Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Curso de Engenharia de Software

APOIO FERRAMENTAL AO PROCESSO DE GARANTIA DA QUALIDADE

Autor: Antonio Bezerra da Silva Júnior
Orientadora: Cristiane Soares Ramos
Co-orientador: André Peron Martins Lanna

Brasília, DF
2013

ANTONIO BEZERRA DA SILVA JÚNIOR

APOIO FERRAMENTAL AO PROCESSO DE GARANTIA DA QUALIDADE: QUASAR – QUALIDADE DE SOFTWARE E AVALIAÇÃO DE RESULTADOS

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Orientador: Cristiane Soares Ramos

Co-Orientador: André Peron Martins Lanna

Brasília, DF
2013

CIP – Catalogação Internacional da Publicação

Bezerra da Silva Júnior, Antonio.
Apoio Ferramental ao Processo de Garantia da Qualidade: QUASAR – Qualidade de Software e Avaliação de Resultados / Antonio Bezerra da Silva Júnior. Brasília: UnB, 2013.

Monografia (Graduação) – Universidade de Brasília Faculdade do Gama, Brasília, 2013. Orientação: Cristiane Soares Ramos e André Peron Martins Lanna.

1.Garantia da Qualidade. 2.Modelos de Referência. 3.Arquitetura de Software I. Soares Ramos, Cristiane, Peron Martins Lanna, Andre. II. Msc.

CDU Classificação

APOIO FERRAMENTAL AO PROCESSO DE GARANTIA DA QUALIDADE

Antonio Bezerra da Silva Júnior

Monografia submetida como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software da Faculdade UnB Gama - FGA, da Universidade de Brasília, em 23/07/2013 apresentada e aprovada pela banca examinadora abaixo assinada:

Profa. (Msc.): Cristiane Soares Ramos, UnB/ FGA
Orientador

Prof. (Msc.): André Peron Martins Lanna, UnB/ FGA
Co-orientador

Profa. (Msc.): Fabiana Freitas Mendes, UnB/ FGA
Membro Convidado

Brasília, DF
2013

Esse trabalho é dedicado à minha mãe, minha rainha, que sempre esteve ao meu lado, dando forças e acreditando em mim.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado forças para superar os obstáculos encontrados, não apenas durante os cinco difíceis anos de graduação, mas também, durante os vinte e um anos de minha vida. Agradeço também à minha mãe, minha rainha, Aurinda Nunes dos Santos, que nunca mediu esforços para garantir o melhor para mim, que sempre abriu mão de qualquer coisa, se colocando em segundo plano para me ajudar a alcançar meus sonhos e objetivos. Por fim, agradeço aos meus amigos e familiares, também àqueles familiares amigos, que sempre estiveram ao meu lado, mesmo quando abri mão da companhia deles por conta dos estudos.

Aproveito para agradecer de maneira especial aos professores André Lanna e Cristiane Soares Ramos que me apoiaram durante toda a execução deste projeto, estando sempre prontos a ajudar e colaborar para a minha evolução e aprendizado.

A Cruz Sagrada seja minha Luz. *(*São Bento)

RESUMO

Este trabalho destina-se à proposta de uma ferramenta para apoio ao processo de Garantia da Qualidade do Processo e do Produto, encontrado em normas de qualidade e modelos de maturidade. Através da adoção desta ferramenta, organizações interessadas podem adequar-se ao processo e realizar atividades de consultoria em Garantia da Qualidade. Além da adequação e apoio ao processo, a ferramenta apresentada, o software QUASAR, destina-se ao cruzamento de informações obtidas pela execução das auditorias da qualidade, com níveis de maturidade do modelo de referência, possibilitando uma visão geral dos processos da organização com relação ao modelo de referência escolhido, revelando práticas falhas e atividades e tarefas que impactam nas mesmas, revelando práticas do modelo que se encontram descobertas ou em não-conformidade, revelando dados preciosos para a organização que almeja uma avaliação de maturidade baseada em modelos de referência. Para tanto será apresentado todo o processo de desenvolvimento, explicitando as atividades de Engenharia de Software executadas até alcançar o produto final.

Palavras-chave: Qualidade de Software. Modelos de Maturidade. Auditoria de Qualidade. Framework. Desenvolvimento. Arquitetura de Software.

ABSTRACT

This work intend to propose a tool to support the process of Quality Assurance Process and Product, found in quality standards and maturity models. Through the adoption of this tool, interested organizations can adapt to the process and perform consulting activities in Quality Assurance. The adequacy and support to the process, the tool presented, the QUASAR software, intended to cross information obtained by implementing quality audits, with maturity levels of the reference model, providing an overview of the organization's processes with respect the reference model chosen, revealing flaws and practical tasks and activities that impact on them, revealing the model practices that are discovered or nonconformity, revealing precious data organization that targets a maturity assessment based on reference models. For that will be presented throughout the development process, explaining the activities of Software Engineering performed to reach the final product.

Keywords: Quality Software. Maturity Models. Quality Audit. Framework. Development. Software Architecture..

LISTA DE FIGURAS

LISTA DE QUADROS

Lista de Siglas

GQA	Garantia da Qualidade
GQPP	Garantia da Qualidade do Processo e Produto
CASE	*Computer-Aided Software Engineering*
MVC	Modelo Visão e Controle
SEI	*Software Engineering Institute*
SOFTEX	Associação para Promoção da Excelência do Software Brasileiro
CMMI	*Capability Maturity Model Integration*
MPS	Melhoria de Processo de Software
RUP	*Rational Unified Process*
UC	*Use Case* (Caso de Uso)
DSS	Diagrama de Sequência do Sistema
FA	Fluxo Alternativo
FB	Fluxo Básico

SUMÁRIO

CAPÍTULO 1 - INTRODUÇÃO

1.1 FORMULAÇÃO DO PROBLEMA

O produto de Software tem se tornado cada vez mais indispensável no cotidiano das pessoas. Desta forma, é real a crescente quantidade de empresas especializadas em desenvolvimento de software. Segundo a publicação “Qualidade e Produtividade no Setor de Software Brasileiro 2009” (MCT/SEPIN, 2010), em 2007 o número de organizações com atividades em software chegava a 31.016 (trinta e um mil e dezesseis) e obtiveram receita líquida de R$ 22,8 (vinte e dois vírgula oito) bilhões e, segundo a pesquisa, a quantidade de empresas e sua receita continua a aumentar.

Ainda de acordo com MCT/SEPIN (2010), é imprescindível que os produtos de software gerados pelas empresas apresentem qualidade e valor para o usuário final. PAULK, CURTIS, CHRISSIS e WEBER (1994) demonstram que a qualidade do produto de software é fortemente influenciada pela qualidade dos processos de software. Por este motivo, na última década muitos estudos estão voltados para a área de melhoria de processos (SOLINGEN, 2004; FERREIRA *et al.*, 2007; FERREIRA *et al.*, 2008; PEIXOTO *et al.*, 2010; SANTOS *et al.*, 2010; CAMPO, 2012). A própria criação e atualização de modelos como o MR-MPS (Modelo de Referência para Melhoria do Processo de Software Brasileiro) e CMMI (*Capability Maturity Model Integration*) (MCT/SEPIN, 2010), que visam fornecer insumos para que as empresas possam melhorar seus processos é uma evidência da preocupação com a qualidade de processos durante o desenvolvimento de software.

Ao definir processos de software, a organização espera que as atividades do processo sejam seguidas da maneira correta, de acordo com padrões estabelecidos previamente. Desta forma, há a necessidade de se auditar os processos, com o intuito de observar sua correta utilização e, assim, garantir que os processos sejam executados da melhor maneira. Neste contexto surge o papel da equipe de Garantia da Qualidade, responsável por auditar os processos e produtos de trabalho, a fim de alcançar produtos de qualidade e identificar não-conformidades dentro dos processos, bem como pontos fracos, oportunidades de melhoria, necessidades de treinamento, sendo todas estas, informações essenciais para a tomada de decisões por parte do Gerente de Projetos (SOFTEX², 2011).

É importante que todos os dados colhidos pela equipe de Garantia da Qualidade sejam mantidos em base de dados da organização, onde seja possível acompanhar as ações corretivas, histórico de pontos fracos e falhas e reincidência de problemas e, para tanto, se torna um facilitador a aplicação de ferramentas que auxiliem a equipe na organização e manutenção de suas tarefas e dados.

1.2. OBJETIVO GERAL

Tem-se como objetivo geral deste projeto, o desenvolvimento da ferramenta QUASAR – Qualidade de Software e Avaliação de Resultados, ferramenta esta que visa auxiliar na execução das atividades e tarefas do processo de Garantia da Qualidade.

1.3. OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral deste trabalho, foram estabelecidos os objetivos específicos:

(i) Estudar e investigar características do processo de Garantia da Qualidade do Processo e do Produto nos principais modelos de referência em melhoria de processo de software;

(ii) Investigar e discutir características e funcionalidades de ferramentas de apoio ao processo de Garantia da Qualidade do Processo e do Produto;

(iii) Planejar, desenvolver e documentar uma ferramenta de apoio ao processo de Garantia da Qualidade do Processo e do Produto com: Modelagem de Negócio; Análise de Requisitos; Análise e Design; Configuração de Ambiente; Gerenciamento de Projeto; Implementação; e Testes.

1.4. JUSTIFICATIVA

Segundo Pressman (2006)

> A engenharia de software é uma tecnologia em camadas. (...) qualquer abordagem de engenharia (inclusive a engenharia de software) deve se apoiar num compromisso organizacional com a qualidade. Gestão de Qualidade Total, Seis Sigmas (Six Sigma) e filosofias análogas levam à cultura de um processo contínuo de aperfeiçoamento, e é essa cultura que, em última análise, leva ao desenvolvimento de abordagens cada vez mais efetivas para a engenharia de software. A base em que se apoia é o foco na qualidade.

Como afirmado por Pressman, “a engenharia de software é uma tecnologia em camadas” (Figura 1). Tem-se as ferramentas, primeira camada, como insumos importantes para apoio a Métodos (“como fazer”), segunda camada, aos Processos (“o que fazer”), terceira camada, todas estas com Foco na Qualidade, quarta camada ou camada de base (PRESSMAN, 2006).

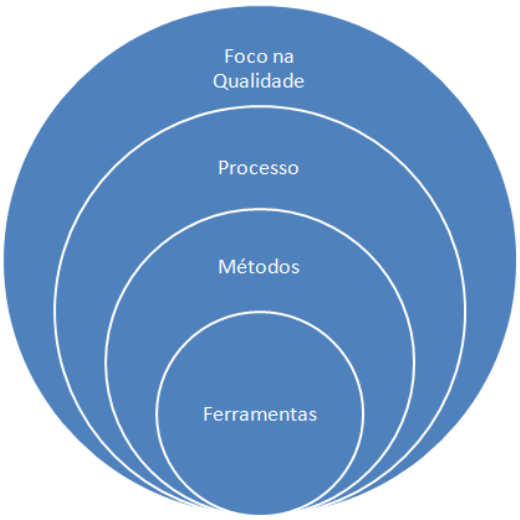

Figura 1 – Camadas da Engenharia de Software (adaptado de PRESSMAN, p.17, 2006)

A partir destas visões, percebe-se a importância do foco na qualidade de software e, ainda mais no uso de ferramentas apoiando processos. Unindo-se as duas visões, é proposto o software para apoio ao processo de garantia da qualidade.

Ao se aplicar uma ferramenta de apoio a qualquer processo, durante o ciclo de vida do desenvolvimento de um projeto de software, são encontradas diversas vantagens, tais como: (i) uniformização do processo, das atividades e dos artefatos gerados; (ii) Reutilização de artefatos ao longo do projeto (consequentemente

aumento da produtividade) e (iii) Qualidade de produto final superior (a utilização de ferramenta impõe rigor e estrutura rígida ao processo) (SILVA E VIDEIRA, 2011).

Porém, a qualidade do produto final também depende da tomada de decisões por parte do Gerente de Projetos, decisões estas que devem ser baseadas em dados coletados sobre o processo e seu uso, estabelecendo “um conjunto de práticas que garantem a integridade e a qualidade dos artefatos do projeto” (RATIONAL, 2001).

Estes dados são coletados também pela equipe de qualidade, que tem como objetivo realizar auditorias para alimentar a base de dados organizacional e identificar não conformidades associadas ao processo e produtos gerados. A equipe de Garantia da Qualidade deve servir para o Gerente de Projetos como “seus olhos e ouvidos” (SOFTEX², 2011).

Com a aplicação de uma ferramenta de apoio ao processo de Garantia da Qualidade, pode-se mais facilmente obter relatórios e analisar os resultados das auditorias de qualidade, identificando falhas em atividades e tarefas dos processos, bem como em artefatos e produtos de trabalho, possibilitando a correção em tempo hábil. Através da aplicação de ferramentas para apoio ao processo de Garantia da Qualidade, pode-se inclusive, se necessário, identificar em que práticas de Modelos de Referência as não conformidades podem impactar, sendo possível realizar as correções nos processos da organização para que estas práticas de modelos sejam corretamente satisfeitas.

Todos os aspectos a cerca do estudo e desenvolvimento da ferramenta de apoio ao processo de Garantia da Qualidade estão dispostos ao longo deste trabalho, organizado em capítulos, tratando de assuntos específicos.

Os capítulos são: Referencial Teórico (Capítulo 2), Ferramentas Relacionadas (Capítulo 3), QUASAR: Qualidade de Software e Avaliação de Resultados (Capítulo 4) Conclusão e Trabalhos Futuros (Capítulo 6), além desta introdução realizada neste capítulo (Capítulo 1). cada capítulo deste trabalho está dividido em seções e subseções, conforme nível de detalhamento necessário.

Encontra-se ao final deste documento, um apêndice (Apêndice I: detalhamento dos Casos de Uso), onde são apresentados os casos de uso do sistema de maneira detalhada.

CAPÍTULO 2 - REFERENCIAL TEÓRICO

2.1 MODELOS DE REFERÊNCIA

Qualidade de software pode ser entendida, dentre outras definições, como “conformidade a requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software profissionalmente desenvolvido (:)” (PRESSMAN, 2006). Para Bartié (2002) “qualidade não é uma fase do ciclo de desenvolvimento de software é parte de todas as fases”.

Padrões para desenvolvimento são explicitados através dos processos de software da organização e, a criação destes processos pode ser feita com base em Modelos de Referência. Modelos de referência são guias adotados pela organização com o objetivo de alinhar seus processos às boas práticas descritas. Segundo Pressman (2006), se uma organização adota processos fracos, os seus produtos finais serão impactados. Sendo assim, ao adotar um modelo de referência, espera-se impactar diretamente na melhoria dos processos e consequentemente dos produtos gerados por esses processos.

Estudo sobre a evolução da qualidade de software no Brasil – período 1994 a 2010 – publicado em 2011 pelo Ministério da Ciência e Tecnologia (NETO *et al.*, 2011), mostra um avanço na adoção dos modelos MR-MPS (SOFTEX, 2012) e CMMI (SEI, 2010). Para o modelo MR-MPS foram publicados resultados com embasamento científico sobre benefícios em melhoria de processo de software, demonstrando que no período de 2008 a 2011 as organizações brasileiras que implantaram o modelo MR-MPS obtiveram, dentre outros, ganhos em relacionados à satisfação dos clientes (TRAVASSOS e KALINOWSKI, 2012).

Nas subseções a seguir são apresentadas as principais características dos modelos referência MR-MPS (subseção 2.1.1) e CMMI-DEV (subseção 2.1.2) por serem os modelos mais utilizados na indústria de software atualmente.

2.1.1 O MODELO DE REFERÊNCIA MR-MPS

O MR-MPS é o modelo de referência do programa para Melhoria do Processo de Software Brasileiro (MPS.Br), criado em 2003, com duas metas a médio e longo prazo:

(i) Meta técnica que visa aprimorar o Modelo MPS e seus componentes (Modelo de Referência MPS para Software, Modelo de Referência para Serviço, Método de Avaliação e Modelo de Negócio);

(ii) Meta de negócio visando disseminar o MPS em todas as regiões do país, com intervalo de tempo e custo justos (SOFTEX [1], 2012).

A Figura 2 apresenta a estrutura do modelo MPS, demonstrando sua composição e suas bases de concepção. No contexto deste trabalho, o modelo usado foi o Modelo de Referência MPS para Software (MR-MPS-SW), desenvolvido com base nas NBR ISO/IEC 12207, NBR ISO/IEC 15504 e no modelo de referência CMMI-DEV (SOFTEX [1], 2012). Ele é composto por:

(i) Guia Geral MPS de Software, que apresenta o modelo e dá uma visão geral sobre níveis de maturidade e processos envolvidos;

(ii) Guia de Aquisição, que não apresenta requisitos ou atributos, apenas boas práticas para adquirentes;

(iii) Guia de Implementação, indo do nível 1 (um) ao 7 (sete) e apresentando em cada um deles os processos de cada nível do MPS-SW (SOFTEX [1], 2012).

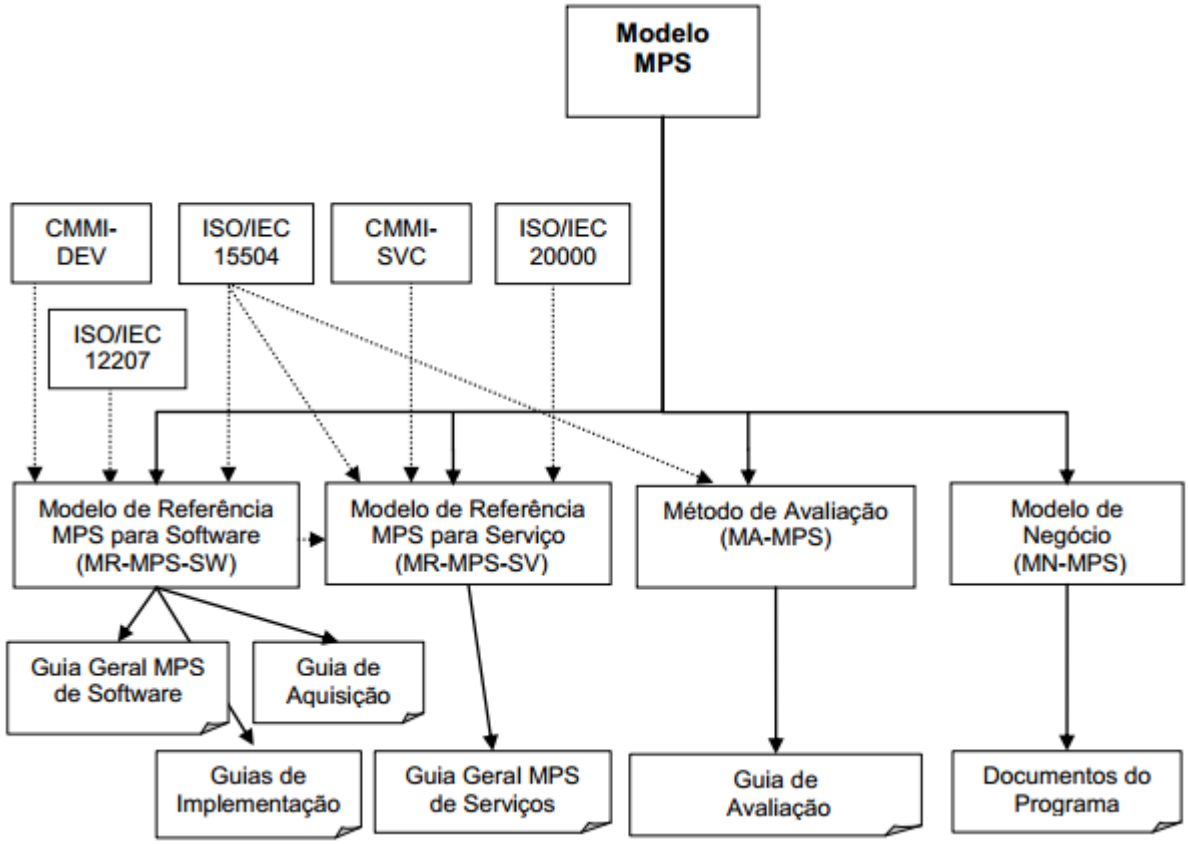

Figura 2 – Estrutura do Modelo MR-MPS (SOFTEX[1], 2012)

Segundo a Softex, “o MR-MPS-SW contém os requisitos que os processos das unidades organizacionais devem atender para estar em conformidade com o MR-MPS-SW. Ele contém as definições dos níveis de maturidade, processos e atributos de processo”. Empresas que adotam o modelo MR-MPS podem ser avaliadas em 7 (sete) níveis de maturidade, que vão do nível G (mais baixo) ao nível A (mais alto), sendo o segundo o melhor nível a ser alcançado, através da melhoria dos processos já implementados na organização (SOFTEX [1], 2012). O Quadro 1 apresenta os níveis de maturidade e os processos do modelo MR-MPS.

Quadro 1 – Processos e Níveis do MR-MPS (SOFTEX [1], 2012)

Nível	Denominação	Processos
A	Em Otimização	-
B	Gerenciado quantitativamente	Gerência de Projetos – GPR (Evolução)
C	Definido	Gerência de Riscos – GRI Desenvolvimento para Reutilização – DRU Gerência de Decisões – GDE
D	Largamente Definido	Verificação – VER Validação – VAL Projetos e Construção do Produto – PCP Integração do Produto – ITP Desenvolvimento de Requisitos – DRE
E	Parcialmente Definido	Gerência de Projetos – GPR (Evolução) Gerência de Reutilização – GRU Gerência de Recursos Humanos – GRH Definição do Processo Organizacional – DFP Avaliação e Melhoria do Processo Organizacional – AMP
F	Gerenciado	Medição – MED Garantia da Qualidade – GQA Gerência de Portfólio e Projetos – GPP Gerência de Configuração – GCO Aquisição - AQU
G	Parcialmente Gerenciado	Gerência de Requisitos – GRE Gerência de Projetos - GPR

Um dos processos apresentados no MR-MPS-SW é o processo de Garantia da Qualidade (GQA), que é encontrado no nível F (nível Gerenciado). “O propósito do processo Garantia da Qualidade é assegurar que os produtos de trabalho e a execução dos processos estão em conformidade com os planos e recursos predefinidos” (Softex [2], 2011).

O processo de GQA é composto por 4 (quatro) resultados esperados, segundo o Guia de Implementação do nível F (SOFTEX [2], 2011), estes são apresentados no Quadro 2.

Quadro 2 – Resultados Esperados do Processo de GQA (SOFTEX ², 2011)

GQA 1	A aderência dos produtos de trabalho aos padrões, procedimentos e requisitos aplicáveis é avaliada objetivamente, antes dos produtos serem entregues e em marcos predefinidos ao longo do ciclo de vida do projeto.
GQA 2	A aderência dos processos executados às descrições de processo, padrões e procedimentos é avaliada objetivamente.
GQA 3	Os problemas e as não conformidades são identificados, registrados e comunicados.
GQA 4	Ações corretivas para as não conformidades são estabelecidas e acompanhadas até as suas efetivas conclusões. Quando necessário, o escalamento das ações corretivas para níveis superiores é realizado, de forma a garantir sua solução.

Segundo Softex ² (2011)

> A Garantia da Qualidade é um apoio para o gerente, servindo como seus "olhos e ouvidos". Também agrega valor à equipe de projeto, ajudando-a a preparar e rever procedimentos, planos e padrões, desde o início do projeto até o seu encerramento. A pessoa ou grupo que executa a atividade de garantir a qualidade de processos e produtos tem uma responsabilidade delicada, pois fiscaliza se as pessoas estão desempenhando adequadamente as suas tarefas e seguindo os procedimentos estabelecidos. Por isso, pode ser necessário instituir mecanismos na organização que permitam aos responsáveis pelas atividades de Garantia da Qualidade executar seu trabalho com independência e autoridade.

A partir dos resultados esperados do processo de GQA e do parágrafo anterior, pode-se inferir que uma ferramenta para suporte a este processo deve ser capaz de prover critérios objetivos para avaliação de processos, conforme cronograma de auditorias alinhado ao cronograma de projeto, também mantidos pela ferramenta. Deve ser possível a abertura de não conformidades onde forem encontradas falhas com relação aos padrões organizacionais ou produtos de trabalho, sempre permitindo a visibilidade e autonomia da equipe de garantia da qualidade e comunicação com os responsáveis.

2.1.2 O MODELO DE REFERÊNCIA CMMI-DEV

Este modelo tem como objetivo principal promover um conjunto de melhores práticas, para que, através destas, as organizações interessadas possam melhorar seus processos de desenvolvimento e manutenção, abrangendo práticas que vão desde a concepção até a entrega e manutenção (SEI, 2010).

O CMMI-DEV é composto por duas representações: a Contínua e a por Estágios. A primeira é focada em melhoria de processos específicos, permitindo que se melhore diferentes processos com determinada ênfase para cada um de acordo com as necessidades da organização. A segunda apresenta uma forma sistemática de se melhorar os processos, todos os processos de um determinado nível de maturidade são implantados de uma só vez na organização.

A representação contínua é obtida através de níveis de “Capacidade”, enquanto a representação por estágios é representada pelos níveis de “Maturidade”. O Quadro 3 apresenta a relação entre os níveis de Capacidade e Maturidade existentes no CMMI-DEV.

Quadro 3 – Comparação entre Níveis de Capacidade e Maturidade (SEI, 2010)

Nível	Representação Contínua (Níveis de Capacidade)	Representação por Estágios (Níveis de Maturidade)
Nível 0	Incompleto	
Nível 1	Executado	Inicial
Nível 2	Gerenciado	Gerenciado
Nível 3	Definido	Definido
Nível 4		Quantitativamente Gerenciado
Nível 5		Otimizado

Como abordado no parágrafo anterior, os níveis de Maturidade são compostos por áreas de processos, que devem ser implementadas para que a organização alcance cada nível. Estes são apresentados no Quadro 4.

Quadro 4 – Processo por Níveis de Maturidade (SEI, 2010)

Nível de Maturidade	Processos
1	-
2	Gestão de Configuração - GC Medição e Análise - MA Monitoramento e Controle de Projeto - MCP Planejamento de Projeto - PP Garantia de Qualidade de Processo e Produto - GQPP Gestão de Requisitos - GR Gestão de Contrato com Fornecedores - GCF
3	Análise e Tomada de Decisões - ATD Gestão Integrada de Projeto - GIP Definição dos Processos da Organização - DPO Foco nos Processos da Organização - FPO Treinamento na Organização - TO Integração de Produto - IP Desenvolvimento de Requisitos - DR Gerenciamento de Riscos - GR Solução Técnica - ST Validação - VAL Verificação - VER
4	Desempenho dos Processos da Organização - DPO Gestão Quantitativa de Projeto - GQP
5	Análise e Resolução de Causas – ARC Inovação e Implantação Organizacional - IIO

Dentre os processos apresentados em cada nível, percebe-se no nível 2 a existência de “Garantia de Qualidade de Processo e Produto” (GQPP). Este processo é aderente ao processo de Garantia da Qualidade, encontrado no nível F do MR-MPS, apresentando os mesmos objetivos. Segundo o SEI (2010), a área de processo PPQA objetiva fornecer visibilidade para a equipe e gerência sobre os processos e seus produtos. As metas e práticas da área de processo são exibidas no Quadro 5.

Quadro 5 – Metas e Práticas do Processo de Garantia da Qualidade de Processo e Produto (SEI, p.303, 2006)

Metas	Práticas
SG1. Avaliar Objetivamente Processos e Produtos de Trabalho	SP 1.1 Avaliar Objetivamente os Processos
	SP 1.2 Avaliar Objetivamente Produtos de Trabalho e Serviços
SG2. Fornecer Visibilidade	SP 2.1 Comunicar e Assegurar a Solução de Não conformidades
	SP 2.2 Estabelecer Registros

2.2 FERRAMENTAS CASE

Segundo Silva e Videira (2001), ferramentas CASE são definidas como

> (...) um conjunto de técnicas e ferramentas informáticas que auxiliam o engenheiro de software no desenvolvimento de aplicações, com o objetivo de diminuir o respectivo esforço e complexidade, de melhorar o controle do projeto, de aplicar sistematicamente um processo uniformizado e de automatizar algumas atividades, nomeadamente a verificação da consistência e qualidade do produto final e a geração de artefatos.

A partir desta definição, percebe-se a importância da aplicação de ferramentas CASE no processo de desenvolvimento de software, já que as mesmas têm como objetivo incrementar a produtividade, melhorar o controle e uniformizar as atividades do processo escolhido. De acordo com Silva e Videira (2001), quando se usa uma ferramenta CASE ou uma ferramenta apoiando um processo de engenharia de software, o resultado esperado é um produto final e artefatos intermediários de melhor qualidade.

Ferramentas CASE podem ser usadas em qualquer processo do ciclo de vida de um projeto de software, desde o gerenciamento até a própria implementação ou programação, abrangendo até mesmo a manutenção (TERRY, 1990).

Segundo Silva e Videira (2001), a maioria das ferramentas CASE são desenvolvidas com foco em uma tarefa ou processo específico do desenvolvimento de software, sendo assim, para cada fase do ciclo de vida de um projeto de software, podem ser usadas diversas ferramentas, não apenas do mesmo fornecedor, o que gera a necessidade de desenvolvimento de mecanismos para troca de informações (integração) entre todas elas.

A integração das ferramentas CASE pode ser de diversas maneiras, as mais comuns são através de repositório ou através de exportação de dados por formatos comuns ou universais (SILVA e VIDEIRA, 2011). O repositório abordado pode ser o mesmo configurado e mantido no processo de Gerência de Configuração (GCO), encontrado no nível F do MR-MPS, conforme observado no Quadro 1 da subseção 2.1.1 (O Modelo de Referência MR-MPS) deste mesmo capítulo.

Ferramentas CASE podem ser divididas em categorias: (i) Modelagem de Processos de negócio; (ii) Modelagem de análise e desenho do sistema; (iii) Desenho de bases de dados; (iv) Programação de aplicações; (v) Gestão de alterações no software; (vi) Testes; e (vii) Orientadas para a Gestão de Projetos (SILVA E VIDEIRA, 2011).

Para o contexto deste trabalho, considerando o desenvolvimento de uma aplicação para suporte ao processo de Garantia da Qualidade, esta ferramenta se enquadraria na categoria "(vii) Orientadas para a Gestão de Projetos", que, segundo Silva e Videira (2011):

> São ferramentas cujas principais funcionalidades se destinam a facilitar as tarefas de gestão e coordenação dos projetos, com recurso a técnicas tais como o planejamento e estimativa de tempos, custos e recursos, a utilização e influência de recursos ao projeto, definição de responsabilidades. (...) (muitas vezes disponibilizando informação sobre melhores práticas, casos de estudo, e uma knowledge base sobre todo o processo).

2.3 FRAMEWORKS

Segundo Fayad e Schmidt (1997), "um framework é um aplicativo 'semi-completo' reutilizável que pode ser especializado para produzir aplicações personalizadas", que tem como base a utilização dos conceitos de Orientação a Objetos, como por exemplo, o uso de biblioteca de classes.

Quando se aplica um Framework, objetiva-se conseguir interfaces estáveis, melhor qualidade de software, redução de esforço de desenvolvimento, compreensão e manutenção de sistemas (FAYAD e SCHMIDT, 1997). Para o contexto deste trabalho, será usado o framework CodeIgniter que será apresentado na subseção 2.3.1 (CodeIgniter v2.1.3), a seguir.

2.3.1 CodeIgniter v2.1.3

O *CodeIgniter* é um framework para desenvolvimento de Aplicações que utilizam a linguagem PHP, gratuito e licenciado sob a licença *Apache/BSD-style*. Segundo Ellislab (2012), a empresa responsável pela sua criação:

> Seu objetivo é permitir que você desenvolva projetos mais rapidamente do que se você estivesse escrevendo código a partir do zero, através de um conjunto de bibliotecas para as tarefas mais comuns necessárias, bem como uma interface simples e estrutura lógica para acessar essas bibliotecas.

O uso deste framework fornece (i) o desempenho excepcional das aplicações; (ii) configuração simples; (iii) simplicidade de soluções; e (iv) documentação clara, além de outras vantagens apresentadas (ELLISLAB, 2012). Importante ressaltar que toda a documentação do Framework encontra-se disponível para consulta.

Para a aplicação deste framework se faz necessário apenas um servidor de aplicações PHP (versão 5.1.6 ou superior) e um servidor de banco de dados (*MySQL (4.1 +), MySQLi, MS SQL, PostgreSQL, Oracle, SQLite* ou *ODBC*), segundo o próprio manual do usuário (ELLISLAB, 2012).

No contexto deste projeto, a utilização deste framework favorece o desenvolvimento de funcionalidades principalmente ligadas à persistência dos dados, tendo em vista que a interação com base de dados se torna simplificada. Através de seu uso, é favorecida a aplicação dos estilos de arquitetura Orientada a Objetos e Cliente-Servidor, além da aplicação do padrão de arquitetura MVC (*Model-View-Controller*) (apresentados com detalhes nas seções 2.4 e 2.5 deste capítulo).

2.4 ESTILOS DE ARQUITETURA

Estilos de Arquitetura são definidos por Taylor, Medvidovid e Dashofy (2009) como decisões de projetos de arquitetura aplicáveis a dado contexto de desenvolvimento, com o intuito de incorporar ao projeto qualidades benéficas, restringindo decisões específicas, possibilitando reuso de projeto, reuso de código, melhor compreensão de organização e melhor visualização. Para o contexto deste projeto, serão aplicados os estilos: (i) Cliente-Servidor; (ii) Código Móvel.

2.4.1 Estilo Cliente-Servidor

Este estilo está presente sempre que clientes realizam requisições a serviços providos a partir de um servidor, sendo aconselhado seu uso quando há a centralização de dados em um único local (o servidor), favorecendo a escalabilidade de recursos e o gerenciamento de dados, bem como o processamento por parte do cliente, que se restringe basicamente à apresentação de dados, segundo Taylor, Medvidovic e Dashofy (2009).

2.4.2 Estilo Código Móvel

Quando há a aplicação deste estilo, o código ou parte deste é movido até um “host” remoto, onde é realizado o seu processamento, sendo este indicado quando o processamento é mais eficiente quando transferido para outro responsável. Um exemplo de aplicação deste estilo é através do uso da linguagem Javascript (TAYLOR, MEDVIDOVIC, DASHOFY, 2009).

2.5 PADRÕES DE ARQUITETURA

Padrões de arquitetura podem ser definidos como um conjunto de decisões tomadas durante um projeto de arquitetura, decisões estas que são aplicáveis a problemas recorrentes e parametrizados lidando com diferentes contextos em que os problemas são encontrados, segundo Taylor, Medvidovid e Dashofy (2009).

Dentro do contexto deste trabalho, será aplicado o padrão *Model-View-Controller*, apresentado na subseção 2.5.1, a seguir.

2.5.1 Padrão *Model-View-Controller* (MVC)

Este padrão tem como objetivo separar a informação, apresentação e processamento de dados, de maneira que, cada um destes itens permaneça em uma camada distinta, que se comunicam através de conectores entre as camadas e seus componentes. A aplicação deste padrão favorece a manutenção e evolução dos sistemas que o aplicam, bem como a compreensão das regras de negócio e fluxos da aplicação.

O padrão funciona da seguinte maneira: (i) o usuário aciona ou inicia um evento; (ii) o evento é encaminhado para a Controladora responsável; (iii) a Controladora realiza o processamento necessário; (iv) o resultado do processamento

por parte da Controladora pode ser repassado para uma entidade da camada de Visão, sendo exibida para o usuário ou, este processamento pode ser repassado para uma entidade da classe de Modelo, onde havendo ou não persistência, são retornados dados solicitados. (v) a Controladora pode processar os dados recebidos da entidade da camada de Modelo ou simplesmente repassar para a camada de Visão; (vi) os dados são exibidos para o usuário. A Figura 3 apresenta um esquema representativo do padrão MVC.

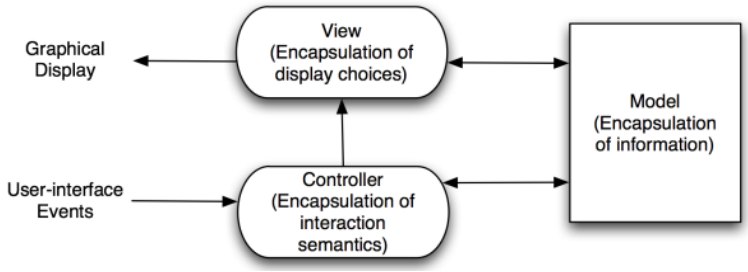

Figura 3 – Esquema representativo do padrão MVC (Taylor 2009)

2.6 PADRÕES DE PROJETO

Quando se fala em Padrões de Projeto, segundo Gamma *et. al*. (2006), “os padrões de projeto são descrições de objetos que se comunicam e classes que são customizadas para resolver um problema genérico de design em um contexto específico”, e, ainda segundo estes autores, quando se desenvolve um software seguindo padrões de projetos, melhora-se a qualidade e ainda ajuda-se na documentação.

Ao longo do projeto QUASAR, foi aplicada uma série de padrões de projeto, mais especificamente, padrões GRASP (*General Responsibility Assignment Softawre Patterns*). Os padrões aplicados serão apresentados nas subseções seguintes e são eles: (i) Baixo Acoplamento; (ii) Alta Coesão; (iii) Indireção; (iv) Especialista da Informação; (v) Controlador; e (vi) Polimorfismo.

A aplicação destes padrões está relacionada principalmente ao fato de ter sido escolhido o paradigma de Orientação a Objetos.

2.6.1 Padrão Baixo Acoplamento

Para um projeto de software orientado a objetos, espera-se que os objetos que compõem o domínio da aplicação tenham entre si sempre o menor acoplamento

possível, fator que impacta diretamente na independência que cada classe tem para realizar suas atribuições (LARMAN, 2007).

Quanto mais baixo for o acoplamento entre as classes de um sistema, melhor e mais simples se dá as atividades relacionadas à manutenção, refatoração e reuso do software (LARMAN, 2007).

2.6.2 Padrão Alta Coesão

Este padrão está diretamente ligado ao acoplamento entre as classes (apresentado na subseção 2.6.1, Padrão Baixo Acoplamento), sendo assim, maior é a coesão das classes que compõem o sistema, menor será o acoplamento entre estas, ou seja, quanto mais focados, inteligíveis e gerenciáveis são os objetos, maior a coesão e menor o acoplamento (LARMAN, 2007).

2.6.3 Padrão Indireção

Este padrão é aplicado quando se deseja reduzir o acoplamento entre duas classes. É inserida, assim, uma camada intermediária, responsável por realizar a comunicação entre as classes que, antes, possuíam forte acoplamento, criando-se uma “indireção” entre as classes. A idéia principal é proteger a classe cliente de futuras variações nos objetos. Funciona como uma interface clara (LARMAN, 2007).

2.6.4 Padrão Especialista da Informação

Quando se distribui responsabilidades para as classes que compõem o projeto, este tende a ser mais inteligível. O padrão especialista possibilita a melhor compreensão do projeto e das responsabilidades, aumentando oportunidades de reuso (LARMAN, 2007).

Importante observar se os objetos realizam manipulações baseadas nos dados que possui, podendo haver também, especialistas parciais, para o caso de informações dispersas por diversas classes de objetos (LARMAN, 2007).

2.6.5 Padrão Controlador

Padrão de extrema importância, pois trata os objetos que receberão as solicitações dos usuários e irão realizar o controle do acionamento das funções, recebendo e tratando as mensagens do usuário, importante observar que, sua

aplicação evita que interfaces de usuário realizem processamentos que não são de sua responsabilidade (LARMAN, 2007).

2.6.6 Padrão Polimorfismo

Padrão que visa colaborar fortemente para o reuso de software, favorecendo atividades de manutenção, principalmente manutenção evolutiva, sem que as classes clientes de informações possam ser afetadas. Seu uso implica na implantação de classes abstratas ou interfaces (LARMAN, 2007).

CAPÍTULO 3 – FERRAMENTAS RELACIONADAS

3.1 INTRODUÇÃO

Antes de propor o desenvolvimento de uma nova ferramenta para apoio ao processo de Garantia da Qualidade, se fez importante o estudo de ferramentas relacionadas, ou seja, ferramentas existentes com o mesmo objetivo geral ou objetivos semelhantes ao software QUASAR, proposto neste trabalho.

Na seção 3.2 serão apresentadas as ferramentas: (i) Estação TABA; (ii) AvalPro; (iii) SeAdudit; (iv) Ferramenta GQA. Todas estas ferramentas estão envolvidas de alguma maneira com o processo de Garantia da Qualidade e serão apresentados seus objetivos, propósitos e público alvo.

3.2 SELEÇÃO DE FERRAMENTAS

Para seleção das ferramentas, primeiramente optou-se pela busca dentro do escopo de ferramentas *OpenSource*, mais especificamente no maior repositório de ferramentas desta categoria, o *SourgeForge*. O portal não apresentava nenhuma ferramenta com foco na Garantia da Qualidade, ou ferramentas de gestão de projetos que tivessem em seu escopo funcionalidades para auditoria de qualidade dos processos e produtos organizacionais, fator este que eliminou o *SourgeForge* do escopo de pesquisa.

Foram buscados, anais de congressos de qualidade, como o Simpósio Brasileiro de Qualidade de Software (SBQS), onde houvesse apresentação de novas ferramentas com o intuito de contribuir com a qualidade de processos e produtos de software.

Nos anais do IV SBQS, realizado em 2005, foi encontrada a apresentação da Estação TABA, apresentada por ROCHA *et al.* (2005). Através da apresentação da estação, foi encontrada a ferramenta AvalPro, proposta por ANDRADE (2005) apresentada e disponibilizada no site de divulgação da Estação TABA.

A fim de encontrar mais ferramentas que tratassem do assunto Garantia da Qualidade, artigos científicos e trabalhos foram pesquisados. Neste contexto, foi encontrada a Ferramenta GQA, nome dado no contexto deste projeto ao trabalho de conclusão de curso desenvolvido pelo graduando em Ciências da Computação, Anildo Loos, da Universidade Regional de Blumenau (LOOS, 2009). Esta ferramenta foi escolhida para compor o escopo deste trabalho pelo fato de se tratar de um dos

poucos trabalhos acadêmicos em nível de graduação que, assim como o projeto QUASAR, se propõem ao desenvolvimento de ferramenta de apoio ao processo de GQA.

Ainda buscando ferramentas passíveis de avaliação, optou-se pela inserção de uma ferramenta de mercado, usada não apenas pela engenharia de software, mas sim, por qualquer ramo da engenharia onde se deseja aplicar a avaliação de processos e auditoria interna de qualidade. Sendo assim, com estes critérios, foi encontrada a ferramenta SE Audit, desenvolvida pela SoftExpert, empresa catarinense de desenvolvimento de software para organizações de todos os portes e ramos, com representantes de venda e suporte em todo o mundo, em mais de 25 (vinte e cinco) países.

3.2.1 Estação TABA

A estação de trabalho TABA (Rocha 1990) é um meta-ambiente que possibilita através de configuração e instanciação a criação de ambientes de desenvolvimento para a Engenharia de Software, que se adéquam aos processos de desenvolvimento e projetos selecionados (Rocha 1990). Segundo Rocha (1990), um meta-ambiente é definido como um conjunto de programas que interagem com os usuários para estabelecer os tipos de objeto que irão compor o ambiente de desenvolvimento, além de definir interfaces e selecionar ferramentas.

Através da estação TABA, podem ser instanciados ambientes de trabalho de software orientados a organizações (ADSOrg), que podem ser personalizados de acordo com o processo e a organização em questão.

A estação apresenta diversos módulos já instanciados para ambiente de trabalho, como por exemplos:

- MedPlan: definição de base de métricas para a Estação TABA, elaboração dos planos de medição, baseando-se no método GQM (Goal / Question / Metric). Fornece ao usuário o conhecimento sobre objetivos, questões e métricas e procedimentos de coleta, armazenamento e análise de dados (Schnaider 2003);
- CustPlan: desenvolvido com o objetivo de apoiar processos de estimativa de tempo e custo de projetos de software, baseando-se no cronograma e recursos humanos previamente alocados (Barcellos 2003);

- Acknowledge: desenvolvido com o objetivo de permitir o registro do conhecimento adquirido pelo usuário durante as atividades do projeto (Montoni 2003);
- RHPlan: desenvolvido com o objetivo de apoiar as atividades de planejamento e alocação de recursos humanos em um projeto (Schnaider 2003);
- GConf: apoiar atividades de planejamento da Gerência de Configuração, Liberação e Distribuição e Controle da Configuração, (Figueiredo 2004);

Além destes módulos, existe uma vasta quantidade de módulos para processos distintos, baseados em MPS-BR e CMMI-DEV. Sendo assim, para o contexto de Garantia da Qualidade, também foi instanciado e implementado um ambiente de trabalho. É o caso do ambiente AvalPro, que será apresentado na subseção 3.2.2.

3.2.2 AvalPro

O AvalPro tem como objetivo auxiliar nas atividades de GQPP – Garantia da Qualidade de Processo e Produto, encontrada no CMMI-DEV nível 2, equivalente ao processo de GQA – Garantia da Qualidade do nível F do MR-MPS. Trata-se de uma das muitas ferramentas criadas em ADSOrg – Ambiente de Desenvolvimento Organizacional, com base na Estação TABA, apresentada na subseção 3.2.1 (ANDRADE 2005).

Para a correta utilização da ferramenta, é necessário que a empresa esteja inserida no contexto de uso de ADSOrg, uma vez que diversos recursos são configurados em outras ferramentas, por exemplo, para a avaliação de processos é necessário que hajam métricas cadastradas previamente nas bases da empresa, estas métricas são gerenciadas pela ferramenta MedPlan (citada na seção 3.2.1).

A execução dos processos de GQPP na ferramenta AvalPro segue o fluxo de trabalho apresentado na Figura 4.

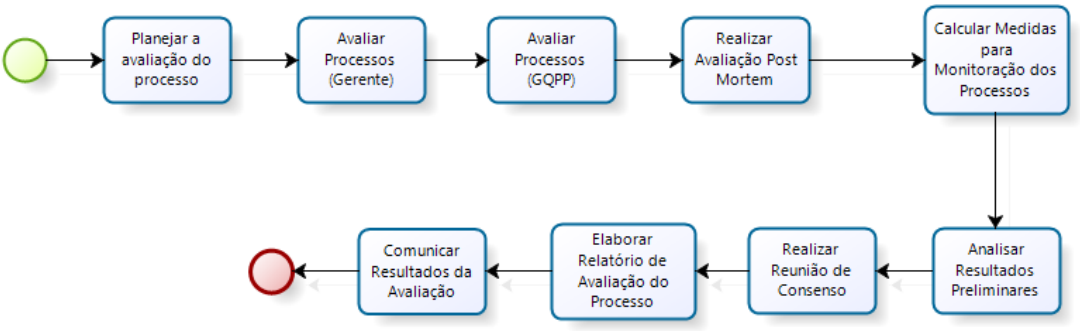

Figura 4 - Processo de GQPP segundo o AvalPro (fluxo definido a partir das descrições de Andrade (2005) e experiência de uso da ferramenta)

O processo inicia com o Planejamento da Avaliação do Processo, nesta atividade o objetivo é a criação do cronograma para as atividades de avaliação do processo no projeto. Esta criação deve ser feita através da ferramenta TempPlan (BARCELLOS 2003), integrada através do ambiente. Esta ação possibilita que o cronograma da equipe de GQPP esteja alinhado com o cronograma do projeto.

A próxima atividade do processo é “Avaliar Processos” (há durante o processo de execução duas atividades de avaliação de processos, esta primeira é executada pelo Gerente de Projetos). Nesta etapa, o Gerente deve avaliar a adequação do processo de acordo com a macro-atividade sendo avaliada (pode ser avaliado como Totalmente adequado, Largamente adequado, Parcialmente adequado e Não adequado) e se a equipe foi aderente ao processo durante a macro-atividade (sendo avaliada como Totalmente aderente, Largamente aderente, Parcialmente aderente ou Não aderente). Esta avaliação é feita a partir de outra ferramenta do ambiente, o “ProjectStatus”.

Adiante se tem a atividade de “Avaliar Processos” agora sendo realizada pela equipe de GQPP, que nesta etapa faz uso de *checklists* para verificação da aderência do processo. Cada *checklist* contém um conjunto de macro-atividades que compõem uma área de processo e é completado de maneira incremental conforme as atividades são realizadas, ou seja, é usado o mesmo *checklist* em todas as etapas. Nesta etapa podem ser anexados identificadores, que confirmam a realização da atividade. Esta atividade é realizada dentro do próprio AvalPro. Importante ressaltar que os *checklists* são documentos (geralmente de editores de texto comuns) anexados à atividade.

Em seguida a atividade de “Realizar avaliação *post mortem*”, que é uma avaliação realizada ao final do projeto, contando com a participação do gerente,

líderes de desenvolvimento e desenvolvedores. Dentro do AvalPro, esta atividade é dividida nas tarefas de “Identificar Avaliadores”, onde a lista de avaliadores é recuperada para que os mesmos possam ser notificados da avaliação. Ainda nesta atividade podem-se analisar cada questionário individual de funcionário. É possível registrar qualquer fato através de campo para observações.

Através da atividade de “Calcular medidas para monitoração dos processos”, as medidas coletadas são armazenadas para serem tratadas na atividade de análise de resultados.

Na atividade de “Análise de resultados preliminares”, são analisados os problemas relacionados à adequação do processo e do projeto e problemas relacionados à aderência da equipe ao processo definido. É possível analisar projetos similares e causas destes.

Em seguida aparece a atividade de “Realizar Reunião de Consenso”, onde o foco é discutir os problemas, causas e oportunidades de melhoria, havendo consenso entre os fatos ocorridos.

Após a reunião de consenso, é executada a atividade de “Elaborar o Relatório de Avaliação do Processo Instanciado”, onde é gerado o relatório final de avaliação a partir dos dados presentes no AvalPro.

Por fim, os resultados da avaliação são comunicados à alta gerência, onde a avaliação é encerrada e são documentados os membros da alta gerência que foram comunicados dos resultados e se houve alguma diretriz ou meta apresentada por eles para o tratamento dos problemas.

Uma desvantagem encontrada nesta ferramenta é o fato de que a mesma necessita de todo o ambiente de desenvolvimento TABA, previamente configurado e pronto para uso, com os módulos necessários instalados (manutenção de cronograma, métricas, gestão de RH, entre outros). Este fato impossibilita o uso isolado de tal ferramenta, demandando esforço de configuração das outras ferramentas.

3.2.3 Ferramenta GQA

Foi proposta e desenvolvida no trabalho de graduação do estudante Anildo Loos, da Universidade Regional de Blumenau, onde ele propõe o desenvolvimento de uma ferramenta que tem como objetivo auxiliar o processo de GQA aderente ao modelo MR-MPS (versão 1.2). As suas principais funcionalidades são:

- Permitir o Cadastro de Projetos;
- Permitir o Cadastro de Processos;
- Permitir o Cadastro de Padrões da Organização;
- Permitir o registro de Critérios de Avaliação, *checklists* e questões;
- Permitir registro de não conformidades;
- Gerar relatórios de acompanhamento;

O desenvolvimento desta ferramenta foi realizado com foco em desktop, ou seja, a mesma seria instalada em uma máquina, de onde a equipe ou responsável controlariam e acompanhariam os processos de GQA, ou seja, não foi projetada para ambiente distribuído. A Figura 5 apresenta o fluxo das atividades da Ferramenta GQA, explicadas anteriormente.

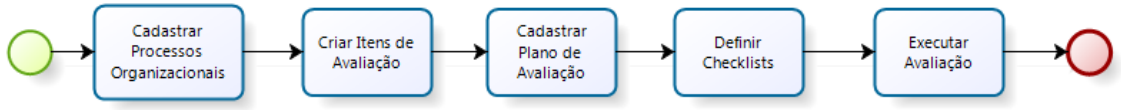

Figura 5 - Fluxo de atividades do Software GQA (fluxo definido a partir de descrições de Loos (2009) e visualização de telas fornecidas)

Primeiramente se faz necessário o cadastro dos processos da organização. Sempre que um novo processo é cadastrado, o mesmo faz referência a determinada disciplina (Arquitetura, Requisitos, Teste entre outras) que também podem ser mantidas pela ferramenta. Ao selecionar a qual disciplina o novo processo pertence, inicia-se o cadastro das tarefas deste processo. Para cada tarefa cadastrada, podem ser cadastrados passos. Sendo assim, os processos mantidos pela ferramenta possuem profundidade de 3 níveis (Disciplina, Tarefa e Passos). Ainda durante o cadastro de processos é possível cadastrar os artefatos.

Após realizar o cadastro dos processos, realiza-se a elaboração do plano de avaliação, sendo que este conta com Nome, Descrição, Responsável e itens de avaliação. Para se cadastrar o plano de avaliação, previamente é necessário que hajam itens de avaliação cadastrados. Importante ressaltar que não é possível criar um cronograma de auditorias, não há campos para definição de datas.

Em seguida, a execução das avaliações. Nesta etapa, a avaliação é realizada com base no *checklist* previamente definido, inserindo a cada item do *checklist* o Resultado (Não Atende ou Atende), a Prioridade (baixa, media ou alta) e

o Status (Em Correção, Corrigido ou Encerrado). Não foi identificada a possibilidade de personalizar estas opções.

Ao se executar uma avaliação, podem-se encontrar não conformidades e estas podem ser registradas. Quando se registra uma não conformidade, surge a opção para envio de e-mail para o responsável pela solução da mesma. O status das não conformidades pode ser atualizado pela equipe conforme respostas dos responsáveis. Ações tomadas para solução destas não são documentadas (armazenamento de histórico), apenas seu atual status é registrado.

Durante todo o projeto da ferramenta, percebe-se que o foco da mesma estava no uso restrito da equipe de Garantia da Qualidade, com foco em dois perfis diferentes: Gerente de Qualidade e Assistente da Qualidade, sendo esta, impossibilitada de ser aberta a todos os empregados da organização, inclusive aos responsáveis pela solução das não conformidades.

3.2.4 SE Audit (SOFTEXPERT)

O SE Audit é um software desenvolvido pela SoftExpert com o objetivo de Gerenciar Auditorias. Importante ressaltar que o mesmo não se restringe apenas ao setor de Tecnologia da Informação, mas auditorias em qualquer segmento empresarial. Para direito de uso é preciso comprar sua licença (SOFTEXPERT). O SE Audit foi projetado com base no uso do estilo de arquitetura Cliente-Servidor (apresentada no Capítulo 2, subseção 2.4.2, Estilo Cliente-Servidor), sendo executado em ambiente web, o que possibilita o acesso de qualquer máquina da organização sem previa instalação na estação de trabalho, sendo esta uma aplicação distribuída.

O software em questão permite seguir o fluxo de Planejamento, Preparação, Execução, Conclusão e Monitoramento, através das funcionalidades apresentadas na Figura 6:

Figura 6 - Fluxo de trabalho SE Audit (SOFTEXPERT)

A etapa de parametrização de auditorias permite o cadastro dos requisitos de auditorias, notas que podem ser usadas como dicas para os auditores e cadastro de critérios de auditoria. São chamados critérios de auditoria as normas selecionadas como base para avaliação. Ainda nesta etapa é possível realizar o cadastro dos níveis de conformidade, por exemplo, boa pratica, situação normal, oportunidade de melhoria e não conformidade.

Na atividade de Planejamento de Auditoria, são definidos: descrição, objetivos, escopo e é associado um ou mais critérios de avaliação, ou seja, uma ou mais normas. É possível também, cadastrar passos e as etapas definindo prazos e responsáveis para cada um destes. Podem-se associar às Auditorias, os processos da organização previamente cadastrados e também os documentos que serão alvos de auditoria.

A preparação da auditoria consiste na definição de datas, cronograma de auditorias, descrevendo quem serão os funcionários, processos e departamentos auditados. Pode ser gerado dinamicamente um cronograma, diário ou semanal apresentando as etapas da auditoria programadas.

A seguir, é realizada a atividade de Execução da Auditoria, onde podem ser registradas as evidências, constatações, nível de conformidade e comentários referentes a cada requisito auditado. Nesta etapa, a avaliação é feita diretamente a partir da norma, avaliando os itens contidos nesta e os requisitos cadastrados previamente.

Partindo para a Finalização da Auditoria, o responsável pode emitir pareceres sobre todo o processo de auditoria, visualizar e imprimir relatórios.

Por fim, o processo de Acompanhamento de Auditoria, que permite visualizar, alterar, cancelar/reativar, excluir, copiar e reabrir/retornar uma auditoria a uma etapa anterior. Nesta etapa também é possível realizar pesquisas e realizar gráficos analíticos.

3.3 DISCUSSÃO E PROPOSTA DE UMA NOVA FERRAMENTA

Através do uso e estudo das ferramentas de auditoria de qualidade citadas na seção 3.2 (Seleção de Ferramentas) deste capítulo, notou-se uma oportunidade de melhoria em todas as ferramentas: seria interessante a existência de uma funcionalidade que auxiliasse o Gerente de Projetos na rastreabilidade de não conformidades com práticas (ou resultados esperados) do modelo de referência

escolhido. Esta funcionalidade possibilitaria ao Gerente de Projetos identificar rapidamente as não conformidades que impossibilitam a organização de conseguir determinado nível de maturidade em uma avaliação do modelo. Vale ressaltar que esta apresenta grande valor para as empresas que almejem um melhor nível de maturidade seja com relação ao MR-MPS, CMMI-DEV ou qualquer outro modelo de melhoria que disponibilize classificações.

Neste contexto, por exemplo, se uma empresa já possui nível G MPS e deseja alcançar o nível F, a mesma pode realizar seu processo de Garantia da Qualidade de maneira a auditar todos os processos que atendem aos processos do nível F MPS. Desta forma, a partir das não conformidades encontradas, o Gerente de Projetos consegue observar quais resultados do nível F são impactados por estas não-conformidades.

A partir da análise das ferramentas, percebe-se a oportunidade de desenvolvimento de uma nova ferramenta pelo fato de terem sido encontradas características em cada uma das demais que impossibilitam a manutenção evolutiva das mesmas para inclusão desta nova funcionalidade. Estas características são: (i) a estação TABA e o AvalPro necessitam de todo um ambiente previamente configurado, com ferramentas específicas instaladas; (ii) a ferramenta GQA não apresenta o conceito de arquitetura Cliente-Servidor (apresentado na seção 2.4.2), que é o estilo mais adequado para o cenário das organizações atualmente; e (iii) o fato de o Se Audit se tratar de uma ferramenta proprietária, não permitindo manutenção evolutiva senão pela empresa desenvolvedora.

Importante ressaltar que para esta nova ferramenta é necessário organizar as informações de Modelos de Referência, Processos e Projetos de Clientes, Checklists de qualidade, para que assim possa ser possível realizar o mapeamento.

O processo de desenvolvimento da ferramenta QUASAR deve seguir as práticas da Engenharia de Software. As disciplinas de Engenharia de Software envolvidas e alvo deste projeto serão Modelagem de Negócios, Análise de Requisitos, Análise e Design, Preparação do Ambiente de Desenvolvimento, Gerência de Projeto, Desenvolvimento, Teste e Implantação. A execução destas disciplinas e os produtos de trabalho foram baseados nos artefatos e processos descritos no *Rational Unified Process* (RATIONAL, 2001), processo popular para o desenvolvimento de software orientado a objetos.

CAPÍTULO 4 - QUASAR: QUALIDADE DE SOFTWARE E AVALIAÇÃO DE RESULTADOS

4.1 INTRODUÇÃO

Em função da análise das ferramentas apresentadas na seção anterior e da demanda pela área de qualidade surge a oportunidade de proposta de uma nova ferramenta para Garantia da Qualidade: QUASAR– Qualidade de Software e Avaliação de Resultados, com a finalidade de suprir as necessidades não atendidas pelas ferramentas avaliadas.

Nas subseções que se seguem, será apresentado o cronograma previsto e realizado durante a gestão do projeto (seção 4.2), a descrição do negócio (seção 4.3), os requisitos e as funcionalidades da ferramenta (seção 4.4), aspectos relacionados à arquitetura do software (seção 4.5), configuração do ambiente de desenvolvimento (seção 4.6), descrição das técnicas e padrões de codificação (seção 4.7) e resultados dos testes (seção 4.8).

4.2 GERENCIAMENTO DE PROJETO

Segundo Pressman (p129, 1995)

> O tempo é o mais valioso bem disponível a um engenheiro de software. Se houver suficiente tempo disponível, um problema pode ser adequadamente analisado, uma solução pode ser compreensivamente projetada, o código-fonte, cuidadosamente implementado e o programa, minuciosamente testado.

A partir desta afirmação, percebe-se a necessidade de criação e acompanhamento de cronograma eficiente de desenvolvimento, organizando-se o tempo da maneira mais proveitosa possível. Sendo assim, para o projeto QUASAR, é apresentado o cronograma estipulado e que deve ser minuciosamente seguido, registrando-se os possíveis desvios.

O cronograma de projeto foi desenvolvido com o intuito de organizar o tempo disponível e dividir de acordo com a necessidade para as atividades necessárias para o desenvolvimento do software. Os quadros Quadro **6**, Quadro **7** e Quadro **8** apresentam o cronograma desenvolvido para as fases de Iniciação, Elaboração, Construção e Transição, respectivamente.

Quadro 6 – Cronograma Fase de Iniciação

Fase	Disciplina	Atividades	Dez				Jan			
			1	2	3	4	1	2	3	4
INICIAÇÃO	Modelagem de Negócio	Descrever Processos	■							
		Descrever Papéis e Responsabilidades	■							
		Desenvolver Modelo de Domínio		■						
	Requisitos	Levantar Necessidades		■	■					
		Desenvolver Diagrama de Casos de Uso			■					
		Validar Casos de Uso			■					
	Análise e Design	Descrever idéia inicial de Arquitetura						■	■	
	Ambiente	Selecionar e Identificar Ferramentas								■

Quadro 7 – Cronograma Fase de Elaboração

	Disciplina	Atividades	Março				Abril			
			1	2	3	4	1	2	3	4
ELABORAÇÃO	Requisitos	Detalhar Casos de Uso			■	■				
		Levantar URPS+				■	■			
		Validar Requisitos					■			
	Análise e Design	Projetar Banco de Dados			■					
		Desenvolver DSS e Contratos de Operação				■	■			
		Desenvolver Diagrama de Classes						■	■	
	Codificação	Desenvolver Núcleo do Framework							■	
		Desenvolver Classes de Persistência							■	■
	Testes	Desenvolver Testes Unitários							■	■
	Ambiente	Configurar Ferramentas			■	■	■	■		

Quadro 8 – Cronograma Fase de Construção

Fase	Disciplina	Atividades	Abil				Maio				Junho				Julho			
			1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
CONSTRUÇÃO	Análise e Design	Atualizar Diagrama de Classes				■												
		Desenvolver Diagrama de Sequência				■												
		Documentar Padrões				■												
	Codificação	Desenvolver Controladoras				■	■	■	■	■	■							
		Desenvolver Visões					■	■	■	■	■	■	■	■				
		Desenvolver Conectores (Javascripts)						■	■	■	■	■	■	■				
	Testes	Atualizar Testes Unitários									■	■	■	■	■	■		
		Executar Revisão em Pares											■	■				

Para o desenvolvimento do software, se fez necessário projetar e incluir no cronograma atividades executadas em paralelo. A organização das atividades paralelas seguiu o Processo Unificado, observando-se as disciplinas e fases de cada etapa de desenvolvimento.

Algumas atividades como o Acompanhamento do Cronograma, Configuração e Manutenção de Ambiente, são sempre executadas independentemente da fase em que o projeto se encontra, por se tratarem de atividades fundamentais e necessárias durante todo o ciclo de vida.

Existem atividades que não podem ser desenvolvidas em paralelo, são elas pertencentes à Modelagem de Negócio e à Análise de Requisitos, uma vez que a segunda é dependente dos resultados da primeira. Para projetos em que a mudança dos requisitos é constante, as atividades destas duas disciplinas podem ser paralelas, porém, para o contexto do projeto QUASAR não houve ocorrência destas atividades em paralelo.

Durante toda a fase de Codificação, as atividades referentes à codificação são sempre paralelas às atividades de teste, justamente por se tratarem de atividades correlacionadas e dependentes. Atividades da disciplina de Análise e Design também podem ser paralelas ao desenvolvimento e teste pelo fato de englobarem a atualização dos diagramas e documentos desenvolvidos.

4.3 MODELAGEM DE NEGÓCIO

4.3.1 Necessidades do Negócio

Como necessidade principal da ferramenta QUASAR, tem-se a possibilidade de mapeamento de processos da organização com processos do modelo de referência selecionado e ainda, associar resultados esperados e atividades a questões de checklists de avaliação, possibilitando a rastreabilidade entre estas entidades.

A Figura 7 apresenta o mapeamento de um processo da organização com o modelo MR-MPS, como ocorre a rastreabilidade, sendo possível, a partir das não conformidades, se identificar os resultados e atividades/tarefas impactados pela sua ocorrência.

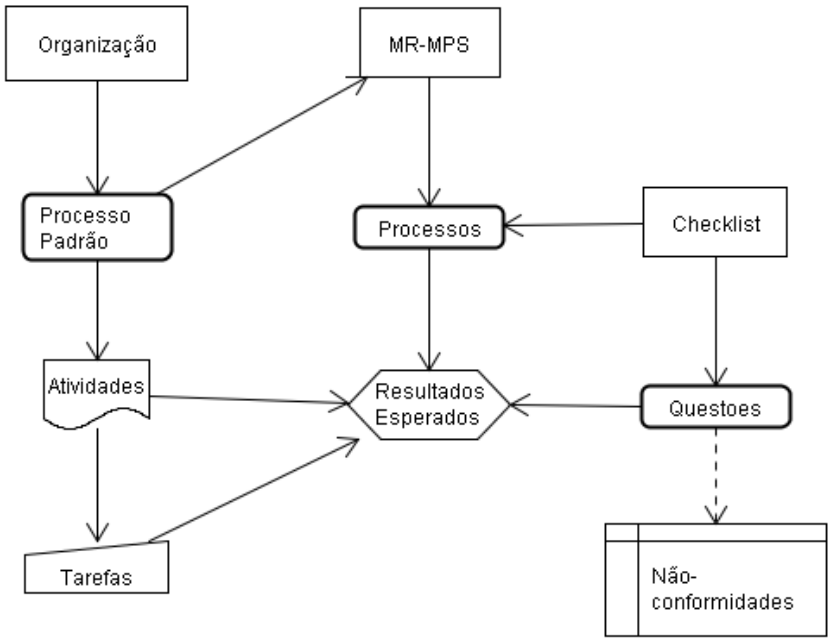

Figura 7 - Mapeamento de Atividades/Tarefas, Resultados Esperados e Não-conformidades

Além da rastreabilidade de não-conformidades e atividades e tarefas do processo, bem como processos e resultados esperados do modelo, outra proposta da ferramenta é atender a todos os resultados esperados do processo de Garantia da Qualidade e ainda implementar os itens onde a avaliação das ferramentas apresentadas foi insatisfatório ou onde há oportunidades de melhoria. Sendo assim, as principais necessidades são:

- Manter os clientes e seus processos;
- Manter Modelos de Referência;
- Criar e controlar Checklists;
- Planejar e executar Auditorias;
- Registrar e acompanhar não-conformidades; e
- Rastrear não conformidades com práticas dos Modelos de Referências.

Satisfeitas estas necessidades, o software QUASAR consegue atender aos resultados esperados do processo de Garantia da Qualidade, apresentados no Capítulo 2 deste documento, subseção 2.1.1 (O Modelo de Referência MR-MPS). O Quadro 9 apresenta os resultados esperados do processo de Garantia da Qualidade e as funcionalidades que atendem a cada um destes.

Quadro 9 – Mapeamento de Resultados Esperados do processo de GQA com necessidades do QUASAR

Resultado	Descrição	Necessidades
GQA 1	A aderência dos produtos de trabalho aos padrões, procedimentos e requisitos aplicáveis é avaliada objetivamente, antes dos produtos serem entregues e em marcos predefinidos ao longo do ciclo de vida do projeto.	• Manter processos; • Criar e controlar Checklists; • Planejar e executar Auditorias; • Manter Modelos de Referência.
GQA 2	A aderência dos processos executados às descrições de processo, padrões e procedimentos é avaliada objetivamente.	• Manter processos; • Criar e controlar Checklists;
GQA 3	Os problemas e as não conformidades são identificados, registrados e comunicados.	• Registrar e acompanhar não-conformidades;
GQA 4	Ações corretivas para as não conformidades são estabelecidas e acompanhadas até as suas efetivas conclusões. Quando necessário, o escalamento das ações corretivas para níveis superiores é realizado, de forma a garantir sua solução.	• Registrar e acompanhar não-conformidades;

4.3.2 Principais funcionalidades

A partir das necessidades apresentados na subseção 4.3.1 (Necessidades do Negócio), podem ser estabelecidas funcionalidades, que são apresentadas no

Quadro **10**, identificando-se também, os resultados esperados do processo de GQA (Quadro 2) atendidos por cada uma:

Quadro 10 - Funcionalidades Propostas QUASAR

Funcionalidades	Descrição
Permitir o Cadastro de Projetos	Todos os projetos da organização devem ser mantidos. Um determinado projeto pertence a um cliente e utiliza determinado processo.
Permitir Cadastro de clientes	Todos os clientes da organização devem ser mantidos com o intuito de identificar os processos e projetos de cada um.
Permitir o Cadastro de Processos	Deve ser possível manter os processos que serão avaliados e auditados.
Permitir o Registro de Critérios de Avaliação, (checklists)	Para auditar determinado processo, deve-se usar um checklist previamente definido.
Gerar Relatórios de Acompanhamento de GQA	Relatórios de acompanhamento podem ser visualizados, filtrando-se por projetos, auditorias, estado de não-conformidades, entre outros. O intuito é acompanhar a resolução das não-conformidades.
Permitir realização de Auditorias	A realização da auditoria é a ação de avaliar determinado processo de acordo com critérios previamente associados (checklists), dentro do prazo estipulado no cronograma, pelos auditores responsáveis designados pelo gerente de qualidade.
Permitir Registro de Não-Conformidades	Sempre que durante uma realização de auditoria for encontrada uma não-conformidade, esta pode ser registrada e acompanhada.
Cadastrar Modelo de Referência	Um processo pode ser associado a um modelo de referência (como, por exemplo, o MR-MPS) que contém as práticas que cada processo deve seguir.
Cadastrar Níveis de Maturidade	Dentro de um modelo de referência existem níveis de maturidade que apresentam os processos que devem ser implementados para se obter aquele nível.
Avaliar Processo (Nível de Maturidade)	A avaliação de processos está relacionada à atividade de avaliar os processos/resultados esperados do nível de maturidade solicitado e apresentar as não conformidades impactantes para uma avaliação daquele nível, apresentando as inconsistências encontradas e os processos do modelo de referência que não estão sendo cobertos pelo processo do cliente ou da organização.
Restringir acesso	Acesso restrito está relacionado à necessidade do sistema de possuir segurança de acesso em diferentes níveis de permissão (Gerente de Qualidade, Auditor e Gerente de Projetos).

4.3.3 Papéis e responsabilidades

O Software QUASAR é destinado a equipes de Garantia da Qualidade, sendo aberto também a equipes de Consultoria na área (por este motivo existe o módulo para cadastro de clientes e processos associados), sendo assim, os papéis encontrados neste são referentes a funções dentro da equipe. Existem três principais papéis, o Gerente de Qualidade, o Auditor e o Gerente de Projetos.

Começando pelo Gerente de Projetos, este tem permissão de acompanhar e resolver não conformidades atribuídas a ele. Este perfil será usado pelo gerente do projeto auditado, responsável por visualizar não conformidades e providenciar soluções. O Gerente de Projetos também pode visualizar relatórios referentes aos projetos em que está inserido.

O Auditor, por sua vez, tem a responsabilidade de efetuar as auditorias, a partir de *checklists* previamente cadastrados para cada processo que será. O Auditor também tem a responsabilidade de realizar a abertura de não-conformidades, sempre que um item é identificado como “em não-conformidade” ao se aplicar o *checklist* selecionado e adaptado (se necessário). As não conformidades abertas pelo Auditor podem ser acompanhadas até sua resolução.

O Gerente de Qualidade é aquele que detêm a maior responsabilidade, uma vez que o Gerente é o responsável pela manutenção dos dados base, são eles: *checklists*, modelos de referência, processos organizacionais, usuários e permissões, planejamento de auditorias, clientes e projetos. O Gerente de Qualidade também pode visualizar relatórios e realizar a “Avaliação de Processos” (funcionalidade pela qual as não conformidades existentes e identificadas para cada processo são rastreadas até que se identifique em que processos do modelo de referência são gerados impactos). Além destas atribuições, o Gerente de Qualidade também tem acesso às funcionalidades dos outros papéis.

4.3.4 Modelo de domínio

A partir das especificações do negócio observada nas subseções 4.3.1 e 4.3.2 foi possível realizar a abstração do modelo de domínio da aplicação, retratando-se a interação entre as estruturas que compõem o domínio da aplicação. A

Figura 8 apresenta o Modelo de Domínio do software QUASAR.

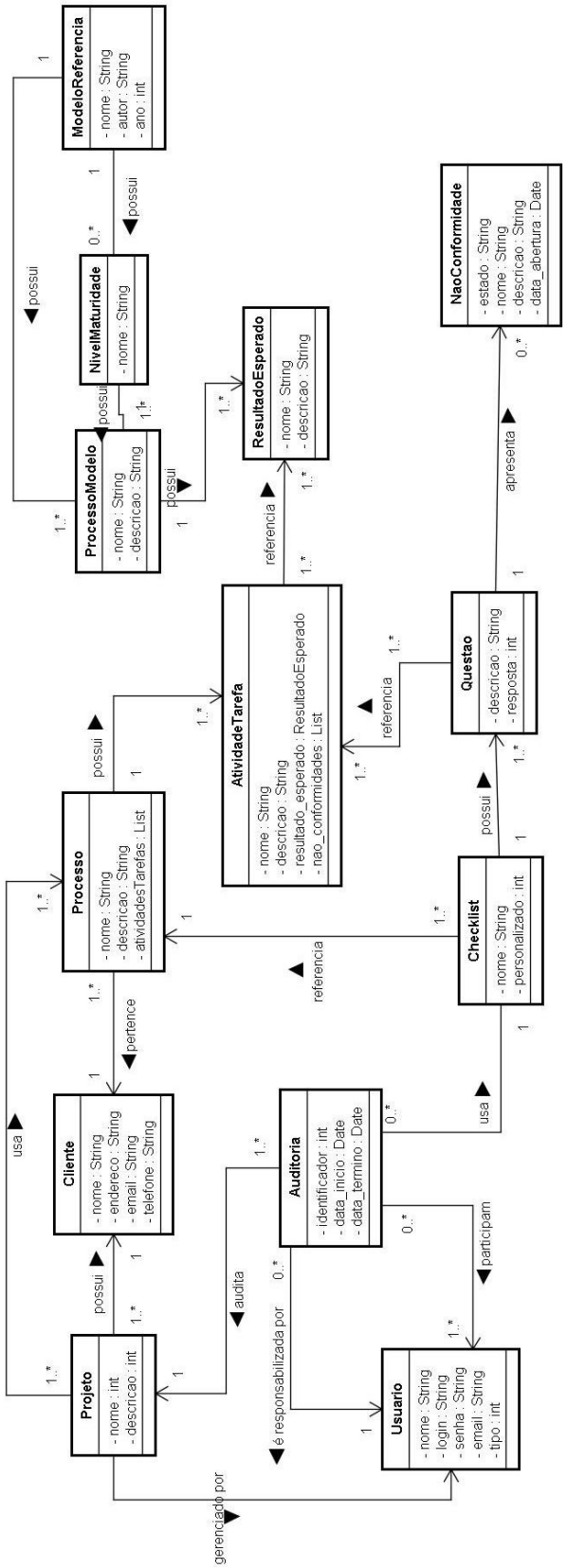

Figura 8 – Modelo de Domínio do Software QUASAR

Importante ressaltar que o Modelo de Domínio é utilizado para o detalhamento da arquitetura e também para o projeto da base de dados e do Modelo Entidade-Relacionamento, segundo o Processo Unificado, que também afirma que o Modelo de Domínio deve ser elaborado na fase de Elaboração do projeto (RATIONAL, 2001).

4.4 REQUISITOS

4.4.1 Requisitos Funcionais

A partir das funcionalidades apresentadas no

Quadro **10**, foi desenvolvido o diagrama de casos de uso, apresentado na Figura 9, todo o detalhamento dos casos de uso encontra-se no Anexo I deste trabalho.

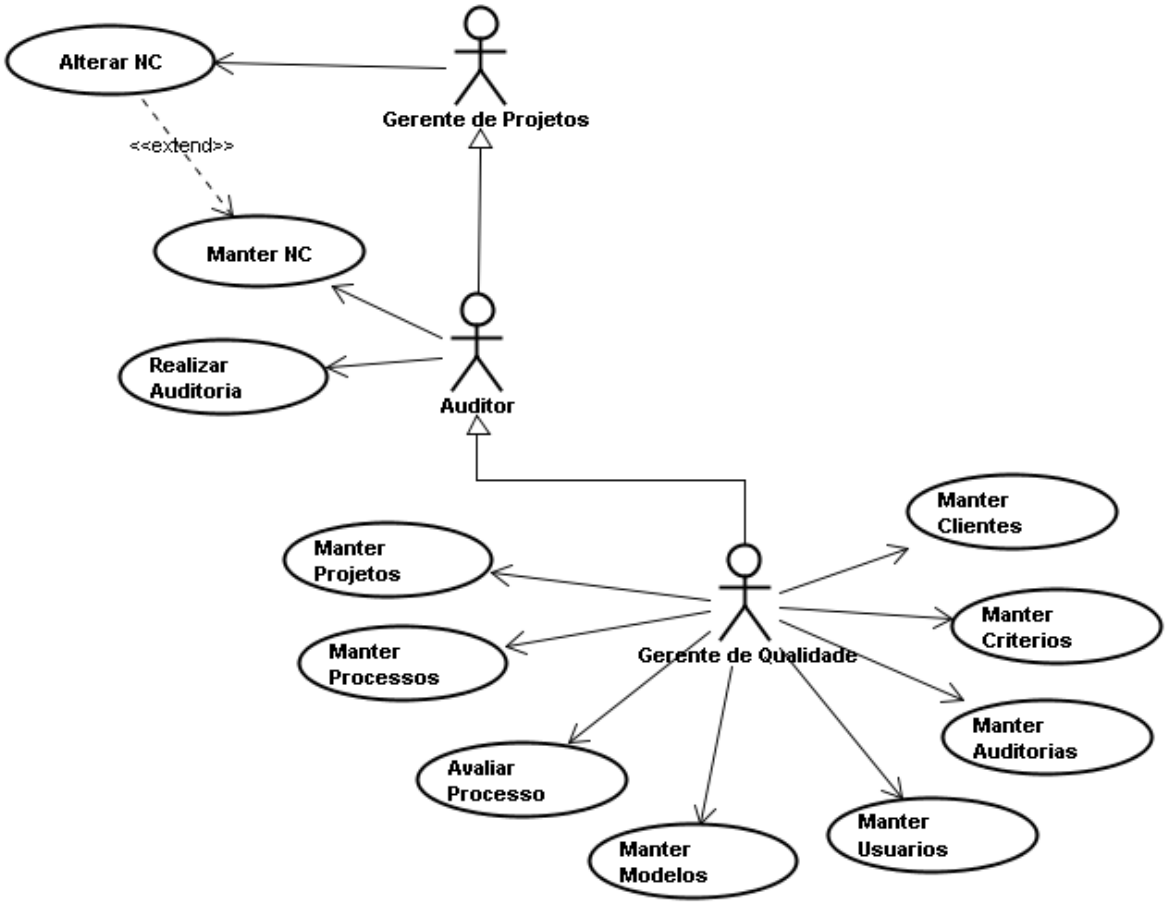

Figura 9 – Diagrama de Casos de Uso do Software QUASAR

4.4.2 Requisitos Não-funcionais

Com a finalidade de identificar “características implícitas” (PRESSMAN, 2006), foram levantados os Requisitos Não-funcionais para o projeto. Foram identificadas características relacionadas à Usabilidade, Desempenho, Suportabilidade e Requisitos de Design.

O Quadro 11 apresenta os requisitos não-funcionais do software QUASAR, baseados nos requisitos não funcionais encontrados na norma NBR ISO/IEC 25000 de 2008 (ABNT 2008), personalizados para o contexto desta aplicação.

Quadro 11 – Requisitos Não-funcionais do Software QUASAR

URPS+	Descrição
Usabilidade	**Interface:** O sistema deve apresentar linguagem usual do contexto de auditoria de qualidade.
Desempenho	**Capacidade de tratamento de requisitos:** o sistema deverá ser executado em ambiente corporativo, permitindo o acesso simultâneo de usuários, independente de nível de acesso.
Suportabilidade	**Compatibilidade:** o sistema será compatível com o IE 8, bem como Mozilla Firefox 20.0 e Google Chrome 26.0. **Expansibilidade:** a arquitetura do sistema deve ser expansível, sendo projetada de maneira a permitir manutenções evolutivas sem a quebra de arquitetura. Se possível, a partir da aplicação de frameworks e padrões de desenvolvimento.
Requisitos de Design	**Padrões de Design**: deve seguir padrões como botões salvar e cancelar com imagens usuais e de fácil identificação e padronização de posicionamento de botões, tabelas e listas.

4.5 ANÁLISE E PROJETO

Para o desenvolvimento do projeto QUASAR será usado o *Framework CodeIgniter* apresentado no Capítulo 2 deste trabalho, subseção 2.3.1 (CodeIgniter v2.1.3). Desta forma, algumas características de projeto foram projetadas de maneira a implementar o Framework.

O software foi projetado seguindo o padrão MVC, apresentado no Capítulo 2, subseção 2.5.1 (Padrão *Model-View-Controller*), suas camadas serão apresentadas na subseção 4.5.1, explicando a funcionalidade e importância de cada

uma, dentro do padrão. A seguir, na subseção 4.5.2, será apresentado o Diagrama de Sequencia do Sistema – DSS, que descreve os principais fluxos de tarefas necessários. Através deste, na subseção 4.5.3 serão apresentados os contratos de operações, para que, por fim, seja possível apresentar em nível maior de detalhamento, os Diagramas de Sequencia de Projeto (subseção 4.5.4). Importante ressaltar que estes artefatos foram criados de acordo com Larman (2007).

4.5.1 Arquitetura em Camadas – Padrão MVC

Devido ao uso do *Framework CodeIgniter*, foi necessário o desenho da arquitetura em camadas, mais especificamente seguindo o padrão MVC, apresentado no Capítulo 2 deste trabalho, subseção 2.5.1.

Para o desenvolvimento do software QUASAR, o padrão MVC é aplicado em conjunto com o uso da estrutura "*Library*", já existente no *Framework CodeIgniter*, que contém uma "biblioteca" de classes que pode ser usada em qualquer uma das camadas (Modelo, Visão ou Controle). Dentro da *Library* foram inseridas as classes que representam os objetos de domínio, por exemplo, as classes que representam Auditorias, Clientes, Projetos, Processos, Modelos, e todas as outras necessárias e apresentadas no Modelo de Domínio, apresentado na subseção 4.3.4 deste mesmo capítulo.

A inserção da estrutura, ou pacote, *Library* tem como objetivo favorecer a implementação do padrão "Indireção", através de seu relacionamento com os objetos presentes na camada de Modelo do padrão MVC, que atua como mediador entre as entidades do domínio da aplicação e a base de dados, apresentado na subseção 2.6.3 do Capítulo 2. A Figura 10 apresenta as camadas da Arquitetura e a Figura 11 um exemplo de implementação dentro do sistema destas camadas.

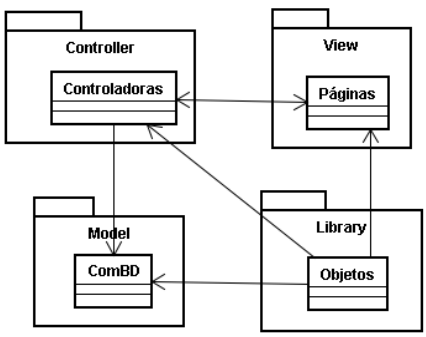

Figura 10 – Camadas da Arquitetura

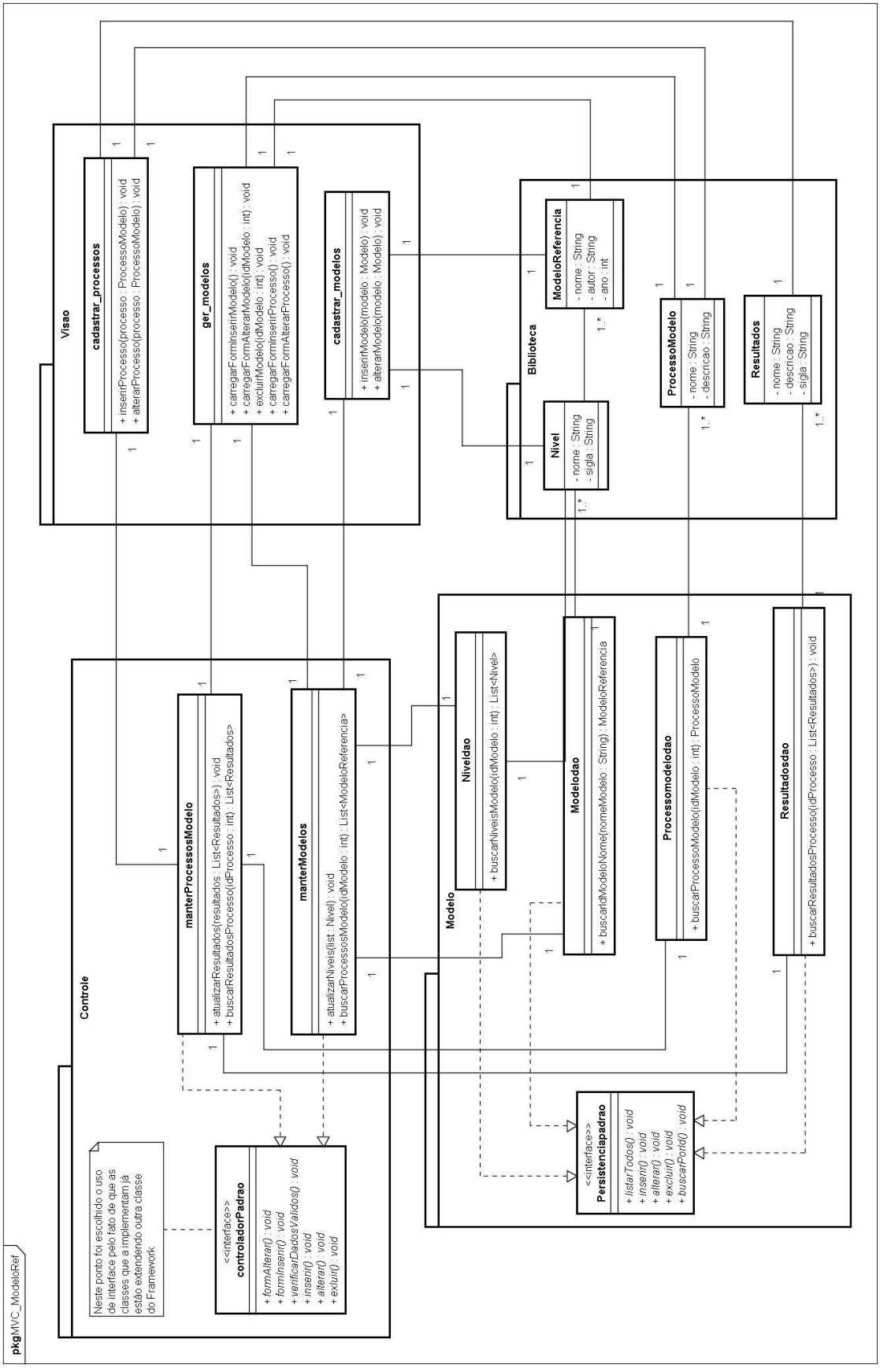

Figura 11 – Camadas do Caso de Uso Manter Modelos

Pode-se perceber que, para cada funcionalidade presente no sistema, ou seja, para cada Caso de Uso previsto (apresentado na subseção 4.4.1), deve haver entidades presentes nas três camadas do modelo MVC e ainda, nas bibliotecas do sistema, desempenhando função específica, de acordo com o local onde se inserem. Sendo assim, de acordo com a Figura 11, ao observar-se a camada de “Controle”, percebe-se a existência de uma classe chamada “ControladorPadrao”, que se trata de uma interface com os métodos a serem implementados por qualquer outra controladora (detalhes de implementação serão apresentados na seção 4.7 Codificação), retratando o uso do padrão “Polimorfismo” (Capítulo 2, subseção 2.3.6). As controladoras “manterProcessoModelo” e “manterModelos” são responsáveis por fluxos específicos de funções, de acordo com o caso de uso ao qual implementam (as controladoras apresentam os mesmos nomes dos casos de uso), percebe-se neste ponto a aplicação do padrão “Controlador” (Capítulo 2, subseção 2.3.5).

Dentro da camada de “Modelo”, estão presentes as classes de persistência necessárias, responsáveis por interagirem diretamente com a base de dados. Neste caso, percebe-se a existência de “Niveldao”, “Processomodelodao”, “Modelodao” e “Resultadodao”, onde a terminação “dao” refere-se à abreviação de “*Data Transfer Object*” ou seja, são objetos de transferência de dados, mais uma vez, referenciando a existência do padrão da indireção.

Bibliotecas de classes podem ser usadas, neste caso, as bibliotecas representam os próprios objetos do domínio da aplicação (como citado anteriormente). Estes objetos foram inseridos no formato de bibliotecas, pois assim se tornam acessíveis em qualquer uma das camadas do padrão MVC, dentro do *CodeIgniter*. A partir da Figura 11 pode-se perceber as classes “Nivel”, “ProcessoModelo”, “ModeloReferencia” e “Resultados” como bibliotecas passíveis de uso.

Por fim, na camada de “Visão” podem ser vistas as classes que entrarão em contato direto com o usuário, ou seja, os arquivos que contém as saídas de dados processados ou solicitados pelo usuário.

Pode-se notar que, dentro de todas as camadas é aplicado o estilo de arquitetura “Orientação a Objetos” (Capítulo 2, subseção 2.4.1), já que, entidades do mundo real são retratados dentro do sistema através de representação em objetos.

4.5.2 Diagrama de Sequencia do Sistema (DSS)

O Diagrama de Sequencia do Sistema, também conhecido como DSS, tem como objetivo identificar os eventos de um determinado sistema tendo como base os casos de uso e seu detalhamento, a troca de mensagens entre os atores e o software. O DSS apresenta as principais operações, ou funcionalidades críticas, que os atores podem realizar no QUASAR, representados através de diagrama de sequencias da UML.

Segundo Larman (2007):

> Todos os sistemas são tratados como uma caixa preta; a ênfase do diagrama está nos eventos que cruzam a fronteira do sistema de atores para o sistema.

Foram selecionados os casos de uso de “Manter Auditorias”, “Realizar Auditoria” e “Avaliar Processos”, onde estes foram separados em diferentes DSSs. Os DSSs de cada um dos casos de uso são apresentados respectivamente pela Figura 12, Figura 13 e Figura 14.

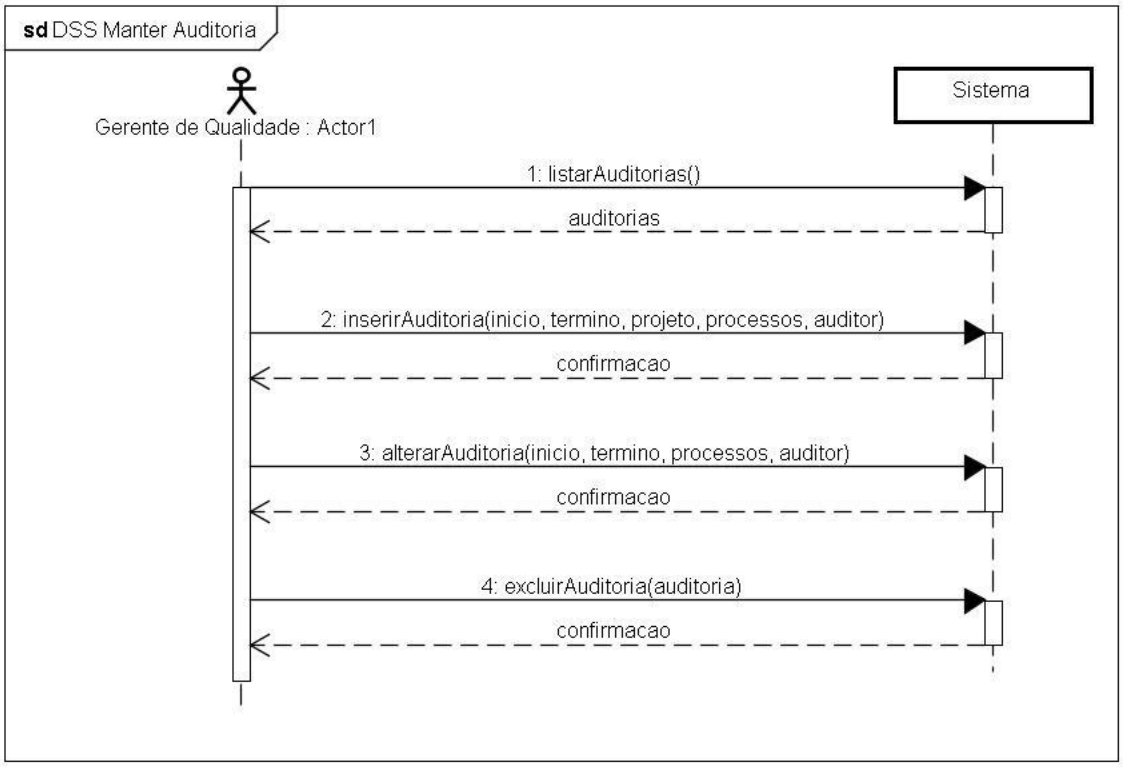

Figura 12 – DSS Manter Auditoria

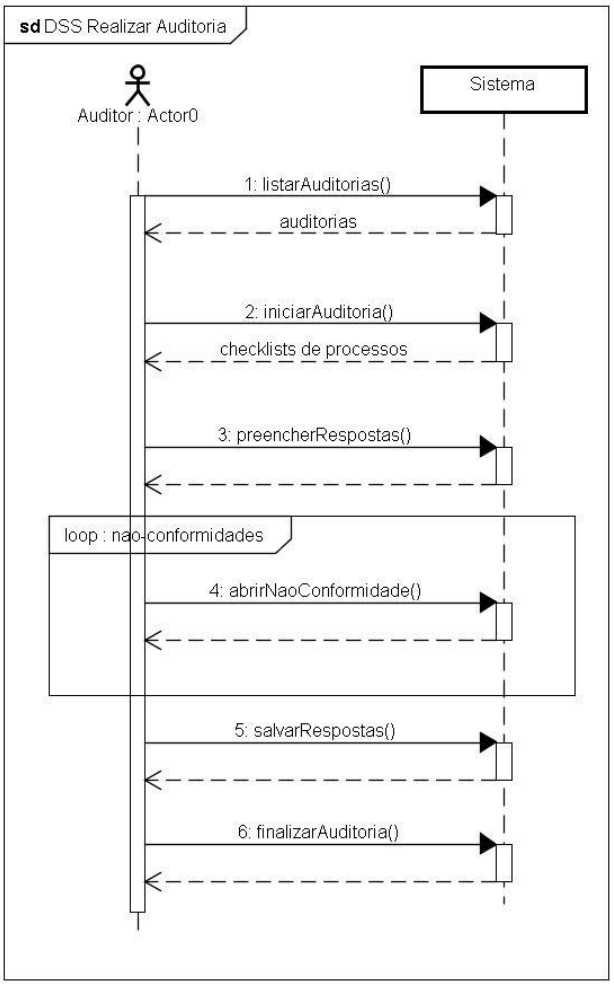

Figura 13 – DSS Realizar Auditoria

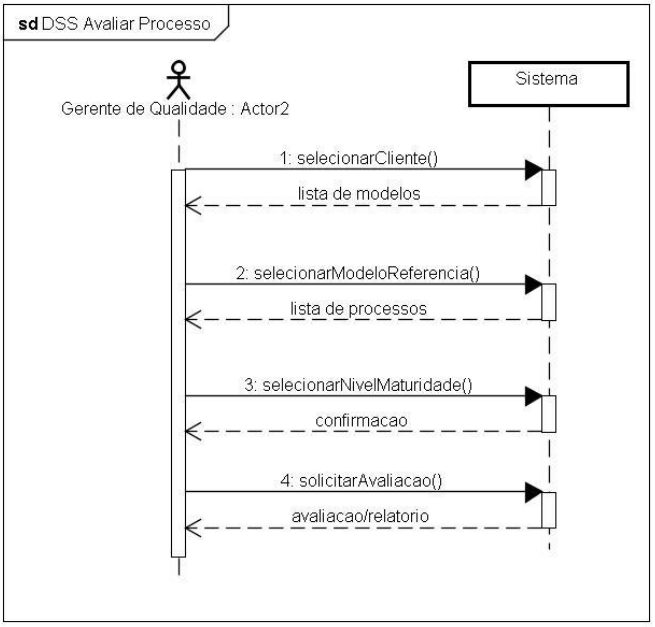

Figura 14 – DSS Avaliar Processo

4.5.3 Contratos de Operação

Cada operação encontrada nos DSS citados na subseção 4.5.3, deste capítulo pode realizar alteração nos objetos de domínio, ou seja, as auditorias e seus atributos, bem como as respostas dos *checklists* e não-conformidades, terão seus estados e dados alterados ao longo da execução do sistema. Sendo assim, foi necessário o desenvolvimento dos Contratos de Operações. Um contrato de operação descreve as mudanças dos objetos ao longo das operações realizadas, apresentando pré e pós-condições (Larman, 2007).

Como exemplos serão apresentados os contratos de operações para o caso de uso “Realizar Auditoria”, apresentados no Quadro 12, Quadro 13 e Quadro 14.

Quadro 12 – Contrato da Operação “iniciarAuditoria”

Nome da Operação	**iniciarAuditoria**
Pré-condições	Entidade do tipo Auditoria existente.
Pós-condições	Lista de entidade do tipo Criterio criada; Lista de entidade do tipo Questao criada e associada a Criterio.
Saída	Checklist de avaliação montado.

Quadro 13 – Contrato da Operação “preencherRespostas”

Nome da Operação	**preencherRespostas**
Pré-condições	Entidades do tipo Auditoria, Criterio e Questao existentes.
Pós-condições	Lista de entidades do tipo Resposta criadas e associadas à Questao.
Saída	Checklist de avaliação salvo.

Quadro 14 – Contrato da Operação “abrirNaoConformidade”

Nome da Operação	**abrirNaoConformidade**
Pré-condições	Entidades do tipo Auditoria, Criterio, Resposta e Questao existentes.
Pós-condições	Lista de entidades do tipo NaoConformidade criada e associada à Questao.
Saída	Checklist de avaliação montado

4.5.4 Diagrama de Sequencia de Projeto

A partir do Diagrama de Sequencia do Sistema apresentado na subseção 4.5.2 deste capítulo, foi possível chegar a um maior nível de detalhamento, gerando os Diagramas de Sequencia de Projeto. As Figuras Figura **15** eFigura **16** apresentam, respectivamente, os diagramas de sequencia que representam os casos de uso “Avaliar Processo” e “Realizar Auditoria”, apresentando as entidades envolvidas e fluxo de funções, observando-se estes, como forma de detalhamento dos “DSSs” apresentados na subseção 4.5.3.

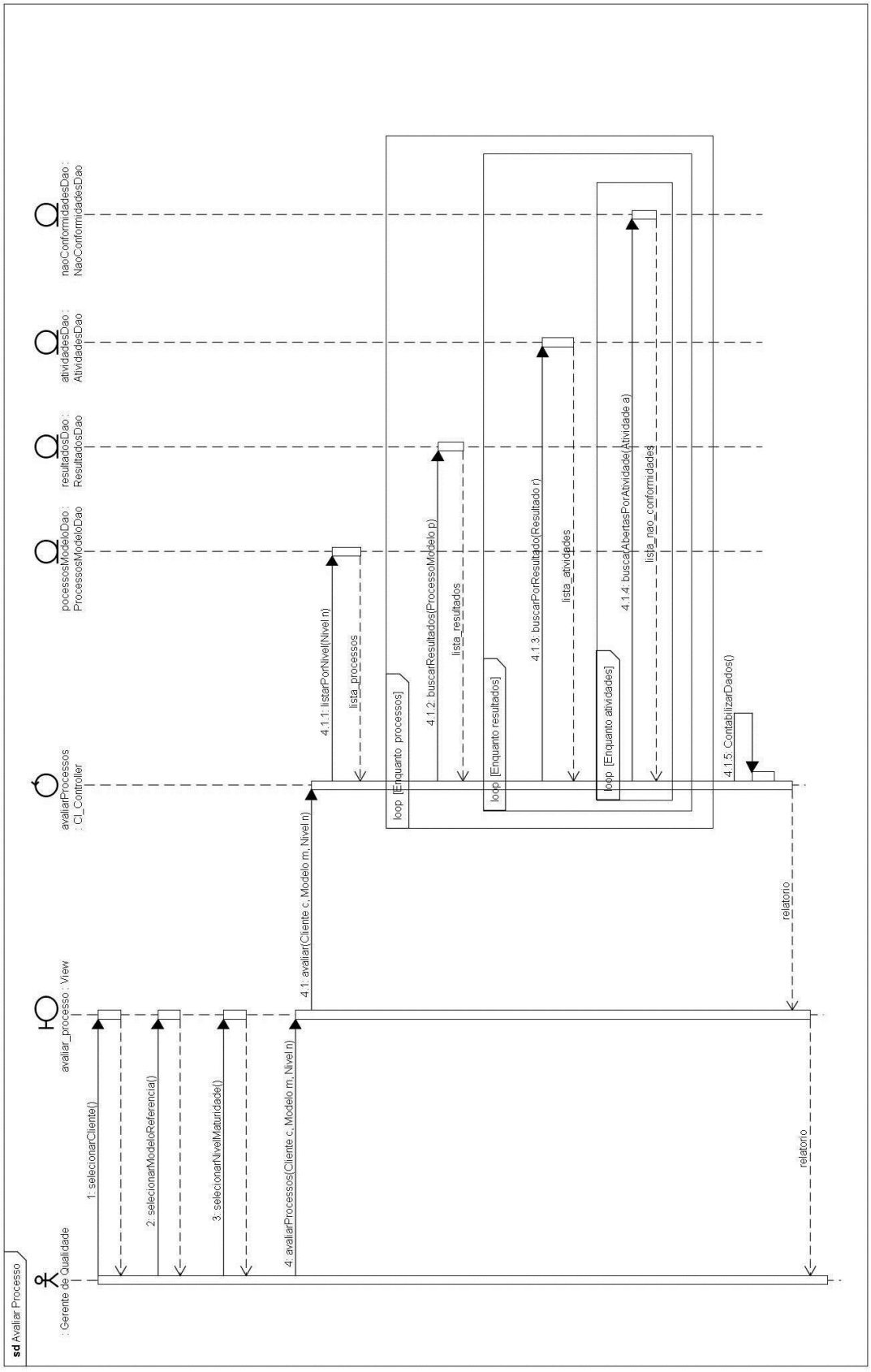

Figura 15 – Diagrama de Sequência de “Avaliar Processo”

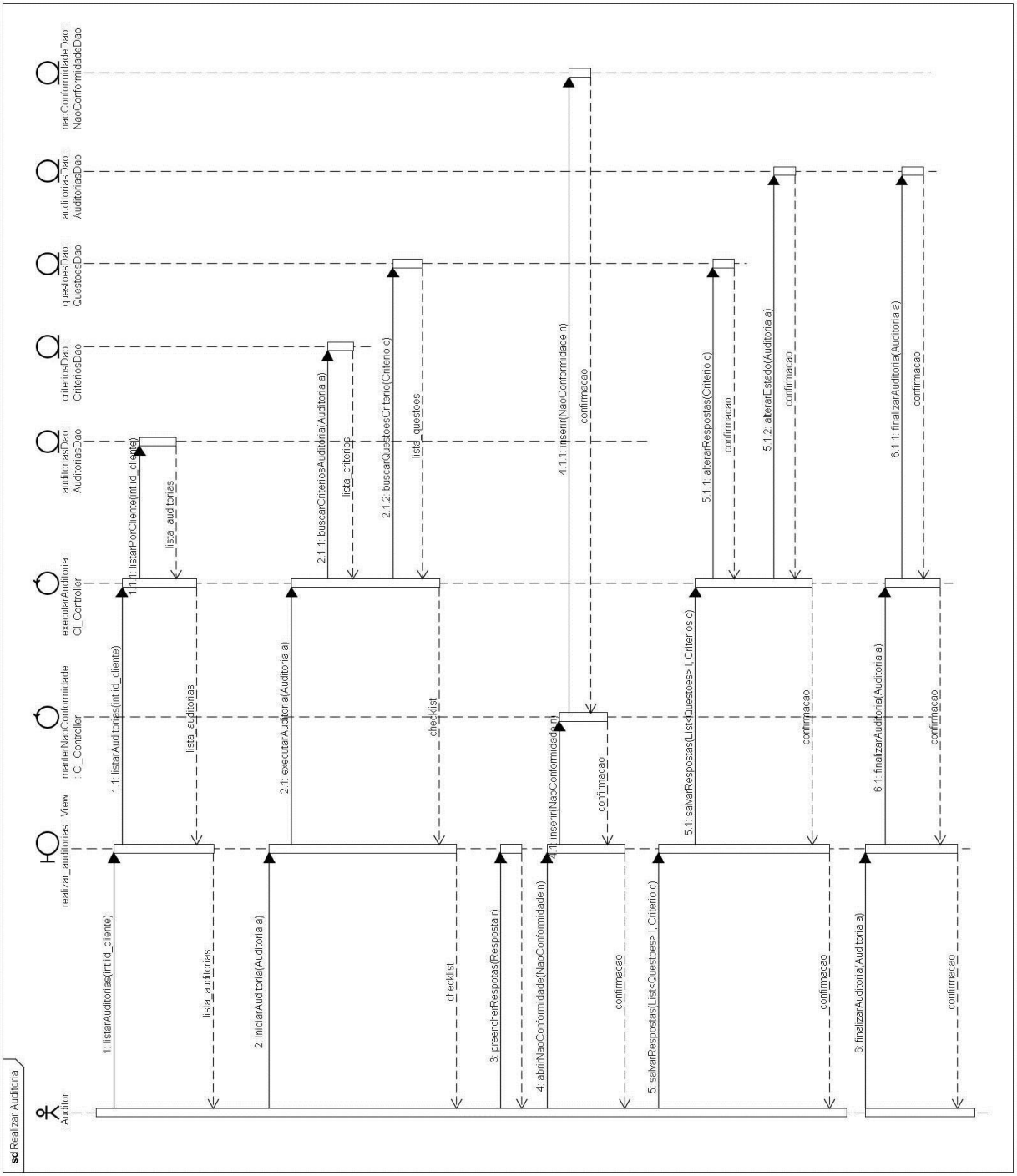

Figura 16 – Diagrama de Sequência de “Realizar Auditoria”

A partir das Figuras Figura **15** e Figura **16**, pode-se perceber que sempre que um ator entra em contato com o sistema, esta interação ocorre através da *Boundary*, também conhecida como interface com o usuário ou *View*. No caso da implementação usada, esta equivale às páginas de acesso do sistema. Através destas interfaces com o usuário, são acionadas funcionalidades disponíveis no

Control, ou na Controladora, entidade responsável por controlar os fluxos de dados e realizar chamadas às funções, agindo como controladora dos casos de uso. A Controladora é a responsável por acionar pesquisas ou alterações nas bases de dados, através da instanciação das entidades “DAO”, responsáveis pela Persistência.

Pode-se observar claramente o fluxo de controle apresentado através do modelo de camadas apresentado na subseção 4.5.1 (Arquitetura em Camadas – Padrão MVC), Figura 11, deste mesmo capítulo.

4.5.5 Modelagem de Dados

A partir da análise do domínio da aplicação, foi realizada a modelagem dos dados, necessária para a criação da base de dados do sistema. Ao desenvolver a modelagem dos dados, é importante ressaltar que todo o esquema relacional foi normalizado, ou seja, foi aplicada uma série de passos para garantir a eficiência e consistência dos dados (Elmasri *et al.*, 2010). Segundo Elmasri *et al.* (2010), a normalização se faz importante para que se possa validar um esquema relacional e minimizar as redundâncias de dados.

A Figura 17 apresenta a modelagem do banco de dados para o Software QUASAR, onde se pode perceber o relacionamento entre as 18 (dezoito) tabelas que o compõem.

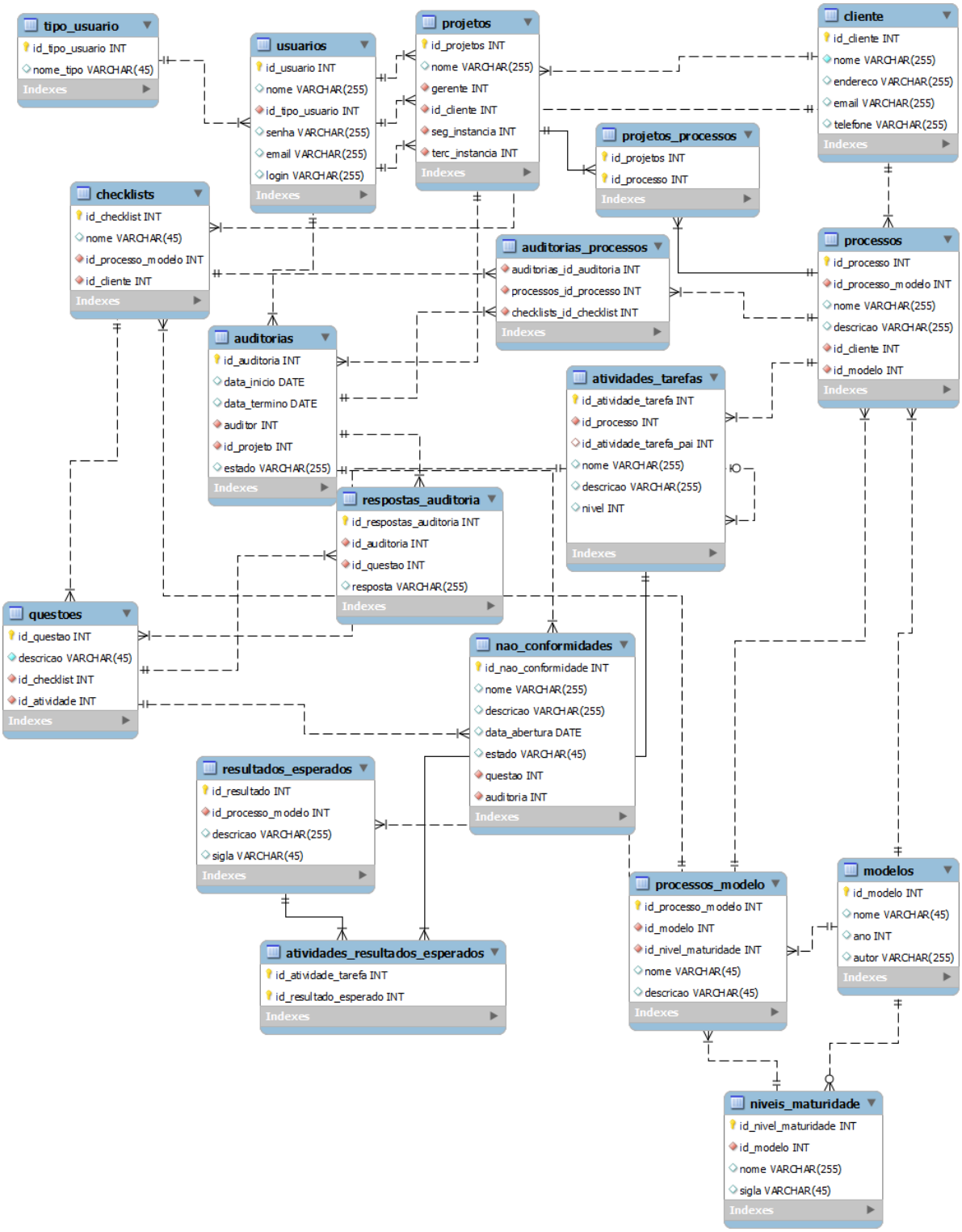

Figura 17 – Modelagem de Dados do Sistema QUASAR

4.6 AMBIENTE

Para alcançar um desenvolvimento eficiente, serão usadas ferramentas de apoio ao projeto. Estas ferramentas são apresentadas no Quadro 15, discriminando-se as disciplinas (RATIONAL) onde serão usadas as mesmas e o propósito em seu uso:

Quadro 15 – Ferramentas usadas durante o desenvolvimento

Ferramenta	Disciplina Associada	Propósito
Netbeans 7.0.1	Codificação	Auxilio ao desenvolvimento e codificação. Dá suporte à linguagem de programação PHP.
GanttProject 2.0.10	Gerenciamento de Projeto	Gerenciamento das atividades e tarefas do projeto, geração de cronograma e acompanhamento de gráfico de Gantt.
Tortoise SVN 1.7.2	Gerenciamento de Configuração e Mudança	Gerenciamento, manutenção e alimentação de repositório SVN, geração de builds e releases.
Astah Professional 6.1	Requisitos, Modelagem de Negócio e Análise e Design	Desenvolvimento e criação de diagramas de Classes, Casos de Uso e Sequência.
MySQL Workbench 5.2.47	Análise e Design	Desenvolvimento de modelo Entidade Relacionamento para projeto de Banco de Dados.
XAMPP 1.7.3	Codificação, Teste, Implantação	Administração dos servidores de aplicação (Apache 2) e banco de dados (MySQL) para execução do software QUASAR.

4.7 CODIFICAÇÃO

Toda e qualquer linha de código inserida no projeto deve estar diretamente ligada e condizente às especificações de design e arquitetura desenvolvidas previamente. O fato de haver um desenvolvedor único e este ser o mesmo analista responsável pelo desenho da arquitetura e especificações do software facilitou de maneira ímpar a codificação, pois não havia esforço com gerenciamento e transmissão de conhecimento. Este fato também facilitou com relação às implementações errôneas muitas vezes ligadas à falta de compreensão de diagramas disponibilizados.

Todo o processo de codificação foi realizado de maneira a facilitar a leitura e compreensão de terceiros, sendo assim, todas as funções desenvolvidas e trechos

críticos foram comentados da mesma maneira que variáveis críticas foram apresentadas e comentadas (entende-se por variáveis críticas aquelas usadas para o correto fluxo de informação entre as páginas do sistema, variáveis de controle de fluxo de informação).

4.7.1 Programação com Interfaces

De acordo com o *framework* CodeIgniter todas as classes controladoras do sistema devem herdar as características do "*CI_Controller*", classe que define uma controladora e suas características dentro do *framework*.

Além de herdar as características de um controlador do framework, para o contexto do projeto QUASAR, foi necessária a implementação da interface "ControladorPadrao", que define os métodos básicos que todo o controlador deve conter em seu escopo.Tal implementação colabora fortemente para a redução do acoplamento.

Um exemplo pode ser visto na Figura 18, onde há a apresentação da criação de duas classes controladoras: (i)"ManterModelos" e (ii)"ManterProcessosModelos", que implementam a interface "ControladoraPadrao" e herdam as características de "CI_Controller". Importante ressaltar que cada trecho destacado (apesar de estar apresentado na mesma figura) está presente em um arquivo diferente, nomeado de acordo com os padrões do Framework CodeIgniter.

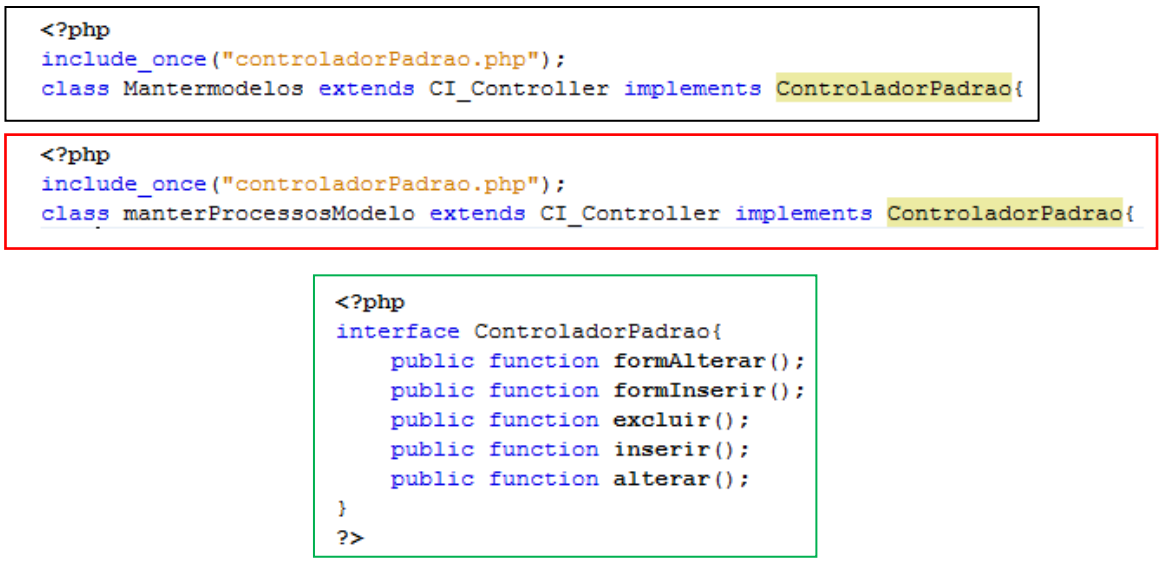

Figura 18 - Codificação do relacionamento entre as Controladoras, a Interface e a classe do Framework

Ao observar a Figura 18, percebe-se que ambas controladoras herdam através da assinatura “extends” as características de “CI_Controller”, para indicar que estas fazem parte das controladoras do sistema. Além disso, ambas implementam (através da assinatura “implements”) o “ControladorPadrao”, que é apresentado como uma interface que obriga a criação das funções “formAlterar()”, “formInserir()”, “excluir()”, “inserir()” e “alterar()”.

O fato de ser usada a interface e não uma herança é que cada controladora implementa de maneira diferente estes métodos, com fluxos de dados diferentes. Também foi usada a interface pelo fato de que já está sendo usada uma herança obrigatória para o Framework, conforme apresentado no parágrafo anterior. Esta mesma estratégia foi usada para as classes de persistência, que implementam a interface “PersistenciaPadrao”, exclusiva do projeto QUASAR e herdam as características da classe “CI_Model”, seguindo a regra do CodeIgniter (o projeto pode ser visto na Figura 11).

4.7.2 Programação Recursiva

Durante diversos momentos foram encontradas dificuldades referentes à maneira pela qual determinada funcionalidade deveria ser implementada para obtenção de sucesso. Um destes momentos foi o desenvolvimento e manipulação da árvore de processos. A dificuldade maior em se trabalhar com árvore de processos, para o contexto desta aplicação, é referente à decisão de não limitar a quantidade de tarefas e sub-tarefas para cada atividade do processo do usuário. Sendo assim, o usuário pode cadastrar atividades e tarefas até o menor nível desejado. Esta característica do sistema aumenta a dificuldade em se manipular os dados, seja para exibição, alteração, inserção ou exclusão, já que um item pode ter zero, um ou até infinitos níveis abaixo.

A solução para este problema foi a aplicação de rotinas recursivas no código, que segundo Alves (2006), trata-se de métodos ou funções que realizam chamadas a si mesmas de maneira direta ou indireta. A função apresentada pela Figura 19 a seguir apresenta um exemplo de uso de recursividade para manipulação dos processos dentro do projeto QUASAR.

```php
/*O objetivo desta função é varrer a árvore de processo excluindo as
 * atividades filhas no momento da exclusão de uma atividade pai.
 * Neste trecho é usada a recursividade pois há a chamada à mesma função
 * de exclusão para cada filha encontrada.
 */
public function excluirAtividade($id){
    $atividades = $this->buscarAtividadesFilhas($id);

    if($atividades != ""){
        foreach($atividades as $atividade){
            $this->excluirAtividade($atividade->id_atividade_tarefa);
        }
    }
    $this->db->where('id_atividade_tarefa', $id);
    $this->db->delete('atividades_tarefas');
}
```

Figura 19 – Uso de recursividade na função “excluirAtividade”

Como observado no comentário da função, esta varre a árvore de processos excluindo todas as atividades filhas e também as “filhas das filhas” num processo recursivo, onde a função realiza uma chamada a ela mesma durante o laço de repetição “*foreach*”, característico da linguagem PHP. Importante ressaltar que toda e qualquer função recursiva deve ter uma condição de parada, caso contrário haveria chamadas infinitas a ela mesma, o que provocaria um *crash* de memória.

Desta forma, a função primeiramente busca todas as atividades filhas da atividade passada como parâmetro (a passagem é feita através do identificador da atividade). Caso hajam atividades filhas (armazenadas na variável $atividade), para cada uma destas é chamada a função “excluirAtividades”. O procedimento final desta função é excluir a própria atividade que deu origem à sua chamada. Sendo assim, resumidamente, procura-se as atividades filhas e estas são excluídas, logo após, a atividade pai é apagada também.

Desta maneira, não é necessário saber quantos níveis possui o processo do cliente, ou seja, quantas atividades filhas existem, a própria função busca de maneira recursiva e realiza as operações necessárias. O procedimento é semelhante quando se apresenta a estrutura do processo, numa busca recursiva por atividades filhas.

4.7.3 Consultas Complexas na Base de Dados

O próprio objetivo principal da ferramenta QUASAR, o cruzamento de informações a respeito de modelos de referência, processos da organização e resultados de auditoria demonstram a necessidade de consultas em diversas tabelas da base de dados para geração de relatórios. Conforme observado na modelagem de dados apresentada na subseção 4.5.5 (Modelagem de Dados), existem no total 18 (dezoito) tabelas, dentre as quais, para realização do caso de uso “Avaliar Processo” (

Quadro **10**, linha 10) estão envolvidas as 9 (nove) seguintes: (i) “cliente”; (ii) ‘processos”; (iii) “atividades_tarefas”; (iv) “processos_modelo”; (v) modelos; (vi) “níveis_maturidade”; (vii) “nao_coformidades”; (viii) “resultados_esperados”; (ix) “atividades_resultados_esperados”.

Como esta se trata de uma consulta complexa na base de dados, foi necessário o uso de recursos da própria linguagem SQL para concretização da pesquisa. Estes recursos usados são os “JOINs”, através dos quais, uma pesquisa pode ser realizada ao mesmo tempo em diferentes tabelas, desde que, estas tabelas contenham atributos coincidentes. Um exemplo da aplicação de um “JOIN” da linguagem SQL é apresentado na Figura 20.

```
public function buscarPorProjeto($id_proj){
    $this->db->where('id_projeto', $id_proj);
    $this->db->select('n.*');
    $this->db->from('nao_conformidades n');
    $this->db->join('auditorias a', 'a.id_auditoria = n.auditoria', 'left');

    $query = $this->db->get();

    if($query->num_rows()>0){
        return $query->result();
    }

    return "";
}
```

Figura 20 – Uso de “JOIN” para pesquisa em diferentes tabelas da base de dados

Observando-se a Figura 20, percebe-se que se trata do método “buscarPorProjeto” que recebe um projeto como parâmetro e, através deste, realiza uma busca na base de dados, com a finalidade de obter as não-conformidades referentes a este.

De acordo com a modelagem de dados apresentada na subseção 4.5.5 (Modelagem de Dados), uma não-conformidade está associada a uma auditoria, que por sua vez está associada a um projeto, sendo assim, para que possa ser possível listar as não-conformidades encontradas e referentes a um projeto, necessita-se primeiramente encontrar as auditorias realizadas para este e as não-conformidades associadas a esta auditoria.

Ainda segundo a Figura 20, percebe-se que é usada a sintaxe padrão do framework CodeIgniter, que, através do uso dos métodos "*where", "select", "from*" e "*join*", contidos no objeto "*db*", resultam na seguinte consulta SQL:

SELECT n.* FROM nao_conformidades n LEFT JOIN auditorias a ON a.id_auditoria = n.auditoria WHERE id_projeto = $id_projeto

No comando SQL apresentado, "$id_projeto" refere-se ao valor da variável recebida como parâmetro da função. A partir deste, são listadas todas as não-conformidades presentes nas auditorias as quais o referido projeto esteve participando.

4.7.4 Diretórios de Código

O Framework CodeIgniter apresenta originalmente uma estrutura de diretórios que facilita a compreensão, organização e desenvolvimento das aplicações que fazem seu uso. Sendo assim, já estão destinados os diretórios corretos para cada tipo de arquivo criado durante o desenvolvimento.

A estrutura de diretórios apresentada e que manteve-se para o projeto QUASAR está representada na Figura 21.

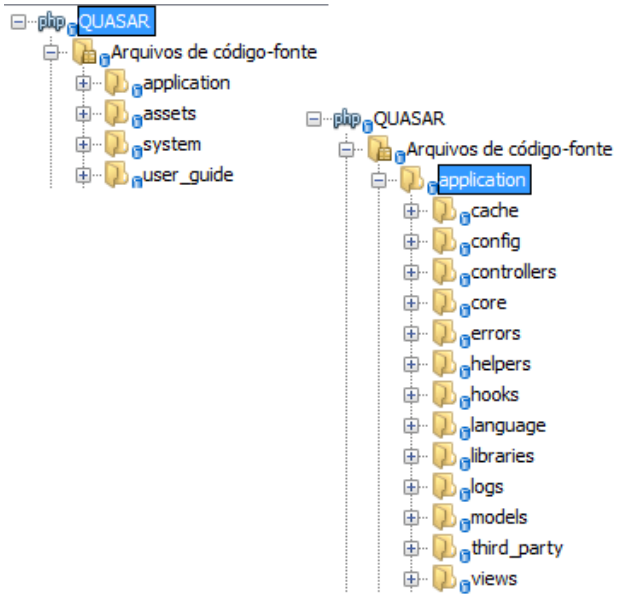

Figura 21 – Estrutura de Diretórios do Projeto QUASAR

A partir da Figura 21 é possível perceber uma estrutura rígida de diretórios, fornecida pela aplicação do próprio *CodeIgniter*. Os principais diretórios são: (i) "*application*", onde estarão as classes que compõem o sistema; (ii) "*assets*", onde se encontram itens que podem ser adicionados às classes, como scripts, imagens ou folhas de estilo; (iii) system, onde a lógica do framework está implementada; e (iv) "*user_guide*", onde é encontrado o manual do usuário. Para o contexto deste projeto, são alterados os diretórios "*application*" e "*assets*".

Navegando-se através do diretório "*application*", ainda conforme apresentado na Figura 21, à sua direita, encontram-se os diretórios onde são organizados os arquivos para compor o software em desenvolvimento. Dentro destes diretórios evidencia-se a aplicação do Padrão MVC, uma vez que, são encontrados os "*controllers*", "*models*" e "*views*", bem como o diretório de "*libraries*", conforme apresentado na subseção 4.5.1 (Arquitetura em Camadas – Padrão MVC) deste mesmo capítulo. Os diretórios restantes fazem parte da lógica do framework, contendo diretórios específicos para configurações, armazenamento de cache, entre outros.

4.8 VERIFICAÇÃO E TESTES

Para o contexto deste trabalho, foram desenvolvidos testes unitários do software QUASAR, sendo estes desenvolvidos paralelamente ao desenvolvimento. O objetivo dos testes unitários durante o desenvolvimento é validar as entradas e

saídas de funções inerentes aos objetos de domínio do software QUASAR. Sendo assim, foram desenvolvidos testes unitários para a grande maioria dos métodos presentes nos objetos modelados a partir da

Figura 8, subseção 4.3.4 (Modelo de Domínio).

Devido ao desenvolvimento da ferramenta ter sido realizado fazendo-se uso da linguagem PHP, foi necessária a instalação e uso do *FrameWork PHPUnit*, sendo usada a versão 3.4.9.

Quanto à organização dos testes, os mesmos foram desenvolvidos em arquivos separados para cada objeto do domínio, porém, a execução dos testes se deu a partir de uma Suíte de Testes que une todos os testes unitários dos objetos.

A Figura 22 apresenta um exemplo de teste unitário de um objeto de domínio, enquanto a Figura 23 apresenta a execução de um teste unitário dentro Suíte de Testes que executa todos os testes unitários feitos para o sistema.

```
class TesteProcesso extends PHPUnit_Framework_TestCase
{
    public function testCriarProcesso(){
        $processo = new Processo();

        $this->assertNotNull($processo);
    }
    public function testIdProcesso($id){
        $processo = new Processo();
        $processo->setId_processo($id);

        $this->assertNotNull($processo->getId_processo());
        $this->assertNotEquals($processo->getId_processo(), '');
    }
    public function testNomeProcesso($nome){
        $processo = new Processo();
        $processo->setNome($nome);

        $this->assertNotNull($processo->getNome());
        $this->assertNotEquals($processo->getNome(), '');
    }
    public function testDescricaoProcesso($descricao){
        $processo = new Processo();
        $processo->setDescricao($descricao);

        $this->assertNotNull($processo->getDescricao());
        $this->assertNotEquals($processo->getDescricao(), '');
        $this->assertEquals($processo->getDescricao(), $descricao);
    }
}
```

Figura 22 – Testes unitários para o objeto “Processo”

```
echo "<center><h2>Testes Unitários Software QUASAR</h2></center>";
echo "<table align='center' border>
      <tr><td align='center'><b>Teste</b></td><td align='center'><b>Resultado</b></td></tr>";
echo"<tr><td>Teste do Objeto Processo</td><td>";

/*TESTE DO OBJETO PROCESSOS*/
try{
    $processoTest->testCriarProcesso();
    $processoTest->testIdProcesso($id);
    $processoTest->testNomeProcesso($nome);
    $processoTest->testDescricaoProcesso($descricao);

    echo "SUCESSO!";

}catch(Exception $e){

    echo "FALHA!<br>";
    echo "Exceção: ".$e;

}
echo "</td></tr>";
```

Figura 23 – Execução do teste unitário de Processos dentro da Suíte de Testes

Outro aspecto que merece atenção foi o fato de que em diversos momentos do desenvolvimento foi feita revisão por pares, onde profissionais e estudantes alheios ao projeto foram convidados para auxiliar na atividade de verificação do desenvolvimento dos códigos e da execução so software em si, buscando identificar falhas de requisitos e codificação no software QUASAR.

4.9 RESULTADO FINAL

Após executadas as disciplinas previstas pelo Processo Unificado e apresentadas nas seções anteriores, obteve-se como resultado um software em condições de uso, pronto para ser implantado em organizações que almejem o fornecimento de consultorias em Garantia da Qualidade ou até mesmo empresas que possuem o processo instanciado e desejam usar a ferramenta para apoio.

Sendo assim, nesta seção será apresentado o software QUASAR, seu fluxo de execução, instruções de uso e o relatório final de avaliação de processos conforme níveis de maturidade de modelos de referência (objetivo principal deste projeto). A Figura 24 apresenta o fluxo de tarefas para execução do software QUASAR.

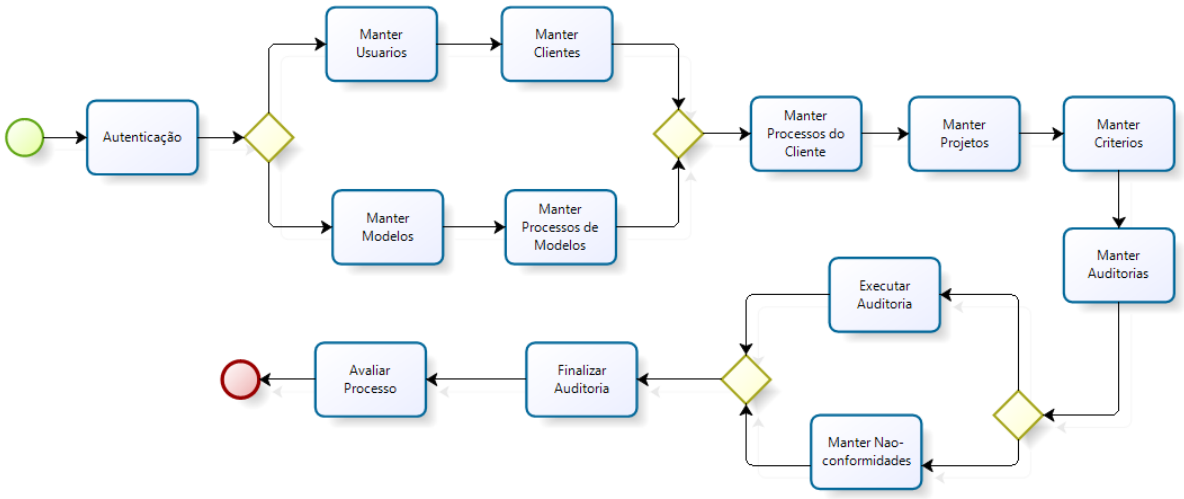

Figura 24 – Fluxo de atividades do Gerente de Qualidade no Software QUASAR

A partir da Figura 24 apresentada acima, percebe-se que todo o fluxo de tarefas do software inicia-se com a autenticação no sistema. Por motivos de segurança, a senha de todos os usuários é tratada através do uso de uma função de *hashing* denominada "*md5*", nativa para a maioria das linguagens. Este fator incrementa na segurança do sistema, já que dificulta que usuários mal-intencionados realizem a quebra da senha de outros usuários. A tela de autenticação do sistema é exibida através da Figura 25.

Figura 25 – Tela de Autenticação do QUASAR

Após autenticar, faz-se necessário que o Gerente de Qualidade cadastre usuários (Manter Usuários) e clientes (Manter Clientes). A primeira funcionalidade permite que sejam dadas permissões de uso dentro do sistema, observando-se as funcionalidades acessíveis para cada um dos três tipos de usuário apresentados neste trabalho, para a segunda funcionalidade, são cadastrados os clientes que serão atendidos, aqueles que contrataram a consultoria, ou a própria empresa (para o caso de uso interno do software QUASAR).

Paralelo às atividades de cadastro de usuários e clientes, podem ser mantidos os modelos de referência (Manter Modelos) e os processos (Manter Processos de Modelos) que compõem estes modelos. Na primeira, são criados novos modelos, indicando seu autor, descrição e ano de criação, bem como seus níveis de maturidade (cadastrando a sigla e descrição para cada nível). Na segunda, são criados os processos que compõem cada modelo de referência, cadastrando os resultados esperados que compõem cada processo, indica-se também, o nível de maturidade do modelo em que cada processo se encaixa. A Figura 26 apresenta a tela de administração de Modelos e a Figura 27 o cadastramento de um novo modelo.

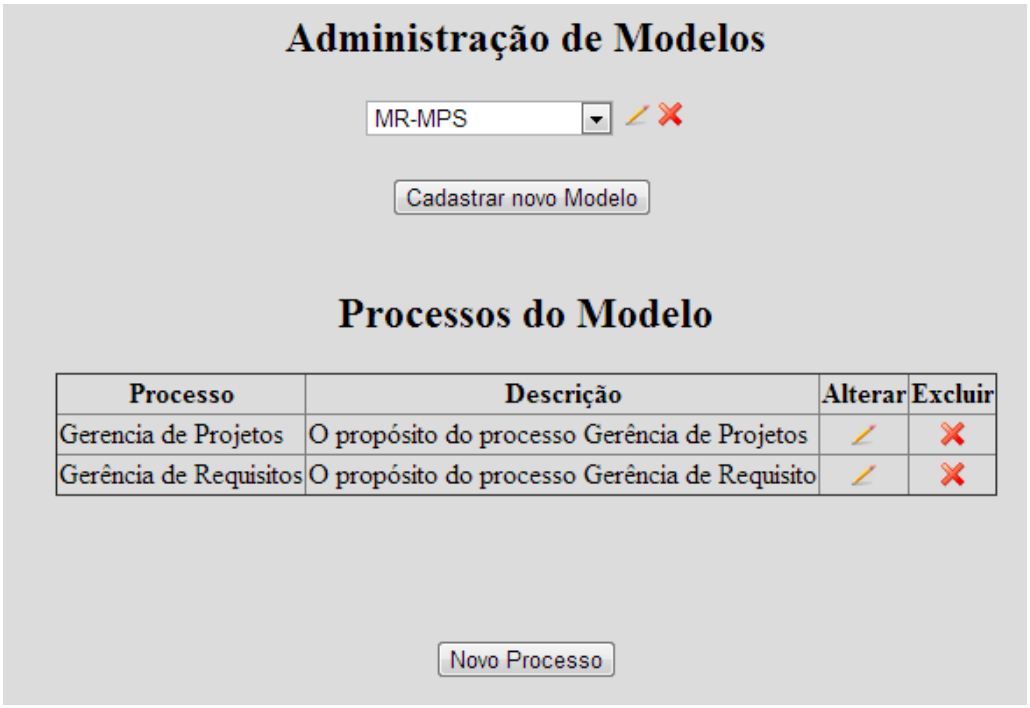

Figura 26 – Tela de Administração de Modelos

Figura 27 – Tela de Cadastramento de Modelos

Em seguida, observando-se a Figura 24, tem-se a atividade de cadastro dos processos dos clientes (Manter Processos do Cliente), onde os processos da organização a ser auditada serão cadastrados. Para cada processo, podem-se adicionar atividades, tarefas, passos, ou na profundidade de especificidade sob a qual a organização desenhou seus processos (por exemplo, atividades, tarefas, passos, sub-passos, itens, etc.), indica-se também a qual modelo de referência o processo está associado. Cada atividade ou tarefa adicionada ao processo deve ser associada a um resultado esperado do modelo de referência. A Figura 28 apresenta o cadastramento de processos do Cliente.

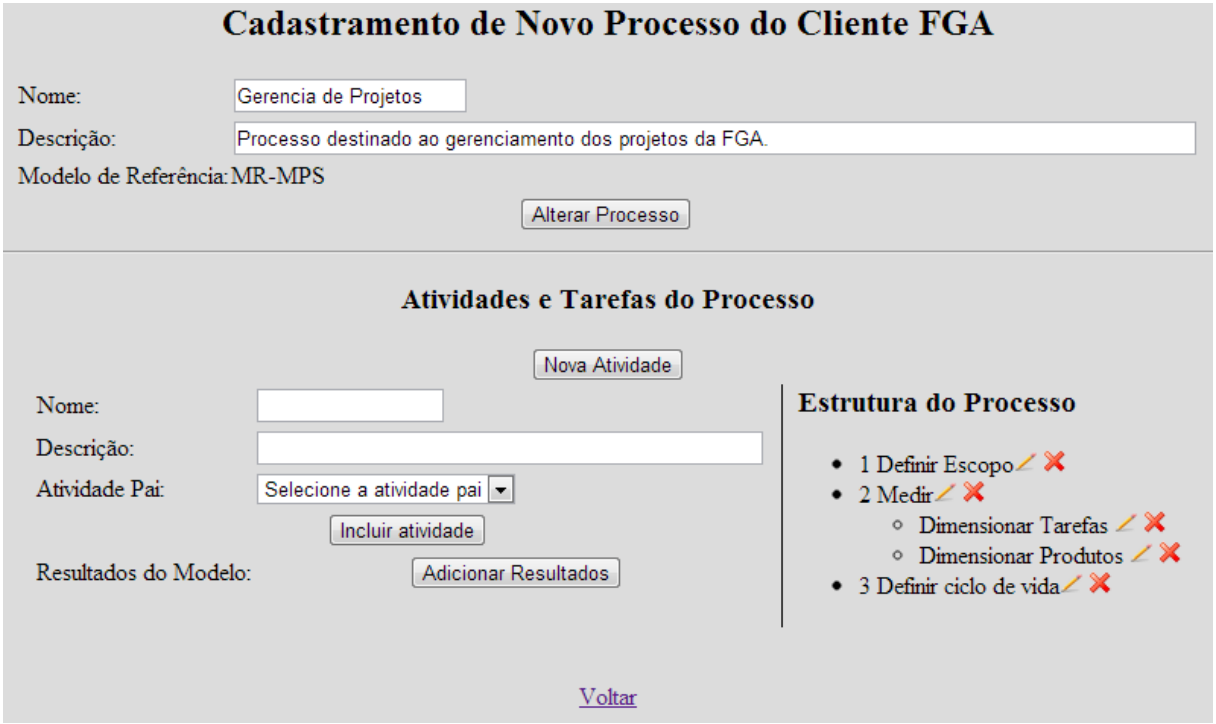

Figura 28 – Tela de Cadastramento de Processos do Cliente

O intuito da atividade de manutenção de processos do cliente é mapear os processos da organização com os processos e resultados esperados do modelo de referência.

A organização ou o cliente podem ter (devem ter) diversos projetos cadastrados para ser auditado, daí a importância da atividade de manutenção de projetos (Manter Projetos). Ao cadastrar um projeto para determinado cliente, pode-se inclusive selecionar quais os processos do cliente que estão sendo ou que serão usados para aquele determinado projeto, indica-se também o Gerente do Projeto, aquele responsável por receber as não-conformidades abertas em auditorias de

seus projetos. Importante ressaltar que, quando um projeto é cadastrado, todos os processos do cliente cadastrados são automaticamente associados, porém, podem ser removidos conforme necessidade. A Figura 29 apresenta a tela para cadastro de projetos dos clientes mantidos no sistema.

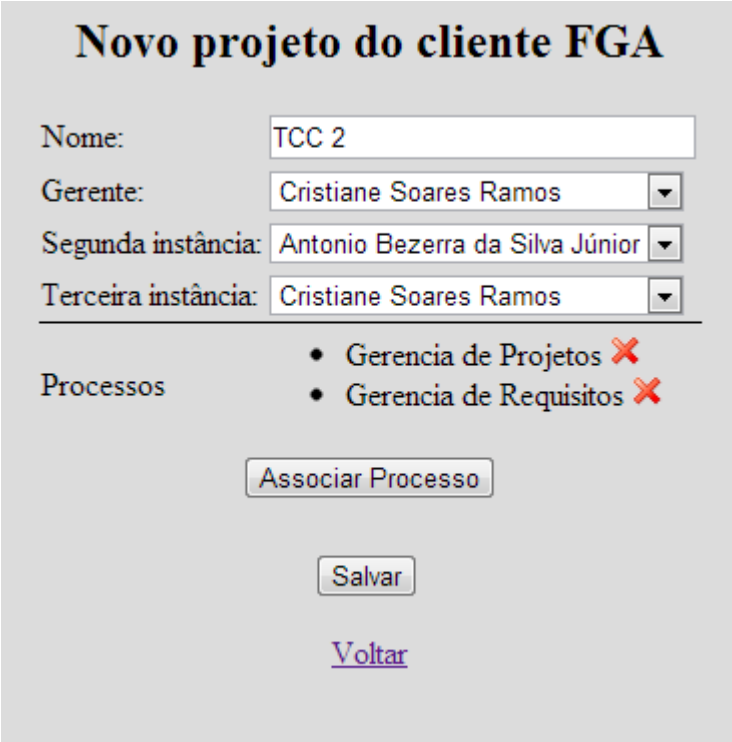

Figura 29 – Tela de Cadastro de Projeto de Clientes

Nesta funcionalidade também estão disponíveis as opções de cadastro de “segunda e terceira instâncias”, este cadastro tem como finalidade o conhecimento das unidades de escalonamento, ou seja, para quem serão designadas as não-conformidades no caso de o Gerente de Projetos não resolvê-las no tempo previsto. Importante ressaltar que para esta primeira versão do software QUASAR, a funcionalidade de escalonamento não se encontra habilitada.

Passando-se para a atividade de manutenção de critérios (Manter Critérios), nesta, são cadastrados os *checklists* referentes aos processos de cada cliente. Estes *checklists* ou critérios (como são chamados dentro do software) apresentam o cliente ao qual se referem, o modelo de referência e processo do modelo no qual se baseiam e, para cada questão cadastrada, a atividade que está associada. Sendo assim, ao cadastrar um critério, tem-se uma série de questões associadas às atividades e tarefas do cliente e que serão avaliadas conforme determinado modelo de referência. Neste ponto, já se sabe a partir do mapeamento realizado

previamente, a qual resultado esperado do modelo cada atividade ou tarefa do processo do cliente está cobrindo. A Figura 30 apresenta a tela para cadastro de um checklist do cliente.

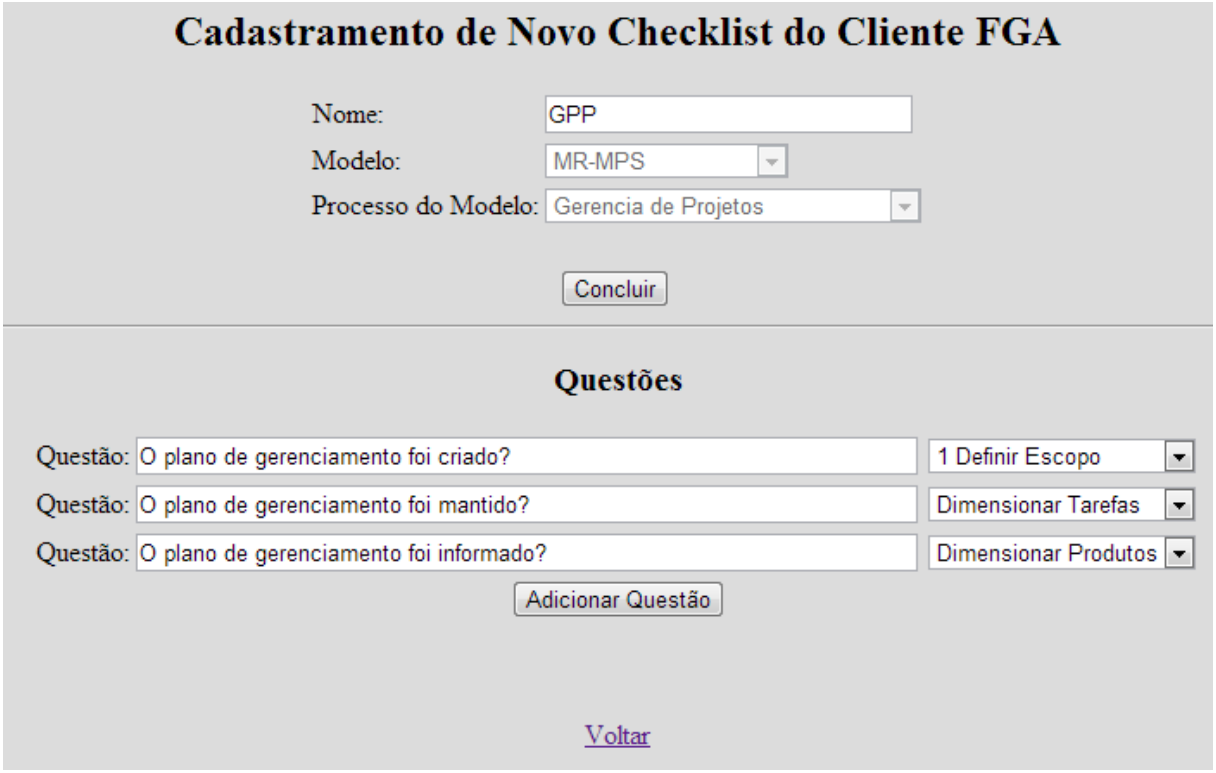

Figura 30 – Tela de cadastro de Checklist do Cliente

A partir deste ponto, tem-se um mapeamento completo de modelo de referência com processos do cliente e *checklists* associados, pode-se, então, realizar as auditorias de cada projeto, passando neste ponto para a atividade de Manter Auditoria.

Nesta atividade, são cadastradas as auditorias, identificando o cliente, projeto e processos que serão auditados, bem como o auditor responsável por esta auditoria. Automaticamente são importados os critérios (*checklists*) que serão usados para auditar cada processo, mais uma vez, de acordo com o mapeamento apresentado anteriormente. Importante ressaltar que, ao selecionar o auditor responsável, somente aparecerão auditores ou gerentes de qualidade que não estão associados ao projeto a ser auditado. A Figura 31 apresenta a tela de cadastro de uma auditoria.

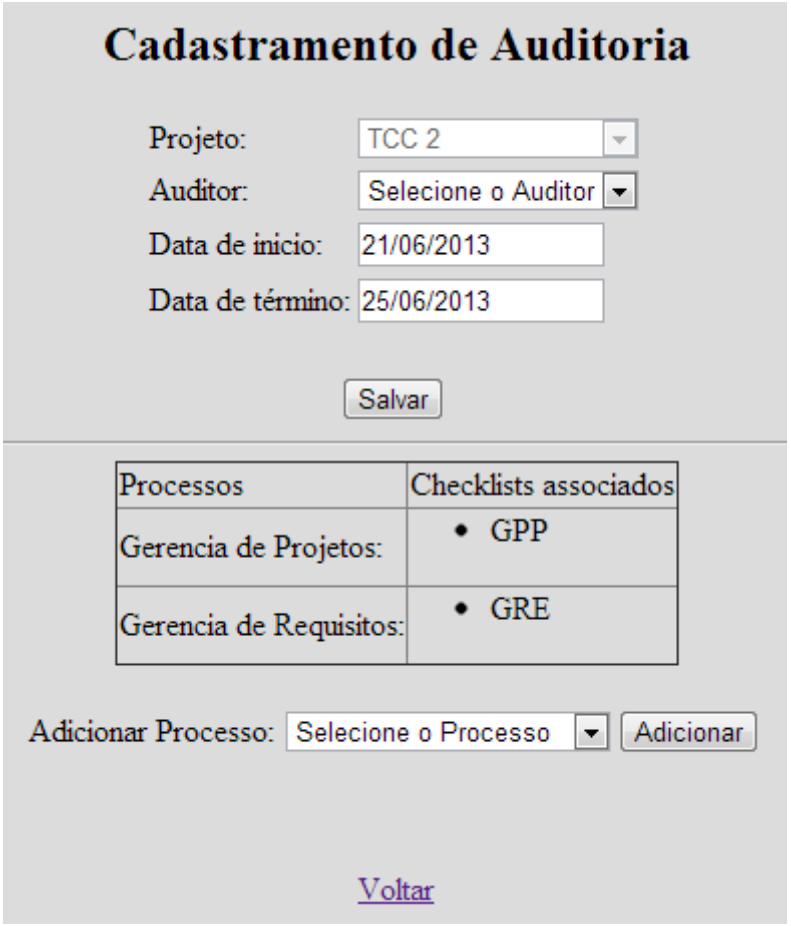

Figura 31 – Tela de cadastro de auditoria

Ainda nesta atividade, são especificadas as datas de início e término previstas para a auditoria, sendo que os campos são validados conforme as entradas possíveis para as datas, ou seja, não é possível agendar uma auditoria para datas passadas e a data de término não pode ser anterior à data de início. O cadastramento das datas impacta na apresentação visual das auditorias, que serão marcadas com o fundo na cor vermelha caso estejam em atraso.

Assim que uma auditoria for agendada ou cadastrada, é possível executá-la. Para tanto, são carregados os processos envolvidos na auditoria e os questionários para cada um destes, de maneira que, as questões referentes a cada processo são agrupadas para facilitar a leitura, compreensão e preenchimento dos *checklists*. Paralelamente à execução da auditoria, não-conformidades podem ser abertas, na medida em que são encontradas. Ao selecionar a resposta “Em não-conformidade” para determinada questão, surge a opção para gerenciamento de não-conformidades (Gerenciar Não-conformidades). Importante observar que, para cada

questão podem haver uma ou várias não-conformidades associadas, ficando a cargo do auditor responsável o cadastramento das mesmas.

A qualquer momento da execução da auditoria, as respostas podem ser salvar sem que a auditoria seja finalizada. Desta maneira, salvando-se um estado da auditoria, as não-conformidades abertas ainda não recebem uma "data de abertura", somente serão notificadas ao Gerente de Projetos e constarão como abertas no momento em que a auditoria for finalizada, porém, estatisticamente, ao se solicitar um relatório de não-conformidades, estas constarão como abertas. A Figura 32 apresenta a tela onde são preenchidas as questões, ou seja, onde a auditoria é de fato realizada.

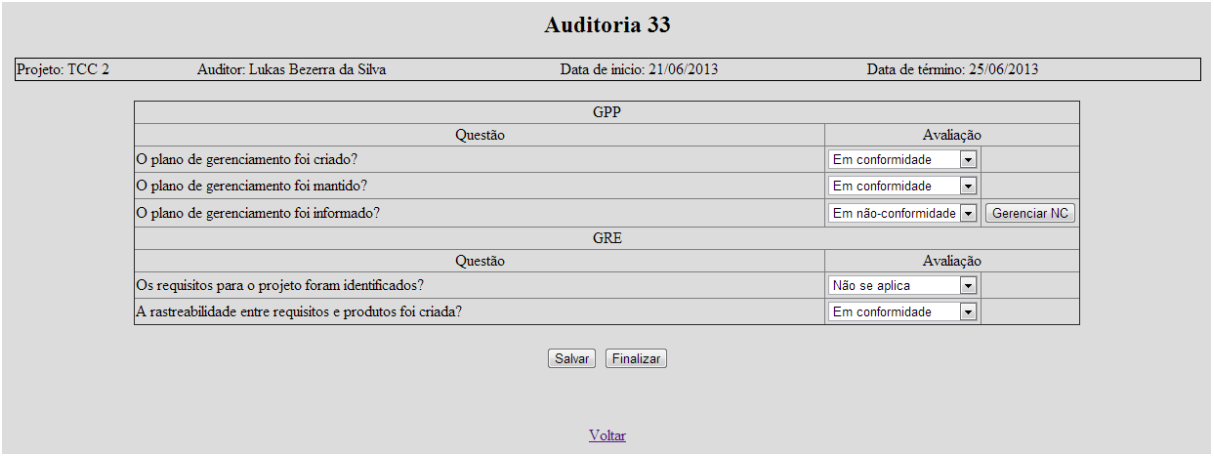

Figura 32 – Tela de execução de auditoria

Concluída a auditoria, o responsável pode, enfim, acessar a opção para "Finalizar a auditoria", momento em que, todas as não-conformidades encontradas receberão data de abertura (coincidente com a data de encerramento da auditoria), os gerentes de projeto serão notificados e as não-conformidades encontradas podem ser acompanhadas até sua solução.

Durante a solução das não-conformidades, as mesmas podem assumir os estados (i) Em Aberto, quando a auditoria é finalizada e os responsáveis são notificados; (ii) Em Solução, quando os responsáveis já estão cientes das não-conformidades e sua solução já está sendo providenciada; (iii) Solucionada, quando os responsáveis já obtiveram a solução das não-conformidades; e (iv) Finalizada, quando o(s) auditor(es) responsável(is) concordam com a solução dada para as não-conformidades, encerrando-se assim, o ciclo de vida das não-conformidades.

Por fim, a atividade de destaque do software QUASAR, a avaliação dos processos (Avaliar Processo). Nesta atividade, é selecionado um cliente e um modelo de referência que será a base para a avaliação. Em seguida, seleciona-se o nível de maturidade do qual se deseja saber se a organização pode ou não solicitar uma avaliação, quais resultados do modelo encontram-se com problemas, qual a porcentagem de cobertura dos processos, quais as não-conformidades associadas, ou seja, em que ponto do processo deve ser dada mais atenção antes de solicitar uma avaliação de maturidade.

Quando se seleciona um nível de maturidade do modelo, automaticamente são resgatados os processos e resultados esperados que compõem aquele nível (previamente cadastrados e mantidos na ferramenta). Após identificar de maneira automatizada quais os processos do modelo de referência, encontram-se os processos, atividades e tarefas da organização que satisfazem aos resultados esperados. Neste ponto, já é possível calcular a porcentagem de processos do modelo mapeados com processos da organização. Indo mais além, são encontradas as não-conformidades impactantes em cada uma das atividades e tarefas que compõem os diversos projetos auditados para aquele cliente. A Figura 33 apresenta a tela onde é solicitada a avaliação de determinado processo com relação a um nível de maturidade do modelo de referência escolhido.

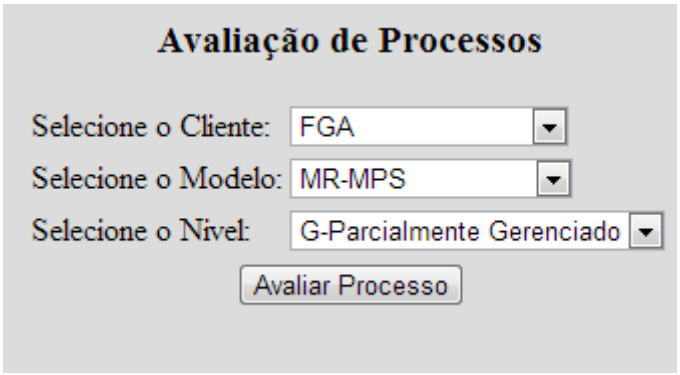

Figura 33 – Tela de solicitação de avaliação de processo

Um relatório completo é gerado, apresentando cada processo daquele nível de maturidade, seus resultados esperados e, quais os resultados cobertos e não cobertos por atividades ou tarefas do cliente. Para cada resultado, são apresentadas as não-conformidades associadas, possibilitando assim, ações corretivas nos processos, de maneira a cobrir 100% (cem por cento) do modelo de referência, no nível desejado, e deixar totalmente “em conformidade” com as práticas

estabelecidas, daí a importância deste relatório final para a organização. O relatório gerado é apresentado na Figura 34.

Resultado de Análise de Processos

Instituição: FGA **Modelo:** MR-MPS
Nivel de Maturidade: G - Parcialmente Gerenciado **Resultado:** Não Apto
Total de Resultados não cobertos: 18 (75.000 %)
Total de Resultados com não-conformidades: 1 (4.167 %)

Gerencia de Projetos			
Resultado Esperado	Atividade	Status	Não-conformidades
GPR1	1 Definir Escopo	Nao OK	• 23 - PLANO AUSENTE
GPR10	-	NAO COBERTO	-
GPR11	-	NAO COBERTO	-
GPR12	-	NAO COBERTO	-
GPR13	-	NAO COBERTO	-
GPR14	-	NAO COBERTO	-
GPR15	-	NAO COBERTO	-
GPR16	-	NAO COBERTO	-
GPR17	-	NAO COBERTO	-
GPR18	-	NAO COBERTO	-
GPR19	-	NAO COBERTO	-
GPR2	Dimensionar Tarefas	ok	-
GPR2	Dimensionar Produtos	ok	-
GPR3	3 Definir ciclo de vida	ok	-
GPR4	-	NAO COBERTO	-
GPR5	-	NAO COBERTO	-
GPR6	-	NAO COBERTO	-
GPR7	-	NAO COBERTO	-
GPR8	-	NAO COBERTO	-
GPR9	-	NAO COBERTO	-

Figura 34 – Relatório de avaliação de processo

CAPÍTULO 5 – CONCLUSÃO E TRABALHOS FUTUROS

Através do desenvolvimento do software QUASAR, após todo o ciclo de vida do projeto, obteve-se uma ferramenta de apoio ao processo de Garantia da Qualidade do Processo e do Produto, conforme objetivo geral deste trabalho.

O sucesso do projeto somente foi possível pela realização dos objetivos específicos que previam o estudo de características da Garantia da Qualidade dentro dos principais modelos de referência, partindo para a investigação e discussão das funcionalidades de ferramentas de apoio a este processo (GQA), para assim, planejar, desenvolver e documentar a ferramenta QUASAR, conforme apresentado na seção 1.3 (Objetivos Específicos) do Capítulo 1 (Introdução).

O produto final do projeto, o software QUASAR, é uma ferramenta de apoio à equipe de Garantia de Qualidade capaz de manter dados organizacionais importantes que auxiliam nas tomadas de decisão do Gerente de Projetos, permitindo uma visão geral sobre os processos vigentes e possíveis problemas e inadequações, seja com relação a padrões estabelecidos ou modelos de referência escolhidos. Esta pode ser considerada a grande colaboração do trabalho.

Outra colaboração está no fato de todo o processo de desenvolvimento estar apresentado, seguindo-se práticas da Engenharia de Software, documentando-se cada etapa e, focando esta documentação principalmente nos padrões e estilos de arquitetura, padrões de projetos usados, diagramas e modelagens, fator este que torna possível o estudo, a compreensão e a manutenção evolutiva da ferramenta.

A presença da documentação detalhada da ferramenta tem o intuito de possibilitar a continuidade ao projeto QUASAR, elevando o nível do software. Foram previstas melhorias a serem implementadas como trabalhos futuros. Estas melhorias incluem funcionalidades que são de extrema importância para a realização dos trabalhos da equipe de qualidade. São elas:

- **Possibilidade de decisão de prazo de escalonamento** – No momento do cadastro de cada projeto, ao especificar o Gerente de Projetos, a Primeira e a Segunda instâncias, possibilitar o cadastro do prazo para escalonamento em cada instância, ou seja, definir os prazos para que, caso a não-conformidade não seja resolvida, passe para o próximo responsável (primeira e segunda instâncias de resolução);

- **Versões de processos** – Para cada processo cadastrado, interessante haver a possibilidade de especificar a versão do processo no momento do cadastro. Esta funcionalidade permite que se selecione a versão do processo a ser usada em cada projeto da organização;
- **Definir reincidência de não-conformidades** – Importante para fins de estatísticas e relatórios, a análise de reincidência de NC. No momento da abertura de uma não-conformidade, possibilitar que seja informado se a mesma refere-se a uma reincidência de determinado problema encontrado;
- **Possibilidade de atribuição de não-conformidade** – Importante que o gerente de projetos possa redirecionar uma não-conformidade para desenvolvedores ou analistas, ou seja, repassar a não-conformidade para os responsáveis diretos pela solução, que os mesmos possam ter acesso ao sistema;
- **Geração de gráficos** – No momento da consulta de relatórios, incluir nestes a geração de gráficos com as informações, de maneira a facilitar a visualização e análise do estado dos processos; e
- **Atributos de processos** – Deve ser possível a análise dos processos com relação aos atributos de processos previstos no modelo MR-MPS. Atributos de processos podem determinar fortes possibilidades de erros na execução dos processos, seu registro é importante pois, nos relatórios finais podem ser apresentadas “fraquezas” nos processos referentes aos atributos.
- **Reestruturação da interface de usuário** – Desenvolver interface com o usuário que vise maior produtividade na execução das tarefas, como por exemplo, facilitadores para organização dos processos, ações vinculadas ao *click* do botão direito do mouse, entre outras.

Todas as melhorias propostas devem observar a atual arquitetura do software, os estilos e padrões definidos e apresentados. A atual arquitetura se mostra estável e passível de expansão.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, C. E. R. **Ensino de programação recursiva em Ciência da Computação**. Junho de 2006.

ANDRADE, J. M. S. **Avaliação de processos de software em ambientes de desenvolvimento de software orientados à organização**. Rio de Janeiro. 2005

Associação Brasileira De Normas Técnicas, ABNT. **NBR ISO/IEC 25000. Engenharia de software - Requisitos e avaliação da qualidade de produtos de software (SQuaRE) - Guia do SQuaRE**. Rio de Janeiro, 2008.

Associação para Promoção da Excelência do Software Brasileiro – SOFTEX[1]. **MPS.BR – Guia Geral MPS de Software**. Campinas-SP: SOFTEX, Dezembro de 2012. Disponível em: <http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_Geral_Software_2012.pdf>. Acessado em Janeiro de 2013

Associação para Promoção da Excelência do Software Brasileiro – SOFTEX[2]. **MPS.BR – Guia para Implementação do Nível F do MR-MPS: 2011**. Campinas-SP: SOFTEX, Julho 2011. Disponível em: <http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_de_Implementacao_Parte_2_2011.pdf>. Acessado em Janeiro de 2013

BARCELLOS, M., FIGUEIREDO, S., ROCHA, A. R., et al. **Utilização de Métodos Paramétricos, Analogias, Julgamento de Especialistas e Conhecimento Organizacional no Planejamento de Tempo e Custos de Projetos de Software**. II Simpósio Brasileiro de Qualidade de Software, Fortaleza, Brasil. 2003.

BARTIË, **A. Garantia da qualidade de software**. São Paulo: Campus, 2002.

CAMPO, M. **Why CMMI maturity level 5?**, *CrossTalk*, v. 25, n. 1, pp. 15-18. 2012

ELLISLAB. **CodeIgniter User Guide Version 2.1.3. 2012**. Disponível em: <http://ellislab.com/codeigniter/user-guide/> Acessado em Fevereiro de 2013

ELMASRI, R., NAVATHE, S. B. **Sistemas de Bancos de Dados. 6ª Edição**. Editora: Pearson Education, 2010.

FAYAD, Mohamed E., SCHMIDT, Douglas C. **Object-Oriented Application Frameworks.** Outubro de 1997.

FERREIRA, A.I.F., SANTOS, G., CERQUEIRA, R.*, et al.* **ROI of software process improvement at BL informática: SPIdex is really worth it**, *Software Process Improvement and Practice*, v. 13, n. 4, pp. 311-318. 2008.

FERREIRA, A.I.F., SANTOS, G., CERQUEIRA, R.*, et al.***Applying ISO 9001:2000, MPS.BR and CMMI to achieve software process maturity: BL informatica's pathway**, pp. 642-651, Minneapolis, MN. 2007.

FIGUEIREDO, S., SANTOS, G., ROCHA, A. R. **Gerência de Configuração em Ambientes de Desenvolvimento de Software Orientados à Organização**, III Simpósio Brasileiro de Qualidade de Software, maio, Brasília, DF, Brasil. 2004.

GAMMA, E., HELM, R., JOHNSON, R., VLISSIDES, J. **Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos**. Editora Bookman. 2006. 364p.

LARMAN, Craig. **Utilizando UML e Padrões: Uma Introdução a Análise e ao Projeto Orientado a Objetos**. 3a. edição. Bookman, 2007

LOOS, A. **Processo de Garantia da Qualidade Baseado no Modelo MPS.BR**. Blumenau. 2009.

MCT/SEPIN – Secretaria de Política de Informática e Automação / Ministério da Ciência e Tecnologia. **Qualidade e Produtividade no Setor de Software Brasileiro 2009**. Brasília. 2010.

MONTONI, M. **Aquisição de Conhecimento: Uma Aplicação no Processo de Desenvolvimento de Software., Tese de MSc**., COPPE/UFRJ, Rio de Janeiro, RJ.

Brasil. 2003.

NETO, P.J.S., ABIB, G., GOMEL, M.M.G.*, et al.* ***Evolução da Qualidade de Software no Brasil de 1994-2010 baseada nas pesquisas e projetos do PBQP Software***, MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO, Brasília. 2011

OLIVEIRA, S., VASCONCELOS, A. e ROUILLER, A. **Uma Proposta de Ambiente de Implementação de Processo de Software**. 2005. Em: Infocomp – Journal of Computer Science, Vol. 4, páginas 71 – 78.

PAULK, M., CURTIS, B., CHRISSIS, M. B., WEBER, C. **Capability Maturity Model for Software: Guidelines for Improving the Software Process.** 1994.

PEIXOTO, D.C.C., BASTISTA, V.A., RESENDE, R.F.*, et al.* **A case study of Software Process Improvement implementation**, pp. 716-721, Redwood City, CA. 2010

PRESSMAN, Roger S. **Engenharia de Software. 1ª. Edição.** Editora Mcgraw-hill, 1995. 1028 p.

PRESSMAN, Roger S. **Engenharia de Software. 6ª. Edição**. Editora Mcgraw-hill, 2006. 720 p.

RATIONAL. **Rational Unified Process: Best Pratices for Software Development Teams**. 2001. Disponível em: <http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf>. Acessado em Abril de 2013

ROCHA, A. R.; MONTONI, M.; SANTOS, G.; MAFRA, S.; FIGUEIREDO, S.; BESSA, A.; MIAN, P. **Estação TABA: Uma Infra-estrutura para Implantação do Modelo de Referência para Melhoria de Processo de Software**. IV SBQS, Porto Alegre, Brasil. 2005.

SANTOS, G., KALINOWSKI, M., ROCHA, A.R.*, et al.* **MPS.BR: A tale of software process improvement and performance results in the Brazilian software industry**, pp. 412-417, Porto. 2010

SCHNAIDER, L.R.C. **Planejamento da Alocação de Recursos Humanos em Ambientes de Desenvolvimento de Software Orientados à Organização, Tese de MSc., COPPE/UFRJ**. 2003. Rio de Janeiro, RJ, Brasil.

SILVA, A. A. C., SILVA, E. J. F., PORTELA, C. S., VASCONCELOS, A. M. L., OLIVEIRA, S. R. B. **Spider-PE: Uma Ferramenta de Apoio à Implementação da Capacidade do MR-MPS Nível F e CMMI-DEV Nível 2.** VIII Workshop Anual do MPS.BR, 2012, Campinas. WAMPS. , 2012.

SILVA, A. M. R., VIDEIRA, C. A. E. **UML, Metodologias e Ferramentas CASE:** Linguagem de Modelação UML, Metodologias e Ferramentas CASE na Concepção e Desenvolvimento de Software**. 1ª. Edição**. Editora: Centro Atlântico PT, 2001. 552p.

SOFTEXPERT. **Catálogo SE Audit: Gestão de Auditorias**. Florianópolis. Disponível em: <http://www.softexpert.com.br/catalogos/SE_Audit.pdf>

Software Engineering Institute – SEI. **CMMI for Development. V.1.3**. Pittsburgh: SEI, Novembro de 2010. Disponível em: <http://www.sei.cmu.edu/reports/10tr033.pdf>

SOLINGEN, R.V. **Measuring the ROI of software process improvement**, *IEEE Software*, v. 21, n. 3, pp. 32-38. 2004

SOMMERVILLE, Ivan. **Engenharia de Software. 8ª. Edição**. Editora: Pearson Education, 2007. 568p.

TAYLOR, R. N., MEDVIDOVIC, N. AND DASHOFY, E. M. **Software Architecture: Fountadions, Theory and Pratice**. Wiley Publishing. 2009.

TERRY, B., LOGEE, D. **Terminology for Software Engineering Environment (SEE) and Computer-Aided Software Engineering (CASE).** Software Engineering Notes. ACM SIGSOFT. Abril 1990.

TRAVASSOS, G.H., KALINOWSKI, M. **iMPS 2011 Resultados de Desempenho das Empresas que Adotaram o Modelo MPS de 2008 a 2011, Softex**. 2012

APÊNDICE I: Detalhamento dos Casos de Uso

Nome	**Manter Projetos**
Descrição	Este caso de uso faz referência à capacidade do sistema em inserir, alterar e excluir os projetos vigentes na organização.
Fluxo Básico	1. Este caso de uso se inicia quando o Gerente de Qualidade acessa a opção para administração de projetos; 2. É exibida uma lista com todos os projetos cadastrados até o momento; 3. O Gerente de Qualidade seleciona a opção para adicionar novo projeto; 4. O Gerente de Qualidade insere os dados do novo projeto (Nome, Estado, Visibilidade, Descrição e Artefatos gerados); 5. O Gerente de Qualidade salva os dados do novo projeto; 6. Fim do caso de uso.
Fluxo Alternativo 1 (Alterar Projeto)	1. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para alteração de determinado projeto; 2. O Gerente de Qualidade altera os dados do projeto selecionado; 3. O Gerente de Qualidade salva os dados do projeto; 4. Fim do FA1;
Fluxo Alternativo 2 (Excluir Projeto)	5. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para exclusão de determinado projeto; 6. O Gerente de Qualidade confirma a exclusão; 7. O projeto é excluído; 8. Fim do FA2;
Condições Prévias	• Para exclusão ou alteração, deve haver projetos cadastrados.
Condições Posteriores	• Projetos inseridos, alterados ou excluídos.
Regra de Negócio	NA

Nome	**Manter Processos**
Descrição	Este caso de uso faz referência à capacidade do sistema em inserir, alterar e excluir os processos e atividades da organização.
Fluxo Básico	1. Este caso de uso se inicia quando o Gerente de Qualidade acessa a opção para administração de processos; 2. É exibida uma lista com todos os processos cadastrados até o momento; 3. O Gerente de Qualidade seleciona a opção para adicionar novo processo; 4. O Gerente de Qualidade insere os dados do novo processo (Nome, Descrição, Atividades, Tarefas, Artefatos, Entradas e Saídas); 5. O Gerente de Qualidade salva os dados do novo processo; 6. Fim do caso de uso.
Fluxo Alternativo 1 (Alterar Projeto)	2.1 Após o item 2 do FB, o Gerente de Qualidade seleciona opção para alteração de determinado processo; 2.2 O Gerente de Qualidade altera os dados do processo selecionado; 2.3 O Gerente de Qualidade salva os dados do processo; 2.4 Fim do FA1;
Fluxo Alternativo 2 (Excluir Projeto)	2.1 Após o item 2 do FB, o Gerente de Qualidade seleciona opção para exclusão de determinado processo; 2.2 O Gerente de Qualidade confirma a exclusão; 2.3 O processo é excluído; 2.4 Fim do FA2;
Condições Prévias	Para exclusão ou alteração, deve haver processos cadastrados.
Condições Posteriores	Processos inseridos, alterados ou excluídos.
Ponto de Extensão	NA
Regra de Negócio	• Um processo deve necessariamente ter atividade(s) associada(s); • As atividades devem necessariamente ser organizadas cronologicamente; • Uma atividade pode ter uma ou mais saídas; • Uma atividade deve ter tarefa(s) associada(s).

Nome	**Avaliar processo**
Descrição	Este caso de uso faz referência à capacidade do sistema em avaliar os processos da organização de acordo com um modelo de referência (o modelo padrão é o MPS.Br). Quando um processo é avaliado, pode ser selecionado o tipo da avaliação, podendo ser por Nível de Capacidade (avaliação de apenas um processo) ou Nível de Maturidade (avaliação do conjunto de processos da organização).
Fluxo Básico	1. Este caso de uso se inicia quando o Gerente de Qualidade acessa a opção para avaliação de processos; 2. É exibida uma lista com todos os processos cadastrados até o momento; 3. O Gerente de Qualidade seleciona o processo que deseja avaliar; 4. O Gerente de Qualidade seleciona o modelo de referência para o qual o processo será avaliado; 5. O processo é avaliado e seu nível de capacidade é exibido para o usuário; 6. Fim do caso de uso.
Fluxo Alternativo 1 (Alterar Projeto)	1. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para avaliação do nível de maturidade da empresa; 2. O Gerente de Qualidade seleciona o modelo de referência para o qual o processo será avaliado; 3. Os processos são avaliados e o nível de maturidade é exibido para o Gerente de Qualidade; 4. Fim do FA1;
Condições Prévias	• Para avaliação do nível de maturidade ou capacidade, deve haver ao menos um processo cadastrado. • Para avaliação deve haver ao menos um modelo de referência cadastrado.
Condições Posteriores	• Processo(s) avaliado(s) e nível de maturidade ou capacidade gerado e exibido ao usuário.
Ponto de Extensão	NA
Regra de Negócio	NA

Nome	Manter Modelos
Descrição	Este caso de uso faz referência à capacidade do sistema em inserir, alterar ou excluir os modelos de referência adotados pela organização. O modelo de referência padrão do sistema é o MPS.Br. Através desta funcionalidade é possível manter também os níveis de avaliação (no caso do MPS.Br seriam os níveis A,B,C,D,E,F e G), indicando quais são os processos que devem constar em cada nível.
Fluxo Básico	1. Este caso de uso se inicia quando o Gerente de Qualidade acessa a opção para administração de Modelos de Referência; 2. É exibida uma lista com todos os modelos de referência cadastrados até o momento; 3. O Gerente de Qualidade seleciona a opção para adicionar novo modelo; 4. O Gerente de Qualidade insere os dados do novo modelo (Nome, Descrição, Processos e Práticas); 5. O Gerente de Qualidade adiciona novos Níveis; 6. O Gerente de Qualidade associa os processos e práticas aos níveis; 7. O Gerente de Qualidade salva os dados do novo modelo; 8. Fim do caso de uso.
Fluxo Alternativo 1 (Alterar Projeto)	1. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para alteração de determinado modelo; 2. O Gerente de Qualidade altera os dados do modelo selecionado; 3. O Gerente de Qualidade salva os dados do modelo; 4. Fim do FA1;
Fluxo Alternativo 2 (Excluir Projeto)	1. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para exclusão de determinado modelo; 2. O Gerente de Qualidade confirma a exclusão; 3. O modelo é excluído; 4. Fim do FA2;
Condições Prévias	• Para exclusão ou alteração, deve haver modelo(s) cadastrado(s).
Condições Posteriores	• Modelo(s) inserido(s), alterado(s) ou excluído(s).
Ponto de Extensão	NA
Regra de Negócio	NA

Nome	**Manter Usuários**
Descrição	Este caso de uso faz referência à capacidade do sistema em inserir, alterar e excluir usuários com permissão para acessar o sistema. Este caso de uso é executado pelo Gerente de Qualidade.
Fluxo Básico	1. Este caso de uso se inicia quando o Gerente de Qualidade acessa a opção para administração de usuários; 2. É exibida uma lista com todos os usuários cadastrados até o momento; 3. O Gerente de Qualidade seleciona a opção para adicionar novo usuário; 4. O Gerente de Qualidade insere os dados do novo usuário (Nome, Matrícula, Tipo, Email e Senha Provisória); 5. O Gerente de Qualidade salva os dados do novo usuário; 6. Fim do caso de uso.
Fluxo Alternativo 1 (Alterar Projeto)	1. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para alteração de determinado usuário; 2. O Gerente de Qualidade altera os dados do usuário selecionado; 3. O Gerente de Qualidade salva os dados do usuário; 4. Fim do FA1;
Fluxo Alternativo 2 (Excluir Projeto)	1. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para exclusão de determinado usuário; 2. O Gerente de Qualidade confirma a exclusão; 3. O usuário é excluído; 4. Fim do FA2;
Condições Prévias	• Para exclusão ou alteração, deve haver usuário(s) cadastrado(s).
Condições Posteriores	• Usuário(s) inserido(s), alterado(s) ou excluído(s).
Ponto de Extensão	NA
Regra de Negócio	• Apenas o Gerente pode acessar este Caso de Uso.

Nome	**Manter Auditorias**
Descrição	Este caso de uso faz referência à capacidade do sistema em inserir, alterar e excluir auditorias, designando um responsável para cada uma delas. Este caso de uso é realizado pelo Gerente de Qualidade.
Fluxo Básico	1. Este caso de uso se inicia quando o Gerente de Qualidade acessa a opção para administração de auditorias; 2. É exibida uma lista com todas as auditorias cadastradas até o momento; 3. O Gerente de Qualidade seleciona a opção para adicionar nova auditoria; 4. O Gerente de Qualidade insere os dados da nova auditoria (Data, Responsável, Objetos a auditar); 5. O Gerente de Qualidade salva os dados do novo usuário; 6. Fim do caso de uso.
Fluxo Alternativo 1 (Alterar Projeto)	1. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para alteração de determinada auditoria 2. O Gerente de Qualidade altera os dados da auditoria selecionada; 3. O Gerente de Qualidade salva os dados da auditoria; 4. Fim do FA1;
Fluxo Alternativo 2 (Excluir Projeto)	1. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para exclusão de determinada auditoria; 2. O Gerente de Qualidade confirma a exclusão; 3. A auditoria é excluída; 4. Fim do FA2;
Condições Prévias	• Para exclusão ou alteração, deve haver auditoria(s) cadastrada(s).
Condições Posteriores	• Auditoria(s) inserida(s), alterado(s) ou excluído(s).
Ponto de Extensão	NA
Regra de Negócio	• Apenas o Gerente pode acessar este Caso de Uso.

Nome	**Manter Critérios**
Descrição	Este caso de uso faz referência à capacidade do sistema em inserir, alterar e excluir critérios objetivos, ou seja, administrar os checklists e itens de checklist. Este caso de uso é realizado pelo Gerente de Qualidade.
Fluxo Básico	1. Este caso de uso se inicia quando o Gerente de Qualidade acessa a opção para administração de critérios objetivos; 2. É exibida uma lista com todos os checklists cadastrados até o momento; 3. O Gerente de Qualidade seleciona a opção para adicionar novo checklist; 4. O Gerente de Qualidade insere os dados do novo checklist (Nome e Finalidade); 5. O Gerente de Qualidade adiciona novos itens de checklist (descrição e possíveis respostas); 6. O Gerente de Qualidade salva os dados do novo checklist; 7. Fim do caso de uso.
Fluxo Alternativo 1 (Alterar Projeto)	1. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para alteração de determinado checklist; 2. O Gerente de Qualidade altera os dados do checklist selecionada; 3. O Gerente de Qualidade salva os dados do checklist; 4. Fim do FA1;
Fluxo Alternativo 2 (Excluir Projeto)	1. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para exclusão de determinado checklist; 2. O Gerente de Qualidade confirma a exclusão; 3. O checklist é excluído; 4. Fim do FA2;
Condições Prévias	• Para exclusão ou alteração, deve haver checklist (s) cadastrada(s).
Condições Posteriores	• checklist (s) inserido(s), alterado(s) ou excluído(s).
Ponto de Extensão	NA
Regra de Negócio	NA

Nome	Manter Clientes
Descrição	Este caso de uso faz referência à capacidade do sistema em inserir, alterar e excluir clientes, ou seja, instituições as quais terceirizam seu processo de Garantia da Qualidade. Caso de uso realizado pelo Gerente de Qualidade.
Fluxo Básico	1. Este caso de uso se inicia quando o usuário acessa a opção para administração de clientes; 2. É exibida uma lista com todos os clientes cadastrados até o momento; 3. O Gerente de Qualidade seleciona a opção para adicionar novo cliente; 4. O Gerente de Qualidade insere os dados do novo cliente (Nome, Endereço, E-mail e Telefone); 5. O Gerente de Qualidade salva os dados do novo cliente; 6. Fim do caso de uso.
Fluxo Alternativo 1 (Alterar Projeto)	1. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para alteração de determinado cliente; 2. O Gerente de Qualidade altera os dados do cliente selecionada; 3. O Gerente de Qualidade salva os dados do cliente; 4. Fim do FA1;
Fluxo Alternativo 2 (Excluir Projeto)	1. Após o item 2 do FB, o Gerente de Qualidade seleciona opção para exclusão de determinado checklist; 2. O Gerente de Qualidade confirma a exclusão; 3. O checklist é excluído; 4. Fim do FA2;
Condições Prévias	• Para exclusão ou alteração, deve haver cliente(s) cadastrado(s).
Condições Posteriores	• Cliente (s) inserido(s), alterado(s) ou excluído(s).
Ponto de Extensão	NA
Regra de Negócio	NA

Nome	**Realizar Auditoria**
Descrição	Este caso de uso faz referência à capacidade do sistema em montar e disponibilizar os *checklists* utilizados para realizar auditorias de qualidade.
Fluxo Básico	1. Este caso de uso se inicia quando o Auditor acessa a opção para realizar auditoria; 2. É exibida uma lista com todas as auditorias cadastradas até o momento; 3. O Auditor seleciona a opção para realizar determinada auditoria; 4. O Auditor responde às questões dos *checklists*; 5. O Auditor salva os dados dos *checklists*; 6. Fim do caso de uso.
Fluxo Alternativo 1 (Alterar Projeto)	1. Após o item 4 do FB, o Auditor seleciona opção para encerrar a auditoria; 2. Fim do FA1;
Condições Prévias	• Para realização da auditoria, deve haver auditoria(s) cadastrada(s).
Condições Posteriores	• Auditoria (s) salva(s), ou encerrada(s).
Ponto de Extensão	NA
Regra de Negócio	NA

Nome	Manter NC
Descrição	Este caso de uso faz referência à capacidade do sistema em inserir, alterar e excluir não conformidades encontradas durante a realização de auditorias de qualidade.
Fluxo Básico	1. Este caso de uso se inicia quando o Auditor responde uma questão como “Em Não conformidade”; 2. O Auditor acessa a opção para manter não conformidades; 3. É exibida uma lista com todas as não conformidades cadastradas até o momento para determinada questão; 4. O Auditor seleciona a opção para inserir uma nova não conformidade; 5. O Auditor insere os dados da não conformidade (Nome, Descrição); 6. O Auditor salva os dados da não conformidade; 7. Fim do caso de uso.
Fluxo Alternativo 1 (Alterar Projeto)	5. Após o item 4 do FB, o Auditor seleciona opção para alteração de determinada não conformidade; 6. O Auditor altera os dados da não conformidade; 7. O Auditor salva os dados da não conformidade; 8. Fim do FA1;
Fluxo Alternativo 2 (Excluir Projeto)	5. Após o item 4 do FB, o Auditor seleciona opção para exclusão de determinada não conformidade; 6. O Auditor confirma a exclusão; 7. A não conformidade é excluída; 8. Fim do FA2;
Condições Prévias	• Para exclusão ou alteração, deve haver não conformidade (s) cadastrada(s).
Condições Posteriores	• Não conformidade (s) incluída(s), alterada(s) ou excluída(s).
Ponto de Extensão	NA
Regra de Negócio	NA

Nome	**Alterar NC**
Descrição	Este caso de uso faz referência à capacidade do sistema em alterar não conformidades encontradas durante a realização de auditorias de qualidade.
Fluxo Básico	1. Este caso de uso se inicia quando o Gerente de Projetos seleciona a opção para visualizar as não conformidades; 2. É exibida uma lista com todas as não conformidades cadastradas até o momento para determinada questão; 3. O Gerente de Projetos seleciona a não conformidade que deseja alterar; 4. O Gerente de Projetos altera os dados (Estado e Comentário) da não conformidade; 5. O Gerente de Projetos salva os dados; 6. Fim do caso de uso.
Condições Prévias	• Para alteração deve haver não conformidade(s) cadastrada(s) e atribuída(s) para o Gerente de Projetos.
Condições Posteriores	• Não conformidade (s) alterada(s).
Ponto de Extensão	NA
Regra de Negócio	NA