de associar objetos de dados a entidades específicas, problemas com escalabilidade, problemas para modelar disponibilidade e mobilidade nos processos, ausência de mecanismos para modelar tolerância a falhas, e ausência de objetos que permitam a modelagem de tempo real.

Por meio desta notação é possível descrever informações da organização e sua arquitetura de integração por meio de seus processos, e diferentes visões organizacionais podem ser

descritas. Para a visualização do fluxo de trabalho são utilizados os diagramas de *Event-Driven Process Chain* (cadeia de processos orientada por eventos), ou simplesmente EPCs. Além da descrição das atividades e seu fluxo, é possível identificar papéis ou perfis, e informações produzidas e consumidas no processo (SANTOS JR, ALMEIDA e GUIZZARDI, 2010).

## 2.5 Modelos da qualidade em Engenharia de Software

A própria Engenharia de Software foi criada com o objetivo de aplicar os conceitos de engenharia ao processo de produção de software, com vistas à construção de um produto com maior qualidade, de maneira sistemática, mais rigorosa, mensurável, obedecendo a prazos e custos, e atendendo ao que foi especificado (NATO, 1968). Ou seja, a construção de software, desde então, era reconhecidamente uma atividade de alta complexidade por gerar um produto com elevada possibilidade de mudanças ao longo do seu ciclo de vida, e que é um produto abstrato por natureza, de difícil percepção concreta pela maioria de seus usuários (BROOKS, 1987).

Um primeiro modelo de qualidade para o produto de software foi apresentado por (MCCALL, RICHARDS e WALTERS, 1977), que já apresentava um modelo com três características principais (*Revision*, *Operations* e *Transitions*), sendo que cada uma dessas características continha suas subcaracterísticas, conforme apresentado na Figura 4.

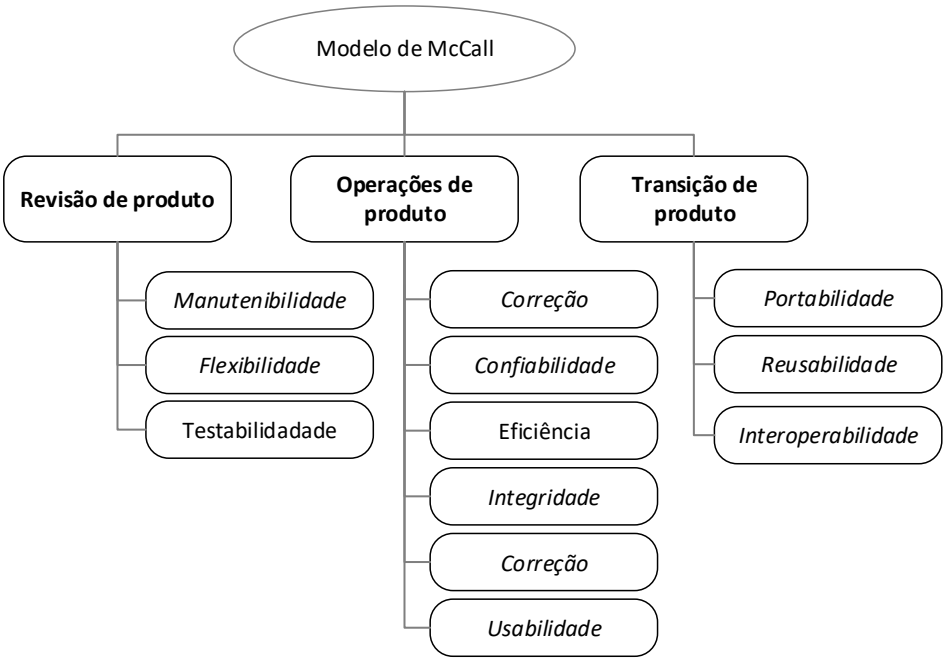



*Figura 4. Modelo de McCall, adaptado de (MCCALL, RICHARDS e WALTERS, 1977).*

Outro modelo de qualidade do produto de software foi proposto por (BOEHM, BROWN e LIPOW, 1976). Este modelo apresenta também três grandes características (*Portability*, *Usability* e *Maintainability*), sendo que essas características se subdividem em mais dois níveis, como pode ser observado na Figura 5. No modelo de Boehm, como é conhecido, já surgem características que teoricamente poderiam ser medidas, que são aquelas na extrema direita da Figura 5, tais como Independência de dispositivo, auto contível, e acurabilidade, entre outros.

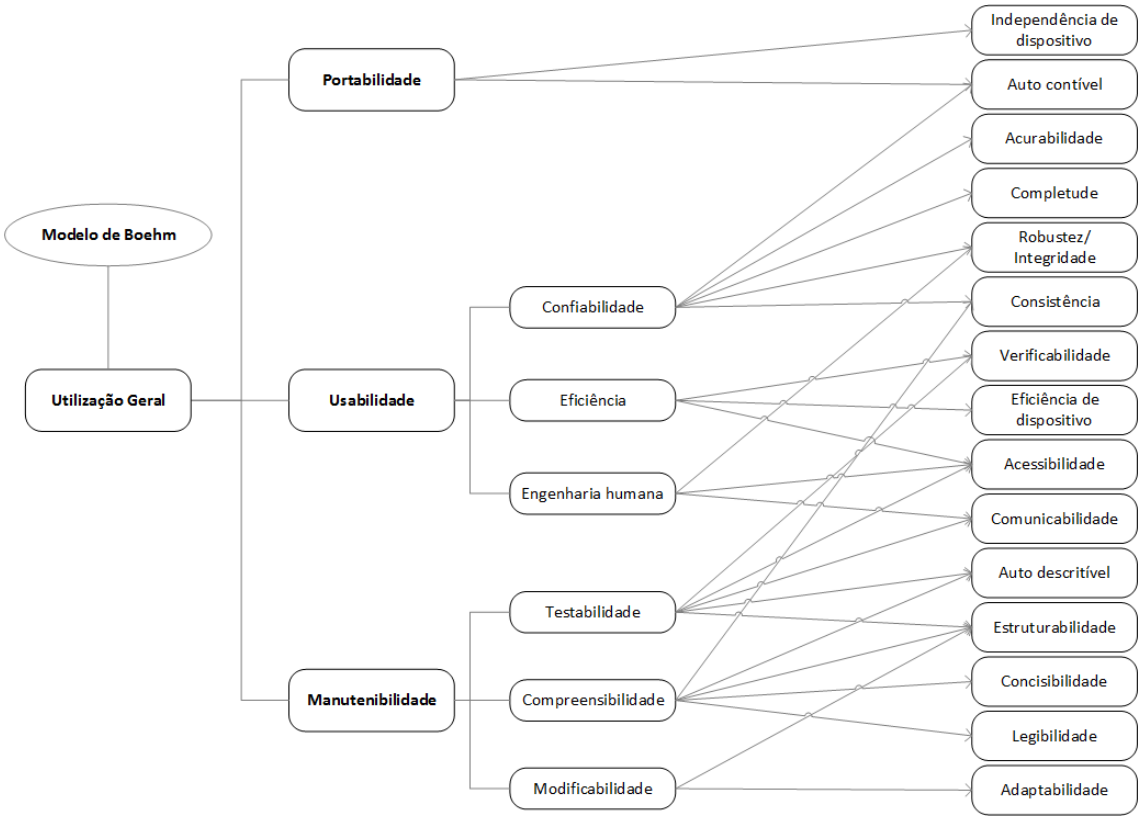



*Figura 5. Modelo de Boehm, adaptado de (BOEHM, BROWN e LIPOW, 1976).*

Em função da complexidade do processo de construção de software, e do desafio crescente da Engenharia de Software, estruturas foram criadas para lidar com esse tipo complexidade, como a criação do *Software Engineering Institute* em 1984 que tinha como objetivo compreender, controlar, medir e melhorar o processo de software (HUMPHREY, 1989), e o subcomitê da ISO/IEC, o JTC 1/SC 7, criado especificamente para tratar de Engenharia de Software. Em 2000 esse subcomitê foi renomeado de "*Software Engineering*" para "*Systems Engineering*". O objetivo desse subcomitê é a

padronização de processos, ferramentas e tecnologias para produtos da Engenharia de Software e de Sistemas (COALLIER e AZUMA, 2013).

Naturalmente, a comunidade de desenvolvimento de software buscava um modelo de qualidade que fosse um consenso na área. Nesta busca a ISO/IEC, com seu subcomitê de Engenharia de Software, propôs em 1991 a norma ISO 9.126, chamado "*Software Product Evaluation: Quality Characteristics and Guidelines for their Use*" (Avaliação do Produto de Software: Características de Qualidade e Orientações para seu Uso) (SINGH e KANNOJIA, 2013). Mais tarde, entre 2001 e 2004, a ISO publicou uma versão estendida da ISO/IEC 9.126, contendo quatro partes, incluindo uma parte para o modelo de qualidade e outras 3 partes com medidas para esse modelo, dividindo essas medidas em internas, externas e de qualidade em uso (AL-QUTAISH, 2009). As características e subcaracterísticas de qualidade da ISO 9.126 são apresentadas na Figura 6.

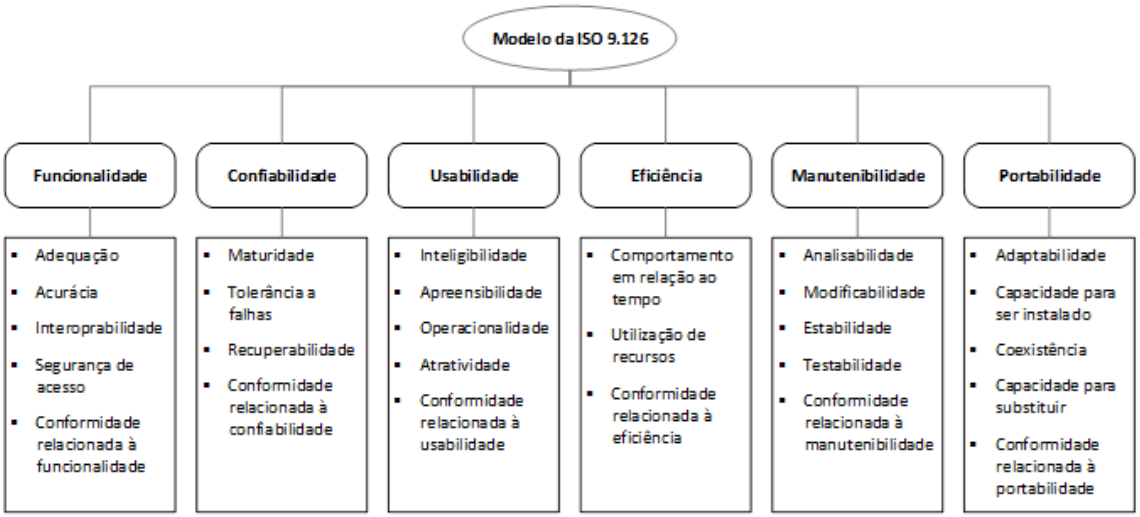



*Figura 6. Modelo da qualidade ISO 9.126, conforme (ISO/IEC, 2003).*

## 2.6 Família ISO/IEC 25.000 (SQuaRE)

Compreendeu-se mais tarde que esta versão estendida não era consistente com a terminologia da ciência de metrologia e engenharia, e apresentava conflito com outros padrões já estabelecidos pela própria ISO, como era o caso da ISO/IEC 15.939. Por isso, em 2005 a ISO passou a elaborar uma série ainda mais extensiva de normas, e este novo conjunto passou a ser conhecido como família ISO/IEC 25.000, ou SQuaRE (*Software Product Quality Requirements and Evaluation*) (SINGH e KANNOJIA, 2013).

### 2.6.1 Divisão ISO/IEC 25.00n

A família ISO/IEC 25.000 está organizada conforme apresentado na Figura 7. As normas são agrupadas por assunto de acordo com sua numeração, contendo as seguintes divisões: gestão da qualidade, modelo de qualidade, requisitos de qualidade, medição de qualidade e avaliação da qualidade.

As normas que compõem a divisão 2500n são a ISO/IEC 25.000, que provê uma estrutura geral para o modelo, a terminologia, e uma visão geral dos documentos, e a ISO/IEC 25.001, que apresenta os requisitos e orientações para quem faz a implementação e gestão da especificação e avaliação da qualidade utilizando esta família de normas (ISO/IEC, 2014).

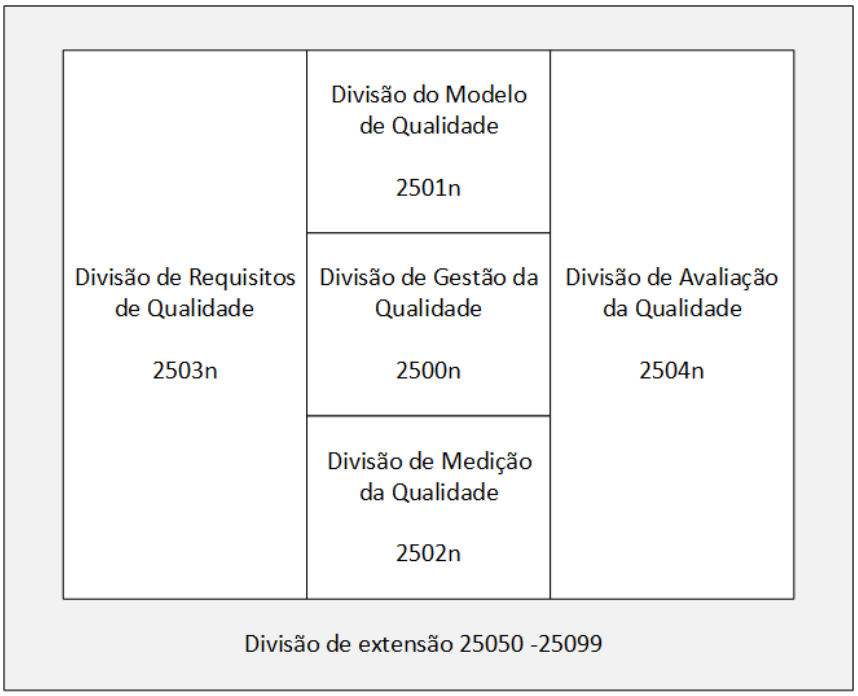



*Figura 7. Estrutura da família ISO 25.000, adaptado de (ISO/IEC, 2014).*

As normas internacionais ISO/IEC que compõe a família 25.000 estão apresentadas na Figura 14. Constam desta figura todas as normas atualmente publicadas desta família de normas, bem como suas interrelações e suas relações com o produto de software, sistema de informação e sistema de negócio.

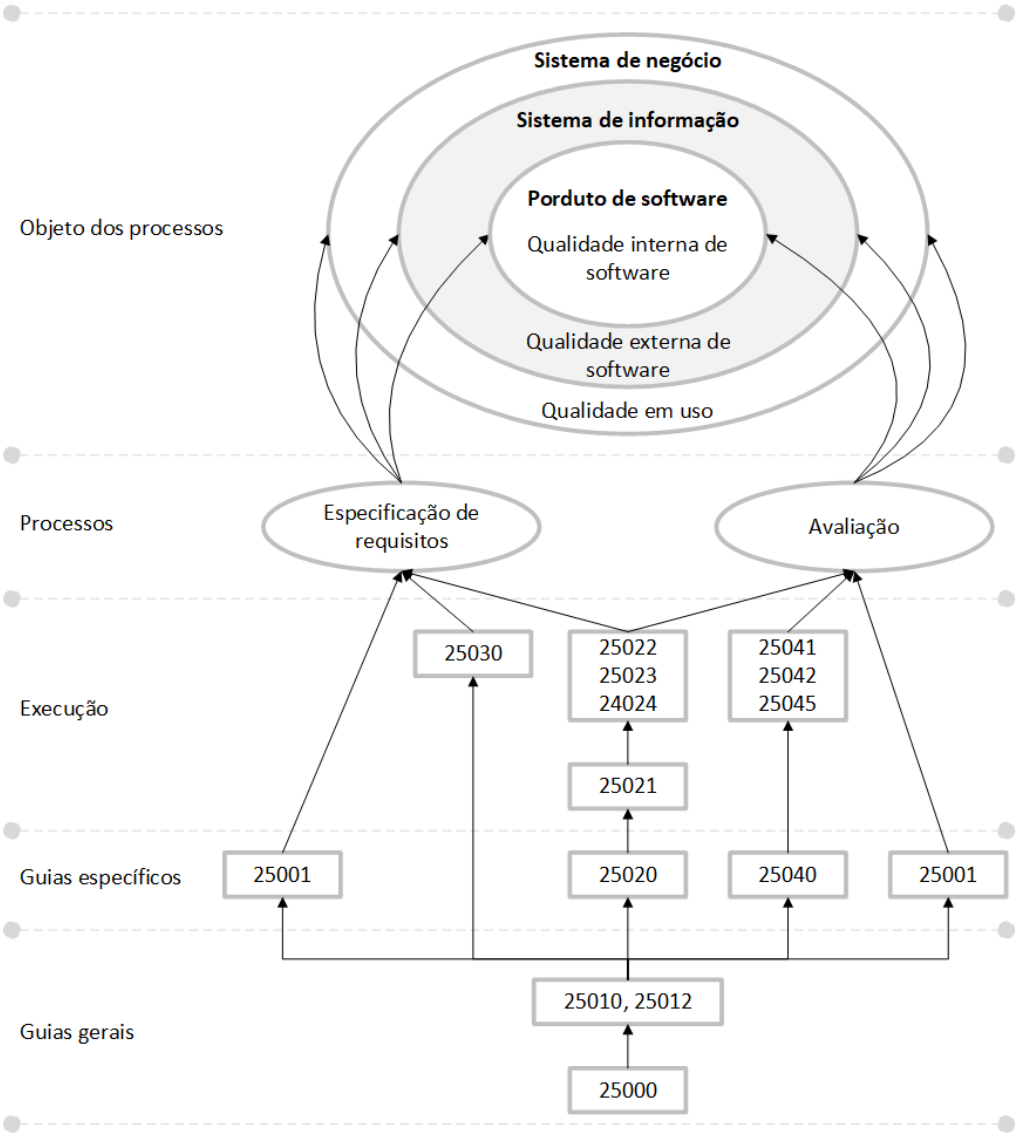



*Figura 8. Referência geral da família ISO/IEC 25.000, adaptado de (ISO/IEC, 2014).*

### 2.6.2 Divisão ISO/IEC 25.01n

A divisão 2501n apresenta os modelos de qualidade, com suas características e subcaracterísticas (ISO/IEC, 2011). A ISO/IEC 25.010 descreve o modelo de qualidade em uso e o modelo de qualidade de produto, com 8 características, como pode ser visto na Figura 9. As características definidas estabelecem uma terminologia consistente para especificação, medição e avaliação da qualidade. A ISO/IEC 25.012, de maneira similar, estabelece um modelo de qualidade para avaliação da qualidade de dados.

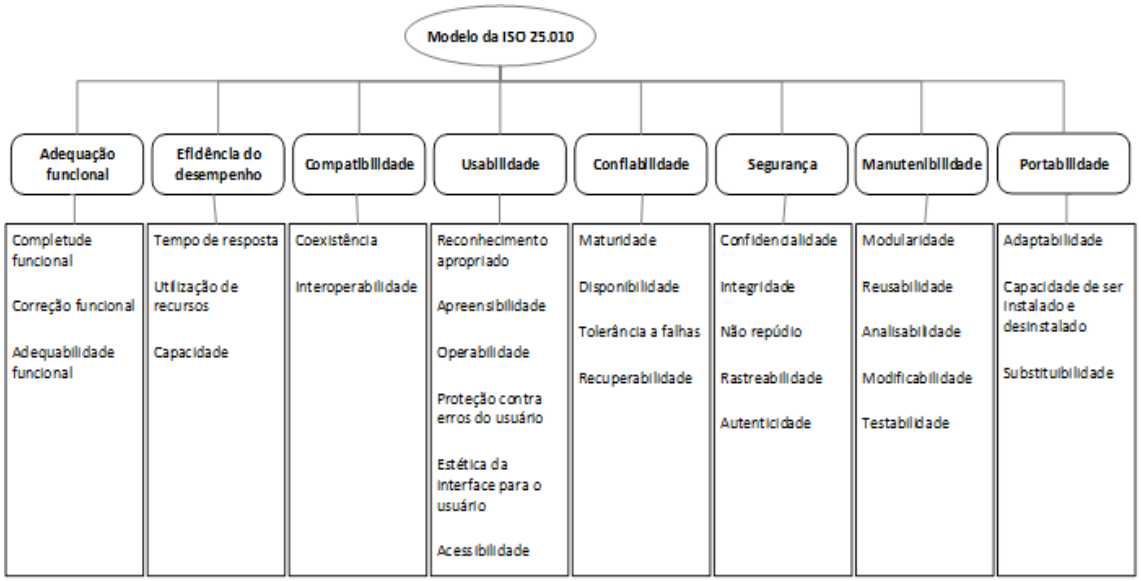



*Figura 9. Modelo da Qualidade da ISO 25.010, adaptado de (ISO/IEC, 2011).*

### 2.6.3 Divisão ISO/IEC 25.02n

A divisão 25.02n, por sua vez, contém o conjunto de normas que tratam da medição da qualidade, conforme pode ser observado na Figura 10. Dentro da divisão 2502n, há a norma ISO 25.021 que trata dos Elementos de Medição da Qualidade – EMQ (*Quality Measure Elements - QME*). O termo QME, em inglês, é na verdade uma adaptação realizada pela ISO/IEC 25.021 a partir do *International Vocabluary of Basic and Gerneral Terms in Metrology* (Vocabulário Internacional de Termos Básicos e Gerais em Metrologia), tratando de dois níveis de medição: a básica e a derivada.

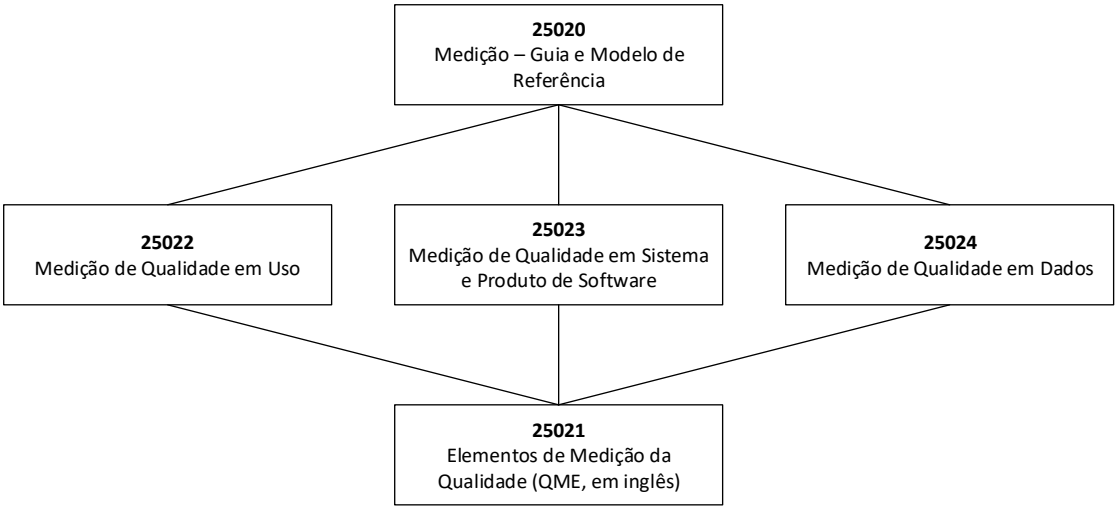



*Figura 10. Divisão ISO 2502n.*

A medição básica se refere à medição mais elementar sobre o objeto em análise, ou seja, se refere a um atributo e a um método para quantificar esse atributo. Esse tipo de medida é tratado na divisão ISO 25.02n como *Quality Measure Element* (QME), que inclui tanto a propriedade a ser medida quanto o método de quantificação dessa propriedade. A medição derivada se refere a uma medida que é o resultado de uma função de dois ou mais valores de medições básicas (RAVANELLO, VILLALPANDO, *et al.*, 2015). Esse tipo de medida é tratado na divisão ISO 25.02n como *Quality Element* (QM), ou seja, como uma função de dois ou mais elementos básicos de medição (QME). Os Elementos de Qualidade - EQ (*Quality Element - QM*) e os Elementos de Medição da Qualidade – EMQ (*Qualiity Measure Element - QME*), bem como sua relação com os requisitos de qualidade podem ser melhor observados na Figura 11.

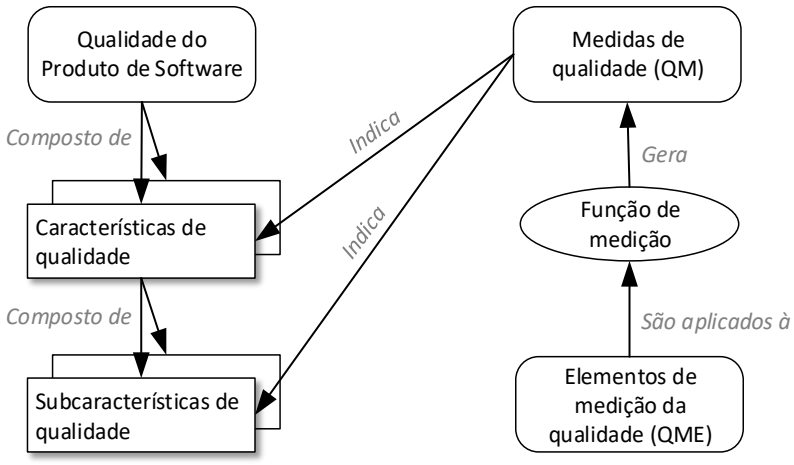



*Figura 11. Modelo de referência de medição, adaptado da ISO 25.020 (ISO/IEC, 2007).*

A ISO/IEC 25.020 apresenta uma introdução geral sobre a divisão, e um modelo de referência que é comum para as medidas internas, externas e de qualidade em uso. Ainda, provê orientações aos usuários para selecionar ou desenvolver, e aplicar medições a partir de padrões internacionais de qualidade.

A ISO/IEC 25.021 apresenta definições e especificações de um conjunto de medidas básicas e derivadas, cujo objetivo é servir de base para todo o desenvolvimento do ciclo de vida do software. O documento descreve um conjunto de medidas que pode ser utilizado como entrada para a medição de qualidade interna, externa ou em uso.

A ISO/IEC 25.022 descreve um conjunto de medidas para medição de qualidade em uso em termos de características e subcaracterísticas, conforme definido na ISO/IEC 25.010, e tem o objetivo de ser utilizada em conjunto com esta norma.

A ISO/IEC 25.023 define medidas de qualidade (QM) para medição quantitativa de sistemas e de produto de software em termos de características e subcaracterísticas, conforme definido na ISO/IEC 25.010, e tem o objetivo de ser utilizada em conjunto com esta norma.

A ISO/IEC 25.024 define medidas de qualidade para medição quantitativa da qualidade em dados, em termos de características e subcaracterísticas, conforme definido na ISO/IEC 25.012, e tem o objetivo de ser utilizada em conjunto com esta norma.

### 2.6.4 Divisão ISO/IEC 25.03n

A divisão ISO/IEC 25.03n se refere à divisão de requisitos de qualidade. Atualmente, a divisão conta com uma norma, a ISO/IEC 25.030. Esta norma provê orientações sobre o processo utilizado para especificar requisitos de qualidade, bem como oferece também requisitos e recomendações para qualidade.

### 2.6.5 Divisão ISO/IEC 25.04n

A divisão ISO/IEC 24.04n se refere à divisão de avaliação da qualidade. Ela é composta de três normas: ISO/IEC 25.040, ISO/IEC 25.041, e ISO/IEC 25.045.

A ISO/IEC 25.040 contém as recomendações para a avaliação de sistema ou de produto de software, provendo a descrição de um processo para avaliação e estabelecendo os requisitos para a aplicação desse processo.

A ISO/IEC 25.041 apresenta requisitos e recomendações específicas para desenvolvedores, compradores (adquirentes) e avaliadores. A ISO/IEC 25.045 prove a especificação para avaliar a subcaracterística de recuperabilidade, definida debaixo da característica de confiabilidade do modelo de qualidade.

### 2.6.6 Divisão ISO/IEC 25.050 a ISO/IEC 25.099

A divisão de extensão, que compreende as normas entre 25.050 e 25.099, está dedicada a agrupar padrões internacionais de qualidade de produto, ou relatórios técnicos, que tratem de domínios específicos de aplicações, ou que possam ser usados para complementar um ou mais padrões internacionais da família ISO/IEC 25.000.

A ISO/IEC 25.051 define requisitos de qualidade para software do tipo RUSP (*Ready to Use Software Product*), bem como requisitos para documentação de testes, incluindo plano de testes, descrição de testes, e resultado de testes.

A ISO/IEC 25.060 descreve uma potencial família de padrões, chamada CIF (*Common Industry Formats*), que documenta a especificação e avaliação da usabilidade de sistemas interativos.

A ISO/IEC 25.062 foi criada com o objetivo de ser uma referência para a elaboração de relatórios de medições obtidas a partir de testes de usabilidade, conforme definido na ISO/IEC 9.241-11, tratando de efetividade, eficiência e satisfação, para um contexto de uso específico.

A ISO/IEC 25.063 define a estrutura de uma descrição de contexto de uso, tanto detalhada quanto de alto nível, para aplicação em um sistema existente, planejado, projetado ou implementado.

A ISO/IEC 25.064 descreve o CIF (*Common Industry Format*) para as necessidades dos usuários. Este documento fornece especificações para a estrutura e o formato do relatório de necessidades dos usuários, incluindo os elementos que devem estar presentes neste relatório. A ISO/IEC 25.065 descreve o CIF (*Common Industry Format*) para as especificações dos requisitos de usuário. Este padrão internacional estabelece a especificação do conteúdo e do formato das especificações de requisitos do usuário, e as relações entre estes elementos. A ISO/IEC 25.066 descreve o CIF (*Common Industry Format*) para relatórios de avaliação. O objetivo deste padrão é definir a especificação do conteúdo dos relatórios de avaliação, incluindo definição e o relacionamento entre os elementos.

## 2.7 Conclusão do capítulo

Este capítulo apresentou um conjunto de conceitos que contribuem para uma melhor compreensão do restante do trabalho. Foi apresentado o conceito de qualidade em modelos conceituais, onde o modelo conceitual foi apresentado como a explicitação de um conjunto de especificações relevantes que contribuem para a compreensão de determinado problema, em determinado domínio, o que inclui o modelo de processo de software, que explicita um conjunto de atividades humanas complexas, desempenhado com o objetivo de produzir software.

Foram apresentados os modelos de avaliação da qualidade de modelos conceituais de (KROGSTIE, LINDLAND e SINDRE, 1995) e de (WAND e WEBER, 2002), destacando que estes modelos buscavam uma forma de organização das características e subcaracterísticas de qualidade, com o intuito de reduzir a ambiguidade do modelo de

qualidade. No entanto, ficou claro estes trabalhos não investiram na forma de medição e no processo de avaliação dos modelos conceituais.

O capítulo considerou ainda as principais gramáticas para representação de processos, incluindo a linguagem natural, e mostrou que alguns autores entendem que as gramáticas diagramáticas tendem a ser menos ambíguas, apesar da maior flexibilidade da linguagem natural para a descrição de processos. Dentre as gramáticas diagramáticas, também chamadas de notações, foram brevemente descritas as seguintes: BPMN, SPEM, IDEF0, YWAL, Petri Nets, Little-JIL e ARIS.

Por fim, foram apresentados os modelos de avaliação da qualidade específicos da área de Engenharia de Software, tais como o de (MCCALL, RICHARDS e WALTERS, 1977) e de (BOEHM, BROWN e LIPOW, 1976). Também foi apresentada a família de normas ISO/IEC 25.000, que é uma evolução do conjunto de normas ISO/IEC 9.126. Foram apresentadas as seis divisões da família ISO/IEC 25.000 e uma breve descrição de cada uma das normas que compõem essas divisões.

# 3 Trabalhos relacionados

## 3.1 Introdução do capítulo

Este capítulo trata dos trabalhos relacionados à pesquisa. São apresentados trabalhos relacionados à avaliação de modelos conceituais de maneira geral e relacionados a modelos conceituais específicos, como o modelo de processo de negócio e o modelo de processo de software.

## 3.2 Modelos conceituais

(MOODY, 2005) observou que existe um conjunto de padrões internacionais criados para a avaliação da qualidade de produtos de software, mas que não existe um conjunto equivalente de padrões internacionais para a avaliação da qualidade de modelos conceituais.

Para o autor, os modelos conceituais continuavam sendo avaliados de maneira *ad hoc*, com base no senso comum e em opiniões subjetivas baseadas somente na experiência pessoal do avaliador. O autor concluiu que as avaliações de modelos conceituais eram realizadas sem o devido suporte científico, sendo mais próximas da "arte" do que de uma disciplina de engenharia.

Para ele, padrões internacionais deveriam ser criados para a avaliação da qualidade de modelos conceituais. Em seu trabalho o autor se propôs a realizar uma revisão da literatura sobre qualidade para modelos conceituais, e a partir desta revisão identificar as questões práticas e teóricas que precisavam ser tratadas para se atingir o objetivo da criação de um padrão internacional de avaliação de qualidade de modelos conceituais. O trabalho buscou ainda definir quais aspectos teóricos e práticos precisariam ser definidos, como a estrutura do modelo de avaliação, e como este modelo de avaliação poderia ser aplicado a modelos conceituais, de modo que se tornassem o padrão de fato.

O autor realizou uma revisão de literatura sobre o tema da qualidade em modelos conceituais, considerando fontes tanto acadêmicas como de mercado. Este trabalho identificou 50 diferentes propostas de frameworks para avaliação de modelos conceituais, embora parte desses frameworks se referissem a extensões ou refinamentos de propostas anteriores. O autor encontrou os seguintes achados em sua pesquisa:

- **Proliferação de propostas**. Havia 50 diferentes propostas publicadas, e novas continuavam surgindo. Estas propostas não eram alinhadas a padrões de qualidade internacionais existentes, como a ISO/IEC 9.126.

- **Ausência de testes empíricos**. Apenas 20% das propostas identificadas haviam realizado algum tipo de teste empírico.
- **Ausência de adoção na prática**. A existência de tantas propostas diferentes implicava em um obstáculo para a adoção de qualquer uma delas, produzindo pouco impacto prático na melhoria de modelos conceituais.
- **Diferentes níveis de generalidade**. As propostas apresentavam diferentes níveis de generalidade, sendo algumas delas aplicáveis a qualquer modelo conceitual enquanto outras eram aplicáveis apenas a modelos conceituais específicos de determinado domínio.
- **Ausência de consenso quanto à terminologia**. Cada proposta utilizava uma terminologia diferente. Por exemplo, o que a ISO/IEC 9.000 chamava de característica de qualidade, recebia denominações diferentes nas diferentes propostas, tais como dimensões de qualidade, fatores, princípios, visões, propriedades, atributos, critérios, categorias e objetivos.
- **Ausência de alinhamento com padrões existentes**. Poucas propostas faziam referência a padrões de qualidade existentes, tais como a ISO/IEC 9.000 ou a ISO/IEC 9.126.
- **Ausência de medição**. A maior parte das propostas especificavam características de qualidade, mas não indicavam como fazer a medição dessas características.
- **Ausência de processo de avaliação**. Com apenas três exceções, todas as demais propostas não especificavam um processo de avaliação a ser conduzido para avaliar a qualidade do modelo conceitual, o que se constituía em uma barreira para a adoção da proposta na prática.
- **Ausência de orientações para melhoria**. O objetivo da maior parte das propostas era a avaliação da qualidade, mas ignoravam a questão de como melhorar esta qualidade a partir da avaliação.
- **Foco em modelos estáticos**. A maior parte das propostas tinha como objeto modelos conceituais estáticos, como o modelo de dados, por exemplo, ao invés de modelos conceituais mais dinâmicos, como o de funcionalidades de um sistema, por exemplo.
- **Foco em produto**. A maior parte das propostas estava focada em produtos e não em processos.

- **Ausência de conhecimento da prática**. A maior parte das propostas avaliavam a qualidade de modelos conceituais, mas não relacionados com as práticas de mercado, ou com modelos conceituais reais.

A seguir, o autor definiu um conjunto de princípios para a estruturação de um framework para avaliação da qualidade de modelos conceituais, que seriam os seguintes:

1. Os requisitos de qualidade para modelos conceituais devem ser decompostos em uma hierarquia de características, subcaracterísticas e medidas;
2. As características e subcaracterísticas devem ser rotuladas com uma única palavra fundamental, e que essas palavras sejam de fácil compreensão;
3. Cada característica e subcaracterística deve ter uma definição única, em uma sentença concisa;
4. Medidas devem ser definidas para cada uma das subcaracterísticas;
5. Deve ser definido um processo claro e suficiente com procedimentos a serem realizados para a avaliação da qualidade do modelo conceitual.

O autor apresentou um projeto inicial para a construção de um framework de avaliação de um modelo conceitual específico, o modelo de dados, alegando haver maior consenso sobre avaliação da qualidade deste tipo de modelo. Também, o autor selecionou a ISO/IEC 9.126 para servir de referência a este projeto. Porém, embora o projeto apresentado tenha sido composto de passos tais como “definir um modelo de qualidade”, “definir características de qualidade”, “definir subcaracterísticas de qualidade” e “definir um modelo de medição”, o trabalho não chegou ao ponto de apresentar a execução deste projeto.

O autor concluiu o trabalho recomendando que esforços precisam ser envidados no sentido de se construir modelos de avaliação da qualidade para modelos conceituais, que sejam baseados em consenso, alinhados com normas de qualidade existentes, utilizando um modelo de qualidade que esteja associado a um sistema de medição, a um processo de avaliação, e que possam produzir recomendações práticas para a melhoria dos modelos conceituais.

(MAES e POELS, 2007) observaram que havia uma proliferação de frameworks de qualidade aplicáveis a modelos conceituais, e que esses modelos conceituais eram utilizados como ferramenta de comunicação entre os analistas e a equipe de desenvolvimento. Para eles era surpreendente que, apesar do aumento da importância dos modelos conceituais, não havia sido desenvolvido um framework para avaliação da

qualidade desses modelos. Em especial, os autores notaram que não havia preocupação em se avaliar a qualidade destes modelos sob o ponto de vista de todas as partes interessadas, incluindo aqueles que forneciam as informações para a construção destes modelos.

A partir desta observação, os autores propuseram elaboração de um framework de avaliação da qualidade para modelos conceituais que incluía o ponto de vista de todas as partes interessadas. Eles utilizaram como referência de modelo de qualidade a proposta de (LINDLAND, SINDRE e SOLVBERG, 1994), e as definições de (WAND e WEBER, 2002) para "gramática" e "script", ou seja, a gramática sendo a notação utilizada para a modelagem, e o script o modelo/diagrama resultante da modelagem. Para tanto, os autores propuseram as seguintes características e definições, como base de seu processo de avaliação:

- **Qualidade semântica percebida** (PSQ, sigla em inglês). Grau em que um usuário consegue compreender o significado dos elementos de um modelo.
- **Facilidade de compreensão percebida** (PEOU, sigla em inglês). Grau em que um usuário consegue compreender um determinado modelo.
- **Usabilidade percebida** (PU, sigla em inglês). Grau em que um usuário acredita que a utilização de um determinado modelo otimiza a performance de seu trabalho.
- **Satisfação do usuário** (US, sigla em inglês). Grau em que o usuário acredita que a informação disponível no modelo atende a suas necessidades.

Os autores optaram por utilizar um modelo conceitual específico em seu trabalho, o modelo conceitual de dados Entidade e Relacionamentos (ER). Dois estudos foram conduzidos, com grupos de estudantes da área de Negócios, que tinham por objeto um modelo ER, embora estes estudantes não dominassem as técnicas de modelagem de dados. Para cada uma das caraterísticas foram propostas uma ou mais medidas, coletadas durante a pesquisa. Os experimentos foram aplicados como exercícios em sala de aula. A partir desses experimentos, os autores fizeram estudos de relação entre as quatro características propostas por eles, encontrando maior correção entre PSQ e PU, e PSEOU e US. Eles concluíram que o trabalho contribuiu para propor um novo modelo para avaliação de modelos conceituais, bem como para demonstrar os relacionamentos entre diferentes dimensões da percepção da qualidade em modelos conceituais.

Contudo, este trabalho não contribuiu na direção do estabelecimento de um processo de avaliação alinhado com padrões existentes e aceitos em nível internacional. De fato, embora parte de um modelo de medição tenha sido proposto, este modelo não está alinhado à nenhuma outra norma de referência. É baseado em medições estritamente qualitativas e subjetivas, e não há a recomendação de estruturas matemáticas que a suportem. Também, não foi proposto um processo de avaliação, que poderia ser repetido por outros pesquisadores a fim de produzir resultados comparáveis e contribuir para o fortalecimento de um padrão para avaliação de modelos conceituais.

## 3.3 Modelos de processos de negócio

(GUCEGLIOGLU e DEMIRORS, 2005) observaram que as organizações investem altas somas em seus sistemas de informação; um investimento superior a 780 bilhões de dólares no ano de 2002, apenas nos Estados Unidos. Eles observaram que, embora houvessem diversas pesquisas para analisar o tempo e o custo em projetos deste tipo, havia espaço para estudos complementares que tratassem da qualidade dos modelos de processo de negócio, utilizados como base da construção destes sistemas.

Os autores decidiram propor um conjunto de características e medidas em processos para avaliar a qualidade de processos de negócio. Eles utilizaram a ISO/IEC 9.126 para selecionar um conjunto de características de qualidade que seriam analisadas, chegando ao seguinte conjunto de características e subcaracterísticas, conforme lista abaixo. Para cada uma dessas características os autores propuseram uma ou mais medidas.

- Manutenibilidade
    - Analisabilidade
- Confiabilidade
    - Maturidade
    - Recuperabilidade
- Funcionalidade
    - Adequabilidade
    - Funcionalidade
    - Acuracidade
    - Interoperabilidade
    - Segurança
- Usabilidade
    - Operabilidade

Eles conduziram um experimento utilizando a característica de “Funcionalidade” e suas cinco subcaracterísticas. Como referência, utilizaram o diagrama de um modelo de processos de uma organização real. O diagrama selecionado foi o “atender requisições de material” do departamento de Almoxarifado, contendo originalmente 29 atividades representadas. O experimento foi rodado em duas etapas, uma primeira baseada neste processo existente (AS-IS) e uma segunda baseada no processo já remodelado (TO-BE), após os ajustes recomendados pela primeira. O processo foi modelado, para ambas as versões, utilizando diagramas de atividade da notação UML.

Todas as medidas foram criadas para receberem resultados entre 0 e 1, e os resultados obtidos em cada etapa são apresentados na Tabela 4.

*Tabela 4. Resultados do experimento de (GUCEGLIOGLU e DEMIRORS, 2005).*

| **Subcaracterística** | **Atributo** | **AS-IS** | **TO-BE** |
|---|---|---|---|
| Adequabilidade | Adequação funcional | 0,793 | 0,916 |
| | Completude funcional | 0,759 | 0,875 |
| Adequação | Grau de uso de TI | 0,241 | 0,667 |
| Acuracidade | Acuracidade funcional | 0,518 | 0,792 |
| Interoperabilidade | Capacidade de troca de dados | 0,857 | 1 |
| Segurança | Auditabilidade | 0,931 | 1 |

Os autores concluíram indicando que seu trabalho era uma proposta inicial e que caberiam novos estudos para consolidá-lo. Afirmaram, ainda, que os atributos utilizados não eram suficientemente claros e não possuíam definições, e que estes atributos não estavam devidamente correlacionados.

Neste trabalho, não foi proposto um modelo de medição devidamente organizado e correlacionado, e também não foi proposto ou descrito o processo utilizado para a construção de um modelo de medição. Adicionalmente, não foi proposto um processo claramente definido para a realização da avaliação utilizando as medidas propostas pelo trabalho.

(ROLÓN, SÁNCHEZ, *et al.*, 2009) observaram que os processos de negócio influenciam significativamente a qualidade dos produtos e do nível de satisfação dos clientes das organizações, fazendo-os compreender a importância destes processos para o mercado. Eles também entenderam que as medições desses processos são importantes para garantir a qualidade dos mesmos. A partir destas observações, os autores propuseram realizar um

trabalho de identificação de medidas que pudessem ser úteis para a avaliação da inteligibilidade e modificabilidade de modelos de processo de negócio representados em notação BPMN. Sendo assim, os autores se preocuparam com as seguintes características e subcaracterísticas, baseados na ISO/IEC 9.126:

- Usabilidade
    - Inteligibilidade
- Manutenibilidade
    - Modificabilidade

Os autores selecionaram um conjunto de medidas que pudessem ser relacionadas às subcaracterísticas. As medidas utilizadas foram selecionadas de (RÓLON, RUIZ, *et al.*, 2006), que por sua vez foram definidas com base na experiência daqueles autores. Desse modo, foram definidas 18 medidas básicas e 11 medidas derivadas.

Os experimentos foram realizados pela apresentação de um modelo de processo representado em BPMN a cinco diferentes grupos de alunos, e pelo preenchimento de dois questionários, um com três questões sobre inteligibilidade, e outro com duas questões sobre a requisição de duas modificações no modelo. Adicionalmente, todos os alunos responderam uma questão sobre complexidade do modelo.

Após a análise dos resultados com base em equações de regressão linear estabeleceram um grupo de medidas que podem ser úteis para a avaliação da qualidade de modelos de processos de negócio em BPMN, no que se refere as subcaracterísticas de inteligibilidade e modificabilidade. Segundo os autores, das 29 medidas originais (entre básicas e derivadas) 12 foram consideradas úteis para a medição de inteligibilidade e 6 foram consideradas úteis para a medição de modificabilidade.

Apesar das contribuições do trabalho, as medidas foram definidas de maneira arbitrária, e o processo de definição dessas medidas não foi explicitado. Também não foi elaborado um processo explícito de avaliação para a aplicação dessas medidas.

(SÁNCHEZ-GONZÁLEZ, GARCÍA, *et al.*, 2013) observaram que a modelagem de processos é uma atividade chave no ciclo de vida dos processos de negócio, pois esta atividade produz uma visão geral da organização através da elaboração de modelos de processos, ou modelos conceituais. A partir desta observação, os autores se propuseram a identificar um grupo de características de qualidade para modelos de processo de negócio e relacionar essas características com medidas de qualidade.

O primeiro passo foi a realização de uma revisão da literatura com o objetivo de identificar características de qualidade em representações de processo de negócio. Os autores sugeriram uma análise comparativa entre seus achados e as características e subcaracterísticas de qualidade propostas pela ISO/IEC 25.010, o modelo de qualidade da família ISO/IEC 25.000. Eles entenderam que embora esta norma tenha sido originalmente criada para avaliação da qualidade do software, poderia ser aplicada ao próprio processo de software, uma vez que o modelo conceitual do processo de software pode ser visto como um artefato de software também, em função de suas características similares ao próprio software (OSTERWEIL, 1987). Indo um pouco além, os autores entenderam que se a norma pode ser aplicada a processos de software, então poderia ser aplicada também a processos de negócios, uma vez que ambos são processos.

Apesar de terem utilizado a norma ISO/IEC 25.010 como referência para a definição de características e subcaracterísticas, eles adaptaram essas características por considerar também partes da ISO/IEC 9.126, que foi a antecessora da ISO/IEC 25.010. Assim, eles chegaram às seguintes características e subcaracterísticas:

- Usabilidade
    - Inteligibilidade
    - Apreensibilidade
    - Estética da interface com o usuário
- Manutenibilidade
    - Modularidade
    - Modificabilidade
- Adaptabilidade
- Correção
- Completude
- Consistência

Após a definição de características e subcaracterísticas, bem como definição de medidas relacionadas, os autores selecionaram um diagrama do modelo de processo real, de um hospital, para coletar as medidas relacionadas com as características de inteligibilidade e modificabilidade. Este modelo de processo estava representado em notação BPMN. Os autores indicaram a necessidade de realizar ajustes no diagrama para aumentar os índices das características e subcaracterísticas de qualidade. No entanto, esta ação não foi descrita no trabalho.

Embora os autores tenham buscado referência em normas de ampla aceitação internacional, eles não seguiram integralmente nem a estrutura de qualidade proposta pela ISO/IEC 25.010 e nem a estrutura de qualidade proposta pela norma anterior, a ISO/IEC 9.126. Ao invés de selecionarem uma das normas e realizarem ajustes no escopo, os autores preferiram selecionar características de uma norma e de outra, e até de propor características não constantes em nenhuma dessas normas. Essa opção, no entanto, representou vulnerabilidade ao estudo, colocando em questão se houve algum tipo de viés na seleção das características e subcaracterísticas a serem utilizadas no trabalho de medição. O trabalho também não propôs um modelo de medição alinhando com um padrão reconhecido, um processo para a construção desse modelo, e nem um processo de avaliação que aplicasse este modelo.

(KAHLOUN e CHANNOUCHI, 2016) observaram que a gestão de processos de negócio (BPM, sigla em inglês) é um importante modo de implementar melhorias na estrutura de uma organização e que, por isso, os modelos de processo de negócio deveriam ser avaliados e melhorados quanto a sua qualidade.

Os autores definiram um conjunto de medidas para a avaliação de qualidade para processos de negócio modelados em notação BPMN. Os autores adaptaram essas medidas a partir da disciplina de Engenharia de Software, e propuseram as seguintes medidas:

- Acoplamento;
- Coesão;
- Complexidade;
- Modularidade;
- Tamanho.

Após a definição das medidas, os autores fizeram a coleta das mesmas a partir de um diagrama de um processo real. Esse diagrama representava o processamento da oferta de currículo em uma instituição de ensino superior. A partir disso os autores propuseram um conjunto de 7 recomendações para a modelagem de processos, que eles chamaram de 7PMG (*seven process modeling guidelines*), que são as seguintes:

1. Reduzir o número de elementos, uma vez que o tamanho afeta a compreensibilidade e a probabilidade de erros no modelo;
2. Minimizar o número de caminhos (fluxos) por elemento;
3. Estar atento ao uso de um evento de início e um evento de fim;

4. Ter um modelo estruturado, no sentido que cada conector de separação tenha um respectivo conector de junção do mesmo tipo;
5. Evitar o uso de caminhos a partir de elementos OR;
6. Usar o formato verbo-objeto no rótulo das atividades;
7. Decompor um diagrama se ele contiver mais do que 50 elementos.

As recomendações apresentadas pelo trabalho são bastante genéricas, como por exemplo a recomendação 1 ou a recomendação 5. Também não são apresentadas justificativas para a maioria das recomendações, como é o caso da recomendação 4 e da recomendação 7.

Adicionalmente, o trabalho não relaciona as medições com subcaracterísticas e características de qualidade, e não propõe um modelo de medição. As medidas são definidas de maneira arbitrária, sem um processo de elaboração associado. Também não há a proposta de um processo de avaliação que possa aplicar as medidas recomendadas.

## 3.4 Modelos conceituais de processos de software

(GARCIA, RUIZ, *et al.*, 2003) observaram que o processo de software possui grande influência sobre a qualidade do produto de software e que, por esta razão, as organizações estão dedicando cada vez mais atenção ao processo de software. Além disso, os autores atribuíram esta atenção ao fato de que o software vem se tornando um produto cada vez mais complexo.

A partir desta observação, os autores propuseram a elaboração de um framework para medição do modelo de processo de software e para sua subsequente melhoria. A proposta incluiu a definição de um conjunto de medidas que pudessem ser coletadas diretamente de modelos de processo de software representados em notação SPEM. Foram criadas 28 medidas a serem aplicadas ao modelo de processo de software.

Os autores produziram um diagrama do modelo de processo de software do modelo de referência RUP (*Rational Unified Process*), e utilizaram este diagrama para a coleta das medidas por eles definidas. Com isso, os autores entenderam terem contribuído no caminho da construção de um framework para a medição e avaliação da qualidade de modelos de processo de software.

No entanto, as medidas foram definidas de maneira arbitrária e nenhum processo para a construção dessas medidas foi proposto. Essas medidas não foram relacionadas a características de qualidade. Também não foi proposto um processo para a avaliação de

modelos de processo de software. As medidas coletadas não implicaram em ajustes no diagrama proposto.

A partir da mesma observação do trabalho anterior, (GARCÍA, RUIZ e PIATTINI, 2004) propuseram um conjunto de medidas para avaliar a influência da complexidade de um modelo de processo de software nas características de usabilidade de manutenibilidade deste modelo. A partir de (GARCIA, RUIZ, *et al.*, 2003), os autores definiram um conjunto de 11 medidas a serem coletadas em modelos de processo de software. Os autores utilizaram o diagrama em SPEM construído em (GARCIA, RUIZ, *et al.*, 2003) para realizarem esta coleta de medidas. A conclusão dos autores foi a de que o trabalho produziu um conjunto representativo de medidas que poderiam ser utilizadas para avaliar a influência da complexidade de um modelo de processo de software em sua usabilidade e manutenibilidade.

Contudo, embora o trabalho tenha buscado uma relação entre as medidas e duas características de qualidade da norma ISO/IEC 9.126, a usabilidade e a manutenibilidade, esta relação não foi definida de maneira explícita. O processo de construção destas medidas também não foi apresentado, além de não ter sido proposto um processo para avaliação da qualidade do processo de software com base nas medidas propostas.

Depois disso, o mesmo grupo de pesquisa publicou novos experimentos sobre a mesma base teórica, com o objetivo de testar as medidas propostas (GARCÍA, RUIZ e PIATTINI, 2004b) (CANFORA, GARCÍA, *et al.*, 2005) (GARCÍA, PIATTINI, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006).

As conclusões destes trabalhos foram similares. (GARCÍA, RUIZ e PIATTINI, 2004b) concluíram que contribuíram com um conjunto representativo de medidas para avaliação da qualidade de processos de software, que relacionam a complexidade estrutural do modelo com sua usabilidade e manutenibilidade. (CANFORA, GARCÍA, *et al.*, 2005) concluíram terem contribuído com um conjunto de medidas empiricamente testadas para a avaliação da manutenibilidade de modelos de processo de software, baseado na complexidade estrutural destes modelos. (GARCÍA, PIATTINI, *et al.*, 2005) concluíram terem contribuído com um framework, que os autores denominaram FMESP, para a avaliação da qualidade de modelos de processo de software, sob a perspectiva da manutenibilidade baseada na complexidade do modelo. (CANFORA, GARCIA, *et al.*, 2006) concluíram que seu trabalho contribuiu com uma proposta empiricamente validada

de medidas para avaliação da manutenibilidade de modelos de processo de software, com base em medidas que permitem avaliar a complexidade destes modelos.

No entanto, apesar de os trabalhos terem buscado parcialmente uma base referencial em padrões de qualidade reconhecidos, como a ISO/IEC 15.939 e a ISO/IEC 9.126, estes trabalhos não propuseram um modelo de medição, ou seja, um padrão de relacionamento entre medidas, subcaracterísticas e características de qualidade. Eles também não contribuíram com um processo para construção das medidas ou com um processo para avaliação de modelos de processo de software.

(DA SILVA, MACIEL e RAMALHO, 2013) observaram que características de qualidade do processo de software poderiam ser aplicadas a processos MDA (*Model Drivem Architecture*) para a construção de software. Segundo os autores, o MDA também requer a definição de um modelo de processo de software que oriente os desenvolvedores na elaboração e geração dos modelos de software, e que este modelo de processo precisa ser avaliado quanto a sua qualidade. Os autores propuseram a avaliação das seguintes característica e subcaracterísticas:

- Manutenibilidade;
    - Analisabilidade;
    - Compreensibilidade;
    - Modificabilidade.

Os autores afirmaram que as subcaracterísticas são influenciadas por medidas internas tais como tamanho, grau de complexidade e grau de acoplamento. O trabalho utilizou as medidas propostas por (CANFORA, GARCÍA, *et al.*, 2005). As medidas foram coletadas a partir de três processos MDA (estudos de caso) mencionados no trabalho, porém não apresentados. Os autores concluíram terem contribuído com uma proposta de framework para avaliação da qualidade de modelos de processos MDA, mas que novos estudos empíricos seriam necessários para consolidar esta proposta.

Neste trabalho os autores não adicionaram à proposta original de (CANFORA, GARCÍA, *et al.*, 2005) um modelo de medição, um processo para a elaboração das medidas, ou um processo para e avaliação da qualidade do modelo de processo de software.

## 3.5 Conclusão do capítulo

Este capítulo apresentou trabalhos relacionados à pesquisa, voltados para a avaliação da qualidade de modelos conceituais, incluindo modelos conceituais de processo de negócio

e modelos conceituais de processo de software. Foram apresentados trabalhos que buscaram definir medidas a serem coletadas a partir dos modelos e que pudessem ser utilizadas para a avaliação da qualidade destes modelos, tanto de maneira individual quanto associados a determinadas características de qualidade.

No entanto, a análise destes trabalhos também forneceu indícios que de há espaço para mais pesquisas nesta área, em especial no que se refere a propor um processo para elaboração de um modelo de medição, para a proposição de um modelo de medição a partir deste processo, e para a proposição de um processo de avaliação da qualidade utilizando este modelo de medição.

# 4 Processo de elaboração de um modelo de medição para avaliação de modelos de processo de software

## 4.1 Introdução do capítulo

O objetivo deste capítulo é propor um processo para construção do modelo de medição para avaliação da qualidade de modelos de processo de software. Este modelo de medição está baseado na família de normas internacionais ISO/IEC 25.000 (ISO/IEC, 2014), mais especificamente na divisão 25.02x desta família (ISO/IEC, 2007). Embora este conjunto de normas seja orientado à avaliação da qualidade de produtos de software, sua estrutura pode ser utilizada para a avaliação de modelos de processo de software.

Este capítulo descreve o modelo de referência para avaliação da estrutura de qualidade de modelos de processos de software e o processo utilizado para a identificação de elementos de medida de qualidade (QME) e medidas de qualidade (QM).

## 4.2 Estrutura do modelo de referência para avaliação da qualidade de modelos de processo de software

O modelo de referência para medição da qualidade de modelos de processo de software (MR-MQMPS) tem a mesma estrutura do modelo de referência para medição da qualidade de produtos da ISO/IEC 25.020 (ISO/IEC, 2007). A Figura 12 apresenta sua adaptação aos objetivos do trabalho.

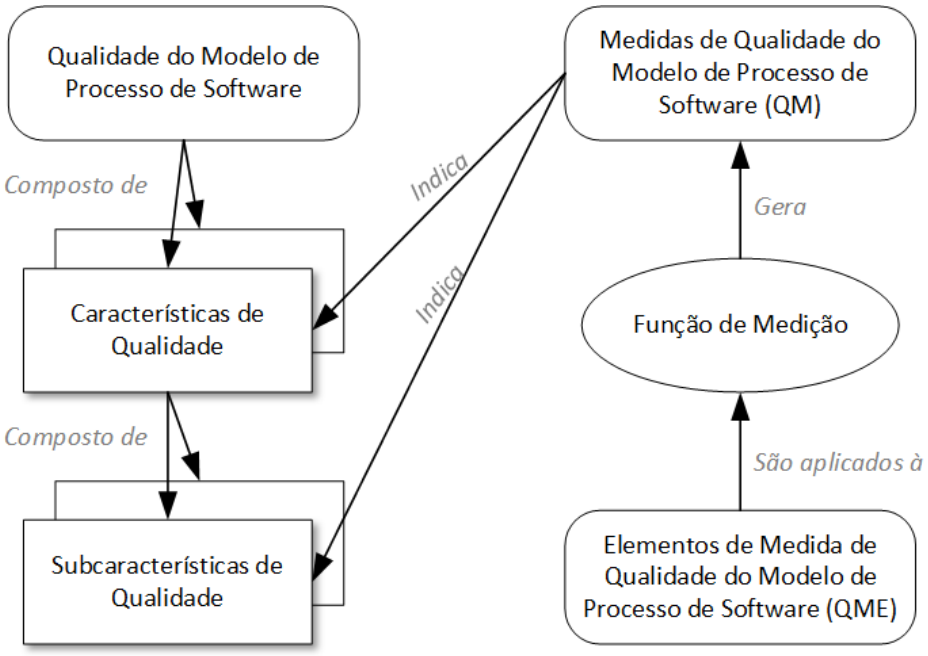



*Figura 12. Modelo de referência para medição da qualidade do modelo de processo de software (MR-MQMPS).*

Este modelo de referência descreve o relacionamento entre o modelo de qualidade, bem como sua associação com características e subcaracterísticas de qualidade, apresentados no lado esquerdo da Figura 12, e os atributos do modelo de processo de software com as correspondentes medidas de qualidade (QM), funções de medição, e elementos de medida de qualidade (QME), apresentados no lado direto da mesma Figura 12.

Para a melhor compreensão do presente capítulo, segue abaixo a lista de termos e definições baseados nas definições da família ISO/IEC 25.000 (ISO/IEC, 2007) (ISO/IEC, 2012) (ISO/IEC, 2016):

*Tabela 5. Termos e definições utilizadas no capítulo.*

| Termo | Definição |
|---|---|
| Entidade | Objeto a ser caracterizado pela medição dos seus atributos. |
| Atributo | Propriedade ou característica inerente de uma entidade que pode ser distinguida quantitativamente ou qualitativamente por humanos ou meios automatizados. |
| Medida | Variável para a qual um valor é atribuído como resultado de uma medição. |
| Medida básica | Medida definida em termos de um atributo e um método de quantificação. |
| Medida derivada | Medida que é definida como a função de dois ou mais valores de medidas básicas. |
| Medição | Conjunto de operações que tem o objetivo de determinar o valor de uma medida. |
| Elemento de medida de qualidade (QME) | Medida definida em termos de uma propriedade e do método de medição para quantificar esta propriedade, incluindo opcionalmente a transformação por uma função matemática. |
| Medida de qualidade (QM) | Medida derivada definida através de uma função de dois ou mais valores de elementos de medida de qualidade. |
| Função de medição | Algoritmo ou cálculo empregado para combinar dois ou mais elementos de medida de qualidade. |
| Método de medição | Organização lógica das operações, descritas de maneira geral, utilizadas em uma medição. |
| Propriedade | Propriedade de uma entidade que está relacionada a um elemento de medida de qualidade (QME) e que pode ser quantificado por um método de medição. |
| Qualidade interna do modelo de processo de software | Capacidade de um conjunto de atributos estáticos de um modelo de processo de software para satisfazer necessidades declaradas e implícitas. |
| Qualidade em uso do modelo de processo de software | Capacidade de um modelo de processo de software de atingir seus objetivos com eficácia, produtividade, segurança e satisfação em determinado contexto de uso. |
| Unidade de medição | Quantidade específica definida e adotada por convenção, com a qual outras quantidadeds do mesmo tipo podem ser comparadas para fins de expressão da magnitute relativa daquela quantidade. |

As medidas de qualidade (QM) e os elementos de medida de qualidade (QME) deverão ser selecionados para atender às necessidades de informação daqueles que elaboram, contratam, utilizam ou de alguma outra forma se relacionam com o modelo de processo de software. Considerando essas necessidades de informação, serão estabelecidos critérios de seleção tanto de QMEs quanto de QMs. Para o caso das QMs, deverá ser explicitada a relação desta QM com o modelo de qualidade, por um lado, e por outro lado com as QMEs.

A estrutura de infomações para as QMEs foi adaptada a partir da ISO/IEC 25.021 (ISO/IEC, 2012), conforme apresentado na Tabela 6. E a estrutura de informações para as QMs foi adaptada da ISO/IEC 25.020 (ISO/IEC, 2007), sendo esta estrutura similar àquela utilizada pela ISO/IEC 25.023 (ISO/IEC, 2016), conforme apresentado na Tabela 7.

*Tabela 6. Estrutura de informações da QME.*

| Item | Campo | Descrição |
|---|---|---|
| 1 | Sigla | Acrônimo identificador da QME. Este identificador é unívoco. |
| 2 | Nome | Nome da QME. Este nome já indica, de maneira geral, o objetivo da QME a ser medida. |
| 3 | Entidade a medir | A entidade a partir da qual será coletada a medida. Por se tratar de modelo de processo de software, esta entidade será, em geral, um diagrama de processo. Contudo, poderá ser também o inteiro modelo, ou apenas parte de um diagrama, como uma Lane. |
| 4 | Elemento notacional a medir | O modelo do processo de software é constituído por diagramas, que contém elementos notacionais. É preciso indicar o elemento notacional a ser utilizado na quantificação. |
| 5 | Propriedade a medir | A propriedade concreta do elemento notacional que será quantificada para a QME. |
| 6 | Unidade de medição | Referência da unidade considerada para medição que possibilite comparar a magnitude da medida em relação a outras quantidades do mesmo tipo. |
| 7 | Método de medição | O método de medição descreve como coletar o valor e como transformar esse valor na propriedade quantificada, através de uma função matemática. |
| 8 | Procedimento de coleta | A forma como a coleta da medida será feita. Pode ser por contagem manual no modelo impresso ou por método automático com software que realize a quantificação das propriedades a medir através do método de medição. |

*Tabela 7. Estrutura de informações da medida de qualidasde QM.*

| Item | Campo | Descrição |
|---|---|---|
| 1 | Sigla | Acrônimo identificador da QM. Este identificador é unívoco. |
| 2 | Nome | Nome atribuído à QM. |
| 3 | Descrição | Descrição do objetivo da QM. |
| 4 | Função de medição | Equação mostrando como os elementos de medida de qualidade (QME) são combinados para produzir a QM. |
| 5 | Característica | Característica de qualidade do modelo de qualidade. |
| 6 | Subcaracterística | Subcaracterística de qualidade do modelo de qualidade. |

## 4.3 Processo de definição de medidas para avaliação do modelo de processo de software

Nesta seção descrevemos o processo para a definição de medidas a serem utilizadas na avaliação da qualidade do modelo de processo de software. O processo está definido em níveis de abstração, e representado em notação BPMN. Este processo é baseado nas normas ISO/IEC 25.020, 25.021 e 25.023 (ISO/IEC, 2007) (ISO/IEC, 2012) (ISO/IEC, 2016).

Os níveis de abstração em modelos BPMN são representados por meio do elemento notacional “Subprocesso”, indicado por um retângulo de cantos arredondados, e com um sinal de “+” na parte inferior deste retângulo. Utilizando o recurso de “Subprocesso” é possível modelar em camadas, criando uma camada para cada nível de abstração. Desta forma, em uma primeira camada é possível ter uma visão bastante ampla do processo, e em uma segunda camada saber mais detalhes dele. Em uma terceira camada, caso necessária, seria possível saber com precisão como as coisas são feitas naquele determinado processo que está sendo modelado, e assim sucessivamente.

A primeira camada do processo é apresentada na Figura 13, e dela constam dois subprocessos e uma atividade:

1. Subrprocesso “Definir elementos de medida de qualidade”, com o objetivo de definir o conjunto de QMEs que serão utilizadas no modelo de medição para avaliação da qualidade de modelos de processo de software;
2. Subprocesso “Definir medidas de qualidade”, com o objetivo de definir o conjunto de QMs que serão utilizadas no modelo de medição para avaliação da qualidade de modelos de processo de software;

3. Atividade “Definir modelo de medição”, com o objetivo de correlacionar as QMs às subcaracterísticas de qualidade, que por sua vez são correlacionadas às características de qualidade.

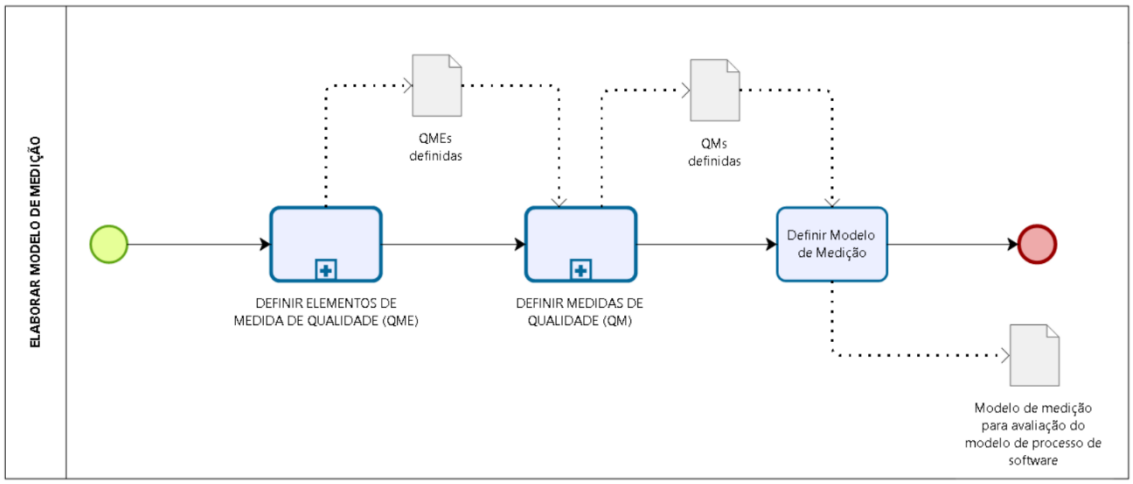



*Figura 13. Processo para elaboração do modelo de medição para avaliação do modelo de processo de software.*

O primeiro subprocesso, “Definir elementos de medida de qualidade”, tem como objetivo identificar e definir QMEs aplicáveis à avaliação da qualidade de modelos de processo de software. A Figura 14 apresenta um segundo nível de abstração para este processo, que é executado por meio das atividades do diagrama descritas na Tabela 8. A estrutura de informações do objeto de dados “QMEs definidas” é composta das informações apresentadas na Tabela 6.

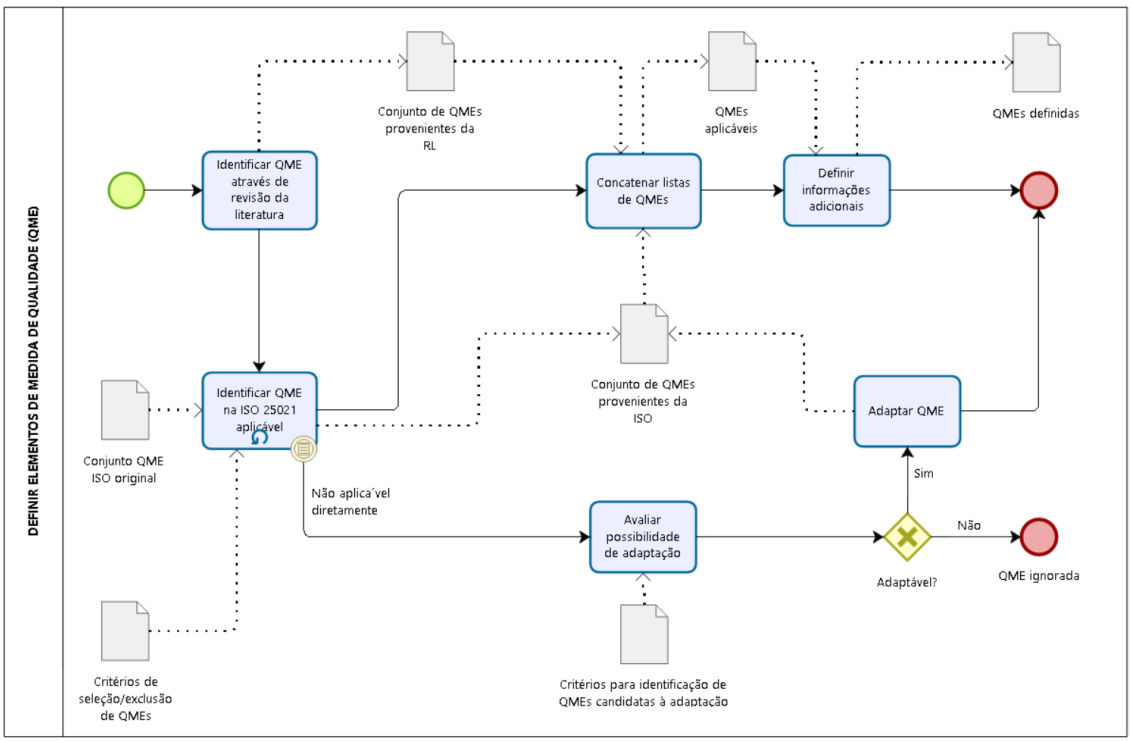



*Figura 14. Processo "Identificar QMEs aplicáveis".*

*Tabela 8. Descrição das atividades "Identificar QMEs aplicáveis".*

| Atividade | Descrição |
|---|---|
| IDENTIFICAR QME ATRAVÉS DE REVISÃO DA LITERATURA | Identificar por meio de revisão da literatura um conjunto de QMEs que possam ser utilizadas para avaliação de modelos de processos de software. A revisão da literatura deve conter na definição do critério de seleção dos abstracts a garantia de que serão considerados apenas os artigos que tratem de modelos de processo de software e de características objetivas de qualidade para estes modelos. Deste modo, não é necessário realizar seleção ou adaptação para o conjunto de QMEs resultantes. O resultado será uma lista de QMEs identificadas por seu nome, sigla, entidade e elemento notacional a medir. |
| IDENTIFICAR QME NA ISO 25021 APLICÁVEL | Identificar cada QME da ISO que seja aplicável à avaliação de modelos de processo de software utilizando os critérios de seleção/exclusão. Caso a QME se enquadre em "não aplicável diretamente" o evento de borda aciona a atividade "Avaliar possibilidade de adaptação". O resultado será uma lista de QMEs identificadas por seu nome, sigla, entidade e elemento notacional a medir. |
| AVALIAR POSSIBILIDADE DE ADAPTAÇÃO | Avaliar se a QME proveniente da ISO, não aplicável diretamente ao modelo de processo de software, pode ser adaptada para este fim. Serão considerados os critérios para identificação de QMEs candidatas à adaptação. Caso seja possível adaptar, o fluxo seguirá para a atividade "Adaptar QME". |
| ADAPTAR QME | Adaptar a QME para aplicação à avaliação de modelos de processo de software, adequando o nome da QME e indicando a entidade e o elemento notacional a serem considerados. Também será considerada a necessidade de adaptações aos aspectos da notação utilizada para a representação do processo. |
| CONCATENAR LISTA DE QMES | Concatenar os dois conjuntos inicias de QMEs, tanto o proveniente da revisão de literatura (RL) quanto o proveniente da ISO. Nesta atividade serão verificadas eventuais duplicidades de QMEs provenientes da revisão de literatura e da ISO. Quando isso ocorrer, será mantida a QME proveniente da ISO, com o objetivo de aumentar a harmonia entre essas QMEs e as QMs eventualmente identificadas também na ISO. |
| DEFINIR INFORMAÇÕES ADICIONAIS | Definir informações adicionais para cada QME, incluindo propriedade a medir, método de medição e procedimento de coleta, produzindo assim um conjunto de QMEs definidas. |

A avaliação da aplicabilidade de uma QME identificada na ISO/IEC 25.021 à avaliação da qualidade de modelos de processo de software considera um conjunto de critérios, conforme listados na Tabela 9.

*Tabela 9. Critérios de seleção/exclusão de QMEs.*

| # | Tipo | Descrição do critério |
|---|---|---|
| S1 | Seleção | Medidas de qualidade interna, ou seja, referentes a atributos inerentes ao modelo de processo de software. |
| S2 | Seleção | Viabilidade de coleta automática. |

| | | |
|---|---|---|
| E1 | Exclusão | Medidas de qualidade em uso, que dependem da opinião do usuário. |

Algumas QMEs podem não ser perfeitamente aplicáveis, mas talvez possam ser adequadas à avaliação de modelos de processo de software após serem objeto de uma adaptação. Para isso, definimos os critérios listados na Tabela 10.

*Tabela 10. Critérios para identificação de QMEs candidatas à adaptação.*

| # | **Descrição do critério** |
|---|---|
| A1 | Similaridade entre a propriedade e entidade da QME na ISO/IEC 25.021 e a propriedade e entidade do modelo do processo de software. |

O resultado final da execução do processo “Definir elementos de medida da qualidade” é um conjunto único de QMEs aplicáveis à avaliação de modelos de processo de software identificadas pelo seu nome e sigla, bem como a entidade e a propriedade a medir e demais informações apresentadas na Tabela 6. Este conjunto será utilizado para a construção das medidas de qualidade (QMs).

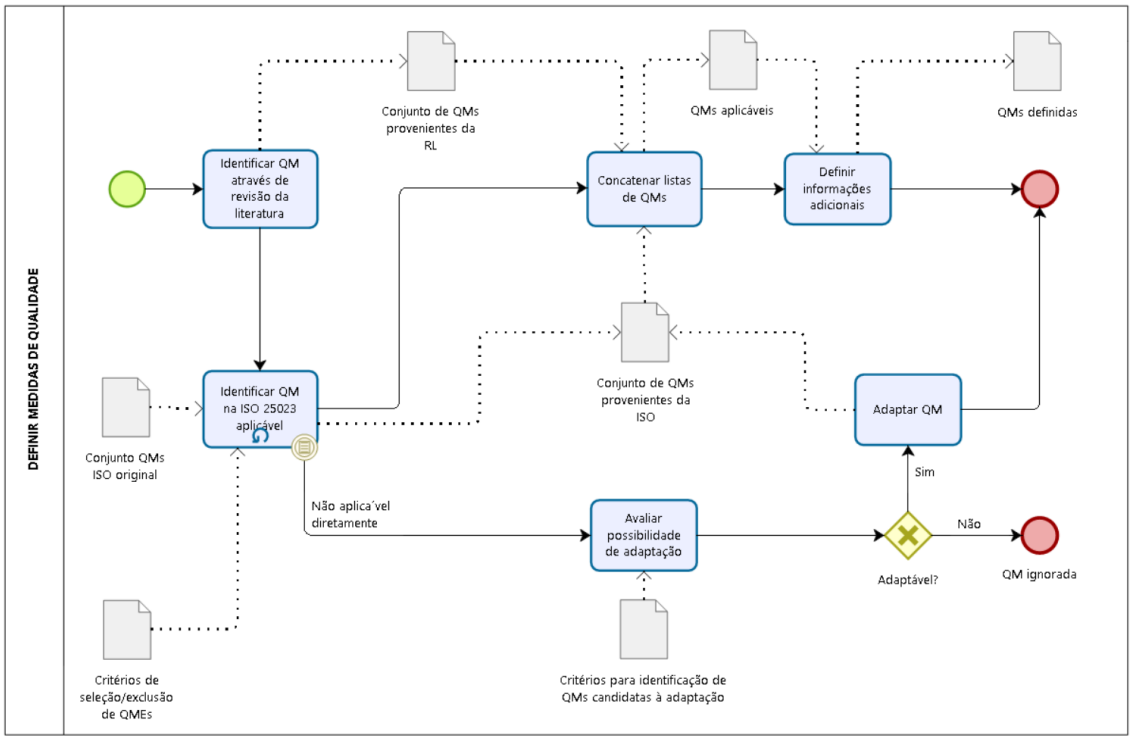



*Figura 15. Processo "Definir medidas de qualidade".*

Para a construção das QMs tem-se o segundo subprocesso proposto na Figura 13, “Definir medidas de qualidade”. Este subprocesso, apresentado na Figura 15, tem o objetivo de

identificar as QMs e fazer o relacionamento dessas QMs às QMEs identificadas. As QMs serão definidas de acordo com a estrutura de informações apresentada na Tabela 7. As atividades deste processo estão descritas na Tabela 11.

*Tabela 11. Descrição do processo "Definir medidas de qualidade".*

| Atividade | Descrição |
|---|---|
| IDENTIFICAR ATRAVÉS DE REVISÃO DA LITERATURA | A partir da revisão de literatura identificar possíveis QMs que possam ser utilizadas para a composição do modelo de medição da qualidade de modelos de processo de software, identificando informações tais como nome, sigla e descrição. |
| IDENTIFICAR QMS NA ISO 25023 APLICÁVEL | Identificar cada QM da ISO que seja aplicável à avaliação de modelos de processo de software utilizando os critérios de seleção/exclusão. Caso a QM se enquadre em "não aplicável diretamente" o evento de borda aciona a atividade "Avaliar possibilidade de adaptação". O resultado será uma lista de QMs identificadas por nome, sigla e descrição. |
| AVALIAR POSSIBILIDADE DE ADAPTAÇÃO | Avaliar se a QM proveniente da ISO, não aplicável diretamente ao modelo de processo de software, pode ser adaptada para este fim. Serão considerados os critérios de identificação de QMs candidatas à adaptação. Caso seja possível adaptar, o fluxo seguirá para a atividade "Adaptar QM". |
| ADAPTAR QM AO MPS | Adaptar a QM à avaliação do modelo de processo de software. |
| CONCATENAR LISTA DE QMS | Concatenar os dois conjuntos de QMs, tanto o proveniente da revisão de literatura quanto o proveniente da ISO. Nesta atividade serão verificadas eventuais duplicidades de QMs. Quando isso ocorrer, será mantida a QM proveniente da ISO, com o objetivo de aumentar a harmonia entre essas QMs e as características e subcaracterísticas de qualidade propostas pela ISO/IEC 25.010. |
| DEFINR INFORMAÇÕES ADICIONAIS | Definir informações adicionais para cada QM, incluindo o relacionamento com a característica e subcaracterística de Qualidade e a função de medição. |

Na atividade "Identificar QMs na ISO 25023 aplicável", são considerados os critérios de seleção exclusão, conforme apresentado na Tabela 12. Caso as QMs identificadas ali não sejam diretamente aplicáveis ao Modelo do Processo de Software, há uma tentativa de adaptação, com base no critério de similaridade de propriedade e entidade utilizados na medição, conforme Tabela 13.

*Tabela 12. Critérios de Seleção/Exclusão de QMs.*

| # | Tipo | Critério de seleção de QMs |
|---|---|---|
| S1 | Seleção | Estar classificada com o nível HR (altamente recomendada) ou com o nível R (recomendada)[2]. |
| S2 | Seleção | Estar dentro do escopo de avaliação pretendido. |
| E1 | Exclusão | Estar classificada com o nível UD[1]. |
| E2 | Exclusão | Estar associado a uso/execução do modelo de processo de software, e não à sua representação. |

*Tabela 13. Critério para identificação de QMs candidatas à adaptação.*

| # | Descrição do critério |
|---|---|
| A1 | Similaridade entre a entidade medida pela QM na ISO/IEC 25.023 e a entidade medida no modelo do processo de software. |

A finalização do processo se dá pela execução da atividade "Integrar modelo de medição", conforme apresentado na Figura 13. O objetivo desta atividade é produzir o modelo de medição para avaliação do modelo de processo de software. Este modelo contém tanto as características e subcaracterísticas de qualidade, como também as QMs e QMEs. Nessa atividade as QMs são correlacionadas às subcaracterísticas, e as subcaracterísticas às características, sempre através de fórmulas matemáticas.

## 4.4 Conclusão do capítulo

Neste capítulo foi proposto um processo para construção de um modelo de medição para avaliação da qualidade de modelos de processo de software. Este processo foi baseado na família de normas ISO/IEC 25.000.

Como base para este processo um modelo de referência para medição da qualidade de modelos de processo de software foi adaptado a partir do modelo de referência para medição da qualidade de produtos da ISO/IEC 25.020. Após a definição de um conjunto

[2] A norma ISO/IEC 25.023 classifica as QMs que apresenta em três níveis de maturidade: HR (*Higly Recommended*), que devem sempre ser utilizadas; R (*Recommended*), que devem ser utilizadas quando apropriado; UD (*User's Discretion*), que devem ser utilizadas apenas como referência, pois sua confiabilidade é desconhecida.

de termos utilizados no capítulo, foram apresentadas duas estruturas de informações, uma para as QMEs e outra para as QMs, conforme Tabela 6 e Tabela 7.

Foi apresentado um processo composto de dois subprocessos, "Definir elementos de medida de qualidade (QME)" e "Definir medidas de qualidade (QM)", e uma atividade, "Definir modelo de medição". Os dois subprocessos foram representados através de suas atividades, sendo estas também descritas. Foram previstos critérios de seleção/exclusão para as QMEs e QMs, bem como critérios para identificação de QMEs e QMs candidatas à adaptação.

O processo prevê a produção de um conjunto de QMEs definidas, um conjunto de QMs definidas, bem como a definição de um modelo de medição para avaliação da qualidade de modelos de processo de software, sendo que este modelo de medição é resultado da integração do modelo de qualidade, com características e subcaracterísticas de qualidade, com um modelo de medição, com QMs e QMEs, conforme apresentado no início do capítulo na Figura 12.

# 5 Execução do processo de elaboração do modelo de medição

## 5.1 Introdução do capítulo

No capítulo 4 foi proposto um processo para a construção de um modelo de medição para avaliação da qualidade de modelos de processo de software. O presente capítulo trata da execução deste processo, produzindo os resultados da cada subprocesso, ou seja, "QMEs definidas", "QMs definidas" e "Modelo de medição" para avaliação do modelo de processo de software. Esta execução foi realizada para as características de qualidade Usabilidade e Manutenibilidade e para modelos de processo de software representados em notação BPMN.

## 5.2 Execução do processo "Definir elementos de medida de qualidade"

O primeiro subprocesso executado tem o objetivo de definir as QMEs, ou seja, o conjunto de elementos de medida de qualidade que serão utilizadas para compor o modelo de medição. Como visto no capítulo anterior, este subprocesso é composto por 6 atividades, cada uma delas contribuindo para o resultado final do subprocesso. Será apresentada a seguir a execução de cada uma dessas atividades.

### 5.2.1 Identificar QME através da revisão da literatura

A primeira atividade se refere à busca na literatura por QMEs já empregadas na realização de medições de qualidade em modelos de processo de software. Com este objetivo foi realizado um trabalho de revisão da literatura baseado nas recomendações de (KITCHENHAM e CHARTERS, 2007) e de (WEBSTER e WATSON, 2002). Considerando o objetivo do trabalho, foram estabelecidas as questões de pesquisa apresentadas na Tabela 14.

*Tabela 14. Questões da pesquisa.*

| Questão | Objetivo |
|---|---|
| Quais são as características de qualidade aplicáveis ao modelo de processo de software? | Identificar as características de qualidade aplicáveis ao modelo de processo de software. |
| Essas características são categorizadas e correlacionadas a padrões e normas existentes? | Verificar se as características de qualidade identificadas são relacionadas a modelos de qualidade propostos por padrões existentes. |
| As características são associadas a medidas e a um sistema de medição? | Verificar se as características identificadas são medidas. |

A primeira questão do estudo busca identificar um conjunto de características de qualidade para o modelo de processo de software. A definição deste conjunto é um passo inicial no sentido de se identificar uma referência objetiva de qualidade para o modelo de processo de software. A segunda questão procura saber se as características identificadas são categorizadas e inter-relacionadas de alguma forma, o que possibilitaria sua comparação com modelos de qualidade que organizam características de qualidade através de relações e interdependências, como por exemplo a ISO/IEC 25.010. E por fim, a terceira questão procura estabelecer uma relação entre o conjunto de características identificadas e uma forma de medição. A partir destas questões foi elaborada uma estratégia para o trabalho de revisão da literatura, conforme apresentado na Tabela 15.

*Tabela 15. Estratégia para a revisão de literatura.*

| Etapa da estratégia | Descrição |
|---|---|
| Definição do objetivo da revisão de literatura | Identificar estudos já realizados sobre avaliação da qualidade do modelo de processo de software de maneira objetiva, ou seja, através de referências mensuráveis. |
| Definição das fontes de pesquisa | A área de pesquisa foi "Ciência da Computação", nas bases ACM Digital Library, IEEE Xplore, Scopus e Science Direct. |
| Definição da string de pesquisa | "software process model" and "quality"[3]. |
| Definição do critério de inclusão | São considerados os estudos que tratam da qualidade em modelos de processo de software e na especificação objetiva de suas características de qualidade. |
| Definição do critério de exclusão | São desconsiderados os estudos que:<br>A - Não abordam de maneira objetiva as dimensões (características) de qualidade modelos de processo de software;<br>B - Publicados em idiomas que não sejam o inglês, português ou espanhol;<br>C - Que estejam indisponíveis para leitura. |
| Definição do critério de avaliação dos abstracts selecionados a partir da *string* de pesquisa | A – O estudo trata de qualidade de modelos de processo de software?<br>B – O estudo apresenta características de qualidade objetivas e mensuráveis para modelos de processo de software? |

[3] Considerando-se o número reduzido de artigos publicados sobre medição da qualidade em modelos de processo de software, optou-se por uma *string* de pesquisa que resultasse em uma amostra maior de artigos, a partir da qual poderiam ser selecionados os artigos mais adequados após análise criteriosa dos abstracts.

A partir da *string* de pesquisa, foram extraídos inicialmente 266 artigos provenientes das fontes pré-definidas. Destes, 39 artigos eram repetidos. Após a análise dos abstracts dos 227 artigos restantes, foram selecionados 28 artigos com base no critério de inclusão.

Sobre esses 28 artigos foram aplicados os critérios de exclusão, após leitura integral desses artigos, o que resultou em uma seleção de 3 artigos. Dos 28 artigos considerados, 23 artigos foram descartados em função do critério de exclusão A, 1 artigo em função do critério de exclusão B, e 1 artigo em função do critério de exclusão C. Os resultados desse processo podem ser observados na Tabela 16.

*Tabela 16. Resultado busca de artigos nas fontes.*

| **Fonte** | **Artigos identificados** | **Artigos não repetidos** | **Selecionados com base nos critérios de inclusão** | **Excluídos com base nos critérios de exclusão** | **Total de artigos inclusos** |
|---|---|---|---|---|---|
| ACM Digital Library | 97 | 87 | 6 | 5 | 1 |
| IEEE Xplore | 43 | 20 | 2 | 2 | 0 |
| Scopus | 114 | 108 | 17 | 15 | 2 |
| Science Direct | 12 | 12 | 3 | 3 | 0 |
| **Total** | **266** | **227** | **28** | **25** | **3** |

Após a aplicação de todo o protocolo de revisão da literatura, apenas 3 artigos foram selecionados. Em função disso, foram realizadas buscas em outros artigos, citados por estes 3 artigos originais (g*o backward*) e também em artigos que citam esses mesmos três artigos (*go foraward*) (WEBSTER e WATSON, 2002). Assim, além dos 3 artigos identificados na pesquisa inicial, dois artigos adicionais foram identificados pela aplicação de *go backward*, e outros dois artigos foram identificados pela aplicação de *go foraward*, totalizando então 7 artigos. Os artigos adicionalmente encontrados nas referências, embora não tivessem sido identificados pela *string* de pesquisa, foram identificados por serem trabalhos do mesmo grupo de pesquisa que havia publicado artigos encontrados pela *string* de pesquisa, porém não apresentavam indicação desta relação no título ou no resumo dos artigos.

Dos artigos identificados, 4 deles são trabalhos produzidos pelo mesmo grupo de pesquisa, o Alarcos Research Group, que tem por missão investigar os diferentes aspectos

relacionados com a qualidade e sustentabilidade de sistemas, atuando na área de Ciência da Computação na Universidade de Castilla-La Mancha, Espanha:

- (Integrated Measurement for the Evaluation and Improvement of Software Processes, 2003)
- (Definition and empirical validation of metrics for software process models, 2004)
- (An experimental replica to validate a set of metrics for software process models, 2004b)
- (A proposal and empirical validation of metrics to evaluate the maintainability of software process models, 2006)

Outros 2 artigos são resultantes de uma colaboração entre o já citado grupo de pesquisa Alarcos e o Research Centre on Software Technology (RCOST) da Universidade de Sannio, na Itália:

- (FMESP: Framework for the modeling and evaluation of software processes, 2005)
- (A family of experiments to validate metrics for software process models, 2005)

O último artigo é um trabalho conjunto de autores da Universidade Federal da Bahia e da Universidade Federal de Campina Grande:

- (Evaluating maintainability of MDA software process models , 2013)

A análise dos estudos selecionados identificou a existência de características de qualidade apontadas pelos estudos primários, conforme apresentado na Tabela 17.

*Tabela 17. Características de qualidade identificadas nos artigos.*

| Característica | Definição dos autores | Artigo |
|---|---|---|
| Compreensibilidade | Não apresenta definição própria. Relacionada a Compreensibilidade como dependente de Complexidade e suporte para Manutenibilidade. | (GARCÍA, RUIZ e PIATTINI, 2004) |
| | Não apresenta definição própria. Relacionada a Compreensibilidade como dependente de Complexidade e suporte para Manutenibilidade. | (GARCÍA, RUIZ e PIATTINI, 2004b) |
| | Facilidade com a qual o modelo pode ser compreendido. | (CANFORA, GARCÍA, *et al.*, 2005) |
| Complexidade | Não apresenta definição própria. Relaciona a complexidade com indicadores obtidos a partir de modelos de processo. | (GARCIA, RUIZ, *et al.*, 2003) |
| | Não apresenta definição própria. Relaciona a complexidade com indicadores obtidos a partir de modelos de processo. | (GARCÍA, RUIZ e PIATTINI, 2004) |

| | | |
|---|---|---|
| | Não apresenta definição própria. Relaciona a complexidade com indicadores obtidos a partir de modelos de processo. | (GARCÍA, RUIZ e PIATTINI, 2004b) |
| Manutenibilidade | Não apresenta definição própria. Relaciona Manutenibilidade como dependente de Compreensibilidade e Modificabilidade. | (GARCÍA, RUIZ e PIATTINI, 2004) |
| | Não apresenta definição própria. Relaciona Manutenibilidade como dependente de Compreensibilidade e Modificabilidade. | (GARCÍA, RUIZ e PIATTINI, 2004b) |
| | Não apresenta definição própria. Relaciona Manutenibilidade como dependente de Compreensibilidade e Modificabilidade. | (GARCÍA, PIATTINI, *et al.*, 2005) |
| | Não apresenta definição própria. | (CANFORA, GARCIA, *et al.*, 2006) |
| | Não apresenta definição própria. Faz referência a (CANFORA, GARCÍA, *et al.*, 2005), que também não define este termo. | (DA SILVA, MACIEL e RAMALHO, 2013). |
| Analisabilidade | Facilidade proporcionada pelo modelo para descoberta de erros ou deficiências e na identificação das partes que devem ser modificadas | (CANFORA, GARCÍA, *et al.*, 2005) |
| Modificabilidade | Não apresenta definição própria. Relaciona Modificabilidade como dependente de Complexidade e suporte para Manutenibilidade. | (GARCÍA, RUIZ e PIATTINI, 2004) |
| | Não apresenta definição própria. Relaciona Modificabilidade como dependente de Complexidade e suporte para Manutenibilidade. | (GARCÍA, RUIZ e PIATTINI, 2004b) |
| | Facilidade com que o modelo pode ser modificado, para corrigir possíveis erros, em resposta a uma requisição de mudança específica ou a um novo requisito. | (CANFORA, GARCÍA, *et al.*, 2005) |

Esses estudos apontaram ainda a existência de um conjunto de medidas básicas, como pode ser visto na Tabela 18.

*Tabela 18. Medidas básicas aplicadas ao modelo de processo de software.*

| Sigla | Medida básica | Artigo |
|---|---|---|
| NA | Número de atividades | (GARCÍA, RUIZ e PIATTINI, 2004) (GARCIA, RUIZ, *et al.*, 2003) (DA SILVA, MACIEL e RAMALHO, 2013) (GARCÍA, RUIZ e PIATTINI, 2004b) (CANFORA, GARCÍA, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006) (GARCÍA, PIATTINI, *et al.*, 2005) |

| | | |
|---|---|---|
| NPD | Número de atividades que são predecessoras | (GARCIA, RUIZ, *et al.*, 2003) |
| NSD | Número de atividades que são sucessoras | (GARCIA, RUIZ, *et al.*, 2003) |
| NDA | Número de dependência entre atividades | (GARCÍA, RUIZ e PIATTINI, 2004) (GARCIA, RUIZ, *et al.*, 2003) (DA SILVA, MACIEL e RAMALHO, 2013) (GARCÍA, RUIZ e PIATTINI, 2004b) (CANFORA, GARCÍA, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006) (GARCÍA, PIATTINI, *et al.*, 2005) |
| NSTP | Número de tarefas | (GARCIA, RUIZ, *et al.*, 2003) (DA SILVA, MACIEL e RAMALHO, 2013) |
| NAWPIn | Número de atividades em que há produtos de trabalho de entrada | (GARCIA, RUIZ, *et al.*, 2003) (DA SILVA, MACIEL e RAMALHO, 2013) (CANFORA, GARCÍA, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006) |
| NAWPOut | Número de atividades em que há produtos de trabalho de saída | (GARCIA, RUIZ, *et al.*, 2003) (DA SILVA, MACIEL e RAMALHO, 2013) (CANFORA, GARCÍA, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006) |
| NAWPInOut | Número de atividades em que há produtos de trabalho de entrada e de saída | (GARCIA, RUIZ, *et al.*, 2003) |
| NoP | Número de fases | (DA SILVA, MACIEL e RAMALHO, 2013) |
| NoI | Número de iterações | (DA SILVA, MACIEL e RAMALHO, 2013) |
| NWP | Número de produtos de trabalho | (GARCÍA, RUIZ e PIATTINI, 2004) (GARCIA, RUIZ, *et al.*, 2003) (DA SILVA, MACIEL e RAMALHO, 2013) (GARCÍA, RUIZ e PIATTINI, 2004b) (CANFORA, GARCÍA, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006) |
| NPR | Número de papéis | (GARCÍA, RUIZ e PIATTINI, 2004) (GARCIA, RUIZ, *et al.*, 2003) (DA SILVA, MACIEL e RAMALHO, 2013) (GARCÍA, RUIZ e PIATTINI, 2004b) (CANFORA, GARCÍA, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006) (GARCÍA, PIATTINI, *et al.*, 2005) |
| NR | Número de papéis responsáveis por atividades | (GARCIA, RUIZ, *et al.*, 2003) |
| NAPR(R) | Número de atividades cuja responsabilidade é do papel R | (GARCIA, RUIZ, *et al.*, 2003) |
| NDWPin | Número de produtos de trabalho dos utilizados por atividades | (GARCÍA, RUIZ e PIATTINI, 2004) (GARCIA, RUIZ, *et al.*, 2003) (GARCÍA, RUIZ e PIATTINI, 2004b) (GARCÍA, PIATTINI, *et al.*, 2005) |
| NDWPout | Número de produtos de trabalho (*work product*) produzidos por atividades | (GARCÍA, RUIZ e PIATTINI, 2004) (GARCIA, RUIZ, *et al.*, 2003) (GARCÍA, RUIZ e PIATTINI, 2004b) (GARCÍA, PIATTINI, *et al.*, 2005) |
| NDWPInOut | Número total de produtos de trabalho relacionados a atividades | (GARCIA, RUIZ, *et al.*, 2003) (GARCÍA, RUIZ e PIATTINI, 2004b) |

Os estudos não classificam essas medidas como QMEs ou QMs. Isto evidencia o fato de que não houve a preocupação em aproximar os resultados desses estudos com padrões de

qualidade existentes na área de Engenharia de Software, tais como a ISO/IEC 15.939 (ISO/IEC, 2009), a ISO/IEC 25.020 (ISO/IEC, 2007), a ISO/IEC 25.021 (ISO/IEC, 2012) e a ISO/IEC 25.023 (ISO/IEC, 2016).

É possível perceber que na Tabela 18 há medidas que se fazem presentes em diversos artigos, ao passo que outras constam de apenas um artigo. Embora os artigos tenham proposto uma relação entre essas medidas e as características de qualidade apresentadas na Tabela 17, os próprios autores recomendam a realização de novos estudos para confirmar essa relação. Ainda, as características de qualidade não foram organizadas através de correlações e interdependências como acontece, por exemplo, na ISO/IEC 25.010.

O número de estudos identificados no protocolo da revisão de literatura foi relativamente pequeno, chegando a apenas 7, mesmo após a aplicação de técnicas de *snowball*, conforme (WEBSTER e WATSON, 2002). Isso pode indicar a necessidade de mais pesquisas sobre o tema da medição de qualidade em modelos de processo de software.

Considerando as características identificadas na revisão da literatura apresentadas na Tabela 17, foi identificada apenas uma característica de qualidade compatível com a ISO/IEC 25.010, a "Manutenibilidade". No que se refere às demais características identificadas no estudo, a "Compreensibilidade" está diretamente relacionada com uma subcaracterística da ISO/IEC 9.126, e outras duas estão relacionadas com subcaracterísticas da ISO/IEC 25.010, que são "Analisabilidade", e "Modificabilidade", ambas subcaracterísticas de "Manutenibilidade".

Ao final desta revisão de literatura, cabe responder às questões de pesquisa estabelecidas na Tabela 14:

1. Quais são as características de qualidade do processo de software?
   Foram identificadas 6 características nos artigos considerados, conforme apresentado na Tabela 17.
2. Essas características são categorizadas e correlacionadas a padrões e normas existentes?
   As características não são categorizadas e não são estruturadas de maneira interdependente, o que seria possível pela proposição de subcaracterísticas, como acontece na norma ISO/IEC 25.010 (ISO/IEC, 2011). Portanto, também não estão explicitamente correlacionadas a padrões existentes.
3. As características são associadas a medidas e a um sistema de medição?

As características são associadas a medidas básicas e derivadas, e estas associações são estabelecidas através de um sistema de medição, ou seja, de funções de medição entre as medidas básicas e derivadas.

Após a identificação das medidas básicas e derivadas, e das conclusões da revisão de literatura, resta apenas elencar as medidas básicas dentro dos parâmetros estabelecidos no capítulo anterior, caracterizando-as definitivamente como QMEs provenientes da revisão, conforme Tabela 20.

*Tabela 19. Conjunto de QMEs da revisão da literatura.*

| Sigla | Nome | Entidade a medir | Elemento a medir |
|---|---|---|---|
| NA | Número de atividades | Diagrama de processo de software | Atividade |
| NPD | Número de atividades que são predecessoras | Diagrama de processo de software | Atividade |
| NSD | Número de atividades que são sucessoras | Diagrama de processo de software | Atividade |
| NDA | Número de dependência entre atividades | Diagrama de processo de software | Atividade |
| NSTP | Número de tarefas | Diagrama de processo de software | Atividade |
| NAWPIn | Número de atividades em que há produtos de trabalho de entrada | Diagrama de processo de software | Atividade |
| NAWPOut | Número de atividades em que há produtos de trabalho de saída | Diagrama de processo de software | Atividade |
| NAWPInOut | Número de atividades em que há produtos de trabalho de entrada e de saída | Diagrama de processo de software | Atividade |
| NoP | Número de fases | Diagrama de processo de software | Subprocesso do diagrama inicial |
| NoI | Número de iterações | N/A[4] | N/A[4] |
| NWP | Número de produtos de trabalho | Diagrama de processo de software | Produtos de trabalho |
| NPR | Número de papéis | Diagrama de processo de software | Lane |

[4] Como este capítulo trata da execução do processo de elaboração de um modelo de medição para modelos de processo de software representados em BPMN, é necessário destacar que não é possível medir essa QME diretamente a partir da representação BPMN.

| | | | |
|---|---|---|---|
| NR | Número de papéis responsáveis por atividades | Diagrama de processo de software | Lane |
| NAPR(R) | Número de atividades cuja responsabilidade é do papel R | Diagrama de processo de software | Atividade e Lane |
| NDWPin | Número de produtos de trabalho dos utilizados por atividades | Diagrama de processo de software | Produtos de trabalho |
| NDWPout | Número de produtos de trabalho produzidos por atividades | Diagrama de processo de software | Produtos de trabalho |
| NDWPInOut | Número total de produtos de trabalho relacionados a atividades | Diagrama de processo de software | Produtos de trabalho |

### 5.2.2 Identificar QMEs na ISO 25.023 aplicáveis à avaliação do modelo de processo de software

A ISO/IEC 25.021 propõe um conjunto de QMEs para suportar o modelo de qualidade de software da família de normas ISO/IEC 25.000 (ST-LOUIS e WITOLD, 2012). De fato, esta norma apresenta dois conjuntos de QMEs, sendo o primeiro um conjunto inicial para ser avaliado quanto à sua aplicabilidade pelos usuários da norma, e se encontra no Anexo A. O segundo é um conjunto estendido a partir do primeiro, apresentado no Anexo C (ISO/IEC, 2012). O conjunto completo de QMEs é apresentado na Tabela 20, sendo os itens de 1 a 22 procedentes do Anexo A, e os itens de 23 a 44 procedentes do Anexo C. A tabela indica ainda a ação tomada durante a execução da atividade "Identificar QME na ISO 25021 aplicável", considerando os critérios de seleção/exclusão, como descrito no capítulo anterior.

*Tabela 20. Conjunto original de QMEs, conforme (ISO/IEC, 2012).*

| # | Elemento de Medição de Qualidade (Quality Measure Element [QME]) | Ação (Considerando Tabela 9) |
|---|---|---|
| 1 | Número de acessos à função de ajuda | E1 |
| 2 | Número de problemas de usuário (reclamações sobre o produto) | E1 |
| 3 | Número de registros de dados | Selecionado |
| 4 | Duração (unidades de período de trabalho [horas, dias, semanas, etc.]) | E1 |
| 5 | Esforço (quantidade de trabalho para atingir um resultado, medido em horas) | E1 |
| 6 | Número de falhas do sistema (impossibilidade de concluir uma função, pelo sistema [incluindo equipamentos, manuais de operação, software, requisitos de operação, etc.]) | E1 |

| | | |
|---|---|---|
| 7 | Número de falhas (impossibilidade de concluir uma função, pelo software) | E1 |
| 8-1 | Número de falhas (código) | E1 |
| 8-2 | Número de falhas (projeto) | E1 |
| 8-3 | Número de falhas (requisitos) | E1 |
| 9 | Tamanho funcional do produto (quantificado pelos requisitos funcionais) | Selecionado |
| 10 | Número de interrupções (suspensão de um processo para tratar de um evento externo ao processo) | E1 |
| 11 | Número de itens de dados | Selecionado |
| 12 | Número de mensagens de erro (mensagem informando que ocorreu um erro de entrada de dados ou um outro erro no processo) | Selecionado |
| 13 | Número de erros (ação humana que produz um resultado incorreto) | E1 |
| 14 | Número de mensagens (uma comunicação enviada de um objeto para outro) | Selecionado |
| 15 | Número de passos (de um procedimento [um elemento ou lista numerada que diz ao usuário como executar uma determinada ação]) | Selecionado |
| 16 | Número de tarefas (ação possível, recomendada ou requerida para a realização de um resultado de um processo) | Selecionado |
| 17 | Número de casos de teste (menor parte executável de um pacote de testes) | Não atende S1 |
| 18 | Número de casos de uso (descrição da interação entre um ator e um software, realizada através de uma sequência de passos) | E1 |
| 19 | Número de operações (execução de atividades que produzem o mesmo produto ou entregam repetitivamente o mesmo serviço) | E1 |
| 20 | Número de erros fatais (erro que resulta na inabilitação de um sistema ou componente) | E1 |
| 21 | Tamanho do banco de dados | Não atende a S2 |
| 22 | Tamanho da memória | Não atende a S1 |
| 23 | Duração do processamento | E1 |
| 24 | Duração da disponibilidade do serviço (sistema operacional) | Não atende a S1 |
| 25 | Tempo de resposta | E1 |
| 26 | Duração do trabalho do operador (tempo para a realização de um conjunto de tarefas pelo operador) | E1 |
| 27 | Duração da operação | E1 |
| 28 | Duração da restauração (em caso de interrupção) | E1 |
| 29 | Duração da migração do sistema | Não atende a S1 |
| 30 | Duração da modificação | Não atende a S1 |
| 31 | Número de usuários (requisições de usuários) | E1 |
| 32 | Número de usuários (autorizados) | E1 |
| 33 | Número de interfaces (Um conjunto nomeado de operações que caracterizam o comportamento de um elemento) | Selecionado |
| 34 | Tamanho dos logs | E1 |
| 35 | Número de documentos | Selecionado |
| 36 | Número de equipamentos e suas localizações | Não atende S1 |
| 37 | Número de conexões (para transmissão de dados) | Não atende S1 |
| 38 | Número de recursos (ativos) | Não atende S1 |

| | | |
|---|---|---|
| | | Não atende S2 |
| 39 | Número de componentes (um conjunto de funcionalidades bem definidas identificado por um único nome) | Selecionado |
| 40 | Volume de dados (conjunto de informações definido por uma estrutura formal e compreensível por humanos) | Número total de elementos de dados (data object e data store) |
| 41 | Número de requisições | E1 |
| 42 | Volume de trabalho que pode ser executado por um computador, sistema ou componente | Não atende a S1 |
| 43 | Número de tolerâncias a falhas (grau em que um sistema, produto ou componente opera adequadamente apesar de falhas de hardware ou software) | Selecionado |
| 44 | Número de requisições do usuário para alteração de interface (referente aos aspectos visuais do software) | E1 |

### 5.2.3 Avaliar possiblidade de adaptação

A atividade anterior, "Identificar QME na ISO 25021 aplicável", identifica QMEs que possam ser aplicadas ao modelo de processo de software. Quando a QM não é aplicável diretamente a atividade corrente avalia a possiblidade de adaptação. No entanto, considerando o objetivo original dessas QMEs, o produto de software, é natural que elas precisem ser adaptadas em algum grau para aplicação ao modelo de processo de software. De fato, a QME que não precisou de adaptação é aquela apresentada na Tabela 21.

*Tabela 21. Conjunto de QMEs da ISO que não precisaram de adaptação.*

| # | Elemento de Medição de Qualidade (QME) |
|---|---|
| 1 | Número de mensagens |

As demais QMEs foram avaliadas em relação ao "critério para identificação de QMEs candidatas à adaptação". Na Tabela 22 são apresentadas as QMEs e sua relação com o critério mencionado.

*Tabela 22. QMEs da ISO após aplicação de critérios para identificação de QMs candidatas à adaptação*

| # | Elemento de Medição de Qualidade (QME) | Ação (Considerando a Tabela 10) |
|---|---|---|
| 1 | Número de registros de dados | Não adaptável |
| 2 | Tamanho funcional do produto (quantificado pelos requisitos funcionais) | Adaptável |
| 3 | Número de itens de dados | Adaptável |
| 4 | Número de mensagens de erro (mensagem informando que ocorreu um erro de entrada de dados ou um outro erro no processo) | Não adaptável |

| | | |
|---|---|---|
| 5 | Número de passos (de um procedimento [um elemento ou lista numerada que diz ao usuário como executar uma determinada ação]) | Adaptável |
| 6 | Número de tarefas (ação possível, recomendada ou requerida para a realização de um resultado de um processo) | Adaptável |
| 7 | Número de interfaces (Um conjunto nomeado de operações que caracterizam o comportamento de um elemento) | Adaptável |
| 8 | Número de documentos | Adaptável |
| 9 | Número de componentes (um conjunto de funcionalidades bem definidas identificado por um único nome) | Adaptável |
| 10 | Volume de dados (conjunto de informações definido por uma estrutura formal e compreensível por humanos) | Adaptável |
| 11 | Número de tolerâncias à falhas (grau em que um sistema, produto ou componente opera adequadamente apesar de falhas de hardware ou software) | Adaptável |

### 5.2.4 Adaptar QME

A atividade de adaptação de QME considera duas premissas: 1) a adaptação deve ser feita para atender ao modelo de processo de software; 2) a adaptação deve considerar as particularidades da notação utilizada para a representação deste processo. Como já definido no início do capítulo, o modelo de medição de qualidade do modelo de processo de software proposto pelo presente trabalho é para aplicação em modelos de processo de software representados em notação BPMN. Sendo assim, cada uma das QMEs foi adaptada já considerando a notação, o que eventualmente implicou na divisão (especialização) das QMEs originais. São apresentadas a seguir as ações de adaptação para cada uma das QMEs.

*Tabela 23. QMEs da ISO após adaptação.*

| Elemento de Medição de Qualidade (QME) | QME adaptada | Considerações |
|---|---|---|
| 1 - Tamanho funcional do produto | 1 - Número de atividades do modelo | O tamanho funcional de um software pode ser derivado pela quantificação de suas funcionalidades, ou requisitos funcionais (ISO/IEC, 2012). Um modelo de processo é a representação de uma sequência de atividades com o objetivo de realizar um trabalho (OMG, 2013). Por similaridade, o tamanho funcional de um modelo de processo de software pode ser derivado pela quantificação de suas atividades. |
| 2 - Número de passos | 2 - Número de subprocessos | Um passo é composto de uma ou mais ações, ou tarefas (ISO/IEC, 2012). Em BPMN uma tarefa é uma atividade atômica, ou seja, a menor unidade de trabalho representável (OMG, 2013). Por similaridade, um passo foi considerado o equivalente a um subprocesso BPMN, considerando que um subprocesso pode conter uma ou mais atividades. |

| | | |
|---|---|---|
| 3 - Número de tarefas | 3 - Número de atividades | Uma tarefa é uma ação ou atividade necessária para se produzir um resultado (ISO/IEC, 2012). Em BPMN uma tarefa é uma atividade atômica, ou seja, a menor unidade de trabalho representável (OMG, 2013). Por similaridade, uma tarefa foi considerada uma atividade em BPMN. |
| 4 - Número de interfaces | 4 - Número de diagramas | Uma interface pode ser entendida como um conjunto de operações que caracterizam um comportamento de um elemento, incluindo interfaces entre funções, objetivos, software sistemas e humanos (ISO/IEC, 2012). Em BPMN um diagrama uma classe que contém todo o conjunto de operações que caracterizam o comportamento de um determinado elemento, contendo as classes *Definitions* e *BaseElementos*, que por sua vez contém os demais elementos notacionais (OMG, 2013). Por similaridade, uma interface foi considerada um diagrama em BPMN. |
| 5 - Número de documentos | 5 - Número elementos com descrição<br>5.1 – Diagrama<br>5.2 – Atividade com descrição simples<br>5.3 – Atividade com descrição detalhada<br>5.4 – Evento de borda tipo erro<br>5.5 – Evento de fim tipo erro<br>5.6 - Mensagem | Um documento é uma unidade de informação identificada para uso humano, como um relatório, especificação, manual ou livro, em formato eletrônico ou impresso. Pode ser visto também como um item de documentação (ISO/IEC, 2012). Em BPMN há a classe *Documentation* que é na verdade um atributo da classe *BaseElement*, que a classe abstract de praticamente todos os elementos notacionais. Este atributo é utilizado para registrar as informações para os usuários do modelo, através de descrições que podem ser mais detalhadas (OMG, 2013). Por similaridade, um documento pode ser considerado uma descrição de elemento notacional BPMN. No entanto, foram selecionados apenas os elementos relacionados com os objetivos do trabalho na coluna QME Adaptada. |
| 6 - Número de componentes | 6 - Subprocesso reutilizável | Uma estrutura discreta, com um módulo, ou uma parte bem definida que compõe um sistema (ISO/IEC, 2012). Em BPMN um processo acionável (*callable*), ou processo reutilizável, é uma estrutura discreta e bem definida que se comporta com um módulo e que pode ser chamado por uma classe do tipo *CallActivity* a partir de qualquer outro ponto do modelo (OMG, 2013). Por similaridade, este tipo de subprocesso BPMN pode ser considerado um componente. |
| 7 - Volume de dados | 7 - Número de produtos de trabalho<br>7.1 – DataObject<br>7.2 – DataStore | Dados são representações de informação de maneira formal, adequadas para a comunicação, interpretação ou processamento, por humanos ou meios automáticos (ISO/IEC, 2012). Em BPMN as informações são representadas pelas classes *DataObjects* e *DataStore* (OMG, 2013). Por similaridade, os dados podem ser considerados um *DataObject* ou um *DataStore*. Em modelos de processo as informações, *data objects* e *data store* são utilizados para representar produtos de trabalho. |
| 8 - Número de tolerâncias a falhas | 8 - Número de elementos que detectam e tratam erros | A tolerância à falha se refere ao grau em que um sistema, produto ou componente continua operando adequadamente apesar da ocorrência de falhas (ISO/IEC, 2012). Em BPMN há mecanismos específicos para se detectar e tratar a ocorrência de |

| | | |
|---|---|---|
| | 8.1 – Evento de fim tipo erro<br>8.2 – Evento de borda tipo erro<br>3.4 – Evento de borda tipo compensação | falhas, tais como a classe *Events*, que possui eventos específicos para detecção de erros, como os eventos de fim tipo erro e evento de borda tipo erro, e o evento de borda tipo compensação, que aciona um tratamento para uma eventual falha. |

Após a identificação das QMEs da ISO aplicáveis ao modelo de processo de software, incluindo as devidas adaptações, resta apenas elencar essas QMEs dentro dos parâmetros estabelecidos no capítulo anterior, caracterizando-as definitivamente como QMEs provenientes da ISO, conforme apresentado na Tabela 24.

*Tabela 24. Conjunto de QMEs ISO.*

| Sigla | Nome | Entidade a medir | Elemento notacional a medir |
|---|---|---|---|
| NM | Número de mensagens | Diagrama de processo de software | Mensagem |
| NID | Número de itens de dados | Diagrama de processo de software | Objeto de dados<br>Objeto de armazenamento |
| NAM | Número de atividades do modelo | Modelo de processo de software | Atividade |
| NPNR | Número de subprocessos | Diagrama de processo de software | Subprocesso |
| NA | Número de atividades | Diagrama de processo de software | Atividade |
| ND | Número de diagramas | Modelo de processo de software | Diagrama |
| NDD | Número de diagramas com descrição | Modelo de processo de software | Diagrama |
| NAD | Número de atividades com descrição simples | Diagrama de processo de software | Atividade |
| NADD | Número de atividades com descrição detalhada | Diagrama de processo de software | Atividade |
| NBED | Número de eventos de borda tipo erro com descrição | Diagrama de processo de software | Atividade |
| NFED | Número de eventos de fim tipo erro com descrição | Diagrama de processo de software | Evento |
| NMD | Número de mensagens com descrição | Diagrama de processo de software | Mensagem |
| NPR | Número de subprocessos reutilizáveis | Diagrama de processo de software | Atividade de chamada |
| NPT | Número de produtos de trabalho | Modelo de processo de software | Objeto de dados<br>Objeto de armazenamento |
| NFE | Número de eventos de fim tipo erro | Diagrama de processo de software | Evento |
| NBE | Número de eventos de borda tipo erro | Diagrama de processo de software | Atividade |
| NBC | Número de eventos de borda tipo compensação | Diagrama de processo de software | Atividade |

### 5.2.5 Concatenar listas de QMEs

A atividade “Concatenar listas de QMEs”, tanto a proveniente da revisão de literatura quanto a proveniente da ISO, tem o objetivo de produzir uma lista única, integrada, e sem repetições, como pode ser observado na Tabela 25. Adicionalmente, nesta tabela, algumas siglas foram ajustadas para combinar com os nomes em português das QMEs.

*Tabela 25. QMEs definidas.*

| Sigla | Nome | Entidade a medir | Elemento a medir |
|---|---|---|---|
| NAP | Número de atividades que são predecessoras | Diagrama de processo de software | Atividade |
| NAS | Número de atividades que são sucessoras | Diagrama de processo de software | Atividade |
| NDA | Número de dependência entre atividades | Diagrama de processo de software | Atividade |
| NAPE | Número de atividades em que há produtos de trabalho de entrada | Diagrama de processo de software | Atividade |
| NAPES | Número de atividades em que há produtos de trabalho de saída | Diagrama de processo de software | Atividade |
| NAPS | Número de atividades em que há produtos de trabalho de entrada e de saída | Diagrama de processo de software | Atividade |
| NF | Número de fases | Diagrama de processo de software | Subprocesso do diagrama inicial |
| NP | Número de papéis | Diagrama de processo de software | Lane |
| NPRAT | Número de papéis responsáveis por atividades | Diagrama de processo de software | Lane |
| NPRAT(P) | Número de atividades cuja responsabilidade é do papel P | Diagrama de processo de software | Atividade e Lane |
| NPTU | Número de produtos de trabalho utilizados por atividades | Diagrama de processo de software | Data object e Atividade |
| NPTP | Número de produtos de trabalho produzidos por atividades | Diagrama de processo de software | Data object e Atividade |
| NPRA | Número total de produtos de trabalho relacionados a atividades | Diagrama de processo de software | Data object |

| | | | |
|---|---|---|---|
| NM | Número de mensagens | Diagrama de processo de software | Mensagem |
| NPT | Número de produtos de trabalho | Diagrama de processo de software | Objeto de dados<br>Objeto de armazenamento |
| NAM | Número de atividades do modelo | Modelo de processo de software | Atividade |
| NPNR | Número de subprocessos não reutilizáveis | Diagrama de processo de software | Subprocesso |
| NA | Número de atividades | Diagrama de processo de software | Atividade |
| ND | Número de diagramas | Modelo de processo de software | Diagrama |
| NDD | Número de diagramas com descrição | Modelo de processo de software | Diagrama |
| NAD | Número de atividades que possuem descrição | Diagrama de processo de software | Atividade |
| NADD | Número de atividades que possuem descrição detalhada | Diagrama de processo de software | Atividade |
| NBED | Número de eventos de borda tipo erro com descrição | Diagrama de processo de software | Atividade |
| NFED | Número de eventos de fim tipo erro com descrição | Diagrama de processo de software | Evento |
| NMD | Número de mensagens com descrição | Diagrama de processo de software | Mensagem |
| NPR | Número de subprocessos reutilizáveis | Diagrama de processo de software | Atividade de chamada |
| NFE | Número de eventos de fim tipo erro | Diagrama de processo de software | Evento |
| NBE | Número de eventos de borda tipo erro | Diagrama de processo de software | Atividade |
| NBC | Número de eventos de borda tipo compensação | Diagrama de processo de software | Atividade |

Algumas QMEs foram identificadas tanto na revisão de literatura quanto na ISO, de modo que foram mantidas as identificadas na ISO em detrimento das identificadas na revisão de literatura, como pode ser observado na Tabela 26.

*Tabela 26. QMEs da revisão de literatura que foram descartadas*

| Sigla | Nome | Entidade a medir | Elemento a medir |
|---|---|---|---|
| NA | Número de atividades | Diagrama de processo de software | Atividade |
| NSTP | Número de tarefas | Diagrama de processo de software | Atividade |
| NWP | Número de produtos de trabalho | Diagrama de processo de software | Data object |

### 5.2.6 Definir informações adicionais

Após o processo de concatenação das QMEs, foi executada a atividade "Definir informações adicionais", cujo objetivo foi completar o conjunto de informações necessárias para a correta coleta e quantificação das QMEs a partir do modelo de processo de software, conforme apresentado na Tabela 27, de acordo com a estrutura proposta no capítulo anterior.

*Tabela 27. QMEs definidas.*

| Sigla | Nome | Entidade a medir | Elemento a medir | Propriedade a medir | Unidade de medida | Método de medição | Procedimento de coleta |
|---|---|---|---|---|---|---|---|
| NAP | Número de atividades que são predecessoras | Diagrama de processo de software | Atividade | Classe *Task* em todos os seus tipos, com o atributo *outgoing* (herdade de *Activity*, herdade de *FlowNode*) preenchido; do pacote *Process*. Representação XML:<br>`<task id="" name="">`<br>`<outgoing> XXX </outgoing>`<br>`</task>` | Atividades predecessoras | Somar todas os elementos *Task* com atributo *outgoing* | Automática por software |
| NAS | Número de atividades que são sucessoras | Diagrama de processo de software | Atividade | Classe *Task* em todos os seus tipos, com o atributo *incoming* (herdade de *Activity*, herdade de *FlowNode*) preenchido; do pacote *Process*. Representação XML:<br>`<task id="" name="">`<br>`<incoming> XXX </incoming>`<br>`</task>` | Atividades sucessoras | Somar todos os objetos *Task* com atributo *incoming* | Automática por software |
| NDA | Número de dependência entre atividades | Diagrama de processo de software | Atividade | Classe *Task* em todos os seus tipos, com os atributos *incoming* e *outcoming* (herdade de *Activity*, herdado de *FlowNode*) preenchidos; do pacote *Process*. Representação XML:<br>`<task id="" name="">`<br>`<outgoing> XXX </outgoing>`<br>`<incoming> XXX </outgoing>`<br>`</task>` | Atividades ligadas | Somar todas os elementos *Task* com atributo *incoming* e/ou *outcoming* | Automática por software |
| NAPE | Número de atividades em que há produtos de trabalho de entrada | Diagrama de processo de software | Atividade | Classe "InputOutputSpecification" associada à classe "Activity", quando houver uma classe "DataInput" agregada a esta "InputOutputSpecification". A representação em XML é como segue:<br>`<task id="" name="">`<br>`<ioSpecification id="">`<br>`<dataInput id="" />`<br>`</ioSpecification>`<br>`</task>` | Atividades com produtos de trabalho de entrada | Somar todas os elementos *Task* ligados a uma entrada de dados | Automática por software |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| NAPES | Número de atividades em que há produtos de trabalho de saída | Diagrama de processo de software | Atividade | Classe “InputOutputSpecification” associada à classe “Activity”, quando houver uma classe “DataOutput” agregada a esta “InputOutputSpecification”. A representação em XML é como segue:<br><br>`<task id="" name="">`<br>`  <ioSpecification id="">`<br>`  <dataOutput id="" />`<br>`  </ioSpecification>`<br>`</task>` | Atividades com produtos de trabalho de saída | Somar todas os elementos *Task* ligados a uma saída de dados | Automática por software |
| NAPS | Número de atividades em que há produtos de trabalho de entrada e de saída | Diagrama de processo de software | Atividade | Classe “InputOutputSpecification” associada à classe “Activity”, quando houver uma classe “DataOutput” agregada a esta “InputOutputSpecification”. A representação em XML é como segue:<br><br>`<task id="" name="">`<br>`  <ioSpecification id="">`<br>`  <dataInput id="" />`<br>`  <dataOutput id="" />`<br>`  </ioSpecification>`<br>`</task>` | Atividades com produtos de trabalho de entrada e saída | Somar todas os elementos *Task* ligados a uma entrada e a uma saída de dados | Automática por software |
| NF | Número de fases | Diagrama de processo de software | Subprocesso do diagrama inicial | Classe *CallActivity*, uma herança da Classe *Activity*, em uma associação com a classe *CallableElement*. Do pacote *Process*. Contar aqueles apenas do diagrama com o menor *process id* (principal). A representação em XML é como segue:<br><br>`<callActivity id="" name="">`<br>`</callActivity>` | Subprocessos reutilizáveis do diagrama principal. | Somar todos os elementos *callActivity* do diagrama principal | Automática por software |
| NP | Número de papéis | Diagrama de processo de software | Lane | Classe “Lane”, associada à classe “LaneSet”, do pacote “Process” da notação BPMN, determinado pela seguinte estrutura XML:<br><br>`<laneSet id="">`<br>`  <lane id="" name=" ">`<br>`  <childLaneSet id="" />`<br>`  </lane>`<br>`</laneSet>` | Papéis | Somar todos os elementos *Lane* do diagrama | Automática por software |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| NPRAT | Número de papéis responsáveis por atividades | Diagrama de processo de software | Atividade e Lane | Agrupamento da classe “Task” por cada classe “Lane”. A ligação se dá pelo elemento *BPMNShape*, ao final do XML do diagrama.<br><br>`<task id="" name="">`<br>`</task>`<br><br>`<lane>`<br>`</lane>` | Papéis com atividades | Somar todos os elementos *Task* do diagrama que esteja contido por um elemento *Lane* | Automática por software |
| NPRAT(P) | Número de atividades cuja responsabilidade é do papel P | Diagrama de processo de software | Atividade e Lane | Agrupamento da classe “Task” por cada classe “Lane” especificada como P. A ligação se dá pelo elemento *BPMNShape*, ao final do XML do diagrama.<br><br>`<task id="" name="">`<br>`</task>`<br><br>`<lane>`<br>`</lane>` | Atividades por papel | Somar todos os elementos *Task* do diagrama que esteja contido por um elemento *Lane*, específicos de um determinado papel P. | Automática por software |
| NPTU | Número de produtos de trabalho utilizados por atividades | Diagrama de processo de software | Atividade | Classe “InputOutputSpecification” associada à classe “Activity”, quando houver uma classe “DataOutput” agregada a esta “InputOutputSpecification”. A representação em XML é como segue:<br><br>`<task id="" name="">`<br>`  <ioSpecification id="">`<br>`  <dataInput id="" />`<br>`  </ioSpecification>`<br>`</task>` | Produtos de trabalho utilizados | Somar todos os atributos *dataInput* dos elementos *Task* | Automática por software |
| NPTP | Número de produtos de trabalho produzidos por atividades | Diagrama de processo de software | Atividade | Classe “InputOutputSpecification” associada à classe “Activity”, quando houver uma classe “DataOutput” agregada a esta “InputOutputSpecification”. A representação em XML é como segue:<br><br>`<task id="" name="">`<br>`  <ioSpecification id="">`<br>`  <dataOutput id="" />` | Produtos de trabalho produzidos | Somar todos os atributos *dataOutput* dos elementos *Task* | Automática por software |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | `</ioSpecification>`<br>`</task>` | | | |
| NPRA | Número total de produtos de trabalho relacionados a atividades | Diagrama de processo de software | Atividade | Classe “InputOutputSpecification” associada à classe “Activity”, quando houver uma classe “DataOutput” agregada a esta “InputOutputSpecification”. A representação em XML é como segue:<br>`<task id="" name="">`<br>`<ioSpecification id="">`<br>`<dataInput id="" />`<br>`<dataOutput id="" />`<br>`</ioSpecification>`<br>`</task>` | Produtos de trabalho relacionados a atividades | Somar todos os atributos *dataInput* e *dataOutput* dos elementos *Task* | Automática por software |
| NM | Número de mensagens | Diagrama de processo de software | Mensagem | Classe *MessageFlow* do pacote *Collaboration*, com a seguinte representação XML:<br>`<messageFlow id="" name="" sourceRef="" targetRef="" />` | Mensagens | Somar todos os elementos *messageFlow* | Automática por software |
| NPT | Número de produtos de trabalho | Diagrama de processo de software | Objeto de dados<br>Objeto de armazenamento | Classe *DataObject* e classe *DataStore*, ambas do pacote *Process*, representados em XML da seguinte forma:<br>`<dataObject id="" name="">`<br>`<dataState id="" name="" />`<br>`</dataObject>`<br>`<dataStore id="" name="">`<br>`<dataState id="" name="" />`<br>`</dataStore>` | Produtos de trabalho | Somar todos os elementos *dataObject* e *dataStore* | Automática por software |
| NAM | Número de atividades do modelo | Modelo de processo de software | Atividade | Classe Task em todos os seus tipos. Representação XML:<br>`<task id=”” name=””>`<br>`</task>`<br>`<sendTask id=”” name=””>`<br>`</sendTask>`<br>`<receiveTask id=”” name=””>`<br>`</receiveTask>` | Atividades | Somar todos os elementos *Task* e suas especializações *k* | Automática por software |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | <serviceTask id=”” name=””><br></serviceTask><br><userTask id=”” name=””><br></userTask><br><manualTask id=”” name=””><br></manualTask><br><scriptTask id=”” name=””><br></scriptTask><br><businessRuleTask id=”” name=””><br></businessRuleTask> | | | |
| NPNR | Número de subprocessos não reutilizáveis | Diagrama de processo de software | Subprocesso | Classe *SubProcess* do pacote *Process*. Representação XML:<br><br><subProcess id="" name=""><br></subProcess> | Subprocessos | Somar todos os elementos *subProcess* | Automática por software |
| NA | Número de atividades | Diagrama de processo de software | Atividade | Classe “Task” e todas as suas especializações, ou seja, “SendTask”, “ReceiveTask”, “ServiceTask”, “UserTask”, “ManualTask”, “ScriptTask” e “BusinesRuleTask”. A representação XML dessas classes é como segue:<br><br><task id=”” name=””><br></task><br><sendTask id=”” name=””><br></sendTask><br><receiveTask id=”” name=””><br></receiveTask><br><serviceTask id=”” name=””><br></serviceTask><br><userTask id=”” name=””><br></userTask><br><manualTask id=”” name=””><br></manualTask><br><scriptTask id=”” name=””><br></scriptTask><br><businessRuleTask id=”” name=””><br></businessRuleTask> | Atividades | Somar todos os elementos *Task* e suas especializações | Automática por software |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ND | Número de diagramas | Modelo de processo de software | Diagrama | Pacote *Collaboration*. Este pacote contém todos os demais elementos BPMN, representando assim um diagrama BPMN completo.<br><br>`<collaboration id="" name="">`<br>`</collaboration>` | Diagramas | Somar todos os elementos *collaboration* | Automática por software |
| NDD | Número de diagramas com descrição | Modelo de processo de software | Diagrama | Classe *Documentation* associada ao pacote *Collaboration*.<br><br>`<collaboration id="" name="">`<br>`<documentation>`<br>`XXX`<br>`</documentation>`<br>`</collaboration>` | Diagramas com descrição | Somar todos os elementos *collaboration* que contenham o atributo *documentation* | Automática por software |
| NAD | Número de atividades que possuem descrição | Diagrama de processo de software | Atividade | Classe “Task” e todas as suas especializações, ou seja, “SendTask”, “ReceiveTask”, “ServiceTask”, “UserTask”, “ManualTask”, “ScriptTask” e “BusinesRuleTask”. Serão contados apenas os elementos com o atributo *name* preenchido. A representação XML dessas classes é como segue:<br><br>`<task id=”” name=”XXX”>`<br>`</task>`<br>`<sendTask id=”” name=”XXX”>`<br>`</sendTask>`<br>`<receiveTask id=”” name=”XXX”>`<br>`</receiveTask>`<br>`<serviceTask id=”” name=”XXX”>`<br>`</serviceTask>`<br>`<userTask id=”” name=”XXX”>`<br>`</userTask>`<br>`<manualTask id=”” name=”XXX”>`<br>`</manualTask>`<br>`<scriptTask id=”” name=”XXX”>`<br>`</scriptTask>`<br>`<businessRuleTask id=”” name=””>`<br>`</businessRuleTask>` | Atividades com descrição simples | Somar todos os elementos *Task* e suas especializações, desde que contenham o atributo *name* | Automática por software |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| NADD | Número de atividades que possuem descrição detalhada | Diagrama de processo de software | Atividade | Classe *Documentation* (associação herdada de *BaseElement*, que é herdada por *Activity* e consequentemente por todas as *Task*), do pacote *Foundation*, apenas quando associada a uma classe *Task*. Representação XML:<br><br>`<task id=”” name=”XXX”><br><documentation><br></documentation></task><br><sendTask id=”” name=”XXX”><br><documentation><br></documentation></sendTask><br><receiveTask id=”” name=”XXX”><br><documentation><br></documentation></receiveTask><br><serviceTask id=”” name=”XXX”><br><documentation><br></documentation></serviceTask><br><userTask id=”” name=”XXX”><br><documentation><br></documentation></userTask><br><manualTask id=”” name=”XXX”><br><documentation><br></documentation></manualTask><br><scriptTask id=”” name=”XXX”><br><documentation><br></documentation></scriptTask><br><businessRuleTask id=”” name=””><br><documentation><br></documentation><br></businessRuleTask>` | Atividades com descrição detalhada | Somar todos os elementos *Task* e suas especializações, desde que contenham o atributo *documentation* | Automática por software |
| NBED | Número de eventos de borda tipo erro com descrição | Diagrama de processo de software | Atividade | Classe *SequenceFlow* do pacote *Common*, com o atributo *name* preenchido. Representação XML:<br><br>`<sequenceFlow id="" name="XXX" sourceRef=""><br></sequenceFlow>` | Eventos de borda de erro | Contar todas os elementos *Task* que contenham um elemento *boundaryEvent* com um elemento *sequenceFlow* associado, desde que esse *sequenceFlow* contenha o atributo *name*. | Automática por software |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| NFED | Número de eventos de fim tipo erro com descrição | Diagrama de processo de software | Evento | Classe *EndEvent* (quer herda de *ThrowEvent*, que herda de *Event*) com o atributo *name* preenchido e com o atributo *errorEventDefinition* preenchido. Do pacote *Process*. Representação XML:<br>`<endEvent id="" name="XXX">`<br>`<errorEventDefinition id="" />`<br>`</endEvent>` | Eventos de fim tipo erro | Somar todos os elementos *endEvent* com o atributo *errorEventDefinition*, desde que contenham também o atributo *name* | Automática por software |
| NMD | Número de mensagens com descrição | Diagrama de processo de software | Mensagem | Classe *MessageFlow* do pacote *Collaboration*, que tenham o atributo *name* preenchido. Representação XML:<br>`<messageFlow id="" name="XXX" sourceRef="" targetRef="" />` | Mensagens com descrição | Somar todos os elementos *messageFlow* com o atributo *name* | Automática por software |
| NPR | Número de subprocessos reutilizáveis | Diagrama de processo de software | Atividade de chamada | Classe *CallActivity*, uma herança da Classe *Activity*, em uma associação com a classe *CallableElement*. Do pacote *Process*. A representação em XML é como segue:<br>`<callActivity id="" name="">`<br>`</callActivity>` | Subprocesso reutilizável | Somar todos os elementos *callActivity* | Automática por software |
| NFE | Número de eventos de fim tipo erro | Diagrama de processo de software | Evento | Classe *EndEvent* (quer herda de *ThrowEvent*, que herda de *Event*) com o atributo *errorEventDefinition* preenchido. Do pacote *Process*. Representação XML:<br>`<endEvent id="" name="">`<br>`<errorEventDefinition id="" />`<br>`</endEvent>` | Evento de fim tipo erro | Somar todos os elementos endEvent com o atributo *errorEventDefinition* | Automática por software |
| NBE | Número de eventos de borda tipo erro | Diagrama de processo de software | Evento de borda | Classe *BoundaryEvent* (que herda de *CatchEvent*), com o atributo *errorEventDefintion* setado.<br>`<boundaryEvent id="" attachedToRef="">`<br>`<errorEventDefinition id="" />`<br>`</boundaryEvent>` | Evento de borda tipo erro | Somar todos os elementos *boundaryEvent* que contenham o atributo *errorEventDefinition* | Automática por software |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| NBC | Número de eventos de borda tipo compensação | Diagrama de processo de software | Evento de borda | Classe *BoundaryEvent* (que herda de *CatchEvent*), com o atributo *compensateEventDefintion* setado.<br><br>`<boundaryEvent id="" attachedToRef="">`<br>`  <compensateEventDefinition id="" />`<br>`</boundaryEvent>` | Eventos de borda tipo compensação | Somar todos os elementos *boundaryEvent* que contenha o atributo *compensateEventDefinition* | Automática por software |

## 5.3 Definir medidas de qualidade

Após a execução do subprocesso “Definir elementos de medida de qualidade”, o processo subsequente é “Definir medidas de qualidade”, conforme apresentado no capítulo anterior na Figura 13, que resulta em um conjunto de “QMs definidas”. A execução de cada uma das atividades desse subprocesso é apresentada a seguir.

### 5.3.1 Identificar QMs através de revisão da literatura

Esta atividade tem o objetivo de identificar as medidas derivadas com possibilidade de uso como QMs para a composição do modelo de medição da qualidade de modelos de processo de software. A revisão de literatura apresentada no início do presente capítulo identificou tanto as medidas básicas quanto as medidas derivadas. As medidas derivadas identificadas são apresentadas na Tabela 28.

*Tabela 28. Medidas derivadas identificadas na revisão da literatura.*

| Sigla | Medida derivada | Fórmula | Artigo |
|---|---|---|---|
| RSTPA | Média de tarefas por atividades. | NSTP/NA | (GARCIA, RUIZ, *et al.*, 2003) |
| PRA | Razão entre atividades predecessoras e total de atividades | NPD/NA | (GARCIA, RUIZ, *et al.*, 2003) |
| PSA | Razão entre atividades sucessoras e total de atividades | NSD/NA | (GARCIA, RUIZ, *et al.*, 2003) |
| NDWP | Número de dependência (ligação) entre produtos de trabalho e atividades | NDWPin + NDWPout | (GARCÍA, RUIZ e PIATTINI, 2004) (GARCIA, RUIZ, *et al.*, 2003) (DA SILVA, MACIEL e RAMALHO, 2013) (CANFORA, GARCÍA, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006) (GARCÍA, PIATTINI, *et al.*, 2005) |
| NAWP | Número de atividades relacionadas com produtos de trabalho | NAWPIn + NAWPOut- NAWPInOut | (GARCIA, RUIZ, *et al.*, 2003) |
| RDWPA | Razão entre atividades relacionadas a produtos de trabalho e o número total de atividades | NAWP / NA | (GARCIA, RUIZ, *et al.*, 2003) (GARCÍA, RUIZ e PIATTINI, 2004b) |
| NCA | Acoplamento das atividades no processo | NA / NDA | (GARCÍA, RUIZ e PIATTINI, 2004) (GARCIA, RUIZ, *et al.*, 2003) (DA SILVA, MACIEL e RAMALHO, 2013) (GARCÍA, RUIZ e PIATTINI, 2004b) (CANFORA, GARCÍA, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006) (GARCÍA, PIATTINI, *et al.*, 2005) |

| | | | |
|---|---|---|---|
| RDWPin | Razão entre os produtos utilizados pelas atividades e o total de produtos no processo | NDWPin / NDWP | (GARCÍA, RUIZ e PIATTINI, 2004) (GARCIA, RUIZ, *et al.*, 2003) (DA SILVA, MACIEL e RAMALHO, 2013) (GARCÍA, RUIZ e PIATTINI, 2004b) (CANFORA, GARCÍA, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006) (GARCÍA, PIATTINI, *et al.*, 2005) |
| RDWPout | Razão entre os produtos produzidos pelas atividades e o total de produtos no processo | NDWPout / NDWP | (GARCÍA, RUIZ e PIATTINI, 2004) (GARCIA, RUIZ, *et al.*, 2003) (DA SILVA, MACIEL e RAMALHO, 2013) (GARCÍA, RUIZ e PIATTINI, 2004b) (CANFORA, GARCÍA, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006) (GARCÍA, PIATTINI, *et al.*, 2005) |
| RDWPInOut | Razão entre total de produtos relacionados a atividades e o total de produtos do modelo | NDWPInOut / NWP | (GARCIA, RUIZ, *et al.*, 2003) |
| RWPA | Razão entre produtos de trabalho e atividades | NWP / NA | (GARCÍA, RUIZ e PIATTINI, 2004) (GARCIA, RUIZ, *et al.*, 2003) (DA SILVA, MACIEL e RAMALHO, 2013) (CANFORA, GARCÍA, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006) (GARCÍA, PIATTINI, *et al.*, 2005) |
| RRPA | Razão entre papéis do processo e atividades | NRP / NA | (GARCÍA, RUIZ e PIATTINI, 2004) (GARCIA, RUIZ, *et al.*, 2003) (GARCÍA, RUIZ e PIATTINI, 2004b) (CANFORA, GARCÍA, *et al.*, 2005) (CANFORA, GARCIA, *et al.*, 2006) (GARCÍA, PIATTINI, *et al.*, 2005) |
| RRPR(R) | Média de reponsabilidades de um papel do processo. | NAPR(R) / NA | (GARCIA, RUIZ, *et al.*, 2003) |

As medidas básicas utilizadas na composição das medidas derivadas na coluna "fórmula", conforme proposto pelos artigos considerados, estão descritas abaixo:

- NA: Número de atividades
- NPD: Número de atividades que são predecessoras
- NSD: Número de atividades que são sucessoras
- NDA: Número de dependência entre atividades
- NSTP: Número de tarefas
- NAWPIn: Número de atividades em que há produtos de trabalho de entrada

- NAWPOut: Número de atividades em que há produtos de trabalho de saída
- NAWPInOut: Número de atividades em que há produtos de trabalho de entrada e de saída
- NoP: Número de fases
- NoI: Número de iterações
- NWP: Número de produtos de trabalho
- NPR: Número de papéis
- NR: Número de papéis responsáveis por atividades
- NAPR(R): Número de atividades cuja responsabilidade é do papel R
- NDWPin: Número de produtos de trabalho dos utilizados por atividades
- NDWPout: Número de produtos de trabalho (*work product*) produzidos por atividades
- NDWPInOut: Número total de produtos de trabalho relacionados a atividades

No entanto, os trabalhos identificados na revisão de literatura não buscaram uma forte relação com padrões de qualidade existentes, tais como aqueles propostos pela família de normas ISO/IEC 25.000. Estes trabalhos estão em uma fase da pesquisa onde a maior preocupação é estabelecer um conjunto de medidas básicas e derivadas, e aplicar estas medidas a modelos de processo de software.

Ainda, essas medidas derivadas não são relacionadas de maneira direta à um modelo de qualidade com subcaracterísticas e características da qualidade. Por exemplo, (GARCIA, RUIZ, *et al.*, 2003) propõe medidas derivadas, tais como:

- Proporção entre passos e atividades, que se daria pela fórmula $X = \frac{NP}{NA}$ onde NP seria o número de passos e NA o número de atividades.
- Acoplamento do modelo, apresentado pela fórmula $X = \frac{NA}{NDA}$ onde NA seria o número de atividades e NDA o número de dependências entre atividades.
- Proporção entre produtos e atividades, definido pela fórmula $X = \frac{NPT}{NA}$ onde NPT seria o número de produtos de trabalho e NA o número de atividades.
- Proporção entre o número de papéis e o número de atividades, definido pela fórmula $X = \frac{NP}{NA}$ onde NP seria o número de papéis e NA o número de atividades.

Em função disso, as medidas derivadas não puderam ser classificadas como QMs, uma vez que seus objetivos não foram claramente definidos para fins de composição de

características ou subcaracterísticas de qualidade. De fato, essas medidas foram utilizadas para a compor a medida “Complexidade” do modelo de processo de software. Neste sentido, os trabalhos de (GARCIA, RUIZ, *et al.*, 2003), (GARCÍA, RUIZ e PIATTINI, 2004) (GARCÍA, RUIZ e PIATTINI, 2004b) apresentam a “Complexidade” como uma medida ligada a características de qualidade, tais como “Compreensibilidade” e “Manutenibilidade”. Sendo assim, a partir da revisão de literatura selecionamos a medida derivada “Complexidade” como uma QM, conforme Tabela 29.

*Tabela 29. Conjunto de QMs provenientes da revisão.*

| Sigla | Nome | Descrição |
|---|---|---|
| C | Complexidade | Complexidade do modelo de processo de software. |

### 5.3.2 Identificar QMs na ISO 25023 aplicáveis ao modelo de processo de software

Esta atividade tem o objetivo de identificar QMs aplicáveis ao modelo de processo de software a partir do conjunto de QMs proposto pela ISO/IEC 25.023. O conjunto de QMs considerado foi aquele apresentado na cláusula 8 da norma ISO/IEC 25.023, que embora não seja uma lista exaustiva é um conjunto inicial a ser avaliado quando à aplicabilidade (ISO/IEC, 2016).

O Anexo A desta norma foi considerado para a aplicação dos critérios de seleção das QMs, definido no capítulo anterior. Neste Anexo as QMs são classificadas por nível de recomendação, que pode ser HR (*highly recomended*/altamente recomendado), R (*recommended*/recomendado) e UD (*user’s discrection*/discrição do usuário). Esta última classificação (UD) se refere a QMs propostas que podem ser usadas como referência para a criação de novas QMs, de acordo com a decisão do usuário do modelo, mas a confiabilidade dessas QMs é ainda desconhecida.

Os critérios de seleção definidos no capítulo anterior, na Tabela 12, são aplicados e demonstrados na Tabela 30. Porém, o critério S2 (Estar dentro do escopo de avaliação pretendido) desta tabela foi aplicado a priori, de modo que apenas as QMs relacionadas com as características Usabilidade e Manutenibilidade estão presentes na mesma.

*Tabela 30. Critérios aplicados às QMs de Usabilidade e Manutenibilidade da ISO 25.023.*

| QM ISO 25023 | Classificação | Seleção |
|---|---|---|
| Completude da descrição | HR | Selecionado |
| Cobertura de demonstração | UD | Não atende a S1 |

| | | |
|---|---|---|
| Autodescrição dos pontos de entrada | UD | Não atende a S1 |
| Completude da orientação ao usuário | HR | Selecionado |
| Preenchimento automático de campos | R | Excluído por E2 |
| Compreensibilidade das mensagens de erro | R | Selecionado |
| Interface de usuário autoexplicativa | UD | Não atende a S1 |
| Consistência operacional | HR | Excluído por E2 |
| Clareza de mensagem | R | Selecionado |
| Funcionalidade customizável | UD | Não atende a S1 |
| Interface de usuário customizável | UD | Não atende a S1 |
| Monitoramento da capacidade | UD | Não atende a S1 |
| Possibilidade de desfazer | R | Selecionado |
| Categorização compreensível da informação | R | Excluído por E2 |
| Consistência de aparência | UD | Não atende a S1 |
| Suporte aos dispositivos de entrada | UD | Não atende a S1 |
| Evitar erro de operação do usuário | HR | Excluído por E2 |
| Correção de erros de entrada | HR | Excluído por E2 |
| Recuperabilidade de erros do usuário | R | Selecionado |
| Aparência agradável das interfaces de usuário | UD | Não atende a S1 |
| Acessibilidade para usuários com deficiência | R | Excluído por E2 |
| Suporte de idiomas | UD | Não atende a S1 |
| Acoplamento dos componentes | R | Selecionado |
| Adequação da complexidade ciclomática | UD | Não atende a S1 |
| Reusabilidade dos ativos | HR | Selecionado |
| Conformidade com regras de codificação | R | Excluído por E2 |
| Completude dos logs do sistema | HR | Excluído por E2 |
| Efetividade da função de diagnóstico | R | Excluído por E2 |
| Suficiência da função de diagnóstico | R | Excluído por E2 |
| Eficiência na modificação | HR | Excluído por E2 |
| Correção na modificação | HR | Excluído por E2 |
| Capacidade de modificação | UD | Não atende a S1 |
| Completude da função de teste | R | Excluído por E2 |
| Testabilidade autônoma | UD | Não atende a S1 |
| Capacidade de reiniciar testes após manutenção | UD | Não atende a S1 |

### 5.3.3 Avaliar possibilidade de adaptação

As QMs que não podem ser aplicadas diretamente ao modelo de processo de software passam pela atividade “Avaliar possibilidade de adaptação”, que verifica se a QM proveniente da ISO pode ser adequada à medição do modelo do processo de software. As QMs selecionadas e que podem ser adaptadas para aplicação ao modelo de processo de software são apresentadas na Tabela 31. Como pode ser observado, todas as QMs

selecionadas, embora aplicáveis ao modelo de processo de software, precisam receber algum tipo de ajuste.

*Tabela 31. QMs da ISO após aplicação do critério para identificação de QMs candidatas à adaptação.*

| # | QM ISO 25023 | Ação (Conforme Tabela 13) |
|---|---|---|
| 1 | Completude da descrição | Adaptável |
| 2 | Completude da orientação ao usuário | Adaptável |
| 3 | Compreensibilidade das mensagens de erro | Adaptável |
| 4 | Clareza de mensagem | Adaptável |
| 5 | Possibilidade de desfazer | Adaptável |
| 6 | Recuperabilidade de erros do usuário | Adaptável |
| 7 | Acoplamento dos componentes | Adaptável |
| 8 | Reusabilidade dos ativos | Adaptável |

### 5.3.4 Adaptar QM

A atividade de adaptação de QMs considera duas premissas: 1) a adaptação deve ser feita para atender ao modelo de processo de software; 2) a adaptação deve considerar as particularidades da notação utilizada para a representação do processo. As adaptações aplicadas às QMs selecionadas estão apresentadas na Tabela 32.

*Tabela 32. QMs da ISO após adaptação.*

| # | QM ISO 25023 | QM adaptada | Considerações |
|---|---|---|---|
| 1 | Completude da descrição | Existência da descrição dos diagramas | Aplicado originalmente à descrição do produto e de seus documentos, foi aplicado à descrição do diagrama que é a estrutura que contém os elementos notacionais que documentam o processo. |
| 2 | Completude da orientação ao usuário | Existência da descrição das atividades | Aplicado originalmente à descrição das funcionalidades do sistema, foi aplicado às atividades que representam as funcionalidades (trabalho) do processo. |

| | | | |
|---|---|---|---|
| 3 | Compreensibilidade das mensagens de erro | Existência da descrição dos eventos de erro | Aplicado originalmente à mensagens de erro do software que orientam o usuário do software quanto a como proceder, foi aplicado aos eventos de erro no modelo, que orientam o usuário do modelo a como prosseguir com o processo em caso de erros. |
| 4 | Clareza de mensagem | Existência da descrição da mensagem | Aplicado originalmente à mensagens entre o software e o usuário, foi aplicado à mensagens entre o subprocesso e os agentes externos. |
| 5 | Possibilidade de desfazer | Possibilidade de desfazer trabalho | Aplicado originalmente à possibilidade de desfazer uma ação no software, foi aplicado a possibilidade de desfazer trabalho realizado em um modelo. |
| 6 | Recuperabilidade de erros do usuário | Recuperabilidade de erros na execução do processo | Aplicado originalmente à possibilidade de recuperação de erros pelo software, foi aplicado à previsão de recuperação de erros pelo processo. |
| 7 | Acoplamento dos componentes | Acoplamento de subprocessos | Aplicado originalmente ao possível acoplamento de componentes do software, foi aplicado ao possível acoplamento de subprocessos. |
| 8 | Reusabilidade dos ativos | Reusabilidade de subprocessos | Aplicado originalmente à reusabilidade de componentes do software, foi aplicado a reusabilidade de subprocessos. |

Após a identificação das QMs da ISO aplicáveis ao modelo de processo de software, incluindo as devidas adaptações, a Tabela 33 apresenta a lista com essas QMs.

*Tabela 33. Conjunto de QMs provenientes da ISO.*

| Sigla | Nome | Descrição |
|---|---|---|
| CDM | Existência da descrição dos diagramas | Existência de descrição nos diagramas, contribuindo para determinar a proporção de diagramas que receberam descrições de orientação para o usuário. |
| COU | Existência da descrição das atividades | Existência de descrição adicional nas atividades (além da descrição básica da atividade), contribuindo para determinar a proporção de atividades que oferecem orientação adicional para o usuário. |

| | | |
|---|---|---|
| CME | Existência da descrição dos eventos de erro | Existência de mensagem descritiva nos eventos de erro, contribuindo para avaliar a proporção de erros que oferecem alguma orientação ao usuário. |
| CM | Existência da descrição da mensagem | Existência de mensagem descritiva, contribuindo para verificar se as mensagens oferecem alguma descrição de orientação para o usuário. |
| PD | Possibilidade de desfazer trabalho | Possibilidade de desfazer trabalhos já realizados em caso de erro, orientando o usuário sobre que trabalhos precisam ser desfeitos nestes casos. |
| REU | Recuperabilidade de erros na execução do processo | Recuperabilidade de erros no processo, com a previsão de ações a serem tomadas em caso da ocorrência de determinados erros durante a execução do processo pelo usuário. |
| AC | Acoplamento de processos | Acoplamento de processos, identificando os processos que guardam maior independência do restante dos processos do modelo. |
| RA | Reusabilidade de processos | Reusabilidade de processos, identificando os processos que podem ser reutilizados pelo modelo, ou até por outros modelos. |

### 5.3.5 Concatenar lista de QMs

A atividade “Concatenar listas de QMs” integra os dois conjuntos de QMs, tanto a proveniente da revisão da literatura quanto a proveniente da ISO/IEC 25.023. Esta lista pode ser vista a seguir, Tabela 33.

*Tabela 34. QMs aplicáveis.*

| Sigla | Nome | Descrição |
|---|---|---|
| CDM | Existência da descrição dos diagramas | Existência de descrição nos diagramas, contribuindo para determinar a proporção de diagramas que receberam descrições de orientação para o usuário. |
| COU | Existência da descrição das atividades | Existência de descrição adicional nas atividades (além da descrição básica da atividade), contribuindo para determinar a proporção de atividades que oferecem orientação adicional para o usuário. |
| CME | Existência da descrição dos eventos de erro | Existência de mensagem descritiva nos eventos de erro, contribuindo para avaliar a proporção de erros que oferecem alguma orientação ao usuário. |
| CM | Existência da descrição da mensagem | Existência de mensagem descritiva, contribuindo para verificar se as mensagens oferecem alguma descrição de orientação para o usuário. |
| PD | Possibilidade de desfazer trabalho | Possibilidade de desfazer trabalhos já realizados em caso de erro, orientando o usuário sobre que trabalhos precisam ser desfeitos nestes casos. |
| REU | Recuperabilidade de erros na execução do processo | Recuperabilidade de erros no processo, com a previsão de ações a serem tomadas em caso da ocorrência de determinados erros durante a execução do processo pelo usuário. |
| AC | Acoplamento de processos | Acoplamento de processos, identificando os processos que guardam maior independência do restante dos processos do modelo. |
| RA | Reusabilidade de processos | Reusabilidade de processos, identificando os processos que podem ser reutilizados pelo modelo, ou até por outros modelos. |
| C | Complexidade | Complexidade do modelo de processo de software. |

### 5.3.6 Definir informações adicionais

Após o processo de concatenação das QMs, foi executada a atividade "Definir informações adicionais", que tem o objetivo de completar as informações necessárias para a adequada quantificação das QMs a partir das QMEs coletadas do modelo de processo de software, de acordo com a estrutura definida na Tabela 7, e conforme apresentado a seguir.

#### *5.3.6.1 Usabilidade*

Medidas de usabilidade são usadas para avaliar o grau em que um modelo de processo de software pode ser usado por usuários específicos para alcançar determinados objetivos com eficiência, eficácia e satisfação em um contexto de uso especificado.

##### 5.3.6.1.1 SRA - Reconhecimento apropriado

O usuário consegue identificar um modelo de processo de software adequado aos seus objetivos. As medidas de reconhecimento apropriado são usadas para avaliar o grau em que os usuários conseguem compreender se um modelo de processo de software é apropriado para suas necessidades ou não.

| Sigla | Nome | Descrição | Função de medição |
|---|---|---|---|
| CDM | Existência da descrição dos diagramas | Existência de descrição nos diagramas, contribuindo para determinar a proporção de diagramas que receberam descrições de orientação para o usuário.<br>Os valores podem variar entre 0 e 1, sendo 1 o número ideal, ou seja, que todos os diagramas possuam uma descrição que possibilite ao usuário compreender seu uso no contexto dos demais diagramas, ou seja, do modelo. | $\frac{\sum_{1}^{dm} \frac{NDD}{ND}}{dm}$ |
| NDD – Número de diagramas com descrição.<br>ND – Número de diagramas.<br>dm – Diagramas do modelo. | | | |

#### 5.3.6.1.2 SA - Apreensibilidade

Medidas de Apreensibilidade são usadas para avaliar o grau em que um modelo de processo de software pode ser usado por determinados usuários para atingir a objetivos de aprendizado específicos de uso do modelo com eficiência, eficácia, satisfação e livre de riscos.

| Sigla | Nome | Descrição | Função de medição |
|---|---|---|---|
| COU | Existência da descrição das atividades | Existência de descrição adicional nas atividades (além da descrição básica da atividade), contribuindo para determinar a proporção de atividades que oferecem orientação adicional para o usuário.<br>Os valores podem variar entre 0 e 1, sendo 1 o número ideal, pois todas as atividades deveriam conter descrições que fornecessem ajuda adicional ao usuário para utilizar o modelo. | $\frac{\sum_1^{dm} \frac{\text{NADD}}{\text{NA}}}{\text{dm}}$ |
| NADD – Número de atividades que possuem descrição detalhada.<br>ND – Número de atividades.<br>dm – Diagramas do modelo. | | | |

| Sigla | Nome | Descrição | Função de medição |
|---|---|---|---|
| CME | Existência da descrição dos eventos de erro | Existência de mensagem descritiva nos eventos de erro, contribuindo para avaliar a proporção de erros que oferecem alguma orientação ao usuário.<br>Os valores podem variar entre 0 e 1, sendo 1 o número ideal, uma vez que seria apropriado que todos os erros estivessem associados a uma descrição.<br>O cálculo da média de CME considera apenas os diagramas que possuam tratamento de erro, uma vez que não é obrigatório que cada diagrama possua pelo menos um tratamento de erro. O objetivo de CME não é avaliar o grau de tratamento de erros no modelo, e sim se os que são aplicados estão descritos. | $\frac{\sum_1^{dmte} \frac{\text{NBE} + \text{NFE}}{\text{NBED} + \text{NFED}}}{\text{dmte}}$ |
| NBE – Número de eventos de borda tipo erro.<br>NFE – Número de eventos de fim tipo erro.<br>NBED – Número de eventos de borda tipo erro com descrição.<br>NFED – Número de eventos de fim tipo erro com descrição.<br>dmte – Diagramas do modelo que possuem tratamento de erro. | | | |

#### 5.3.6.1.3 SO - Operabilidade

Medidas de operabilidade são usadas para avaliar o grau em que um modelo de processo de software possui características que tornam viável a sua operação e controle.

| Sigla | Nome | Descrição | Função de medição |
|---|---|---|---|
| CM | Existência da descrição da mensagem | Existência de mensagem descritiva, contribuindo para verificar se as mensagens oferecem alguma descrição de orientação para o usuário.<br>O valor pode variar entre 0 e 1, sendo 1 o número ideal, pois todas as mensagens deveriam estar descritas.<br>O cálculo da média de CM considera apenas os diagramas que possuam troca de mensagens, o que não é obrigatório em todos os diagramas. CM busca identificar se as mensagens estão ou não descritas. | $\frac{\sum_{1}^{dmm} \frac{NMD}{NM}}{dmm}$ |
| NMD – Número de mensagens com descrição.<br>NM – Número de mensagens.<br>dmm – Diagramas do modelo que possuem troca de mensagens. | | | |

| Sigla | Nome | Descrição | Função de medição |
|---|---|---|---|
| PD | Possibilidade de desfazer trabalho | Possibilidade de desfazer trabalhos já realizados em caso de erro, orientando o usuário sobre que trabalhos precisam ser desfeitos nestes casos.<br>O valor pode variar entre 0 e 1. No entanto, não é possível ainda definir o número ideal de ações previstas para desfazer atividades (*compensation*). Porém, é razoável esperar que este número seja maior do que 0. | $\frac{\sum_{1}^{dm} \frac{NBC}{NA}}{dm}$ |
| NBC – Número de eventos de borda tipo compensação.<br>NA – Número de atividades.<br>dm – Diagramas do modelo. | | | |

#### 5.3.6.1.4 SP - Proteção contra erros do usuário

Medidas de proteção contra erros do usuário são usadas para avaliar o grau em que um modelo de processo de software contribui para que erros sejam evitados e tratados durante sua utilização.

| Sigla | Nome | Descrição | Função de medição |
|---|---|---|---|
| REU | Recuperabilidade de erros na execução do processo | Recuperabilidade de erros no processo, com a prevenção de ações a serem tomadas em caso da ocorrência de determinados erros durante a execução do processo pelo usuário.<br>O valor pode variar entre 0 e 1. No entanto, não é possível ainda definir o número ideal de previsões para tratamento de erros e de compensações em caso de um processo. Porém, é razoável esperar que este número seja maior do que 0. | $\frac{\sum_{1}^{dm} \frac{NFE + NBC}{NA}}{dm}$ |
| NFE – Número de eventos de fim tipo erro.<br>NBC – Número de eventos de borda tipo compensação.<br>NA – Número de atividades.<br>dm – Diagramas do modelo. | | | |

### *5.3.6.2 Manutenibilidade*

Medidas de Manutenibilidade são usadas para avaliar o grau de eficiência e eficácia com que um modelo de processo de software pode ser modificado por seus mantenedores.

#### 5.3.6.2.1 SM – Modularidade

Medidas de modularidade são usadas para avaliar o grau em que um modelo de processo de software é composto por componentes discretos de modo que as mudanças em um componente tenham menor impacto em outros componentes.

| Sigla | Nome | Descrição | Função de medição |
|---|---|---|---|
| AC | Acoplamento de processos | Acoplamento de processos, identificando os processos que guardam maior independência do restante dos processos do modelo.<br>O valor pode variar entre 0 e 1. O valor ideal é 1, quando todos os componentes são independentes e não sofrem impacto das mudanças em outros componentes.<br>A média de AC considera apenas os diagramas que possuem subprocessos (reutilizáveis ou não), uma vez que há diagramas que não possuem qualquer processo a ser considerado. | $\frac{\sum_{1}^{dmp} \frac{NPR}{NPNR + NPR}}{dmp}$ |
| NPR – Número de subprocessos reutilizáveis.<br>NPNR – Número de subprocessos não reutilizáveis.<br>dmp – Diagramas do modelo que possuem processos, reutilizáveis ou não. | | | |

#### 5.3.6.2.2 SR – Reusabilidade

Medidas de Reusabilidade são usadas para avaliar o grau em que existem subprocessos que podem ser utilizados em outros processos ou para compor outros subprocessos reutilizáveis.

| Sigla | Nome | Descrição | Função de medição |
|---|---|---|---|
| RA | Reusabilidade de processos | Reusabilidade de processos, identificando os processos que podem ser reutilizados pelo modelo, ou até por outros modelos.<br>O valor pode variar entre 0 e 1. No entanto, não é possível ainda definir um número ideal de componentes reutilizáveis. Porém, espera-se que este número seja maior que 0, e que seja tanto melhor quanto mais próximo de 1. | $\sum_{1}^{dm} \frac{[\![(\frac{NPR}{NPR + NPNR + NA})]\!]}{dm}$ |
| NPR – Número de subprocessos reutilizáveis.<br>NPNR – Número de subprocessos não reutilizáveis.<br>NA – Número de atividades.<br>dm – Diagramas do modelo. | | | |

#### 5.3.6.2.3 SMD - Modificabilidade

Medidas de Modificabilidade são usadas para avaliar o grau em que um modelo de processo de software pode ser eficiente e eficazmente modificado sem reduzir a qualidade deste modelo.

| Sigla | Nome | Descrição | Função de medição |
|---|---|---|---|
| C | Complexidade | Complexidade do modelo de processo de software.<br>O valor pode variar entre 0 e 1. Embora não seja possível ainda definir um número ideal, espera-se que esse número seja tanto melhor quanto mais próximo de 1, e tanto pior quanto mais próximo de 0. | $\frac{\sum_{1}^{dm} 1 - \left(\frac{NPTU + NPTP}{NA + NPTU + NPTP}\right)}{dm}$ |
| NPTU – Número de produtos de trabalho utilizados por atividades.<br>NPTP – Número de produtos de trabalho produzidos por atividades.<br>NA – Número de atividades.<br>dm – Diagramas do modelo. | | | |

| Sigla | Nome | Descrição | Função de medição |
|---|---|---|---|
| AC | Acoplamento de processos | Acoplamento de processos, identificando os processos que guardam maior independência do restante dos processos do modelo.<br>O valor pode variar entre 0 e 1. O valor ideal é 1, quando todos os componentes são independentes e não sofrem impacto das mudanças em outros componentes.<br>A média de AC considera apenas os diagramas que possuem processos (reutilizáveis ou não), uma vez que há diagramas que não possuem qualquer processo a ser considerado. | $\frac{\sum_{1}^{dmp} \frac{NPR}{NPNR + NPR}}{dmp}$ |
| NPR – Número de subprocessos reutilizáveis.<br>NPNR – Número de subprocessos não reutilizáveis.<br>dmp – Diagramas do modelo que contenham processos, reutilizáveis ou não. | | | |

Cada uma das QMs apresentadas está relacionada, por um lado com características de qualidade, e por outro lado com as QMEs que serão coletadas a partir do modelo de processo de software.

## 5.4 Definir modelo de medição

A atividade “Definir modelo de medição” tem o objetivo de correlacionar QMs às subcaracterísticas de qualidade, que por sua vez estão correlacionas às características de qualidade, conforme apresentado na Figura 13.

A partir dos subprocessos “Definir elementos de medida de qualidade” e “Definir medidas de qualidade”, apresentados na mesma figura, é possível apresentar as correlações entre as características de qualidade, subcaracterísticas de qualidade, QMs e QMEs, conforme Figura 16.

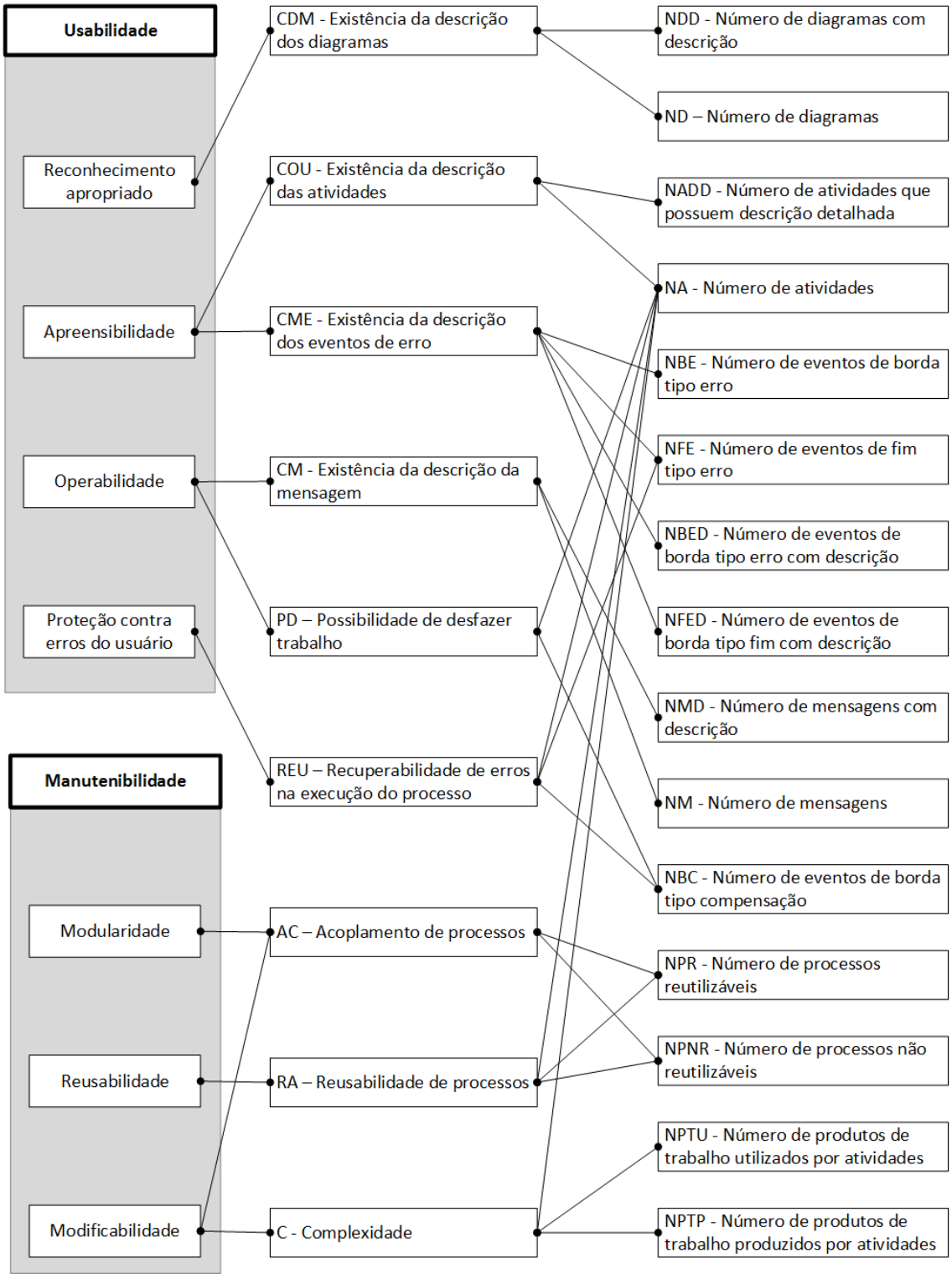



*Figura 16. Modelo de medição para o modelo de processo de software.*

## 5.5 Conclusão do capítulo

Neste capítulo foi executado o processo de construção de um modelo de medição para avaliação do modelo de processo de software. O primeiro passo foi a definição dos elementos de medida de qualidade, que resultou em um conjunto de QMEs aplicáveis ao

modelo de processo de software. Neste passo, as QMEs foram definidas a partir da revisão de literatura e a partir de uma proposta inicial de QMEs da norma ISO/IEC 25.021.

Em seguida, foram definidas as medidas de qualidade, resultado em um conjunto de QMs aplicáveis ao modelo de processo de software. Estas QMs também foram obtidas da revisão de literatura e de uma proposta inicial QMs da norma ISO/IEC 25.023. Após isso, foi definido um modelo de medição, que apresenta as relações entre características de qualidade, subcaracterísticas de qualidade, QMs e QMEs.

# 6 Proposta de um processo de avaliação para modelos de processo de software representados em BPMN

## 6.1 Introdução do capítulo

O objetivo deste capítulo é propor um processo para avaliação da qualidade do modelo de processo de software. Este processo de avaliação está baseado na família de normas internacionais ISO/IEC 25.000 (ISO/IEC, 2014), mais especificamente na norma ISO/IEC 25.040 (ISO/IEC, 2011). Embora este conjunto de normas seja orientado à avaliação da qualidade de produtos de software e de sistemas, sua estrutura pode ser utilizada para a avaliação da qualidade de modelos de processo de software.

## 6.2 O processo de avaliação

O processo de avaliação de modelos de processo de software está representado em notação BPMN (OMG, 2013), em dois níveis de abstração, sendo o primeiro apresentado na Figura 17. Uma descrição de cada um dos subprocessos é apresentada na Tabela 35.

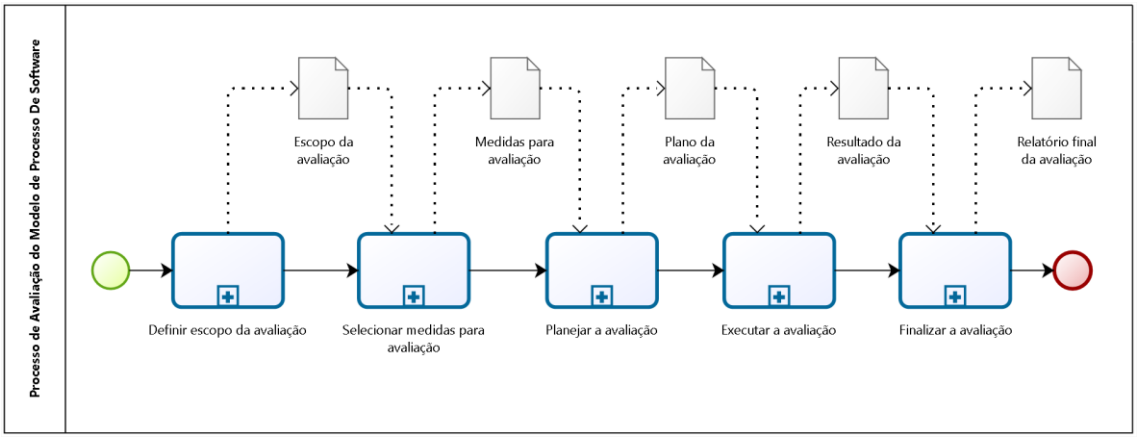



*Figura 17. Subprocessos do processo de avaliação do modelo de processos de software.*

*Tabela 35. Descrição dos subprocessos do processo de avaliação.*

| Processo | Descrição |
|---|---|
| Definir escopo da avaliação | Estabelecer o propósito, características de qualidade que serão consideradas na avaliação, e partes do modelo que serão avaliadas. Este conjunto de informações comporá o escopo da avaliação, para aprovação do solicitante da avaliação. |
| Selecionar medidas para avaliação | Selecionar as medidas de qualidade, o plano de ações para essas medidas e os critérios para agregação do resultado. |

| | |
|---|---|
| Planejar a avaliação | Elaborar o plano de avaliação, incluindo a estimativa de tempo para realização da avaliação. |
| Executar a avaliação | Realizar as medições, organizar os resultados e aplicar o plano de ações elaborado a partir das medidas coletadas, resultando em recomendações. |
| Finalizar a avaliação | Elaborar o relatório da avaliação e entregar a documentação. |

### 6.2.1 Definir escopo da avaliação

O objetivo deste subprocesso é definir o escopo da avaliação, considerando o propósito da avaliação, indicando as características de qualidade que serão consideradas e identificando as partes do modelo que serão avaliadas.

O subprocesso é apresentado na Figura 18 e descrito na Tabela 36. Neste subprocesso há uma comunicação entre o solicitante da avaliação e o avaliador, onde o avaliador recebe a solicitação de avaliação e depois solicita a aprovação do solicitante para o escopo definido.

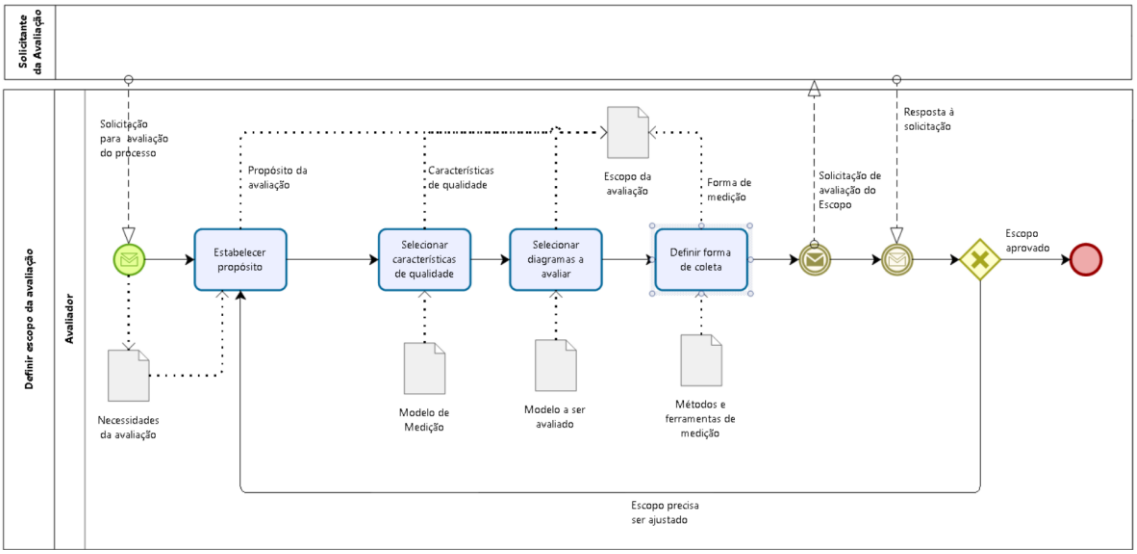



*Figura 18. Subprocesso "Definir escopo da avaliação".*

*Tabela 36. Descrição do subprocesso "Definir escopo da avaliação".*

| Atividade | Descrição |
|---|---|
| Estabelecer o propósito | Estabelecer o propósito da avaliação, considerando a necessidade apresentada pelo solicitante. |
| Selecionar características de qualidade | Selecionar as características de qualidade que serão consideradas na avaliação. O conjunto de características e subcaracterísticas a ser considerado é aquele definido no capítulo 5, composto das características Usabilidade e Manutenibilidade e suas respectivas subcaracterísticas. |
| Selecionar diagramas a avaliar | Selecionar os diagramas do modelo que serão avaliados. Caso haja exclusões, estas devem ser justificadas. |
| Definir forma de coleta | Definir a forma de coleta das medidas de qualidade. |

Este subprocesso produz o documento “Escopo da avaliação” que deve obrigatoriamente conter os itens apresentados na Tabela 37.

*Tabela 37. Itens que devem constar do documento “Escopo da avaliação”.*

| Campo | Descrição |
|---|---|
| Propósito da avaliação | Objetivo da avaliação, incluindo os resultados esperados. |
| Características de qualidade | Características de qualidade selecionadas para serem avaliadas. |
| Diagramas a avaliar | Diagramas do modelo de processo em BPMN selecionados para avaliação. |
| Procedimento de medição | Procedimento utilizado para realizar a medição, que pode ser manual, automática ou integrar estas duas formas. |

### 6.2.2 Selecionar medidas para avaliação

O objetivo deste subprocesso é selecionar as medidas de qualidade, o plano de ações para essas medidas e os critérios para agrupamento do resultado. O subprocesso é apresentado na Figura 19 e detalhado na Tabela 38. Uma vez que o modelo de medição utilizado é aquele apresentado no Capítulo 5, há uma comunicação entre o processo de avaliação e o processo “Elaborar modelo de medição”, definido no Capítulo 4.

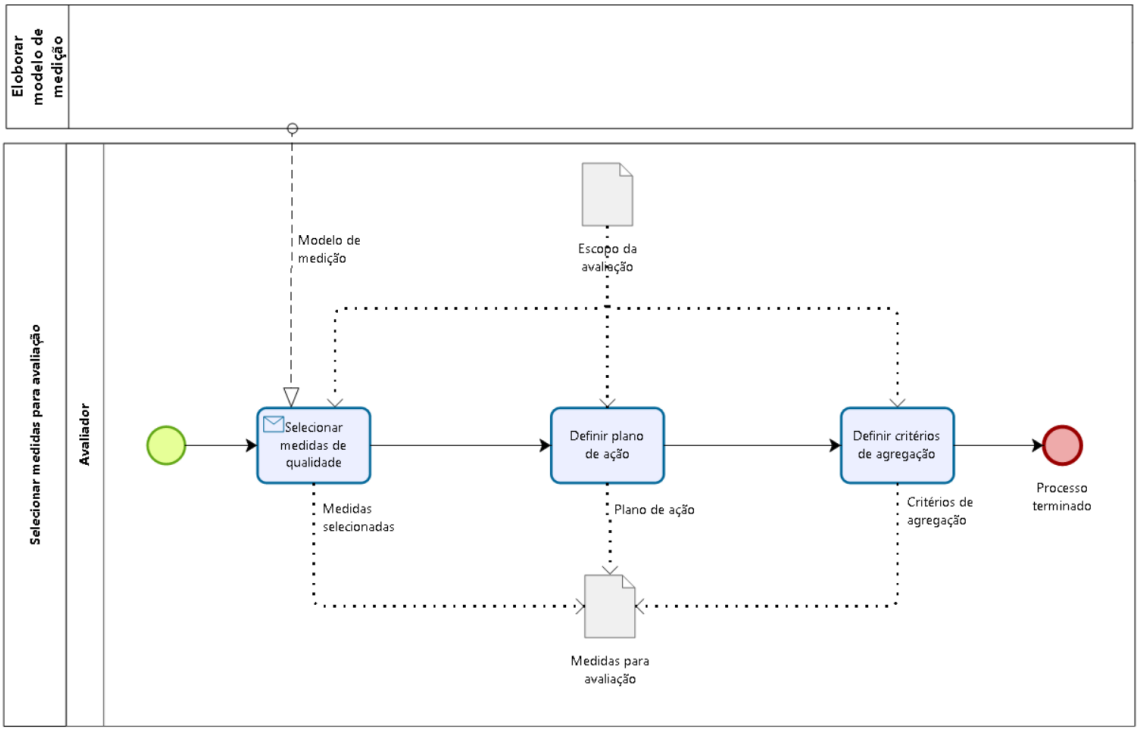



*Figura 19. Subprocesso " Selecionar medidas para avaliação".*

*Tabela 38. Detalhamento do subprocesso " Selecionar medidas para avaliação".*

| Atividade | Descrição |
|---|---|
| Selecionar medidas de qualidade | Selecionar as medidas de qualidade que serão utilizadas para avaliar o modelo do processo de software, a partir do modelo de medição produzido pelo processo "Elaborar modelo de medição". |
| Definir plano de ação | Definir plano de ação para as medidas individuais, podendo utilizar para isso benchmarks, dados históricos, requisitos dos usuários ou dos clientes, entre outros elementos de apoio para este fim. O plano de ação é associado às QMs. |
| Definir critérios para agregação | Definir critérios que serão considerados para a agregação das características e subcaracterísticas de qualidade. |

Este subprocesso produz o documento “Medidas para avaliação” que deve conter obrigatoriamente os itens apresentados na Tabela 39.

*Tabela 39. Itens que devem constar do documento “Medidas para avaliação”.*

| Item | Descrição |
|---|---|
| Medidas selecionadas | Lista de medidas a serem coletadas no modelo, indicando o ponto do modelo a ser verificado para a coleta da medida. |
| Plano de ação | As ações a serem realizadas para cada medição. Estas ações são definidas com base em parâmetros numéricos ou objetivos. |
| Agregação do resultado | Critérios para agregação do resultado. |

### 6.2.3 Planejar a avaliação

O objetivo deste subprocesso é elaborar o plano de avaliação, incluindo o cronograma. Este subprocesso pode ser observado na Figura 20, e uma descrição para cada atividade é apresentada na Tabela 40.

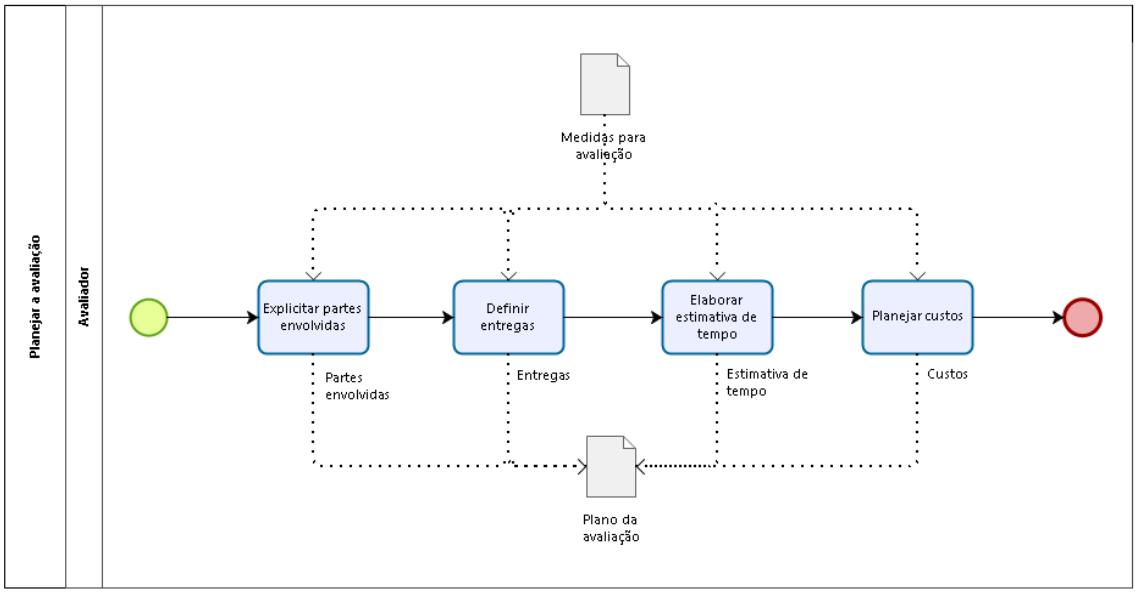



*Figura 20. Subprocesso "Planejar a avaliação".*

*Tabela 40. Descrição do subprocesso "Planejar a avaliação".*

| Atividade | Descrição |
|---|---|
| Explicitar partes envolvidas | Explicitar os atores que estarão envolvidos no processo de avaliação e suas responsabilidades. |
| Definir entregas | Estruturar as entregas a serem realizadas pela avaliação. |
| Elaborar estimativa de tempo | Com base nas entregas definidas, estabelecer o conjunto das atividades necessárias para a realização de cada uma dessas entregas, bem como as interdependências entre estas atividades e os tempos estimados para cada uma delas. |
| Planejar custos | Planejar o custo para a realização da avaliação, considerando a necessidade de pessoas, ferramentas, equipamentos, viagens e demais recursos eventualmente necessários. |

Este subprocesso produz o documento “Plano a avaliação”, que deve conter minimamente os itens apresentados na Tabela 41.

*Tabela 41. Itens que devem constar do documento "Plano da avaliação".*

| Item | Descrição |
|---|---|
| Partes envolvidas | Explicitação dos atores envolvidos e sua relação com a avaliação. |
| Entregas | Produtos de informação esperados a partir da execução do plano de avaliação. |
| Estimativa de tempo | Lista das atividades a serem realizadas durante a avaliação, com suas interdependências e tempos estimados. |
| Custo estimado | Estimativa de custo para o projeto de avaliação. |

### 6.2.4 Executar a avaliação

O objetivo deste subprocesso é realizar as medições, organizar os resultados e aplicar o plano de ações elaborado a partir das medidas coletadas, resultando em recomendações. O subprocesso é apresentado na Figura 21 e a descrição das atividades consta da Tabela 42.

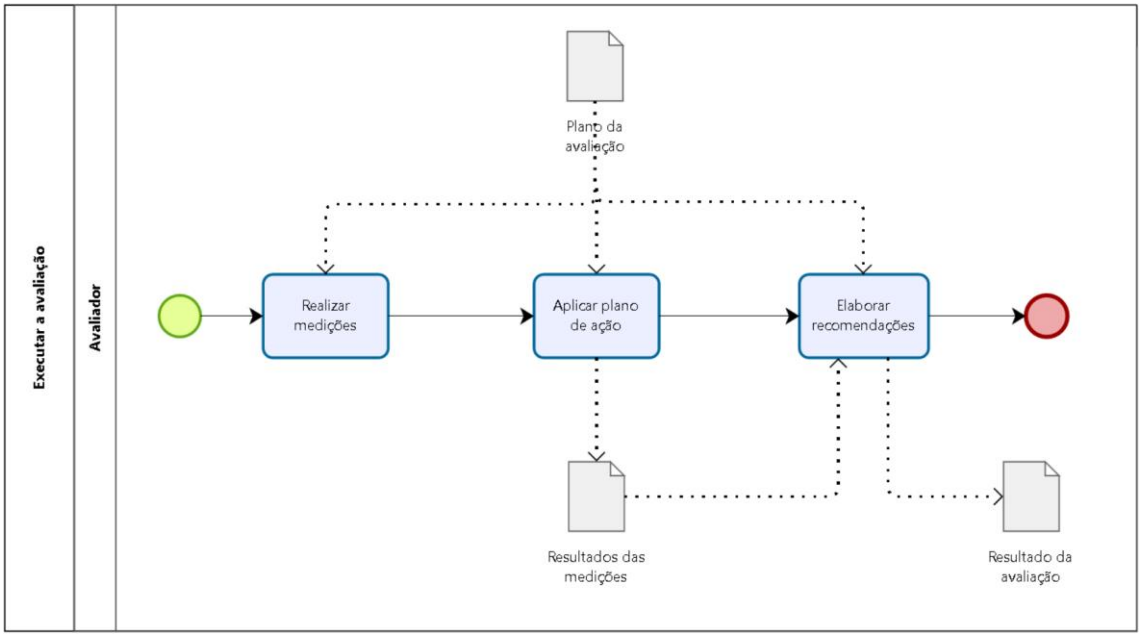



*Figura 21. Subprocesso "Executar a avaliação".*

*Tabela 42. Descrição do subprocesso "Executar a avaliação".*

| Atividade | Descrição |
|---|---|
| Realizar medições | Realizar as medições no modelo selecionado de acordo com o plano de avaliação. |
| Aplicar planos de ação | Aplicar plano de ação para as medições realizadas. |
| Elaborar recomendações | Com base nos resultados das medições recomendar eventuais ajustes ao modelo. |

Este subprocesso produz o documento “Resultado da avaliação”, que deve conter obrigatoriamente os itens apresentados na Tabela 43.

*Tabela 43. Itens que devem constar do documento “Resultado da avaliação”.*

| Item | Descrição |
|---|---|
| Característica | Nome da característica de qualidade. |
| Subcaracterística | Sigla e nome da subcaracterística de qualidade. |
| Valor | Valor atribuído à subcaracterística após a realização das medições. |
| Análise | Análise do valor em relação ao modelo. |
| Recomendação | Recomendação resultante da análise. |

### 6.2.5 Finalizar a avaliação

O objetivo deste subprocesso é elaborar o relatório da avaliação e entregar a documentação. O subprocesso é apresentado na Figura 22. Neste subprocesso há uma comunicação com um agente externo, ou seja, o “Solicitante da Avaliação”. Este agente externo recebe o relatório de avaliação. As descrições das atividades são apresentadas na Tabela 44.

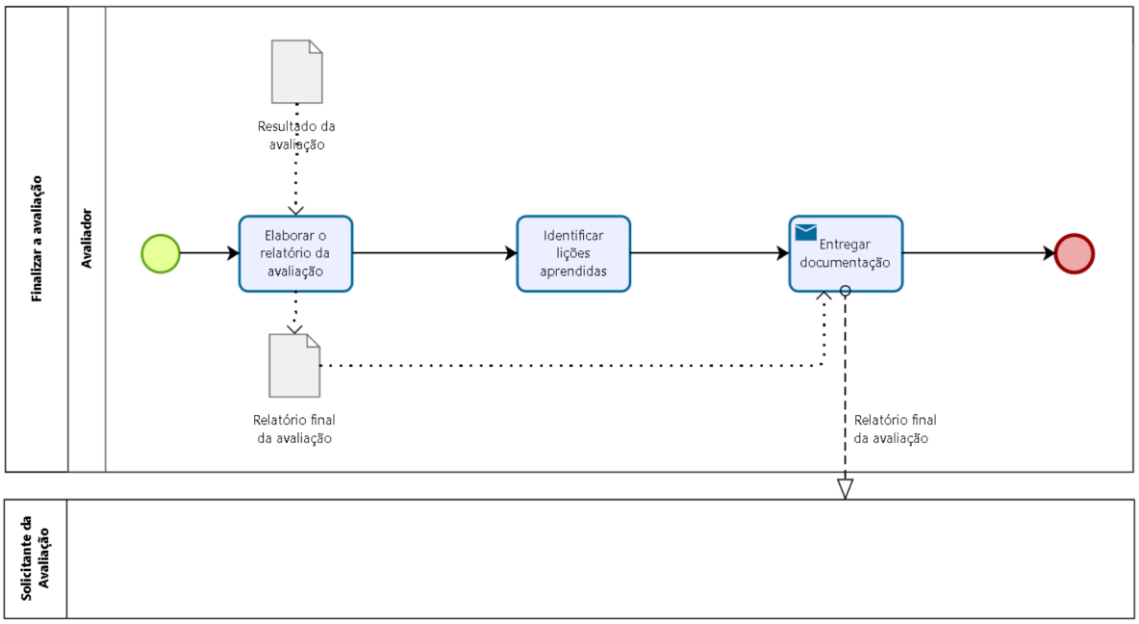



*Figura 22. Subprocesso "Finalizar a avaliação".*

*Tabela 44. Detalhamento do subprocesso "Finalizar a avaliação".*

| Atividade | Descrição |
|---|---|
| Elaborar o relatório da avaliação | Elaborar o relatório final da avaliação, de acordo com estrutura definida para este documento. |
| Identificar lições aprendidas | Avaliar o processo de avaliação e identificar as lições aprendidas. Deve ser registrado um feedback sobre o processo de avaliação a fim de melhorar o processo de avaliação continuamente. |
| Entregar documentação | Todos os documentos do solicitante utilizados na avaliação são devolvidos, bem como os resultados são entregues. |

Este subprocesso produz o documento “Relatório final da avaliação”, que deve conter obrigatoriamente os itens apresentados na Tabela 45.

*Tabela 45. Itens que devem constar do documento "Relatório final da avaliação".*

| Item | Objetivo |
|---|---|
| Organização onde foi realizada a avaliação | Identificação da organização. |
| Modelo avaliado | Identificação do modelo de processo de software avaliado. |
| Versão | Identificação da versão do modelo avaliado, seja por numeração da versão ou pela data da publicação da mesma. |
| Período da avaliação | Indicativo do período no qual a avaliação foi realizada. |
| Diagramas avaliados | Relação dos diagramas que foram considerados na avaliação. |
| Considerações sobre o resultado | Análise final a respeito da avaliação. |
| Curso de ação decidido | Curso de ação geral a ser adotado em decorrência da avaliação. |
| Ajustes recomendados pela avaliação | Ajustes propostos para aplicação no modelo avaliado. |
| Limitações/Restrições | Limites e restrições da avaliação realizada. |

## 6.3 Conclusão do capítulo

Neste capítulo foi proposto um processo para a avaliação de modelos de processo de software. Este processo foi baseado na família de normas ISO/IEC 25.000, mais especificamente na norma ISO/IEC 25.040.

O processo apresentado é composto de 5 subprocessos, que ao serem executados resultam na produção dos seguintes documentos principais: “Escopo da Avaliação”, “Medidas para avaliação”, “Plano da avaliação”, “Resultado da avaliação” e “Relatório Final da Avaliação”.

# 7 Execução do processo de avaliação para modelos de processo de software

## 7.1 Introdução do capítulo

No capítulo 5 foi proposto um conjunto de medidas para avaliação de modelos de processo de software quanto às características de qualidade “Usabilidade” e “Manutenibilidade”. No capítulo 6 foi proposto um processo de avaliação da qualidade do processo de software.

Neste capítulo é executado o processo para a avaliação de um modelo de processo real com o objetivo de avaliar a viabilidade e utilidade das medidas definidas. A avaliação é realizada na versão original do modelo, produzindo recomendações de ajuste. Após a implementação destes ajustes é realizada uma nova avaliação, e os resultados são comparados.

Para viabilizar a coleta automática de medidas foi desenvolvida a ferramenta **SDPQualityForms**.

Com o objetivo de avaliar a viabilidade de aplicação do processo de avaliação em um caso real foi selecionado, por conveniência, o modelo do processo de desenvolvimento de software da Fundação Oswaldo Cruz, ou Fiocruz. A Fiocruz é uma organização que tem o objetivo de produzir, disseminar e compartilhar conhecimentos e tecnologias voltados para o fortalecimento e a consolidação do Sistema Único de Saúde (SUS) e que contribuam para a promoção da saúde e da qualidade de vida da população brasileira, para a redução das desigualdades sociais e para a dinâmica nacional de inovação, tendo a defesa do direito à saúde e da cidadania ampla como valores centrais (FIOCRUZ, 2017). É constituída de 17 Unidades de Negócio, chamadas de Unidades técnico-científicas, que estão distribuídas nos Estados do Rio de Janeiro, Amazonas, Bahia, Minas Gerais, Paraná, Pernambuco, além de uma Unidade na África, em Moçambique. Cada uma dessas Unidades possui sua estrutura de Tecnologia da Informação, e diversas delas desenvolvem ou compram o desenvolvimento de sistemas de informação. Em função disso, a Fiocruz iniciou um projeto de criação de um Modelo de Desenvolvimento de Software (MDS) unificado para toda a instituição, com vistas ao aumento da padronização, eficiência, eficácia e interoperabilidade de seus diversos sistemas informatizados. A MDS-Fiocruz foi, portanto, objeto de avaliação utilizando a proposta deste trabalho.

## 7.2 Ferramenta de apoio à avaliação

No intuito de automatizar a coleta de medidas a partir do modelo do processo de software foi construída uma ferramenta específica, que chamamos de **SDPQualityForms**. Essa ferramenta analisa modelos de processo em formato BPMN, tanto as informações semânticas quanto as informações visuais do diagrama.

A notação BPMN é suportada por uma estrutura formal em formato XMI (OMG, 2017). Esta estrutura é chamada de BPMN DI (*BPMN Data Interchange*), e tem por objetivo facilitar a troca de diagramas entre ferramentas de modelagem de processos. Este mecanismo viabiliza a representação não ambígua de todos os elementos constantes no metamodelo BPMN. O metamodelo BPMN DI é definido com base no MOF (OMG Meta-Object Facility (MOF) Core Specification, Version 2.4.1). O BPMN DI é na verdade uma evolução de um metamodelo anterior, o BPMN DD, e a serialização do BPMN DI pode ser feita por meio de arquivos XML (OMG, 2013). O esquema XML do BPMN DI pode ser visto a seguir:

```
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
xmlns:bpmndi=http://www.omg.org/spec/BPMN/20100524/DI
xmlns:dc=http://www.omg.org/spec/DD/20100524/DC
xmlns:di=http://www.omg.org/spec/DD/20100524/DI
targetNamespace=http://www.omg.org/spec/BPMN/20100524/DI
elementFormDefault="qualified" attributeFormDefault="unqualified">

   <xsd:import namespace="http://www.omg.org/spec/DD/20100524/DC"
   schemaLocation="DC.xsd" />
   <xsd:import namespace="http://www.omg.org/spec/DD/20100524/DI"
   schemaLocation="DI.xsd" />
   <xsd:element name="BPMNDiagram" type="bpmndi:BPMNDiagram" />
   <xsd:element name="BPMNPlane" type="bpmndi:BPMNPlane" />
   <xsd:element name="BPMNLabelStyle" type="bpmndi:BPMNLabelStyle" />
   <xsd:element name="BPMNShape" type="bpmndi:BPMNShape"
   substitutionGroup="di:DiagramElement" />
   <xsd:element name="BPMNLabel" type="bpmndi:BPMNLabel" />
   <xsd:element name="BPMNEdge" type="bpmndi:BPMNEdge"
   substitutionGroup="di:DiagramElement" />

   <xsd:complexType name="BPMNDiagram">
      <xsd:complexContent>
            <xsd:extension base="di:Diagram">
                  <xsd:sequence>
                       <xsd:element ref="bpmndi:BPMNPlane" />
                       <xsd:element ref="bpmndi:BPMNLabelStyle" maxOccurs="unbounded"
                       minOccurs="0" />
                  </xsd:sequence>
            </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="BPMNPlane">
      <xsd:complexContent>
            <xsd:extension base="di:Plane">
                  <xsd:attribute name="bpmnElement" type="xsd:QName" />
            </xsd:extension>
```

```
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="BPMNEdge">
   <xsd:complexContent>
         <xsd:extension base="di:LabeledEdge">
               <xsd:sequence>
                     <xsd:element ref="bpmndi:BPMNLabel" minOccurs="0" />
                </xsd:sequence>
                <xsd:attribute name="bpmnElement" type="xsd:QName" />
                <xsd:attribute name="sourceElement" type="xsd:QName" />
                <xsd:attribute name="targetElement" type="xsd:QName" />
                <xsd:attribute name="messageVisibleKind"
                type="bpmndi:MessageVisibleKind" />
         </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="BPMNShape">
    <xsd:complexContent>
               <xsd:extension base="di:LabeledShape">
                     <xsd:sequence>
                       <xsd:element ref="bpmndi:BPMNLabel" minOccurs="0" />
               </xsd:sequence>
               <xsd:attribute name="bpmnElement" type="xsd:QName" />
               <xsd:attribute name="isHorizontal" type="xsd:boolean" />
               <xsd:attribute name="isExpanded" type="xsd:boolean" />
                <xsd:attribute name="isMarkerVisible" type="xsd:boolean" />
                <xsd:attribute name="isMessageVisible" type="xsd:boolean" />
                <xsd:attribute name="participantBandKind"
                type="bpmndi:ParticipantBandKind" />
                <xsd:attribute name="choreographyActivityShape" type="xsd:QName"/>
         </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="BPMNLabel">
    <xsd:complexContent>
               <xsd:extension base="di:Label">
                     <xsd:attribute name="labelStyle" type="xsd:QName" />
         </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="BPMNLabelStyle">
    <xsd:complexContent>
               <xsd:extension base="di:Style">
               <xsd:sequence>
                     <xsd:element ref="dc:Font" />
               </xsd:sequence>
         </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="ParticipantBandKind">
   <xsd:restriction base="xsd:string">
   <xsd:enumeration value="top_initiating" />
   <xsd:enumeration value="middle_initiating" />
   <xsd:enumeration value="bottom_initiating" />
   <xsd:enumeration value="top_non_initiating" />
   <xsd:enumeration value="middle_non_initiating" />
   <xsd:enumeration value="bottom_non_initiating" />
   </xsd:restriction>
```

```
    </xsd:simpleType>

    <xsd:simpleType name="MessageVisibleKind">
       <xsd:restriction base="xsd:string">
       <xsd:enumeration value="initiating" />
       <xsd:enumeration value="non_initiating" />
       </xsd:restriction>
    </xsd:simpleType>

</xsd:schema>
```

A partir destes elementos fundamentais, todos os elementos da notação BPMN em um diagrama podem ser representados em XML, possibilitando sua serialização, análise e troca entre aplicações e ferramentas.

A ferramenta **SDPQualityForms** analisa arquivos de extensão BPMN, que são produzidos a partir da implementação do BPMN DI. Deste modo, é possível o processamento de um conjunto de arquivos representativos dos diagramas que compõem um modelo de processo de software.

Para isso, a **SDPQualityForms**, conta com a classe **QME**, que trata das medidas diretamente coletadas dos diagramas, com a classe **QM**, que trata da análise das QMEs e da composição das QMs, ou medidas indiretas, da classe **Subcaracterística**, que trata da composição das subcaracterísticas a partir das QMs, e da classe **Característica**, que trata da composição das características a partir das subcaracterísticas. Estas classes podem ser vistas na Figura 23.

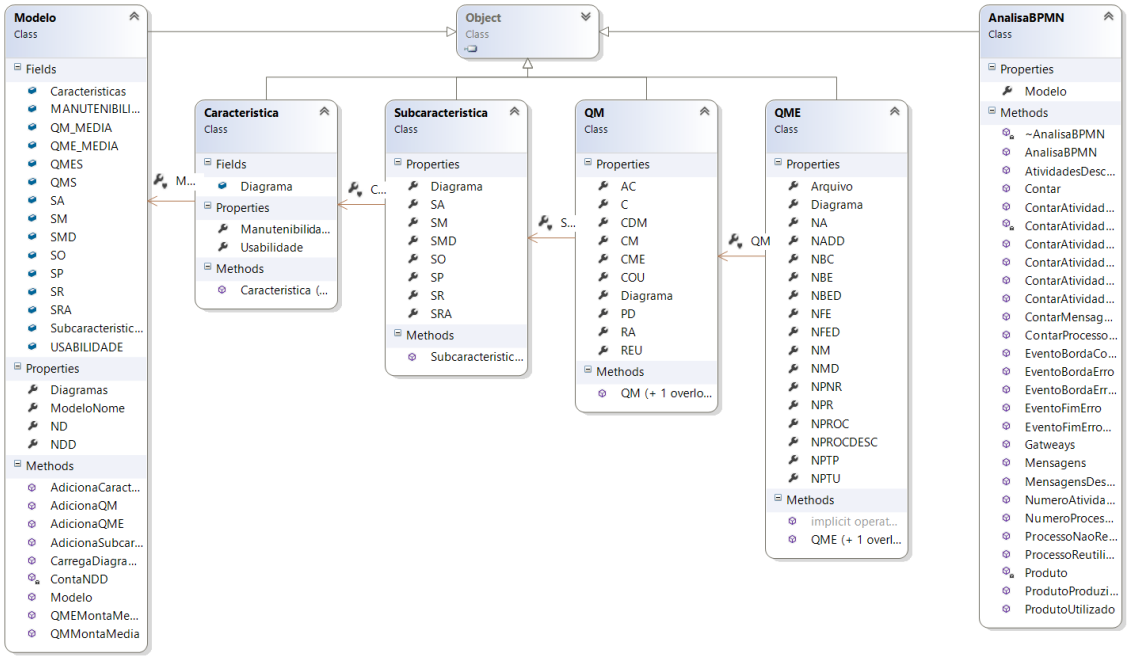



*Figura 23. Modelo de classes simplificado do SDPQualityForms.*

Há ainda as classes **Modelo** e **AnalisaBPMN**. A classe **AnalisaBPMN** é uma classe estática, responsável por realizar os acessos de baixo nível aos diagramas, fazendo as operações de leitura dos arquivos, coleta e contagem dos elementos para medição. A classe **Modelo** é responsável por organizar todas as informações das demais classes, compondo uma estrutura que representa as medições realizadas em todos os diagramas de um modelo.

A **SDPQualityForms** processa um modelo de processo de software a partir de uma seleção de pasta. Após a seleção da pasta onde se encontram os modelos BPMN de um mesmo modelo, a análise é realizada, primeiro pela coleta automática das QMEs, depois pelo cálculo das QMs a partir das QMEs coletadas, em seguida pelo cálculo das subcaracterísticas e características.

## 7.3 Primeira avaliação – MDS-Fiocruz

### 7.3.1 Execução do subprocesso “Definir do escopo da avaliação”

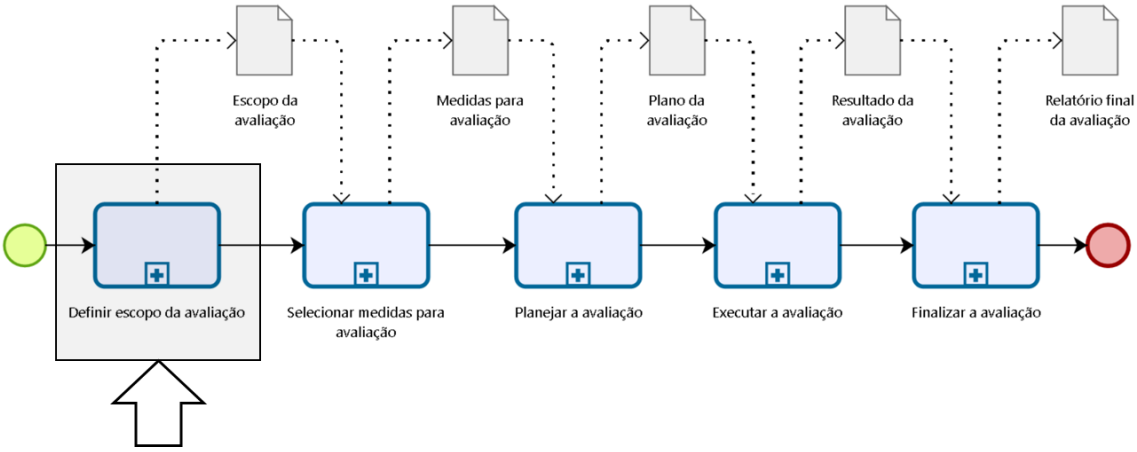



*Figura 24. Execução do subprocesso "Definir escopo".*

O primeiro subprocesso do processo de avaliação de modelos de processo de software é o de definição de escopo, conforme Figura 24. Neste subprocesso foram executadas as atividades “Estabelecer o propósito”, “Selecionar características de qualidade”, “Selecionar diagramas a avaliar” e “Definir forma de coleta”.

No que se refere à atividade “Estabelecer o propósito” foram verificados os objetivos da Fiocruz para sua metodologia de desenvolvimento de sistemas, a MDS-Fiocruz. Considerando que esta MDS havia sido utilizada, até então, apenas em sua área principal de TI, e que esta MDS deveria ser adotada pelas demais Unidades da instituição, a organização desejava avaliar sua MDS com o objetivo de identificar oportunidades de melhoria nas características de Usabilidade, que contribuiria para a adoção por outras

Unidades, e de Manutenibilidade, que contribuiria para a manutenção da MDS mesmo quando utilizada por várias Unidades.

A atividade seguinte, “Selecionar características de qualidade”, estabeleceu as características e subcaracterísticas de qualidade definidas no modelo de medição, conforme capítulo 5, como as características de qualidade para esta avaliação. A execução da atividade “Selecionar diagramas a avaliar” resultou na seleção de todos os 11 diagramas que compunham a MDS-Fiocruz como objeto da avaliação. A execução do subprocesso “Definir escopo da avaliação” produziu as informações do documento “Escopo da avaliação”, conforme Tabela 46.

*Tabela 46. Escopo da avaliação.*

| Campo | Objetivo |
|---|---|
| Propósito da avaliação | Identificar oportunidades de melhoria em termos de Usabilidade e Manutenibilidade da MDS, com vistas à sua adoção por outras Unidades da instituição. |
| Características de qualidade | Características de Usabilidade e de Manutenibilidade. |
| Diagramas a avaliar | • Atualizar Modelo de Processo de Negócio.bpmn<br>• Desenvolver Software.bpmn<br>• Especificar Requisitos.bpmn<br>• Gerenciar Mudanças.bpmn<br>• Gerenciar Projeto.bpmn<br>• Inspecionar Artefato.bpmn<br>• Mapear Processos de Negócio.bpmn<br>• MDS Fiocruz.bpmn<br>• Planejar Testes.bpmn<br>• Projetar Sistema.bpmn<br>• Testar Sistema.bpmn |
| Forma de coleta | A coleta das medidas será realizada de forma automática, utilizando-se a ferramenta de software **SDPQualitForms**. |

### 7.3.2 Execução do subprocesso “Selecionar medidas para avaliação”

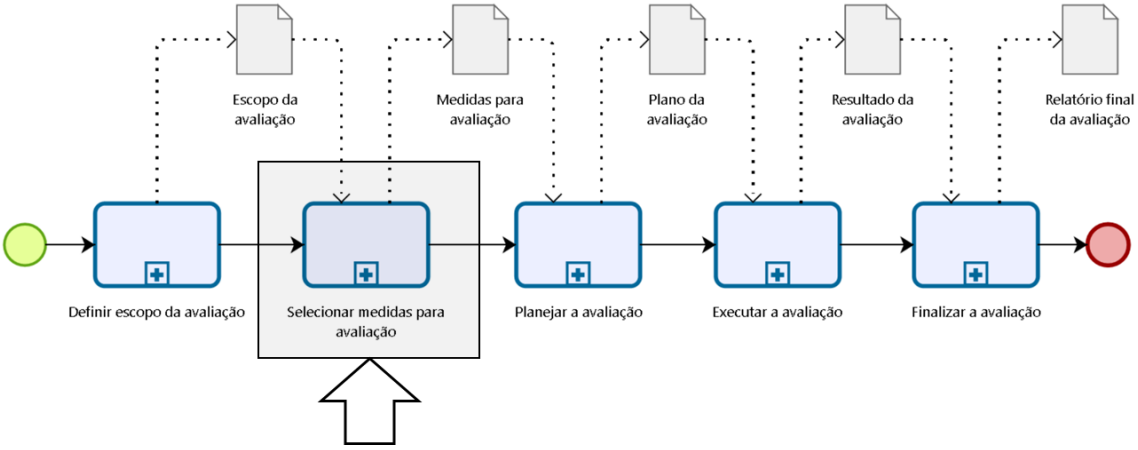



*Figura 25. Execução do subprocesso "Selecionar medidas para avaliação".*

O segundo subprocesso do processo de avaliação da qualidade de modelos de processo de software é o “Selecionar medidas para avaliação”, conforme Figura 25. Neste subprocesso foram executadas as atividades “Selecionar medidas de qualidade”, “Definir plano de ação”, e “Definir critérios de agregação”.

A execução da primeira atividade “Selecionar medidas de qualidade”, considerou o modelo de medição produzido no capítulo 5, que definiu medidas para as características “Usabilidade” e “Manutenibilidade” e suas subcaracterísticas. A atividade “Definir plano de ação” estabeleceu um conjunto de ações para as QMs que indicam possíveis ajustes a partir dos resultados da medição. Por fim, a atividade “Definir critérios de agregação” estabeleceu os critérios a serem considerados para a compreensão das características e subcaracterísticas de qualidade a partir da agregação dos resultados. A execução deste subprocesso produziu as informações para o documento “Medidas para avaliação”, conforme apresentadas na Tabela 47.

*Tabela 47. Medidas para avaliação.*

| Campo | Objetivo |
|---|---|
| Medidas selecionadas | Medidas a serem coletadas diretamente dos diagramas do modelo de processo de software:<br>• NPTP - Número de produtos de trabalho produzidos por atividades (dados ou documentos)<br>• NPTU - Número de produtos de trabalho utilizados por atividades (dados ou documentos)<br>• NA - Número de atividades<br>• NADD - Número de atividades que possuem descrição detalhada<br>• ND - Número de diagramas |

| | |
|---|---|
| | • NDD - Número de diagramas com descrição<br>• NBE - Número de eventos de borda tipo “erro”<br>• NBED - Número de eventos de borda tipo “erro” com descrição<br>• NFE - Número de eventos de fim tipo “erro”<br>• NFED - Número de eventos de fim tipo “erro” com descrição<br>• NPNR - Número de processos não reutilizáveis<br>• NPR - Número de processos reutilizáveis<br>• NBC - Número de eventos de borda tipo “compensação”<br>• NMD - Número de mensagens que possuem descrição<br>• NM - Número de mensagens (mensagens trocadas entre o processo e agentes externos)<br><br>QMs calculadas a partir das QMs:<br><br>• CDM - Completude da descrição do modelo<br>• COU - Completude da orientação ao usuário<br>• CME - Compreensibilidade das mensagens de erro<br>• CM - Clareza de mensagem<br>• PD - Possibilidade de desfazer (*undo*)<br>• REU - Recuperação de erros do usuário<br>• AC - Acoplamento dos componentes<br>• RA - Reusabilidade dos ativos<br>• C – Complexidade |
| Plano de ação | • CDM – Descrever os diagramas com índice menor do que 1.<br>Cada um dos diagramas do modelo deve ser descrito de modo a orientar o usuário e complementar sua compreensão do modelo.<br><br>• COU – Descrever as atividades e subprocessos dos diagramas que estiverem com índice menor do que 1.<br>Cada um dos subprocessos não reutilizáveis, processos reutilizáveis e atividades devem ser descritos de modo a orientar o usuário e complementar sua compreensão desses elementos notacionais no diagrama.<br><br>• CME – Descrever os eventos de erro nos diagramas em que o índice estiver menor do que 1.<br>Todos os eventos de erro existentes no diagrama devem estar descritos de modo a complementar a compreensão do usuário em relação ao erro previsto no diagrama.<br><br>• CM – Descrever as mensagens dos diagramas em que o índice estiver menor do que 1.<br>Todas as mensagens existentes no diagrama devem estar descritas de modo a complementar a compreensão do usuário em relação a mensagem apresentada no diagrama.<br><br>• PD – Implementar opções de desfazer, quando aplicável, nos diagramas em que o índice estiver menor do que 1. |

| | |
|---|---|
| | Quando aplicável, devem ser implementadas opções de desfazer atividades e subprocessos, indicando exatamente que atividades ou subprocessos serão desfeitos e em que circunstância isso deverá ocorrer.<br><br>• REU – Implementar opções de tratamento de erro nos diagramas em que o índice estiver menor do que 1, quando aplicável.<br>Quando aplicável, devem ser modeladas as ações de exceção pela modelagem de atividades ou subprocessos a serem executados nos casos de ocorrência de erro.<br><br>• AC – Aumentar o número de processos reutilizáveis, quando aplicável, onde os diagramas em que o índice estiver menor do que 1.<br>Quando aplicável, agrupar atividades e subprocessos em subprocessos reutilizáveis, aumentando a coesão e reduzindo o acoplamento do modelo.<br><br>• RA – Aumentar o número de processos reutilizáveis, quando aplicável, onde os diagramas em que o índice estiver menor do que 1.<br>Quando aplicável, transformar grupos de atividades e subprocessos que se repetem em diferentes partes do modelo em um único processo reutilizável que receba um ou mais parâmetros de entrada, por mensagens e objetos de dados, e retorno de uma ou mais informações por meio de de mensagens e objetos de dados.<br><br>• C – Reduzir a complexidade dos diagramas em que o índice estiver menor do que 1.<br>Reduzir o número de produtos de trabalho produzidos e/ou utilizados em relação ao número de atividades existentes no diagrama. |
| Agregação do resultado | As características de Usabilidade e Manutenibilidade deverão ser compreendidas a partir dos resultados de suas subcaracterísticas. Quanto às subcaracterísticas, estas deverão ser compreendidas a partir dos valores de suas QMs associadas. |

### 7.3.3 Executar o subprocesso “Planejar a avaliação”

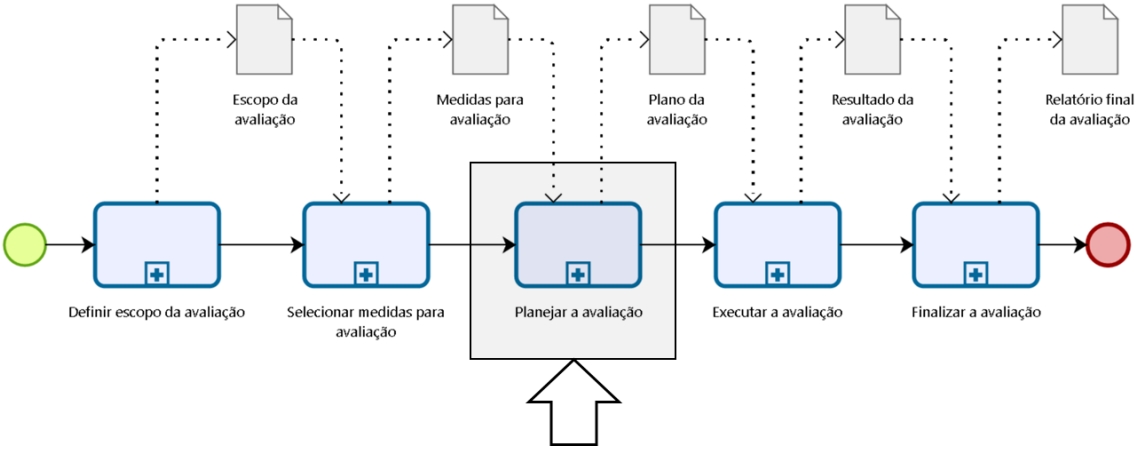



*Figura 26. Execução do subprocesso "Planejar avaliação".*

O terceiro subprocesso do processo de avaliação da qualidade de modelos de processo de software é o “Planejar a avaliação”, conforme Figura 26. Neste subprocesso são executadas as atividades “Explicitar partes envolvidas”, “Definir entregas”, “Elaborar cronograma” e “Planejar custos”. Ao executarmos este subprocesso foram produzidas as informações para o documento “Plano da avaliação”, conforme apresentado na Tabela 48.

*Tabela 48. Plano da avaliação.*

| Campo | Objetivo |
|---|---|
| Partes envolvidas | • Fundação Oswaldo Cruz (Fiocruz), responsável por fornecer o modelo vigente de sua MDS.<br>• Avaliador, responsável por avaliar a MDS através do processo de avaliação definido neste trabalho e, a partir dele, propor ajustes de melhoria. |
| Entregas | • Resultado da avaliação.<br>• Relatório final da avaliação. |
| Estimativa de tempo | 1. Recebimento da MDS Vigente da organização. Tempo estimado: 1 dia.<br>2. Aplicação do processo de avaliação. Tempo estimado: 1 dia.<br>3. Análise dos resultados. Tempo estimado 2 dias.<br>4. Definição dos ajustes necessários. Tempo estimado: 1 dia.<br>5. Elaboração e entrega do relatório final da avaliação: Tempo estimado: 1 dia. |
| Custo estimado | Não existem custos associados a esta avaliação. |

### 7.3.4 Executar o subprocesso “Executar a avaliação”

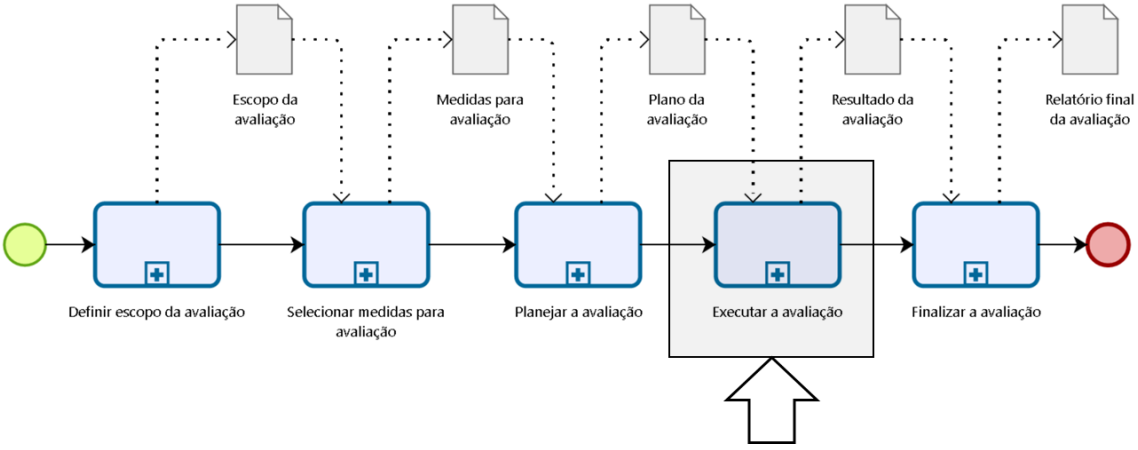



*Figura 27. Execução do subprocesso “Executar avaliação".*

O quarto subprocesso do processo de avaliação da qualidade de modelos de processo de software é o “Executar a avaliação”, conforme Figura 27. Neste subprocesso foram realizadas as atividades “Realizar medições”, “Aplicar plano de ação” e “Elaborar recomendações”.

A execução da atividade “Realizar medições” implicou na coleta das QMEs a partir dos diagramas do modelo selecionado no plano de avaliação. Para isso, foi utilizada a ferramenta **SDPQualityForms**, que retornou o conjunto de resultados apresentado na Tabela 49.

*Tabela 49. Resultados para QMEs da MDS-Fiocruz*

| ND | NDD | Diagrama | NPTP | NPTU | NA | NADD | NBE | NBED | NFE | NFED | NPNR | NPR | NBC | NMD | NM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 11 | 0 | Atualizar Modelo de Processo de Negócio.bpmn | 1 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Desenvolver Software.bpmn | 8 | 10 | 5 | 5 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 |
| | | Especificar Requisitos.bpmn | 7 | 12 | 5 | 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | | Gerenciar Mudanças.bpmn | 0 | 0 | 21 | 21 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 |
| | | Gerenciar Projeto.bpmn | 3 | 5 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Inspecionar Artefato.bpmn | 6 | 12 | 6 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Mapear Processos de Negócio.bpmn | 9 | 8 | 7 | 7 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MDS Fiocruz.bpmn | 7 | 3 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| | | Planejar Testes.bpmn | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Projetar Sistema.bpmn | 3 | 7 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Testar Sistema.bpmn | 5 | 7 | 9 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| - | | Min | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Max | 9 | 12 | 21 | 21 | 0 | 0 | 0 | 0 | 0 | 5 | 4 | 0 | 0 |
| | | **Média** | **4,545** | **6,000** | **6,000** | **5,727** | **0,000** | **0,000** | **0,000** | **0,000** | **0,000** | **0,909** | **0,364** | **0,000** | **0,000** |
| | | Desv. Padrão | 2,965 | 4,178 | 5,257 | 5,361 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 1,564 | 1,150 | 0,000 | 0,000 |
| | | Variância | 8,793 | 17,455 | 27,636 | 28,744 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 2,446 | 1,322 | 0,000 | 0,000 |

A ferramenta **SDPQualityForms** foi utilizada ainda para a realização dos cálculos das QMs a partir das QMEs coletadas. O resultado é apresentado na Tabela 50.

*Tabela 50. Resultado da medição para a MDS-Fiocruz.*

| Diagrama | CDM | COU | CME | CM | PD | REU | AC | RA | C |
|---|---|---|---|---|---|---|---|---|---|
| Atualizar Modelo de Processo de Negócio.bpmn | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,250 |
| Desenvolver Software.bpmn | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 1,000 | 0,500 | 0,357 |
| Especificar Requisitos.bpmn | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 1,000 | 0,167 | 0,240 |
| Gerenciar Mudanças.bpmn | 0,000 | 1,000 | 0,000 | 0,000 | 0,190 | 0,190 | 0,000 | 0,000 | 1,000 |
| Gerenciar Projeto.bpmn | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,273 |
| Inspecionar Artefato.bpmn | 0,000 | 0,833 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,250 |
| Mapear Processos de Negócio.bpmn | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 1,000 | 0,125 | 0,320 |
| MDS Fiocruz.bpmn | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 1,000 | 0,429 | 0,412 |
| Planejar Testes.bpmn | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,500 |
| Projetar Sistema.bpmn | 0,000 | 0,500 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,286 |
| Testar Sistema.bpmn | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,429 |
| Min | 0,000 | 0,500 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,240 |
| Max | 0,000 | 1,000 | 0,000 | 0,000 | 0,190 | 0,190 | 1,000 | 0,500 | 1,000 |
| **Média** | **0,000** | **0,939** | **0,000** | **0,000** | **0,017** | **0,017** | **0,364** | **0,111** | **0,392** |
| Desv. Padrão | 0,000 | 0,147 | 0,000 | 0,000 | 0,055 | 0,055 | 0,481 | 0,176 | 0,209 |
| Variância | 0,000 | 0,024 | 0,000 | 0,000 | 0,003 | 0,003 | 0,255 | 0,034 | 0,048 |

A execução da atividade “Aplicar plano de ação” considerou os critérios propostos no subprocesso de planejamento, e a partir da Tabela 49 e da Tabela 50 propôs um conjunto de ações possíveis. Estas ações estão apresentadas na Tabela 51.

*Tabela 51. Ações possíveis a partir dos critérios de decisão para medições.*

| Critério de decisão | Ação possível |
|---|---|
| CDM – Descrever as atividades e processos para os diagramas com índice menor do que 1. | Uma vez que a QM CDM é composta das QMEs ND e NDD, e que NDD é igual a zero, esta QME deve ser elevada para o equivalente ao número de diagramas do modelo, ou seja, todos os diagramas do modelo devem ter uma descrição. |
| COU – Descrever os diagramas que estiverem com índice menor do que 1. | A QM COU é composta das QMEs NA e NADD. No entanto, a maior parte das atividades possui descrições detalhadas, elevando o valor de COU para próximo de 1. Por isso, não foram consideradas necessárias ações de ajuste para esta QM. |
| CME – Descrever os eventos de erro nos diagramas em que o índice estiver menor do que 1. | A QM CME é composta das QMEs NBE, NFE, NBED e NFED. Considerando o baixo valor de CME, estas QMEs devem ser elevadas por se prever caminhos de exceção para eventuais erros que ocorram durante o processo. |
| CM – Descrever as mensagens dos diagramas em que o índice estiver menor do que 1. | A QM CM é composta das QMEs NM e NMD. Considerando o baixo valor de CM, estas QMES devem ser elevadas pela implementação de agentes externos aos processos, e pela explicitação da troca de mensagens com descrições específicas, aumentando a clareza do processo. |
| PD – Implementar opções de desfazer, quando aplicável, nos diagramas em que o índice estiver menor do que 1. | A QM PD é composta principalmente das QMEs NBC e NA. Considerando o baixo valor de PD, a QME NBC deve ser aumentada pela implementação de mecanismos de *Compensation* para operações que podem ser interrompidas, caso exista esta possibilidade no modelo de processo em questão. |
| REU – Implementar opções de tratamento de erro nos diagramas em que o índice estiver menor do que 1. | A QM REU é composta das QMEs NFE, NBC e NA. Considerando o baixo valor de REU, as QMEs NFE e NBC devem ser elevadas pela inclusão de atividades para prevenção e recuperação de erros ao longo do processo, caso isto seja aplicável ao modelo de processo em questão. |
| AC – Aumentar o número de processos reutilizáveis, quando aplicável, onde os diagramas em que o índice estiver menor do que 1. | A QM AC é composta pelas QMEs NPR e NPNR. Na medida do aplicável, a QME NPR deve ser elevada pela implementação de processos reutilizáveis, proporcionando assim a introdução de componentes independentes no modelo, que sofram pouco ou nenhum impacto em função da eventual mudança de outros componentes. |
| RA – Aumentar o número de processos reutilizáveis, quando aplicável, onde os diagramas em que o índice estiver menor do que 1. | A QM RA é composta das QMEs NPR, NPNR e NA. Considerando o baixo valor da QM RA, a QME NPR deve ser elevada pela implementação de processos reutilizáveis, proporcionando assim a introdução de componentes independentes no modelo. Deve-se notar que esta ação proposta contribui para a elevação tanto da QM AC quanto da QM RA. |
| C – Reduzir a complexidade dos diagramas em que o índice estiver menor do que 1. | A QM C é composta das QMEs NPTU, NPTP e NA. Considerando o baixo valor da QM C, as QMEs NPTU e NPTP devem ser reduzidas em relação à QM NA, pela otimização dos produtos de trabalho utilizados no modelo. |

A execução da atividade “Elaborar recomendações” considerou os resultados agregados para as subcaracterísticas da qualidade, produzindo desta maneira as informações para o documento “Resultado da avaliação”, conforme apresentado na Tabela 52.

*Tabela 52. Resultado da avaliação.*

| **Característica** | **Subcaracterística** | **Valor** | **Análise** | **Recomendação** |
|---|---|---|---|---|
| Usabilidade | SRA - Reconhecimento apropriado | 0,000 | A QM CDM está zerada para todos os diagramas. | Uma vez que a QM CDM é composta de NDD, e NDD é igual a zero, esta QME deve ser elevada para o equivalente ao número de diagramas do modelo, ou seja, todos os diagramas do modelo devem ter uma descrição de apoio ao usuário. |
| | SA - Apreensibilidade | 0,470 | A QM CME está zerada para todos os diagramas | Uma vez que a QM CME é composta de NBE, NFE, NBED e NFED, estas QMEs devem ser elevadas por se prever caminhos de exceção para eventuais erros que acontecerem durante o processo. |
| | SO - Operabilidade | 0,009 | As QMs CM e PD estão zeradas para todos os diagramas, menos PD no diagrama “Gerenciar mudanças”, no qual tem o índice 0,190. | Uma vez que a QM CM é composta de NM e NMD, estas QMEs devem ser elevadas pela implementação de agentes externos aos processos, e trocas de mensagens com descrições específicas, aumentando a clareza do processo.<br>Uma vez que a QM PD é composta principalmente de NBC, esta QME deve ser aumentada por se implementar mecanismos de *Compensation* (desfazer/*undo*) para operações que podem ser interrompidas, caso exista esta possibilidade. |
| | SP - Proteção de erros do usuário | 0,017 | A QM REU está zerada para todos os diagramas, menos no diagrama “Gerenciar mudanças”, no qual tem o índice 0,190. | Uma vez que REU é composta principalmente de NFE e NBC, estas QMEs devem ser elevadas pela inclusão de atividades para prevenção e recuperação de erros ao longo do processo. |
| Manutenibilidade | SM - Modularidade | 1,000 | SM utiliza a QM AC, que possui 4 diagramas com índice 1 e 7 diagramas com índice zero. | Considerando a existência de 7 diagramas sem subprocessos, é possível haver possibilidade de aumento de subprocessos deste tipo. |

| | | | | |
|---|---|---|---|---|
| | | | A média de AC considera apenas os diagramas que possuem subprocessos. O resultado é 1 porque nos 4 diagramas que possuem subprocessos, todos são reutilizáveis. | Uma vez que AC é composta principalmente de NPR, esta QME deve ser elevada pela implementação de processos reutilizáveis, se aplicável, proporcionando assim a introdução de componentes independentes no modelo. |
| | SR - Reusabilidade | 0,305 | A QM RA se apresenta com índice zero em 7 diagramas, e com índices variando entre 0,125 e 0,500 nos outros 4 diagramas. | Uma vez que RA é composta principalmente de NPR, esta QME deve ser elevada pela implementação de processos reutilizáveis, proporcionando assim a introdução de componentes independentes no modelo. |
| | SMD - Modificabilidade | 0,696 | A QM C apresenta índices entre 0.240 e 1, e AC que possui 4 diagramas com índice 1 e 7 diagramas com índice zero. | Uma vez que AC é composta principalmente de NPR, esta QME deve ser elevada pela implementação de processos reutilizáveis, proporcionando assim a introdução de componentes independentes no modelo. |

### 7.3.5 Execução do subprocesso “Finalizar a avaliação’

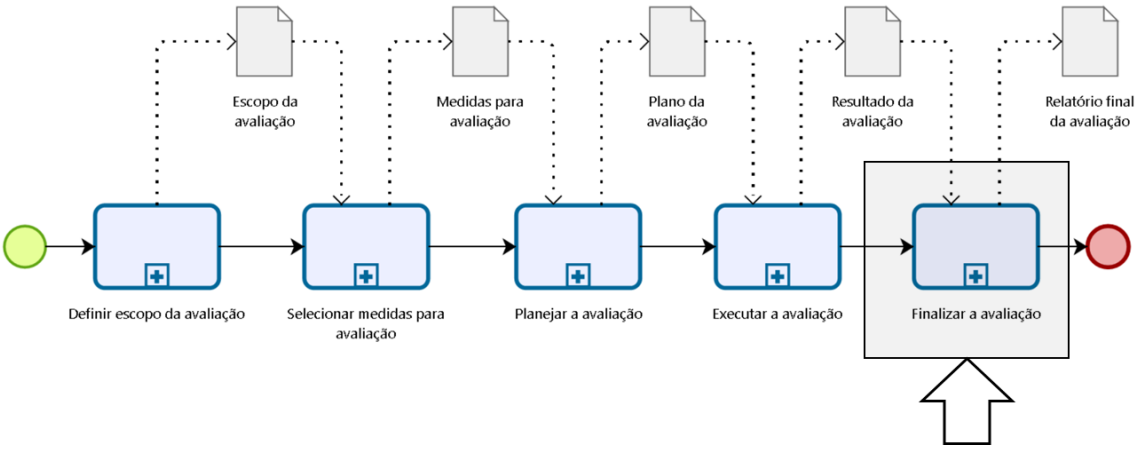



*Figura 28. Execução do subprocesso "Finalizar a avaliação" do processo de avaliação.*

O quinto e último subprocesso do processo de avaliação da qualidade de modelos de processo de software é o “Finalizar a avaliação”, conforme Figura 28. Neste subprocesso são executadas as atividades “Elaborar o relatório da avaliação”, “Identificar lições aprendidas” e “Entregar documentação”. A execução deste subprocesso produziu as informações para o documento “Relatório final da avaliação”, como pode ser observado na Tabela 53.

*Tabela 53. Relatório final da avaliação.*

| Campo | Objetivo |
|---|---|
| Organização onde foi realizada a avaliação | Fundação Oswaldo Cruz |
| Modelo avaliado | MDS-Fiocruz (Metodologia de Desenvolvimento de Sistemas) |
| Versão | 4.0 |
| Período da avaliação | De 7 a 18 de novembro de 2016. A avaliação foi realizada em dias intercalados dentro deste período. |
| Diagramas avaliados | • Atualizar Modelo de Processo de Negócio.bpmn<br>• Desenvolver Software.bpmn<br>• Especificar Requisitos.bpmn<br>• Gerenciar Mudanças.bpmn<br>• Gerenciar Projeto.bpmn<br>• Inspecionar Artefato.bpmn<br>• Mapear Processos de Negócio.bpmn<br>• MDS Fiocruz.bpmn<br>• Planejar Testes.bpmn<br>• Projetar Sistema.bpmn<br>• Testar Sistema.bpmn |
| Considerações sobre o resultado | O modelo de processo de software corrente foi considerado inadequado para cumprir sua nova função institucional, ou seja, se tornar o modelo padrão de desenvolvimento para todas as Unidades da Fiocruz. |
| Curso de ação decidido | O modelo de processo de software avaliado foi considerado insatisfatório para os objetivos originais da organização, de proporcionar níveis adequados nas características de Usabilidade e Manutenibilidade para sua ampla adoção em todas as Unidades da instituição. Deverão ser realizados os ajustes recomendados pela avaliação com o objetivo de melhorar os índices das QMEs e QMs associadas a estas características de qualidade. A partir destes ajustes, uma nova versão do modelo deve ser produzida e avaliada quanto a Usabilidade e Manutenibilidade. Para este fim, cada um dos diagramas do modelo será avaliado em relação aos ajustes recomendados pela avaliação. |
| Ajustes recomendados pela avaliação | 1. Elevar o valor da QME NDD para o equivalente ao número de diagramas do modelo, ou seja, todos os diagramas do modelo devem ter uma descrição de apoio ao usuário. Impacto na QM CDM. Todos os diagramas do modelo deverão ser analisados para este fim.<br>2. Elevar os valores das QMEs NBE, NFE, NBED e NFED por se prever caminhos de exceção para eventuais erros na execução do processo. Impacto na QM CME. Todos os diagramas do modelo deverão ser analisados para este fim.<br>3. Elevar os valores das QMEs NM e NMD pela implementação de agentes externos aos processos e pela implementação de trocas de mensagens com descrições específicas, quando aplicável. Impacto na QM CM. Todos os diagramas do modelo deverão ser analisados para este fim.<br>4. Elevar o valor da QME NBC por se implementar mecanismos de *Compensation* (desfazer/*undo*) para operações que podem |

| | |
|---|---|
| | ser interrompidas, quando aplicável. Impacto na QM PD. Todos os diagramas do modelo deverão ser analisados para este fim.<br>5. Elevar o valor das QMEs NFE e NBC pela inclusão de atividades para prevenção e recuperação de erros ao longo do processo. Impacto na QM REU. Todos os diagramas do modelo deverão ser analisados para este fim.<br>6. Elevar o valor da QME NPR pela implementação de processos reutilizáveis, se aplicável, proporcionando assim a introdução de componentes independentes no modelo. Impacto na QM AC e na QM RA. Todos os diagramas do modelo deverão ser analisados para este fim. |
| Limitações/Restrições | Não houve limitações/restrições impostas a este projeto de avaliação. |

## 7.4 Implementação dos ajustes

Após a realização da avaliação, os ajustes recomendados foram implementados na versão original do modelo de processo de software da Fiocruz (MDS-Fiocruz), produzindo como consequência uma nova versão desse modelo de processo, a MDS-Fiocruz ajustada. A seguir, são apresentados os ajustes recomendados pela avaliação e exemplos de sua implementação no modelo.

### 7.4.1 Elevar NDD para o equivalente ao número de diagramas do modelo

No modelo original nenhum dos diagramas continha uma descrição abrangente que permitisse ao usuário compreender o objetivo daquele diagrama. No modelo ajustado, todos os diagramas receberam uma descrição abrangente, porém adequada para o apoio à compreensão do usuário. Por exemplo, para o diagrama apresentado na Figura 29, foi adicionada a seguinte descrição:

> “O início do desenvolvimento do sistema integra atividades específicas de desenvolvimento e processos de gestão de projetos. O Termo de Abertura do Projeto (TAP) é o documento que dá uma ideia geral do projeto, inclusive em termos de custo e prazo, e este documento também autoriza o início e prosseguimento do projeto”.

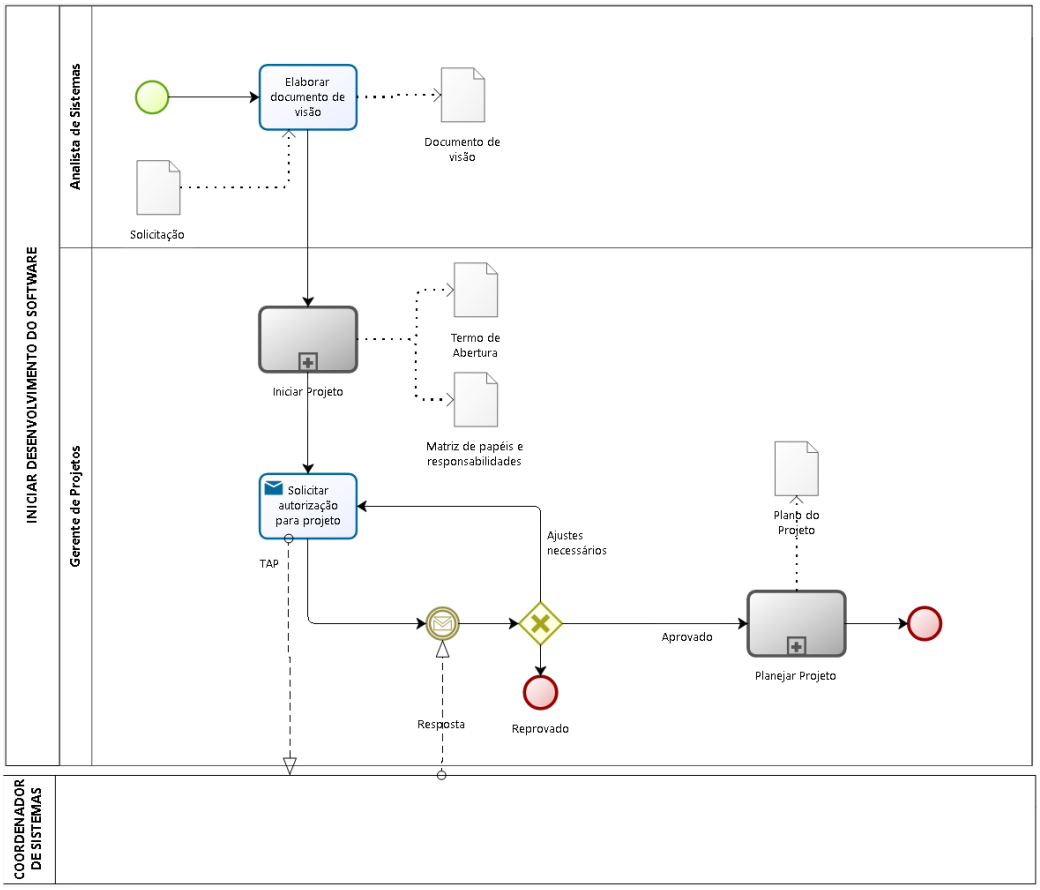



*Figura 29. Subprocesso "Iniciar desenvolvimento do software", do processo MDS-Fiocruz ajustado.*

### 7.4.2 Elevar NBE, NFE, NBED e NFED por se prever caminhos de exceção para eventuais erros

Os modelos devem apresentar, além dos caminhos normais, os caminhos de exceção. Caso contrário, quando estas exceções ocorrem os usuários não sabem o que fazer. Como resultado natural, cada usuário poderia tomar uma linha de ação diferente, eventualmente implicando em impactos negativos para todo o processo. O processo apresentado na Figura 30 trata da possibilidade de haver falhas no artefato. Neste caso, o desvio é devidamente registrado com um evento de fim descrito da seguinte maneira: “Artefato contém falhas”. Caso contrário, o evento de fim indica “Artefato Ok”. Com esta indicação, o processo chamador poderá tratar o erro apropriadamente.

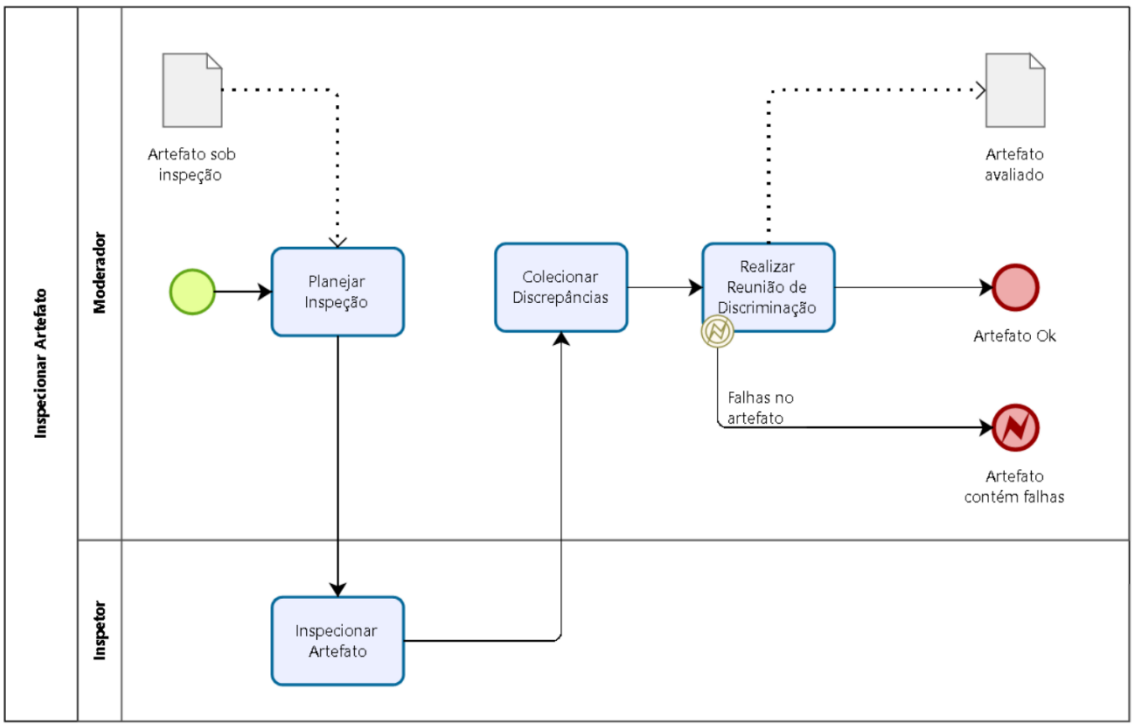



*Figura 30. Subprocesso "Encerrar projeto", do processo MDS-Fiocruz ajustado.*

### 7.4.3 Elevar NM e NMD pela implementação de agentes externos e trocas de mensagens com descrições

As comunicações entre o subprocesso e os agentes externos devem ser registradas, para a melhor compreensão do usuário. Os agentes externos podem ser atores (pessoas, departamentos, empresas) ou outros processos com o qual o processo atual troca informações. Estas comunicações foram registradas no modelo apresentado na Figura 31. É possível perceber que existem duas entradas neste processo, que são uma nova solicitação de serviço (também chamada de RSI pelos usuários da organização em questão), ou pode ser a resposta a uma pergunta enviada pelo próprio processo. A pergunta é enviada pela atividade “Solicitar informações adicionais”. Deste modo, os três pontos de comunicação entre o processo e o “Solicitante” ficam explicitados e documentados.

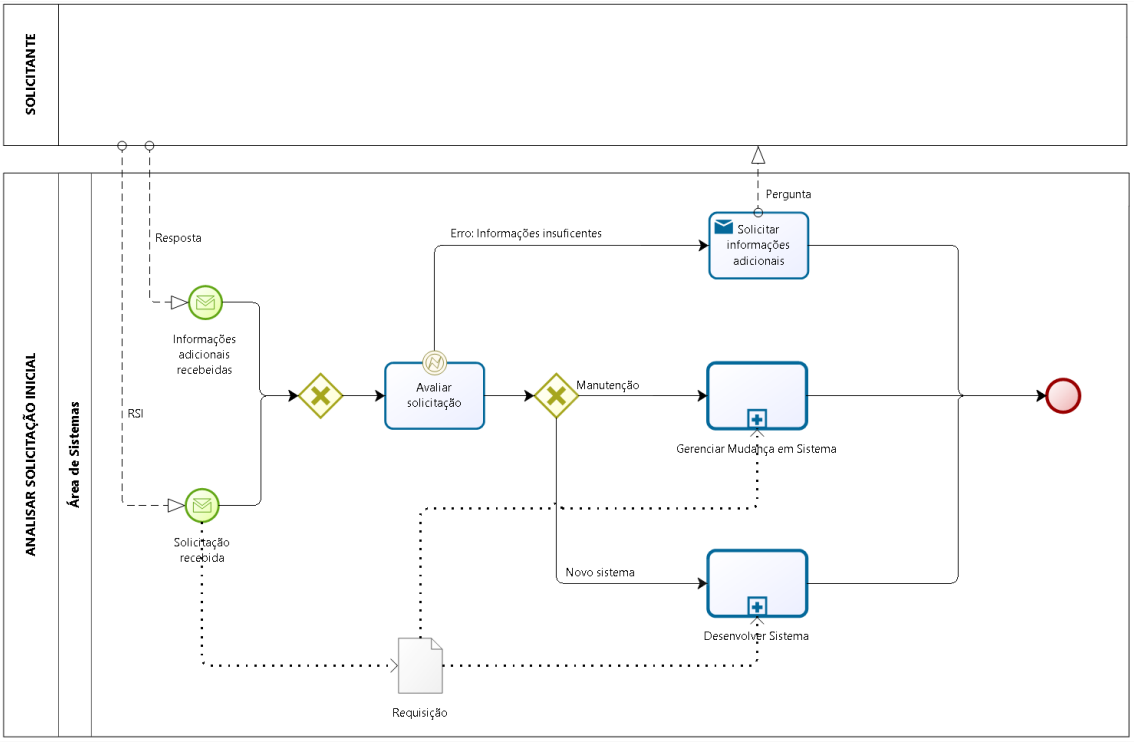



*Figura 31. Subprocesso "Analisar solicitação inicial", do processo MDS-Fiocruz ajustado.*

### 7.4.4 Elevar NBC por se implementar mecanismos de *Compensation* (desfazer/*undo*)

O subprocesso apresentado na Figura 32 trata da implementação de dois mecanismos de compensação (*undo*) no subprocesso “Encerrar projeto”. As atividades de compensação têm o objetivo de desfazer ações durante a ocorrência de erros, para evitar prejuízos ao processo. As ações de compensação são ativadas por condições previstas no evento de compensação. Para facilitar a compreensão das compensações implementadas no modelo, seguem suas respectivas descrições:

> **“Reativar projeto**. Caso sejam identificadas falhas na atividade de cancelar o projeto por parte dos interessados, ou se houver condições não atendidas para o encerramento do projeto, o cancelamento do projeto é revertido”.
>
> “**Manter contratos ativos**. Caso existam contratos que não possam ser cancelados no momento em função de impedimentos técnicos ou legais, os mesmos deixam de figurar como cancelados e se mantém ativos”.

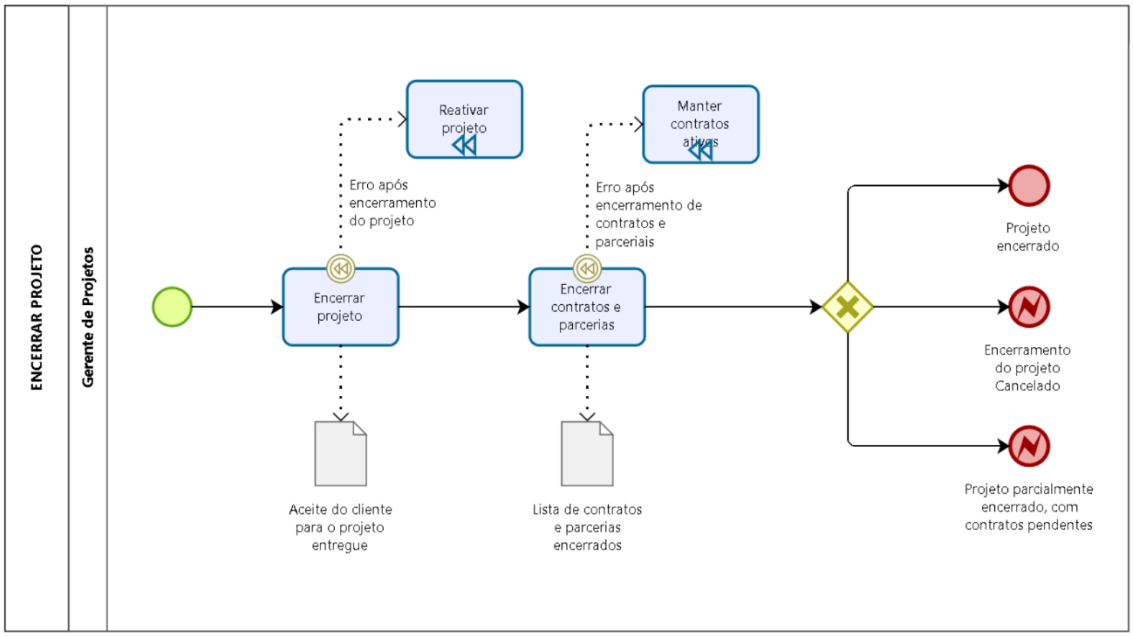



*Figura 32. Subprocesso "Encerrar projeto", do processo MDS-Fiocruz ajustado.*

### 7.4.5 Elevar NFE e NBC pela inclusão de atividades para prevenção e recuperação de erros ao longo do processo

Esta recomendação foi atendida e descrita nas seções 0 e 7.4.4 Ela também foi implementada em outros diagramas, como no “Desenvolver” e “Controlar projeto”. No entanto, é importante registrar que os erros e compensações devem ser implementados apenas nos diagramas que demandem por este tipo de previsão, e não simplesmente para aumentar o número de tratamento de erros e de compensação do modelo, o que pode eventualmente deixar o modelo confuso para os usuários.

### 7.4.6 Elevar NPR pela implementação de processos reutilizáveis

A utilização de processos reutilizáveis vai tornar o modelo mais organizado, mais simples de entender, e mais fácil de manter. Isso acontece porque os diagramas ficam menos acoplados, ou seja, mais independentes. Por conta disso, ajustes em um diagrama terão pouco ou nenhum impacto nos demais diagramas. Isto também evita retrabalho, uma vez que um mesmo diagrama pode ser reutilizado por diversos outros diagramas. Por exemplo, o diagrama apresentado na Figura 33 é utilizado por outros subprocessos. O artefato “Plano do projeto” funciona como parâmetro de entrada, e os eventos de fim padrão e “Projeto abortado” funcionam como retorno. Desse modo, o diagrama atua de maneira similar a uma função de sistema, ou a um método de classe, recebendo um

parâmetro, processando este parâmetro, e oferecendo um retorno para o diagrama que o chama.

É necessário destacar que a utilização de subprocessos reutilizáveis poderá implicar em um aumento do número de diagramas do modelo, uma vez que atividades serão destacadas e agrupadas em subprocessos independentes, representados em diagramas distintos.

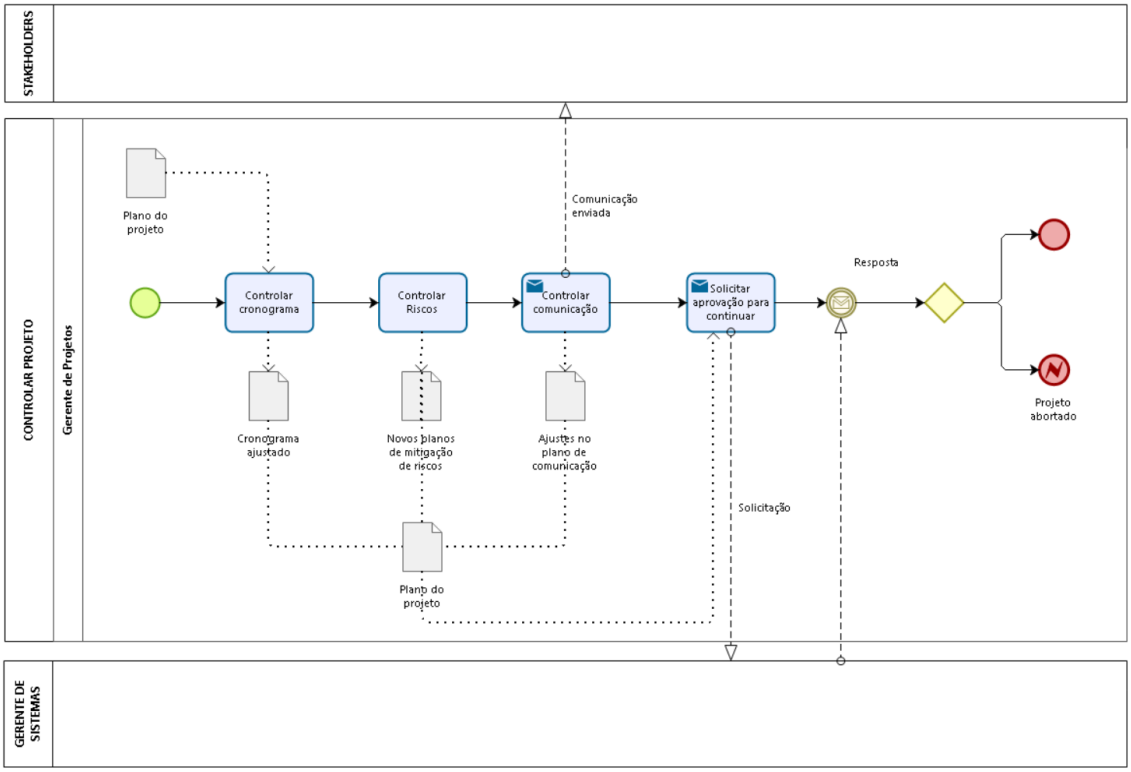



*Figura 33. Subprocesso "Controlar projeto", do processo MDS-Fiocruz ajustado.*

## 7.5 Segunda avaliação – MDS-Fiocruz ajustada

Após a implementação dos ajustes recomendados pela primeira avaliação a Fiocruz produziu uma nova versão de sua MDS, a MDS-Fiocruz ajustada. O passo seguinte foi avaliar esta nova versão para fins de comparação com a versão original.

### 7.5.1 Execução do subprocesso “Definir do escopo da avaliação”

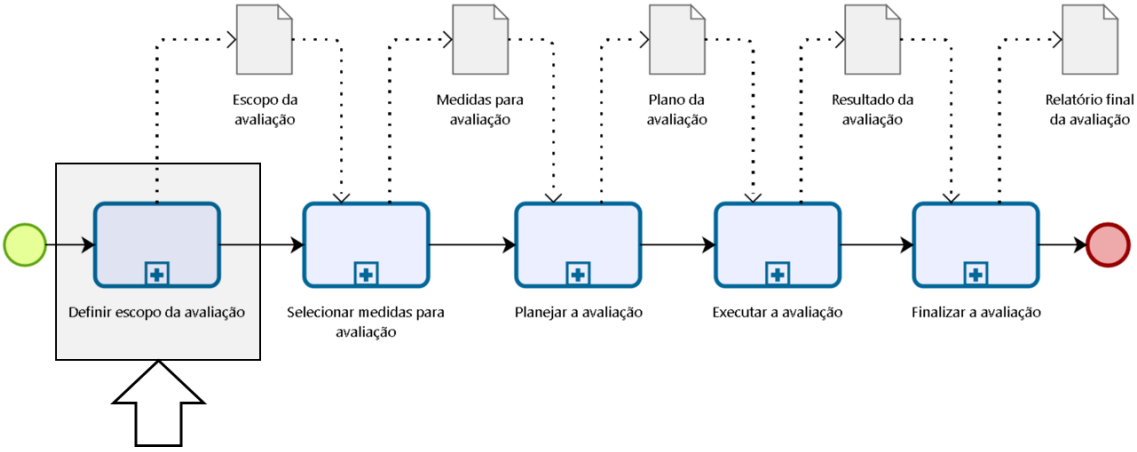



*Figura 34. Execução do subprocesso "Definir escopo da avaliação", segunda avaliação.*

O primeiro subprocesso do processo de avaliação de modelos de processo de software é o de definição de escopo, conforme Figura 32. Neste subprocesso foram executadas as atividades “Estabelecer o propósito”, “Selecionar características de qualidade”, “Selecionar diagramas a avaliar” e “Definir forma de coleta”.

No que se refere à atividade “Estabelecer o propósito”, considerando que a MDS-Fiocruz ajustada foi o resultado da implementação de ajustes propostos por uma avaliação anterior, o propósito da avaliação corrente foi o de verificar se houve melhoria nas características de Usabilidade e Manutenibilidade em relação à versão original do modelo de processo de software.

A atividade seguinte, “Selecionar características de qualidade”, definiu as características de qualidade para essa avaliação. Naturalmente, em função do propósito desta avaliação as características de qualidade selecionados foram idênticas àquelas utilizadas na primeira avaliação.

A execução da atividade “Selecionar diagramas a avaliar” resultou na seleção de todos os 20 diagramas da MDS-Fiocruz ajustada. É importante ressaltar que o número de diagramas da versão ajustada foi maior do que o da versão original, pois as recomendações da primeira avaliação implicaram na reorganização das atividades e na criação de novos subprocessos reutilizáveis. A execução do subprocesso “Definir escopo da avaliação” produziu as informações do documento “Escopo da avaliação”, conforme Tabela 54.

*Tabela 54. Escopo da avaliação, segunda avaliação.*

| Campo | Objetivo |
|---|---|
| Propósito da avalição | Verificar se houve melhoria em termos de Usabilidade e Manutenibilidade da MDS após a implementação dos ajustes propostos pela primeira avaliação. |
| Características da qualidade | Características de Usabilidade e de Manutenibilidade. |
| Diagramas a avaliar | • 1-MDS.bpmn<br>• 2-Desenvolver.bpmn<br>• 2.1-Iniciar.bpmn<br>• 2.2-Elaborar.bpmn<br>• 2.2.1-Modelar processo.bpmn<br>• 2.2.2-Especificar Requisitos.bpmn<br>• 2.2.3-Projetar Sistema.bpmn<br>• 2.3-Construir.bpmn<br>• 2.3.1-Implementar.bpmn<br>• 2.3.2-Testar.bpmn<br>• 2.3.2.1-Implementar Testes.bpmn<br>• 2.3.2.2-Executar Testes.bpmn<br>• 2.3.2.3-Aceitação.bpmn<br>• 2.4-Implantar.bpmn<br>• 3-Gerenciar Mudança.bpmn<br>• Inspecionar Artefato.bpmn<br>• P-Controlar.bpmn<br>• P-Encerrar.bpmn<br>• P-Iniciar.bpmn<br>• P-Planejar.bpmn |
| Forma de coelta | A coleta será realizada de forma automática, utilizando-se a ferramenta de software **SDPQualitForms**. |

### 7.5.2 Execução do subprocesso “Selecionar medidas para avaliação”

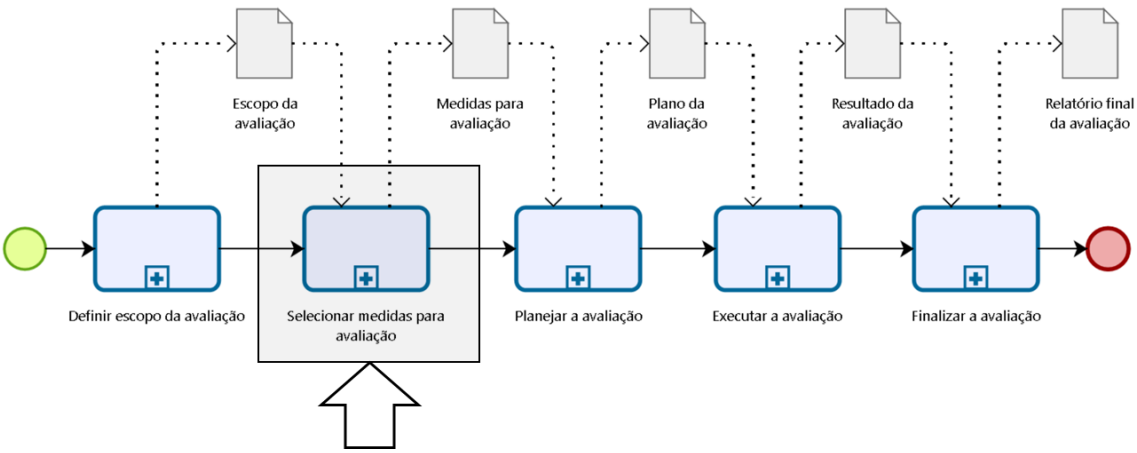



*Figura 35. Execução do subprocesso “Selecionar medidas para avaliação", segunda avaliação.*

O segundo subprocesso do processo de avaliação da qualidade de modelos de processo de software é o “Selecionar medições para avaliação”, conforme Figura 35. Neste subprocesso foram executadas as atividades “Selecionar medidas de qualidade”, “Definir plano de ação”, e “Definir critérios de agregação”.

A execução da primeira atividade “Selecionar medidas de qualidade” utilizou as mesmas medidas para as características Usabilidade e Manutenibilidade e suas subcaracterísticas que aquelas utilizadas na primeira avaliação. A atividade “Definir plano de ação” estabeleceu apenas o objetivo de comparar os resultados das QMs com os resultados da avaliação anterior, de acordo com o propósito da avaliação. Por fim, a atividade “Definir critérios de agregação” estabeleceu os critérios a serem considerados para a compreensão das características e subcaracterísticas de qualidade. Ao executarmos este subprocesso foram produzidas as informações do documento “Medidas para avaliação”, conforme apresentado na Tabela 55.

*Tabela 55. Medidas para avaliação, segunda avaliação.*

| Campo | Objetivo |
|---|---|
| Medidas selecionadas | Medidas a serem coletadas diretamente dos diagramas do modelo de processo de software:<br>• NPTP - Número de produtos de trabalho produzidos por atividades (dados ou documentos)<br>• NPTU - Número de produtos de trabalho utilizados por atividades (dados ou documentos)<br>• NA - Número de atividades<br>• NADD - Número de atividades que possuem descrição detalhada<br>• ND - Número de diagramas<br>• NDD - Número de diagramas com descrição<br>• NBE - Número de eventos de borda tipo “erro”<br>• NBED - Número de eventos de borda tipo “erro” com descrição<br>• NFE - Número de eventos de fim tipo “erro”<br>• NFED - Número de eventos de fim tipo “erro” com descrição<br>• NPNR - Número de processos não reutilizáveis<br>• NPR - Número de processos reutilizáveis<br>• NBC - Número de eventos de borda tipo “compensação”<br>• NMD - Número de mensagens que possuem descrição<br>• NM - Número de mensagens (mensagens trocadas entre o processo e agentes externos)<br>QMs calculadas a partir das QMs:<br>• CDM - Completude da descrição do modelo |

| | |
|---|---|
| | • COU - Completude da orientação ao usuário<br>• CME - Compreensibilidade das mensagens de erro<br>• CM - Clareza de mensagem<br>• PD - Possibilidade de desfazer<br>• REU - Recuperação de erros do usuário<br>• AC - Acoplamento dos componentes<br>• RA - Reusabilidade dos ativos<br>• C – Complexidade |
| Plano de ação | De acordo com o propósito da avaliação, a ações se resumem a comparar os resultados da avaliação corrente com a avaliação anterior:<br><br>CDM – Comparar com o resultado da primeira avaliação.<br>COU – Comparar com o resultado da primeira avaliação.<br>CME – Comparar com o resultado da primeira avaliação.<br>CM – Comparar com o resultado da primeira avaliação.<br>PD – Comparar com o resultado da primeira avaliação.<br>REU – Comparar com o resultado da primeira avaliação.<br>AC – Comparar com o resultado da primeira avaliação.<br>RA – Comparar com o resultado da primeira avaliação.<br>C – Comparar com o resultado da primeira avaliação. |
| Agregação do resultado | As características de Usabilidade e Manutenibilidade deverão ser compreendidas a partir dos resultados de suas subcaracterísticas. Por se tratarem das primeiras avaliações utilizando o processo de avaliação proposto, optou-se por não conformar as subcaracterísticas em fórmulas para compor as características, neste momento. Quanto às subcaracterísticas, estas serão representadas pelo valor da QM associada. Quando houver mais de uma QM associada, a subcaracterística será representada pela média de suas QMs. |

### 7.5.3 Execução do subprocesso “Planejar a avaliação”

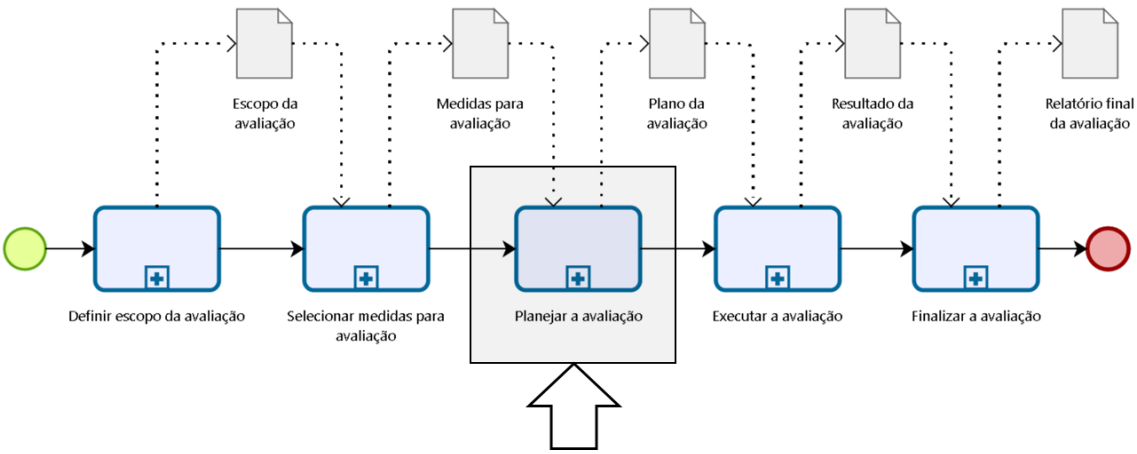



*Figura 36. Execução do subprocesso "Planejar a avaliação, segunda avaliação.*

O terceiro subprocesso do processo de avaliação da qualidade de modelos de processo de software é o “Planejar a avaliação”, conforme Figura 36. Neste subprocesso são

executadas as atividades “Explicitar partes envolvidas”, “Definir entregas”, “Elaborar cronograma” e “Planejar custos”. Ao executarmos este subprocesso foram produzidas as informações para o documento “Plano da avaliação”, conforme apresentado na Tabela 56.

*Tabela 56. Plano da avaliação, segunda avaliação.*

| Campo | Objetivo |
|---|---|
| Partes envolvidas | • Fundação Oswaldo Cruz (Fiocruz), responsável por fornecer o modelo ajustado de sua MDS.<br>• Avaliador, responsável por avaliar a MDS ajustada através do processo de avaliação definido neste trabalho e, a partir dele, comparar com os resultados da avaliação anterior. |
| Entregas | • Relatório final da avaliação. |
| Estimativa de tempo | • Recebimento da MDS ajustada da organização. Tempo estimado: 1 dia.<br>• Aplicação do processo de avaliação. Tempo estimado: 1 dia.<br>• Análise dos resultados. Tempo estimado 1 dias.<br>• Elaboração e entrega do relatório final da avaliação. Tempo estimado: 1 dia. |
| Custo estimado | Não houve custos associados a esta avaliação. |

### 7.5.4 Execução do subprocesso “Executar a avaliação’

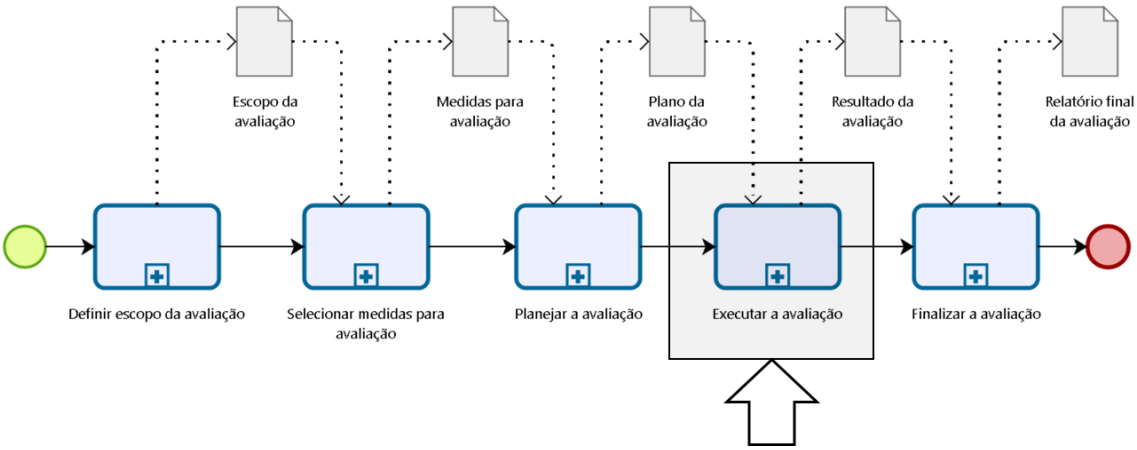



*Figura 37. Execução do subprocesso "Executar avaliação", segunda avaliação.*

O quarto subprocesso do processo de avaliação da qualidade de modelos de processo de software é o “Executar avaliação”, conforme Figura 37. Neste subprocesso foram realizadas as atividades “Realizar medições”, “Aplicar plano de ação” e “Elaborar recomendações”. A execução da atividade “Realizar medições” implicou na coleta das QMEs a partir dos diagramas do modelo selecionado no plano de avaliação. Para isso, foi utilizada a ferramenta **SDPQualityForms**, quer retornou o conjunto de resultados apresentado na Tabela 57.

*Tabela 57. Resultados para QMEs da MDS-Fiocruz ajustada.*

| ND | NDD | Diagrama | NPTP | NPTU | NA | NADD | NBE | NBED | NFE | NFED | NPNR | NPR | NBC | NMD | NM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | 20 | 1-MDS.bpmn | 0 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 2 | 0 | 1 | 3 |
| | | 2-Desenvolver.bpmn | 4 | 5 | 0 | 0 | 1 | 1 | 2 | 2 | 0 | 6 | 0 | 0 | 0 |
| | | 2.1-Iniciar.bpmn | 4 | 1 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 2 |
| | | 2.2-Elaborar.bpmn | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| | | 2.2.1-Modelar processo.bpmn | 3 | 2 | 7 | 7 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 3 | 3 |
| | | 2.2.2-Especificar Requisitos.bpmn | 3 | 2 | 4 | 4 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | | 2.2.3-Projetar Sistema.bpmn | 3 | 1 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2.3-Construir.bpmn | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| | | 2.3.1-Implementar.bpmn | 0 | 2 | 5 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 |
| | | 2.3.2-Testar.bpmn | 2 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| | | 2.3.2.1-Implementar Testes.bpmn | 1 | 4 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | 2.3.2.2-Executar Testes.bpmn | 2 | 2 | 5 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 |
| | | 2.3.2.3-Aceitação.bpmn | 1 | 2 | 5 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 |
| | | 2.4-Implantar.bpmn | 1 | 1 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 3-Gerenciar Mudança.bpmn | 0 | 1 | 9 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 6 |
| | | Inspecionar Artefato.bpmn | 1 | 2 | 5 | 5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | P-Controlar.bpmn | 3 | 2 | 4 | 4 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 3 | 3 |
| | | P-Encerrar.bpmn | 2 | 0 | 4 | 4 | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 0 | 0 |
| | | P-Iniciar.bpmn | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | P-Planejar.bpmn | 4 | 1 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Min | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Max | 4 | 5 | 9 | 9 | 1 | 1 | 2 | 2 | 0 | 6 | 2 | 6 | 6 |
| | | **Média** | **2,05** | **1,90** | **3,35** | **3,35** | **0,20** | **0,20** | **0,30** | **0,30** | **0,00** | **1,00** | **0,10** | **1,10** | **1,20** |
| | | Desv. Padrão | 1,32 | 1,18 | 2,33 | 2,33 | 0,40 | 0,40 | 0,64 | 0,64 | 0,00 | 1,55 | 0,44 | 1,55 | 1,60 |
| | | Variância | 1,84 | 1,46 | 5,71 | 5,71 | 0,17 | 0,17 | 0,43 | 0,43 | 0,00 | 2,53 | 0,20 | 2,52 | 2,69 |

A ferramenta foi utilizada ainda para a realização dos cálculos das QMs a partir das QMEs coletadas. O resultado é apresentado na Tabela 58.

*Tabela 58. Resultados para QMs da MDS-Fiocruz ajustada.*

| Diagrama | CDM | COU | CME | CM | PD | REU | AC | RA | C |
|---|---|---|---|---|---|---|---|---|---|
| 1-MDS.bpmn | 1,000 | 1,000 | 1,000 | 0,333 | 0,000 | 0,000 | 1,000 | 0,500 | 0,667 |
| 2-Desenvolver.bpmn | 1,000 | 1,000 | 1,000 | 0,000 | 0,000 | 0,000 | 1,000 | 1,000 | 0,400 |
| 2.1-Iniciar.bpmn | 1,000 | 1,000 | 0,000 | 1,000 | 0,000 | 0,000 | 1,000 | 0,500 | 0,444 |
| 2.2-Elaborar.bpmn | 1,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 1,000 | 1,000 | 0,333 |
| 2.2.1-Modelar processo.bpmn | 1,000 | 1,000 | 1,000 | 1,000 | 0,000 | 0,000 | 1,000 | 0,125 | 0,615 |
| 2.2.2-Especificar Requisitos.bpmn | 1,000 | 1,000 | 1,000 | 0,000 | 0,000 | 0,000 | 1,000 | 0,200 | 0,500 |
| 2.2.3-Projetar Sistema.bpmn | 1,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,429 |
| 2.3-Construir.bpmn | 1,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 1,000 | 1,000 | 0,400 |
| 2.3.1-Implementar.bpmn | 1,000 | 1,000 | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,714 |
| 2.3.2-Testar.bpmn | 1,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 1,000 | 1,000 | 0,375 |
| 2.3.2.1-Implementar Testes.bpmn | 1,000 | 1,000 | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,286 |
| 2.3.2.2-Executar Testes.bpmn | 1,000 | 1,000 | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,556 |
| 2.3.2.3-Aceitação.bpmn | 1,000 | 1,000 | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,625 |
| 2.4-Implantar.bpmn | 1,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,600 |
| 3-Gerenciar Mudança.bpmn | 1,000 | 1,000 | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,900 |
| Inspecionar Artefato.bpmn | 1,000 | 1,000 | 1,000 | 0,000 | 0,000 | 0,200 | 0,000 | 0,000 | 0,625 |
| P-Controlar.bpmn | 1,000 | 1,000 | 1,000 | 1,000 | 0,000 | 0,250 | 0,000 | 0,000 | 0,444 |
| P-Encerrar.bpmn | 1,000 | 1,000 | 1,000 | 0,000 | 0,500 | 1,000 | 0,000 | 0,000 | 0,667 |
| P-Iniciar.bpmn | 1,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,500 |
| P-Planejar.bpmn | 1,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,444 |
| Min | 1,000 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,286 |
| Max | 1,000 | 1,000 | 1,000 | 1,000 | 0,500 | 1,000 | 1,000 | 1,000 | 0,900 |
| **Média** | **1,000** | **1,000** | **0,350** | **0,417** | **0,025** | **0,073** | **0,400** | **0,266** | **0,526** |
| Desv. Padrão | 0,000 | 0,000 | 0,477 | 0,482 | 0,109 | 0,223 | 0,490 | 0,396 | 0,146 |
| Variância | 0,000 | 0,000 | 0,239 | 0,244 | 0,013 | 0,052 | 0,253 | 0,165 | 0,023 |

A execução da atividade “Aplicar plano de ação” considerou os critérios propostos no subprocesso de planejamento. A partir destes critérios e dos resultados de QMEs e QMs, foi realizado um conjunto de análises, conforme apresentado na Tabela 59.

*Tabela 59. Análise comparativa dos resultados.*

| Critério de decisão | Análise |
|---|---|
| CDM – Comparar com o resultado da primeira avaliação. | O resultado de CDM na primeira avaliação foi 0, e na segunda avaliação o resultado foi 1. |

| | |
|---|---|
| COU – Comparar com o resultado da primeira avaliação. | O resultado de COU na primeira avaliação foi de 0,939, e na segunda avaliação o resultado foi 1. Isto representou uma diferença positiva de 6,5%. |
| CME – Comparar com o resultado da primeira avaliação. | O resultado de CME na primeira avaliação foi de 0, e na segunda avaliação o resultado foi de 0,35. |
| CM – Comparar com o resultado da primeira avaliação. | O resultado de CM na primeira avaliação foi de 0, e na segunda avaliação o resultado foi de 0,417. |
| PD – Comparar com o resultado da primeira avaliação. | O resultado de PD na primeira avaliação foi 0,017, e na segunda avaliação o resultado foi de 0,025. Isto representou uma diferença positiva de 47,06%, embora a diferença absoluta tenha sido relativamente pequena. |
| REU – Comparar com o resultado da primeira avaliação. | O resultado de REU na primeira avaliação foi de 0,017, e na segunda avaliação o resultado foi de 0,0,73. Isto representou uma diferença positiva de 329,41%, embora a diferença absoluta tenha sido relativamente pequena. |
| AC – Comparar com o resultado da primeira avaliação. | O resultado de AC na primeira avaliação foi de 0,364, e na segunda avaliação o resultado foi de 0,400. Isto representou uma diferença positiva de 9,89%, embora a diferença absoluta tenha sido relativamente pequena. |
| RA – Comparar com o resultado da primeira avaliação. | O resultado de RA na primeira avaliação foi de 0,111, e na segunda avaliação o resultado foi de 0,266. Isto representou uma diferença positiva de 139,64%. |
| C – Comparar com o resultado da primeira avaliação. | O resultado de C na primeira avaliação foi de 0,392, e na segunda avaliação o resultado foi de 0,526. Isto representou uma diferença positiva de 34,18%. |

A execução da atividade “Elaborar recomendações” considerou os resultados agregados para as subcaracterísticas da qualidade, produzindo desta maneira as informações para o documento “Resultado da avaliação”, conforme apresentado na Tabela 60.

*Tabela 60. Resultado da segunda avaliação.*

| **Característica** | **Subcaracterística** | **Valor** | **Análise** | **Recomendação** |
|---|---|---|---|---|
| Usabilidad | SRA - Reconhecimento apropriado | 1,000 | A QM CDM foi elevada até o seu limite para este modelo. | Ná há recomendação de ajuste. |

| | | | | |
|---|---|---|---|---|
| | SA - Apreensibilidade | 1,00 | A QM CME foi elevada até o seu limite para este modelo. | Ná há recomendação de ajuste. |
| | SO - Operabilidade | 0,475 | As QMs CM e PD foram elevadas, aumentando SO em de 0,009 para 0,475. | Ainda há espaço para elevação da subcaracterística SO. No entanto, esta segunda avaliação tem o objetivo de estabelecer parâmetros para comparação com a primeira avaliação. |
| | SP - Proteção de erros do usuário | 0,073 | A QM REU foi elevada, aumentando SP de 0,017 para 0,073. | Ainda há espaço para elevação da subcaracterística SP. No entanto, esta segunda avaliação tem o objetivo de estabelecer parâmetros para comparação com a primeira avaliação. |
| Manutenibilidade | SM - Modularidade | 1,000 | A QM AC não foi alterada. | Não há recomendação de ajuste. |
| | SR - Reusabilidade | 0,666 | A QM RA foi elevada, aumentando SR de 0,305 para 0,666. | Ainda há espaço para elevação da subcaracterística RA. No entanto, esta segunda avaliação tem o objetivo de estabelecer parâmetros para comparação com a primeira avaliação. |
| | SMD - Modificabilidade | 0,763 | As QMs e AC foram elevadas, aumentando SMD de 0,696 para 0,763. | Ainda há espaço para elevação da subcaracterística SMD. No entanto, esta segunda avaliação tem o objetivo de estabelecer parâmetros para comparação com a primeira avaliação. |

### 7.5.5 Execução do subprocesso “Finalizar a avaliação”

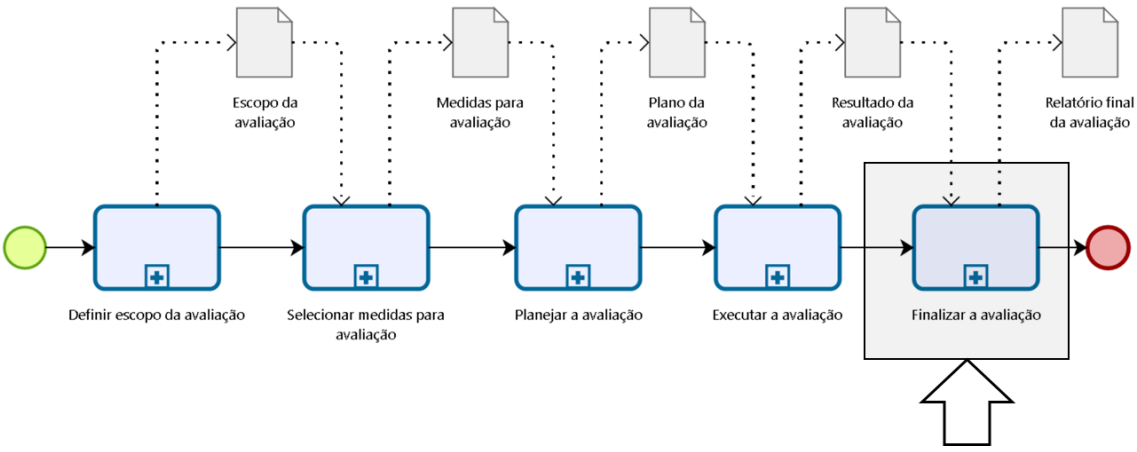



*Figura 38. Execução do subprocesso "Finalizar a avaliação", segunda avaliação.*

O quinto e último subprocesso do processo de avaliação da qualidade de modelos de processo de software é o “Finalizar a avaliação”, conforme Figura 38. Neste subprocesso são executadas as atividades “Elaborar o relatório da avaliação”, “Identificar lições aprendidas” e “Entregar documentação”. A execução deste subprocesso produziu as informações para o documento “Relatório final da avaliação”, como pode ser observado na Tabela 61.

*Tabela 61. Relatório final da avaliação.*

| Campo | Objetivo |
|---|---|
| Organização onde foi realizada a avaliação | Fundação Oswaldo Cruz. |
| Modelo avaliado | MDS-Fiocruz ajustada. |
| Versão | 4.1 |
| Período da avaliação | De 21 a 24 de novembro de 2016. |
| Diagramas avaliados | • 1-MDS.bpmn<br>• 2-Desenvolver.bpmn<br>• 2.1-Iniciar.bpmn<br>• 2.2-Elaborar.bpmn<br>• 2.2.1-Modelar processo.bpmn<br>• 2.2.2-Especificar Requisitos.bpmn<br>• 2.2.3-Projetar Sistema.bpmn<br>• 2.3-Construir.bpmn<br>• 2.3.1-Implementar.bpmn<br>• 2.3.2-Testar.bpmn<br>• 2.3.2.1-Implementar Testes.bpmn<br>• 2.3.2.2-Executar Testes.bpmn<br>• 2.3.2.3-Aceitação.bpmn<br>• 2.4-Implantar.bpmn<br>• 3-Gerenciar Mudança.bpmn<br>• Inspecionar Artefato.bpmn<br>• P-Controlar.bpmn<br>• P-Encerrar.bpmn<br>• P-Iniciar.bpmn<br>• P-Planejar.bpmn |
| Considerações sobre o resultado | O modelo de processo de software ajustado foi considerado adequado para cumprir sua nova função institucional, ou seja, se tornar o modelo padrão de desenvolvimento para todas as Unidades da Fiocruz. Apesar disso, ficou a recomendação de realização de nova avaliação dentro do prazo de 1 ano. |
| Curso de ação decidido | Implementar em nível institucional o novo modelo de processo de software. |
| Ajustes recomendados pela avaliação | Não houve recomendação de ajustes, pois não era este o propósito desta avaliação. |
| Limitações/Restrições | Não houve limitações/restrições impostas a este projeto de avaliação. |

### 7.5.6 Avaliação da percepção dos usuários na melhora de Usabilidade

Ao se comparar os resultados da primeira avaliação com os resultados da segunda avaliação é possível perceber que houve uma diferença positiva dos resultados nas subcaracterísticas de “Usabilidade” e de “Manutenibilidade”. Contudo, a diferença foi maior para “Usabilidade”. A partir desta constatação, decidiu-se pela avaliação da percepção dos usuários quanto a esta diferença. As seguintes subcaracterísticas de “Usabilidade” forma consideradas:

- Reconhecimento apropriado
- Apreensibilidade
- Operabilidade
- Proteção contra erros
- Estética
- Acessibilidade

Considerando as definições de (ISO/IEC, 2014) para cada uma das subcaracterísticas foi elaborada uma afirmação, como apresentado na Tabela 62, para as quais os participantes deveriam indicar seu grau de concordância. Estas opiniões foram coletadas utilizando-se uma escala *Likert* (ALEXANDRE, 2003) (ALLEN e SEAMAN, 2007), estabelecida da seguinte maneira:

1. Discordo totalmente
2. Tendo a discordar
3. Sou incapaz de opinar
4. Tendo a concordar
5. Concordo plenamente

*Tabela 62. Subcaracterísticas de qualidade a afirmações que as definem para o escopo da pesquisa.*

| Subcaracterística | Afirmação |
|---|---|
| Reconhecimento apropriado | 1 - O modelo é prático e parece totalmente adequado para uso por uma equipe de desenvolvimento de software |
| Apreensibilidade | 2 - O modelo é muito fácil de se ensinar e de se aprender |
| Operabilidade | 3 - Há características neste modelo que facilitam muito o seu uso |
| Proteção contra erros | 4 - O modelo deixa claro os erros podem acontecer e assim ajuda a evitá-los e ou tratá-los |
| Estética | 5 - A estrutura visual dos elementos do modelo formam um conjunto muito agradável e bem organizado |

| | |
|---|---|
| Acessibilidade | 6 - O modelo é tão simples e prático que qualquer profissional de sistemas poderia compreendê-lo facilmente |

A pesquisa de percepção foi baseada em um dos diagramas da MDS-Fiocruz original, que foi ajustado na segunda versão desta MDS. Os colaboradores da Fiocruz conheciam e utilizavam o modelo original, e isto poderia representar um viés nos resultados. De acordo com a recomendação de (LAKATOS e MARCONI, 2010) optou-se pela utilização de um grupo de controle, ou seja, um conjunto de participantes que não conhecessem o modelo original e que não tivessem qualquer ligação com as atividades de desenvolvimento de sistemas da Fiocruz. O grupo de controle, com 7 participantes, foi composto de alunos de mestrado, doutorado e professores da Universidade Federal do Rio de Janeiro (UFRJ), das disciplinas relacionadas a Informática e Engenharia de Software.

#### *7.5.6.1 Pesquisa com o grupo de controle*

Os participantes do grupo de controle observaram os dois diagramas (CLORO e IODO), e preencheram o formulário com suas avaliações. Não havia indicação alguma sobre qual diagrama era o original e qual diagrama era o ajustado. As opiniões para cada afirmação foram registradas com um "X" com base nos identificadores já apresentados ("Discordo totalmente", "Tendo a discordar", "Sou incapaz de opinar", "Tendo a concordar", e "Concordo plenamente"), ao invés de responder com números (1, 2, 3, 4 ou 5).

O objetivo foi afastar a ideia de um valor razão que pudesse ser base de cálculos aritméticos. Em seguida, as seleções foram contadas, conforme apresentados na Figura 39.

É importante ressaltar que, para facilitar a compreensão, o "Discordo totalmente" e o "Tendo a discordar" foram agrupados por soma das contagens em "Discorda", e o "Tendo a concordar" e "Concordo plenamente" foram agrupados por soma das contagens em "Concorda". As seleções "Sou incapaz de opinar" são apresentadas como "Não opinou".

É possível perceber que para o grupo CLORO há uma contagem maior de "Concorda" do que das demais opções em todas as subcaracterísticas, exceto em "Operabilidade", onde há um empate. Já no grupo IODO apenas em "Estética" e "Apreensibilidade" há uma contagem maior para "Concorda". Uma visão um pouco mais sintética dessas opiniões pode ser observada na Figura 40.

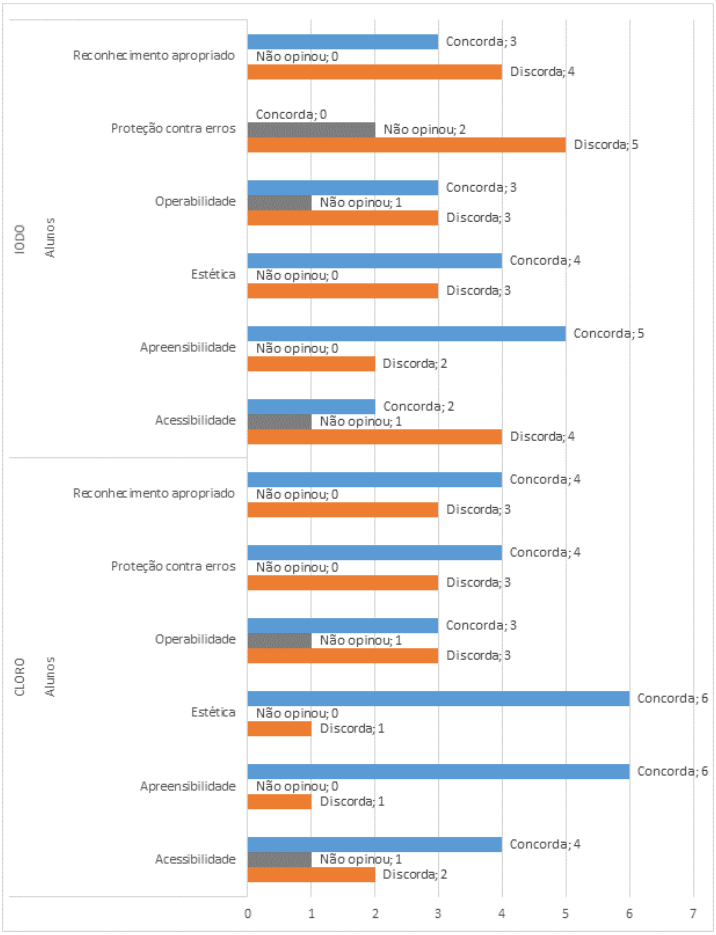



*Figura 39. Contagem das seleções aplicadas aos modelos pelo grupo de controle.*

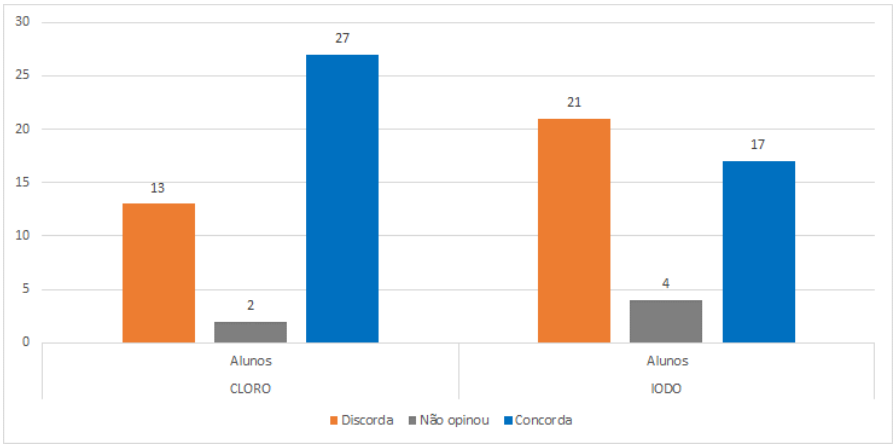



*Figura 40. Visão da Característica Usabilidade pelo grupo de controle.*

Nesta figura, que agora considera a contagem do ponto de vista da “Usabilidade” como um todo, e não mais de suas subcaracterísticas, é possível perceber uma vantagem na contagem das opiniões positivas para o modelo CLORO, sendo 27 concordâncias para este modelo e apenas 17 concordâncias para o modelo IODO.

Na Figura 41 é apresentada uma comparação entre o modelo IODO, diagrama original da MDS Fiocruz, e o modelo CLORO, diagrama equivalente, porém ajustado segundo as recomendações da avaliação realizada. Considerando apenas as concordâncias e discordâncias, nota-se uma evolução de 45% para 68% nas concordâncias, e uma redução de 55% para 32% nas discordâncias.

Como base nestas informações, há indícios da possibilidade de que os participantes do grupo de controle tenham percebido um aumento da “Usabilidade” do diagrama após a aplicação dos ajustes recomendados.

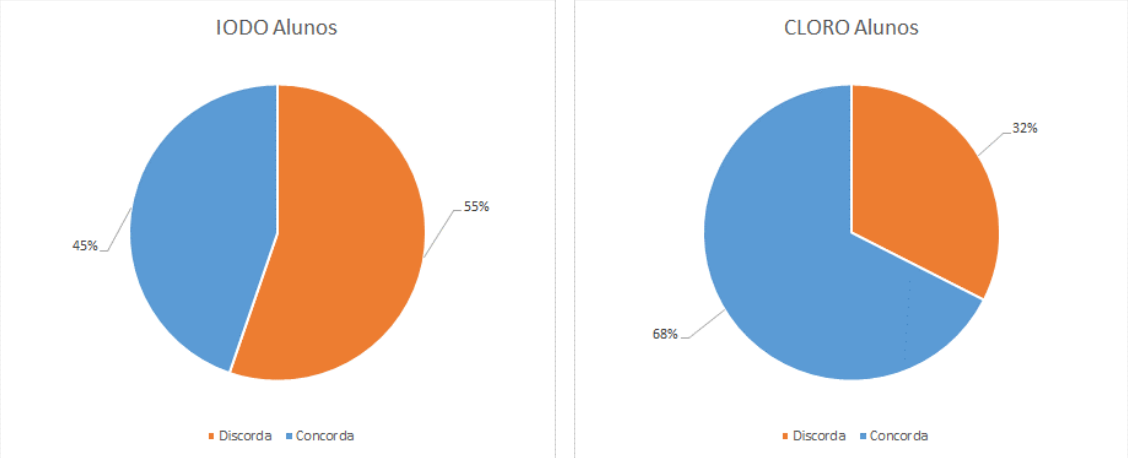



*Figura 41. Mudança de percepção do grupo de controle da “Usabilidade” entre diagramas.*

### *7.5.6.2 Pesquisa com o grupo Fiocruz*

O mesmo protocolo foi aplicado para o grupo Fiocruz. Este grupo foi composto de 12 colaboradores das áreas de Qualidade, Modelagem de Processos, Elicitação de Requisitos, e Desenvolvimento de Sistemas. As contagens foram agrupadas de maneira idêntica ao da pesquisa com o grupo de controle. O resultado pode ser visto na Figura 42.

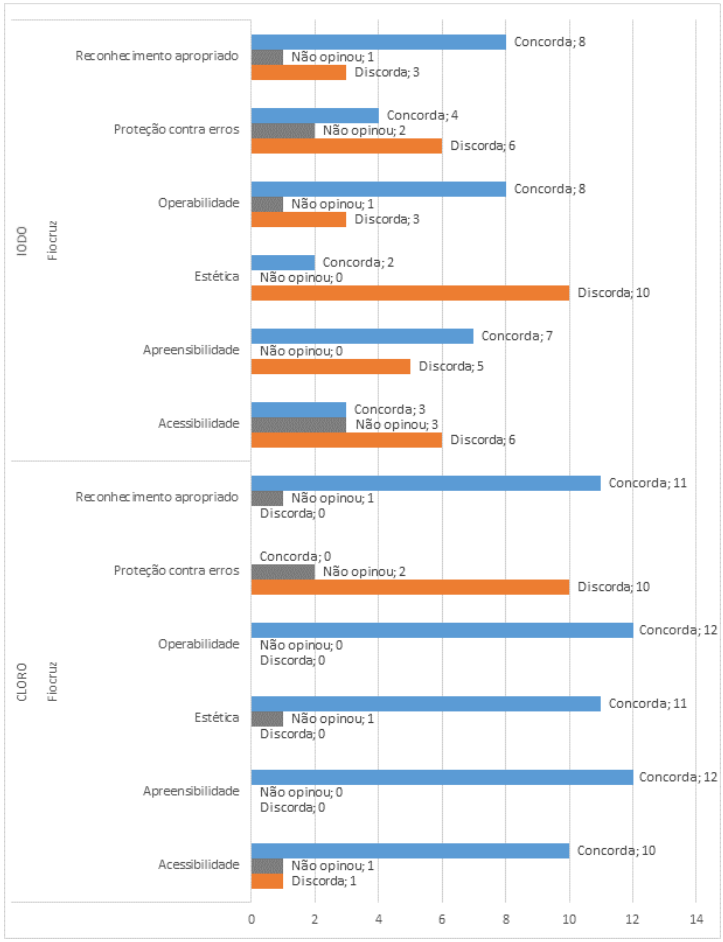



*Figura 42. Contagem das seleções aplicadas aos modelos pelo grupo Fiocruz.*

As concordâncias superaram as discordâncias no modelo IODO para as subcaracterísticas de “Reconhecimento apropriado”, “Operabilidade” e “Apreensibilidade”. Já no caso do modelo CLORO, as concordâncias foram maiores do que as discordâncias em todas as subcaracterísticas, excetuando a subcaracterística “Proteção contra erros”.

A contagem das opiniões do grupo Fiocruz seguiu a um padrão similar ao da contagem realizada com o grupo de controle. Uma visão sintética dessa análise pode ser vista na Figura 43. No modelo CLORO há 56 concordâncias totais contra 32 concordâncias totais do modelo IODO.

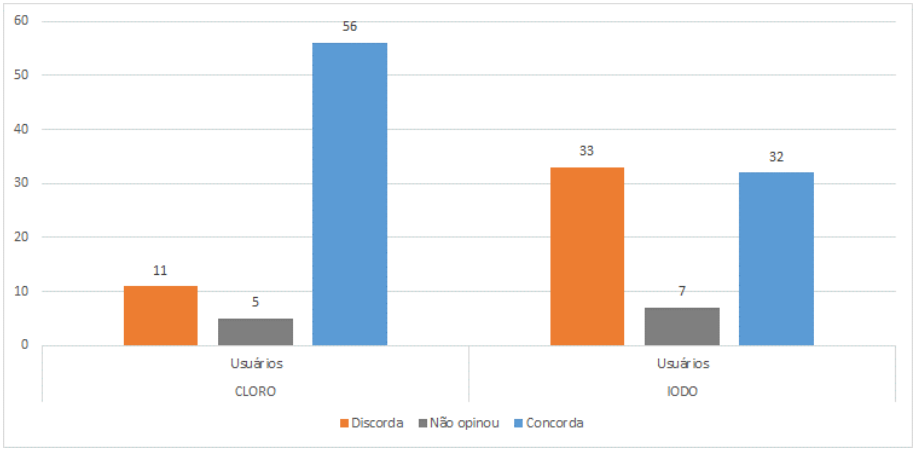



*Figura 43. Visão da Característica Usabilidade pelo grupo Fiocruz.*

Na Figura 44 é apresentada uma comparação entre o modelo IODO, diagrama original da MDS Fiocruz, e o modelo CLORO, diagrama equivalente, porém ajustado segundo as recomendações da avaliação realizada. Considerando apenas as concordâncias e discordâncias, nota-se uma evolução de 49% para 84% nas concordâncias, e uma redução de 51% para 16% nas discordâncias.

Como base nestas informações, há indícios da possibilidade de que os participantes do grupo Fiocruz tenham percebido um aumento da "Usabilidade" do diagrama após a aplicação dos ajustes recomendados. No entanto, a diferença na mudança de percepção neste grupo foi de 35 pontos percentuais contra 23 pontos percentuais para o grupo de controle.

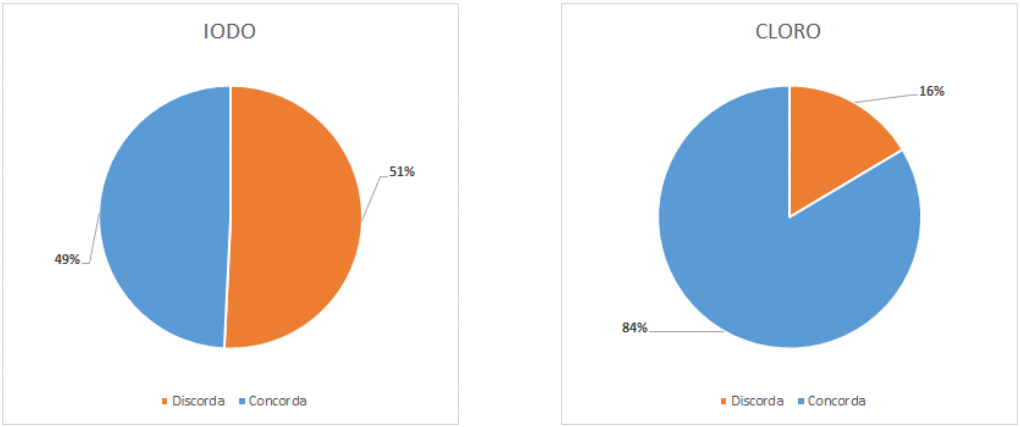



*Figura 44. Mudança de percepção do grupo Fiocruz da "Usabilidade" entre diagramas.*

A Figura 45 apresenta uma comparação entre as opiniões dos dois grupos, ou seja, do grupo de controle e do grupo Fiocruz. É interessante notar que as opiniões dos dois grupos referentes ao modelo IODO (modelo original) é bastante similar, variando em apenas alguns pontos percentuais. No caso do modelo CLORO, embora o número de concordâncias seja maior do que no IODO no grupo de controle, ele é ainda maior no grupo Usuários.

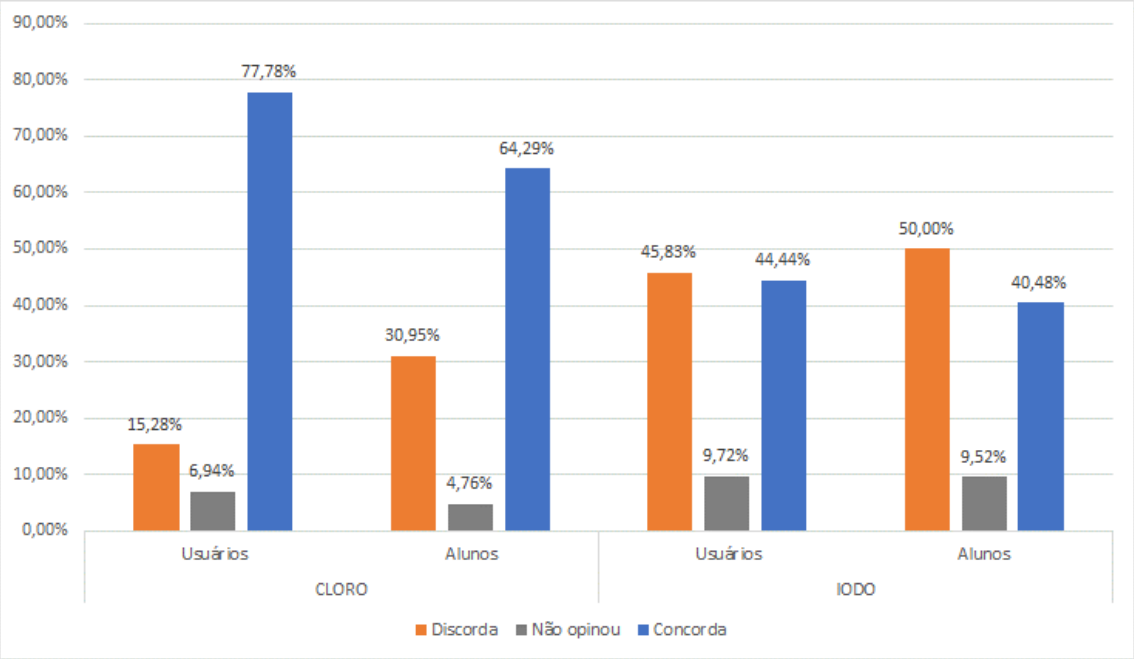



*Figura 45. Comparativo das opiniões dos grupos de controle e Fiocruz.*

Embora ambos os grupos tenham mostrado indícios de aumento da percepção da "Usabilidade", o grupo Fiocruz apresentou uma mudança de percepção significativamente maior em relação ao grupo de controle, o que pode indicar um viés.

## 7.6 Conclusão do capítulo

O objetivo deste capítulo foi o de executar o processo para a avaliação de um modelo de processo real, no intuito de avaliar a viabilidade e a utilidade do processo de avaliação e do modelo de medição, propostos nos capítulos anteriores. O processo de avaliação foi executado sobre o processo de software MDS-Fiocruz, produzindo recomendações de ajustes. Estes ajustes foram implementados, e nova avaliação foi realizada sobre esta MDS-Fiocruz ajustada. Os resultados das duas avaliações foram comparados, havendo uma diferença positiva especialmente para a característica de "Usabilidade". Em seguida,

uma pesquisa de opinião foi conduzida com usuários do modelo e com um grupo de controle, para avaliar esta diferença de percepção da “Usabilidade” entre o modelo original e o modelo ajustado.

No que se refere à pesquisa de percepção, embora os resultados não possam ser considerados conclusivos, oferecem indícios da utilidade dos processos propostos pelo presente trabalho, e demandam pela realização de novas pesquisas de percepção no sentido de consolidar esta primeira análise.

# 8 Conclusão

## 8.1 Introdução do capítulo

Este trabalho tratou da avaliação da qualidade de modelos de processo de software representados em notação BPMN. Foi destacada a importância do processo de software e de sua representação para a melhoria da qualidade do software e foram considerados os indícios de que há espaço para pesquisas relacionadas à melhoria da qualidade do modelo de processo de software.

O objetivo principal do trabalho foi o de propor um processo de avaliação da qualidade para modelos de processo de software, baseado em medições associadas a um modelo de qualidade de referência. O escopo do trabalho foi delimitado para modelos de processo de software representados em notação BPMN, usando como referência as características de “Usabilidade” e “Manutenibilidade” do modelo de qualidade proposto pela ISO/IEC 25.010. A partir disso, as contribuições do trabalho foram a definição de um processo para elaboração do modelo de medição e a subsequente execução deste processo, com a definição de um processo de avaliação de modelos de processo de software e sua subsequente execução, aplicada a um processo real.

Alguns conceitos fundamentais para o trabalho foram apresentados. Um destes conceitos foi o de qualidade em modelos conceituais, onde o modelo conceitual foi definido como a explicitação de um conjunto de especificações relevantes que contribuem para a compreensão de determinado problema, em determinado domínio, o que inclui o modelo de processo de software. Foram apresentados os modelos de avaliação da qualidade para modelos conceituais de (KROGSTIE, LINDLAND e SINDRE, 1995) e de (WAND e WEBER, 2002), destacando que estes modelos buscavam uma forma de organização das características e subcaracterísticas de qualidade, com o intuito de reduzir a ambiguidade. No entanto, estes trabalhos não propuseram mecanismos de medição ou processos para construção do modelo de medição e para avaliação dos modelos conceituais. Também foram apresentadas notações para representação de processos, além da linguagem natural, tais como BPMN, SPEM, IDEF0, YWAL, Petri Nets, Little-JIL e ARIS. Ainda, foram apresentados os modelos de avaliação da qualidade específicos da área de Engenharia de Software, tais como o de (MCCALL, RICHARDS e WALTERS, 1977) e de (BOEHM, BROWN e LIPOW, 1976), e a família de normas ISO/IEC 25.000, que foi uma evolução do conjunto de normas ISO/IEC 9.126. Foram apresentadas as seis divisões da família

ISO/IEC 25.000 e uma breve descrição para cada uma das normas que compõem essas divisões.

Foram considerados os trabalhos relacionados à pesquisa, voltados para a avaliação da qualidade de modelos conceituais, incluindo modelos de processo de negócio e modelos de processo de software. Ainda, foram analisados trabalhos que buscaram definir medidas a serem coletadas a partir dos modelos de processo de software e que pudessem ser utilizados para a avaliação da qualidade destes modelos, tanto de maneira individual quanto associados a determinadas características de qualidade. A análise destes trabalhos forneceu indícios de que há espaço para mais pesquisas nesta área, em especial no que se refere a um processo para elaboração de um modelo de medição, um modelo de medição propriamente dito, e um processo para avaliação da qualidade de modelos de processo de software.

Considerando isso, foi proposto um processo para a construção de um modelo de medição que pudesse ser utilizado para avaliar a qualidade de modelos de processo de software. Este processo foi baseado na família de normas ISO/IEC 25.000. O processo produz um conjunto de medidas básicas, as QMEs e um conjunto de medidas derivadas, as QMs. O modelo de medição seria a integração do modelo de qualidade, com características e subcaracterísticas, e os conjuntos de medidas (QMs e QMEs)

O processo de construção de um modelo de medição foi executado, definindo um conjunto de QMEs aplicáveis ao modelo de processo de software e um conjunto de QMs aplicáveis ao modelo de processo de software. Desse modo, foi definido um modelo de medição contendo as relações entre características de qualidade, subcaracterísticas de qualidade, QMs e QMEs.

Para aplicar o modelo de medição foi proposto um processo para a avaliação de modelos de processo de software. Este processo usou como referência a família de normas ISO/IEC 25.000, mais especificamente a norma ISO/IEC 25.040. Este processo de avaliação foi executado tendo por objeto um modelo de processo de software real, no intuito de avaliar a viabilidade e a utilidade deste processo de avaliação e do modelo de medição. O processo de software utilizado foi o da Fundação Oswaldo Cruz, a MDS-Fiocruz. A execução do processo de avaliação produziu algumas recomendações de ajustes, que foram implementadas, resultando em uma MDS-Fiocruz ajustada. Uma nova execução do processo de avaliação foi realizada sobre esta MDS-Fiocruz ajustada. Os resultados das duas avaliações foram comparados, havendo uma diferença positiva

especialmente para a característica de “Usabilidade”. Em seguida, uma pesquisa de opinião foi conduzida com usuários do modelo e com participantes de um grupo de controle, para avaliar a percepção desta diferença de “Usabilidade” entre o modelo original e o modelo ajustado.

## 8.2 Contribuições

Neste trabalho, foi apresentada uma abordagem sistemática para avaliar a qualidade dos modelos de processo de software em relação às características de usabilidade e manutenibilidade. Esta abordagem envolveu a proposição de dois processos e de um modelo de medição aplicável a modelos de processo de software.

Uma contribuição foi a proposição de um processo para a construção do modelo de medição, e o próprio modelo de medição resultante da execução deste processo. Tanto o processo quanto o modelo utilizaram como referência normas internacionais de ampla aceitação.

Sobre o modelo de medição, a contribuição do trabalho foi a seleção e adaptação para o modelo do processo de software, das QMEs propostas pela ISO 25.021, originalmente aplicáveis ao produto de software, além de uma revisão de literatura para identificar QMEs aplicadas a modelos de processo de software. Estas duas etapas, a revisão de literatura e a seleção/adaptação das QMEs da ISO, resultaram em um conjunto de 26 QMEs aplicáveis ao modelo do processo de software. Este conjunto poderá utilizado por outros estudos, tanto no sentido de identificar eventuais inadequações quanto no sentido de propor a evolução do mesmo.

Outra contribuição foi a proposição de um processo para avaliação de modelos de processo de software, sendo referenciados também na família de normas ISO/IEC 25.000, mantendo harmonia com o processo anterior e com o modelo de medição. Este processo poderá ser reaplicado, avaliado, e melhorado por outros trabalhos de pesquisa, refinando este processo e tornando-o cada vez mais eficiente e eficaz.

O fato de os processos e o modelo de medição estarem alinhados com normas internacionais da área de Engenharia de Software contribui para sua maior aceitação e utilização, podendo se tornar uma referência inicial para novos estudos na área de qualidade para modelos de processo de software.

A ferramenta **SDPQualityForms**, construída para apoio ao processo de avaliação na coleta de informações a partir do modelo de processo de software, também pode ser

considerada uma contribuição. Esta ferramenta poderá ser utilizada por outros pesquisadores para a análise da qualidade de modelos conceituais de processos, tanto de software quanto de negócio, viabilizando análises comparativas com o presente trabalho e estabelecendo índices cada vez mais confiáveis. A ferramenta está disponibilizada no site do grupo de pesquisa Prisma (http://prisma.cos.ufrj.br/).

## 8.3 Trabalhos futuros

Futuros trabalhos poderão aplicar o processo de avaliação e o modelo para medição a outros processos de software reais, buscando consolidar os resultados obtidos pelo presente, e ainda aprimorá-los. Em apoio a este tipo de pesquisa, seria adequada a construção de uma ferramenta como o SDPQualityForms em uma versão WEB, que pudesse ser utilizada por diversos pesquisadores, e até profissionais de mercado, de modo que o conjunto de medidas coletadas a partir dos mais diversos modelos de processo de software constituíssem, com o passar do tempo, referências cada vez mais sólidas para a avaliação da qualidade em modelos deste tipo. Neste caso, os dados estariam integrados em um ponto único, e automaticamente disponíveis para análise.

Ainda no âmbito da ferramenta SDPQualityForms, com o objetivo de ampliar sua capacidade de suporte a melhoria do modelo do processo de software, futuras pesquisas poderão ser conduzidas no sentido de se propor um mecanismo de recomendação a partir dos resultados do modelo de medição, de modo que ajustes específicos no modelo possam ser recomendados automaticamente pela própria ferramenta.

São necessárias também pesquisas adicionais que permitam ampliar o modelo de medição. Utilizando o processo elaboração de modelo para medição proposto pelo presente trabalho será possível ampliar o número de QMEs coletáveis a partir dos modelos de processo de software, o número de QMs que podem ser calculadas a partir dessas QMEs, e ainda o número de subcaracterísticas e características de qualidade que podem ser abrangidas em futuros modelos de medição.

Ainda, novas pesquisas poderão ser conduzidas no sentido de avaliar a possibilidade de aplicação das propostas deste trabalho, ou seja, um modelo de medição, um processo de avaliação, e uma ferramenta de coleta automática de medidas, para outras notações que sejam utilizadas para a representação do modelo do processo de software, tais como SPEM, Aris EPC, e inclusive para linguagem natural estruturada.

Com a eventual ampliação para avaliação da qualidade de modelos construídos em outras notações e linguagens, e com ampliação das medidas do modelo de medição, é possível que o processo de avaliação, o modelo de medição e a ferramenta **SDPQualityForms** possam servir de apoio em avaliações da qualidade do processo de software, tais como as conduzidas no âmbito do MPS.BR ou do CMMI. Uma linha de pesquisa possível seria a avaliação de viabilidade desse tipo de aplicação.

Por fim, para além da avaliação da qualidade de processos de software, trabalhos poderão ser conduzidos especificamente sobre conjuntos de processos de negócio, a fim de estabelecer um modelo para medição e um processo de avaliação calibrados para este tipo de modelo, e a **SDPQualityForms** poderia ser utilizada neste tipo de pesquisa.

## Bibliografia

ALEXANDRE, J. W. C. Análise do número de categorias da escala Likert aplicada a gestão pela qualidade total através da teoria da resposta ao item. **Encontro Nacional de Engehnaria de PRodução**, 2003.

ALLEN, I. E.; SEAMAN, C. A. Likert scales and data analyses. **Quality Progress**, v. 40, n. 7, p. 64-65, July 2007. ISSN 0033524X.

AL-QUTAISH, R. E. **An investigation of the weaknesses of the ISO 9126 international standard**. International Conference on Computer and Electrical Engineering, ICCEE 2009. Dubai, United Arab Emirates: International Association of Computer,Science and Information Technology. 2009. p. 275-279.

ALSHAYEB, M.; LI, W. An empirical study of system design instability metric and design evolution in an agile software process. **Journal of Systems and Software**, v. 74, n. 3, p. 269-274, February 2005. ISSN 01641212.

ANDRADE, J. et al. A methodological framework for viewpoint-orientend conceptual modeling. **IEEE Transactions on Software Engineering**, v. 30, n. 5, p. 283-294, May 2004. ISSN 0098-5589/.

BADDOO, N.; HALL, T. De-motivators for software process improvement: An analysis of practitioners' views. **Journal of Systems and Software**, v. 66, n. 1, p. 23-33, April 2003. ISSN 01641212.

BARRETO, A. S.; MURTA, L. G. P.; ROCHA, A. R. C. Software process definition: A reuse-based approach. **Journal of Universal Computer Science**, v. 17, n. 13, p. 1765-1799, 2011. ISSN 0958695X.

BASILI, V.; GREEN, S. Software Process Evolution at the SEL. **IEEE Software**, v. 11, n. 4, p. 58-66, July 1994. ISSN 07407459.

BERANDER, P.; JONSSON, P. A goal question metric based approach for efficient measurement framework definition. **ISESE'06 - Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering**, Rio de Janeiro, 21-22 September 2006. 316-325.

BERKI, E.; GEROGIADOU, E.; HOLCOMBE, M. Requirements engineering and process modelling in software quality management - Towards a generic process metamodel. **Software Quality Journal**, v. 12, n. 3, p. 265-283, September 2004. ISSN 09639314.

BOEHM, B. W.; BROWN, J. R.; LIPOW, M. **Quantitative Evaluation of Software Quality**. Proceedings of the 2Nd International Conference on Software Engineering. San Francisco, California, USA: IEEE Computer Society Press. 1976. p. 592-605.

BROOKS, F. P. . J. Essence and Accidents of Software Engineering. **Computer**, v. 20, n. 4, p. 10-19, April 1987. ISSN 00189162.

BUTTLER, K. **The economics benefits of software process improvement**. [S.l.]: Cross Talk, 1995.

CAMPOS, A. L. N.; OLIVERIA, T. C. **Modeling work processes and software development:** Notation and tool. ICEIS 2011 - Proceedings of the 13th International Conference on Enterprise Information Systems 3 ISAS , pp. 337-343. China: [s.n.]. 2011. p. 226-237.

CANFORA, G. et al. A family of experiments to validate metrics for software process models. **Journal of Systems and Software**, August 2005. 113-129.

CANFORA, G. et al. **A proposal and empirical validation of metrics to evaluate the maintainability of software process models**. IEEE Instrumentation and Measurement Technology Conference. [S.l.]: [s.n.]. 2006.

CARNAGHAN, C. Business process modeling approaches in the context of process level audit risk assessment: An analysis and comparison. **International Journal of Accounting Information Systems** , 28 October 2006. pp. 170-204.

CASS, A. G. et al. **Little-JIL/Juliette:** a process definition language and interpreter. Proceedings - International Conference on Software Engineering. [S.l.]: [s.n.]. 2000. p. 754-757.

CHINOSI, M.; TROMBETTA, A. BPMN: An introduction to the standard. **Computer Standards and Interfaces**, v. 34, n. 1, p. 124-134, January 2011. ISSN 09205489.

CLARKE, P.; O'CONNOR, R. V. An empirical examination of the extent of software process improvement in software SMEs. **Journal of software: Evolution and Process** , v. 25, n. 9, p. 981-998, September 2013. ISSN 20477481.

COALLIER, F.; AZUMA, M. **Introduction to Software Engineering Standard**. ISO/IEC. [S.l.], p. 18. 2013.

COLEMAN, G.; O'CONNOR, R. Investigating software process in practice: A grounded theory perspective. **Journal of Systems and Software**, v. 81, n. 5, p. 772-784, May 2008.

CONRADI, R.; FUGGETA, A. Improving software process improvement. **IEEE Software**, v. 19, n. 4, p. 92-99, July 2002. ISSN 07407459.

COOK, J. E.; WOLF, A. L. Software process validation: Quantitatively measuring the correspondence of a process to a model. **ACM Transactions on Software Engineering and Methodology**, v. 8, n. 2, p. 147-176, April 1999. ISSN 1049331X.

DA SILVA, B. C.; MACIEL, R. S. P.; RAMALHO, F. Evaluating maintainability of MDA software process models. **Lecture Notes in Computer Sciences**, v. 7983, p. 199-2013, June 2013. ISSN 03029743.

DAVIES, I. et al. How do practitioners use conceptual modeling in practice? **Data and Knowledge Engineering** , v. 58, n. 3, p. 358-380, September 2006. ISSN 0169023X.

DAVIS, A. **Software Requirements:** Analysis and Specification. Englewood Cliff, NJ, USA: Prentice-Hall, 1990.

DENARO, G.; PEZZÈ, M. **An empirical evaluation of fault-proneness models**. Proceedings - International Conference on Software Engineering. Orlando, United States: [s.n.]. 2002. p. 241-251.

DIAZ, M.; SLIGO, J. How software process improvement helped motorola. **IEEE Software** , v. 14, n. 5, p. 75-80, 1997. ISSN 07407459.

DIBA, T. Instrument for measuring the key factors of success in software process improvement. **Empirical Software Engineering** , v. 5, n. 4, p. 357-390, December 2000. ISSN 13823256.

DYBA, T.; KITCHENHAM, B. A.; JORGENSEN, M. Evidence-based software engineering for practitioners. **IEEE Software**, v. 22, n. 1, p. 58-65, January 2005. ISSN 07407459.

ECLIPSE FOUNDATION. OpenUP. **EPF Eclipse**, 2013. ISSN openup_1.5.1.4_20120530. Disponivel em: <http://epf.eclipse.org/wikis/openup/>. Acesso em: 14 jan. 2014.

FALESSI, D.; SHAW, M.; MULLEN, K. Achieving and maintaining CMMI maturity level 5 in a small organization. **IEEE Software**, v. 31, n. 5, p. 80-86, October 2014. ISSN 07407459.

FANTECHI, A. et al. **Application of linguistic techniques for Use Case analysis**. Proceedings of the IEEE International Conference on Requirements Engineering. Essen, Germany: [s.n.]. 2002. p. 157-164.

FERREIRA, A. I. F. et al. **ISO 9001:** 2000, MPS.BR Nível F e CMMI Nível 3: Uma estratégia de melhoria de processos na BL Informática. V Simpósio Brasileiro de Qualidade de Software. Vila Velha, Brasil: [s.n.]. 2006. p. 375-382.

FERREIRA, A. L.; MACHADO, R. J.; PAULK, M. C. **An approach to software process design and implementation using transition rules**. Proceedings - 37th EUROMICRO Conference on Software Engineering and Advanced Applications. [S.l.]: [s.n.]. 2011. p. 330-333.

FIOCRUZ. Fiocruz - A Fundação. **Fiocruz**, 2017. Disponivel em: <http://portal.fiocruz.br/pt-br/content/funda%C3%A7%C3%A3o>. Acesso em: 10 jan. 2017.

FREUND, J.; RÜCKER, B. **Using BPMN 2.0 to Analyze, Improve and Automate Process in Your Company**. Colorado, USA: Jamees Venis of Lakewood, 2012.

GARCIA, F. et al. **Integrated Measurement for the Evaluation and Improvement of Software Processes**. 9th European Workshop Proceedings. Helsinki, Finland: Springer Berlin Heidelberg. 2003. p. 94-111.

GARCÍA, F. et al. FMESP: Framework for the modeling and evaluation of software processes. **Journal of Systems Architecture**, Newport Beach, CA, USA, November 2005. 627-639.

GARCÍA, F.; RUIZ, F.; PIATTINI, M. Definition and empirical validation of metrics for software process models. **Lecture Notes in Computer Science**, p. 146-158, 2004.

GARCÍA, F.; RUIZ, F.; PIATTINI, M. An experimental replica to validate a set of metrics for software process models. **Lecture Notes in Computer Science**, v. 3281, p. 79-90, 2004b. ISSN 03029743.

GENERO, M.; POELS, G.; PIATTINI, M. Defining and validating metrics for assessing the understandability of entity–relationship diagrams. **Data & Knowledge Engineering**, v. 64, n. 3, p. 534-557, March 2008. ISSN 0169-023X.

GIBSON, L. D.; GOLDENSON, D. R.; KOST, K. **Performance results of CMMI - Based process improvement**. Pitsburgh, PA. 2006.

GONZALES-PEREZ, C.; HENDERSON-SELLERS, B. Modelling software development methodologies: a conceptual foundation. **The journal of Systems and Software**, 2007.

GRUHN, V.; LAUE, R. What business process modelers can learn from programmers. **Since of Computer Programming**, 2007. 4-13.

GUCEGLIOGLU, A. S.; DEMIRORS, O. **Using software quality characteristics to measure business process quality**. Internaional Conference on Business Process Management. Nacy, France: [s.n.]. 2005. p. 374-379.

GUERRA, E. et al. **Melhoria de processos no desenvolvimento de software e hardware**. Simpósio Brasileiro de Qualidade de Software. Villa Velha, Brasil: [s.n.]. 2006. p. 326-333.

GUIZZARDI, G.; FALBO, R. A.; GUIZZARDI, R. S. S. The role of foundational ontologies for domain ontology engineering: A case study in the software process domain. **IEEE Latin America Transactions**, v. 6, n. 3, p. 244-251, 2008. ISSN 15480992.

HANI, S. U. Impact of process improvement on software development predictions, for measuring software development project's performance benefits. **ACM - Proceedings of the 7th International Conference on Frontiers of Information Technology**, 2009. 54:1 - 54:5.

HENDERSON-SELLERS, B.; GONZALEZ-PEREZ, C. A comparison of four process metamodels and the creation of a new generic standard. **Information and Software Technology**, 19 jul. 2004. 49-65.

HERBSLEB, J. et al. Software Quality and the Capability Maturity Model. **Communications of the ACM** , v. 40, n. 6, p. 30-40, June 1997. ISSN 00010782.

HJALMARSSON, A.; LIND, M. **Managing the dynamic agenda in process modelling seminars:** enhancing communication quality in process modelling. Proceedings ALOIS. Linkoping, Sweden: [s.n.]. 2004.

HUMPHREY, W. S. **Managing the Software Process**. [S.l.]: Addison-Wesley, 1989.

HUMPHREY, W. S.; SNYDER, T. R.; WILIS, R. R. Software Process Improvement at Hughes Aircraft. **IEEE Software**, v. 8, n. 4, p. 11-23, July 1991. ISSN 07407459.

ISO/IEC. **ISO 9.126 - Engenharia de software - Qualidade de produto. Parte 1: Modelo de Qualidade**. ABNT - Associação Brasileira de Normas Técnicas. Rio de Jaqneiro, p. 1-21. 2003.

ISO/IEC. **ISO 25.020 - Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Measurement reference model and guide**. International Organization for Standardization. Geneva, Switzerland, p. 15. 2007. (ISO/IEC JTC 1/SC 7).

ISO/IEC. **ISO 15.939 - Engenharia de sistemas e de software - Processo de medição**. ABNT - Associação Brasileira de Normas Técnicas. São Paulo, p. 38. 2009. (35.080).

ISO/IEC. **ISO 25.010 - Systemas and software egnineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models**. ISO. [S.l.], p. 1-34. 2011. (ISO/IEC 25010:2011(E)).

ISO/IEC. **ISO 25.040 - Systems and software engineering: System and software Quality Requirements and Evaluation (SQuaRE) - Evaluation process**. International Organization for Standardization. [S.l.], p. 1-45. 2011. (35.080).

ISO/IEC. **ISO 25.021 - Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- Quality measure elements.** International Organization for Standardization. Geneva, Switzerland, p. 37. 2012.

ISO/IEC. **ISO 25.000 - Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE**. ISO. Geneva, Switzerland, p. 1-41. 2014.

ISO/IEC. **ISO 25.023 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality**. ISO/IEC JTC 1/SC 7 Software and systems engineering. [S.l.], p. 45. 2016.

IVERSEN, J. H.; MATHIASSEN, L.; NIELSEN, P. A. Managing risk in software process improvement: AN action research approach. **MIS Quarterly: Management Information Systems**, v. 28, n. 3, p. 395-434, September 2004. ISSN 02767783.

JALOTE, P.; SAXENA, A. Optimum control limits for employing statistical process control in software process. **IEEE Transactions on Software Engineering**, v. 28, n. 12, p. 1126-1134, 12 December 2002. ISSN 00985589.

JENNINGS, N. R. On agent-based software engineering. **Artificial Intelligence**, v. 117, n. 2, p. 277-296, 2000.

JIANG, J. J. et al. An exploration of the relationship between software development process maturity and project performance. **Information & Management**, v. 41, n. 3, p. 279-288, Janeiro 2004.

KAHLOUN, F.; CHANNOUCHI, S. A. **Quality Criteria and Metrics for Business Process Models in Higher Education Domain:** Case of a Tracking of Curriculum Offers Process. Procedia Computer Science. Porto City, Portugal: [s.n.]. 2016. p. 1016-1023.

KALINOWSKI, M. et al. **Results of 10 years of software process improvement in Brazil based on the MPS-SW model**. 9th International Conference on the Quality of Information and Communications Technology. Campinhas: Institute of Electrical and Electronics Engineers Inc. 2014.

KHOSHGOFTAAR, T. M. Using process history to predict software quality. **Computer**, v. 31, n. 4, p. 66-72, August 1998. ISSN 0018-9162.

KITCHENHAM, B.; BRERETON, P. A systematic review of systematic review process research in software engineering. **Information and Software Technology**, v. 55, n. 12, p. 2049-2075, December 2013. ISSN 09505849.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. School of Computer Science and Mathematics, University of Durham. Durham, UK, p. 1-65. 2007.

KROGSTIE, J.; LINDLAND, O. I.; SINDRE, G. Towards a deeper understanding of quality in requirements engineering. **Lecture Notes in Computer Science**, v. 932, p. 82-95, 1995. ISSN 03029743.

KROGSTIE, J.; SINDER, G.; JORGENSEN, H. Process models representing knowledge for action: a revised quality framework. **European Journal of Information Systems**, v. 15, n. 1, p. 91-102, February 2006. ISSN 0960085X.

KUNG, C. An Analysis of Three Conceptual Models with Time Perspective. **Information Systems Design Mehtodologies**, Amesterdam, 1983. 141-168.

LAKATOS, E. M.; MARCONI, M. A. **Fundamentos de Metodologia Científica**. Rio de Janeiro: Atlas, 2010.

LEHMAN, M. M.; PERRY, D. E.; RAMIL, J. F. **Implications of evolution metrics on software maintenance**. Conference on Software Maintenance. Betheseda, United States: [s.n.]. 1998. p. 208-217.

LINDLAND, O. I.; SINDRE, G.; SOLVBERG, A. Understanding Quality in Conceptual Modeling. **IEEE Software**, v. 11, n. 2, p. 42-49, March 1994. ISSN 07407459.

LIOU, J. C. On improving CMMI in an immature world of software development. **Journal of Information Science and Engineering**, v. 27, n. 1, p. 213-226, January 2011. ISSN 10162364.

MAES, A.; POELS, G. Evaluating quality of conceptual modelling scripts based on user perceptions. **Data & Knowledge Engineering**, v. 63, n. 3, p. 701-724, December 2007. ISSN 0169023X.

MAGDALENO, A. M. et al. Collaboration optimization in software process composition. **Journal of Systems and Software**, v. 103, n. 1, p. 452-466, May 2015. ISSN 01641212.

MAGDALENO, A. M.; WERNER, C. M. L.; ARAUJO, R. M. Reconciling software development models: a quasi-systematic review. **The Journal of Systems and Software**, 2011.

MCCABE, W. J. **Business process management**. Conference Record - International Conference on Communications. Seattle, USA: [s.n.]. 1987. p. 535-541.

MCCALL, J. A.; RICHARDS, P. K.; WALTERS, G. F. **Factors in Software Quality. Volume I. Concepts and Definitions of Software Quality.** GENERAL ELECTRIC CO SUNNYVALE CA. [S.l.], p. 1-42. 1977.

MENDLING, J.; REIJERS, H. A.; VAN DER AALST, W. M. P. Seven process modeling guidelines (7PMG). **Information and Software Technology**, p. 127-136, 2010.

MEYER, S. et al. **Towards modeling real-world aware business processes**. WoT '11 Proceedings of the Second International Workshop on Web of Things. New York, USA: [s.n.]. 2011.

MILLER, G. A. The magical number seven, plus or minus two: some limits on our capacity for processing information. **Psychological Review**, v. 63, n. 2, p. 81-97, March 1956. ISSN 0033295X.

MOHAGHEGHI, P.; CONRADI, R. Quality, productivity and economic benefits of software reuse: A review of industrial studies. **Empirical Software Engineering**, v. 12, n. 5, p. 471-516, October 2007. ISSN 13823256.

MOHD, N.; AHMAD, R.; HASSAN, N. Resistence factores in the implementation of software process improvement project. **Journal of Computer Science**, 2008. 211-219.

MONTONI, M. A.; ROCHA, A. R. **Applying grounded theory to understand software process improvement implementation**. Proceedings - 7th International Conference on the Quality of Information and Communications Technology. Porto, Portugal: [s.n.]. 2010. p. 25-34.

MONTONI, M. A.; ROCHA, A. R.; WEBER, K. C. MPS.BR: A successful program for software process improvement in Brazil. **Software Process Improvement and Practice**, v. 14, n. 5, p. 289-300, September 2009.

MOODY, D. The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. **IEEE Transactions on Software Engineering**, v. 35, n. 6, p. 576-779, 2009. ISSN 00985589.

MOODY, D. L. Theoretical and practical issues in evaluating the quality of conceptual models: Current state and future directions. **Data and Knowledge Engineering**, v. 55, n. 3, p. 243-276, December 2005. ISSN 0169023X.

MURATA, T. **Petri nets:** properties, analysis and applications. Proceedings of the IEEE 77 (4) , pp. 541-580. [S.l.]: [s.n.]. 1989.

NATO. **Software Engineering**. NATO Science Committee. Brussels, Belgium, p. 228. 1968.

NELSON, H. J. et al. Quality in conceptual modeling: Five examples of the state of the art. **Source of the DocumentData and Knowledge Engineering**, v. 55, n. 3, p. 237-242, December 2005. ISSN 0169023X.

NELSON, H. J. et al. A conceptual modeling quality framework. **Software Quality Journal**, v. 20, n. 1, p. 201-228, March 2012. ISSN 09639314.

NIAZI, M. A comparative study of software process improvement implementation success factors. **Journal of Software: Evolution and Process**, v. 27, n. 9, p. 700-722, September 2015. ISSN 20477481.

NIAZI, M.; BABAR, M. A.; VERNER, J. M. Software Process Improvement barriers: A cross-cultural comparison. **Information and Software Technology** , v. 52, n. 11, p. 1204-1216, November 2010. ISSN 09505849.

NIAZI, M.; WILSON, D.; ZOWGHI, D. A maturity model for the implementation of software process improvement: An empirical study. **Journal of Systems and Software**, v. 74, n. 2, p. 155-172, January 2005. ISSN 01641212.

OMG. **BPMN - Business Process Model and Notation, v 2.0.1**. OBJECT MANAGEMENT GROUP. [S.l.], p. 532. 2013. (formal/2013-09-02).

OMG. Diagram Definition, Version 1.0, 2017. Disponivel em: <http://www.omg.org/spec/DD/1.0/>. Acesso em: 19 fev. 2017.

OMG Meta-Object Facility (MOF) Core Specification, Version 2.4.1. **Object Management Group**. Disponivel em: <http://www.omg.org/spec/MOF/2.4.1/PDF>. Acesso em: 19 fev. 2017.

OSTERWEIL, L. **Software processes are software too**. Proceedings - International Conference on Software Engineering. Monterey, USA: [s.n.]. 1987. p. 2-13.

PITTERMAN, B. Telcordia technologies: the journey to high maturity. **IEEE Software**, abr. 2000. 89-96.

PRAHALAD, C. K.; KRISHNAN, M. S. **The new age of innovation**. [S.l.]: McGraw Hill, 2008.

RAINER, A.; HALL, T. A quantitative and qualitative analysis of factors affecting software processes. **Journal of Systems and Software**, v. 66, n. 1, p. 7-21, April 2003. ISSN 01641212.

RAVANELLO, A. et al. **Associating Performance Measures with Perceived End User Performance:** ISO 25023 compliant Low Level Derived Measures. he Seventh International Conference on Adaptive and Self-Adaptive Systems and Applications. Nice, France: [s.n.]. 2015. p. 110-115.

RAVICHANDRAM, T.; RAI, A. Quality management in systems development: An organizational system perspective. **MIS Quarterly: Management Information Systems** , v. 24, n. 3, p. 381-410, 2000. ISSN 02767783.

RÓLON, E. et al. Applying Software Metrics to elavate Business Process Models. **CLEI Eletronic Journal**, v. 9, n. 1, June 2006. ISSN ISSN 0717- 5000.

ROLÓN, E. et al. **Prediction models for BPMN usability and maintainability**. IEEE Conference on Commerce and Enterprise Computing. Vienna, Austria: [s.n.]. 2009. p. 383-390.

ROMAN, G. A Taxonomy of Current Issues in Requirements Engineering. **Computer**, p. 14-22, April 1985.

SAHA, D.; MUKHERJEE, A. Pervasive computing: A paradigm for the 21st centrury. **Computer**, v. 36, n. 3, p. 25-31, March 2003. ISSN 0018-9162.

SALAS, J. et al. WS-replication: A framework for highly available web services. **Proceedings of the 15th International Conference on World Wide Web**. pp. 357-366.

SÁNCHEZ-GONZÁLEZ, L. et al. Toward a quality framework for business process models. **International Journal of Cooperative Information Systems**, v. 22, n. 1, p. 1-35, March 2013. ISSN 02188430.

SANTOS JR, P. S.; ALMEIDA, J. P. A.; GUIZZARDI, G. An ontology-based semantic foundation for ARIS EPCs. **ACM Symposium on applied computing**, 2010.

SERRANO, A.; WOLFF, J. F. S. Representações mentais: uma abordagem cognitivista. **Textura**, Canoas, v. 2, n. 3, p. 115-121, 2000. ISSN 2358-0801.

SILLITTI, A. et al. Collecting, integrating and analyzing software metrics and personal software process data. **Euromicro Conference, 2003. Proceedings. 29th**, 1-6 September 2003. 336-342.

SINGH, B.; KANNOJIA, S. P. **A review on software quality models**. Proceedings - 2013 International Conference on Communication Systems and Network Technologies. Gwalior, India: [s.n.]. 2013. p. 801-806.

SOFTEX. **Guia Geral MPS de Software**. Sociedade SOFTEX. [S.l.], p. 57. 2016. (ISBN 978-85-99334-84-3).

SOUNG-HIE, K.; KI-JIN, J. Desiging performance analysis and IDEF0 for enterprise modelling BPR. **Elsevier International Journalof Production Economics**, 2000.

STAPLES, M. et al. An exploratory study of why organizations do not adopt CMMI. **Journal of Systems and Software** , v. 80, n. 6, p. 883-895, June 2007. ISSN 01641212.

ST-LOUIS, D.; WITOLD, S. **Enhancing ISO/IEC 25021 quality measure elements for wider application within ISO 25000 series**. IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society. Montreal, Canada: IEEE. 2012. p. 3120-3125.

SURRY, D. W.; BAKER, F. W. The co-dependent relationship of technology and communities. **British Journal of Educational Technology**, v. 47, n. 1, p. 13-28, January 2016. ISSN 00071013.

SWELLER, J. Instructional design consequences of an analogy between evolution by natural selection and human cognitive architecture. **Instructional Science**, v. 32, n. 1-2, p. 9-31, January 2004. ISSN 00204277.

SWENSON, K. D.; SHAPIRO, R. M. **BPM in Pratice**. [S.l.]: Keith Swenson & Robert Shapiro, 2008.

TAYLOR, S. **Extreme terseness:** Some languages are more agile than others. 4th International Conference on Extreme Programming and Agile Processes in Software Engineering. Genova, Italy: [s.n.]. 2003. p. 334-336.

TCU. **Levantamento de governança de TI 2010**. Brasília, Brasil. 2010.

TCU. **Levantamento de Governança de TI 2012**. Brasília, Brasil. 2012.

UNTERKALMSTEINER, M. et al. Evaluation and measurement of software process improvement-A systematic literature review. **IEEE Transactions on Software Engineering**, v. 38, n. 2, p. 398-424, 2012. ISSN 00985589.

VAN DER AALST, W. M. P. Business process management: a comprehensive survey. **IRSN Software Engineering**, v. 2013, p. 1-37, 2013.

VAN DER AALST, W. M. P.; TER HOFSTEDE, A. H. M. YAWL: yet another workflow language. **Sience Direct**, v. 30, p. 245-275, 2005.

WAND, Y.; WEBER, R. Research comentary: Information systems and comnceptual modeling - A research agenda. **Information Systems Research**, v. 13, n. 4, p. 363-376, December 2002. ISSN 1526-5536.

WAZLAWICK, R. S. **Metodologia de Pesquisa para Ciência da Computação**. 1. ed. [S.l.]: Elsevier, 2014.

WEBER, K. C. et al. **Brazilian software process reference model and assessment method**. 20th International Symposium on Computer and Information Sciences, ISCIS 2005. Istanbul, Turkey: [s.n.]. 2005.

WEBSTER, J.; WATSON, R. T. Analyzing the Past to Prepare for the Future: Writing a Literature Review. **MIS Q.**, June 2002. xiii--xxiii.

WHITE, S. Introduction do BPMN. **BPTrends**, 2004.

WIL M. P., V. D. A. **Process mining - discovery, conformance and enhancement of business processes**. [S.l.]: Springer, 2011.

YAMAMURA, G. Sotware process satisfied employees. **IEEE Software**, set. 1999. 83-85.

YEH, R. T. et al. Software Requirements: New Directions and Perspectives. **Handbook of Sofware Engineering**, p. 519-543, 1984.

YU, E. S. K.; MYLOPOULOS, J. **Understanding 'why' in software process modelling, analysis, and design**. Proceedings - International Conference on Software Engineering. Sorrento, Italy: Elsevier Science B.V., Amsterdam. 1994. p. 159-168.

# Anexo I – A notação BPMN

A BPMN (*Business Process Modeling and Notation*) é uma notação aberta, cujo objetivo é ser de fácil compreensão por todos os profissionais da área de negócios, desde o analista de negócios, passando pelos desenvolvedores técnicos que implementam as tecnologias que suportam os processos, até as pessoas que gerenciam e monitoram esses processos (OMG, 2013). A notação foi proposta pelo Business Process Management Initiative (BPMI) em 2002, e passou a contar com o apoio da Workflow Management Coalition (WfMC), que inclusive contribuiu com a linguagem formal para a notação, a XML Process Definition Language (XPDL) (SWENSON e SHAPIRO, 2008). Atualmente, a BPMN é mantida pela Object Management Group (OMG), uma organização sem fins lucrativos que mantém padrões da indústria de computação e que conta com o apoio de fabricantes de software, usuários finais, agências de governo e da academia (OMG, 2013), e é considerada o estado da arte em termos de notação para a modelagem de processos (CHINOSI e TROMBETTA, 2011). A objetividade dessa notação, bem como a clareza e facilidade de uso (CAMPOS e OLIVERIA, 2011) são, provavelmente, as características que tornaram a notação BPMN a mais amplamente utilizada para a modelagem de processos de trabalho (WIL M. P., 2011).

A notação BPMN define um diagrama de processos que é baseado em gráficos de fluxo (*flowchart*) para a descrição de operações de trabalho. Trata-se de uma rede de representações gráficas composta basicamente por atividades (trabalho) e setas (fluxo). O objetivo da notação é ser facilmente compreendida, mas também capaz de gerar informações para ambientes de execução de processos, por exemplo, através de *Business Process Execution Languate* (BPEL). De fato, faz parte da própria concepção do modelo e estruturação de uma ponte entre os processos de trabalho e a execução desses processos (WHITE, 2004).

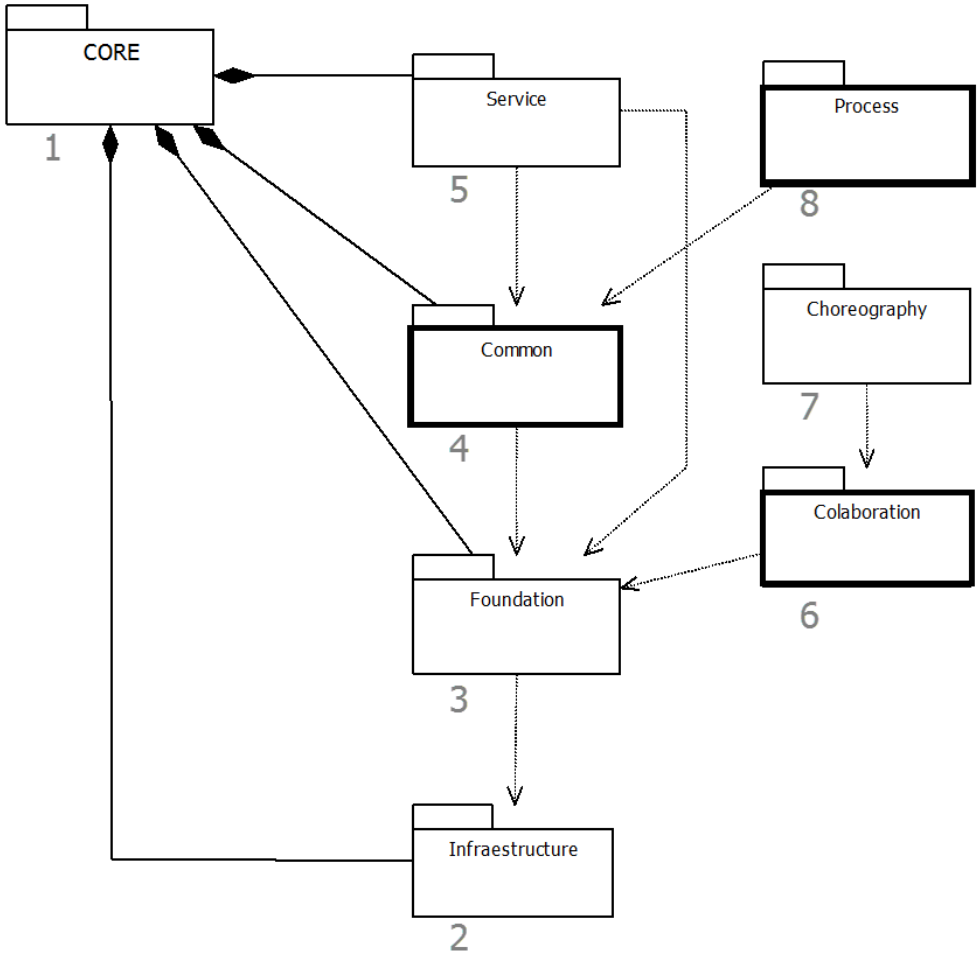



*Figura 46. Estrutura do metamodelo BPMN 2.0, elaborado a partir de (OMG, 2013).*

O metamodelo do BPMN é estruturado em 8 pacotes, conforme apresentado na Figura 46. O pacote "Core" é composto de outros quatro pacotes: "Infraestructure", "Foundation", "Common" e "Service". Além desses, há o pacote "Process" que estende o "Common", o "Colaboration" que estende o "Foundation" e o "Choreography" que estende o "Colaboration". Esses pacotes são definidos da seguinte maneira:

1. **Core**. Contém os elementos mais fundamentais do BPMN, e tem o objetivo de ser simples, conciso, extensível e com comportamentos bem definidos.
2. **Infraestructure**. Contém duas classes que são usados tanto por classes abstratas quanto por modelos de diagrama BPMN. Essas duas classes são "Definitions", que é a classe mais ampla e que contém todos os elementos BPMN, como se fosse um pacote de elementos BPMN, e a "Import", que referencia documentos externos ao modelo, que podem ser tanto documentos BPMN quanto documentos de outros modelos.

3. **Foundation**. Esse pacote contém classes que são compartilhadas entre os demais pacotes contidos em “Core”, sendo responsável, entre outas coisas, pela extensibilidade do BPMN, através da classe “Extension”, e pela classe “ExternalRelationship”, que viabiliza a construção de relacionamentos tipados entre elementos estendidos do BPMN com elementos externos ao BPMN, como elementos UML, por exemplo.
4. **Common**. O pacote “Common” contém as classes que são utilizadas nos diversos tipos de diagrama, ou seja, “Process”, “Collaboration” e “Choreography”.
5. **Service**. O pacote “Services” contém os elementos necessários para a modelagem de serviços, interfaces e operações.
6. **Collaboration**. Esse pacote contém o conjunto de classes necessárias para a modelagem de Colaborações, ou seja, de um conjunto de participantes representados por Pools, suas interações representadas por Fluxos de Mensagem (Message Flows) e pode incluir Processos com Pools e/ou Coreografias (Choreographies) entre esses Pools.
7. **Choreography**. Esse pacote é uma extensão do pacote “Collaboration”, e também é um tipo de Processo. Contudo, nesse caso, o objetivo não é o de orquestração, ou seja, de saber como o trabalho acontece, mas sim o de saber como os participantes do processo coordenam suas interações. Assim, o foco é em como as informações são trocadas entre esses participantes.
8. **Process**. Esse é, sem dúvida, o maior pacote de todo o BPMN. Ele contém as classes necessárias para a modelagem dos processos, descrevendo uma sequência de atividades em diversos níveis de abstração.

Embora todos os pacotes do BPMN tenha sua relevância e papel determinado no ciclo de vida de gestão de processos, para os objetivos do presente trabalho o foco estará nos pacotes “Common”, que defini um importante conjunto de elementos para a modelagem de processos, no “Process” que possui todos os demais elementos necessários à modelagem de processos, e no “Collaboration”, que pode ser utilizado em conexão com a modelagem de processos para identificar a troca de mensagens entre os participantes do processo.

## 8.4 Pacote “Common”

O pacote “Common” contém um conjunto de superclasses que são utilizadas por classes dos demais pacotes.

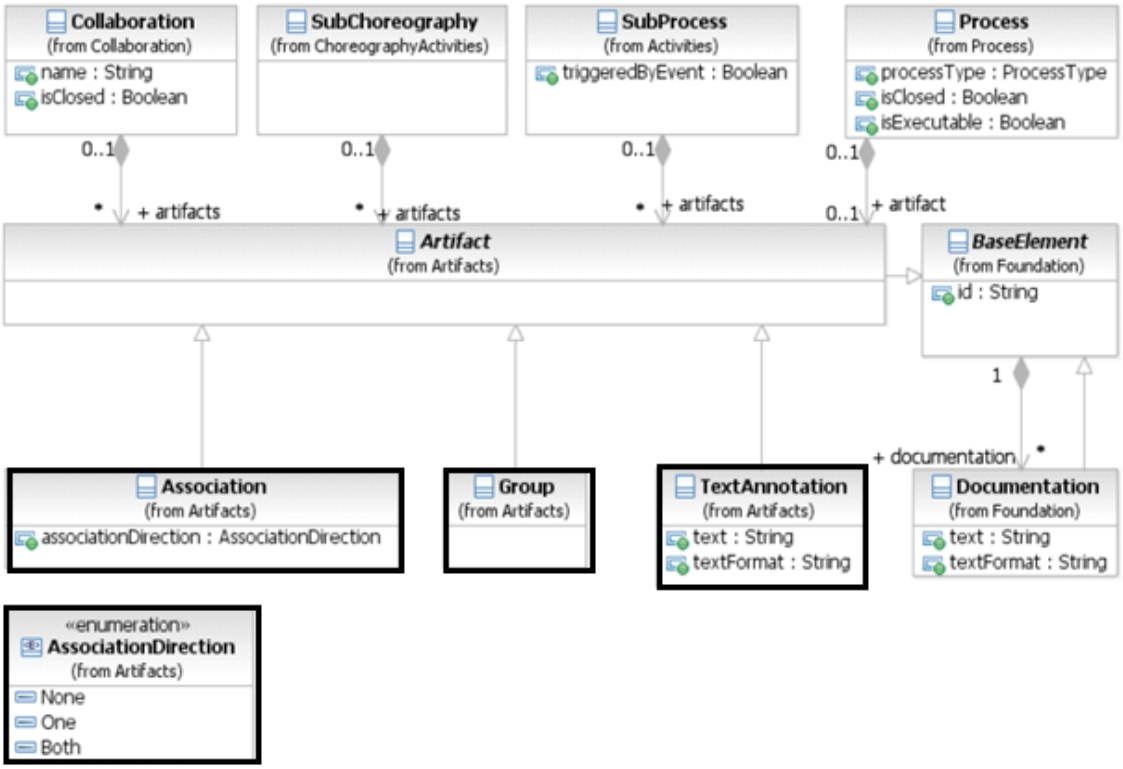
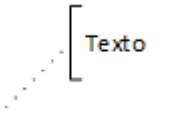



*Figura 47. Diagrama de classe de "Artifacts", conforme (OMG, 2013)*

A classe “Artifacts” permite adicionar informação adicional sobre o processo modelado, não havendo relação com o fluxo do processo ou com a troca de mensagens desse processo. Conforme pode ser visto na Figura 47, são três os artefatos providos por essa classe: 1) “Associations”; 2) “Groups”; 3) “Text Annotations”. Esses elementos estão descritos na Figura 47.

*Tabela 63. Elementos BPMN contidos da classe “Artifacts”, no pacote “Common”.*

| Elemento BPMN | Descrição |
|---|---|
| . . . . . | Uma associação (especialização “Association”) possibilita a ligação de informações e especializações de “Artifacts” com o objetos de fluxo (flow objects), tais como atividades, gateways e eventos. |
| . . . . . › | Na Figura 47 pode ser observado atributo “associationDirection” da classe “Association”, que utiliza a enumeração “AssociationDirection”, com os valores *None*, *One*, e *Both*. Assim, como pode ser visto nessa linha da coluna, é possível indicar visualmente a direção da associação. |

| | |
|---|---|
|  | A classe “TextAnnotation” é uma especialização de “Artifacts” cujo objetivo é possibilitar a adição de textos explicativos que sejam visualmente observáveis no modelo do processo, ou seja, quando a descrição nos próprios objetos (que não aparecem visualmente no modelo) são insuficientes para proporcionar a devida clareza ao modelo. |
|  | A classe “Group”, por sua vez, tem a função de prover um mecanismo visual para agrupar elementos de um diagrama, de maneira informal. Isso significa que o principal objetivo desse elemento é um modelo de processo é facilitar a compreensão humana do mesmo. Vale destacar que esse elemento não fica restrito por “Pools” e “Lanes” (que ainda serão abordados nesse trabalho). |

A classe “SequenceFlow” é usada para ordenar o fluxo dos elementos de processo (“FlowNode”) em um modelo de processo. Como pode ser visto na Figura 48, cada “SequenceFlow” tem uma única origem e um único destino, que podem se ligar aos elementos do tipo atividade, evento e gateway.

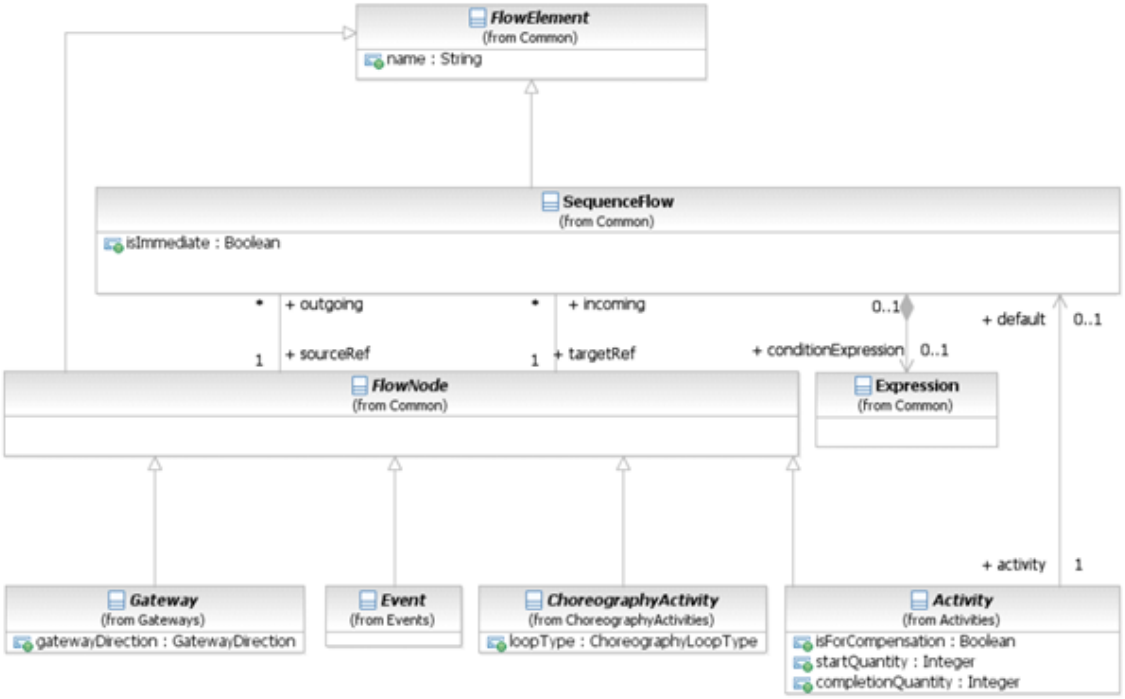



*Figura 48. Diagrama de classes do "SequenceFlow", do pacote "Foundation", conforme (OMG, 2013).*

A representação visual da classe “SequenceFlow” pode ser vista na Tabela 64.

*Tabela 64. Elementos BPMN contidos da classe “SequenceFlow”, no pacote “Common”.*

| Elemento BPMN | Descrição |
|---|---|
|  | Um fluxo sequencial (“SequenceFlow”) possibilita definir o caminho percorrido dentro de um processo, indicando o próximo “FlowNode” (atividade, evento, gateway) a ser executado. Essa classe possui o atributo “conditionExpression”, do tipo “Expression”, que pode ser opcionalmente utilizado para estabelecer uma expressão a ser avaliada, cujo resultado indicará se o fluxo do processo seguirá por |

| | |
|---|---|
| | esse “SequenceFlow”, se o resultado for *verdadeiro*, ou não, se o resultado for *falso*. |

Existes diversas outras classes base no pacote “Common” das quais são herdadas as classes dos pacotes “Collaboration” e “Process”, tais como

- Correlation;
- Error;
- Escalation;
- Events;
- Expressions;
- Flow Element;
- Gateways;
- Item Definition;
- Message;
- Resources.

Apesar de todas essas classes serem representadas visualmente na notação BPMN, trataremos dessas representações nos pacotes que herdam definitivamente essas classes, ou seja, em “Collaboration” e “Process”. Contudo, há duas classes que não serão plenamente cobertas nesses pacotes, a “Expressions”.

A classe “Expressions” é usada para especificar uma expressão usando linguagem natural, ou seja, sem base formal. É usada amplamente na notação BPMN, especialmente como critérios de decisão, ou seja, como elementos de avaliação utilizados em Gateways (ainda por serem abordados nesse trabalho). Contudo, para a definição de expressões formais há também a classe “FormalExpression”, onde uma linguagem formal de expressões pode ser indicada e utilizada.

## 8.5 Pacote Collaboration

Uma colaboração é basicamente a representação da troca de mensagens entre participantes de um processo, ou seja, entre os atores desse processo. Os participantes podem ser unidades específicas, como uma organização, um departamento ou um setor, ou ainda um papel dentro do processo, como comprador, vendedor, etc. Como pode ser observado na Figura 49, os participantes estão contidos em uma “Collaboration” e também podem estar referenciados em um “Process”. Ainda nessa figura, é possível notar

o destaque para “ParticipantAssociation”, que possibilita que um mesmo participante exerça papéis diferentes em diferentes diagramas, como por exemplo, um mesmo colaborador (ou um mesmo cargo) exerça o papel de gerente de projetos em um modelo (diagrama) e o de elicitador de requisitos em outro modelo. Assim, “ParticipantAssociation” permite mapear um participante em outro, ou seja, um papel em outro.

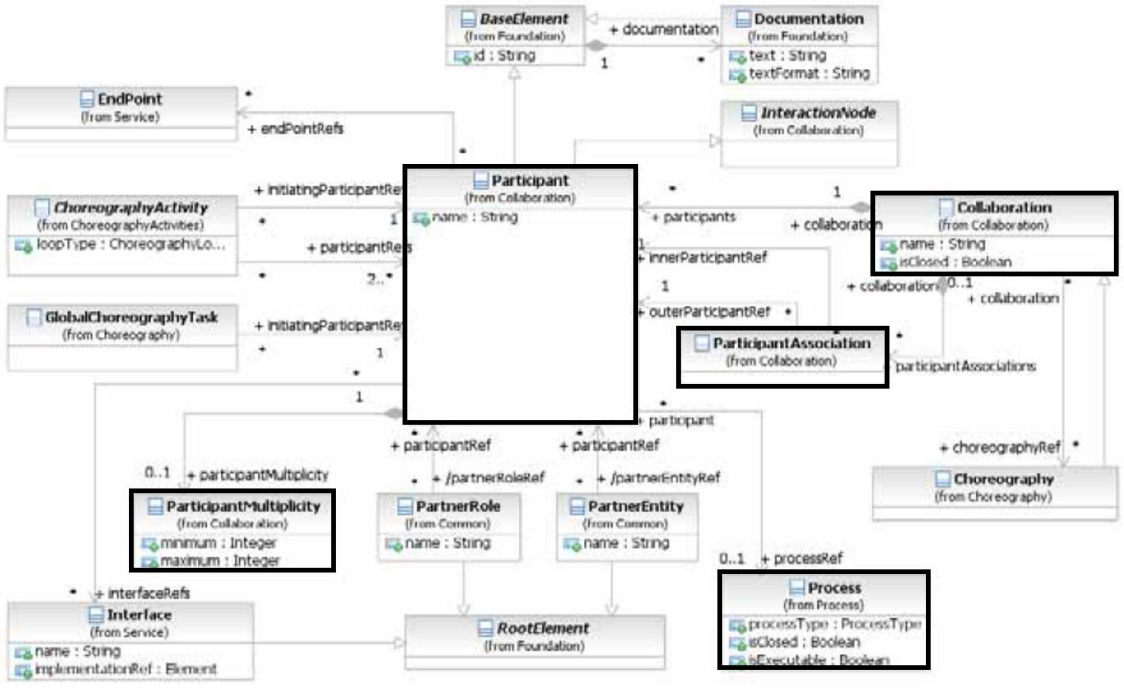



*Figura 49. Diagrama de classes de “Participant", conforme (OMG, 2013).*

É importante notar também a classe “ParticipantMultipicity”, que é uma composição de “Participant”. Essa classe possibilita a indicação da multiplicidade do participante, ou seja, de quantas instâncias do participante haverá no processo.

O outro elemento fundamental de uma colaboração é a mensagem (“MessageFlow”). A Figura 50 apresenta as classes fundamentais para a troca de mensagens entre participantes em uma colaboração. Como pode ser observado, uma colaboração poderá ter diversos participantes e diversas mensagens enviadas e recebidas. A classe “InteractionNote” é usada para prover um elemento único com origem e destino para as mensagens (“MessageFlow”), e apenas os elementos “Pool” (“Participant”), “Activity” e “Event” podem ser configurar em um “InterationNode”, ou seja, apenas esses quatro elementos podem ser ligar a uma mensagem. Esses elementos, que possuem representação visual, podem ser observados na

Tabela 65.

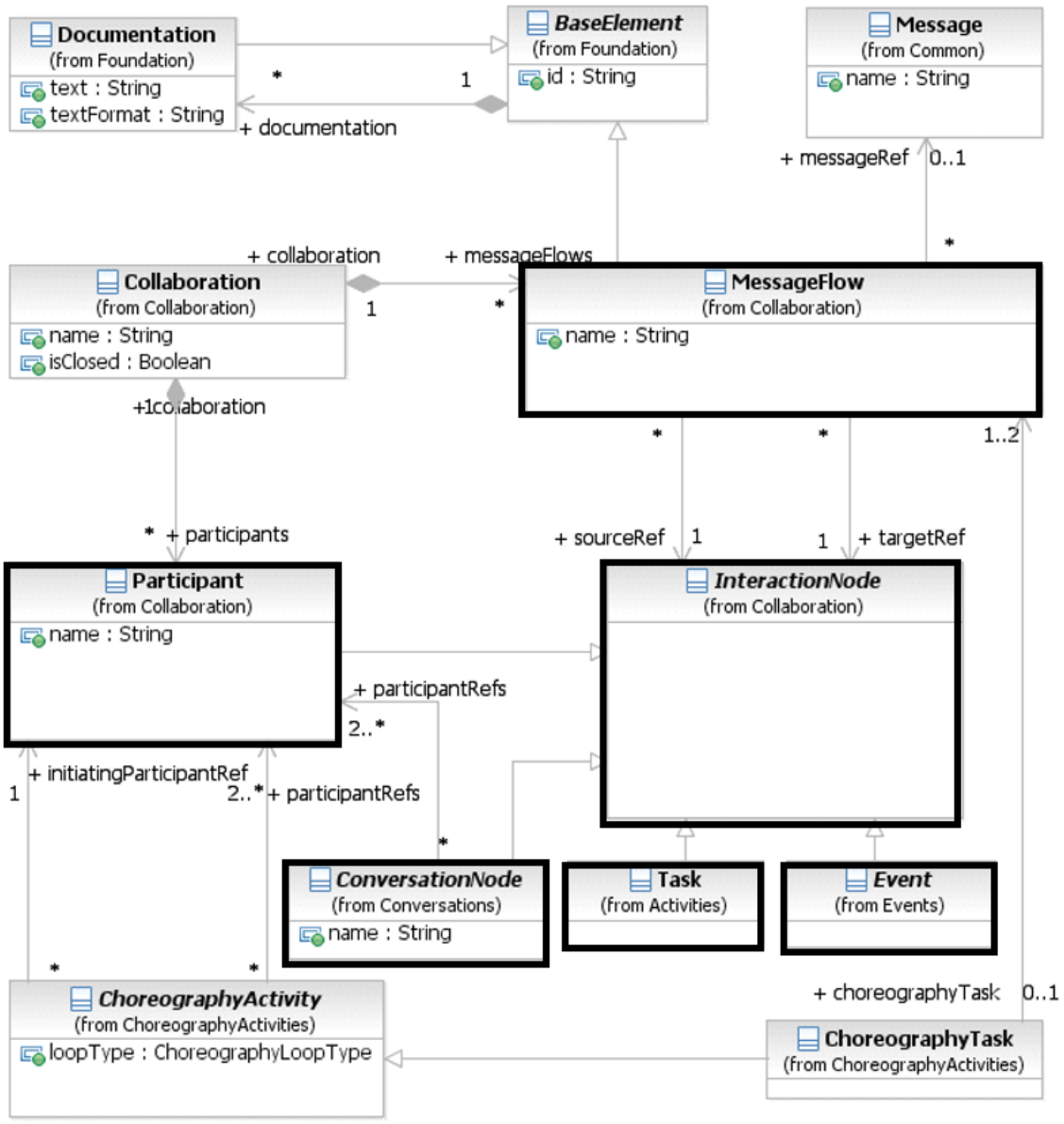
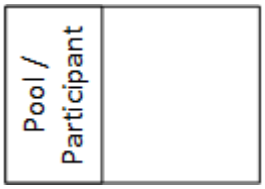



*Figura 50. Diagrama de classe de "MessageFlow, conforme (OMG, 2013) ".*

*Tabela 65. Elementos BPMN do pacote "Collaboration".*

| Elemento BPMN | Descrição |
|---|---|
|  | O elemento Pool (ou Participante), representa uma organização que participa do processo modelado, de um papel desempenhado nesse processo, tal como Comprador, Vendedor, ou ainda de uma área da organização que desempenha esse papel, tal como Financeiro, Produção, ou Expedição. |
|  | Um participante poderá aparecer como única instância em um processo ou como múltiplas instâncias nesse processo. Por exemplo, um setor de Suporte ao Usuário (instância única) poderá fornecer suporte a diversos usuários, nesse caso a múltiplas instâncias de Usuário. |

| | |
|---|---|
|  | A mensagem se refere à informação que é enviada de um participante para o outro, no sentido da seta. É importante destacar que o objetivo das mensagens é trocar informações entre diferentes participantes (Pools), e por isso não deve ser utilizada para trocar informações dentro de um mesmo Pools. Também, deve-se destacar que a troca de mensagens não representa um fluxo sequencial de processo, ou seja, uma mensagem vai de um Pool para outro, mas o fluxo sequencial do processo seguirá seu caminho, diverso daquele apontado pela seta de mensagem. |

Um processo poderá estar contido em uma colaboração, e na verdade isso é comum na modelagem de processos utilizando a notação BPMN. Apenas para ilustrar os conceitos apresentados, a Figura 51 apresenta um exemplo de troca de mensagens entre participantes, ou seja, de uma colaboração BPMN. O participante "Usuário" pode ter múltiplas instâncias, cujos chamados que são enviados em forma de mensagem são atendidos pelo processo executado por "Suporte", um participante com uma única instância. Quando o "Suporte" realiza o atendimento, envia uma mensagem de volta para a instância de "Usuário" que fez o chamado, indicando a conclusão do referido chamado.

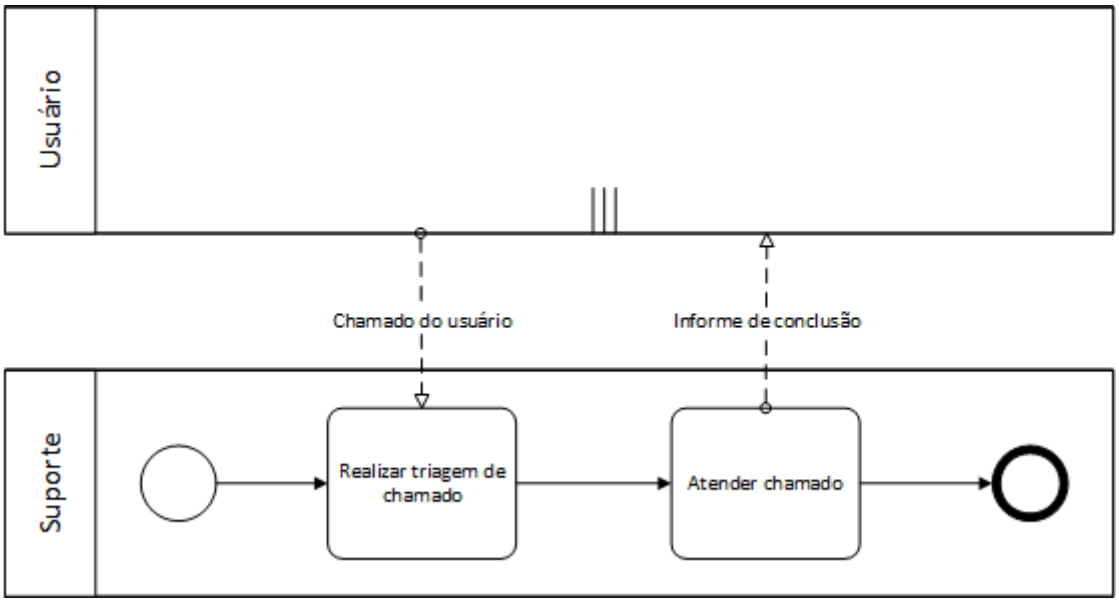



*Figura 51. Exemplo de colaboração em BPMN.*

## 8.6 Pacote "Process"

Um processo representa uma sequência de fluxos de trabalho, ou fluxos de atividade. A notação BPMN implementa o conceito de Processo através de elementos visuais que representam esse processo, ou seja, atividades ("Activities"), eventos ("Events"), junções e disjunções ("Gateways") e sequência do fluxo ("SequenceFlow"), como pode ser visto na Figura 52. A classe "Process" não possui representação visual, mas herda

“FlowElementsContainer”, que contém uma ou mais classes “FlowElement”. É importante notar que a classe “FlowNode” herda de “FlowElement”, que por sua vez pode ser herdado por “Activity”, ou “Event”, ou “Gateway”. Ainda, “SequenceFlow” pode herdar de “FlowElement”. Isso significa que os elementos “FlowElement” contidos em “FlowElementsContainer” (e consequentemente em “Process”) podem ser do tipo atividade, evento, gateway ou fluxo sequencial.

Esses quatro elementos (atividades, eventos, gateways e fluxos sequenciais) possuem representação visual e, portanto, fazem parte da notação BPMN para a modelagem de processos. Contudo, cada um desses elementos possui particularidades que precisam ser observadas em maior detalhe.

Ainda sobre a classe “Process”, é importante destacar também que “Process” herda de “CallabeElement”, o que possibilita a instância de um processo que pode ser chamado de diversos outros processos, ou seja, um processo reutilizável.

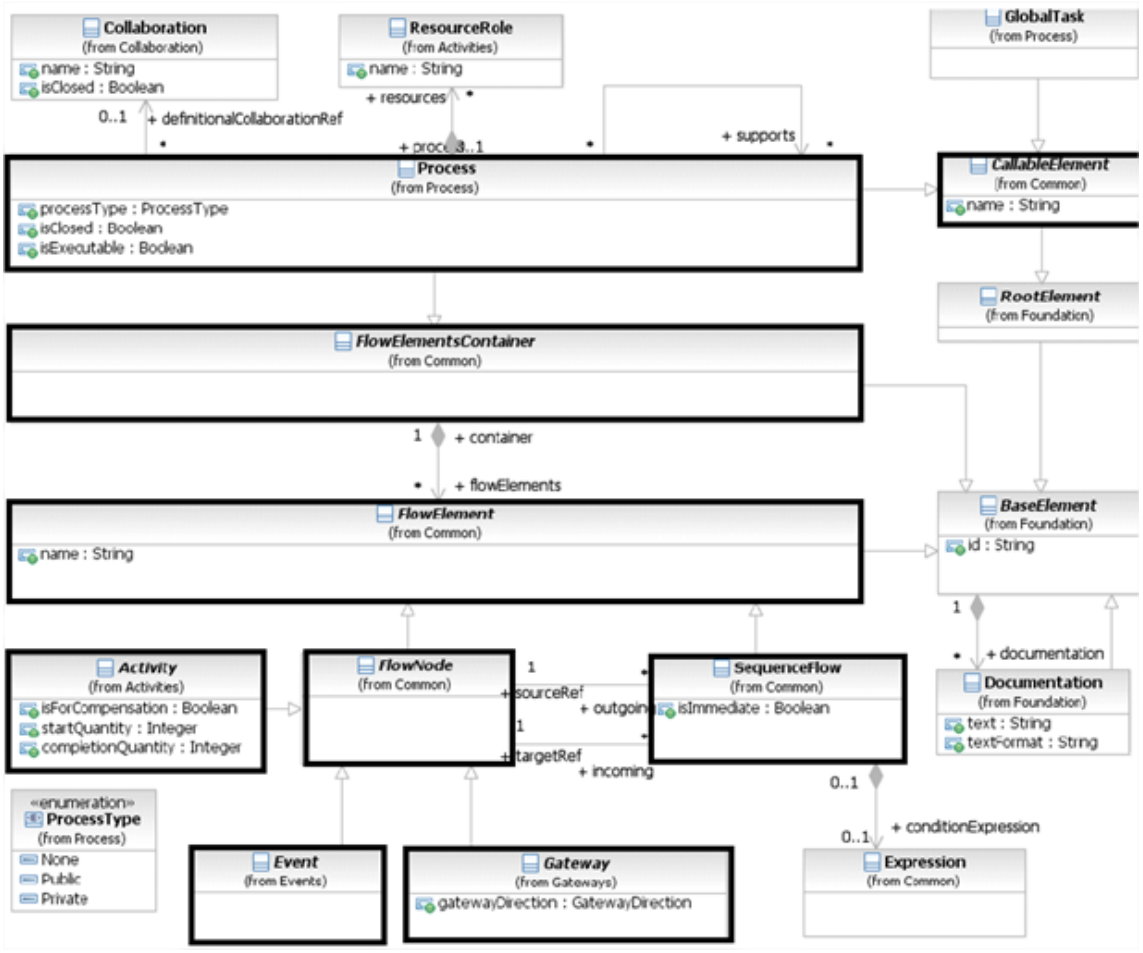


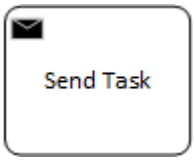
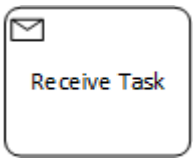

*Figura 52. Diagrama de classe do pacote "Process", conforme (OMG, 2013).*

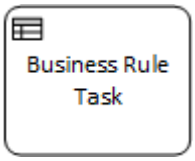
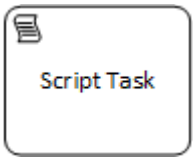
A classe “Activity” do pacote “Process” torna possível a representação do trabalho em um modelo processo, e a notação BPMN oferece uma rica estrutura de representação para esse fim. As “Activity” podem ser basicamente de 3 tipos:

1. Task
2. Sub-Process
3. Call activity

Uma atividade do tipo “Task” é aquela que representa o trabalho que não pode ser quebrado em sub-níveis, ou seja, que não pode (ou não interessa) ser detalhado ainda mais. Esse tipo de atividade é chamado também de atividade atômica. Existem diferentes tipos de “Task”, que estão descritos na Tabela 66.

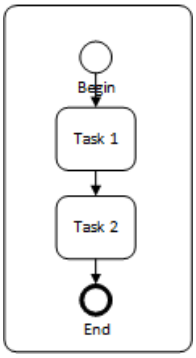
*Tabela 66. Atividades do tipo “Task”, do pacote "Process", conforme (OMG, 2013).*

| Elemento BPMN | Descrição |
|---|---|
|  | Uma atividade atômica padrão, que pode ser utilizada para representar atividades genéricas que não exijam maior detalhe de especificação quanto a seu tipo. De fato, costuma ser uma representação bastante utilizada para tarefas atômicas em modelos de processo construídos com BPMN. |
|  | Utilizada para indicar a utilização de algum tipo de serviço, como um webservice, ou de uma aplicação para a realização da tarefa. Possui o atributo “implementation”, onde será indicada a tecnologia utilizada para a execução da tarefa em questão. |
|  | Essa tarefa é utilizada para enviar uma mensagem do processo que a contém para um participante (“Pool”), de modo que tão logo a mensagem seja enviada a tarefa é concluída e o fluxo segue para o próximo “FlowNode”. A indicação do participante destinatário da mensagem é indicado pela ligação entre o “Send Task” e o “Pool/Participant” através de uma “MessageFlow”, ou seja, de uma mensagem da tarefa para a borda do Pool. |
|  | Essa tarefa interrompe a execução do processo para aguardar a chegada de uma mensagem que vem de fora do processo no qual ela está contida. A mensagem é obtida de um participante (“Pool”), que é indicado pela ligação entre o “Pool/Participant” e o “Receive Task” através de uma “MessageFlow”, ou seja, de uma mensagem da borda do Pool para a tarefa. |
|  | A tarefa do tipo “User Task” representa um trabalho realizado por humano com o suporte de um software de workflow, e essa tarefa é agendada através de algum tipo de gerenciador de lista de tarefas. |
|  | A tarefa do tipo “Manual Task” representa um trabalho puramente realizado por humano, sem qualquer suporte de máquina de execução ou aplicativo de software. |

| | |
|---|---|
|  | A “Business Rule Task” viabiliza um mecanismo para prover informações à um motor de regas de negócio (*Business Rules Engine*) e receber o retorno desse motor. Essa classe possui um atributo “implemantion”, onde é indicado o tipo de tecnologia que será utilizada para implementar a tarefa, como um webservice, por exemplo. |
|  | O “Script Task” é executado por um motor de processos, e o modelador ou implementador deve entrar com um script em linguagem que o motor possa interpretar. Após a execução do script, a tarefa é encerrada e o fluxo segue para o próximo “FlowNode”. |

Os “Sub-Process” são atividades (“Activity”) que possuem detalhamento expresso através de outros “FlowNode”, ou seja, atividades, gateways, eventos e fluxos sequenciais. Assim, o “Sub-Process” é uma representação gráfica contida por uma classe “Process” (que não possui representação gráfica), e que viabiliza construir camadas de abstração em um modelo de processo, ou seja, possibilita “explodir” uma atividade e detalhar seu funcionamento através de outras atividades. Os diferentes tipos de “Sub-Process” são apresentados na Tabela 67.

*Tabela 67. Atividades do tipo “Sub-Process”, do pacote "Process", conforme (OMG, 2013).*

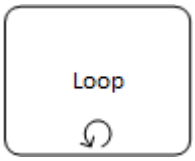
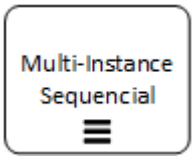
| Elemento BPMN | Descrição |
|---|---|
|  | Esse tipo de atividade indica a existência de subatividades descritas também com a notação BPMN, que estarão embutidas no processo que a contém, ou seja, terá acesso às mesmas informações do desse processo e também será realizada implicitamente pelo participante no qual a atividade está associada. |
|  | A mesma tarefa, do tipo “Embedded” pode ser descrita também em formato expandido, exibindo o detalhamento dentro de si. |
|  | A “Call Activity” não é um processo propriamente dito, mas sim uma referência ou uma chamada a um processo global (processos reutilizáveis) ou a uma atividade global (atividade reutilizável) existente. A execução desta atividade transfere o controle do processo para o processo ou atividade global. Assim como a atividade do tipo “Embedded Sub-Process”, a “Call Activity” também pode exibir seu conteúdo de maneira expandida. |

| | |
|---|---|
|  | Uma atividade do tipo “GlobalTask” é uma classe que herda da classe abstrata “CallableElement”, que por sua vez recebe uma associação da classe “CallActivity”. Desse modo, “GlobalTask” representa uma atividade atômica que pode ser chamada de qualquer processo através de uma classe “CallActivity”. |
|  | Uma atividade do tipo “Event Sub-Process” é um tipo de subprocesso especial, que não é ativado por um fluxo sequencial (“SequenceFlow”). O gatilho para ativação de uma atividade desse tipo é, na verdade, um evento dentro do próprio subprocesso, que pode ser uma mensagem, uma condição, um tempo, entre outros (os eventos serão abordados mais à frente no presente trabalho). Assim, essa atividade poderá ocorrer nenhuma, uma, ou várias vezes durante a execução do processo regular, dependendo de quantas vezes ocorra o evento que a dispara ou ativa. O “Event Sub-Process” também pode ser exibido no modo expandido. |
|  | A atividade do tipo “Transaction” é na verdade uma classe especializada de “Sub-Process”, que implementa um comportamento diferenciado através de um protocolo de transação que pode ser indicado, como por exemplo o WS-Transaction (SALAS, PEREZ-SORROSAL, *et al.*). Essa atividade conterá outras atividades, estabelecendo assim um escopo para as mesmas. O encerramento da “Transaction” se dará por uma das seguintes formas: 1) *Sucessful completion*, e a “Transaction” será encerrada normalmente; 2) *Failed completion*, e a transação será cancelada, de modo que as atividades dentro da “Transaction” estarão sujeitas a ações de cancelamento, incluindo *rooling back* do processo e ativação de atividades de “Compensation” para desfazer o que eventualmente já tenha sido feito no processo; 3) *Hazard*, o que indica que algo errado ocorreu ao ponto de impedir a finalização normal (sucesso) e mesmo a finalização por cancelamento, inviabilizando a compensação. Nesse caso, a atividade é interrompida sem compensação e segue o fluxo indicado por um evento intermediário do tipo “Error” (que ainda será tratado nesse trabalho). |

As atividades BPMN, tanto do tipo “Task” quanto do tipo “Sub-Process”, podem receber marcadores que alteram o comportamento dessas atividades. Alguns desses marcadores são comuns a “Task” e “Sub-Process”, enquanto que outros são específicos de “Sub-Process”. A

Tabela 68 apresenta os marcadores que podem ser utilizados em atividades, bem como a descrição do comportamento adicionado, além de indicar se são aplicáveis à “Task”, “Sub-Process” ou a ambos.

*Tabela 68. Marcadores de "Activity", do pacote "Process", conforme (OMG, 2013).*

| **Elemento BPMN** | **Descrição** | **“Task”** | **“Sub-Process”** |
|---|---|---|---|
|  | Uma atividade ou subprocesso pode ser repetido sequencialmente, como um Loop. Esse loop poderá ocorrer um determinado número de vezes, com base no atributo “loopCounter”, ou enquanto determinada “Expressions” informada pelo modelador retornar *verdadeiro*. | Sim | Sim |
|  | Esse marcador indica que a “Task” ou “Sub-Process” será instanciado um determinado número de vezes. Nesse sentido, esse marcador é similar ao marcador de *loop*. Contudo, o *multi-instance sequencial* pode utilizar a classe | Sim | Sim |

| | | | |
|---|---|---|---|
| | "Expressions" ou uma entrada de dados (*Data Association*) para determinar o número de instâncias. | | |
|  | Promove comportamento similar ao *multi-instance sequencial*, com a diferença de que as instâncias são paralelas e não sequenciais. | Sim | Sim |
|  | A atividade do tipo "Ad-Hoc" é na verdade uma classe especializada de "Sub-Process", que viabiliza o agrupamento de um conjunto de outras atividades que serão realizadas, mas que não guardam nenhuma obrigatoriedade de sequência de execução. Por isso mesmo, essas atividades dentro de "Ad-Hoc Sub-Process" não estão interligadas por fluxos sequenciais ("SequenceFlow"). Contudo, essa classe possui o atributo "completionCondition", que é do tipo "Expression" (que, como já apresentado, pode ser informal ou formal). Quando essa "Expression" retorna *verdadeiro*, o subprocesso é encerrado. Apesar de não haver ordem, a inclusão de "DataObjects" com as respectivas associações de entrada e saída poderão indicar uma sequência preferencial de maneira implícita. | Não | Sim |
|  | Como pode ser visto na Figura 52, a super classe abstrata "Activity" possui o atributo "isForCompensation", do tipo *boolean*. Quando o atributo for *falso*, a atividade segue o fluxo normal, mas quando o atributo for *verdadeiro*, a atividade só é ativada quando um evento do tipo "Compensation" a inicia (ainda trataremos sobre eventos nesse trabalho). | Não | Sim |

Outra importante classe do modelo BPMN é a classe "ItemAwareElement". Ela possibilita a utilização de elementos de dados em modelos de processo construídos com a notação BPMN. A Figura 53 apresenta o diagrama de classes dessa classe, abrangendo classes associadas e heranças a partir dela. As classes que estão associadas a elementos visuais da notação são:

- "DataObject" e "DataObjectReference";
- "DataStore" e "DataStoreReference";
- "DataInput" e "DataOutput.

O modelo BPMN possibilita, ainda, através de "ItemDefinition", a associação de uma estrutura de dados para o "ItemAwareElement" em questão, de modo que a modelagem de dados possa ser feita em nível maior de detalhamento utilizando uma linguagem específica para isso. A notação BPMN define como linguagem padrão para esse fim o XML/XPath, mas os fabricantes de ferramentas de modelagem BPMN poderão optar por outras linguagens de modelagem de dados.

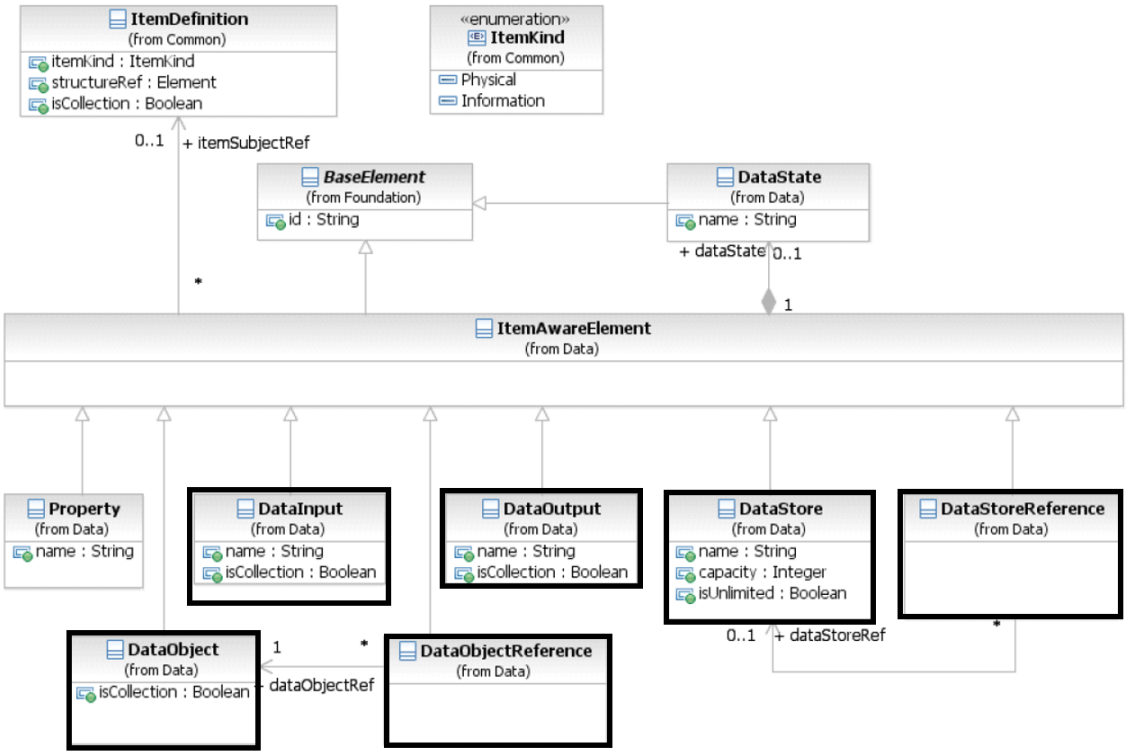
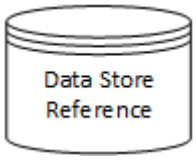



*Figura 53. Diagrama de classe de "ItemAware", do pacote "Process", conforme (OMG, 2013).*

. Os “ItemAwareElement” poderão estar ligados a atividades (“Activity” de “Process”), fluxos sequenciais (“SequenceFlow” de “Common”) ou mensagens (“MessageFlow” de “Collaboration”), através de uma associação (“Association” de “Common”). Essa associação terá o atributo “AssociationDirection” indicando a direção para a qual o dado está sendo transferido.

*Tabela 69. Classes herdadas de “ItemAwareElement", do pacote "Process", conforme (OMG, 2013).*

| Elemento BPMN | Descrição |
|---|---|
|  | A classe “DataObject” possibilita a indicação de dados produzidos e consumidos ao longo do processo modelado em BPMN. Seu funcionamento é similar ao funcionamento de variáveis em linguagens de programação. Inclusive, os elementos do tipo “DataObject” existem apenas durante uma instância do processo na qual estão modelados. |
|  | Um “DataObject” pode ser definido como “Collection”, através da propriedade “isCollection”. Nesse caso, o “DataObject” se assemelharia ao conceito de Array (ou vetores) em uma linguagem de programação, podendo conter vários dados similares (mesma estrutura). |

| | |
|---|---|
|  | A utilização de “DataObject” em um processo pode implicar em um grande números de associações com os elementos de fluxo (“FlowNode”), de modo a causar poluição visual no modelo. Ademais, em muitos modelos se verifica a necessidade de alterar o estado do dado, ou seja, alterar o estado do “DataObject” ao longo do processo. Para atender a essas duas necessidades, o BPMN oferece a classe “DataObjectReference”, que possibilita fazer várias “cópias” do “DataObject” ao longo do modelo, que são na verdade referências ao mesmo “DataObject”. Ainda, no “DataObjectReference” é possível informar o estado do dado entre colchetes (opcional). |
|  | A classe “DataInput” possibilita definir um determinado dado que será usado em um subprocesso reutilizável invocado por uma “CallActivity”. Comparando a uma linguagem de programação, seria algo semelhante à definição de parâmetros de uma função ou método. |
|  | A classe “DataOutput” possibilita definir um determinado dado que será produzido em um subprocesso reutilizável invocado por uma “CallActivity”. Comparando a uma linguagem de programação, seria algo semelhante à definição de retorno de uma função ou método. |
|  | A classe “DataStore” torna possível que atividades (“Activity”) armazenem e recuperem informação armazenada de maneira persistente, ou seja, as informações armazenadas em “DataStore” permanecem disponíveis mesmo após o encerramento da instância do processo que produziu e armazenou esses dados. |
|  | De maneira similar à “DataObjectReference”, a classe “DataStoreReference” possibilita fazer “cópias” de uma “DataStore”, que na verdade são referências a essa “DataStore”. No entando, para esta classe, não é possível informar do estado do dado, uma vez que esse conceito não se aplica a dados armazenados de maneira perene. |

Outro elemento importante da notação BPMN são os eventos, que também são especializações da classe “FlowNode”. Os eventos são coisas que acontecem, ou ocorrências ao longo do processo, como por exemplo o início ou fim de uma atividade, uma mensagem que chega, ou uma mudança de estado em um documento. O metamodelo BPMN agrupa os eventos em classes que provocam ocorrências (*throw*) e classes que detectam (*catch*) uma ocorrência, como pode ser visto na Figura 54.

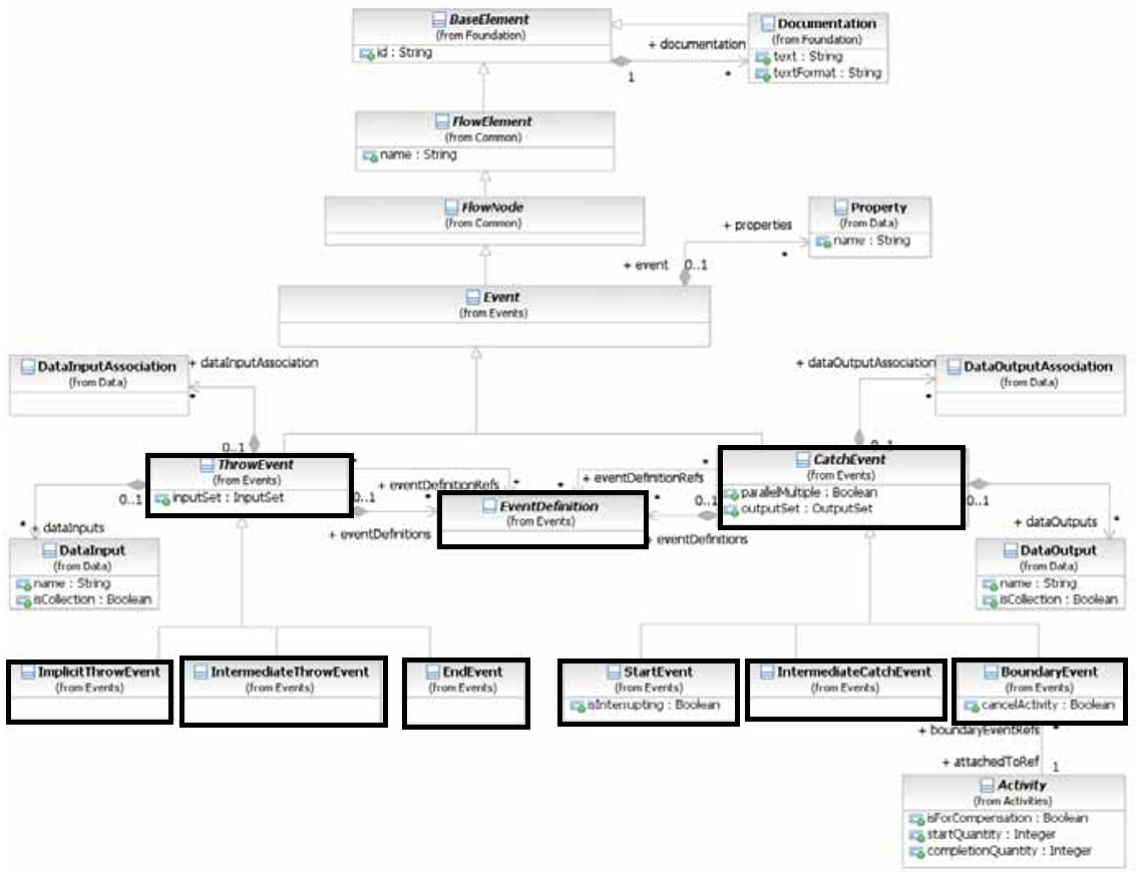



*Figura 54. Diagrama de classe de "Event", do pacote "Process", conforme (OMG, 2013).*

De acordo com o metamodelo BPMN, ainda na Figura 54, os eventos do tipo *throw* podem receber dados do processo e enviá-lo para fora dele, ao passo que os eventos do tipo *catch* podem receber dados de fora do processo e enviar esses dados para um "FlowNode" do próprio processo. Ainda, os eventos estão estão subclassificados (heranças) a partir dessas duas classes abstratas, "ThowEvent" e "CatchEvent".

Contudo, como os eventos em BPMN possibilitam modelar processos com riqueza de detalhes, foi elaborado uma forma de classificação que facilita a compreensão dos mesmos para efeito de notação, ou seja, para efeito de uso desses elementos notacionais em modelos de processo. Essa outra classificação é apresentada na Figura 55.

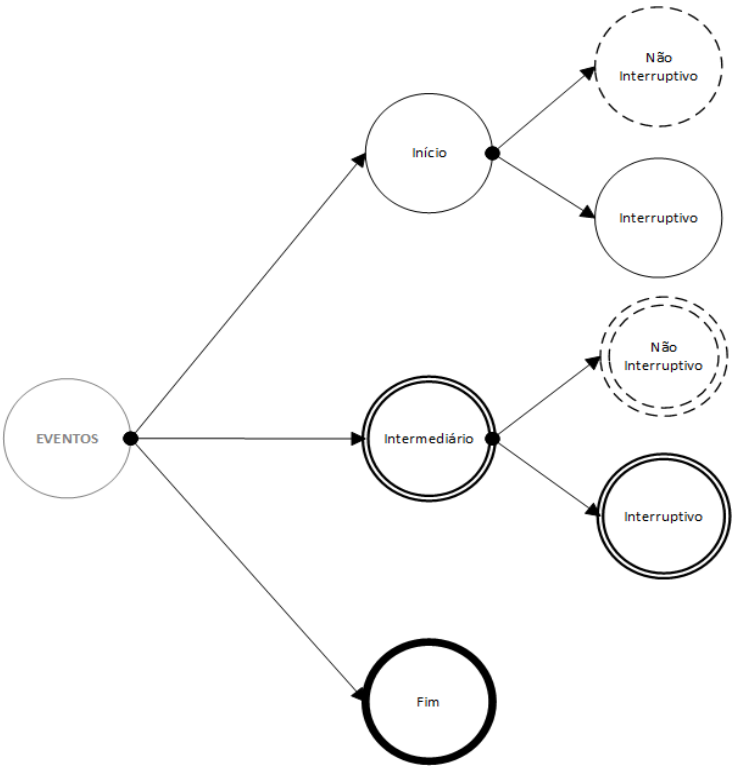



*Figura 55. Tipos de eventos.* Classificação elaborada pelo autor, com base em *(OMG, 2013)*.

Os eventos são elementos amplamente utilizados na modelagem de processos, e esses processos podem ser modelados em diferentes níveis de abstração. Por exemplo, um processo "Comprar" pode ser decomposto em subprocessos "Cotar", "Selecionar", "Pagar" e "Receber". Assim, cada nível de decomposição do processo pode ser chamado também de nível de abstração.

Os eventos podem representar comportamentos diferentes dependendo do nível de abstração modelado. Por exemplo, há eventos que podem ser utilizados em processos "Top-Level", ou seja, em um nível maior de abstração, mas que não seriam aplicáveis a um processo em outro nível. Além dos subprocessos, que podem ser de decomposição (*embedded*) ou reutiliázeves (*callable*), há também os tipos de processos, como o "Event Sub-Process" e o "Transaction", e que também podem afetar o comportamento dos eventos.

Contudo, em função da riqueza de elementos para a modelagem de processos e das diversas possibilidades de uso combinado aos tipos de processo, todo esse detalhamento não será apresentado nessa Seção. Quando necessário, por ocasião da apresentação de determinado modelo ao longo do presente trabalho, tais informações serão fornecidas.

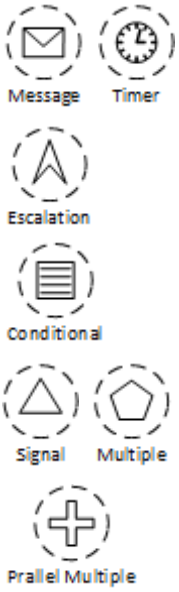
Na Figura 54 é possível observar a classe "EventDefinitions". Essa classe se refere aos gatilhos que podem ser detectados (*Cacth*) e aos que podem ser gerados (*Throw*). Existem

diversos tipos de gatilho, que serão apresentados a seguir. Os eventos de início detectam esses gatilhos, os intermediários podem tanto detectar quanto gera gatilhos, e os eventos de fim geram esses gatilhos.

Como mencionado, os eventos de início são aqueles que detectam os gatilhos que iniciam o processo ou subprocesso, ou ainda os processos do tipo “Event Sub-Process” ou “Transaction”, ou seja, detecta a ocorrência que ativa o início do processo. Os tipos de evento desse tipo são apresentados na Tabela 70.

*Tabela 70. Elementos BPMN da classe "EventDefinition", de Início, do pacote "Process", conforme (OMG, 2013).*

| Elemento BPMN | Descrição |
|---|---|
|  | O evento do tipo “None” não detecta um gatilho definido que o ativa. Nesse caso, trata-se de um início simples. |
|  | O evento do tipo “Message” detecta o gatilho de ativação por mensagem que chega de um “Participant”, ou seja, de outro “Pool”. Desse modo, é um requisito que esse evento esteja ligado a uma instância de “MessageFlow” de entrada. |
|  | O evento do tipo “Timer” detecta o gatilho de um tempo pré-definido, que pode ser uma data, uma hora, ou uma quantidade de dias, horas, minutos, segundos, etc. Desse modo, seria possível ajustar para que o evento fosse ativado toda terça as 15h, ou no primeiro dia de cada mês, por exemplo. |
|  | O evento do tipo “Conditional detecta o gatilho de uma condição que será definida com base em uma classe “Expression”. Essa, por sua vez, poderá ser formal ou informal. Um exemplo de “Expression” informal seria “Umidade relativa do ar abaixo de 60%” ou “Queda acumulada da Bovespa maior do que 10%”. |
|  | O evento do tipo “Signal” detecta o gatilho de um sinal gerado a partir de outro processo. Ele opera de maneira similar ao evento “Message”, porém com a diferença de que no “Message” a comunicação é entre dois pontos, e no “Signal” o sinal é gerado em um ponto e dispara eventos “Signal” em diversos pontos, em diversos processos, ou seja, trata-se de uma mensagem do tipo broadcast. |
|  | O evento do tipo “Multiple” detecta o gatilho da ocorrência de um de diversos possíveis gatilhos, ou seja, é o caso onde a classe “CatchEvent” tem associada mais de uma instância da classe “EventDefinition”. Dese modo, podem haver diversos gatilhos associados. Aquele que ocorrer primeiro ativará o evento “Multiple”, o que configura uma operação de OR exclusivo entre os diversos possíveis gatilhos. |
|  | O evento do tipo “Parallel Multiple” opera de maneira similar ao evento “Multiple”, com a diferença de que o “Parallel Multiple” exige o atendimento de diversos gatilhos para que ele seja ativado, o que equivale a uma operação de AND entre esses diversos gatilhos. |
|  | O evento do tipo “Error” detecta o gatilho da ocorrência de um critério que caracterize um erro no processo. Ele ativará um “Event Sub-Process”, obrigatoriamente de maneira interruptiva. |

| | |
|---|---|
|  | O evento do tipo “Escalation” detecta o gatilho da ocorrência de um critério que determina se o processo foi concluído satisfatoriamente ou não, como por exemplo um prazo máximo ou um resultado esperado. Caso esse critério não seja atingido, ele ativa um “Event Sub-Process”. |
|  | O evento do tipo “Compensation” detecta o gatilho da necessidade de uma compensação de uma atividade, seja do tipo “Task” ou “Process”, de maneira que as ações realizadas na atividade a ser compensada sejam desfeitas. |
|  | Os eventos do tipo não interruptivo apresentam operação similar aos seus equivalentes do tipo interruptivo, com a diferença de que os não interruptivos provocam a ativação de “Event Supb-Process” de maneira paralela ao próprio processo que a provocou, ou seja, não interrompe o fluxo do processo principal. |

Os eventos intermediários podem tanto detectar gatilhos quanto provocá-los, ou lançá-los. Esses eventos são fundamentais para a descrição adequada dos motivos de ativação de outros elementos especializados a partir da classe “FlowNode”. Ainda, alguns desses eventos podem ser adicionados à borda de uma atividade (seja “Task” ou “Process”). Nesse caso, os eventos de borda podem ter dois comportamentos. Quando o evento é do tipo interruptivo e ele é detectado, a atividade é interrompida e o fluxo do processo é desviado para onde for indicado pelo “SquenceFlow” que sai do referido evento. Quando o evento é do tipo não interruptivo, a atividade continua seu processo e paralelamente há também uma saída pelo “SequenceFlow” que sai do evento não interruptivo. No primeiro caso fica configurado uma operação do tipo “OR”, e no segundo caso uma operação do tipo “AND”. Os tipos de evento intermediário são apresentados na Tabela 71.

*Tabela 71. Elementos BPMN da classe "EventDefinition", Intermediários, do pacote "Process", conforme (OMG, 2013).*

| **Elemento BPMN** | | | | **Descrição** |
|---|---|---|---|---|
| **Detecta (Catch)** | **Borda Interruptivo** | **Borda Não Interruptivo** | **Provoca (Throw)** | |
| | | |  | Os eventos do tipo “None” são utilizados para representar mudança de estado ao longo do processo, e não possuem um gatilho específico (classe “EventDefinitions”). |
|  |  |  |  | Os eventos do tipo “Message” podem ser amplamente utilizados para enviar uma mensagem (*Throw*), ou para detectá-la, quer dizer, para recebê-la (*Catch*). Ainda pode ser utilizado na borda de atividades, tanto de modo interruptivo quanto não interruptivo. |
|  |  |  | | Os eventos do tipo “Timer” são utilizados apenas para detectar uma data/hora específica. Então pode ser utilizado como *delay* em um fluxo normal de processo ou como um temporizador na borda de uma atividade. |
| |  | | | Os eventos do tipo “Error” podem ser utilizados apenas na borda de atividades, e provocam o cancelamento dessa atividade ao serem ativados. |
| |  |  |  | Os eventos do tipo “Escalation” são utilizados para provocar a ativação de um comportamento de escalação em um processo, ou seja, ele não aborta a atividade ou o processo, apenas direciona para o tratamento em um caminho alternativo. |
| |  | | | Os eventos do tipo “Cancel” são utilizados apenas na borda de atividades do tipo “Transaction Sub-Process”, e servem para indicar que a transação foi cancelada definitivamente, sem *rollback*. |
| |  | |  | Os eventos do tipo “Compensation” são utilizados para detectar a necessidade de uma compensação, quando utilizados na borda de atividades, ou para indicar a necessidade de uma compensação, se utilizados no fluxo normal do |

| | | | | |
|---|---|---|---|---|
| | | | | processo em sua versão de lançamento (*Throw*). Ele é utilizado para provocar um *rollback*. |
|  |  |  | | Os eventos do tipo "Conditional" são utilizados para detectar a ocorrência de uma determinada condição, definida por uma classe "Expressions" (formal ou informal). O evento pode ser utilizado tanto no fluxo normal do processo quanto na borda de atividades. Nesse caso, tanto de maneira interruptiva quanto não interruptiva. |
|  | | |  | Os eventos do tipo "Link" são válidos apenas em fluxo normal de processo, ou seja, não podem ser utilizados na borda de atividades. Eles são utilizados para conectar duas seções de um mesmo processo, quando eventualmente há a necessidade de indicar a continuidade de um processo de uma página para outra do mesmo diagrama. A aplicação desse evento dependerá em grande parte da implementação realizada pela ferramenta de modelagem. |
|  |  |  |  | Os eventos do tipo "Signal" são utilizados para enviar (provocar) e receber (detectar) sinais, ou seja, de transmitir uma informação entre diferentes participantes, ou "Pools". Embora tenham funcionalidade similar à "Message", os eventos do tipo "Signal" enviam uma mensagem em boradcast, ou seja, uma única mensagem que pode ser recebida simultaneamente em várias "Pools". |
|  |  |  |  | Os eventos do tipo "Multiple" são utilizados para provocar e detectar múltiplos gatilhos ("EventDefinitions"), sendo que a detecção de apenas um dentre os diversos definidos já é suficiente para ativar o evento. |
|  | |  | | Os eventos do tipo "Multiple" são utilizados para detectar múltiplos gatilhos ("EventDefinitions"), sendo que será necessária a detecção de todos os gatilhos definidos. |

Os eventos de fim, naturalmente, identificam o fim de um processo, após o qual não há mais nenhum outro elemento de fluxo (classes “SequenceFlow” e “FlowNode”). Contudo, um processo poderá encerrar disparando um gatilho (classe “EventDefinitions”) a ser detectado por outro processo, ou seja, gerando um resultado. E para esse objetivo existem os eventos de fim apresentados na

Tabela 72.

*Tabela 72. Elementos BPMN da classe "EventDefinition", de Fim, do pacote "Process", conforme (OMG, 2013).*

| Elemento BPMN | Descrição |
|---|---|
|  | Os eventos do tipo “None” não definem nenhum resultado, ou seja, não geram nenhum gatilho. |
|  Message | Os eventos do tipo “Message” finalizam o processo enviando uma mensagem para um participante (“Pool”), identificado através da uma “MessageFlow” entre o processo de origem e o participante (destinatário). |
|  Error | Os eventos do tipo “Error” finalizam o processo gerando um gatilho de erro. Todas as threads ativas da classe “Sub-Process” em questão são terminadas. O erro poderá ser recebido (detectado) por um evento intermediário de erro na borda da atividade pai, cuja sincronização será feita através do código de erro. |
|  Escalation | Os eventos do tipo “Escalation” geram um gatilho de escalação, que será recebido (detectado) por evento intermediário de “Escalation” na borda da atividade pai, possibilitando a execução por um fluxo que trata a situação identificada, mas que também mantém o fluxo normal das atividades subsequentes. |
|  Cancel | Os eventos do tipo “Cancel” são utilizados dentro de atividades do tipo “Transaction Sub-Process”, e indicam que a transação precisa ser cancelada. O gatilho gerado será recebido (detectado) por evento intermediário de “Cancel” na borda da atividade pai. |
|  Compensation | Os eventos do tipo “Compensation” são utilizados para compensar (desfazer) atividades que tenham sido completadas. |
|  Signal | Os eventos do tipo “Signal” encerram o processo disparando um sinal em broadcast, que será detectado por eventos de início ou mesmo intermediários do tipo “Signal”. |
|  Terminate | Os eventos do tipo “Terminate” são utilizados para indicar que todas as atividades de um processo devem ser encerradas imediatamente, incluindo todas as instâncias simples e instâncias múltiplas que eventualmente estiverem ativas. O encerramento é realizado sem compensação e sem geração de gatilhos. |

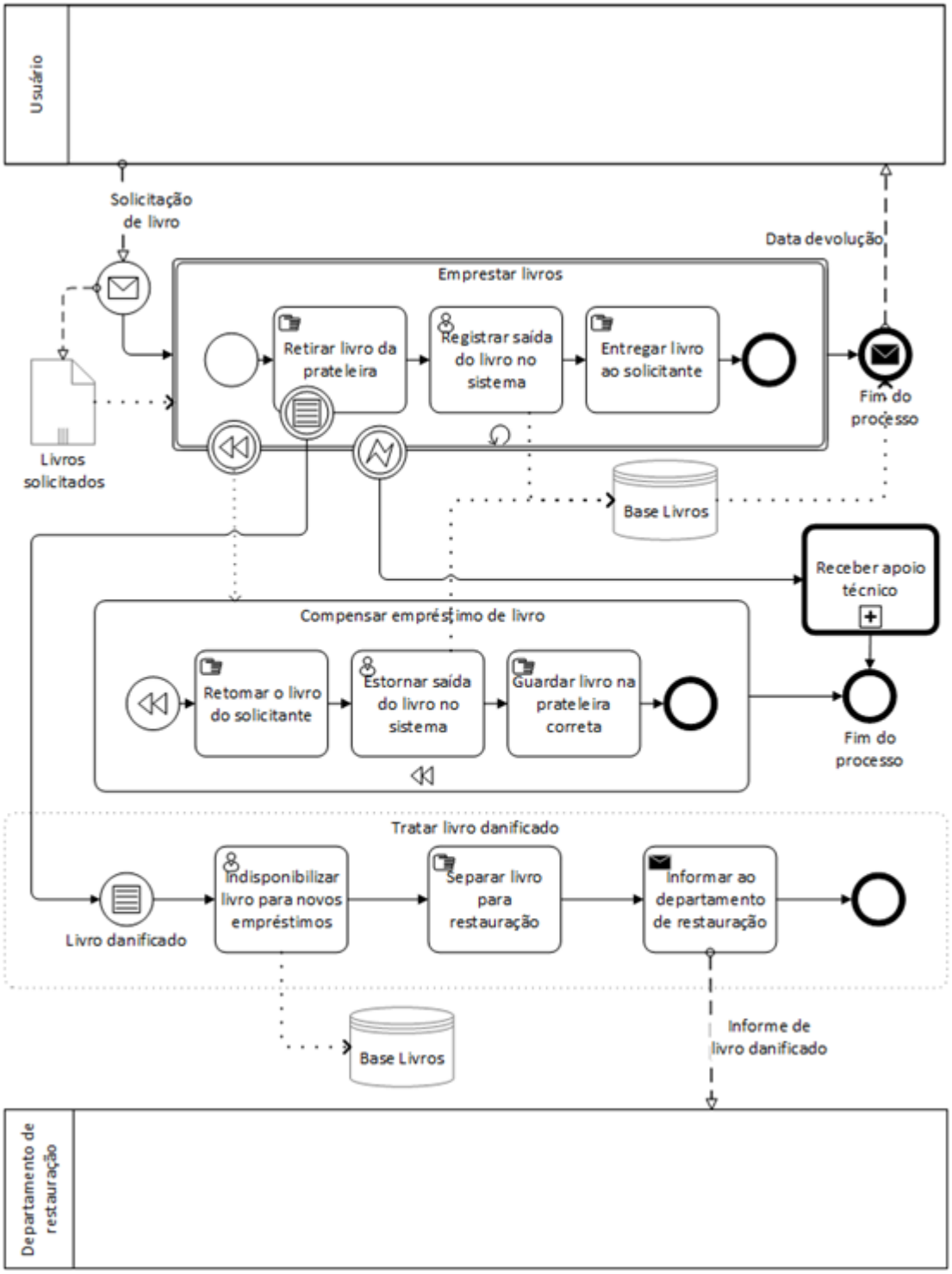



*Figura 56. Processo ilustrativo do uso dos pacotes "Process" e "Common" (elaborado pelo autor).*

Com o objetivo de ilustrar a utilização dos elementos apresentados até este ponto do trabalho, referente as classes do pacote "Process", em conexão com as classes do pacote

“Common”, um modelo de processo hipotético é apresentado na

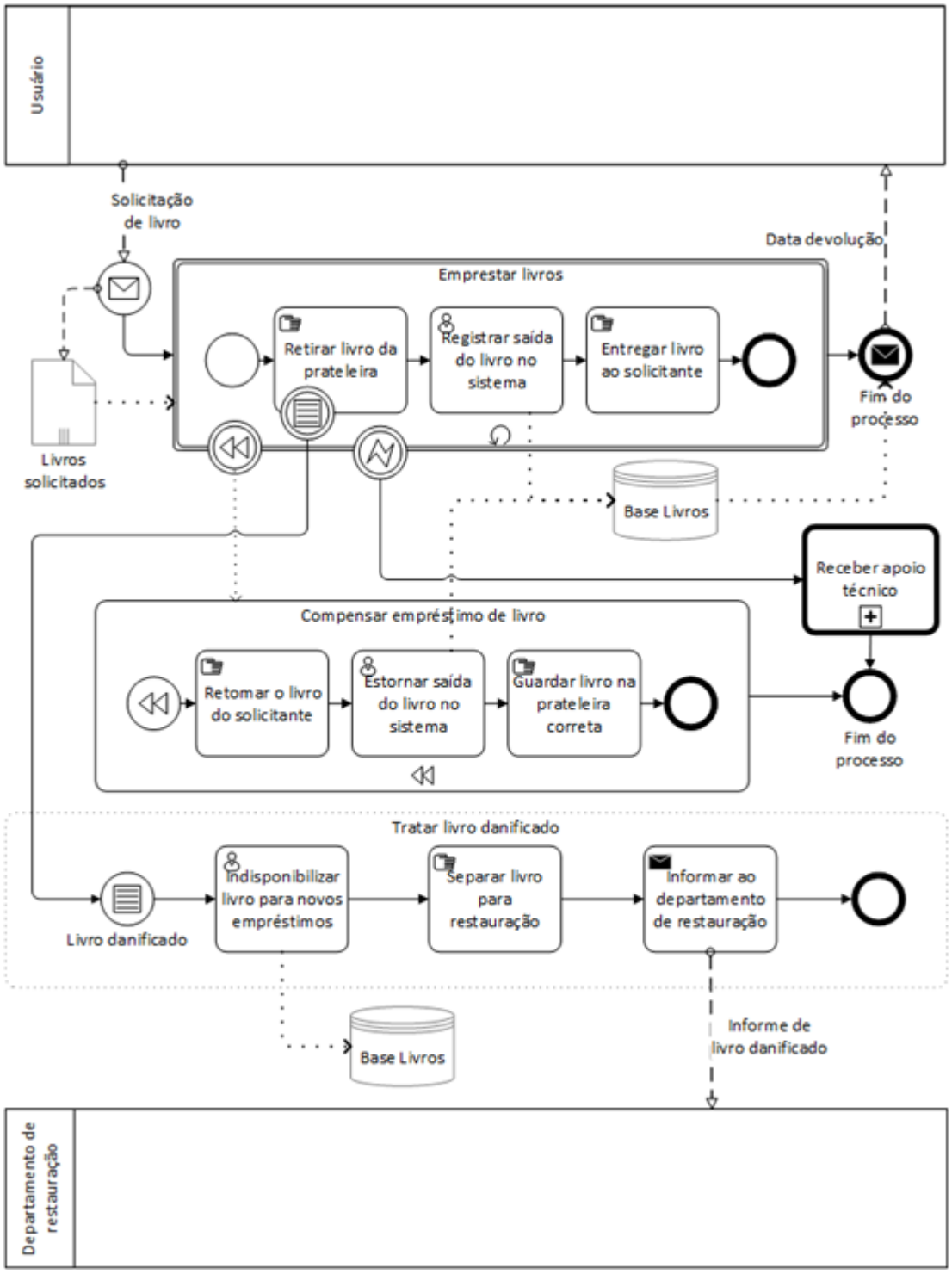



Figura 56. Nesse processo, o usuário figura como participante externo (Pool), remetente da mensagem “Solicitação do livro”. O processo “Emprestar livros” é um subprocesso do tipo “Transaction”, o que estabelece um escopo para suas atividades internas, de modo que todas as atividades precisam ser bem-sucedidas a fim de que o processo encerre seu fluxo normal. Caso isso ocorra, o processo é finalizado com o envido da mensagem “Data de devolução” para o participante externo “Usuário”.

É importante destacar que uma mensagem “Solicitação de livro” poderá gerar uma ou várias instâncias de “Emprestar livros”, que é um subprocesso com o marcador de “Loop”. Ou seja, haverá uma instância de “Emprestar livros” para cada um do total de livros informados na mensagem “Solicitação de livro”.

Contudo, há duas possibilidades de exceção no processo transacional “Emprestar livros”. A primeira delas é a possibilidade de haver problemas em alguma de suas atividades (“Task”), como por exemplo, um dado inconsistente no sistema de registro de livros. Para uma atividade transacional isso caracterizaria uma ***Failed Completion***, pois embora não seja um erro grave, o mais adequado seria desfazer a transação apenas para o livro em questão (ou seja, a instancia corrente do processo) e seguir com o empréstimo de outros eventuais livros solicitados pelo usuário. Isso ativaria o evento “Compensation”, ativando o subprocesso “Compensar empréstimo de livro”, e seguindo para o encerramento do processo para aquela instância de “Emprestar livros”.

Outra possibilidade de exceção do processo transacional “Emprestar livros” é a ocorrência de um erro, ou seja, um ***Hazard***. Nesse caso, trata-se de uma exceção grave que impede a continuidade do processo, e nem mesmo o subprocesso de compensação pode ser executado. Talvez o sistema utilizado no processo apresente mal funcionamento, ao ponto de ser necessário abortar o processo e executar o subprocesso “Receber apoio técnico”, após o qual aquele processo é encerrado. Qualquer medida corretiva em relação ao erro ocorrido no processo “Emprestar livros” será tomada dentro do processo “Receber apoio técnico”.

Por fim, há um subprocesso que pode acontecer a qualquer momento do processo. Trata-se do subprocesso por evento “Tratar livro danificado”. Ele não é ativado por uma seta de fluxo sequencial, mas sim por uma condição: “Livro danificado”. Então, se um livro danificado é identificado no processo, ele é retirado da lista de possíveis empréstimos e separado para futura restauração. De fato, antes do encerramento desse subprocesso, uma atividade envia a mensagem “Informe de livro danificado” para o participante externo “Departamento de restauração”.

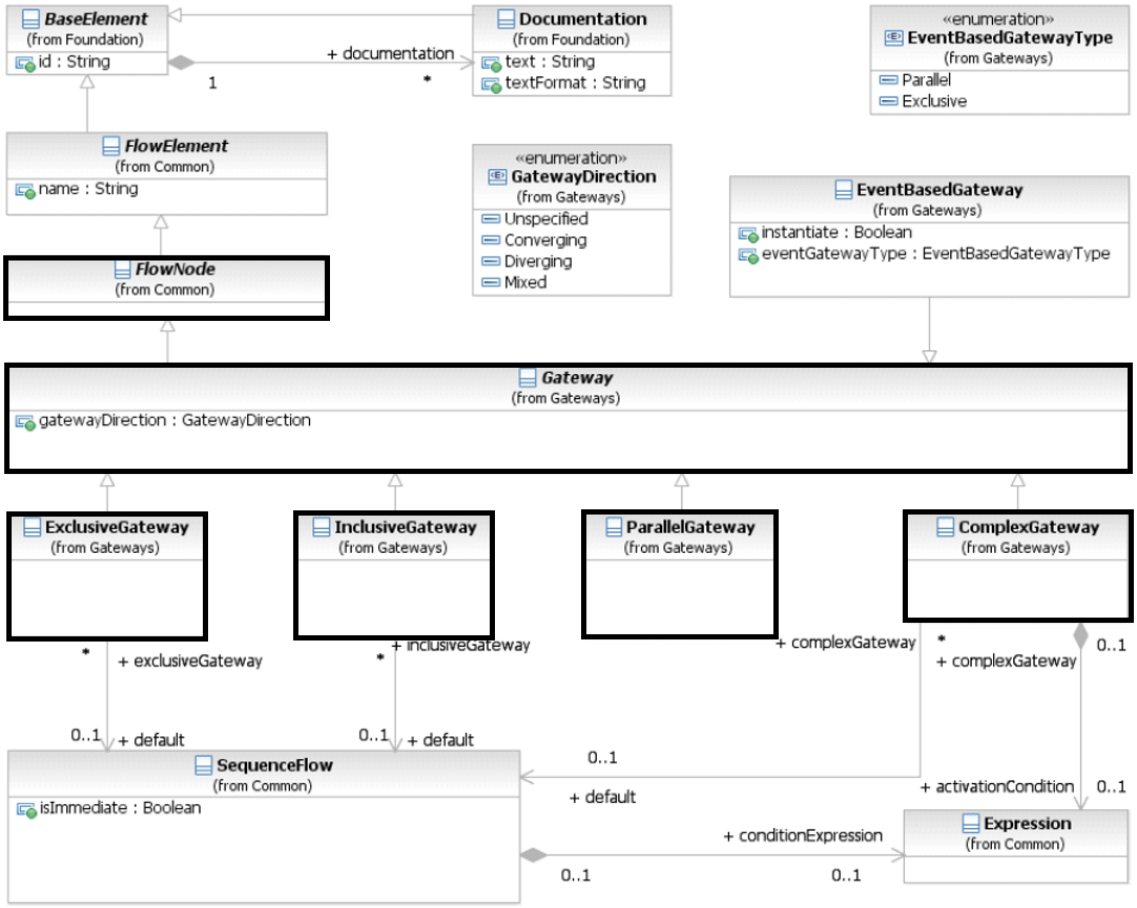


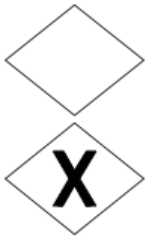

*Figura 57. Diagrama de classe de "Gateway", do pacote "Process", conforme (OMG, 2013).*

Além das classes referentes a eventos, outro importante conjunto de classes do metamodelo BPMN são aqueles referentes aos Gateways, apresentadas na Figura 57. A classe abstrata “Gatway” é uma herança da classe abstrata “FlowNode”, como os demais elementos do pacote “Process”, a partir da qual são herdadas quatro classes concretas; “ExclusiveGateway”, “InclusiveGateway”, “ParallelGateway” e “ComplexGatway”.

Os elementos do tipo “Gateway” são usados para controlar a interação dos “SequenceFlows” no processo, tratando das convergências e divergências desses fluxos sequenciais. A execução do processo acontece através da passagem por uma sequência de elementos do tipo “FlowNode”, ou seja, atividades, eventos, fluxos sequenciais e gateways. Essa passagem pode ser compreendida como um pequeno círculo imaginário que trafega através dos “FlowNode”, e esse pequeno círculo imaginário é chamado de *token*. As classes “Gateway” (“Portão”, em português) são assim denominadas por viabilizarem um mecanismo de “pare/siga” ao processo modelado, possibilitando que o fluxo do *token* seja controlado pela ativação ou não dos gateways. Assim, diversos *tokens* podem chegar a um gateway e serem convertidos em um único *token* de saída, ou um

token poder chegar a um gateway e ser transformado em diversos *tokens* de saída, ou ainda um *token* pode chegar a um gateway e sair por uma ou mais saídas entre diversas opções de saída.

*Tabela 73. Elementos BPMN das classes "Gateway", do pacote "Process", conforme (OMG, 2013).*

| Elemento BPMN | Descrição |
|---|---|
|  | Um gateway do tipo exclusivo (classe “ExclusiveGatway”) é utilizado para estabelecer caminhos alternativos para o fluxo do processo, que pode tanto ser uma definição de diversos caminhos de entrada e um de saída (convergência) quanto um caminho de entrada e diversos caminhos de saída (divergência). A decisão sobre o caminho a ser tomado (entrada ou saída) é feito com base em questão com uma série de alternativas, sendo cada alternativa indicada por uma “Exspression”, que pode ser formal ou informal. O símbolo padrão para esse tipo de gateway é o losango sem marcação, mas pode ser utilizado opcionalmente o losango com a marcação de um “X” em seu interior. Ambos os símbolos possuem o mesmo significado. Há ainda a possibilidade de estabelecer um dos caminhos múltiplos, seja na entrada ou na saída, como o caminho padrão (*default*), o que visualmente é identificado por uma seta de fluxo com uma pequena linha cortando-o diagonalmente. |
|  | Um gateway do tipo inclusivo (classe “InclusiveGateway”) pode ser utilizado tanto para convergência de vários caminhos de entrada, quanto para divergência, gerando diversos caminhos de saída. A diferença é que mais de uma “Expression” poderá ser verdadeira, tanto na entrada quanto na saída, de modo que múltiplos caminhos poderão ser seguidos simultaneamente. |
|  | Um gateway tipo paralelo (classe “ParallelGateway”) pode ser utilizado tanto para convergência quanto para divergência, porém exige que a “Expression” de todos os caminhos seja verdadeira para que o “portão” se abra, ou seja, para que o fluxo entre ou saia dele. Ele é utilizado para abrir ou fechar caminhos paralelos em processos. |
|  | Um gateway do tipo complexo (classe “ComplexGateway”), utilizado em convergências e divergências, possibilita a utilização de condições complexas, através de uma “Expression” bem definida. Por exemplo, podem haver cinco caminhos de entrada, sendo que se dois desses caminhos forem verdadeiros, independentemente de quais dois sejam, o “portão” se abre para o caminho de saída. Ou, por outro lado, se há cinco caminhos de saída, dependendo do que foi definido em “Expression” os caminhos de saída um e três são ativados, ou os caminhos dois e quatro são ativados, ou os caminhos dois, três e cinco são ativados. |
|  | Um gateway do tipo baseado em evento (classe “EventBasedGateway”, uma herança da classe abstrata “Gateway”, com a seleção “Exclusive” da enumeração “EventBasedGatewayType”) opera de maneira similar ao gateway do tipo exclusivo, com diferença de que esse gateway define um entre vários caminhos com base em um evento, e não com base em uma “Expression” formal ou informal. O evento que fornece informação para uma divergência traz informação de fora do processo (como por exemplo as classes “Message” e “Signal”) que são detectadas no processo corrente, e a partir da informação nesse evento, um entre vários caminhos é seguido no fluxo do processo. |
|  | Um gateway do tipo evento paralelo (classe “EventBasedGateway”, uma herança da classe abstrata “Gateway”, com a seleção “Parallel” da enumeração “EventBasedGatewayType”), opera de maneira similar ao gateway do tipo “EventBasedGateway” exclusivo, com a diferença de que apenas quando todos os eventos ocorrem é que o fluxo segue. |

A Figura 58 apresenta o uso de gateways em um modelo de processo, mas especificamente os gateways do tipo “Exclusive” e do tipo “Event-Base” setado para “Exclusive”, que foi criado com base no modelo OpenUP (ECLIPSE FOUNDATION, 2013). O processo de desenvolvimento do documento de visão (*Develop Techinical Vision*) é realizado pelo analista (*Analyst*), que troca diversas informações com os stakeholders do projeto em questão. O primeiro passo desse processo é identificar os stakeholders, e caso esses somem mais do que 10 pessoas, esses stakeholders são agrupados em classes (*Cluster stakeholders*), tais como, por exemplo, gerentes, usuários finais, decisores, fonte de informação, ou qualquer outra classificação que se faça necessária. Em seguida o analista definirá o problema, ou escopo a ser abordado pelo projeto, e solicitará a concordância dos stakeholders (*Request agreement on the problem*). Logo após essa atividade, há um gateway do tipo evento, que poderá ser seguido de um dos dois eventos de mensagem, ou seja, poderá receber uma das duas mensagens: “Agreed” ou “Not Agreed”. O primeiro evento que ocorrer ativará o fluxo (“FlowSequence”) a ele ligado, quer dizer, se a resposta for “Not agreed”, o fluxo segue novamente para “Request agreement on the problem”, e se a resposta for “Agreed” o fluxo segue para a atividade “Gather stakeholder requests”. Situação similar acontece com a atividade “Achieve concurrence”.

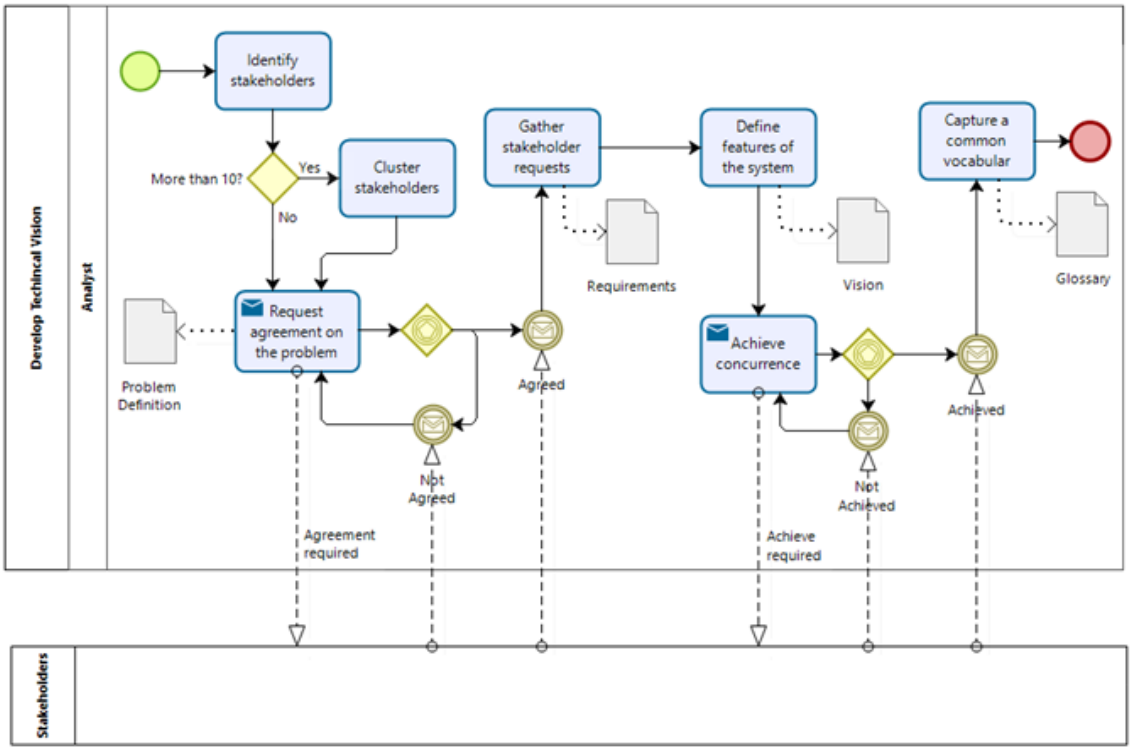



*Figura 58. Processo ilustrativo do uso dos gateways, e outros elementos dos pacotes “Process” e “Common” (elaborado pelo autor).*

Como pode ser observado, o metamodelo BPMN proporciona uma notação abrangente para a modelagem de processos, que além de possibilitar uma infinidade de combinações entre seus elementos para modelar virtualmente todo tipo de processo, incluindo o de software (CAMPOS e OLIVERIA, 2011), ainda conta com a possibilidade de suportar essa modelagem com estruturas formais, que viabilizarão a passagem da representação para a execução desse processo.

Desse modo, um processo de software modelado com a notação BPMN poderia estabelecer um caminho do processo à execução, sem quebras conceituais entre os mecanismos que suportam a representação, como ferramentas de modelagem, e os que suportam a execução, como ferramentas de execução do processo, tais como, por exemplo, as ferramentas de BPMS (Business Process Management Suite), que incluem as máquinas de execução do processo. Esse fluxo integrado é apresentado na Figura 59.

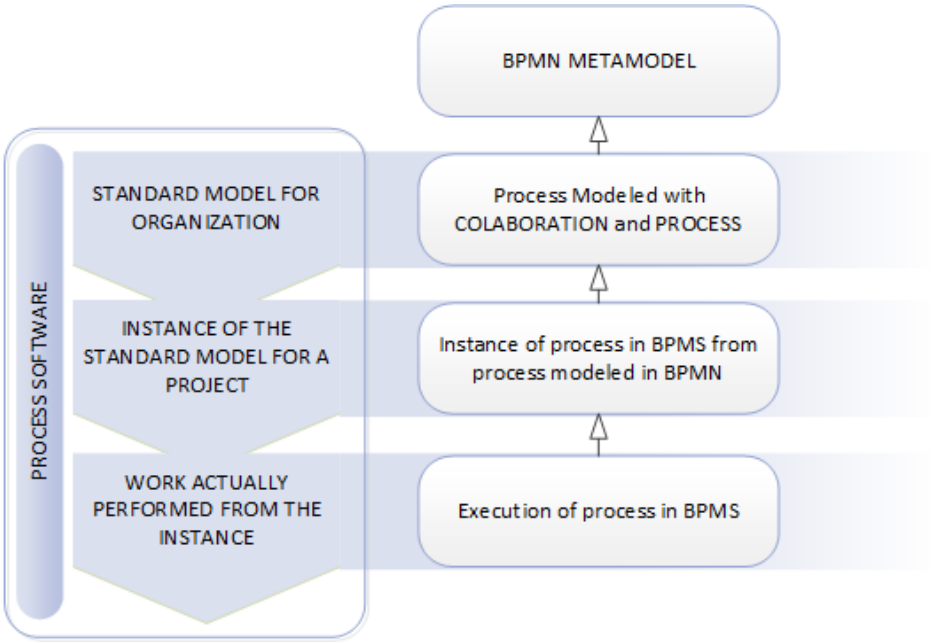
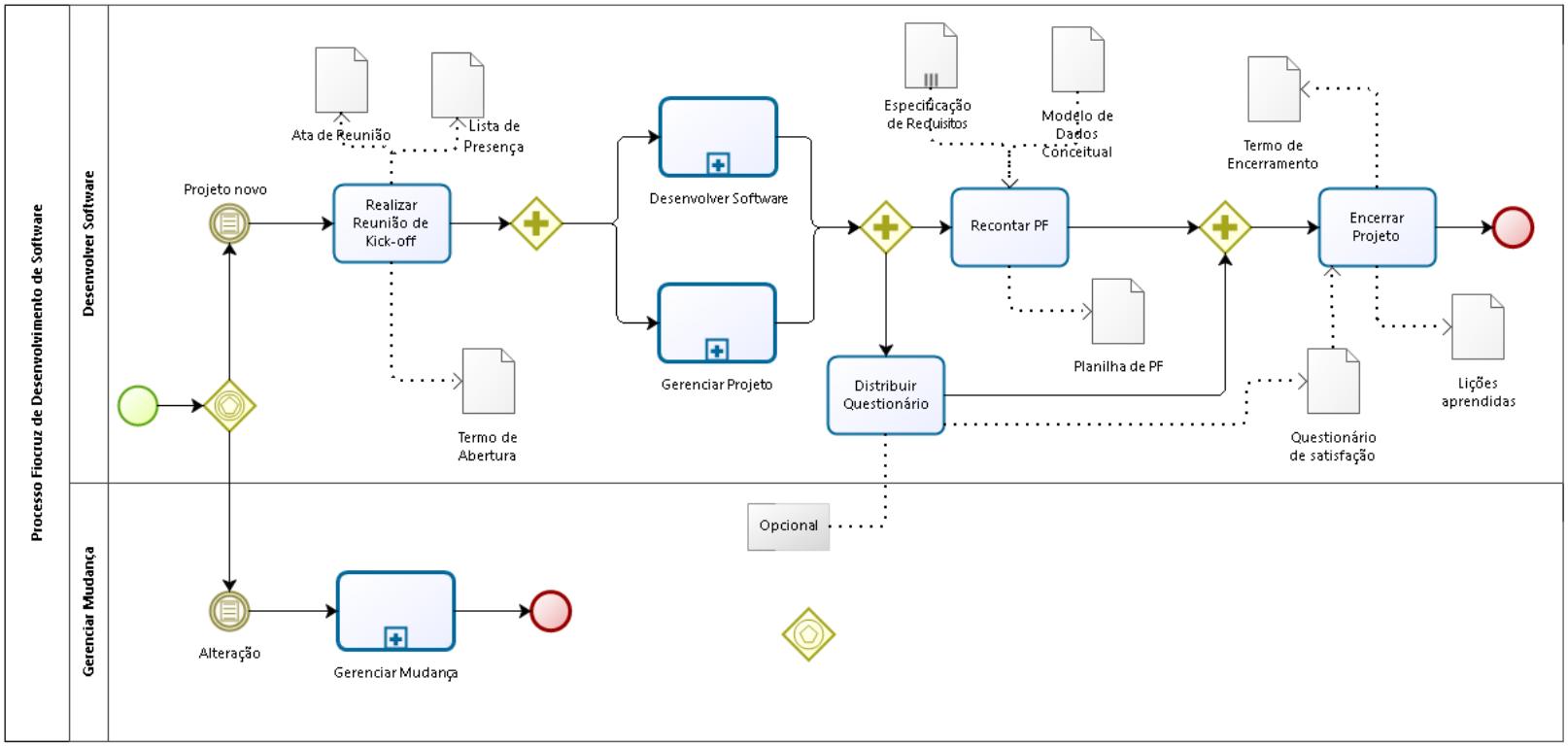



*Figura 59. Estrutura de utilização do BPMN para PDS (figura do autor).*

**Anexo II – MDS Fiocruz original**

# 1] MDS Fiocruz



**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

# Processo Fiocruz de Desenvolvimento de Software

Processo principal que descreve o desenvolvimento de software no âmbito da Fiocruz. Incorpora características de processo evolutivo e incremental, controle de qualidade através de mecanismos de verificação (inspeção) e validação (testes funcionais) e gerenciamento básico das atividades de desenvolvimento.

## Elementos do processo

### *Realizar Reunião de Kick-off*

Nessa atividade é realizada uma apresentação da metodologia a ser seguida no desenvolvimento do sistema. Além disso, é gerado o Termo de Abertura e uma Ata de Reunião com as principais decisões tomadas. São definidos na reunião:

- Pontos focais
- Cronograma inicial para reuniões de mapeamento de processos de negócio
- Cadeia de valor agregado (lista de macroprocessos)
- Visão inicial das fronteiras do sistema

### *Desenvolver Software*

Processo que descreve o desenvolvimento do software propriamente dito, com atividades que vão desde o mapeamento de processos de negócio até o teste e implantação do sistema.

### *Recontar PF*

Nessa atividade são recontados os pontos de função do sistema, com o objetivo de comparar a contagem final com a contagem inicial.

### *Encerrar Projeto*

Nessa atividade é realizada a reunião de fechamento do projeto para avaliação de pendências, pontos positivos e pontos negativos do projeto. Também são elaborados o Termo de Encerramento e o documento de Lições Aprendidas.

### *Distribuir Questionário*

Essa atividade é opcional, e deve ser realizada com o objetivo de elaborar e distribuir para os stakeholders um questionário de satisfação do projeto. Na versão atual do MDS não existe um template ou sugestão para elaboração desse questionário.

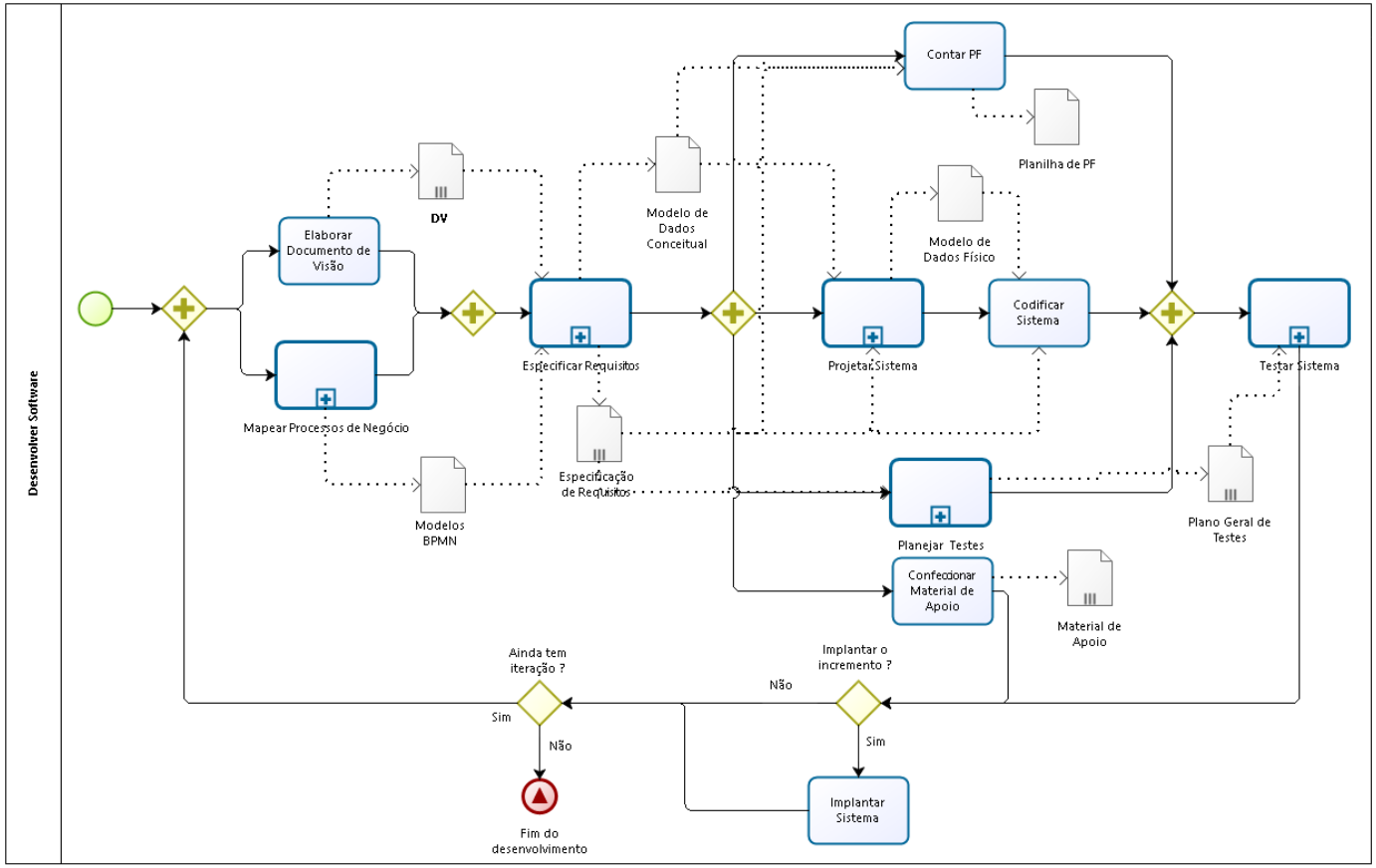
### *Gerenciar Projeto*

Processo que descreve o gerenciamento do projeto, englobando a elaboração do cronograma e o acompanhamento periódico das atividades planejadas.

 *Gerenciar Mudança*

Processo responsável por tratar as diversas solicitações de mudança, seja em sistemas que estão em processo de desenvolvimento ou em sistema em produção (manutenção).

# 2] Desenvolver




Powered by bizagi Modeler

**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

## Desenvolver Software

Processo que descreve o desenvolvimento do software propriamente dito, com atividades que vão desde o mapeamento de processos de negócio até o teste e implantação do sistema.

## Elementos do processo

### *Mapear Processos de Negócio*

Processo que define o fluxo de atividades necessárias para a elicitação, modelagem, verificação e validação dos processos de negócio a serem capturados pelo sistema em desenvolvimento. O conjunto de modelos de processos de negócio validado oferece uma visão geral das potenciais atividades a serem automatizadas pelo sistema.

### *Especificar Requisitos*

Processo que tem como objetivo a elicitação, análise e especificação dos requisitos do sistema em desenvolvimento.

### *Projetar Sistema*

Processo responsável pela modelagem arquitetural e em nível de projeto do sistema em desenvolvimento, a partir dos requisitos elicitados na macroatividade de Especificação de Requisitos. Esta modelagem envolve tanto aspectos estruturais quanto comportamentais dos componentes do sistema.

### *Codificar Sistema*

Atividade de codificação ou programação onde o sistema é construído com base nas especificações e modelos previamente definidos.

### *Testar Sistema*

Processo que tem como objetivo executar os testes funcionais do sistema em desenvolvimento com base nas especificações de testes elaboradas previamente.

### *Implantar Sistema*

Nesta atividade deve ser preparado um pacote contendo os componentes necessários (incluindo o próprio sistema e scripts de instalação) para implantação do sistema em seu ambiente de operação, bem como treinamento dos usuários.

### *Elaborar Documento de Visão*

Atividade com objetivo de produzir os primeiros artefatos técnicos, que organizam a visão geral do sistema a ser desenvolvido.

### *Contar PF*

Atividade de contagem de Pontos de Função do sistema com base na especificação de requisitos e nos modelos produzidos.

### *Confeccionar Material de Apoio*

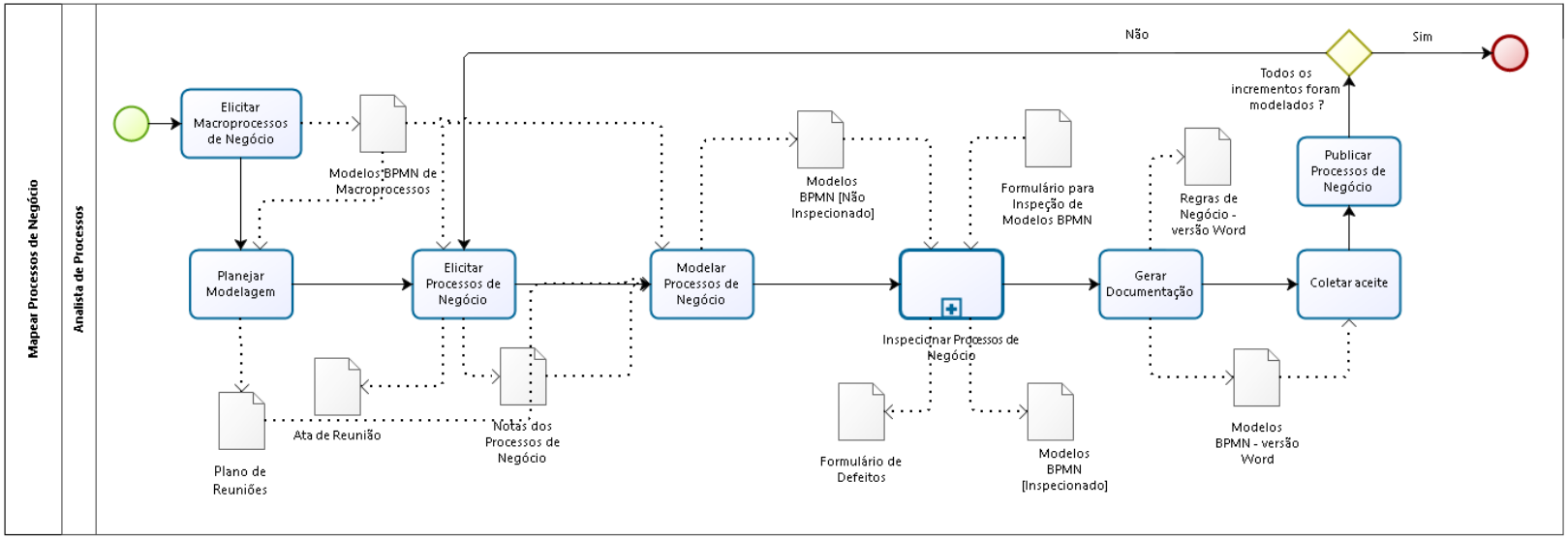
Nessa atividade deverá ser confeccionado o material de apoio aos usuários. Esse material pode ser um manual do usuário, um manual de operação, um vídeo de treinamento, um guia de navegação ou qualquer outro material de apoio aos usuários do sistema. É importante que o material a ser produzido esteja definido de forma detalhada no escopo do projeto que consta no Termo de Abertura.

### *Planejar Testes*

Processo que tem como objetivo elaborar a documentação necessária para planejar e especificar os testes do sistema:

- Definir escopo de teste e abordagens de teste (funcional, desempenho, usabilidade, dentre outros)
- Definir itens de teste (para os testes funcionais, definir quais casos de uso serão testados)
- Definir critérios de aceitação
- Definir a equipe de testes (analistas de teste e testadores)
- Definir grupo de usuários do teste de aceitação
- Elaborar casos e procedimentos de teste

# 2.1] Modelar processo




Powered by bizagi Modeler

**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

# Mapear Processos de Negócio

Processo onde os processos e as regras de negócio são elicitados, modelados usando BPMN e inspecionados. Como resultado principal temos os Modelos BPMN que representam o escopo do negócio onde o sistema está inserido.

## Elementos do processo

### *Elicitar Macroprocessos de Negócio*

Nesta atividade são definidos, por meio de uma ou mais reuniões, uma lista com os macroprocessos de negócio que definem o escopo do negócio onde o sistema a ser desenvolvido está inserido, juntamente com uma descrição associada a cada macroprocesso e, também é criada a estrutura de diretórios do projeto.

### *Planejar Modelagem*

Nesta atividade é definido como o processo de modelagem será executado, ou seja, se os processos de negócio serão elicitados, modelados, verificados e validados de forma completamente sequencial ou se os mesmos serão divididos em conjuntos menores (por macroprocesso, por unidade funcional, ou outras divisões), com o processo sendo executado de forma iterativa (em ciclos) para cada conjunto definido.

### *Elicitar Processos de Negócio*

Para cada macroprocesso identificado anteriormente, seus subprocessos, responsáveis (papéis), atividades (e seus encadeamentos) e documentos (consumidos e produzidos) são identificados. São realizadas entrevistas ou reuniões com os stakeholders relacionados a estes subprocessos.

### *Modelar Processos de Negócio*

Conforme os processos de negócio vão sendo entendidos pelos Analistas de Negócio, os mesmos são modelados utilizando a notação BPMN na ferramenta BizAgi.

### *Inspecionar Processos de Negócio*

Nesta atividade, os modelos de processos de negócio são verificados com base em um checklist que captura heurísticas de boa construção de modelos.

A atividade de verificação é interna ao projeto e não envolve o cliente.

### *Gerar Documentação*

A partir dos modelos BPMN devem ser gerados dois documentos (usando as funcionalidades do Bizagi):

- Documento completo com os todos os modelos de processos de negócio

- Documento reduzido contendo apenas as regras de negócio do modelo

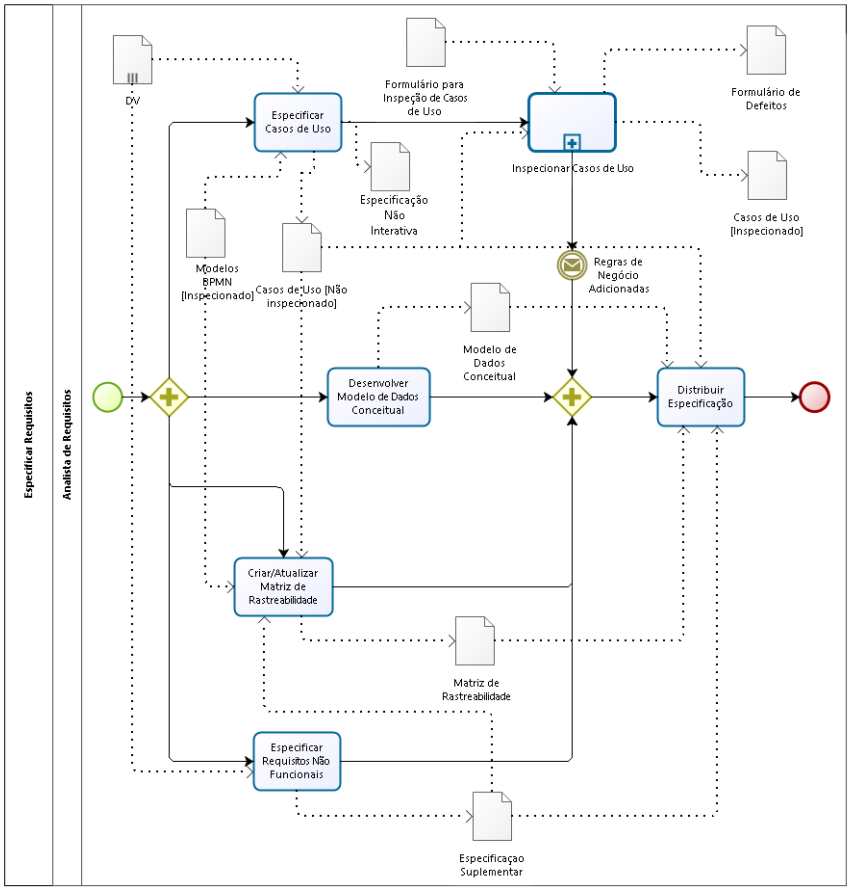
### *Coletar aceite*

São coletadas as assinaturas dos principais envolvidos no próprio documento de modelos de processos de negócio gerado com o Bizagi

☐ *Publicar Processos de Negócio*

Após a validação dos modelos de processos de negócio com a aprovação pela Equipe de Validação, é necessário tornar estes modelos públicos por meio de uma versão web e notificar todos interessados.

# 1 2.2] Especificar Requisitos



**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

# Especificar Requisitos

Processo onde é realizado a elicitação, a especificação e a inspeção dos requisitos funcionais e não funcionais do sistema.  Como resultado principal temos o Documento de Especificação de Requisitos que será a base de todo o desenvolvimento.

## Elementos do processo

### *Especificar Casos de Uso*

Nesta atividade, os requisitos funcionais identificados e registrados no Documento de Visão são analisados e então descritos no formato de casos de uso e de processos não interativos.

Os Modelos BPMN também devem ser consultados para capturar a parte dos processos de negócio que estão sendo automatizadas, bem como as regras de negócio

### *Inspecionar Casos de Uso*

Nesta atividade, os casos de uso são verificados com base em um checklist que define regras de boa construção.

A atividade de verificação é interna ao projeto e não envolve o cliente.

### *Distribuir Especificação*

Distribuição ou liberação dos documentos de especificação para a equipe de desenvolvimento. Quando for utilizado um software de controle de versão, é recomendado que essa liberação ocorra usando o mecanismo de TAG ou BRANCH do controle de versão e comunicando aos demais membros da equipe o diretório no qual os documentos liberados estão armazenados.

### *Criar/Atualizar Matriz de Rastreabilidade*

Esta atividade está associada ou à criação ou atualização da Matriz de Rastreabilidade

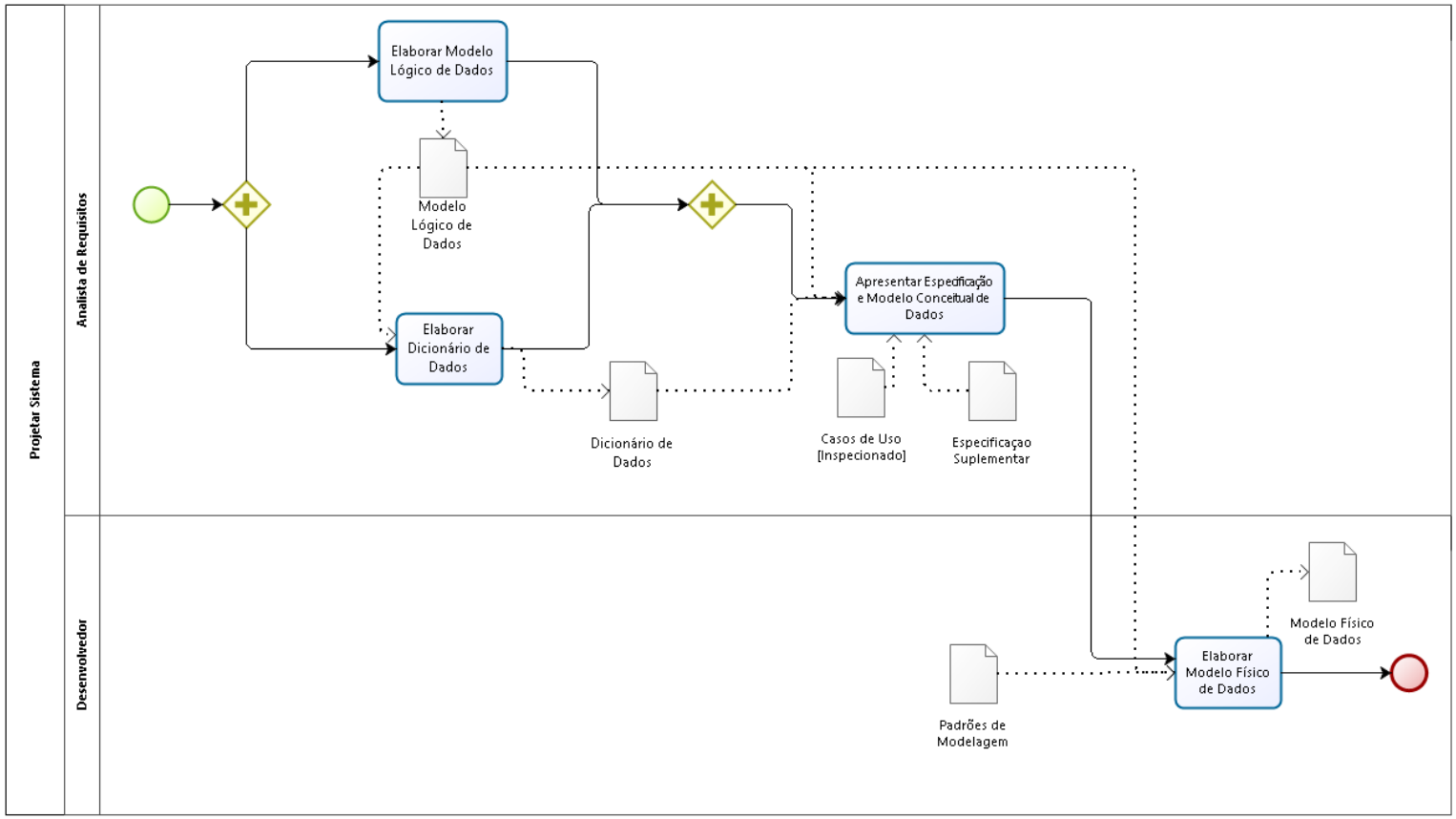
### *Especificar Requisitos Não Funcionais*

Esta atividade tem como objetivo especificar os requisitos não-funcionais do sistema, com base na visão geral do sistema e informações fornecidas pelos stakeholders.

### *Desenvolver Modelo de Dados Conceitual*

Desenvolvimento do modelo de dados em nível conceitual ou de domínio

# 2 2.3] Projetar Sistema




Powered by bizagi Modeler

**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

# Projetar Sistema

Processo onde o projeto do sistema é realizado. Aqui são elaboradas as soluções em termos de arquitetura, modelos de dados, interface com o usuário e com outros sistemas/dispositivos para atender aos requisitos funcionais e não funcionais pré-estabecidos.

## Elementos do processo

☐ *Elaborar Modelo Lógico de Dados*

☐ *Apresentar Especificação e Modelo Conceitual de Dados*

Consiste em realizar uma reunião com a equipe de desenvolvimento para apresentar os artefatos gerados durante a atividade de Especificação. Devem ser apresentados:

- Documento de Visão
- Casos de Uso
- Validação/Atualização do Modelo Lógico de Dados com a equipe de desenvolvimento
- Especificação Suplementar
- Modelo de Dados conceitual
- Documento de Regras de Negócio

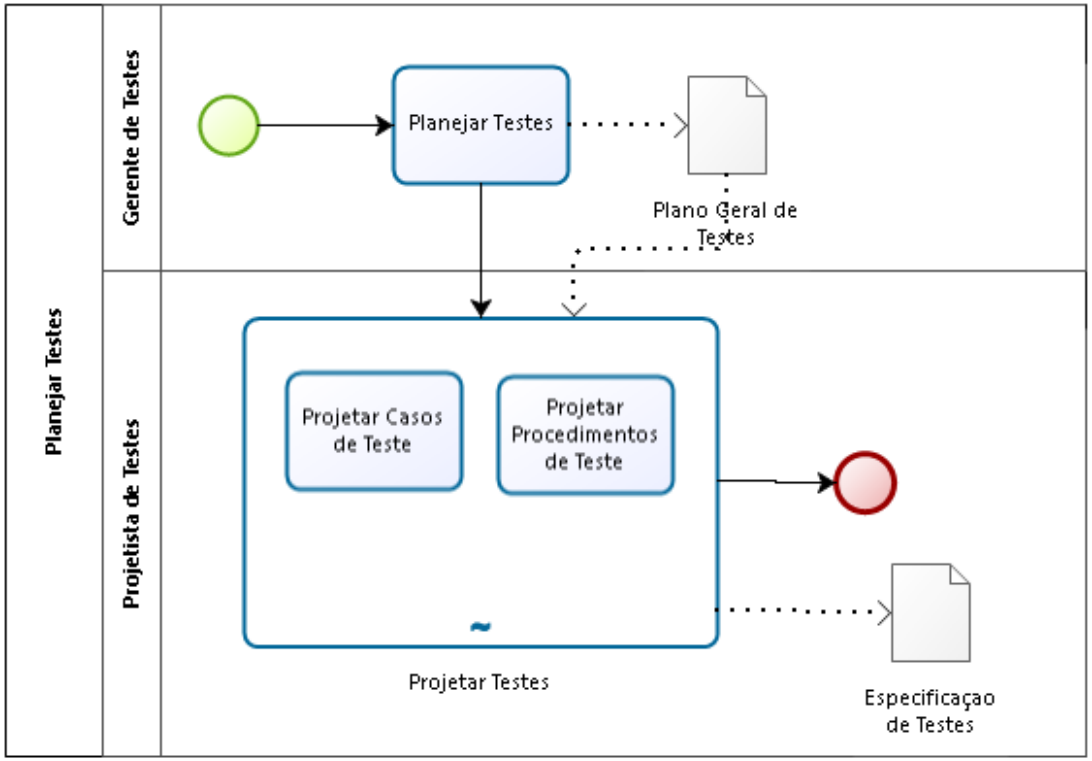
☐ *Elaborar Modelo Físico de Dados*

Nessa atividade deve ser desenvolvido o Modelo de Dados no nível físico, ou seja, no nível de implementação. Essa atividade pode ser executada de duas formas:

a) Tendo como base o Modelo de Dados conceitual gerado pelo Analista de Requisitos: nesse caso, os programadores deverão gerar o modelo de dados físico com base no modelo conceitual

b) Tendo como base somente as especificações: nesse caso o Analista deverá participar de forma mais ativa a fim de apoiar a elaboração da parte conceitual do modelo (entidades, chaves primárias, relacionamentos), para que os programadores possam ter subsídios para gerar o modelo de dados físico.

# 3 2.4] Planejar Testes




Powered by bizagi Modeler

**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

# Planejar Testes

## Elementos do processo

*Planejar Testes*

Essa atividade visa elaborar o Plano Geral de Testes com o objetivo de mostrar como o projeto está organizado para garantir a qualidade dos produtos entregues.

*Projetar Testes*

Gerar a Especificação de Testes contendo os casos de procedimentos de testes de cada item de teste definido no Plano de Testes

# Projetar Testes


Powered by bizagi Modeler

## Elementos do processo

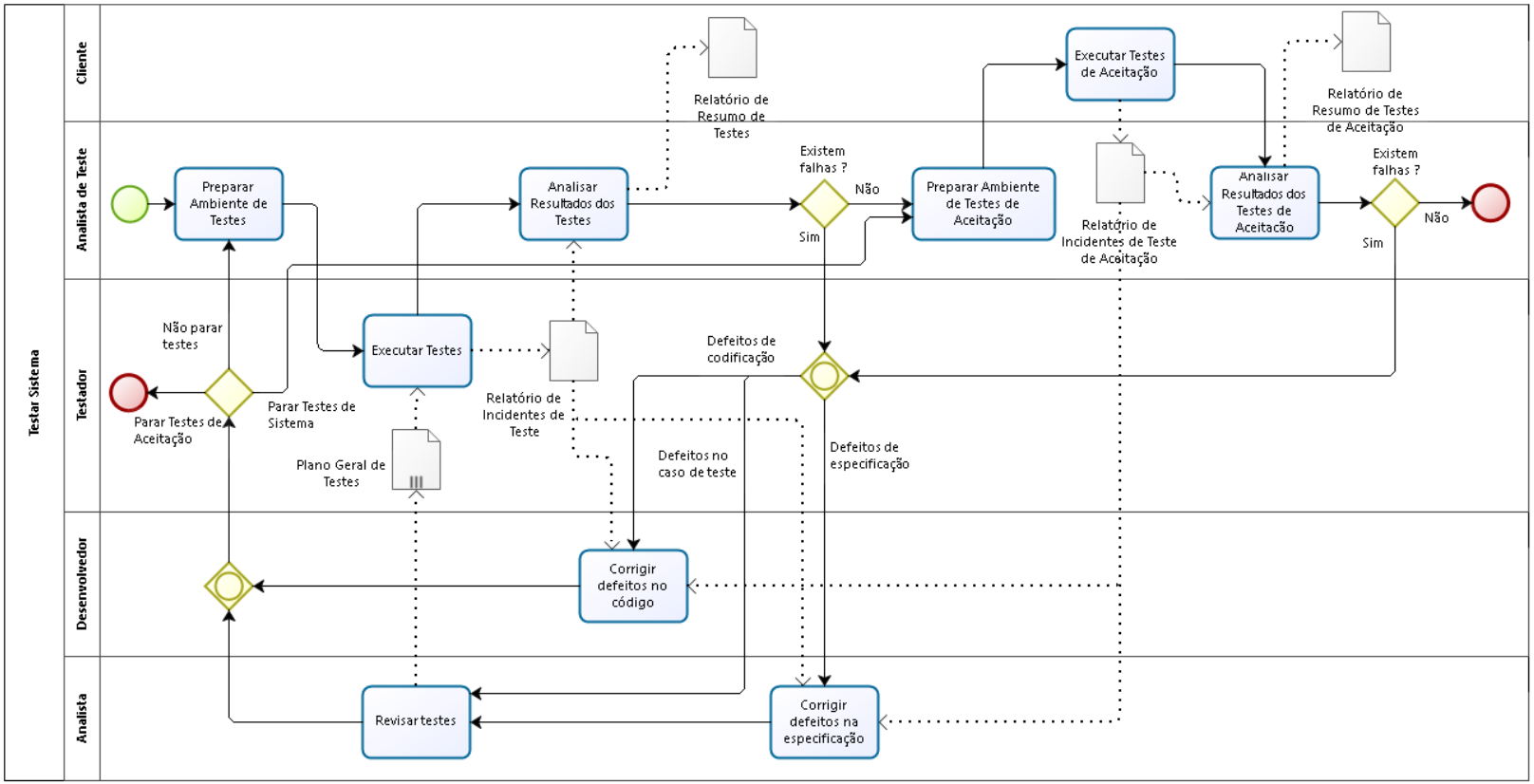
*Projetar Procedimentos de Teste*

Planejar os procedimentos de teste para os itens de teste definidos no Plano Geral de Testes. Os procedimentos deverão ser registrados no documento de Especificação de Testes.

*Projetar Casos de Teste*

Planejar os casos de teste para os itens de teste definidos no Plano Geral de Testes. Os casos de teste deverão ser registrados no documento de Especificação de Testes.

# 4 2.5] Testar Sistema




Powered by bizagi Modeler

**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

# Testar Sistema

Esse processo define as atividades relacionadas aos testes funcionais do sistema (realizado pela equipe de desenvolvimento) e também aos testes de aceitação (realizado por um grupo de usuários previamente definidos no planejamento dos testes)

## Elementos do processo

### *Preparar Ambiente de Testes*

Preparação do ambiente de teste em termos de hardware e software. Após essa preparação, o ambiente deve estar totalmente pronto para início da execução dos testes. São atividades típicas:

- Configuração de equipamentos (servidores web, servidores de e-mail e clientes)

- Instalação e configuração de softwares, bibliotecas e plug-ins de apoio

- Instalação e configuração da versão a ser testada

- Execução de scripts de criação do banco de dados

- Execução de scripts de inicialização do banco de dados (por exemplo, dados para casos de teste, usuários e perfis)

### *Executar Testes*

Execução dos casos e procedimentos de teste, conforme definido no plano de testes.

As falhas devem ser registradas no Relatório de Incidentes de Teste, bem como a captura das telas do sistema.

Além disso, todos os eventos da execução devem ser registrados no Log de Testes.

### *Analisar Resultados dos Testes*

Avaliar os resultados relatados nos Relatórios de Incidentes de Teste. Um resumo do processo de execução e dos incidentes identificados devem ser relatados no Relatório de Resumo de Testes.

Caso existam falhas a serem corrigidas, deve ser decidido se, após a correção dessas falhas, os casos/procedimentos de teste associados ao item que falhou deverão ser reexecutados.

Essa decisão deve estar baseada na criticidade do item e na gravidade da falha. Além disso, a falha deve ser analisada para definir se existe a necessidade de acerto na especificação de requisitos e nos casos/procedimentos de teste.

### *Preparar Ambiente de Testes de Aceitação*

Preparação do ambiente de testes de aceitação em termos de hardware e software.

### *Executar Testes de Aceitação*

Execução dos testes de aceitação pelo grupo de usuários definido no plano de testes.

As falhas devem ser registradas no Relatório de Incidentes de Testes de Aceitação, bem como a captura das telas do sistema.

☐ *Analisar Resultados dos Testes de Aceitação*

Avaliar os resultados relatados nos Relatórios de Incidentes de Testes de Aceitação. Um resumo do processo de execução e dos incidentes identificados devem ser relatados no Relatório de Resumo de Testes de Aceitação. Caso existam falhas a serem corrigidas, deve ser decidido se, após a correção dessas falhas, os testes de aceitação deverão ser reexecutados.

Essa decisão deve estar baseada na criticidade do item e na gravidade da falha juntamente com o usuário testador. Além disso, a falha deve ser analisada para definir se existe a necessidade de acerto na especificação de requisitos e nos casos/procedimentos de teste.

☐ Corrigir defeitos no código

O desenvolvedor designado para correção deve avaliar a falha relatada e corrigir o defeito.

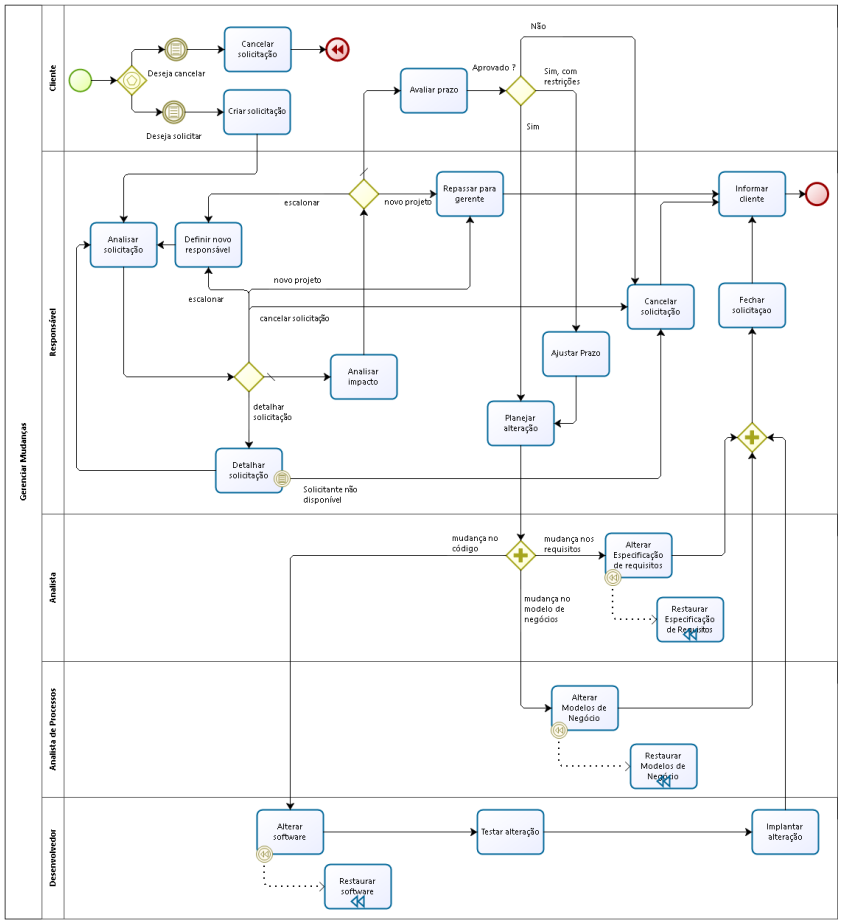
☐ *Corrigir defeitos na especificação*

Nessa atividade, a especificação de requisitos deve ser corrigida em função da análise da falha apontada na execução dos testes.

☐ *Revisar testes*

Nessa atividade, os casos e procedimentos de teste devem ser revisados em função de alterações na especificação de requisitos.

# 5 3] Gerenciar Mudanças



**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

# Gerenciar Mudanças

## Elementos do processo

### *Cancelar solicitação*

Nessa atividade o cliente cancela uma solicitação prévia, antes da sua conclusão.

### *Criar solicitação*

Nessa atividade o cliente cria uma nova solicitação e descreve as alterações desejadas. Isso pode ser feito através de algum documento ou planilha previamente definida ou através de um sistema de controle de solicitações.

### *Analisar solicitação*

O Analista analisa a solicitação:

- A solicitação possui algum problema que impede de continuá-la (dados inconsistentes, solicitação já existe, falta de recursos da área solicitante, etc.)

- O analista entende o que é a solicitação e avalia que será necessário tratá-la como um novo projeto devido ao seu porte.

- O analista não entende o que é a solicitação e necessita de maiores detalhes para avaliar o impacto

- O analista entende que o tratamento da solicitação deve ser feito por outra esfera de decisão

- O analista entende a solicitação e já tem os detalhes necessários para avaliar o impacto

### *Analisar impacto*

Com base nos dados da solicitação é feita uma estimativa dos impactos: prazo, alteração no modelo de negócios, funcionalidades, etc.

Essa análise de impacto pode levar a uma das seguintes situações:

- A solicitação possui algum problema que impede de continuá-la

- O analista entende que será necessário tratá-la como um novo projeto devido ao seu porte.

- O analista entende que o tratamento da solicitação deve ser feito por outra esfera de decisão.

- O analista consegue avaliar o impacto. Nesse caso, esse impacto é registrado.

### *Avaliar prazo*

O solicitante avalia o prazo definido e:

- Não aprova

- Aprova com o prazo fornecido

- Renegocia o prazo com o analista/gerente e aprova com restrições

### *Cancelar solicitação*

O Analista cancela a solicitação informando o motivo.

☐ *Informar cliente*

O Analista informa ao cliente sobre o término da solicitação.

☐ *Planejar alteração*

Planejar a data de entrega e os recursos que estarão envolvidos. Define, baseado na análise de impacto, as atividades necessárias: alterações no modelo de processos, alterações na especificação de requisitos (casos de uso, modelo de dados, interface com o usuário, interface com outros sistemas, etc.) e alterações na aplicação, bem como os responsáveis por essas alterações. Todos esses dados devem ser registados na solicitação.

☐ *Alterar Modelos de Negócio*

O Analista de processos realiza as atividades de elicitação e altera os modelos de processos para refletir a alteração. Podem ser realizadas atividades de inspeção nos modelos, se for pertinente.

☐ Alterar Especificação de requisitos

O Analista realiza as atividades de elicitação, se necessário, e altera os documentos de especificação de requisitos para refletir a alteração. Podem ser realizadas atividades de inspeção no documento de requisitos, se for pertinente.

☐ *Alterar software*

O Desenvolvedor altera o software para refletir as alterações solicitadas.

☐ *Testar alteração*

O Desenvolvedor realiza o teste a partir das alterações realizadas

☐ *Implantar alteração*

O Desenvolvedor implanta ou dispara os processos internos para implantação das alterações.

☐ *Ajustar Prazo*

O Analista/Gerente altera o prazo da solicitação conforme acordado com o Solicitante

☐ *Repassar para gerente*

Ao analisar o impacto da solicitação, o Analista percebe que se trata de uma alteração de grande porte e que, por isso, deve ser tratada como um novo projeto.

A solicitação é repassada para o gerente responsável pelos projetos de TI e finalizada.

☐ *Definir novo responsável*

Caso o responsável não tenha os subsídios para decidir ou está super alocado, ele repassa a Solicitação para o nível superior (por exemplo, o Desenvolvedor, repassa para o Analista)

☐ *Detalhar solicitação*

O analista entra em contato com o solicitante para detalhar a solicitação.

Esse contato pode ser feito pessoalmente (reunião, entrevista) ou por telefone, e-mail ou outros meios.

Os detalhes coletados pelo analista devem ser realimentados como parte da solicitação. Caso o solicitante não esteja disponível para conclusão dessa tarefa, a solicitação é cancelada.

☐*Restaurar Modelos de Negócio*

As alterações são abandonadas e os modelos de processos são restaurados com a última versão.

☐Restaurar Especificação de Requisitos

As alterações são abandonadas e os documentos de especificação são restaurados com a última versão.

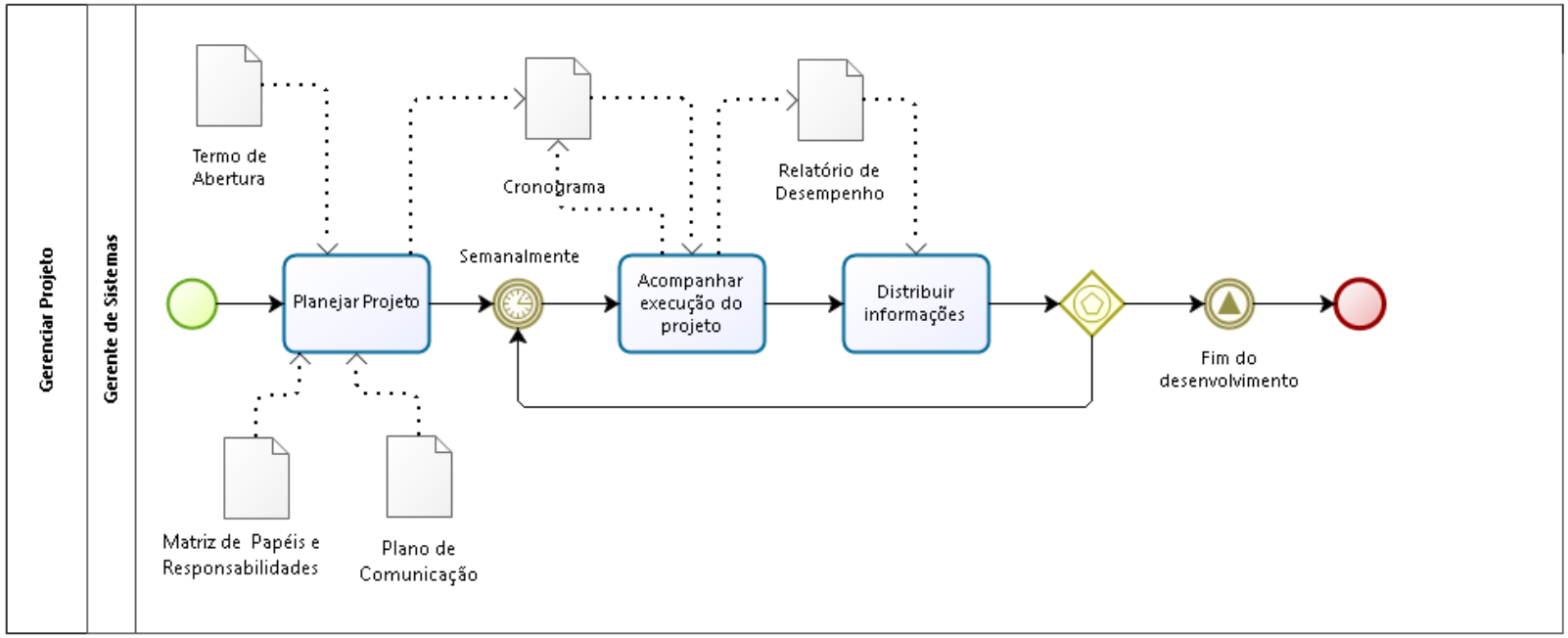
☐*Restaurar software*

As alterações são abandonadas e o código é restaurado com a última versão.

☐*Fechar solicitação*

O Analista fecha a solicitação.

# 6 4] Gerenciar Projeto




Powered by bizagi Modeler

**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

# Gerenciar Projeto

Processo que descreve o gerenciamento do projeto, englobando a elaboração do cronograma e o acompanhamento periódico das atividades planejadas.

## Elementos do processo

### ☐ *Planejar Projeto*

Nesta atividade é realizado o Planejamento do projeto com base no Termo de Abertura, definido durante a reunião de kick-off, na Matriz de Papéis e Responsabilidade e no Plano de Comunicação.

Como produto é gerado o cronograma do projeto.

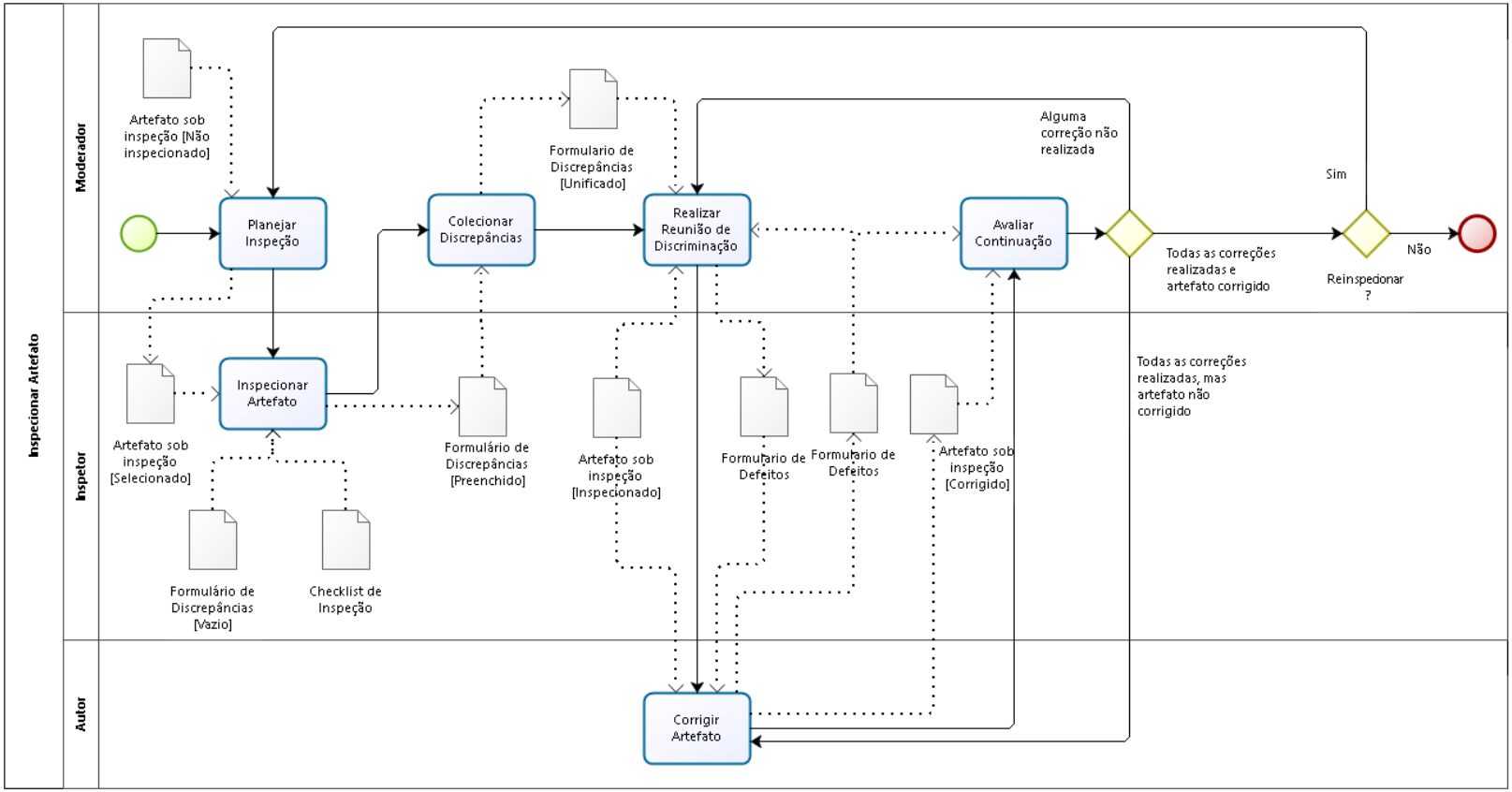
### ☐ *Acompanhar execução do projeto*

Esta é uma atividade que deve acontecer periodicamente.  Seu objetivo é: acompanhar as atividades definidas no cronograma do projeto, verificar se prazos e orçamento, bem como recursos (materiais e humanos) estão dentro do planejado e gerenciar riscos. Caso haja necessidade específica do projeto. De Matriz de Papéis e Responsabilidade e do Plano de Comunicação podem ser atualizados.

### ☐ *Distribuir informações*

Distribuir as informações sobre o andamento do projeto para os stakeholders. E, caso haja necessidade específica do projeto. de Matriz de Papéis e Responsabilidade e do Plano de Comunicação podem ser atualizados.

# 7 Inspecionar Artefato




Powered by bizagi Modeler

**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

# Inspecionar Artefato

Processo responsável por avaliar a qualidade dos artefatos produzidos.

Atualmente esse processo é usado para:

- Verificação da qualidade dos modelos BPMN
- Verificação da qualidade dos casos de uso

## Elementos do processo

### *Planejar Inspeção*

O processo de inspeção é planejado com a realização das seguintes tarefas:

- Definição dos artefatos a serem inspecionados
- Definição dos inspetores
- Atribuição dos artefatos aos inspetores
- Definição do cronograma da inspeção
- Distribuição do material aos inspetores

### *Inspecionar Artefato*

É realizada a inspeção propriamente dita. As discrepâncias são registradas no formulário de discrepâncias e, ao final, o formulário é enviado para o moderador.

### *Colecionar Discrepâncias*

Os diversos formulários de discrepância são reunidos em um só formulário, que é ordenado pela localização da discrepância no artefato.

### *Realizar Reunião de Discriminação*

Nessa reunião as discrepâncias registradas pelos inspetores são classificadas como defeitos ou falso positivos.

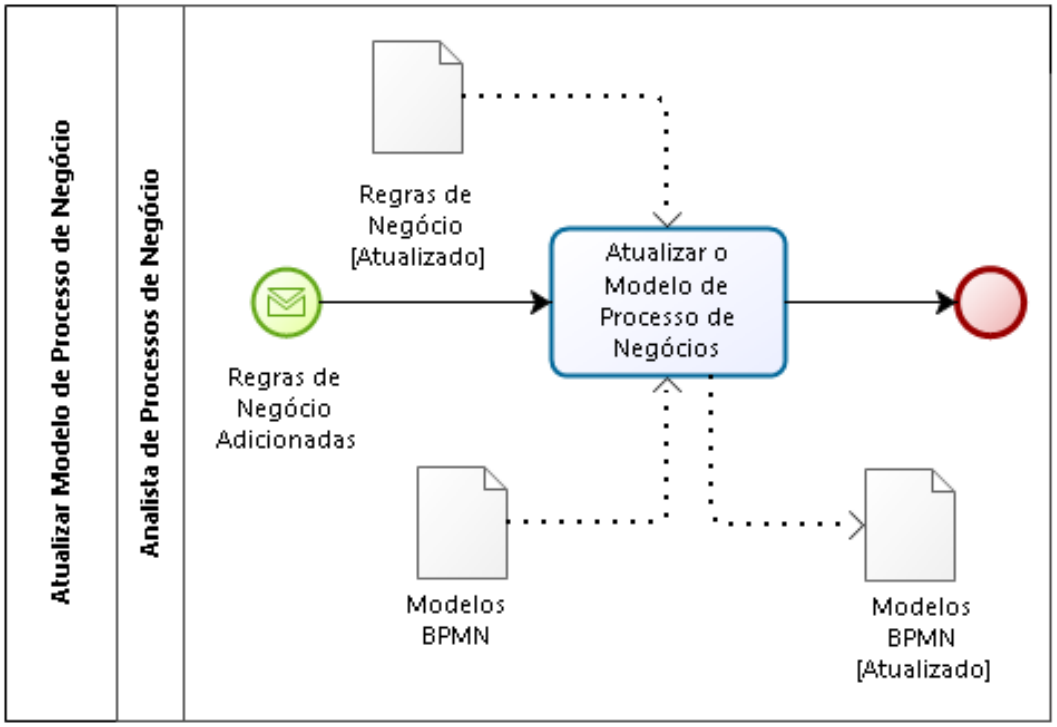
### *Corrigir Artefato*

### *Avaliar Continuação*

Devem ser realizadas as seguintes tarefas:

- Avaliar se todos os defeitos registrados na planilha estão marados como corrigidos
- Avaliar se realmente os defeitos foram corrigidos no artefato
- Decidir se o artefato será reinspecionado

# 8 Atualizar Modelo de Processo de Negócio




Powered by bizagi Modeler

**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

# Atualizar Modelo de Processo de Negócio

## Elementos do processo

☐ *Atualizar o Modelo de Processo de Negócios*

**Descrição**

A atualização dos modelos BPMN, em função das novas regras identificadas

# Resources

## Gerente de Sistemas (Função)

Responsável por atender as demandas ao setor de desenvolvimento e gerenciar a execução dos projetos.

Perfil: Profissional com conhecimento de práticas de gerenciamento de projetos de software, soluções tecnológicas para sistemas de informação e capacidade de tomada de decisão.

## Analista de Requisitos (Função)

Responsável por identificar e especificar os requisitos do sistema em desenvolvimento.

Perfil: Profissional com habilidades em elicitação de requisitos a partir de diferentes fontes (pessoas e documentações técnicas), análise de requisitos funcionais e não funcionais, especificação de software sob perspectivas de dados e função. É necessário conhecimento em técnicas como entrevistas, para elicitação de requisitos, e técnicas de especificação como modelagem de casos de uso e diagramas ER/UML.

## Projetista de Sistemas (Função)

Responsável por desenhar a solução técnica do sistema com base nos requisitos identificados.

Perfil: Profissional com habilidades de transformar os requisitos em soluções de software por meio de abstrações, utilizando modelos arquiteturais e conceitos de coesão e acoplamento, bem como sólido conhecimento das tecnologias envolvidas na solução técnica (frameworks, bibliotecas, e linguagens de programação).

É importante para o arquiteto ter conhecimento de estilos e padrões arquiteturas, linguagens e notações para especificação de software como modelos ER/UML em diferentes níveis de abstração, bem como o ferramental de apoio às atividades de projeto.

## Analista de Processos (Função)

Responsável por identificar e modelar os processos de negócio que fazem parte do escopo do projeto.

Perfil: Profissional com habilidades em elicitação de processos de negócio a partir de diferentes fontes (pessoas e documentações técnicas), análise e especificação de processos de negócio, reconhecendo atividades e suas entradas e saídas, bem como um encadeamento lógico entre elas. Ainda, é necessário que o analista de processos entenda que existem regras associadas à execução dos processos precisam também ser capturadas e especificadas. É necessário conhecimento em técnicas como entrevistas, para elicitação de processos, e técnicas de especificação como modelagem em BPMN.

## Programador (Função)

Responsável por construir o código do sistema com base nas especificações. Também denominado Desenvolvedor.

Perfil: Profissional com habilidades de abstração e desenvolvimento de algoritmos nas linguagens de programação adotadas pela equipe de desenvolvimento. É necessário ainda que o programador tenha conhecimento para manipulação de banco de dados, ferramentas de desenvolvimento como IDE, frameworks e bibliotecas.

## Testador (Função)

Responsável por realizar as atividades de planejamento, projeto e execução de testes do sistema.

Perfil: Profissional com conhecimento de técnicas para elaboração de procedimentos e casos de teste, além de instrumentação de testes utilizando as tecnologias relacionadas ao desenvolvimento dos sistemas a serem testados.

## Moderador (Função)

Responsável por coordenar o processo de inspeção

## Autor (Função)

Autor do artefato sob inspeção

## Inspetor (Função)

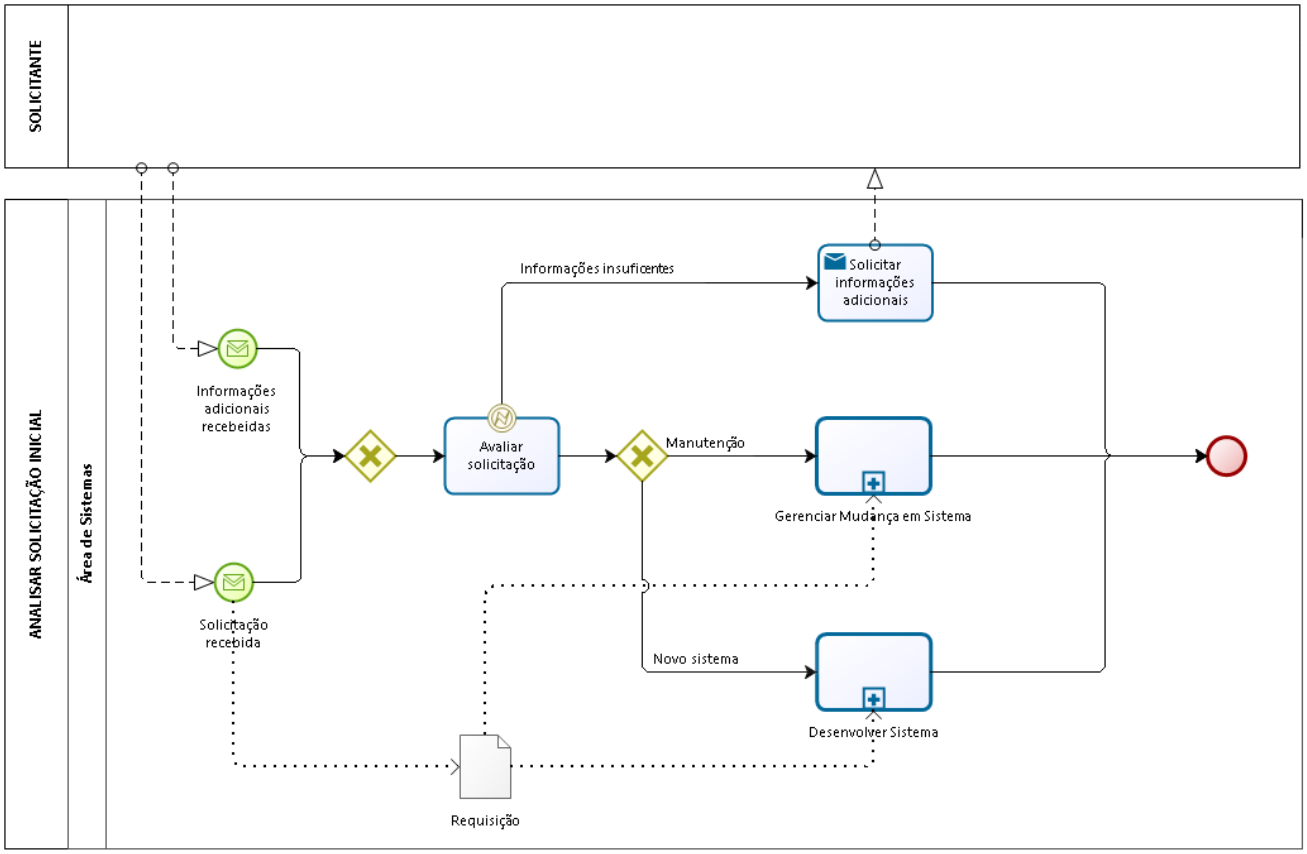
Responsável por realizar a inspeção propriamente dita no artefato. A alocação ao papel de inspetor deve respeitar a premissa de que o mesmo não seja autor do artefato.

Perfil: Profissional com conhecimento sobre os artefatos a serem inspecionados/revisados, bem como sobre os procedimentos de construção e características de qualidade de tais artefatos.

## Desenvolvedor (Função)

Este termo é designado a qualquer membro da equipe de desenvolvimento responsável por produzir artefatos que levam ao produto final. Entretanto, este termo é comumente associado ao papel de Programador.

# Anexo III – MDS Fiocruz ajustada pelas recomendações da avaliação



# 1 - MDS

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

A MDS inicia-se a partir de uma solicitação do cliente da área de Tecnologia da Informação. Com base na análise desta solicitação é decidido se a mesma será atendida como uma manutenção no sistema, ou como um projeto de desenvolvimento de novo sistema.

# ANALISAR SOLICITAÇÃO INICIAL

## Elementos do processo

### *Desenvolver Sistema*

O desenvolvimento de sistemas é realizado em quatro fases fundamentais inspiradas no modelo de referência Unified Process, que são Inciar desenvolvimento (Inception), Elaborar software (Elaboration), Construir software (Construction), e Implantar software (Transition). Adicionalmente, os processos de Gestão de Projetos são integrados aos processos de Engenharia de Software, exatamente como recomendado em modelos inspirados no UP, como o OpenUP.

### Gerenciar Mudança em Sistema

As solicitações de manutenção são tratadas no processo "Gerenciar Mudança". As solicitações de manutenção corretiva não passam por maiores análises, e são tratadas mais rapidamente. Os demais tipos de manutenção passam por uma aprovação do custo e do prazo pelo solicitante, bem como por uma análise de impacto para verificar a partir de que ponto (processo, requisito, código) os ajustes devem ser implementados.

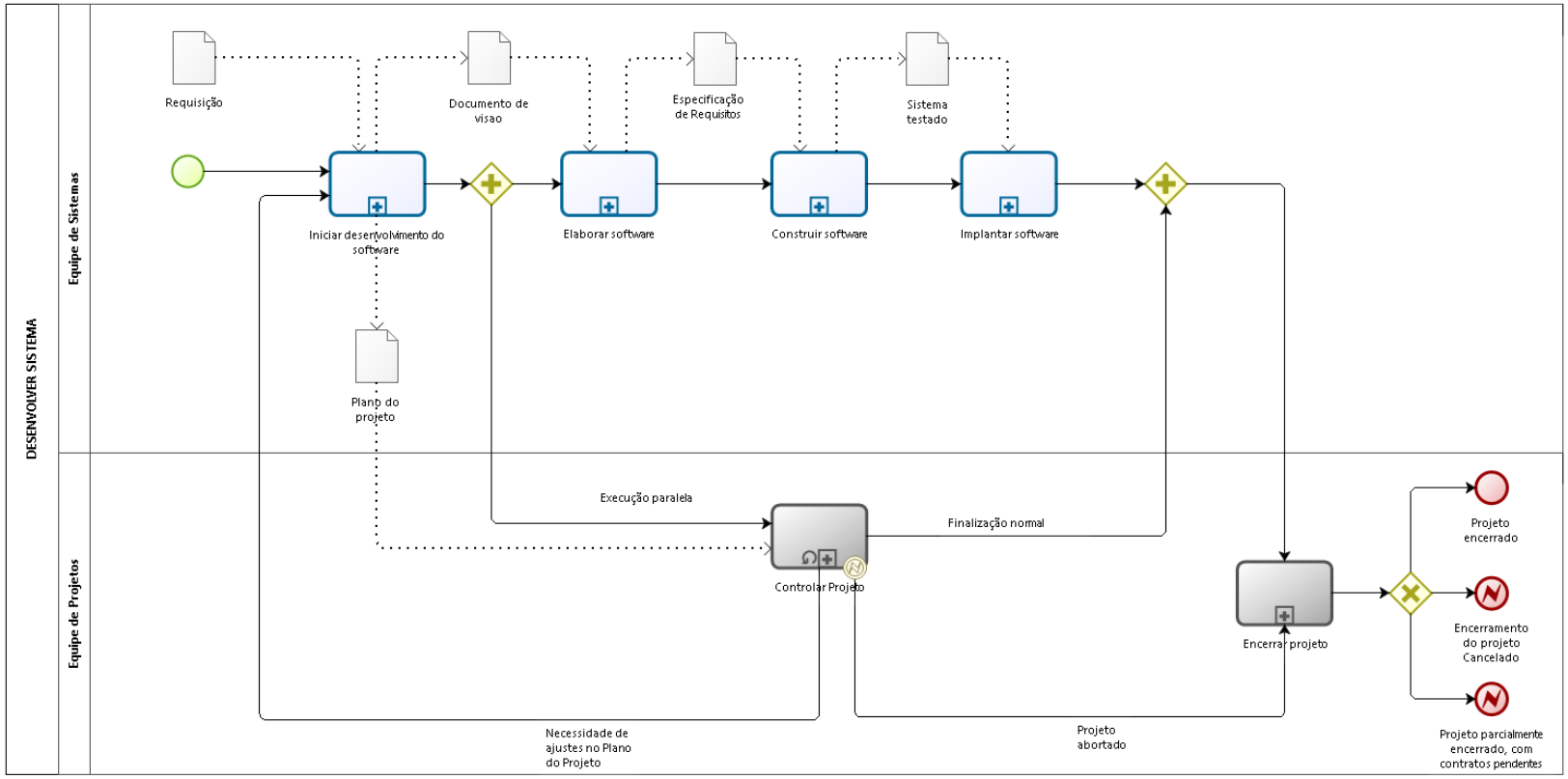
### Avaliar solicitação

Avaliar se a solicitação se refere a um pedido de manutenção (corretiva, preventiva, adaptativa ou perfectiva), ou se é um pedido de um novo sistema de informação.

### *Solicitar informações adicionais*

Caso as informações do solicitante forem insuficientes ou confusas, mais detalhes serão requisitados, de modo a melhorar a qualidade da análise da solicitação.

# 9 2-Desenvolver



**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

O desenvolvimento de sistemas é realizado em quatro fases fundamentais inspiradas no modelo de referência Unified Process, que são Inciar desenvolvimento (Inception), Elaborar software (Elaboration), Construir software (Construction), e Implantar software (Transition). Adicionalmente, os processos de Gestão de Projetos são integrados aos processos de Engenharia de Software, exatamente como recomendado em modelos inspirados no UP, como o OpenUP.

# DESENVOLVER SISTEMA

## Elementos do processo

### *Iniciar desenvolvimento do software*

O início do desenvolvimentodo sistema integra atividades específicas de desenvolvimento e processos de gestão de projetos. O Termo de Abertura do Projeto (TAP) é o documento que dá uma ideia geral do projeto, inclusive em termos de custo e prazo, e este documento também autoriza o início e prosseguimento do projeto.

### *Elaborar software*

A elaboração do software se concentra na estruturação do sistema a ser desenvolvido, em uma cadeia de valor iniciada na modelagem dos processos de negócio, da elicitação dos requisitos a partir desse modelo de processos, e do projeto do sistema a partir da especificação dos requisitos.

### *Construir software*

A implementação do software tem como sua atividade central a codificação do sistema. No entanto, o processo se preocupa com a prática de reuso, com a avaliação (teste) do produto construído, e com a comunicação de eventuais decisões tomadas nesta fase.

### *Implantar software*

A implantação do software passa pela disponibilização do software em seu ambiente final, acessível aos usuários autorizados. Também, inclui a elaboração de material didático de apoio ao usuário, que poderá ser utilizado tanto pelo próprio usuário quanto em eventuais treinamentos proporcionados pela área de Tecnologia da Informação.

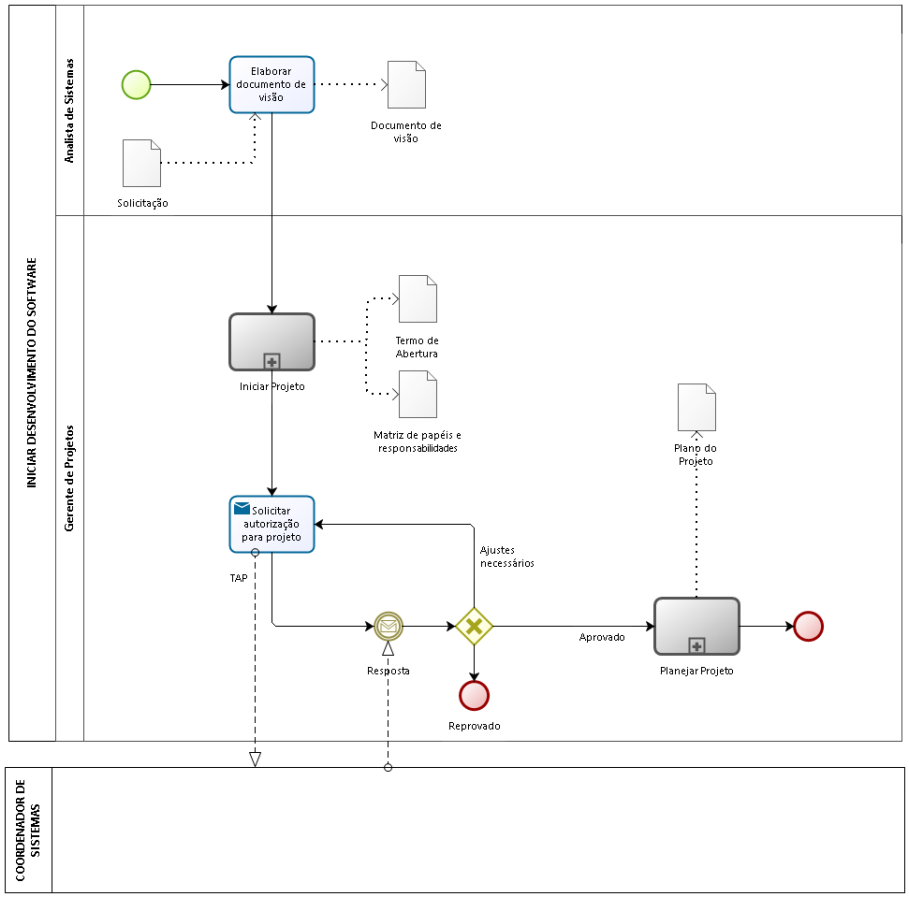
### Controlar Projeto

O controle do projeto envolve acompanhar, revisar e reportar o progresso do projeto, bem como ajustar os rumos do projeto durante sua execução. Este processo ocorre em ciclo semanal durante toda a duração do projeto.

### *Encerrar projeto*

O encerramento do projeto garante que o cliente do projeto concorda e aceita o encerramento, bem como garante a finalização oficial de todos os contratos e parcerias ligados a este projeto.

# 2.1-Iniciar


INICIAR DESENVOLVIMENTO DO SOFTWARE
Analista de Sistemas
Gerente de Projetos
COORDENADOR DE SISTEMAS
Elaborar documento de visão
Documento de visão
Solicitação
Iniciar Projeto
Termo de Abertura
Matriz de papéis e responsabilidades
Plano do Projeto
Solicitar autorização para projeto
TAP
Ajustes necessários
Resposta
Aprovado
Reprovado
Planejar Projeto

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

O início do desenvolvimento do sistema integra atividades específicas de desenvolvimento e processos de gestão de projetos. O Termo de Abertura do Projeto (TAP) é o documento que dá uma ideia geral do projeto, inclusive em termos de custo e prazo, e este documento também autoriza o início e prosseguimento do projeto.

# INICIAR DESENVOLVIMENTO DO SOFTWARE

## Elementos do processo

### *Iniciar Projeto*

O início do projeto é materializado no Termo de Abertura do Projeto (TAP), pela identificação e classificação dos stakeholders, e pela realização da reunião inicial do projeto, também chamada de reunião de kick-off.

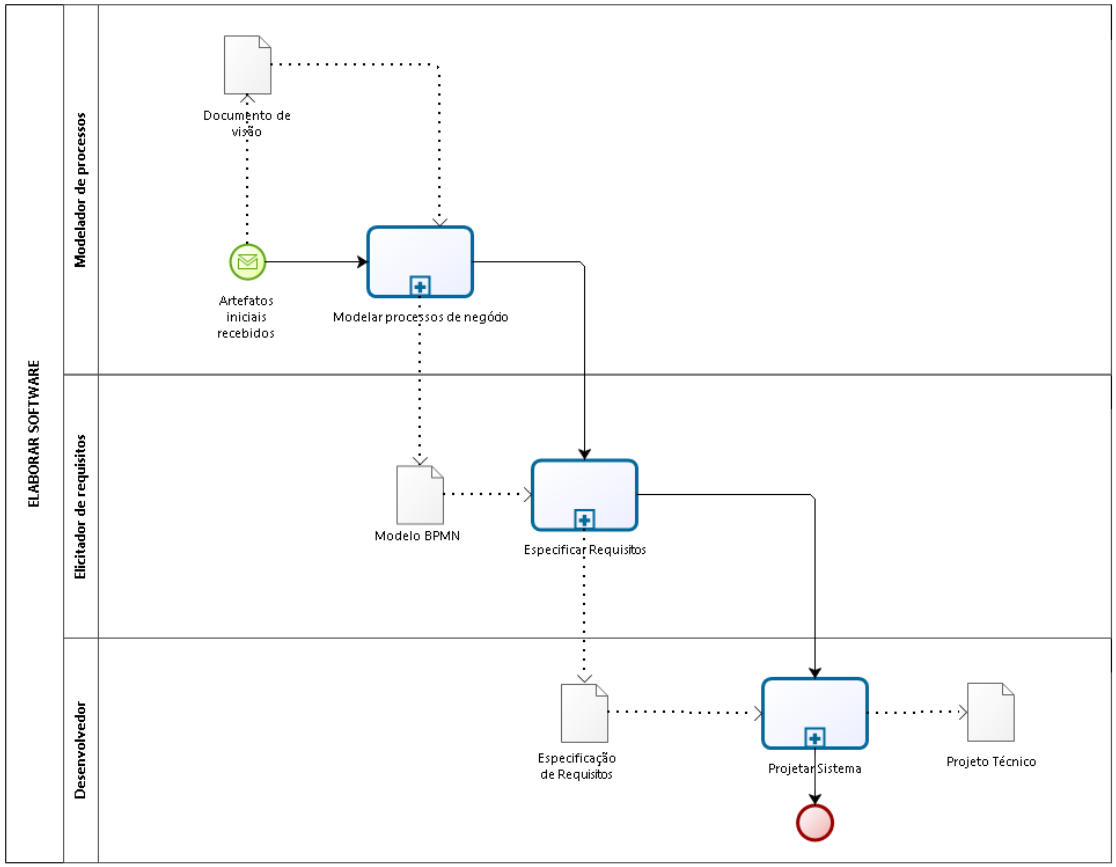
### *Solicitar autorização para projeto*

É solicitada a autorização para a o início do projeto de desenvolvimento de software. O TAP é o documento utilizado como base de decisão, e nele também é registrada a eventual aprovação para o início do projeto.

### *Planejar Projeto*

O planejamento do projeto de software baseia-se na construção de um cronograma, que será a base para o acompanhamento do projeto, bem como a elaboração de um plano de riscos e um plano de comunicação para o projeto.

# 2.2-Elaborar




Powered by bizagi Modeler

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

A elaboração do software se concentra na estruturação do sistema a ser desenvolvido, em uma cadeia de valor iniciada na modelagem dos processos de negócio, da elicitação dos requisitos a partir desse modelo de processos, e do projeto do sistema a partir da especificação dos requisitos.

# ELABORAR SOFTWARE

## Elementos do processo

### *Modelar processos de negócio*

Processo que define o fluxo de atividades necessárias para a elicitação, modelagem, verificação e validação dos processos de negócio a serem capturados pelo sistema em desenvolvimento. O conjunto de modelos de processos de negócio validado oferece uma visão geral das potenciais atividades a serem automatizadas pelo sistema.

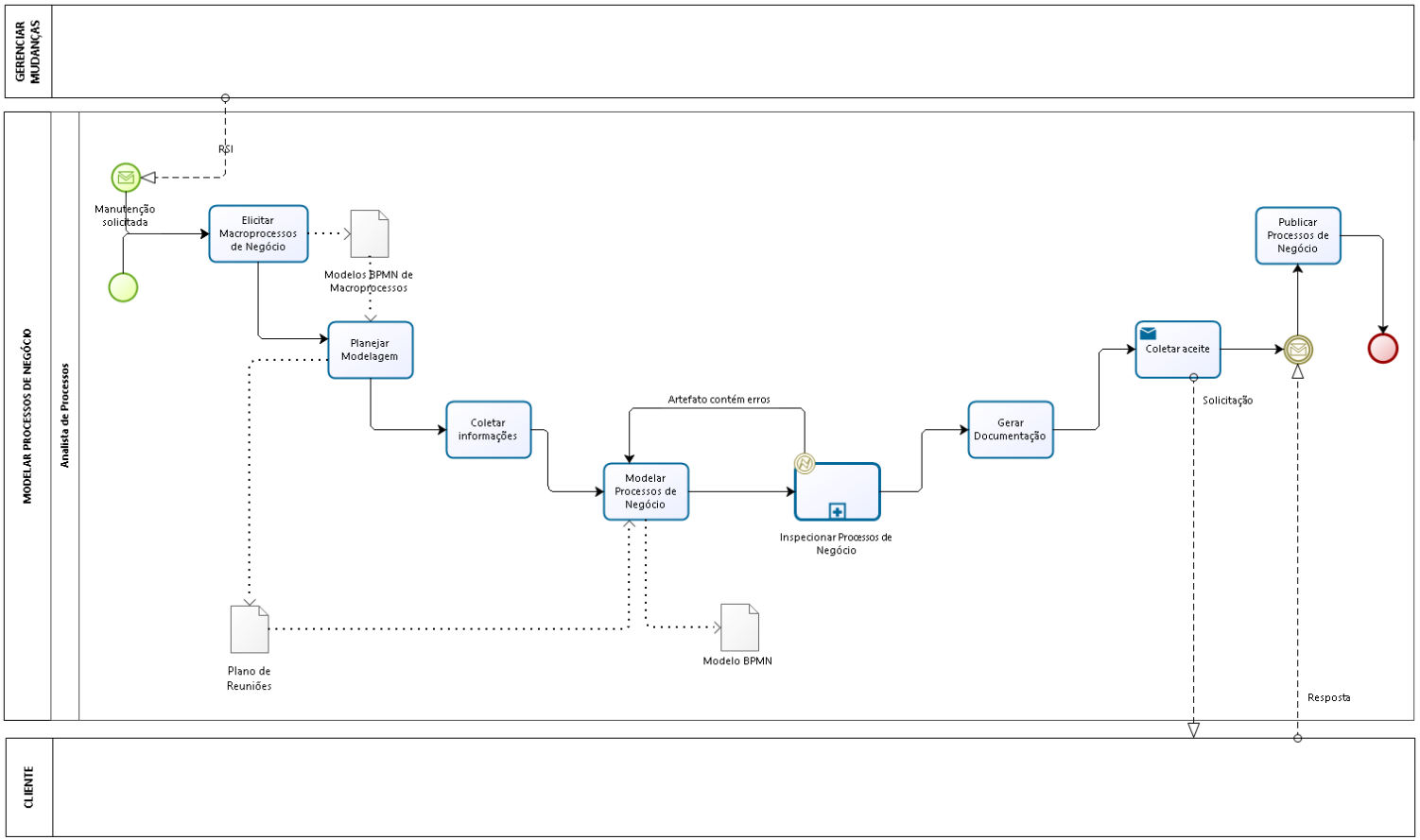
### *Especificar Requisitos*

Processo que tem como objetivo a elicitação, análise e especificação dos requisitos do sistema em desenvolvimento.

### *Projetar Sistema*

Processo responsável pela modelagem arquitetural e em nível de projeto do sistema em desenvolvimento, a partir dos requisitos elicitados na macroatividade de Especificação de Requisitos. Esta modelagem envolve tanto aspectos estruturais quanto comportamentais dos componentes do sistema.

## 2.2.1-Modelar processo



**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

Processo que define o fluxo de atividades necessárias para a elicitação, modelagem, verificação e validação dos processos de negócio a serem capturados pelo sistema em desenvolvimento. O conjunto de modelos de processos de negócio validado oferece uma visão geral das potenciais atividades a serem automatizadas pelo sistema.

# MODELAR PROCESSOS DE NEGÓCIO

Processo onde os processos e as regras de negócio são elicitados, modelados usando BPMN e inspecionados. Como resultado principal temos os Modelos BPMN que representam o escopo do negócio onde o sistema está inserido.

## Elementos do processo

### *Elicitar Macroprocessos de Negócio*

Nesta atividade são definidos, por meio de uma ou mais reuniões, uma lista com os macroprocessos de negócio que definem o escopo do negócio onde o sistema a ser desenvolvido está inserido, juntamente com uma descrição associada a cada macroprocesso e, também é criada a estrutura de diretórios do projeto.

### *Planejar Modelagem*

Nesta atividade é definido como o processo de modelagem será executado, ou seja, se os processos de negócio serão elicitados, modelados, verificados e validados de forma completamente sequencial ou se os mesmos serão divididos em conjuntos menores (por macroprocesso, por unidade funcional, ou outras divisões), com o processo sendo executado de forma iterativa (em ciclos) para cada conjunto definido.

### *Coletar informações*

Para cada macroprocesso identificado anteriormente, seus subprocessos, responsáveis (papéis), atividades (e seus encadeamentos) e documentos (consumidos e produzidos) são identificados. São realizadas entrevistas ou reuniões com os stakeholders relacionados a estes subprocessos.

### *Modelar Processos de Negócio*

Conforme os processos de negócio vão sendo entendidos pelos Analistas de Negócio, os mesmos são modelados utilizando a notação BPMN na ferramenta BizAgi.

### *Publicar Processos de Negócio*

Após a validação dos modelos de processos de negócio com a aprovação pela Equipe de Validação, é necessário tornar estes modelos públicos por meio de uma versão web e notificar todos interessados.

### *Inspecionar Processos de Negócio*

Nesta atividade, os modelos de processos de negócio são verificados com base em um checklist que captura heurísticas de boa construção de modelos. A atividade de verificação é interna ao projeto e não envolve o cliente. Caso o artefato analisado (especificação) contenha erros, o processo é interrompido e retorna para a a correção dos erros identificados.

### *Gerar Documentação*

**Descrição**

A partir dos modelos BPMN devem ser gerados dois documentos (usando as funcionalidades do Bizagi):

- Documento completo com os todos os modelos de processos de negócio

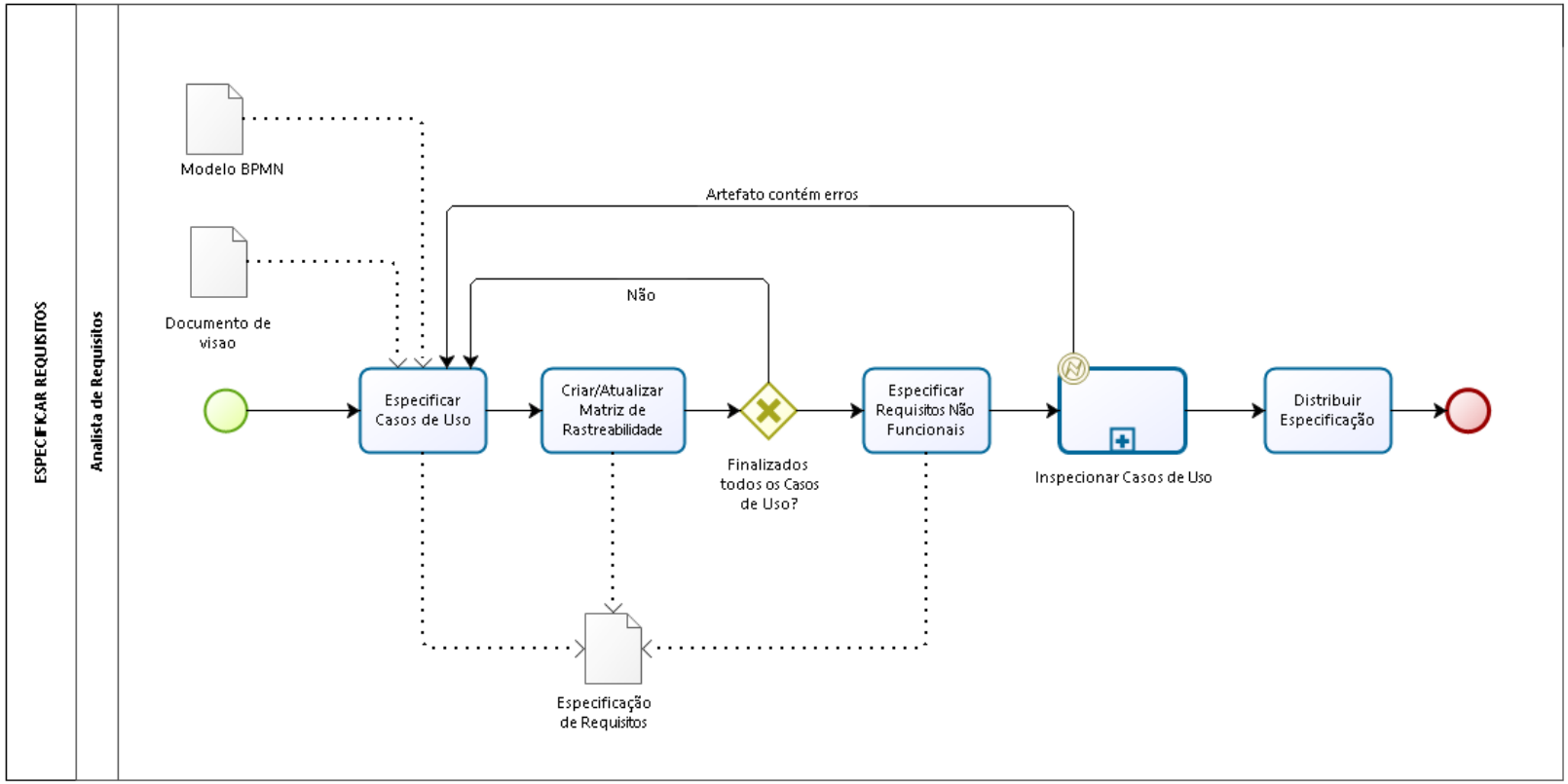
- Documento reduzido contendo apenas as regras de negócio do modelo

 *Coletar aceite*

São coletadas as assinaturas dos principais envolvidos no próprio documento de modelos de processos de negócio gerado com o Bizagi

Web

## 2.2.2-Especificar Requisitos




Powered by bizagi Modeler

**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

Processo que tem como objetivo a elicitação, análise e especificação dos requisitos do sistema em desenvolvimento.

# ESPECIFICAR REQUISITOS

Processo onde é realizado a elicitação, a especificação e a inspeção dos requisitos funcionais e não funcionais do sistema. Como resultado principal temos o Documento de Especificação de Requisitos que será a base de todo o desenvolvimento.

## Elementos do processo

### *Especificar Casos de Uso*

Nesta atividade, os requisitos funcionais identificados e registrados no Documento de Visão são analisados e então descritos no formato de casos de uso e de processos não interativos.

Os Modelos BPMN também devem ser consultados para capturar a parte dos processos de negócio que estão sendo automatizadas, bem como as regras de negócio

### *Criar/Atualizar Matriz de Rastreabilidade*

Esta atividade está associada ou à criação ou atualização da Matriz de Rastreabilidade

### *Especificar Requisitos Não Funcionais*

Esta atividade tem como objetivo especificar os requisitos não-funcionais do sistema, com base na visão geral do sistema e informações fornecidas pelos stakeholders.

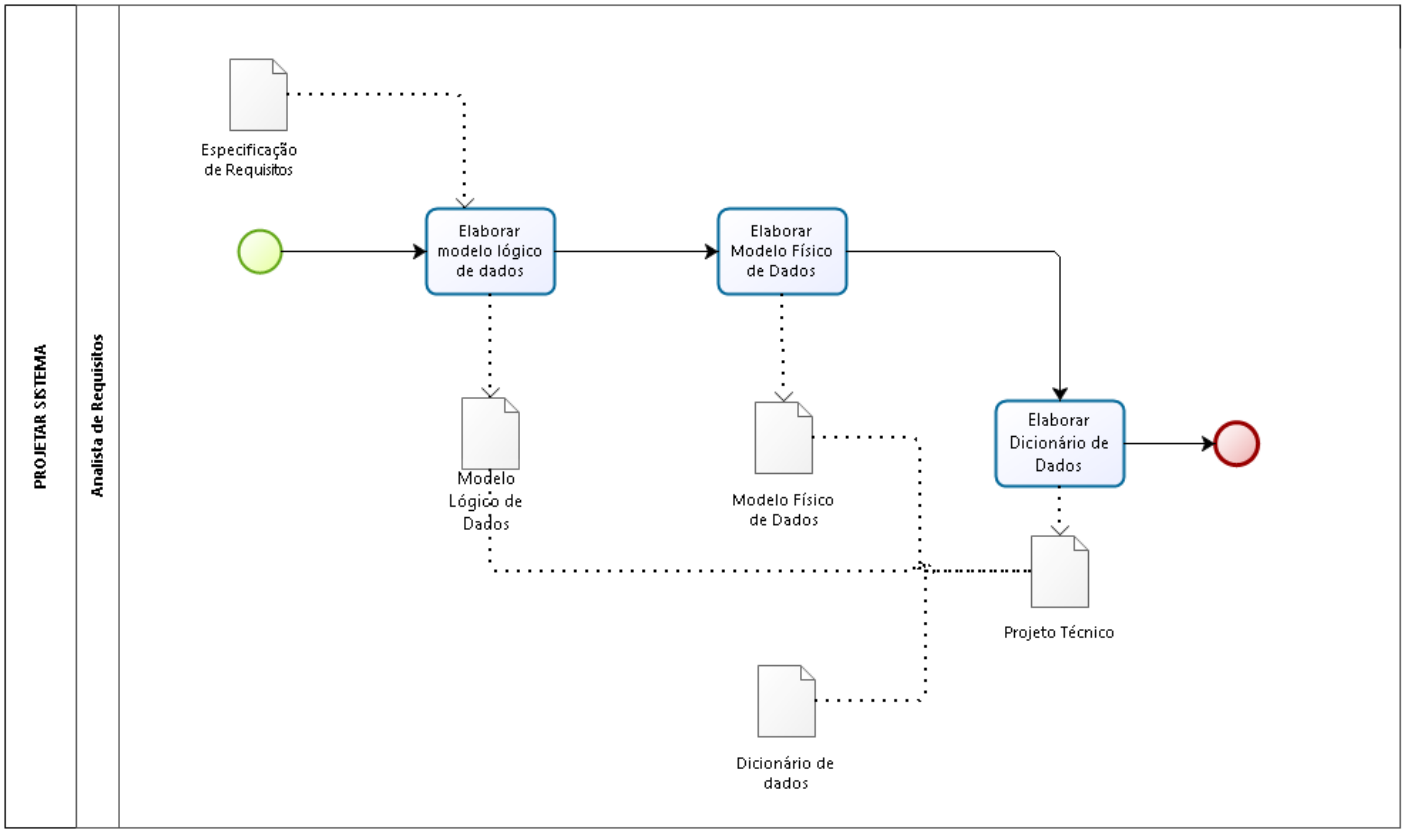
### *Inspecionar Casos de Uso*

Nesta atividade, os casos de uso são verificados com base em um checklist que define regras de boa construção. A atividade de verificação é interna ao projeto e não envolve o cliente. Caso o artefato analisado (especificação) contenha erros, o processo é interrompido e retorna para a correção dos erros identificados.

### Distribuir Especificação

Distribuição ou liberação dos documentos de especificação para a equipe de desenvolvimento. Quando for utilizado um software de controle de versão, é recomendado que essa liberação ocorra usando o mecanismo de TAG ou BRANCH do controle de versão e comunicando aos demais membros da equipe o diretório no qual os documentos liberados estão armazenados.

## 2.2.3-Projetar Sistema




Powered by bizagi Modeler

**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

Processo responsável pela modelagem arquitetural e em nível de projeto do sistema em desenvolvimento, a partir dos requisitos elicitados na macroatividade de Especificação de Requisitos. Esta modelagem envolve tanto aspectos estruturais quanto comportamentais dos componentes do sistema.

# PROJETAR SISTEMA

Processo onde o projeto do sistema é realizado. Aqui são elaboradas as soluções em termos de arquitetura, modelos de dados, interface com o usuário e com outros sistemas/dispositivos para atender aos requisitos funcionais e não funcionais pré-estabecidos.

## Elementos do processo

### ☐ *Elaborar Modelo Físico de Dados*

Nessa atividade deve ser desenvolvido o Modelo de Dados no nível físico, ou seja, no nível de implementação. Essa atividade pode ser executada de duas formas:

a) Tendo como base o Modelo de Dados conceitual gerado pelo Analista de Requisitos: nesse caso, os programadores deverão gerar o modelo de dados físico com base no modelo conceitual

b) Tendo como base somente as especificações: nesse caso o Analista deverá participar de forma mais ativa a fim de apoiar a elaboração da parte conceitual do modelo (entidades, chaves primárias, relacionamentos), para que os programadores possam ter subsídios para gerar o modelo de dados físico.

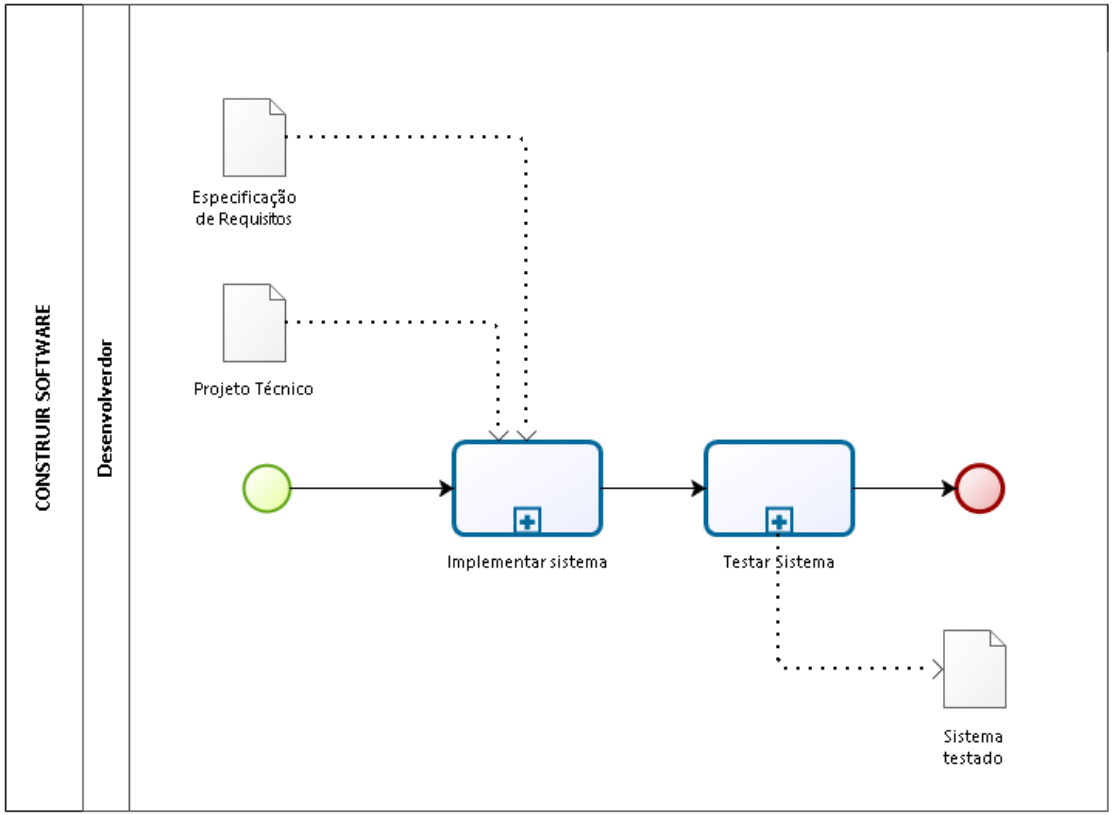
### ☐ *Elaborar Dicionário de Dados*

Esta atividade compreende a construção de um dicionário que descreva em detalhes os elementos que compõem o modelo lógico de dados e também o modelo físico de dados.

### ☐ *Elaborar modelo lógico de dados*

A elaboração do modelo lógico é realizada a partir do documento de visão do projeto, que define sua problemática e a indicação das soluções funcionais e não funcionais.

## 2.3-Construir




Powered by bizagi Modeler

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

A implementação do software tem como sua atividade central a codificação do sistema. No entanto, o processo se preocupa com a prática de reuso, com a avaliação (teste) do produto construído, e com a comunicação de eventuais decisões tomadas nesta fase.

# CONSTRUIR SOFTWARE

## Elementos do processo

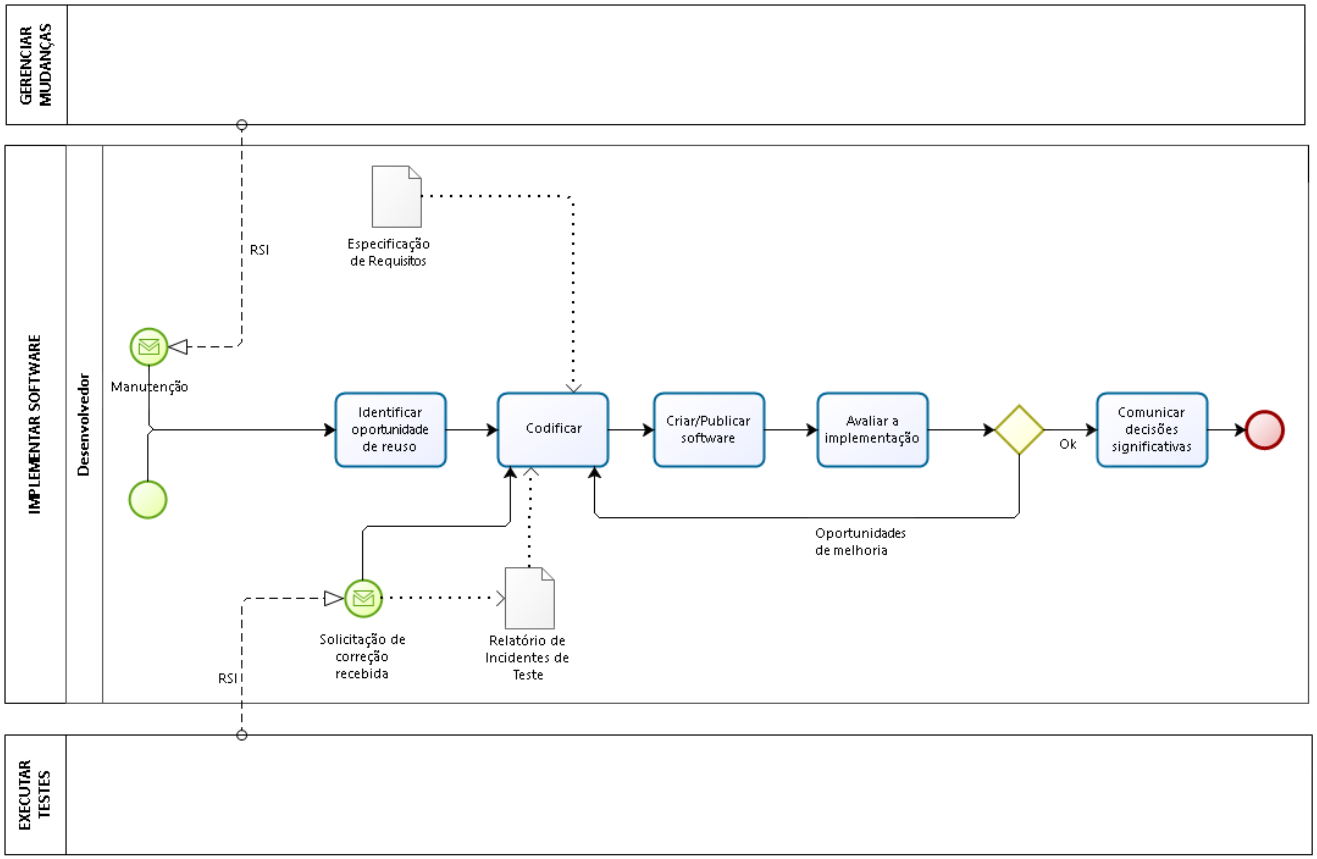
### *Implementar sistema*

A implementação do software tem como sua atividade central a codificação do sistema. No entanto, o processo se preocupa com a prática de reuso, com a avaliação (teste) do produto construído, e com a comunicação de eventuais decisões tomadas nesta fase.

### *Testar Sistema*

Processo que tem como objetivo executar os testes funcionais do sistema em desenvolvimento com base nas especificações de testes elaboradas previamente.

## 2.3.1-Implementar




Powered by bizagi Modeler

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

A implementação do software tem como sua atividade central a codificação do sistema. No entanto, o processo se preocupa com a prática de reuso, com a avaliação (teste) do produto construído, e com a comunicação de eventuais decisões tomadas nesta fase.

# IMPLEMENTAR SOFTWARE

## Elementos do processo

### ☐ *Identificar oportunidade de reuso*

Esta atividade avalia a possibilidade de reuso de código, classes, estruturas de dados e outros elementos reutilizáveis, com vistas ao aumento da eficiência e eficácia na produção de software.

### ☐ *Codificar*

Esta atividade se refere à própria codificação do sistema.

### ☐ *Criar/Publicar software*

O sistema é disponibilizado em ambiente de homologação para sua adequada testagem.

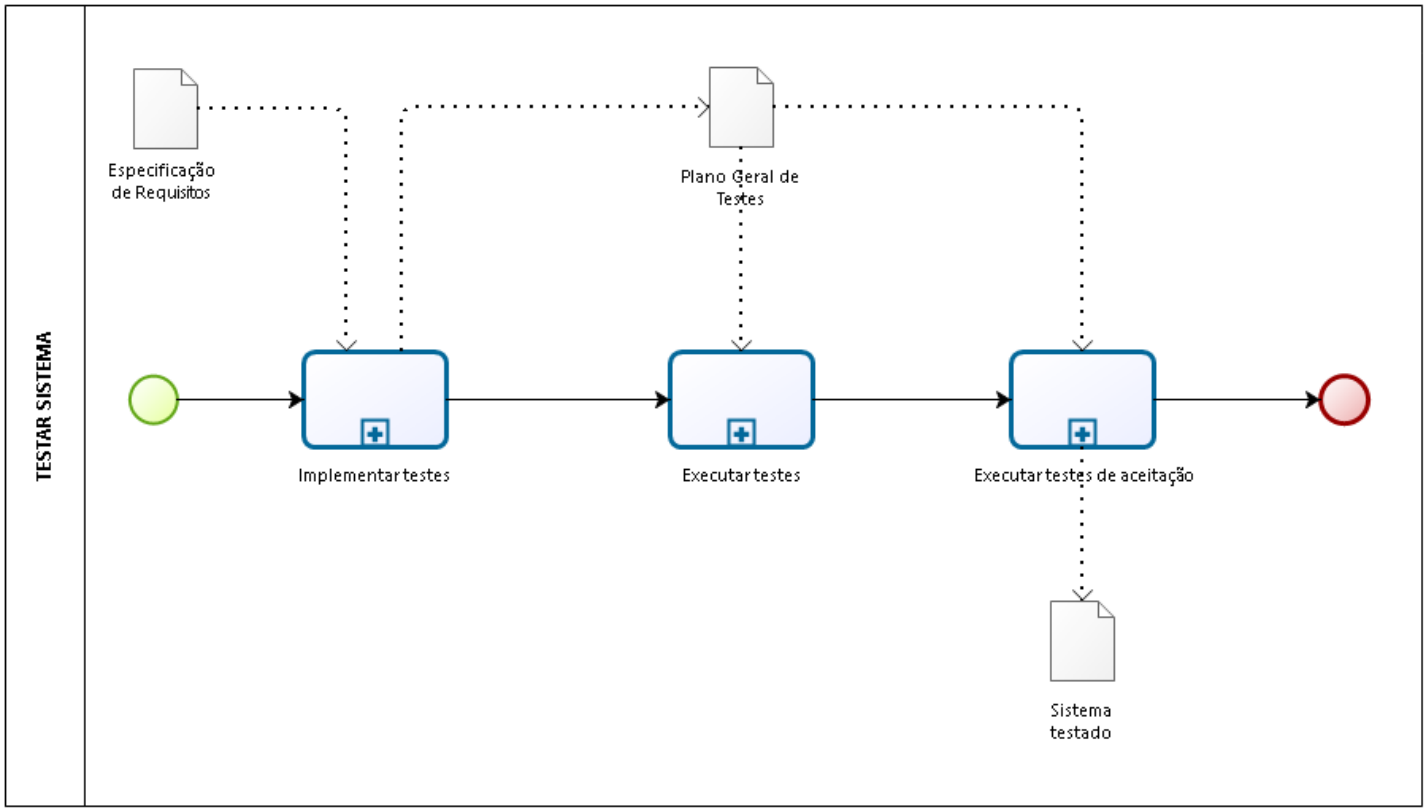
### ☐ *Avaliar a implementação*

A implementação deve ser testada, ou avaliada. Esta avaliação se refere a um teste preliminar, realizado pelo próprio desenvolvedor, para verificar que as funcionalidades que ele implementou estão operando adequadamente.

### ☐ *Comunicar decisões significativas*

Decisões técnicas tomadas durante o processo de implementação são comunicadas para as partes interessadas, de modo a garantir uniformidade do conhecimento da equipe sobre o produto em construção.

## 2.3.2-Testar




Powered by bizagi Modeler

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

Processo que tem como objetivo executar os testes funcionais do sistema em desenvolvimento com base nas especificações de testes elaboradas previamente.

# TESTAR SISTEMA

## Elementos do processo

### *Implementar testes*

Neste processo os testes são planejados e especificados de maneira completa, tornando possível sua posterior aplicação.

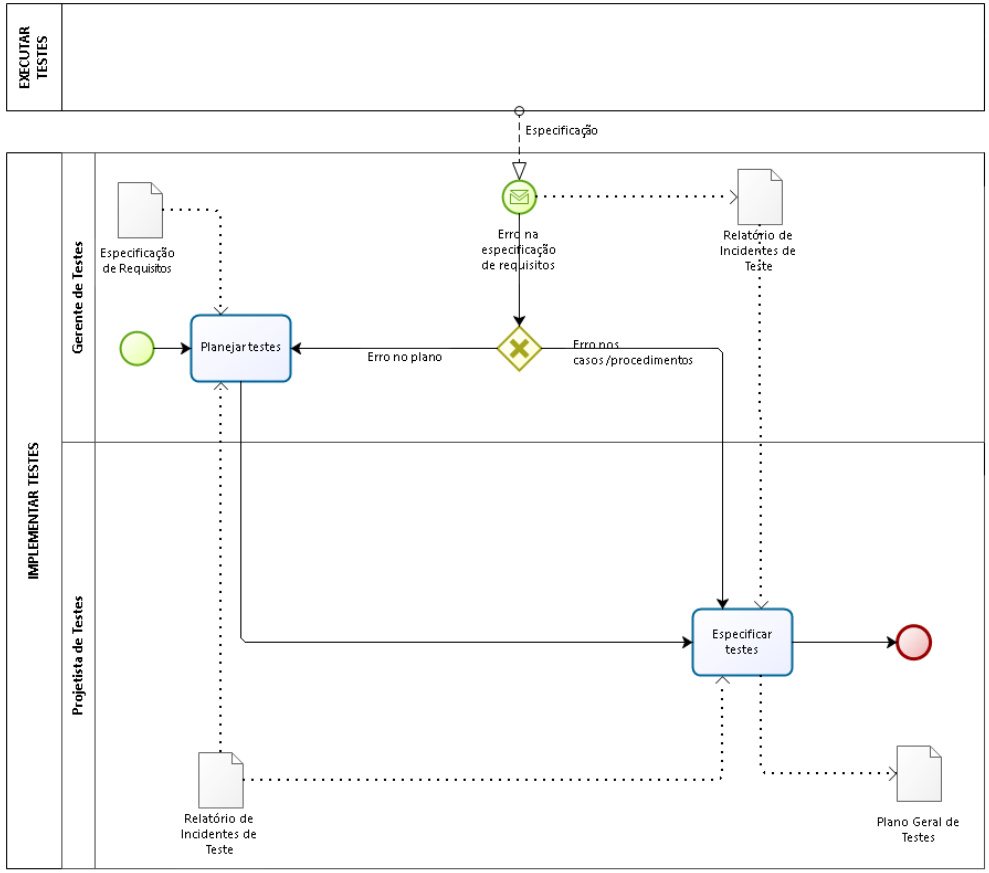
### *Executar testes*

Neste processo os testes planejados anteriormente são efetivamente realizados, podendo implicar na necessidade de solicitações de correção do código ou mesmo da especificação dos testes.

### *Executar testes de aceitação*

O teste de aceitação é também conhecido como homologação, ou mesmo validação. A diferença deste teste é que ele é conduzido pela equipe de Sistemas, mas avaliado pelo cliente, ou usuário final indicado para este fim.

## 2.3.2.1-Implementar Testes



**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

Neste processo os testes são planejados e especificados de maneira completa, tornando possível sua posterior aplicação.

IMPLEMENTAR TESTES

## Elementos do processo

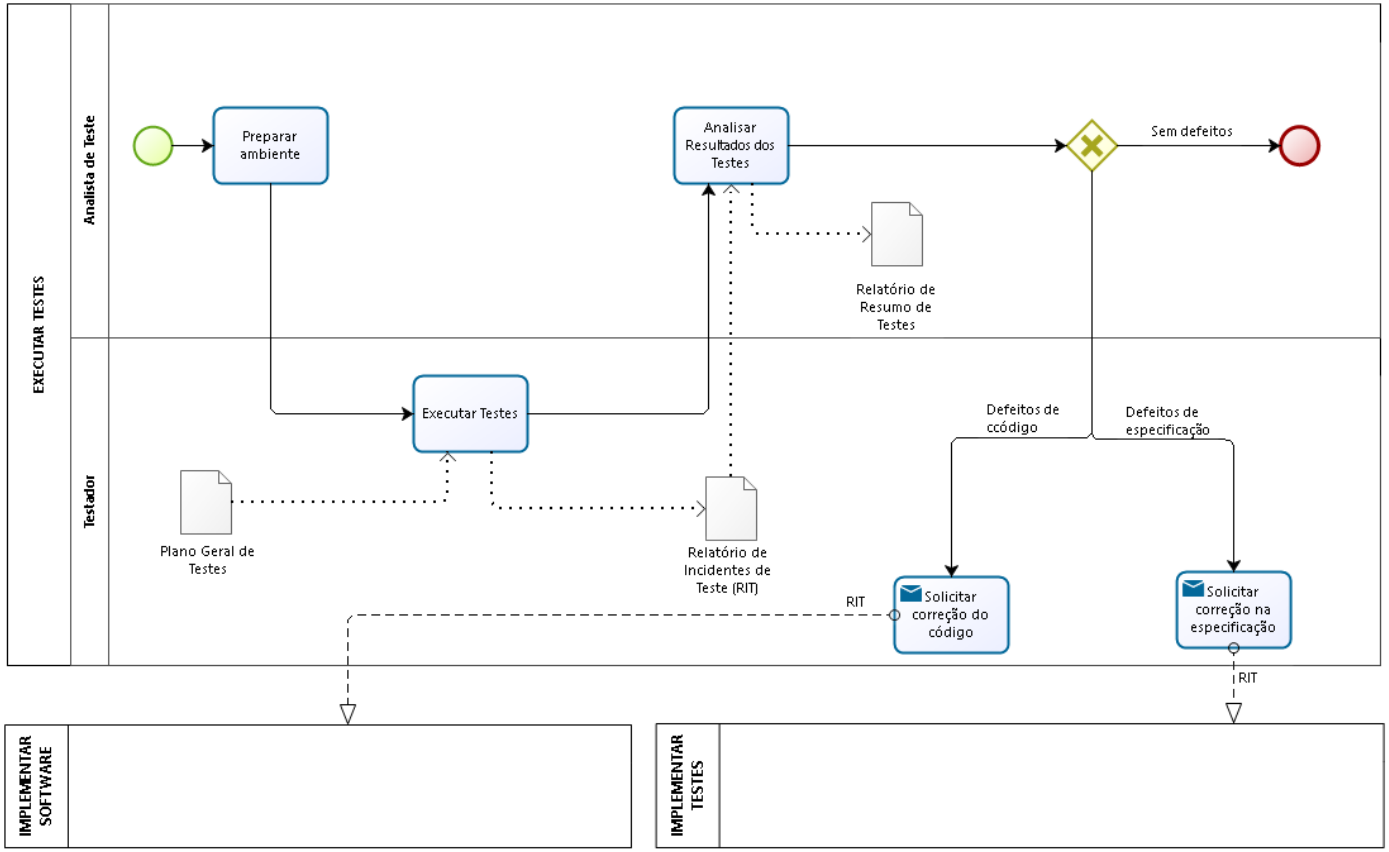
☐ *Planejar testes*

Esta atividade consiste no planejamento das etapas de construção dos testes, a serem aplicadas no sistema em questão.

☐ *Especificar testes*

Elaborar casos de testes e procedimentos para aplicação desses casos de teste. Estas informações ficam registradas no Plano Geral de Testes (PGT).

## 2.3.2.2-Executar Testes




Powered by bizagi Modeler

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

Neste processo os testes planejados anteriormente são efetivamente realizados, podendo implicar na necessidade de solicitações de correção do código ou mesmo da especificação dos testes.

# EXECUTAR TESTES

Esse processo define as atividades relacionadas aos testes funcionais do sistema (realizado pela equipe de desenvolvimento) e também aos testes de aceitação (realizado por um grupo de usuários previamente definidos no planejamento dos testes)

## Elementos do processo

### *Analisar Resultados dos Testes*

Avaliar os resultados relatados nos Relatórios de Incidentes de Teste. Um resumo do processo de execução e dos incidentes identificados devem ser relatados no Relatório de Resumo de Testes.

Caso existam falhas a serem corrigidas, deve ser decidido se, após a correção dessas falhas, os casos/procedimentos de teste associados ao item que falhou deverão ser reexecutados.

Essa decisão deve estar baseada na criticidade do item e na gravidade da falha. Além disso, a falha deve ser analisada para definir se existe a necessidade de acerto na especificação de requisitos e nos casos/procedimentos de teste.

### *Executar Testes*

Execução dos casos e procedimentos de teste, conforme definido no plano de testes.

As falhas devem ser registradas no Relatório de Incidentes de Teste, bem como a captura das telas do sistema.

Além disso, todos os eventos da execução devem ser registrados no Log de Testes.

### *Preparar ambiente*

Preparação do ambiente de teste em termos de hardware e software. Após essa preparação, o ambiente deve estar totalmente pronto para início da execução dos testes. São atividades típicas:

- Configuração de equipamentos (servidores web, servidores de e-mail e clientes)

- Instalação e configuração de softwares, bibliotecas e plug-ins de apoio

- Instalação e configuração da versão a ser testada

- Execução de scripts de criação do banco de dados

- Execução de scripts de inicialização do banco de dados (por exemplo, dados para casos de teste, usuários e perfis)

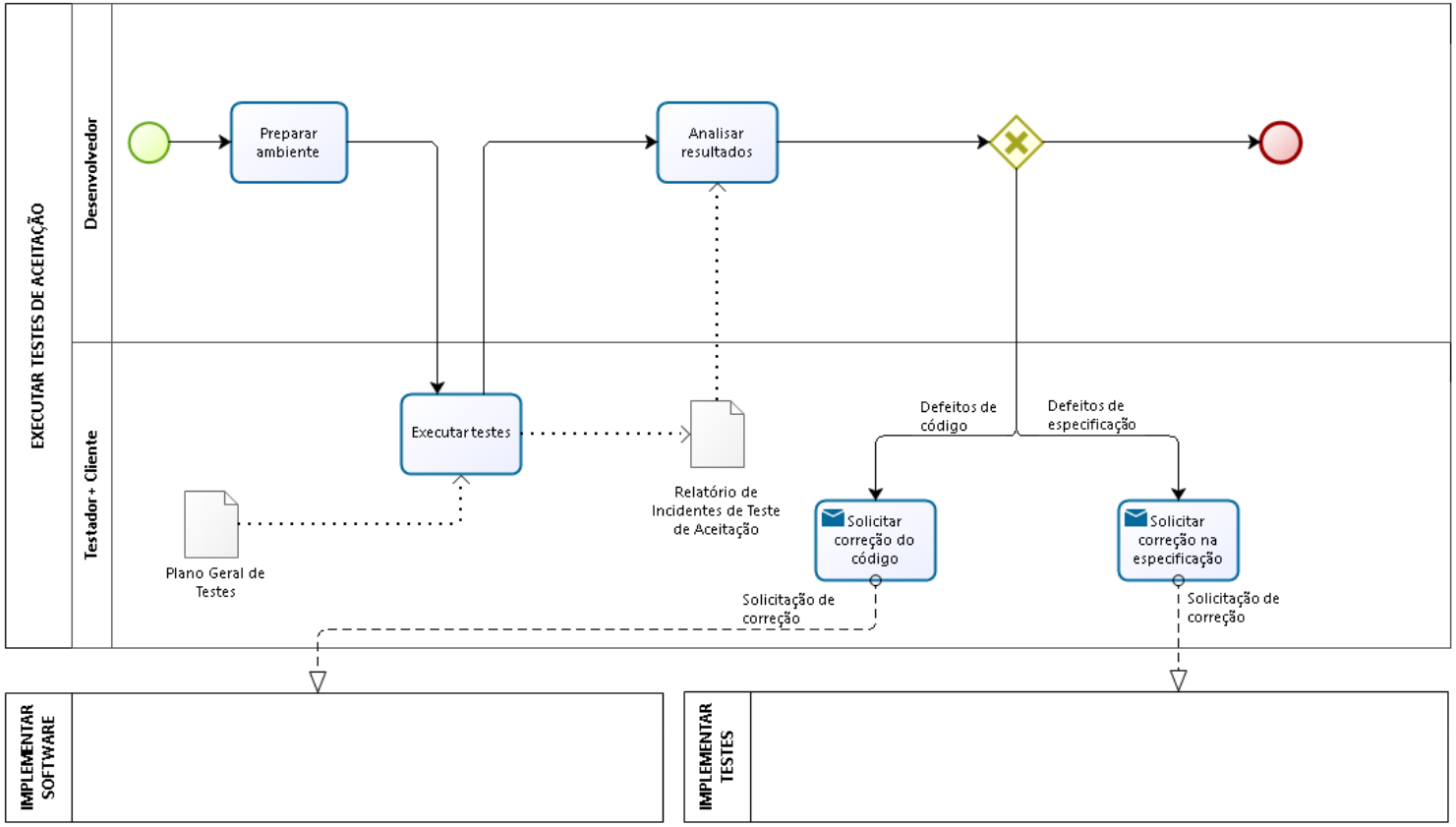
### Solicitar correção na especificação

O testador deve solicitar a correção da especificação incorreta, conforme apontado pelos testes.

### *Solicitar correção do código*

O testador deve solicitar a correção do código incorreto, conforme apontado pelos testes.

## 2.3.2.3-Aceitação




Powered by bizagi Modeler

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

O teste de aceitação é também conhecido como homologação, ou mesmo validação. A diferença deste teste é que ele é conduzido pela equipe de Sistemas, mas avaliado pelo cliente, ou usuário final indicado para este fim.

# EXECUTAR TESTES DE ACEITAÇÃO

## Elementos do processo

### *Executar testes*

Execução dos testes de aceitação pelo grupo de usuários definido no plano de testes.

As falhas devem ser registradas no Relatório de Incidentes de Testes de Aceitação, bem como a captura das telas do sistema.

### *Preparar ambiente*

Preparação do ambiente de testes de aceitação em termos de hardware e software.

### *Analisar resultados*

Avaliar os resultados relatados nos Relatórios de Incidentes de Testes de Aceitação. Um resumo do processo de execução e dos incidentes identificados devem ser relatados no Relatório de Resumo de Testes de Aceitação. Caso existam falhas a serem corrigidas, deve ser decidido se, após a correção dessas falhas, os testes de aceitação deverão ser reexecutados.

Essa decisão deve estar baseada na criticidade do item e na gravidade da falha juntamente com o usuário testador. Além disso, a falha deve ser analisada para definir se existe a necessidade de acerto na especificação de requisitos e nos casos/procedimentos de teste.

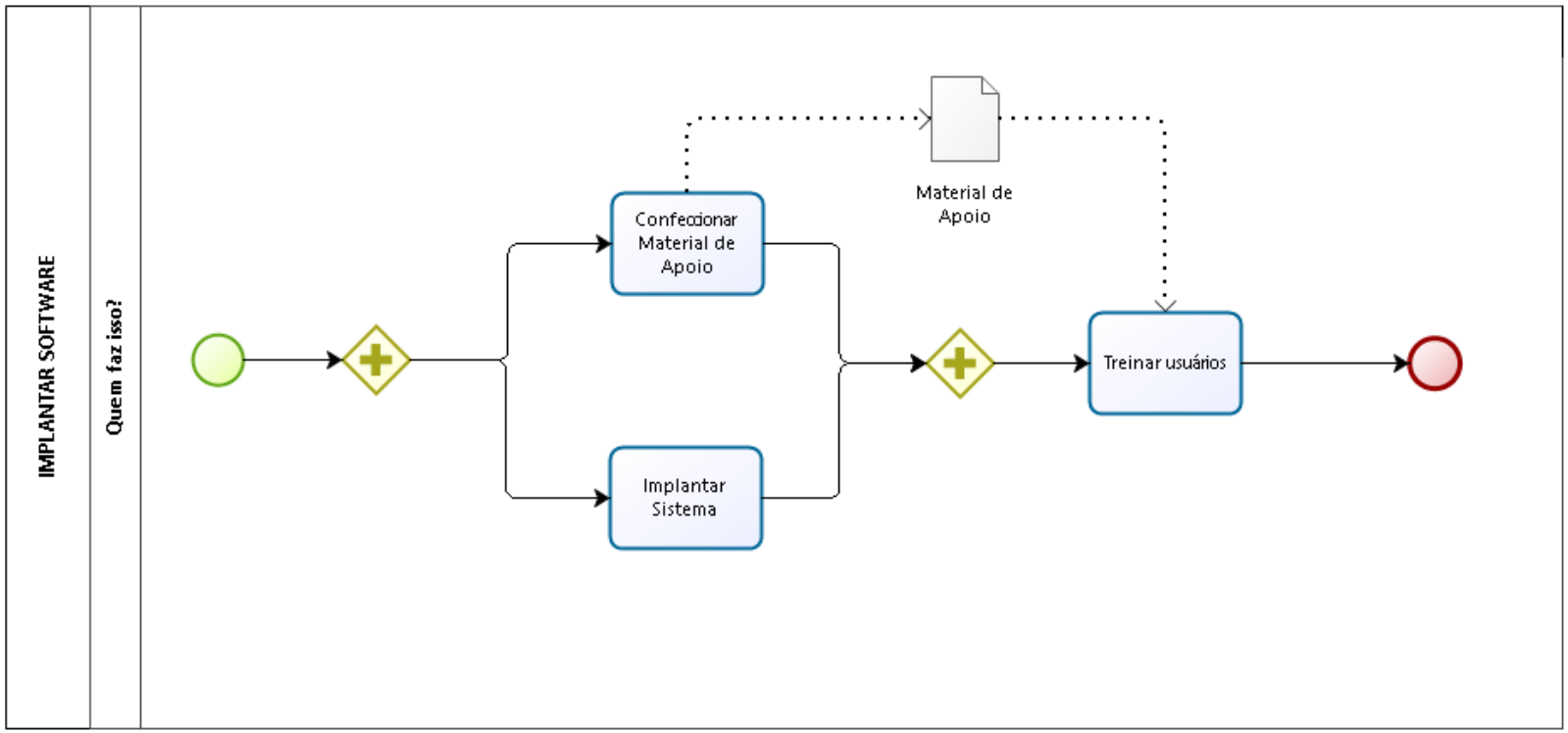
### *Solicitar correção do código*

O testador deve solicitar a correção do código incorreto, conforme apontado pelos testes.

### *Solicitar correção na especificação*

O testador deve solicitar a correção da especificação incorreta, conforme apontado pelos testes.

## 2.4-Implantar




Powered by bizagi Modeler

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

A implantação do software passa pela disponibilização do software em seu ambiente final, acessível aos usuários autorizados. Também, inclui a elaboração de material didático de apoio ao usuário, que poderá ser utilizado tanto pelo próprio usuário quanto em eventuais treinamentos proporcionados pela área de Tecnologia da Informação.

# IMPLANTAR SOFTWARE

## Elementos do processo

### ☐ *Implantar Sistema*

Nesta atividade deve ser preparado um pacote contendo os componentes necessários (incluindo o próprio sistema e scripts de instalação) para implantação do sistema em seu ambiente de operação, bem como treinamento dos usuários.

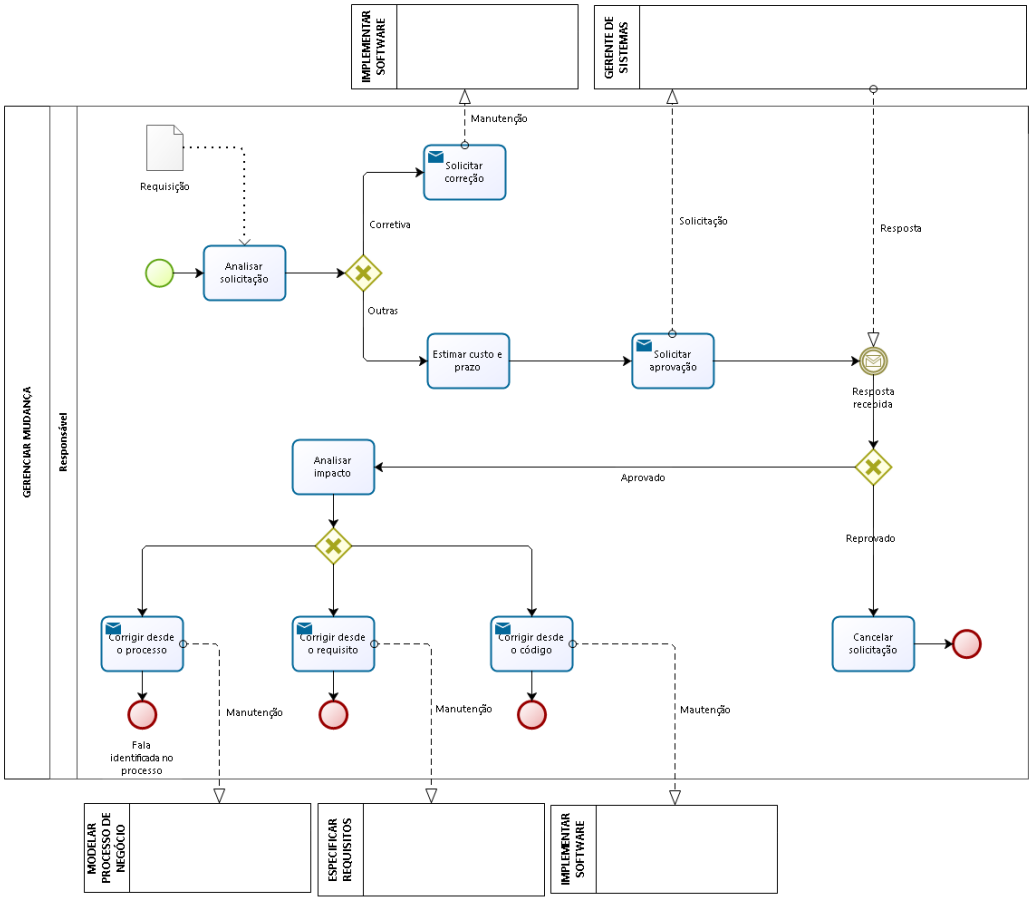
### ☐ *Confeccionar Material de Apoio*

Nessa atividade deverá ser confeccionado o material de apoio aos usuários. Esse material pode ser um manual do usuário, um manual de operação, um vídeo de treinamento, um guia de navegação ou qualquer outro material de apoio aos usuários do sistema. É importante que o material a ser produzido esteja definido de forma detalhada no escopo do projeto que consta no Termo de Abertura.

### ☐ *Treinar usuários*

Esta atividade envolve a capacitação inicial dos usuários para a utilização do sistema.

# 3-Gerenciar Mudança



**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

As solicitações de manutenção são tratadas no processo "Gerenciar Mudança". As solicitações de manutenção corretiva não passam por maiores análises, e são tratadas mais rapidamente. Os demais tipos de manutenção passam por uma aprovação do custo e do prazo pelo solicitante, bem como por uma análise de impacto para verificar a partir de que ponto (processo, requisito, código) os ajustes devem ser implementados.

# GERENCIAR MUDANÇA

## Elementos do processo

### *Analisar solicitação*

A requisição do cliente é analisada para verificar o procedimento de manutenção mais adequado.

### *Solicitar correção*

Caso se trata de uma falha/erro no sistema, a solicitação é encaminhada para correção imediata.

### *Estimar custo e prazo*

É realizada uma estimativa de prazo e custo para o entendimento da requisição. Como base para a estimativa de custo e prazo é utilizada a técnica de cálculo de Pontos de Função (PF).

### *Solicitar aprovação*

Após identificado o prazo e o custo estimados, é necessário que o Gerente de Sistemas aprove a continuidade do atendimento.

### *Cancelar solicitação*

Caso a solicitação não seja aprovada, a mesma é finalizada e isto deve ser registrado no sistema para eventual futura consulta e análise.

### *Analisar impacto*

É analisado o impacto da mudança para o sistema e sua documentação. O objetivo é identificar o quanto deve ser modificado em termos de sistema e documentação.

### *Corrigir desde o processo*

É solicitada a correção do processo. Mudanças maiores impactarão no Modelo de Processo, sendo que este deve ser revisto e passar por todo o processo de avaliação novamente.

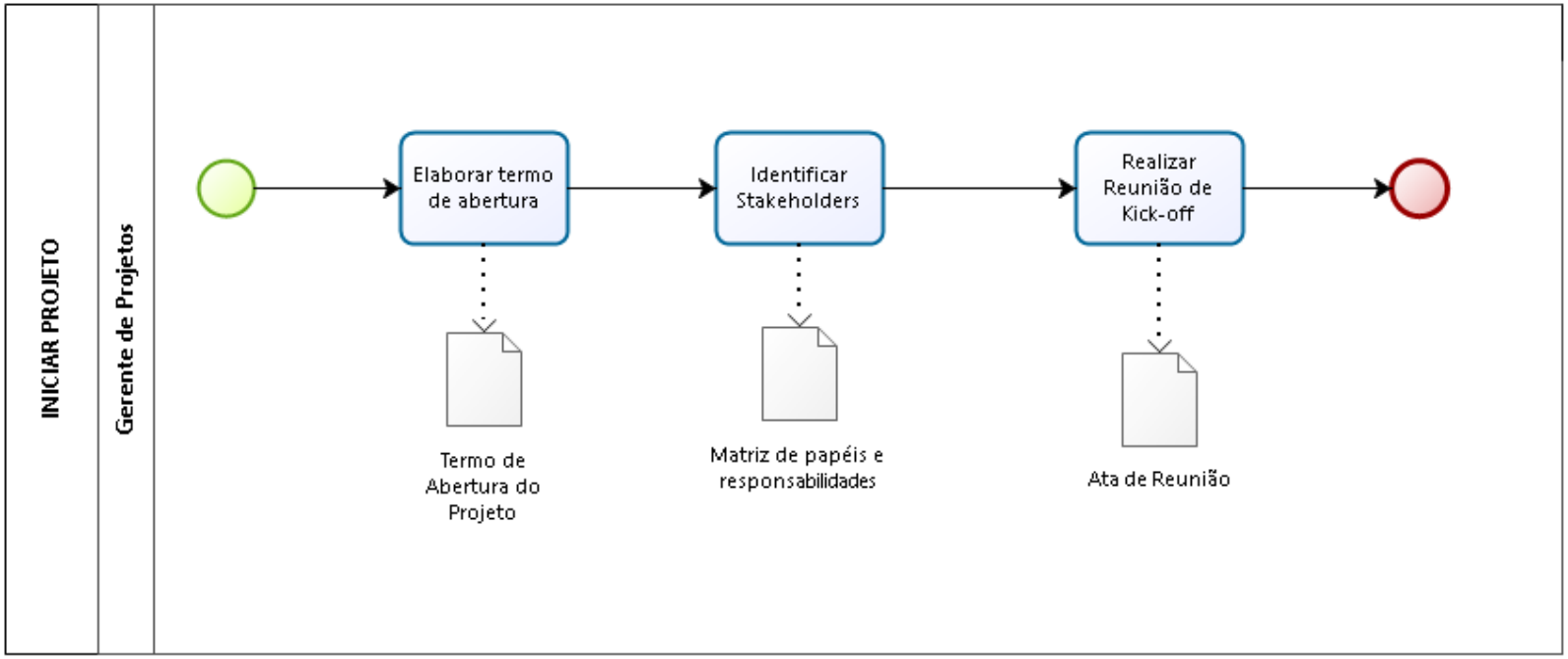
### *Corrigir desde o requisito*

É solicitada a correção do documento de requisitos. Mudanças médias impactarão na Especificação dos Requisitos, sendo que este deve ser revisto e passar por todo o processo de avaliação novamente.

### Corrigir desde o código

É solicitada a correção do código do programa. Mudanças menores impactarão no apenas no código, sendo que este deve ser revisto e passar por todo o processo de avaliação novamente.

# P-Iniciar




Powered by bizagi Modeler

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

O início do projeto é materializado no Termo de Abertura do Projeto (TAP), pela identificação e classificação dos stakeholders, e pela realização da reunião inicial do projeto, também chamada de reunião de kick-off.

# INICIAR PROJETO

## Elementos do processo

### ☐ *Realizar Reunião de Kick-off*

Nessa atividade é realizada uma apresentação da metodologia a ser seguida no desenvolvimento do sistema. Além disso, é gerado o Termo de Abertura e uma Ata de Reunião com as principais decisões tomadas. São definidos na reunião:

- Pontos focais

- Cronograma inicial para reuniões de mapeamento de processos de negócio

- Cadeia de valor agregado (lista de macroprocessos)

- Visão inicial das fronteiras do sistema

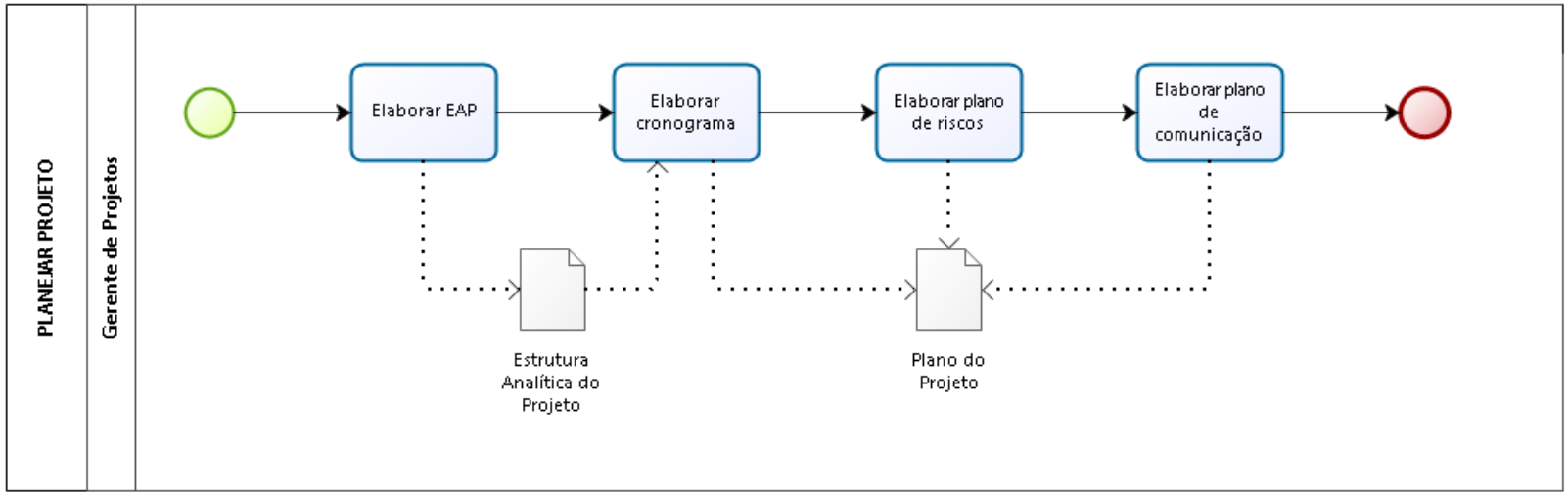
### ☐ *Elaborar termo de abertura*

A elaboração do termo de abertura evolve a construção do cronograma, do plano de custos e do plano de riscos, sendo todas estas informações de caráter preliminar e estimativo. O objetivo é fornecer informações básicas que possam ajudar na decisão de executar o projeto ou não.

### ☐ *Identificar Stakeholders*

Todas as partes interessadas no projeto devem ser identificadas, e seus respectivos papéis devem ser definidos.

# P-Planejar




Powered by bizagi Modeler

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

O planejamento do projeto de software baseia-se na construção de um cronograma, que será a base para o acompanhamento do projeto, bem como a elaboração de um plano de riscos e um plano de comunicação para o projeto.

# PLANEJAR PROJETO

## Elementos do processo

☐ *Elaborar EAP*

Elaborar Estrtura Analítica do Projeto (EAP), representando as principais partes entregáves do projeto.

☐ Elaborar cronograma

Elaborar um cronograma baseado na EAP, com atividades para cada entrega e com prazos para a execução de cada atividade. Com base na alocação de pessoal para cada atividade (e outros recursos), deve ser estimado o custo para o projeto.

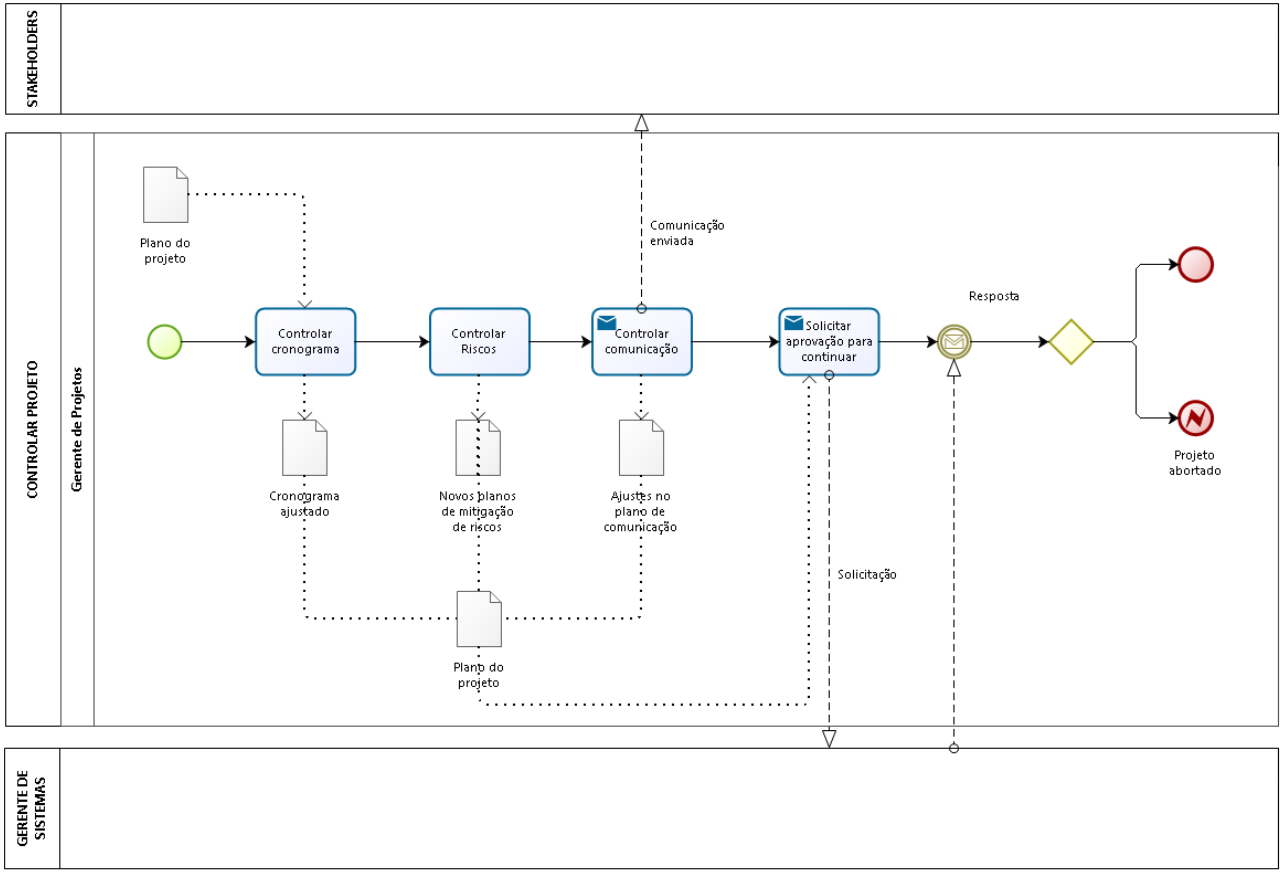
☐ *Elaborar plano de riscos*

Todos os possíveis riscos devem ser identificados, e suas probablidades de ocorrência devem ser calculados. Para os riscos com maior probabilidade de ocorrência, devem ser elaborados planos de contingência.

☐ *Elaborar plano de comunicação*

Um plano de comunicação deve ser elaborado, definindo as pessoas para quem serão enviadas as comunicações, as mídias a serem utilizadas, e os tipos de comunicação que serão enviadas para cada pessoa.

# P-Controlar


STAKEHOLDERS
CONTROLAR PROJETO
Gerente de Projetos
GERENTE DE SISTEMAS
Plano do projeto
Comunicação enviada
Controlar cronograma
Controlar Riscos
Controlar comunicação
Solicitar aprovação para continuar
Resposta
Projeto abortado
Cronograma ajustado
Novos planos de mitigação de riscos
Ajustes no plano de comunicação
Solicitação
Plano do projeto

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

O controle do projeto envolve acompanhar, revisar e reportar o progresso do projeto, bem como ajustar os rumos do projeto durante sua execução.

# CONTROLAR PROJETO

## Elementos do processo

### *Controlar cronograma*

Processo de monitorar a situação das atividades do projeto e de seus custos associados, atualizando o cronograma, incluindo os registros de eventuais antecipações e atrasos nessas atividades. Isto possibilita identificar desvios em relação ao plano corrente e tomar ações para corrigir esses desvios.

### *Controlar Riscos*

O processo elabora planos de resposta aos riscos identificados previamente, monitorando eventuais riscos residuais e/ou novos riscos não identificados me momento anterior, viabilizando a avaliação desses riscos e a elaboração de planos para tratá-los. O principal benefício gerado é o aumento da eficiência no tratamento dos riscos do projeto, de modo contínuo, durante todo o ciclo de vida do projeto.

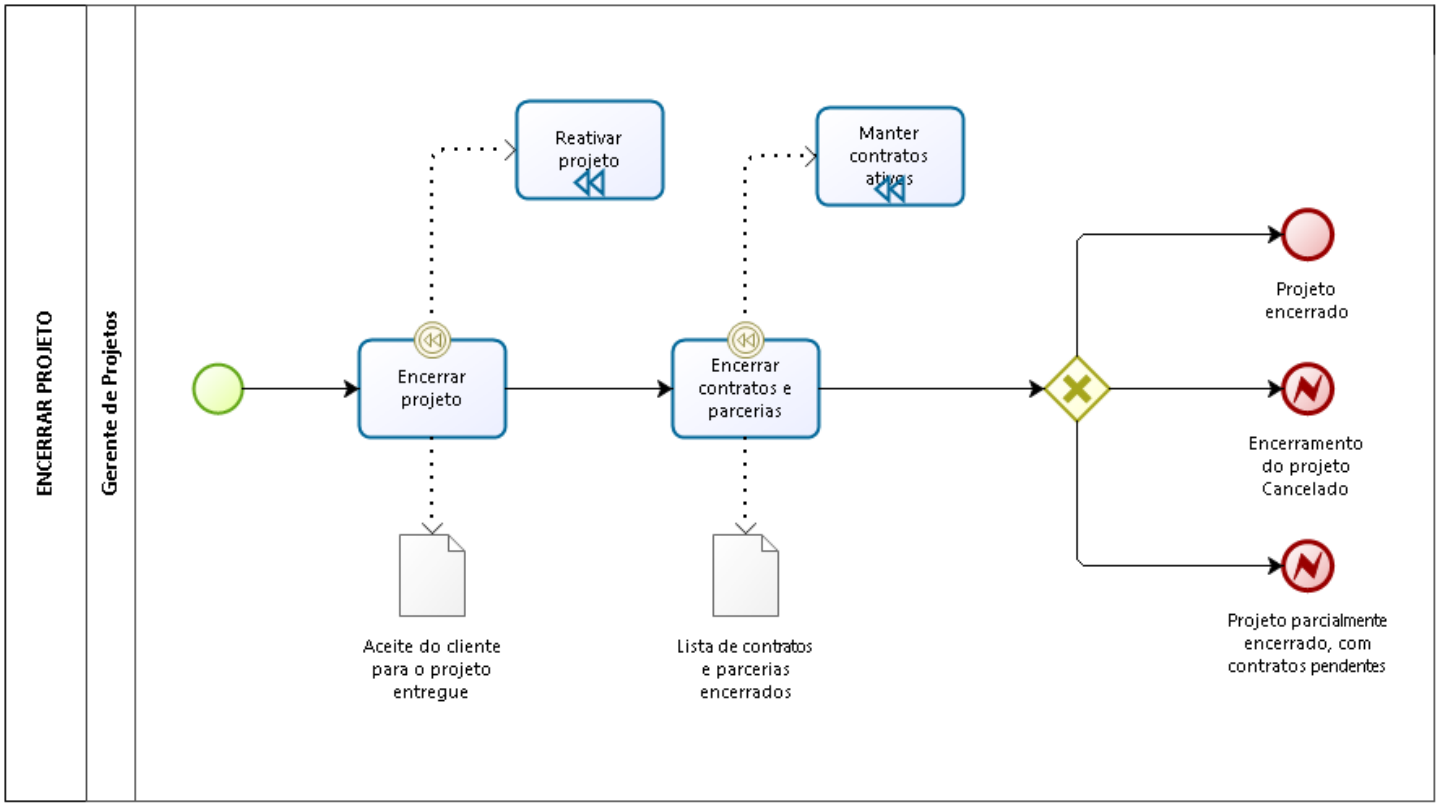
### *Controlar comunicação*

O processo avalia a necessidade de ajustes operacionais e táticos no plano de comunicação, incluindo a análise da necessidade de se ajustar a lista de stakeholders que são comunicados, bem como sua segmentação. As comunicações precisam ser enviadas em intervalos regulares pré-definidos.

### *Solicitar aprovação para continuar*

Ao passo que mudanças são realizadas no Plano do Projeto é possível que elementos como custo, prazo e escopo se tornem inaceitáveis para os stakeholders do projeto. Neste caso, é preciso garantir que o patrocinador do projeto, representando os demais stakeholders, aprove a continuidade do projeto.

# P-Encerrar




Powered by bizagi Modeler

**Versão: Evolução da 4.0**

**Autor: Autor da Teste/Fiocruz**

**Descrição**

O encerramento do projeto garante que o cliente do projeto concorda e aceita o encerramento, bem como garante a finalização oficial de todos os contratos e parcerias ligados a este projeto.

# ENCERRAR PROJETO

## Elementos do processo

☐ *Encerrar projeto*

O projeto deve ser oficialmente encerrado, inclusive com a anuência do cliente e do patrocinado.

☐ *Encerrar contratos e parcerias*

Contratos, convênios e eventuais parcerias devem ser encerradas oficialmente, com o termo de aceite das partes envolvidas.

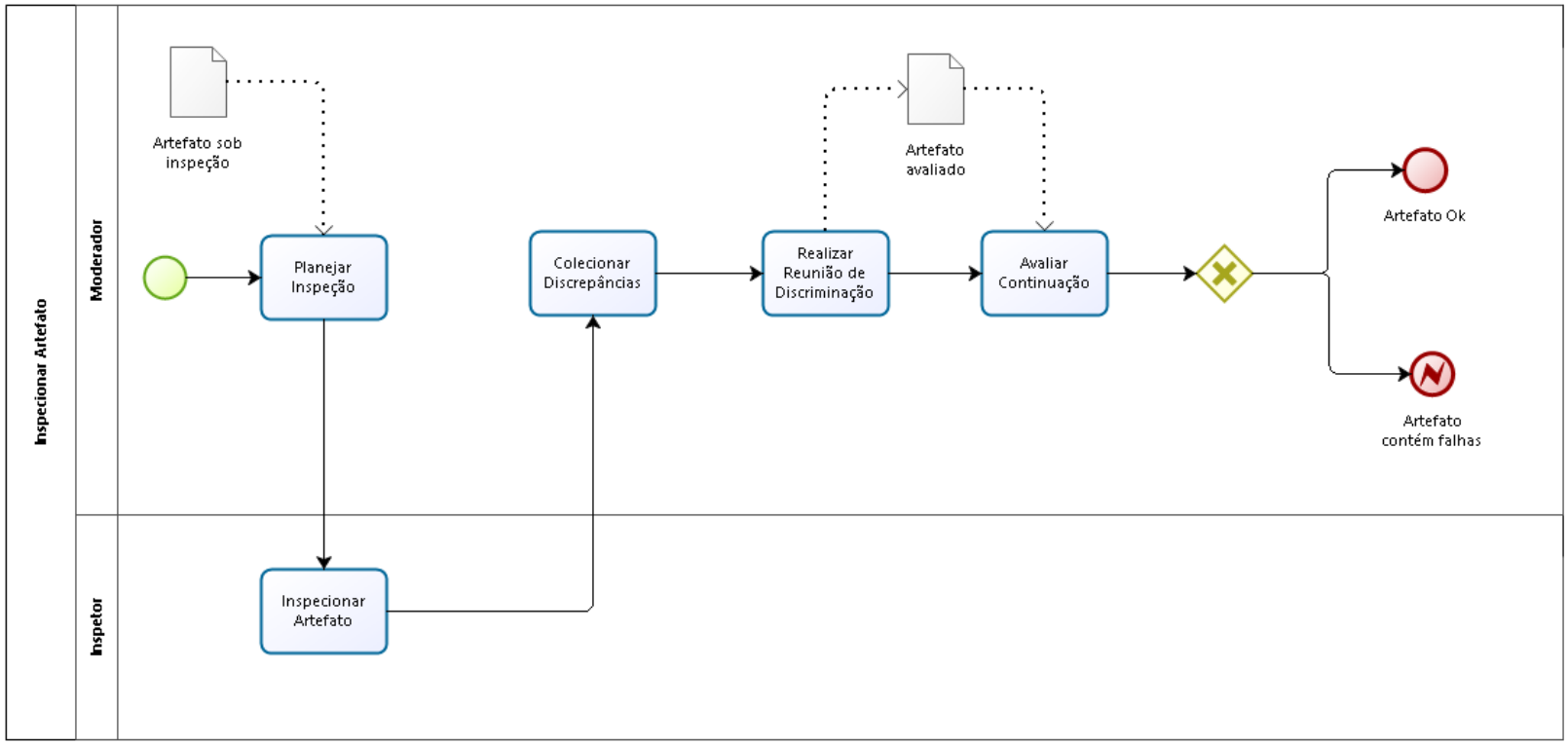
☐ *Reativar projeto*

Caso sejam identificadas falhas na intenção de cancelar o projeto por parte dos stakeholderes, ou se o cliente aceitar o encerramento do projeto apenas sob algumas condições de entrega, o cancelamento do projeto é revertido e as condições necessárias para o encerramento do projeto são buscadas.

☐ *Manter contratos ativos*

Caso existam contratos que não podem ser cancelados no momento em função de impedimentos técnicos ou legais, os mesmos deixa de figurar como cancelados e se mantém ativos até que a condição para encerramento seja viabilizada.

# Inspecionar Artefato




Powered by bizagi Modeler

**Versão:** 3.0

**Autor:** Equipe FIOCRUZ/CONSULTORIA CONTRATADA

Nesta atividade, os modelos de processos de negócio são verificados com base em um checklist que captura heurísticas de boa construção de modelos. A atividade de verificação é interna ao projeto e não envolve o cliente. Também são verificados os Casos de Uso.

# Inspecionar Artefato

Processo responsável por avaliar a qualidade dos artefatos produzidos.

Atualmente esse processo é usado para:

- Verificação da qualidade dos modelos BPMN
- Verificação da qualidade dos casos de uso

## Elementos do processo

### *Planejar Inspeção*

O processo de inspeção é planejado com a realização das seguintes tarefas:

- Definição dos artefatos a serem inspecionados
- Definição dos inspetores
- Atribuição dos artefatos aos inspetores
- Definição do cronograma da inspeção
- Distribuição do material aos inspetores

### *Inspecionar Artefato*

É realizada a inspeção propriamente dita.  As discrepâncias são registradas no formulário de discrepâncias e, ao final, o formulário é enviado para o moderador.

### *Colecionar Discrepâncias*

Os diversos formulários de discrepância são reunidos em um só formulário, que é ordenado pela localização da discrepância no artefato.

### *Realizar Reunião de Discriminação*

Nessa reunião as discrepâncias registradas pelos inspetores são classificadas como defeitos ou falso positivos.

### *Avaliar Continuação*

Devem ser realizadas as seguintes tarefas:

- Avaliar se todos os defeitos registrados na planilha estão marados como corrigidos
- Avaliar se realmente os defeitos foram corrigidos no artefato
- Decidir se o artefato será reinspecionado

# Resources

## Gerente de Sistemas (Função)

### Descrição

Responsável por atender as demandas ao setor de desenvolvimento e gerenciar a execução dos projetos.

Perfil: Profissional com conhecimento de práticas de gerenciamento de projetos de software, soluções tecnológicas para sistemas de informação e capacidade de tomada de decisão.

## Analista de Requisitos (Função)

Responsável por identificar e especificar os requisitos do sistema em desenvolvimento.

Perfil: Profissional com habilidades em elicitação de requisitos a partir de diferentes fontes (pessoas e documentações técnicas), análise de requisitos funcionais e não funcionais, especificação de software sob perspectivas de dados e função. É necessário conhecimento em técnicas como entrevistas, para elicitação de requisitos, e técnicas de especificação como modelagem de casos de uso e diagramas ER/UML.

## Projetista de Sistemas (Função)

Responsável por desenhar a solução técnica do sistema com base nos requisitos identificados.

Perfil: Profissional com habilidades de transformar os requisitos em soluções de software por meio de abstrações, utilizando modelos arquiteturais e conceitos de coesão e acoplamento, bem como sólido conhecimento das tecnologias envolvidas na solução técnica (frameworks, bibliotecas, e linguagens de programação).
É importante para o arquiteto ter conhecimento de estilos e padrões arquiteturas, linguagens e notações para especificação de software como modelos ER/UML em diferentes níveis de abstração, bem como o ferramental de apoio às atividades de projeto.

## Analista de Processos (Função)

Responsável por identificar e modelar os processos de negócio que fazem parte do escopo do projeto.

Perfil: Profissional com habilidades em elicitação de processos de negócio a partir de diferentes fontes (pessoas e documentações técnicas), análise e especificação de processos de negócio, reconhecendo atividades e suas entradas e saídas, bem como um encadeamento lógico entre elas. Ainda, é necessário que o analista de processos entenda que existem regras associadas à execução dos processos precisam também ser capturadas e especificadas. É necessário conhecimento em técnicas como entrevistas, para elicitação de processos, e técnicas de especificação como modelagem em BPMN.

## Programador (Função)

Responsável por construir o código do sistema com base nas especificações. Também denominado Desenvolvedor.

Perfil: Profissional com habilidades de abstração e desenvolvimento de algoritmos nas linguagens de programação adotadas pela equipe de desenvolvimento. É necessário ainda que o programador tenha conhecimento para manipulação de banco de dados, ferramentas de desenvolvimento como IDE, frameworks e bibliotecas.

## Testador (Função)

Responsável por realizar as atividades de planejamento, projeto e execução de testes do sistema.

Perfil: Profissional com conhecimento de técnicas para elaboração de procedimentos e casos de teste, além de instrumentação de testes utilizando as tecnologias relacionadas ao desenvolvimento dos sistemas a serem testados.

## Moderador (Função)

Responsável por coordenar o processo de inspeção

## Autor (Função)

Autor do artefato sob inspeção

## Inspetor (Função)

Responsável por realizar a inspeção propriamente dita no artefato. A alocação ao papel de inspetor deve respeitar a premissa de que o mesmo não seja autor do artefato.

Perfil: Profissional com conhecimento sobre os artefatos a serem inspecionados/revisados, bem como sobre os procedimentos de construção e características de qualidade de tais artefatos.

## Desenvolvedor (Função)

Este termo é designado a qualquer membro da equipe de desenvolvimento responsável por produzir artefatos que levam ao produto final. Entretanto, este termo é comumente associado ao papel de Programador.

## Anexo IV – Pesquisa de percepção após ajuste de modelo

Trata-se de uma pesquisa simples no modelo *Likert*, onde você lerá uma afirmação e dirá se concorda ou discorda, e em que grau. Você precisará ler os modelos e responder a um formulário contendo 6 afirmações para cada um desses modelos, que estão no corpo da mensagem. Esta pesquisa foi executada experimentalmente e o processo de ler e responder para os dois modelos está levando algo entre 12 e 17 minutos, dependendo da pessoa.

Não há ordem definida, ou seja, você pode ler e responder primeiro o **IODO**, ou ler e responder primeiro o **CLORO**. A escolha da ordem é sua. Por isso mesmo os modelos não foram numerados ou identificados por letras.

**MODELO IODO**

| Afirmação | Discordo totalmente | Tendo a discordar | Sou incapaz de opinar | Tendo a concordar | Concordo plenamente |
|---|---|---|---|---|---|
| 1 - O modelo é prático e parece totalmente adequado para uso por uma equipe de desenvolvimento de software | | | | | |
| 2 - O modelo é muito fácil de se ensinar e de se aprender | | | | | |
| 3 - Há características neste modelo que facilitam muito o seu uso | | | | | |
| 4 - O modelo deixa claro os erros podem acontecer e assim ajuda a evitá-los e ou tratá-los | | | | | |
| 5 - A estrutura visual dos elementos do modelo formam um conjunto muito agradável de ler e bem organizado | | | | | |
| 6 - O modelo é tão simples e prático que qualquer profissional de sistemas poderia compreendê-lo facilmente | | | | | |

**MODELO CLORO**

| Afirmação | Discordo totalmente | Tendo a discordar | Sou incapaz de opinar | Tendo a concordar | Concordo plenamente |
|---|---|---|---|---|---|
| 1 - O modelo é prático e parece totalmente adequado para uso por uma equipe de desenvolvimento de software | | | | | |
| 2 - O modelo é muito fácil de se ensinar e de se aprender | | | | | |
| 3 - Há características neste modelo que facilitam muito o seu uso | | | | | |
| 4 - O modelo deixa claro os erros podem acontecer e assim ajuda a evitá-los e ou tratá-los | | | | | |
| 5 - A estrutura visual dos elementos do modelo formam um conjunto muito agradável e bem organizado | | | | | |
| 6 - O modelo é tão simples e prático que qualquer profissional de sistemas poderia compreendê-lo facilmente | | | | | |

O Modelo IODO se refere ao Diagrama “Desenvolver software”, do modelo original, apresentado no Anexo II – MDS Fiocruz original, e o modelo CLORO, já do modelo ajustado, é o diagrama “Desenvolver sistema”.