UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ

CAMPUS DE FOZ DO IGUAÇU

PROGRAMA DE PÓS-GRADUAÇÃO EM
ENGENHARIA DE SISTEMAS DINÂMICOS E ENERGÉTICOS

DISSERTAÇÃO DE MESTRADO

APLICAÇÃO DE PRINCÍPIOS DE QUALIDADE DE DADOS DURANTE O DESENVOLVIMENTO DE UM SISTEMA COMPUTACIONAL MÉDICO PARA A CIRURGIA COLOPROCTOLÓGICA

WILSON JUNG

FOZ DO IGUAÇU
2012

Wilson Jung

Aplicação de Princípios de Qualidade de Dados durante o Desenvolvimento de um Sistema Computacional Médico para a Cirurgia Coloproctológica

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Sistemas Dinâmicos e Energéticos como parte dos requisitos para obtenção do título de Mestre em Engenharia de Sistemas Dinâmicos e Energéticos. Área de concentração: Sistemas Dinâmicos e Energéticos.

Orientadora: Prof.ª Dr.ª Huei Diana Lee
Co-orientador: Prof. Dr. Wu Feng Chung
Co-orientador: Prof. Dr. Cláudio Saddy Rodrigues Coy

Foz do Iguaçu
2012

Aplicação de Princípios de Qualidade de Dados durante o Desenvolvimento de um Sistema Computacional Médico para a Cirurgia Coloproctológica

Wilson Jung

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Engenharia de Sistemas Dinâmicos e Energéticos e aprovada pela Banca Examinadora:
Data da defesa pública: 25/04/2012.

Prof.ª Drª **Huei Diana Lee** - (Orientadora)
Universidade Estadual do Oeste do Paraná - UNIOESTE

Prof.ª Drª **Maria Carolina Monnard**
Universidade de São Paulo - USP

Prof. Dr. **Romeu Reginatto**
Universidade Estadual do Oeste do Paraná - UNIOESTE

Prof. Dr. **Eugênio de Bona Castelan Neto**
Universidade Federal de Santa Catarina - UFSC

Resumo

Atualmente, diversas áreas do conhecimento humano fazem uso de sistemas computacionais para auxiliar no gerenciamento de dados, que são a base para o processo de tomada de decisão. A Qualidade de Dados (QD) constitui uma característica fundamental cuja ausência pode comprometer a utilidade da informação e os processos que a utilizam. Na literatura são apresentados diversos casos que relatam o impacto de problemas de QD nas mais diversas áreas, representando perdas econômicas e sociais. Assim, a área de QD visa o estudo das causas de problemas nos dados e a proposição de métodos de avaliação e processos que auxiliem na garantia da qualidade. Na área da saúde os dados constituem elementos importantes que são utilizados como base para a aplicação de tratamentos e procedimentos médicos aos pacientes, fatores que exigem um nível elevado de qualidade. Esses dados também são utilizados em pesquisas e aplicações de métodos computacionais de extração de conhecimento, como a Mineração de Dados. Assim, o objetivo deste trabalho consiste em estudar a aplicação de princípios que auxiliem na garantia da QD durante o desenvolvimento de um sistema computacional médico. Tal objetivo motivou a realização de um estudo de caso relacionado à especialidade da Coloproctologia, no qual foi desenvolvido o protótipo de um sistema para gerenciamento de dados de cirurgia coloproctológica em parceria com o Serviço de Coloproctologia da FCM - UNICAMP. A interação com os especialistas de domínio constituiu um fator fundamental durante o processo de desenvolvimento, possibilitando a modelagem adequada da estrutura dos dados que forma o sistema. Também foi incorporado ao protótipo um módulo para monitoramento de problemas específicos nos dados, auxiliando tanto no preenchimento adequado da informação quanto no controle dos registros de pacientes que apresentam problemas de QD. Ao final, o protótipo foi submetido à avaliação por colaboradores da área da computação e da saúde, que após a utilização do sistema responderam a um formulário para avaliação qualitativa de QD. Os resultados da avaliação indicaram a adequação do protótipo para as atividades a que é destinado, orientaram para a revisão de funcionalidades específicas e poderão auxiliar na evolução do sistema proposto e em trabalhos futuros.

Palavras-chave: Qualidade de Dados, Informática Biomédica, Sistemas de Informação, Mineração de Dados, Dados Médicos.

Abstract

Lately, many human knowledge fields use computer systems to support data management which are the foundation to the decision making process. Data Quality (DQ) is a key feature whose absence can undermine the usefulness of the information and the processes that use it. There can be found in the literature several cases of DQ problems with impact in many areas, resulting in economic and social losses. Therefore, DQ research aims to study data problems' causes and proposes assessment methods and processes to assist in quality assurance. In healthcare, data constitutes an important element used as the basis for applying medical treatments and procedures to patients, thus requiring a high quality level. The data is also used in the research and application of computational knowledge discovery methods, such as Data Mining. Therefore, the goal of this work is to study the implementation of principles to assist DQ guarantee during the medical software development. This goal motivated the development of a case study related to Coloproctology, in which a surgery data management system prototype was developed in partnership with the Coloproctology Service of FCM - UNICAMP. The interaction with domain experts was a key factor during the development process, providing the adequate data structure modeling that composes the system. A module to monitor specific data problems has also been incorporated into the prototype to assist the appropriate information insertion as much as the control of patients' records which have DQ problems. The prototype has been evaluated by computer and healthcare's colaborators, who, after using the system, answered to a qualitative DQ assessment form. The assessment's results pointed out the prototype suitability to the activities it is aimed for, guided specific functionalities' review and may support the proposed software evolution and future related work.

Keywords: Data Quality, Biomedical Informatics, Information Systems, Data Mining, Medical Data.

À minha esposa,
Viviane.

Aos meus pais,
Lurdes e Wilson, Gessi e Enio.

Ao Laboratório de Bioinformática – LABI.

Agradecimentos

Chega um momento em que paramos para pensar em nossa caminhada e nas pessoas que fizeram parte dela e que de alguma forma, muitas vezes sem conhecimento do fato, nos ajudam a continuar nos momentos difíceis e tornam mais alegres as comemorações e as conquistas. É difícil expressar em palavras toda a minha gratidão e afeto, mas fica aqui um singelo reconhecimento e agradecimento pelas contribuições que recebi nesta fase de minha vida.

Agradeço em primeiro lugar a Deus e a sua Divina Providência pelas oportunidades colocadas em minha vida.

À minha amada esposa, Viviane Rech Jung, por suportar tantas ocasiões em que estive distante e pelo apoio incondicional, principalmente nos momentos de difícil decisão. Seu amor me fortalece e seu carinho me conforta. Te amo muito "pequena"!

Aos meus pais, Lurdes Maria Jung e Wilson Antonio Jung, pelo amor e sacrifício incondicional para oferecer aos filhos a melhor educação e a minha irmã Jéssica Aparecida Jung por todo amor e carinho.

À minha mais nova família, Enio Rech, Gessi Rech, William Rech e Márcio Catafesta, pelo apoio e compreensão dos momentos em que estive ausente. Agradeço a Deus ter colocado vocês em minha vida.

Aos meus amigos Claudemir Vieira e Rosemeri Vieria, por todos os sorrisos, apoio e por estarem presentes em momentos importantes de minha vida.

Aos professores Igor Vinicius Mussoi de Lima e Alessandra Bortoletto Garbelotti Hoffmann por acreditarem em mim e me recomendarem para este mestrado.

Aos professores Wu Feng Chung e Huei Diana Lee pelo carinho e pelos ensinamentos que levarei por toda a vida. Sinto-me afortunado pela oportunidade de aprender com grandes mestres como vocês. A universidade carece de mais educadores com tal honra e dedicação.

Aos professores Renato Bobsin Machado, André Gustavo Maletzke, Carlos Andres Ferrero e Willian Zalewski por todo conhecimento e reflexões compartilhadas e pelos momentos de entretenimento.

Ao meu amigo Jorge Aikes Junior, pela longa caminhada que trilhamos desde o período da faculdade. Obrigado por sua amizade, ajuda nos estudos e por manter alegres as incontáveis horas de estudo.

À Adrieli Cristina da Silva, Bianca Espindola, Luiz Henrique Dutra da Costa e Richardson Floriani Voltolini por toda a colaboração em diferentes etapas deste trabalho. A ajuda de vocês foi inestimável.

A todos que dispuseram de tempo para contribuir com a avaliação do protótipo desenvolvido no estudo de caso deste trabalho.

A todos os "LABIANOS" com quem convivi grande parte do tempo durante esses dois anos de mestrado.

Aos professores do PGESDE e a equipe da UNIOESTE.

À CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – e à FPTI – Fundação Parque Tecnológico Itaipu – pelo apoio por meio da linha de financiamento de bolsas prestadas ao PGESDE.

Aos membros da banca por contribuir com o enriquecimento deste trabalho.

A todos que direta ou indiretamente contribuíram com o desenvolvimento deste trabalho e em minha caminhada de mestrado.

“A cada dia que vivo, mais me convenço de que o desperdício da vida está no amor que não damos, nas forças que não usamos, na prudência egoísta que nada arrisca, e que, esquivando-se do sofrimento, perdemos também a felicidade.”

Carlos Drummond de Andrade

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

Ajax	*Asynchronous JavaScript and XML*
ANS	Agência Nacional de Saúde Suplementar
API	*Application Programming Interface*
CFM	Conselho Federal de Medicina
COOINFO/MS	Comitê de Informação e Informática em Saúde
CRUD	*Create, Read, Update, Delete*
CSS	*Cascading Style Sheets*
DINCON	Conferência Brasileira de Dinâmica, Controle e Aplicações
DRY	*Don't Repeat Yourself*
DSL	*Domain-Specific Language*
FAQP	Formulário para Avaliação Qualitativa do Protótipo
GED	Gerenciamento Eletrônico de Documentos
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IBM	*International Business Machines*
ICP-Brasil	Infraestrutura de Chaves Públicas Brasileiras
IDE	Integrated Development Environment
IETF	The Internet Engineering Task Force
IQA	*Information Quality Assessment*
JSON	*Javascript Object Notation*
MCSRES	Manual de Certificação para Sistemas de Registro Eletrônico em Saúde

MD	Mineração de Dados
MVCC	*Multi-Version Concurrency Control*
MVC	*Model View Controller*
NGS1	Nível de Garantia de Segurança 1
NGS2	Nível de Garantia de Segurança 2
NIST	*National Institute of Standards and Technology*
OMT	*Object Modeling Technique*
OOSE	*Object-Oriented Software Engineering*
ORM	*Object-Relational Mapping*
PGDCC	Protótipo para Gerenciamento de Dados de Cirurgia Coloproctológica
PL/pgSQL	Procedural Language/PostgreSQL Structured Query Language
PL/SQL	*Procedural Language/Structured Query Language*
QD	Qualidade de Dados
REST	*Representational state transfer*
S-RES	Sistema para Registro Eletrônico em Saúde
SBIS	Sociedade Brasileira de Informática em Saúde
SGBD	Sistema Gerenciador de Banco de Dados
SHA	*Secure Hash Algorithm*
SQL	*Structured Query Language*
SSL	*Secure Sockets Layer*
TCP	*Transmission Control Protocol*
TDQM	*MIT Total Data Quality Management*
TISS	Troca de Informação em Saúde Suplementar
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
UML	Unified Modeling Language

W3C	*World Wide Web Consortium*
XHR	*XMLHttpRequest*
XHTML	*Extensible Hypertext Markup Language*
XML	*Extensible Markup Language*

Capítulo 1

Introdução

A evolução de tecnologias ao longo dos anos tem estimulado grande parte das áreas do conhecimento humano a utilizar sistemas eletrônicos como ferramentas de auxílio ao desenvolvimento de suas atividades. Essas ferramentas incluem o uso de sistemas computacionais que destacam-se na importante tarefa de gerenciamento de dados.

Grande parte das operações presentes nesses sistemas realizam a geração, a aquisição ou a transformação da informação[1], de modo que seja possível utilizá-la como base para auxiliar em processos de tomada de decisão. Assim, a presença de problemas nos dados pode prejudicar as atividades que fazem uso deles, repercutindo desde reveses mínimos até desastres.

Como exemplo de graves consequências decorrentes de problemas de Qualidade de Dados (QD) podem ser citados os incidentes relacionados à explosão do ônibus espacial *Challenger* em 1986 e o ataque militar errôneo ao *Iranian Airbus* pelo *USS Vincennes* da marinha americana, em 1988 (Fisher, 2001). O incidente de 1986, envolvendo o ônibus espacial *Challenger*, ocorreu devido a falhas no provimento de informações adequadas pelo sistema de gestão do projeto que não permitiu aos responsáveis identificar problemas técnicos relacionados ao combustível do ônibus espacial, fator que possibilitaria a interrupção da missão. Essa falha resultou na explosão do ônibus espacial e na morte de sete pessoas que formavam a tripulação. Em relação ao desastre de 1988, o sistema de informação utilizado pela marinha no *USS Vinceness* apresentava problemas na representação da informação sobre aeronaves no radar, fazendo com que o comandante responsável identificasse erroneamente o *Iranian Airbus* como uma aeronave inimiga, o que resultou na morte de 290 civis inocentes.

Do mesmo modo, falhas e problemas relacionados a sistemas computacionais de apoio a saúde podem resultar em consequências graves. Na literatura são relatadas situações que prejudicaram a vida de muitas pessoas, como a overdose de radiação ocorrida em alguns tratamentos médicos que utilizaram as unidades de terapia Therac-25 no ano de 1986, devido a problemas no *software* do equipamento (Leveson and Turner, 1993). Nos anos de 2000 e 2001 outro caso de overdose de radiação envolvendo unidades de radioterapia Cobalto-60 prejudicou cerca de

[1]Neste trabalho os termos dado e informação são utilizados indistintamente.

28 pessoas no Panamá. O problema era decorrente da aceitação de dados em formato incorreto pelo sistema computacional, que também não exibia mensagens de erro (Borrás, 2006).

Além dos casos apresentados, os problemas de QD podem gerar perdas econômicas significativas. Um exemplo extremo de perdas financeiras devido a problemas nos dados pode ser representado pelo "*bug* do ano 2000", que afetou diversas organizações devido ao modo inadequado na representação de datas em bancos de dados. O custo de reparo deste problema foi estimado em 1.5 trilhão de dólares. Outros exemplos incluem problemas de QD em setores governamentais e empresas de grande porte cujas perdas chegam a milhões de dólares (English, 1999).

Assim, a área de QD visa tanto a avaliação e a correção de problemas existentes nos dados quanto a proposição de processos e métodos que auxiliem na garantia de qualidade. Nesse sentido, a proposta deste trabalho inclui estudar conceitos que possam auxiliar no desenvolvimento de um sistema computacional médico visando à qualidade da informação gerenciada, como mecanismos de validação para orientar a inserção correta dos dados pelos usuários, criação de uma interface gráfica organizada que ofereça uma experiência de utilização adequada, além do provimento de ferramentas de apoio para o monitoramento de QD.

Na literatura há trabalhos que incorporam funcionalidades a sistemas de informação para auxiliar na redução de problemas durante o processo de utilização. No trabalho de Fadel et al. (2006) foi proposto o desenvolvimento de um prontuário eletrônico para auxiliar no tratamento da Hanseníase, no qual foram desenvolvidas funcionalidades que oferecem ao usuário do sistema meios para a verificação de possíveis problemas nas informações. Entre os recursos implementados no sistema cita-se o controle de registros com preenchimento incompleto, o armazenamento de informações a respeito das operações realizadas por cada usuário do sistema, a indicação do tratamento de acordo com o diagnóstico cadastrado e o acompanhamento dos pacientes que estão com o tratamento em atraso. Além de melhorar a confiabilidade dos dados cadastrados, os princípios adotados orientam o correto atendimento e tratamento dos pacientes.

De modo semelhante, no trabalho de Herzberg et al. (2011) foi proposto o desenvolvimento de um mecanismo para auxiliar na redução de incompletudes nos dados clínicos de um hospital universitário. Esse mecanismo consistiu no desenvolvimento de rotinas automáticas que verificam informações incompletas e emitem comunicados por e-mail aos médicos responsáveis, relatando os registros que precisam ser finalizados. O comunicado era enviado até duas vezes para o médico responsável, de acordo com um intervalo de tempo pré-definido entre cada uma das notificações. Caso o registro incompleto não fosse corrigido após a segunda notificação, o comunicado era enviado a um supervisor. Esse procedimento aumentou consideravelmente o nível de completude dos registros clínicos do sistema estudado.

Outro exemplo de mecanismos auxiliares em sistemas de informação, visando a QD, está relacionado à área da Biologia, na qual o trabalho de Veiga et al. (2011) relata a aplicação de técnicas e recursos de maneira a auxiliar no correto preenchimento de uma base de dados de

ocorrências de espécies. Entre os recursos utilizados nesse trabalho destacam-se o auxílio ao preenchimento de dados taxonômicos de espécies, de acordo com a base de referência *Catalog of Life*, e a utilização de técnicas de georeferenciamento, disponibilizadas pela *Application Programming Interface* (*API*) do *Google Earth* para auxiliar o registro de dados geoespaciais.

Uma característica comum às pesquisas citadas e ao presente trabalho consiste no desenvolvimento de mecanismos e funcionalidades em sistemas de informação para auxiliar na correta inserção dos dados, visando à QD. Nessa proposta, o objetivo é destacar a importância de medidas de prevenção à ocorrência de problemas na informação e reduzir os custos decorrentes da adoção de medidas corretivas (Guerra-García et al., 2010).

Outra preocupação presente neste trabalho consiste na preparação de uma base adequada para futuras aplicações do processo de Mineração de Dados (MD), que pode ser afetado por fatores relacionados à baixa QD, como apresentado no trabalho de Blake and Mangiameli (2011) que verificou a influência negativa de problemas de QD específicos no processo de classificação. Não só a QD pode contribuir com a MD, como o processo inverso também é possível, sendo encontrados trabalhos na literatura que utilizam técnicas de MD como regras de associação (Hassine et al., 2008; Tremblay et al., 2010), algoritmos genéticos (Das and Saha, 2009) e redes neurais artificiais (Omara et al., 2011) para auxiliar na redução de problemas nos dados.

Ainda no presente trabalho também se faz evidente o conceito de interdisciplinaridade pelo qual o conhecimento da área da computação e da saúde converge para contribuir com a solução de um problema comum. A união de diferentes áreas para a realização de pesquisas tem se tornado um tema, cada vez mais, em discussão na comunidade acadêmica. A interdisciplinaridade, atualmente também denominada de convergência tecnológica, tem motivado pesquisadores a unir as especialidades para a realização de trabalhos a partir de problemas que precisam ser resolvidos, compartilhando as diferentes capacidades dos profissionais de cada área (Fioravanti, 2012).

A Figura 1.1 representa, esquematicamente, a localização do presente trabalho em relação ao projeto de Análise Inteligente de Dados. Neste trabalho buscam-se meios para prover dados com qualidade tanto para auxiliar na tomada de decisão por profissionais de diferentes áreas quanto para contribuir com o sucesso na aplicação de métodos computacionais, como a MD.

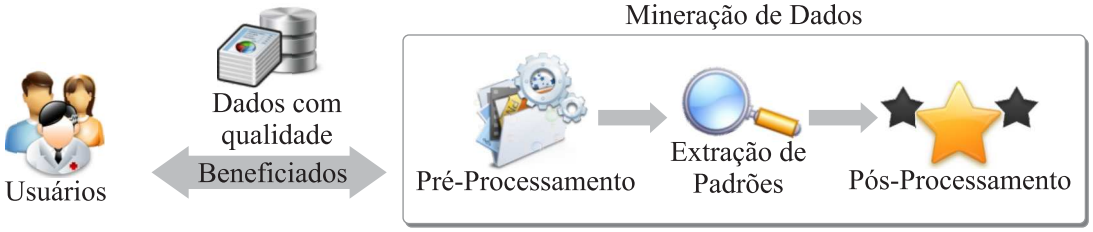

Figura 1.1: Representação esquemática da interação deste trabalho com o projeto de Análise Inteligente de Dados.

O projeto de Análise Inteligente de Dados, do qual este trabalho faz parte, é desenvolvido

em uma parceria entre o Laboratório de Bioinformática (LABI) da Universidade Estadual do Oeste do Paraná (UNIOESTE), Foz do Iguaçu, o Serviço de Coloproctologia da Faculdade de Ciências Médicas (FCM) da Universidade Estadual de Campinas (UNICAMP), o Laboratório de Inteligência Computacional (LABIC) da Universidade de São Paulo (USP), São Carlos e o Grupo Interdisciplinar em Mineração de Dados e Aplicações (GIMDA) da Universidade Federal do ABC (UFABC), Santo André.

1.1 Objetivos

O objetivo geral deste trabalho consiste em realizar o estudo da aplicação de medidas que auxiliem na garantia da QD durante o desenvolvimento de um sistema computacional médico.

Para alcançar esse objetivo geral, neste trabalho os objetivos específicos são:

- Realizar levantamento bibliográfico sobre os conceitos de QD discutidos na literatura, bem como métodos de avaliação e medidas para auxiliar na garantia da QD;
- Desenvolver um sistema computacional de informação para a área da saúde, envolvendo a especialidade da Coloproctologia, aplicando conceitos de QD;
- Desenvolver um módulo de monitoramento, baseado nos conceitos de QD, para auxiliar no controle de registros com problemas específicos;
- Realizar a avaliação qualitativa do protótipo desenvolvido de modo a analisar a visão dos usuários em relação à qualidade da informação gerenciada e aspectos de usabilidade do sistema.

1.2 Organização deste Trabalho

Este trabalho está organizado da seguinte maneira:

Capítulo 2 — Qualidade de Dados: neste capítulo são discutidos conceitos sobre o tema de QD, abordando os problemas comuns relacionados a essa área e suas causas. Também são apresentadas algumas abordagens descritas na literatura que auxiliam no processo de avaliação da qualidade. Ao final são discutidos conceitos sobre o processo de MD que, assim como outros processos e atividades, pode sofrer influência do nível de qualidade dos dados disponibilizados.

Capítulo 3 — Desenvolvimento de Sistemas de *Software*: neste capítulo são apresentados conceitos de engenharia de *software* que direcionam o correto processo de desenvolvimento de sistemas computacionais.

Capítulo 4 — Ferramentas para Desenvolvimento de Sistemas de *Software*: neste capítulo são apresentadas algumas ferramentas utilizadas para o desenvolvimento de sistemas de *software*, como a linguagem de programação *Ruby*, o *framework* para desenvolvimento *Web Ruby on Rails* e o Sistema Gerenciador de Banco de Dados *PostgreSQL*.

Capítulo 5 — Sistemas de Informação para Saúde: neste capítulo são apresentados conceitos sobre dados médicos e suas características, bem como a legislação que regula o armazenamento dos registros de saúde dos pacientes. Ao final são discutidas questões referentes ao processo de certificação para sistemas de informação destinados à saúde.

Capítulo 6 — Estudo de Caso: neste capítulo é apresentado o estudo de caso realizado neste trabalho, no qual foi desenvolvido o protótipo de um sistema para gerenciamento de dados de cirurgia coloproctológica. O desenvolvimento desse protótipo partiu da análise de um sistema legado utilizado pelo Serviço de Coloproctologia da FCM – UNICAMP para o registro de informações sobre as cirurgias realizadas neste serviço. Durante o processo de desenvolvimento foram aplicadas medidas visando à QD que resultaram no desenvolvimento de um processo para auxiliar no monitoramento de problemas.

Capítulo 7 — Resultados e Discussão: neste capítulo são apresentados e discutidos os resultados obtidos a partir da avaliação do protótipo desenvolvido no estudo de caso.

Capítulo 8 — Conclusão: neste capítulo são apresentadas as conclusões deste trabalho, suas principais contribuições, algumas limitações encontradas durante seu desenvolvimento e as questões que podem ser abordadas em trabalhos futuros, dando continuidade a este trabalho.

Capítulo 2

Qualidade de Dados

2.1 Considerações Iniciais

Com o aumento da utilização dos sistemas de informação para gerenciamento das mais diversas atividades, bem como a adoção do meio digital para armazenamento de dados, simultaneamente, a preocupação com a QD tem crescido. Devido à velocidade em que os dados são manipulados nos meios digitais, um problema de QD pode propagar-se rapidamente, podendo até mesmo comprometer a utilidade da base de dados em contexto. Em contrapartida, a potencialidade dos recursos computacionais também pode ser utilizada para executar mecanismos que auxiliem na verificação e na avaliação da QD.

A área de QD é muito ampla, estando ligada a diferentes contextos que envolvem a informação[1] e seu uso. A pesquisa no tema de QD pode ser dividida considerando diferentes aspectos (Ge and Helfert, 2007):

Avaliação da QD: processo no qual são quantificados aspectos específicos de qualidade de um conjunto de dados;

Gerenciamento da QD: processo com o objetivo de melhorar a utilidade e a validade dos dados, não somente focado no presente, mas em um processo contínuo;

Contextualização da QD: processo no qual é investigada a influência da QD no contexto da instituição em questão.

O objeto de estudo deste trabalho faz uso de conceitos relacionados aos aspectos de avaliação e gerenciamento da QD. Neste capítulo são apresentados conceitos que direcionam a reflexão sobre os problemas de QD e suas causas, bem como métodos de avaliação propostos para identificar a origem dos problemas. Os benefícios repercutem em diversas áreas, por exemplo a Mineração de Dados que é fortemente influenciada pelo nível de qualidade da informação utilizada durante seu processo.

[1] Neste trabalho os termos dado e informação são utilizados indistintamente.

2.2 Problemas de Qualidade de Dados e Suas Causas

Problemas de QD afetam diversas organizações e podem gerar perdas econômicas e sociais (Wang, 1996). Esses problemas não afetam somente o setor privado, mas também órgãos governamentais (Strong et al., 1997) e repercutem desde reveses mínimos até desastres, como os incidentes relacionados ao ônibus espacial *Challenger* em 1986 e ao *Iranian Airbus* abatido erroneamente em 1988 (Fisher, 2001), como mencionado no Capítulo 1.

Os problemas de QD podem ser considerados, basicamente, em relação aos dados e ao usuário. Sob a perspectiva de dados, entende-se que a informação deve estar de acordo com suas especificações e seus requisitos. Já sob a perspectiva de usuário, o foco está na adequação da informação às necessidades do usuário (Wang, 1998; Eppler, 2006). Por meio dessas características, o conceito de QD pode ser definido como a busca pela informação livre de defeitos e que apresenta características desejadas pelos usuários.

Assim, os problemas de QD apresentados na literatura, considerando essas duas perspectivas e a dependência do contexto, podem ser classificados em quatro categorias (Ge and Helfert, 2007):

Perspectiva de Dados / Independente de Contexto: são problemas de qualidade presentes em bases de dados e referem-se a quaisquer conjuntos de dados;

Perspectiva de Dados / Dependente de Contexto: são problemas de qualidade que violam as especificações de negócio e podem ser detectados por regras contextuais;

Perspectiva de Usuário / Independente de Contexto: são problemas de qualidade que ocorrem no processamento da informação;

Perspectiva de Usuário / Dependente de Contexto: são problemas de qualidade que indicam que os dados não estão adequados para atender às necessidades dos usuários.

Exemplos de problemas relatados na literatura, de acordo com as categorias apresentadas, são apresentados na Tabela 2.1.

A causa dos problemas de *QD* pode ter origem a partir da influência de diversos processos (Figura 2.1), os quais podem ser agrupados, por afinidade, em três categorias (Maydanchik, 2007):

Processos que trazem dados de fontes externas: um dos problemas em dados provenientes de fontes externas ocorre em projetos que envolvem a migração de bases de dados legadas, sendo que há casos em que é necessário realizar a **conversão de dados** para os padrões adotados no novo sistema. Porém, as informações sobre a lógica envolvida na criação dos dados antigos nem sempre são documentadas e quando documentadas nem

Tabela 2.1: Classificação de problemas de QD (Ge and Helfert, 2007).

Categoria	Perspectiva de Dados	Perspectiva de Usuário
Independente de Contexto	- Erros de ortografia - Dados faltantes - Dados duplicados - Valores incorretos - Formato inconsistente de dados - Dados desatualizados - Formato de dados incompletos - Violação de sintaxe - Violação de valor único - Violação de restrições de integridade - Formatação de texto	- A informação é inacessível - A informação é insegura - Dificuldade em recuperar a informação - Dificuldade em agregar a informação - Erros na transformação da informação
Dependente de Contexo	- Violação de restrição de domínio - Violação das regras de negócio da empresa - Violação dos regulamentos da empresa e do governo - Violação das restrições fornecidas pelo administrador de banco de dados	- A informação é formada por significados inconsistentes - A informação é incompleta - A informação é representada de modo compacto - Dificuldade em manipular a informação - Dificuldade em entender a informação

sempre estão atualizadas, repercutindo no processo de migração. O nível de complexidade fica ainda maior em projetos que envolvem a fusão de diferentes bases de dados oriundas da **consolidação de sistemas**, por exemplo, devido à união de duas empresas, pois são projetos que devem ser executados com o menor tempo possível, em meio ao impacto da fusão de equipes de TI e possível mudança no quadro de profissionais. Porém uma das causas mais comuns de problemas de QD está relacionada a erros durante a **entrada manual de dados**. Esse tipo de problema é influenciado por interfaces gráficas mal elaboradas que resultam em uma experiência frustrante do usuário que muitas vezes deixa campos de dados sem preenchimento e quando estes são obrigatórios preenche-os com quaisquer valores. Problemas dessa categoria podem ser amenizados por formulários organizados, claros, informativos e pela conscientização de seus usuários. Outra fonte de problemas está relacionada à **alimentação de dados em lotes** que é um recurso utilizado, por exemplo, para a comunicação entre diferentes sistemas. Assim, caso um lote de dados apresente erros, seu impacto pode ser desastroso sobre sistemas que gerem mais dados a partir do processamento dessa informação. **Interfaces em tempo real** também representam uma categoria na qual os problemas são dificilmente monitoráveis, devido à velocidade e a granularidade da informação que é dividida em pacotes de dados, repercutindo no fato de que a qualidade pode ser comprometida pela velocidade em que a informação é obtida;

Processos que transformam a informação interna: o **processamento de dados** é uma das tarefas essenciais do *software*. Mesmo pequenas mudanças na codificação do *software*, quando não devidamente testadas, podem gerar problemas nos dados por ele processados, propagando erros rapidamente. Outra preocupação dessa atividade está rela-

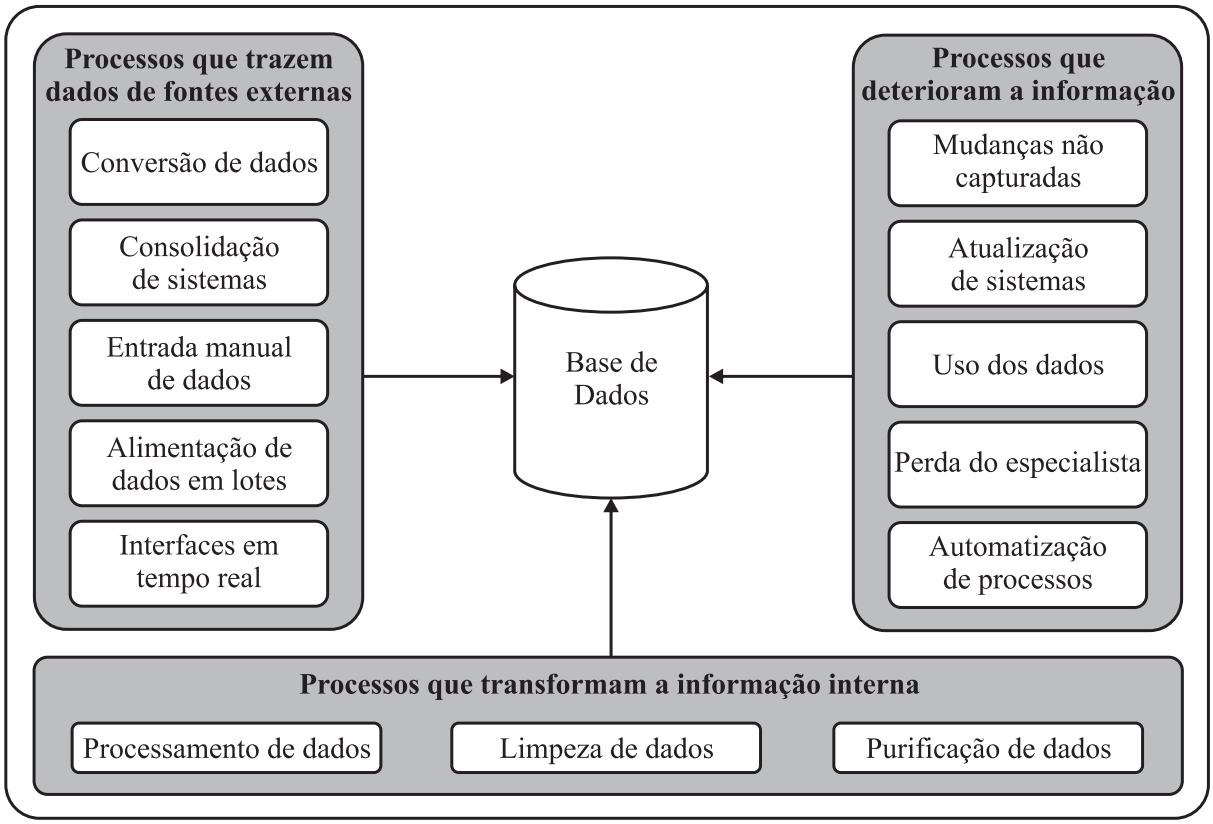

Figura 2.1: Processos que influenciam a QD (modificado de Maydanchik (2007)).

cionada ao momento em que a informação deve ser processada, podendo ser necessário que os dados estejam em um estado adequado. Já o processo de **limpeza de dados** visa à correção de problemas e muitos métodos utilizam algoritmos para automatizá-lo. Porém, este procedimento apresenta riscos, pois os problemas geralmente são complexos e estão interrelacionados, e quando não são bem entendidos a correção de um problema pode gerar muitos outros. De modo semelhante, o processo de **purificação de dados** busca remover dados antigos que não são mais necessários. Este processo pode ser prejudicial à QD caso os critérios de seleção incluam erroneamente dados relevantes ou excluam erroneamente dados desnecessários;

Processos que deterioram a informação: por vezes os objetos do mundo real podem ter seu estado alterado, não refletindo as mudanças nos dados presentes em sistemas de informação. Estas **mudanças não capturadas** refletem em dados imprecisos e até mesmo errôneos. A **atualização de sistemas** também pode introduzir problemas de QD em casos em que a estrutura de dados sofre alterações, modificando de alguma forma o significado de dados previamente cadastrados. Outro aspecto relaciona-se à mudança dos propósitos de **uso dos dados**, pois mesmo que não ocorra mudança na informação, o nível de qualidade pode ser deteriorado, já que informações destinadas a um propósito nem sempre são adequadas a outros propósitos. Outro fator que impacta sobre a QD, porém não é um elemento do sistema de informação, é o especialista que trabalha diretamente com os da-

dos. A **perda do especialista** muitas vezes significa a perda da única fonte de informação sobre características particulares dos dados, já que muitas vezes os dicionários de dados não são mantidos atualizados. Por fim, a **automatização de processos** não pode substituir a capacidade do ser humano em validar os dados com todas as suas peculiaridades antes de utilizá-los.

Como mencionado, um dos problemas mais comuns para o surgimento de dados com baixa qualidade está relacionado ao preenchimento manual no sistema de dados provenientes de, por exemplo, formulários e prontuários (Olson, 2003; Maydanchik, 2007; Zalewski et al., 2008). Um dos procedimentos que pode auxiliar na melhoria da qualidade deste processo é a determinação de supervisores para controle e avaliação do trabalho realizado (Chapman, 2005). Ainda, é importante que os usuários do sistema sejam conscientizados e entendam a responsabilidade da inserção correta dos dados no sistema. A interface gráfica do sistema deve ser organizada, intuitiva e conter técnicas de validação de formulários, fatores que auxiliam o preenchimento correto dos dados por parte dos usuários.

Pelo entendimento dos problemas de QD e suas causas é possível estudar meios para avaliar os dados e identificar as fontes e as características necessárias para que a qualidade seja garantida, sendo este tema discutido na sequência.

2.3 Avaliação de Qualidade de Dados

A avaliação de QD busca meios para mensurar o nível de QD indicando características da informação que possam apresentar deficiências. Como mencionado, os problemas de QD podem ser vistos sob a perspectiva de dados e de usuários. Para avaliar essas duas perspectivas são propostos na literatura métodos de avaliação subjetiva[2], que são utilizados para investigar a visão do usuário em relação a seus dados, e métodos de avaliação objetiva que são aplicados diretamente sobre os dados (Lee et al., 2002; Pipino et al., 2002).

Para realizar a avaliação foram propostas diversas dimensões de QD que representam aspectos específicos de QD (Batini and Scannapieca, 2006), como completude e consistência. Essas diversas dimensões, as quais podem ser de natureza multivariada e dependentes ou não de contexto, podem estar conectadas a outras dimensões e a um conjunto de métricas, como representado pela Figura 2.2. De modo a executar a avaliação de QD, cabe a cada instituição definir as dimensões importantes ao seu contexto, as variáveis que serão avaliadas e as métricas que serão utilizadas para mensurar cada variável (Lee et al., 2006).

As dimensões propostas na literatura podem ser agrupadas em diferentes abordagens conhecidas como intuitiva, teórica e empírica. A abordagem intuitiva define as dimensões a partir

[2]Neste trabalho os termos subjetiva e qualitativa são utilizados indistintamente.

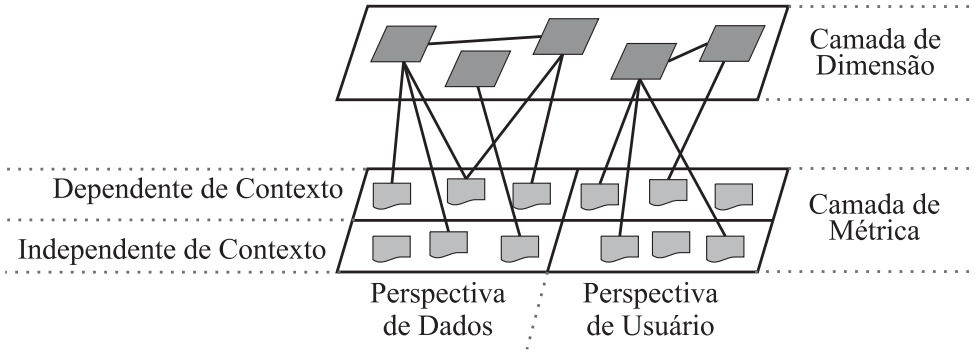

Figura 2.2: Relação entre os componentes da avaliação da QD: dimensões, métricas, perspectiva e contexto (modificado de Ge and Helfert (2007)).

da perspectiva de dados, de acordo com o senso comum e as experiências profissionais. A abordagem teórica, por sua vez, define as dimensões a partir da perspectiva do mundo real, com base no processo de manufatura de dados. Já a abordagem empírica define as dimensões a partir da perspectiva do usuário, baseando-se em entrevistas, questionários e na experiência dos usuários (Batini and Scannapieca, 2006; Ge and Helfert, 2007).

A abordagem teórica foi definida por Wand and Wang (1996) que considera os dados do sistema de informação como uma representação dos fatos do mundo real. Com base nessa definição são determinadas as dimensões que identificam deficiências nos dados. Já a abordagem empírica foi proposta por Wang (1996) cujo trabalho iniciou com a análise de 179 dimensões e concluiu com 15 dimensões agrupadas em quatro categorias:

QD intrínseca: que representa a qualidade que os dados possuem por si próprios;

QD contextual: que leva em consideração o contexto em que os dados estão inseridos;

QD representacional: que engloba aspectos relacionados a representação dos dados;

QD alcançável: que leva em consideração a acessibilidade dos dados, incluindo aspectos de segurança.

O autor defende a premissa de que para melhorar a QD é necessário entender o que isso significa para o usuário. Por fim, Redman (1997) utiliza a abordagem intuitiva classificando 12 dimensões em três categorias denominadas de esquema conceitual, valores de dados e formato de dados.

Dentro de cada abordagem de QD podem haver dimensões comuns, porém com definições diferentes. A discussão entre as diferenças das dimensões de cada abordagem pode ser encontrada em Batini and Scannapieca (2006). Neste trabalho adota-se a abordagem de Wang (1996), pela qual é possível analisar e relacionar os níveis de QD, de acordo com as perspectivas de dados e de usuário. Na Tabela 2.2 são apresentadas as definições de cada dimensão de QD.

Tabela 2.2: Definição de dimensões de QD propostas na literatura (Wand and Wang, 1996; Wang, 1996; Lee et al., 2006).

Dimensão	Definição
Acessibilidade (*Accessibility*)	grau de disponibilidade da informação ou grau de facilidade e rapidez para obtê-la
Quantidade Adequada (*Appropriate Amount*)	quão apropriado é o volume de informação para a tarefa em contexto
Credibilidade (*Believability*)	grau de veracidade e credibilidade da informação
Completude (*Completeness*)	grau em que a informação não está ausente e é de amplitude e profundidade suficiente para a tarefa em contexto
Consistência (*Consistency*)	pode ser observada sob diferentes perspectivas, por exemplo dados redundantes em uma ou mais tabelas ou elementos de dados relacionados, tal como endereço e código postal
Representação Concisa (*Concise Representation*)	grau em que a informação é representada de modo compacto
Representação Consistente (*Consistent Representation*)	grau em que a informação é apresentada no mesmo formato
Facilidade de Operação (*Ease of Operation*)	quão facilmente os dados são gerenciados e manipulados
Livre de Erros (*Free of Error*)	quão corretos e confiáveis são os dados
Interpretabilidade (*Interpretability*)	grau em que os dados estão em uma linguagem apropriada e as unidades e definições dos dados são claras
Objetividade (*Objectivity*)	quão imparciais e não-tendenciosos (não-prejudiciais) são os dados
Relevância (*Relevancy*)	grau de aplicabilidade e auxílio dos dados para a tarefa em contexto
Reputação (*Reputation*)	quão fortemente os dados são considerados em relação à sua origem ou conteúdo
Segurança (*Security*)	quão apropriadamente restrito é o acesso aos dados para manter sua segurança
Pontualidade (*Timeliness*)	grau de atualidade dos dados e se esse grau é adequado a tarefa em que será utilizado
Entendimento (*Understandability*)	quão facilmente compreensível é a informação
Valor adicionado (*Value-Added*)	quão benéficos e vantajosos são os dados em relação a sua utilização

Como mencionado na Figura 2.2 é possível observar que cada dimensão está relacionada a uma ou mais métricas, que representam diferentes problemas de QD (Ge and Helfert, 2007). Na literatura são apresentados alguns exemplos de métricas comumente utilizadas e os cálculos que podem ser executados para que as medidas sejam obtidas. Como exemplo citam-se (Lee et al., 2006):

Livre de erros: indica um nível que reflete se os dados estão corretos, de acordo com as definições que determinam quando um dado é considerado errôneo:

$$Livre\ de\ Erros = 1 - \left(\frac{QE}{QT}\right) \tag{2.1}$$

onde QE representa a quantidade de dados com erros e QT representa a quantidade total de dados;

Completude: indica o nível de completude dos dados, que pode ser visto por diferentes perspectivas, como completude de linha, completude de coluna ou completude de tabela:

$$Completude = 1 - \left(\frac{QI}{QT}\right) \tag{2.2}$$

onde *QI* representa a quantidade de itens incompletos e *QT* representa a quantidade total de itens;

Consistência: indica o nível de consistência dos dados, que pode ser visto por diferentes perspectivas, como dados redundantes em uma ou mais tabelas e inconsistências entre atributos relacionados:

$$Consistência = 1 - \left(\frac{QC}{QT}\right) \tag{2.3}$$

onde *QC* representa a quantidade de instâncias violando um tipo específico de consistência e *QT* representa a quantidade total de verificações de consistência realizadas.

Cada métrica requer parâmetros previamente especificados, por exemplo, os parâmetros que identificam o que é um dado errôneo, se a completude dos dados é avaliada em nível de coluna ou de registro e quais os tipos de inconsistência que serão avaliados. Quando determinada dimensão faz uso de diferentes métricas torna-se necessário aplicar métodos que calculem um valor final que caracterize a dimensão. Caso as métricas possuam diferentes importâncias, o método indicado é o da média ponderada. Em situações mais conservadoras é indicada a aplicação do operador *min*, caso contrário o operador *max* pode ser utilizado (Pipino et al., 2002).

Para realizar a avaliação qualitativa de QD, ou seja, sob a perspectiva dos usuários, é proposto na literatura um método que inclui a aplicação de entrevista a diferentes níveis de usuários de um sistema de informação. A entrevista é realizada por meio de questionário específico que contém questões destinadas a capturar aspectos de diferentes dimensões de QD.

O questionário conhecido como *Information Quality Assessment* (*IQA*) é um instrumento de pesquisa adotado pelo método *AIMQ* de avaliação de QD (Lee et al., 2002). A partir de dimensões de QD presentes na literatura foram selecionadas pelos autores desse método, empiricamente, dimensões consideradas importantes para os utilizadores dos dados. O desenvolvimento do questionário teve início com a definição de 12 a 20 questões para cada dimensão, que foram apresentadas e discutidas com outros pesquisadores e com utilizadores de dados. Após sucessivas revisões foram realizadas modificações no questionário que resultaram em um conjunto de oito questões por dimensão. Em seguida o questionário foi utilizado em um estudo piloto envolvendo 52 entrevistados de diferentes funções, provenientes de seis companhias, fato que auxiliou a revisar e reduzir o número de questões por dimensão, variando entre quatro a cinco questões.

No total, o *IQA* é composto por 65 questões representando 15 dimensões de QD, cujas estatísticas de confiabilidade foram avaliadas pelo índice *Cronbach alpha* (Lehman, 2005) a partir de 261 entrevistados em cinco organizações. Esse questionário de avaliação qualitativa apresenta como opção de resposta para cada questão valores no intervalo entre zero, que significa de modo nenhum, e dez, que significa completamente. Nos casos apresentados na literatura foram considerados três categorias de usuário: os coletores de dados, os mantenedores e os utilizadores de dados.

A partir do resultado das avaliações objetivas e subjetivas, torna-se possível adotar medidas que visam à melhoria da QD em ambos os contextos, como apresentado na Figura 2.3. De posse dos resultados das avaliações objetivas e subjetivas, ambos são comparados em busca de discrepâncias e suas causas. Após, ações devem ser tomadas para a correção e a melhoria da QD em um processo contínuo, até que o nível desejado de qualidade seja atingido (Lee et al., 2006).

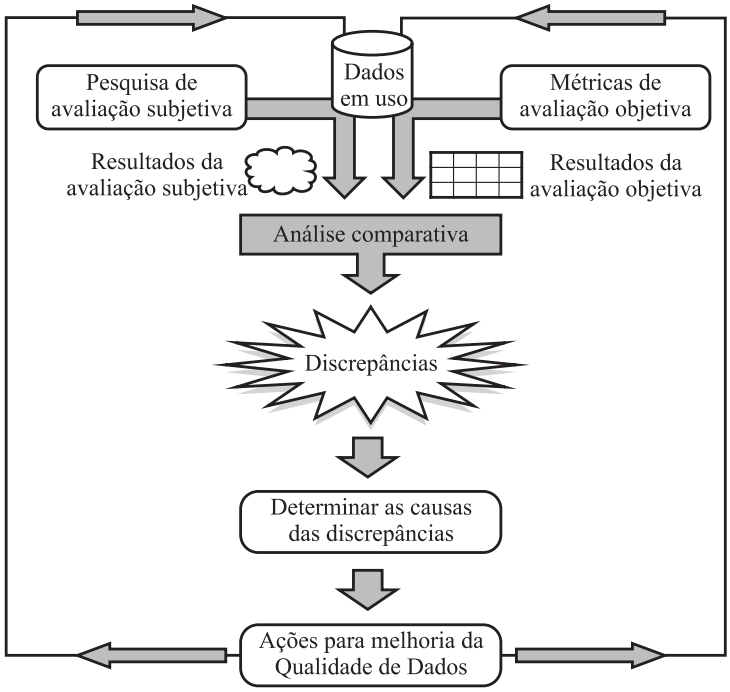

Figura 2.3: Processo de avaliação de QD (modificado de Lee et al. (2002)).

O processo de correção de dados errôneos é trabalhoso e demanda muita atenção. Realizar a correção de maneira manual é inviável, pois apresenta alto custo de tempo e exige, durante um longo período, profissionais especializados. No entanto, existem situações em que os erros são sistemáticos e seguem um padrão. Após analisado e descoberto esse padrão, é possível aplicar algoritmos para a correção em massa sobre alguns desses casos. Porém, deve-se estar atento às especificações do sistema que podem apresentar interdependências e se violadas, a tentativa de correção de um problema pode gerar muitos outros erros (Maydanchik, 2007).

A ausência de problemas nos dados, além de contribuir com as atividades da qual fazem parte, favorece também processos que auxiliem na análise da informação em busca de padrões e conhecimento embutido nos dados, como o processo de MD, apresentado na seção seguinte.

2.4 Mineração de Dados

A evolução de tecnologias e métodos computacionais em diversas áreas tem permitido a geração e o armazenamento de uma grande quantidade de dados, o que torna inviável a análise mais completa desses dados por meio de processamento manual. O processo de MD (Witten and

Frank, 2005; Han and Kamber, 2006) tem como objetivo a descoberta de padrões interessantes a partir de dados disponíveis, sendo uma ferramenta de grande importância para aplicações como controle de produção (Kwak and Yih, 2004), detecção de fraudes (Chan et al., 1999) e análise de dados biomédicos (Steiner et al., 2006; Gopalakrishnan et al., 2010).

As etapas envolvidas no processo de MD são ilustradas na Figura 2.4. Basicamente, consistem em três principais etapas: pré-processamento, extração de padrões e pós-processamento, que são descritas a seguir.

2.4.1 Pré-Processamento

Durante a etapa de pré-processamento é realizada a preparação dos dados para a fase de extração de padrões. Esta etapa é de grande importância para todo o processo de MD e detém em torno de 80% de todo seu tempo (Pyle, 1999).

Muitas bases de dados apresentam sérios problemas que podem prejudicar todo o processo de MD se não forem adequadamente tratados. Entre esses problemas destacam-se a presença de ruídos, dados incompletos, valores faltantes e *outliers*. Pode-se também encontrar problemas na lógica utilizada para a criação da base de dados, tal como a presença de dados redundantes. Para tratar esses problemas e fornecer dados consistentes aos mecanismos de extração de padrões, a etapa de pré-processamento comporta, entre outras, as seguintes tarefas (Larose, 2005; Lee, 2005; Han and Kamber, 2006):

Limpeza de dados: realiza-se a correção das inconsistências dos dados e remoção de ruídos;

Integração de dados: realiza-se a integração de dados distribuídos em diversas fontes;

Transformação de dados: aplicam-se operações de transformação, como métodos de normalização e agregação, visando a adequação dos dados para a extração de padrões;

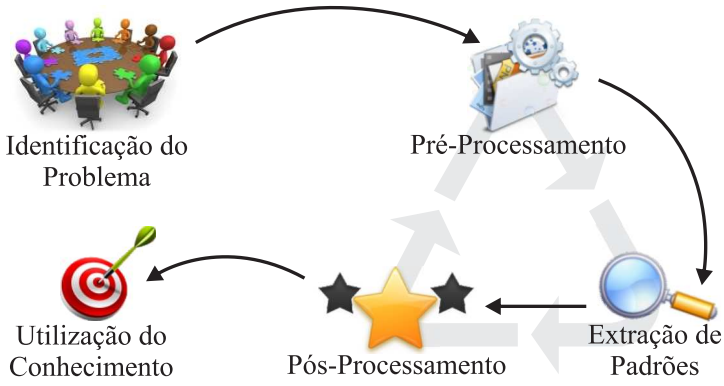

Figura 2.4: Etapas envolvidas no processo de MD (modificado de Voltolini (2006)).

Redução de dados: realiza-se a redução do número de atributos ou exemplos de modo que os resultados finais não sejam afetados. Entre as estratégias utilizadas nessa etapa podem ser citadas a agregação de dados, redução de dimensionalidade, seleção de subconjunto de atributos e eliminação de atributos redundantes;

Seleção de atributos: realiza-se a seleção dos atributos que melhor representam o conjunto geral de dados, reduzindo o tempo e o custo computacional necessário para obtenção do modelo. Entre algumas contribuições da seleção de atributos pode-se citar a ordenação dos atributos segundo critérios de importância e a remoção de ruídos nos dados, além de contribuir com a qualidade dos dados e influenciar na compreensão dos modelos obtidos durante a etapa de extração de padrões.

É interessante ressaltar que, embora as tarefas acima tenham sido descritas individualmente, elas são aplicadas de maneira integrada umas às outras.

Assim, o objetivo da etapa de pré-processamento apresenta intersecção com os objetivos da área de QD. A garantia de QD pode reduzir significativamente o tempo despendido na etapa de pré-processamento e contribuir com a precisão do modelo obtido na etapa de extração de padrões. Diversos estudos associam QD à MD de modo a analisar a influência de baixa qualidade sobre o processo de MD e também utilizar técnicas provenientes dessa área para auxiliar na busca por qualidade (Hassine et al., 2008; McClanahan, 2008; Blake and Mangiameli, 2011).

2.4.2 Extração de Padrões

De posse dos dados pré-processados, na etapa de extração de padrões podem ser utilizados algoritmos de aprendizado de máquina para a análise dos dados e a descoberta de padrões.

Nesta etapa a escolha dos algoritmos a serem aplicados deve ser realizada de acordo com a característica do problema em questão e os objetivos pretendidos. Entre esses métodos pode-se citar árvores de decisão, métodos estatísticos, redes neurais e algoritmos genéticos, que são amplamente conhecidos na literatura (Mitchell, 1997).

As tarefas realizadas na etapa de extração de padrões podem ser classificadas de acordo com as suas características (Fayyad et al., 1996; Larose, 2005):

Predição: trata-se da estimativa de um possível valor como base no conjunto histórico dos dados analisados. Quando os dados envolvidos na tarefa possuem informações sobre sua classe ou rótulo e os dados são discretos, o procedimento recebe o nome de classificação. Quando a classe é composta por dados contínuos, o procedimento realizado é o de regressão;

Associação: consiste na análise para tentar descobrir possíveis associações ou conexões entre objetos;

Agrupamento: é executado quando não se tem informações sobre a classe associada aos dados, sendo o seu objetivo realizar o agrupamento dos casos, por meio da identificação de grupos (*clusters*), maximizando as similaridades e minimizando as diferenças segundo algum critério definido;

Evolução: consiste na avaliação ou detecção do comportamento ou características de determinados objetos ao longo do tempo;

Desvio: consiste na descoberta e na avaliação de dados que apresentem comportamento distinto ao longo do tempo em relação ao conjunto geral de dados.

A execução da tarefa apropriada resulta em um modelo que deve ser testado e posteriormente analisado durante a fase de pós-processamento.

2.4.3 Pós-Processamento

A etapa de pós-processamento consiste na fase em que ocorre a visibilização, a interpretação e o entendimento dos padrões encontrados. Caso necessário, pode-se realizar novas interações com as etapas anteriores. Após consolidado, o conhecimento obtido deve ser documentado e, se necessário, incorporado a outros sistemas para futuras iterações (Fayyad et al., 1996).

Em geral, os métodos aplicados nesta etapa dividem-se em duas categorias (Carvalho et al., 1999):

Método subjetivo: também conhecido como *user-driven*, requer que o usuário estabeleça o conhecimento para que o sistema possa minerar o conjunto obtido na etapa de extração de padrões;

Método objetivo: também conhecido como *data-driven*, no qual não é necessário um conhecimento prévio para minerar o conjunto obtido na etapa de extração de padrões.

As duas abordagens podem ser utilizadas para selecionar padrões interessantes. Em um primeiro momento, a abordagem objetiva pode auxiliar para que sejam selecionados os padrões potencialmente interessantes e em um segundo momento a abordagem subjetiva para uma filtragem final, selecionando os padrões mais importantes (Freitas, 1999).

2.5 Considerações Finais

A QD é uma característica fundamental e desejável em todas as áreas que fazem uso de sistemas de informação para gerenciamento de dados. O impacto causado por problemas nos

dados pode ser muito prejudicial, repercutindo em custos elevados para execução do processo de correção.

Para investigar possíveis problemas nos dados são propostos diversos métodos de avaliação que por meio de características específicas dos dados e pela visão dos usuários em relação a eles, possibilita não só a identificação dos problemas, mas também a origem deles, resultando na correção de processos ou falhas de *software* responsáveis pela degradação da informação.

Essas medidas contribuem tanto diretamente, com o contexto envolvido, quanto indiretamente, com processos externos que são alimentados pelos dados, como é o caso da MD cujos modelos resultantes são influenciados pelo nível de qualidade.

No Capítulo 3 serão apresentados métodos de engenharia de *software* que visam orientar a construção de sistemas de informação de qualidade, preocupando-se com todos os aspectos envolvidos no desenvolvimento do produto, entre os quais o gerenciamento de dados constitui o principal motivo de sua construção.

Capítulo 3

Desenvolvimento de Sistemas de *Software*

3.1 Considerações Iniciais

O conceito de *software* é amplo, podendo ser sintetizado como um conjunto de instruções computacionais e estruturas de dados que possibilitam a manipulação de informações, caracterizando-se mais como um sistema lógico do que físico. Ao contrário de produtos de gêneros variados que são criados a partir do um processo de fabricação, um *software* é desenvolvido apoiado por conceitos e processos de engenharia de *software* que auxiliam e orientam as diversas etapas que compõem seu desenvolvimento (Pressman, 2011).

Grande parte dos *software* desenvolvidos interagem com diversos componentes, como banco de dados, servidores de aplicação e até mesmo com outros *software*, que integram-se para atingir um certo objetivo, sendo assim conhecidos como sistemas (Pfleeger, 2004; Sommerville, 2007). Para garantir a qualidade de um sistema, seu processo de desenvolvimento deve seguir etapas bem definidas e deve ser mantido um relacionamento estreito e contínuo entre a equipe de desenvolvimento e os interessados[1]. As principais etapas do desenvolvimento de sistemas são apresentadas neste capítulo.

3.2 Processos de *Software*

O desenvolvimento de *software* não é composto de apenas uma ação, mas sim por diversas atividades que ao serem executadas em uma sequência específica resultam em um produto final. Esse conjunto de atividades que culminam em um *software* ou sistema consistem em um processo e como o processo de desenvolvimento de *software* está envolvido na criação de um produto, também recebe o nome de **ciclo de vida de *software*** (Pfleeger, 2004; Sommerville, 2007).

Na área de engenharia de *software* não há definição de um único processo de *software*,

[1] Pessoas que beneficiam-se de modo direto ou indireto do sistema (Pressman, 2011).

podendo o mesmo ser adaptado de acordo com a realidade da equipe de desenvolvimento e do projeto, porém sempre visando a qualidade e os prazos de entrega do produto (Pressman, 2011).

Algumas características comuns a diversos processos de *software* existentes são (Sommerville, 2007):

Especificação do *software*: consiste na definição de funcionalidades e restrições;

Projeto e implementação do *software*: visa a criação de um produto que atenda às especificações;

Validação do *software*: garantia de que o produto atenda às solicitações dos interessados;

Evolução do *software*: possibilidade de evolução do *software* para atender às necessidades dos interessados.

Apesar da existência de diversos processos de *software* na literatura, é possível identificar cinco principais atividades, as quais podem ser aplicadas tanto para a produção de *software* simples e pequenos, quanto para a criação de sistemas complexos e grandes. Essas cinco atividades são (Pressman, 2011):

Comunicação: a comunicação constante com os interessados é fundamental para o correto entendimento dos objetivos do projeto e para definir as funcionalidades e as características do sistema;

Planejamento: define as etapas necessárias para o desenvolvimento do sistema, fornecendo à equipe envolvida um guia sobre as atividades a serem executadas, os riscos, os recursos necessários para o projeto e um cronograma de trabalho;

Modelagem: compreende a criação de modelos para o entendimento dos requisitos e de como atendê-los;

Construção: compreende a geração de códigos e a elaboração de testes que avaliem erros na codificação;

Emprego: o sistema é entregue ao interessado que realiza a avaliação do produto e apresenta os resultados para a equipe de desenvolvimento.

Durante a execução de um processo de *software*, nem sempre é possível determinar previamente com exatidão todos os requisitos necessários para a construção de um sistema. Durante o levantamento dos requisitos, os interessados podem não verbalizar todas as informações necessárias para o correto desenvolvimento do produto. Além disso, quando são utilizados processos de *software* tradicionais, como o modelo cascata, os interessados têm contato com o

sistema apenas ao final do processo, fato que pode ocasionar uma experiência frustrante se o sistema não estiver de acordo com o que seria necessário. Este tipo de situação pode elevar os custos do projeto e atrasar as expectativas de entrega do produto final (Pfleeger, 2004; Sommerville, 2007; Pressman, 2011).

Na literatura são propostos alguns modelos de processo como a prototipação e o modelo incremental, cuja proposta busca contornar as dificuldades encontradas em modelos mais tradicionais e minimizar os riscos e incertezas do desenvolvimento (Pfleeger, 2004). As características desses modelos são apresentados a seguir.

Prototipação e Modelo Incremental

Existem na literatura modelos de processos com características interativas, conhecidos como modelos evolucionários e modelos incrementais, nos quais os interessados mantém um contato com o sistema em desenvolvimento desde as etapas iniciais. Desse modo, é possível que os problemas a serem resolvidos sejam mais bem compreendidos pelos interessados de modo que os requisitos do produto possam ser revisados ainda nas etapas iniciais ou intermediárias do projeto. Os benefícios dessa interação são refletidos no produto final (Sommerville, 2007).

Um dos modelos bastante conhecidos na literatura é o processo de prototipação, com seu ciclo representado na Figura 3.1. Esse processo orienta o desenvolvimento de um sistema por meio da construção e do refinamento de protótipos, em sucessivas interações com os usuários, auxiliando-os a compreender com mais clareza o produto que está sendo desenvolvido (Rezende, 2005).

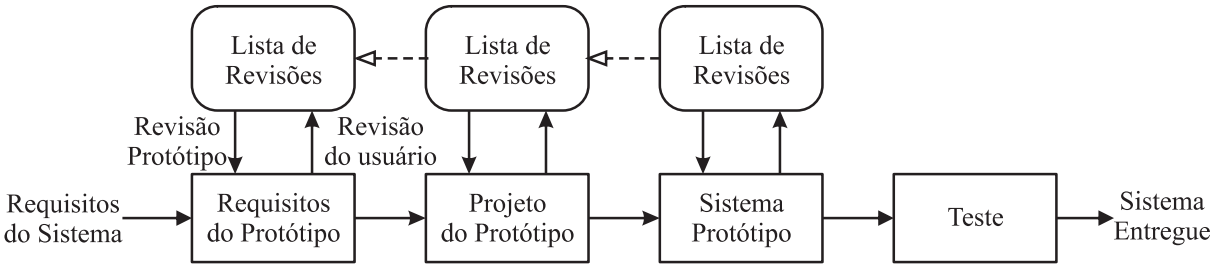

Figura 3.1: Modelo da prototipação (modificado de Pfleeger (2004)).

De acordo com o ciclo apresentado na Figura 3.1, o desenvolvimento inicia-se com um conjunto simples de requisitos fornecidos pelos interessados. São examinados os recursos e funções necessários ao sistema e em seguida são aplicados os ajustes necessários aos requisitos, de acordo com a revisão realizada. Desse modo, é possível realizar a codificação e apresentar uma versão do protótipo, que também é revisado e, se necessário, são realizadas novas interações para ajustes nos requisitos e no projeto (Pfleeger, 2004).

Como mencionado, no processo de prototipação, o desenvolvimento é iniciado com um conjunto simples de requisitos que a cada interação são revistos e refinados. Porém, em alguns

projetos há a necessidade de se obter rapidamente uma versão inicial do produto, que pode receber posteriormente novas funcionalidades. Em projetos que possuem essas características é indicado o modelo de processo incremental (Sommerville, 2007).

O modelo incremental combina as abordagens utilizadas em processos lineares, como o modelo cascata, e paralelos, como a prototipação. Nesse modelo são identificados os serviços que deverão ser fornecidos pelo sistema, classificando-os em subsistemas de acordo com suas funcionalidades. A cada versão, é adicionado um conjunto de funcionalidades ao sistema, iniciando com os elementos essenciais e finalizando após a implementação de todos os subsistemas (Pfleeger, 2004).

3.3 Levantamento e Especificação de Requisitos

Os requisitos de um sistema consistem na descrição dos serviços que devem ser fornecidos pelo sistema e suas restrições operacionais (Sommerville, 2007). Eles são obtidos a partir do entendimento das necessidades que devem ser atendidas pelo produto, sendo necessária uma forte interação com os interessados. Essa etapa é responsável pela ligação entre o projeto e a implementação do sistema (Pressman, 2011).

O levantamento dos requisitos ocorre na fase inicial do processo de desenvolvimento de sistemas e é de vital importância para o sucesso do projeto. Um levantamento de requisitos deficiente terá um grande impacto nas fases posteriores do projeto, pois quanto mais avançado estiver o projeto, maior serão os custos em decorrência da modificação dos requisitos iniciais.

Assim, os requisitos descrevem as funcionalidades e os comportamentos do sistema, podendo ser classificados como **funcionais**, os quais estão relacionados à descrição do funcionamento do sistema em seu ambiente, e **não-funcionais**, nos quais estão descritas as restrições do sistema (Pfleeger, 2004).

De posse dos requisitos é possível executar o processo de especificação que consiste em uma forma de documentação para fundamentar as etapas subsequentes do desenvolvimento de sistemas, podendo ser composta por uma combinação de linguagem natural e elementos gráficos (Pressman, 2011).

Para auxiliar o processo de especificação dos requisitos pode-se utilizar a linguagem natural estruturada, sendo esta uma forma de documentar os requisitos do sistema de maneira em que se une a facilidade de expressão e compreensão, provida pela linguagem natural, com a padronização pela limitação da terminologia utilizada na linguagem natural estruturada (Sommerville, 2007). Modelos de documentos, também conhecidos como *templates*, podem ser utilizados para especificar os requisitos. Caso alguma informação não possa ser esclarecida pelas limitações da linguagem natural estruturada, pode-se fazer uso de gráficos e tabelas como informação complementar.

3.4 Projeto

Na etapa de projeto, os requisitos são utilizados para a criação de modelos para o desenvolvimento do sistema. Os conceitos de projeto de *software* fornecem base para que o sistema desenvolvido funcione corretamente. Um projeto bem elaborado resulta em um produto de qualidade, atendendo aos requisitos e às restrições do sistema e facilitando sua manutenção.

O projeto apresenta a tradução dos requisitos, em forma de documentos, que servem de base para a construção do sistema, descrevendo as partes envolvidas e definindo como devem ser montadas (Braude, 2005). Assim, uma especificação completa, pode ser formada por quatro principais modelos de projeto (Pressman, 2011):

Projeto de classe: transformação dos modelos de classe, que representam os objetos que serão manipulados pelo sistema, e seus relacionamentos em estruturas de dados necessárias para a implementação do *software*;

Projeto arquitetural: relação entre os elementos estruturais que serão usados para satisfazer os requisitos ou as restrições do sistema;

Projeto da interface: descrição de como ocorre a comunicação entre o *software* e os demais elementos do sistema e entre o usuário;

Projeto em nível de componentes: elaboração de uma descrição procedural dos componentes do *software*, como modelos de fluxo e modelos comportamentais, presentes nos elementos estruturais de sua arquitetura.

UML como ferramenta de apoio ao projeto

Com a grande adoção por parte da indústria do paradigma de orientação a objetos para o desenvolvimento de sistemas, a linguagem *Unified Modeling Language* (*UML*)[2] pode ser considerada uma ferramenta fundamental para o desenvolvimento de projetos. A *UML* é uma linguagem de modelagem, extremamente útil para especificar, visualizar e documentar as características estruturais e comportamentais do projeto, não estando vinculada a nenhum processo específico de desenvolvimento (Pfleeger, 2004; Guedes, 2009).

A *UML* surgiu da união de alguns métodos de modelagem que eram, até a década de 90, os mais utilizados pelos profissionais de desenvolvimento de *software*. Esses métodos eram *Booch*, *Object Modeling Technique* (*OMT*) e *Object-Oriented Software Engineering* (*OOSE*). Através do apoio e do financiamento da *Rational Software*, a primeira versão da linguagem foi

[2]http://www.uml.org/

lançada em 1996 e atualmente encontra-se na versão 2.3[3], lançada oficialmente em maio de 2010 (Guedes, 2009).

Esta linguagem é composta por diversos diagramas, cada qual com uma função específica e representando somente parte do sistema, que devem ser usados conforme as necessidades do projeto. Dessa forma, o propósito geral de um conjunto de diagramas escolhido é formar um modelo geral do sistema, buscando minimizar a ocorrência de erros no projeto (Miles and Hamilton, 2006).

3.5 Implementação

Existem diversas maneiras de implementar um sistema, desde a utilização das linguagens de programação básicas, como C++ ou Java, até a utilização de *frameworks* e *IDE*'s (*Integrated Development Environment*) que automatizam e auxiliam na padronização de diversas tarefas durante o processo de implementação. Além disso, cada linguagem de programação foi criada com um objetivo específico, contendo características que permitem um melhor rendimento em áreas ou arquiteturas específicas, para as quais elas foram desenvolvidas.

Durante a fase de implementação, ocorre a tradução das decisões tomadas durante o projeto, em uma linguagem de programação. O código deve ser escrito de maneira cuidadosa e compreensível, seguindo as boas práticas de programação, de modo que a tarefa de revisão ou manutenção, mesmo quando realizada por outro profissional, seja a menos custosa possível. As características da linguagem de programação escolhida devem ser bem aproveitadas e os *software* devem ser desenvolvidos de forma a favorecer sua reutilização.

É importante observar algumas práticas durante a implementação de sistemas, de forma a manter a qualidade do produto e também facilitar o trabalho entre os profissionais de um mesmo projeto. Um dos itens importantes a considerar é a documentação do código implementado. Esta prática facilita o entendimento da lógica por parte de outros desenvolvedores e auxilia em processos posteriores de manutenção do código. Outro tópico fundamental é a padronização do estilo, formato ou conteúdo do código, principalmente quando diversos profissionais trabalham no mesmo sistema, o que mantém a correspondência entre os componentes de projeto e os componentes de código (Pfleeger, 2004).

3.6 Teste de *Software*

Os testes de *software* têm início durante a fase de implementação, quando após o desenvolvimento de procedimentos, funções ou métodos, o próprio desenvolvedor avalia a ló-

[3]http://www.omg.org/spec/UML/

gica implementada por meio de procedimentos conhecidos como testes de unidade (Crespo et al., 2011). Do mesmo modo, após o processo completo de implementação, o sistema desenvolvido deve ser testado para assegurar seu correto funcionamento. Porém, essa etapa inclui diferentes tipos de teste destinados a avaliar características específicas, de modo a minimizar a ocorrência de erros em quaisquer de seus componentes.

O processo de teste é uma atividade complexa, fundamental e de alto custo, representando de 50% a 80% do custo total do projeto. A engenharia de *software* propõe processos com o objetivo de otimizar os testes para que, utilizando um número limitado, sejam bem sucedidos. Assim, os testes consistem em executar de maneira controlada o *software* para verificar se o funcionamento está de acordo com as especificações e para expor seus defeitos antes que o sistema seja entregue aos interessados (Sommerville, 2007; Crespo et al., 2011).

O processo de teste pode ser dividido de acordo com a visão que o caracteriza. A visão interna consiste em um primeiro momento analisar a lógica interna do programa, sendo esta etapa conhecida como teste de **caixa-branca**. A visão externa consiste em realizar os testes conhecidos como **caixa-preta**, que são conduzidos pela interface do *software*. Este teste preocupa-se com a funcionalidade do sistema, sem levar em consideração a lógica interna, assumindo que os dados de entrada sejam adequadamente aceitos e a saída seja corretamente produzida (Pressman, 2011).

Dentro da engenharia de *software*, os testes são compostos de vários estágios, como representado na Figura 3.2. Como mencionado, em um primeiro momento, um componente do *software* é testado de forma isolada dos outros elementos do sistema, de maneira que seja verificado se o seu funcionamento está de acordo com suas especificações. Esta fase recebe o nome de **teste de unidade**. Em uma etapa seguinte, que recebe o nome de **teste de integração**, testa-se o funcionamento de vários componentes do sistema de maneira integrada, verificando se as especificações e o projeto do sistema foram corretamente seguidos. Após, o **teste de sistema** verifica o *software* como um todo, analisando se ele está de acordo com os requisitos funcionais e não funcionais, descritos na Seção 3.3. Ao final é executado o **teste de aceitação** que analisa se o sistema satisfaz os critérios de aceitação dos interessados (Pfleeger, 2004; Crespo et al., 2011).

Em grande parte dos projetos de desenvolvimento de sistemas, os testes são de responsabilidade da equipe de desenvolvimento e, em geral, são executados ao longo do processo. Em casos que envolvem sistemas críticos, torna-se necessário uma equipe separada para realizar os testes que são executados de acordo com os casos de teste criados com base na especificação detalhada do sistema. Entende-se **Casos de Teste** como um conjunto das especificações de entrada para o teste, a saída esperada do sistema e a declaração do que está sendo testado (Sommerville, 2007). Os testes de integração também podem ser realizados por uma equipe separada, baseando-se nos requisitos do sistema previamente levantados.

Conforme o tamanho do sistema desenvolvido, torna-se inviável a utilização de testes

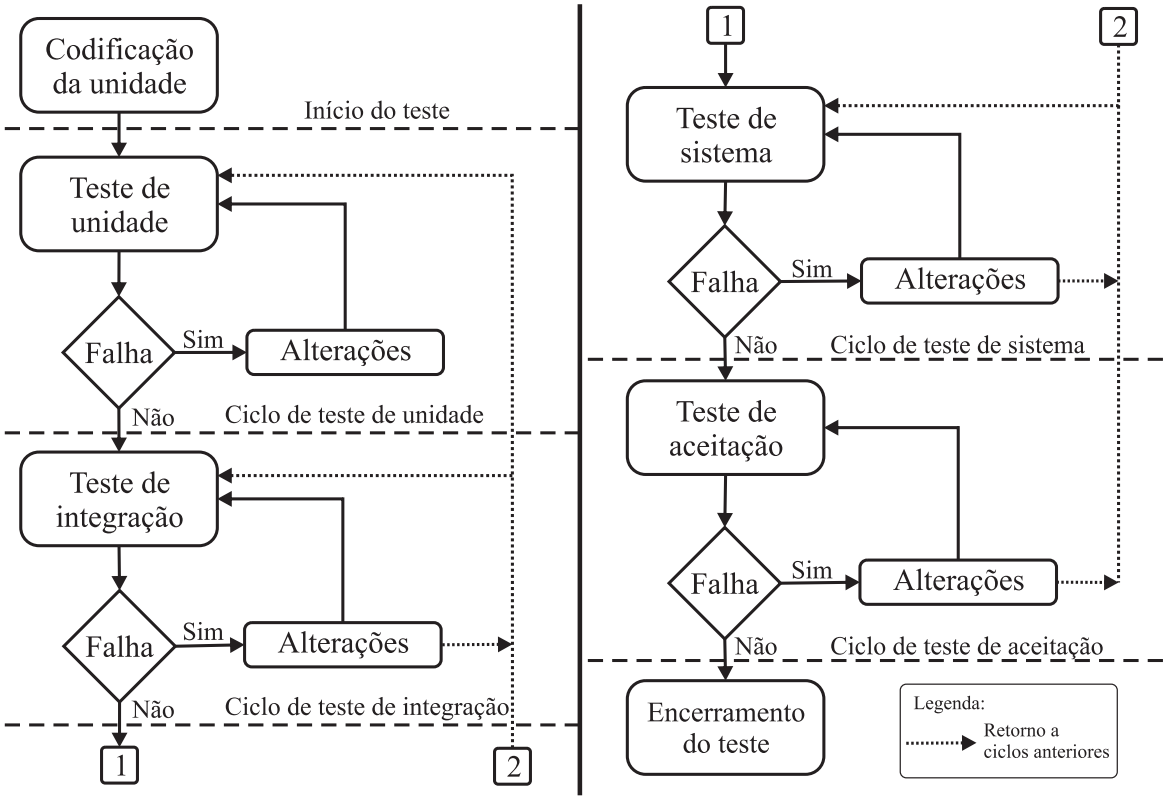

Figura 3.2: Ciclo de teste de *software* (modificado de Crespo et al. (2011)).

exaustivos, ou seja, que avaliem todas as possibilidades de execução, ficando a cargo da gerência determinar os casos de testes que serão aplicados ao produto. Eles devem ser elaborados de modo que possa ser encontrada a maior quantidade possível de erros com a menor quantidade possível de tempo e esforço (Pressman, 2011).

3.7 Criptografia em Sistemas de Informação

Sistemas computacionais podem concentrar diversos tipos de informações, como dados pessoais, procedimentos empresariais e informações financeiras. Em muitos casos exige-se um certo nível de confidencialidade, não sendo permitido o acesso por qualquer usuário à todas as partes do sistema. Outra preocupação de segurança envolve o canal de comunicação por onde trafegam os dados, que pode estar sujeito à interceptações não autorizadas, resultando na quebra de sigilo da informação. Para tratar essas e outras questões, diversos métodos e algoritmos têm sido propostos com o objetivo de garantir a segurança dos dados. Pode-se definir dois objetivos principais quanto aos métodos de segurança (Denning and Elizabeth, 1982):

- Evitar o acesso não autorizado;
- Garantir a autenticidade prevenindo modificações não autorizadas.

Para auxiliar o processo de segurança da informação são utilizadas diversas técnicas relacionadas a uma área de estudo denominada criptografia (Stallings, 2002). O termo criptografia pode ser definido como uma escrita secreta, em cifra, por meio de abreviaturas ou sinais convencionais (*Criptografia*, 2010). Também pode ser definido como o estudo de proteção da informação por meio de técnicas matemáticas de embaralhamento de modo que não seja possível decifrá-la sem a utilização da equação original ou chave criptográfica (Graves, 2007).

Essa técnica pode ser utilizada para garantir a segurança dos dados armazenados em disco ou daqueles que trafegam por meio de canais de comunicação, deixando-os em um formato ininteligível. Porém, também é importante utilizar mecanismos de controle de acessos, para que os dados estejam protegidos enquanto estão sendo processados (Denning and Elizabeth, 1982; Graves, 2007). A utilização de técnicas de criptografia para segurança dos dados tem como objetivo atender à quatro princípios fundamentais (Stewart et al., 2008; Oppliger, 2009):

Confidencialidade: garante que a privacidade dos dados esteja garantida enquanto são transmitidos entre duas ou mais partes;

Integridade: garante que os dados não sejam alterados durante sua transmissão, oferecendo a segurança de que a mensagem recebida é a mesma que foi enviada;

Autenticação: verifica a veracidade da identidade de um usuário do sistema, sendo uma das principais funções da criptografia;

Não-repúdio: provê a garantia de que os dados têm origem em um remetente autêntico. Previne que uma entidade envolvida na comunicação, negue sua participação após o ocorrido.

As diferentes abordagens para implementação de técnicas criptográficas podem ser organizadas em dois tipos (Graves, 2007):

Criptografia simétrica: tanto o remetente quanto o destinatário utilizam a mesma chave para criptografar e descriptografar os dados, porém não há uma maneira segura de compartilhar a chave entre múltiplos sistemas;

Criptografia assimétrica: foi criada com o intuito de contornar as deficiências da criptografia simétrica. Surgem os conceitos de chave pública e chave privada, na qual cada aplicação possui seu respectivo par de chaves. Quando duas aplicações precisam transmitir informações criptografadas, o remetente cifra os dados utilizando a chave pública do destinatário, que por sua vez consegue decifrar a mensagem com a utilização de sua chave privada. Esse processo é representado pela Figura 3.3.

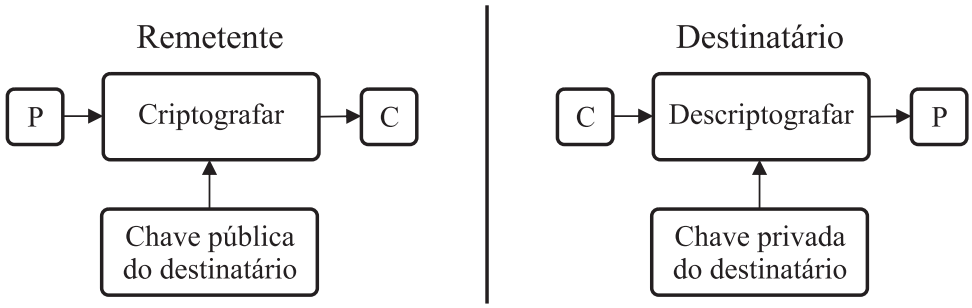

Figura 3.3: Modelo de criptografia assimétrica. (P) indica dados sem criptografia e (C) indica dados cifrados (modificado de Stewart et al. (2008)).

Tanto a abordagem simétrica quanto a assimétrica podem fazer uso de diferentes algoritmos, entre os quais encontram-se os algoritmos de *hashing* que auxiliam na garantia de autenticidade da informação. Alguns exemplos de algoritmos dessa categoria são apresentados a seguir.

3.7.1 Algoritmos de *Hashing*

Os algoritmos de *hashing* sumarizam a informação contida nos dados gerando uma chave numérica única, o que permite a verificação da autenticidade dos dados ou até mesmo a criação de assinaturas digitais, assegurando a origem da informação e que ela não foi modificada durante sua transmissão (Stewart et al., 2008). Diversos algoritmos foram concebidos ao longo do tempo, provendo diferentes níveis de criptografia. A seguir são descritos alguns dos algoritmos mais utilizados na literatura (Stanger et al., 2002; Graves, 2007; Stewart et al., 2008):

***Message Digest 2*:** O algoritmo *Message Digest 2* foi desenvolvido por Ronald Rivest em 1989, direcionado para processadores 8-bit, com capacidade de gerar um valor *hash* de 129-bit;

***Message Digest 4*:** diversas melhorias na implementação do antecessor *Message Digest 2* levaram à criação do algoritmo *Message Digest 4* em 1990, já com suporte para processadores 32-bit, provendo um valor *hash* de 128-bit e executando mais rapidamente que o algoritmo *Message Digest 2*;

***Message Digest 5*:** a fragilidade na segurança do algoritmo *Message Digest 4* levou à implementação do algoritmo *Message Digest 5* em 1991, sendo muito utilizado para a criação de assinatura digital de dados, *hashing* de senhas e verificação de integridade de arquivos. A adição de novas características de segurança ocasionou a diminuição de seu desempenho em relação ao seu antecessor;

***SHA*:** O *Secure Hash Algorithm* (*SHA*) e o *SHA-1* compõem um padrão governamental de funções *hash* desenvolvido pelo *National Institute of Standards and Technology*

(*NIST*) nos Estados Unidos. O *SHA-1* é o sucessor do padrão *SHA*, permitindo uma entrada de dados de qualquer tamanho e gerando um valor *hash* de 160-bit. Apesar de seu desempenho ser inferior ao algoritmo *Message Digest 5*, este algoritmo é considerado mais seguro. Existem diversas variações deste algoritmo, provendo uma segurança maior do valor *hash*.

Outra preocupação existente em relação à segurança dos dados é garantir que as informações não sejam interceptadas enquanto trafegam em canais de comunicação, ou até mesmo, se forem capturadas, que não seja possível compreender a informação. Para tratar este problema, foram criados mecanismos que criam canais seguros de comunicação e cifram a informação que trafega por eles, como é o caso do *Secure Sockets Layer* (*SSL*) e do *Transport Layer Security* (*TLS*), descritos a seguir.

3.7.2 Segurança na Transmissão de Dados

O protocolo *SSL* foi desenvolvido pela Netscape[4] com intuito de criptografar os dados trafegados pela *Web* entre o servidor e o cliente. A primeira versão do protocolo *SSL* surgiu em 1994 e mais duas versões foram lançadas em um curto espaço de tempo, resultando na versão 3 do protocolo em 1995. Um dos motivos de sua popularização foi a adoção pela *Microsoft* como padrão de segurança para o navegador *Internet Explorer* (Thomas, 2000; Stewart et al., 2008). Dentro do protocolo *SSL* existem outros subprotocolos (The Mozilla Foundation, 2007; Oppliger, 2009):

***SSL record*:** é responsável pelo encapsulamento dos dados de protocolos das camadas de mais alto nível;

***SSL handshake*:** envolve a utilização do protocolo *SSL record* para realizar uma série de trocas de mensagens entre o cliente e o servidor a fim de executar a autenticação e negociar o método de criptografia e a compressão utilizados para a comunicação. A negociação entre o servidor e o cliente ocorre durante o estabelecimento da primeira conexão;

***SSL Change Cipher Spec*:** coloca em prática os parâmetros de segurança negociados pelo protocolo *SSL handshake*;

***SSL Alert Protocol*:** emite sinais indicadores de possíveis problemas e possibilita a troca dessas mensagens de alerta;

***SSL Application Data Protocol*:** realiza a transmissão segura dos dados envolvidos na comunicação entre cliente e servidor.

[4]http://isp.netscape.com/

A comunicação criptografada desse protocolo é realizada por padrão por meio da porta 443 (Stewart et al., 2008). Basicamente, o procedimento de comunicação inicia com a solicitação da comunicação criptografada por parte do cliente. O servidor envia sua chave pública que serve para criptografar uma chave criada aleatoriamente pelo navegador do cliente. Após criptografada, a chave é enviada para o servidor que estabelece uma conexão segura com o cliente (The Mozilla Foundation, 2007).

O protocolo *SSL* suporta diversos algoritmos de criptografia, escolhendo aquele que é compatível com o cliente que solicita a conexão segura. Ele utiliza uma combinação de criptografia assimétrica e simétrica, devido ao fato de que a criptografia assimétrica oferece um melhor mecanismo de autenticação, enquanto que a criptografia simétrica apresenta um melhor desempenho. Durante a fase da autenticação realizada pelo protocolo *SSL handshake*, o algoritmo de criptografia simétrica é escolhido e a comunicação entre o cliente e o servidor é estabelecida (The Mozilla Foundation, 2007). Apesar de ser muito utilizado, o protocolo *SSL* apresenta algumas limitações (Thomas, 2000):

- Funciona apenas sobre o protocolo *Transmission Control Protocol*[5];
- Não oferece suporte ao princípio de não-repúdio, que é responsável pela prevenção de situações em que uma parte assina digitalmente os dados e após enviá-los nega o fato de tê-los assinado;
- Está sujeito às limitações dos algoritmos de criptografia utilizados.

Como evolução do *SSL* foi criado o protocolo *TLS*, ambos possuindo a mesma estrutura. Sua primeira versão foi lançada em 1999 a partir de diversas melhorias em relação ao seu antecessor. Em conjunto com o *SSL*, são os protocolos de segurança mais utilizados atualmente na internet. As diferenças existentes entre os dois protocolos são sutis, sendo algumas delas (Oppliger, 2009):

- O protocolo *TLS* também pode ser aplicado ao protocolo *User Datagram Protocol*[6];
- Diferente do *SSL*, no qual há sempre quatro estados de conexão pendentes (estado *read* e *write* atual; estado *read* e *write* pendente), os registros no protocolo *TLS* são processados sob um único estado (estado *read* e *write* atual);
- Nas versões mais antigas do *TLS* é utilizada uma função pseudorrandômica que combina *Message Digest 5* e *SHA-1* para criação de chaves criptográficas, sendo uma das principais diferenças em relação ao *SSL* que não utiliza a combinação de dois algoritmos de *hashing*.

Atualmente o protocolo *TLS* se encontra na versão 1.2 que foi lançada em 2008, sendo regulamentado pela norma **RFC 5246**[7] do *The Internet Engineering Task Force* (*IETF*), que

[5] *TCP* - http://www.ietf.org/rfc/rfc793.txt
[6] *UDP* - http://tools.ietf.org/html/rfc768
[7] http://tools.ietf.org/html/rfc5246

é um órgão internacional de padronização das questões relacionadas à estrutura da internet. Essa norma apresenta todas as características técnicas do protocolo bem como as diferenças em relação a sua versão anterior.

3.8 Considerações Finais

O processo de desenvolvimento de sistemas computacionais requer uma forte interação entre a equipe de desenvolvimento e os interessados, de modo que sejam captadas todas as necessidades e restrições que precisam ser incorporadas ao projeto. O projeto, por sua vez, é uma ferramenta importante que direciona à um produto de qualidade e orienta a equipe para que o produto seja entregue dentro dos prazos estabelecidos, sendo esta uma tarefa difícil de gerenciar. Dentro do projeto algumas etapas merecem uma atenção especial, como as fases de implementação e testes. Nestas etapas deve-se ter em mente a construção de um *software* que possa ser facilmente mantido e assegurar que o mesmo atenda às especificações e às restrições definidas no projeto.

A segurança dos dados também representa uma preocupação que deve ser incorporada aos projetos de *software*, não só em relação à autenticação dos usuários do sistema, mas também em relação ao uso da rede para tráfego de dados. Sendo assim, técnicas de *hashing* e criptografia estão entre os recursos mais comumente utilizados para a segurança da informação, porém muitas outras técnicas têm sido utilizadas em associação, de modo a aumentar o nível de segurança tanto dos dados quanto dos sistemas, como é o caso da biometria.

Outro ponto importante durante a etapa de desenvolvimento de sistemas computacionais é a escolha da linguagem de programação e outras ferramentas que sejam adequadas aos propósitos do projeto. Existem diversas linguagens de programação e *frameworks* que auxiliam os desenvolvedores em suas tarefas, cada qual possuindo características e propriedades mais adequadas a determinadas situações. No Capítulo 4 são apresentadas as ferramentas escolhidas para o desenvolvimento do protótipo projetado e desenvolvido neste trabalho.

Capítulo 4

Ferramentas para Desenvolvimento de Sistemas de *Software*

4.1 Considerações Iniciais

A área de desenvolvimento de sistemas tem evoluído muito ao longo dos anos, apresentando diversas ferramentas e linguagens de programação, cada qual criadas para um fim específico. Atualmente, uma área em ascensão é o desenvolvimento de sistemas *Web* devido às características desses sistemas, entre elas a facilidade de utilização pelo usuário, já que o mesmo aplicativo pode ser acessado por meio de diferentes sistemas operacionais, necessitando apenas de um navegador *Web*. Outra característica relevante de sistemas *Web* é o fato de não ser necessário realizar a instalação na máquina do usuário, permitindo o acesso ao sistema por meio de diferentes computadores que estejam conectados ao servidor onde está instalada a aplicação.

Geralmente, o desenvolvimento de sistemas *Web* é composto pela intersecção de diversas tecnologias, por exemplo *HTML*, *CSS*, *JavaScript*, entre outras, exigindo um conhecimento técnico mais amplo para a fase de implementação. Além disso, essa área requer uma atenção especial ao comportamento do sistema quando executado em diferentes navegadores *Web*, já que cada navegador apresenta características próprias, podendo resultar em um comportamento inesperado da aplicação.

Diversas linguagens de programação e *frameworks* destinados à *Web* visam tornar o desenvolvimento mais ágil, robusto e melhorar a interatividade do sistema com o usuário. Neste capítulo são apresentadas algumas dessas tecnologias que também foram adotadas na implementação do protótipo desenvolvido neste trabalho.

4.2 Linguagem *Ruby*

Ruby é uma linguagem de programação livre e portável, podendo ser utilizada em ambientes como *GNU/Linux*, *Windows*, *Mac OS*, entre outros (Ruby Visual Identity Team, 2010).

Foi criada por Yukihiro Matsumoto e teve sua primeira versão lançada em 1995. Muitas de suas características vêm de linguagens de *script* como *Python* e *Perl*. Trata-se de uma linguagem interpretada, e puramente orientada a objetos, cujo comportamento foi fortemente influenciado por outras linguagens como *Perl*, *Smalltalk*, *Eiffel*, *Ada* e *Lisp* (Collingbourne, 2008).

O intuito de Matsumoto, também conhecido como Matz, era construir uma linguagem que equilibra a programação funcional com a programação imperativa. A popularidade da linguagem teve um grande crescimento a partir de 2006, sendo um dos fatores de contribuição para sua adoção a criação e a ascensão do *framework* para desenvolvimento *Web Ruby on Rails* (Ruby Visual Identity Team, 2010).

Uma das características da linguagem *Ruby* é a sua flexibilidade, permitindo aos desenvolvedores redefinirem ou acrescentarem partes na linguagem, por exemplo, a substituição do operador "+" pelo operador personalizado "plus" (Ruby Visual Identity Team, 2010). Outra característica forte da linguagem é a metaprogramação, ou seja, a capacidade de produzir codificações com a habilidade de gerar ou manipular código. É possível ainda fazer uso dessa característica para a produção de *Domain-Specific Language* (*DSL*), sendo este um recurso que define uma linguagem focada no domínio de um problema específico (Perrotta, 2010).

Todos os componentes da linguagem *Ruby* são objetos, até mesmo números ou outros elementos considerados tipos primitivos em linguagens como *Java* e *C++*. Sua sintaxe é intuitiva, assemelhando-se a palavras da língua inglesa, assim como *Pascal*. *Ruby* possui gerenciamento automático de memória e é uma linguagem de tipagem dinâmica e forte, não sendo necessária a declaração de variáveis para a utilização. Alguns caracteres especiais são utilizados para especificar a condição de um determinado elemento, como os caracteres "$" e "@", que identificam as variáveis como global e de instância, respectivamente (Ruby Visual Identity Team, 2010).

Outra característica importante de *Ruby* consiste na possibilidade de desenvolver bibliotecas de código e até mesmo aplicações e disponibilizá-las em um formato padronizado de pacote que recebem o nome de *gem*. Esses pacotes são facilmente instaláveis, permitindo de modo simples utilizar os recursos disponibilizados por cada *gem* em uma aplicação em desenvolvimento. O gerenciamento das *gems* de um ambiente ocorre por meio do *framework RubyGems* (Thomas et al., 2009).

Existem paralelamente diferentes versões da linguagem *Ruby*. A versão 1.9 representa o último *release* da linguagem, possuindo uma nova implementação que provê um aumento de desempenho, algumas mudanças na sintaxe e nos recursos da linguagem, entre outras melhorias (Cooper, 2007). A partir da versão 1.9 o gerenciador de pacotes *RubyGems* é instalado juntamente com a linguagem, diferentemente das versões anteriores de *Ruby* nas quais era necessário instalar o *RubyGems* separadamente (Thomas et al., 2009).

4.3 *Framework Ruby on Rails*

Como mencionado, *Ruby on Rails* é um *framework* baseado na linguagem *Ruby* que foi criado com o intuito de facilitar a implementação, *deploy* e a manutenção de sistemas *Web*. Este *framework* foi desenvolvido por David Heinemeier Hansson e teve seu primeiro lançamento em 2004, sob licença MIT[1]. Atualmente é mantido por muitos voluntários do mundo todo que compõem uma comunidade ativa (Hellsten and Laine, 2006).

O objetivo da construção deste *framework* é resolver 80% dos problemas relacionados ao desenvolvimento de aplicações *Web*, considerando que os outros 20% são dependentes do contexto do projeto em questão (Carneiro Junior and Barazi, 2010). A filosofia incorporada ao *framework Ruby on Rails* é composta por uma série de princípios, sendo os principais (Monteiro, 2010):

***Don't Repeat Yourself (DRY)*:** Desencoraja a repetição do mesmo código em diversas partes do sistema, estimulando a criação de códigos que possam ser reutilizados em diferentes locais;

***Convention Over Configuration*:** Estabelece convenções para o desenvolvimento de aplicações, diminuindo a necessidade de definir diversos parâmetros em arquivos de configuração;

***Representational State Transfer (REST)*:** Trata-se de uma série de padrões para realizar a comunicação entre os componentes de um sistema *Web* por meio de um conjunto de operações do protocolo *Hypertext Transfer Protocol* (*HTTP*), como *POST*, *GET*, *PUT* e *DELETE*.

Sua arquitetura para desenvolvimento segue o padrão *Model View Controller* (*MVC*), representado na Figura 4.1, o que permite uma melhor organização do sistema, separando o modelo de dados (*Model*), a camada de visualização (*View*) e a lógica de negócios (*Controller*) (Carneiro Junior and Barazi, 2010).

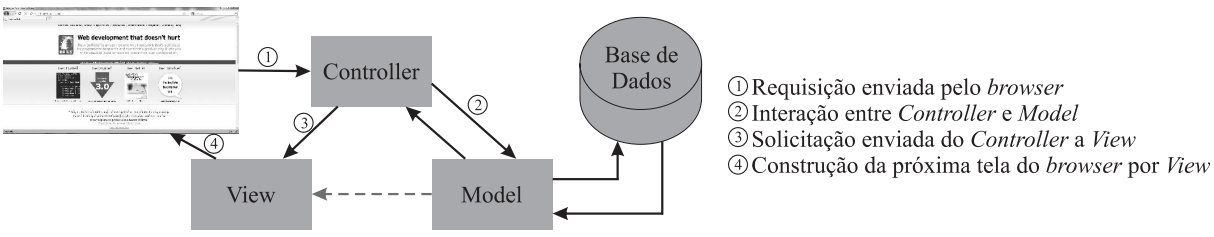

Figura 4.1: O modelo *MVC* (modificado de Ruby et al. (2010)).

Ruby on Rails possui mecanismos que impulsionam o desenvolvimento de sistemas, automatizando diversas tarefas, tal como o *script generate*, responsável por gerar *Controllers*, *Views*

[1]Permite a reutilização do software licenciado em programas livres ou proprietários (Initiative, 2010).

e *Models* do projeto, criando toda a estrutura de arquivos e o código base da aplicação (Hellsten and Laine, 2006). Outro comando muito conhecido e utilizado é o *scaffold*, responsável pela criação de toda estrutura *CRUD*, acrônimo para *Create, Read, Update* e *Delete*, que representa as operações básicas de manipulação de dados em um sistema (Griffiths, 2008). Além de ser possível gerar estruturas *CRUD* com facilidade, são disponibilizados mecanismos para auxiliar na validação dos dados inseridos nos formulários presentes na interface gráfica do sistema, sendo uma alternativa para assegurar padrões específicos dos dados que serão inseridos na aplicação (Carneiro Junior and Barazi, 2010).

Ruby on Rails ainda oferece suporte para a utilização de diversos bancos de dados, como *DB2*, *Oracle*, *MySQL*, *PostgreSQL*, *Firebird* e *SQL Server*, realizando de maneira padronizada as operações de manipulação de dados. Essas operações são gerenciadas por meio do módulo *Active Record*, um *Object-Relational Mapping* (*ORM*) que compõe o modelo da aplicação (Ruby et al., 2010). Isso permite que, quando necessário, as configurações relativas ao banco de dados escolhido sejam alteradas com a edição de apenas alguns parâmetros em arquivos de configuração do *framework* (Ruby et al., 2010).

Atualmente, *Ruby on Rails* está em sua versão 3, resultado de muitas modificações e melhorias em relação a sua versão anterior. Além das características apresentadas, muitas outras fazem com que cada vez mais desenvolvedores adotem *Ruby on Rails* como plataforma de desenvolvimento (Carneiro Junior and Barazi, 2010).

4.4 Linguagem *JavaScript*

JavaScript é uma linguagem interpretada, com tipagem fraca e características de orientação a objeto. Sua sintaxe assemelha-se muito às linguagens *C*, *C++* e *Java*. Sua primeira versão foi desenvolvida em 1996, por Brendan Eich, na *Netscape*[2] (The Mozilla Foundation, 2010a). Em muitas características, percebe-se sua inspiração na linguagem *Perl*, tal como expressões regulares (Flanagan, 2006). Esta linguagem é *prototype-based*, que basicamente difere de linguagens *class-based*, como *Java* e *C++*, por não fazer distinção entre classes e objetos, considerando somente o conceito de objetos (The Mozilla Foundation, 2010b).

JavaScript é uma linguagem muito utilizada na *Web* e possibilita a criação de sites mais interativos por meio de funcionalidades como efeitos visuais, validação de formulários, entre outros (Pollock, 2004). Os códigos *JavaScript* são incorporados em arquivos *HyperText Markup Language* (*HTML*) e são executados na máquina cliente, não utilizando os recursos do servidor *Web* durante sua execução. Como as páginas *Web* são acessadas por meio de diferentes navegadores, é necessária uma atenção especial para o uso desta linguagem, pois o comportamento da aplicação pode variar de acordo com o navegador *Web* utilizado (Flanagan, 2006).

[2] http://isp.netscape.com

Por meio da integração de tecnologias com *JavaScript*, como *Extensible Markup Language* (*XML*) ou *Javascript Object Notation* (*JSON*), podem ser desenvolvidas aplicações *Web* dinâmicas, oferecendo suporte a diversas funcionalidades que proporcionam uma melhor interação entre a página *Web* e o usuário. A linguagem *XML* é uma linguagem de marcação, criada em 1996, que tem como finalidade a descrição e o armazenamento de dados por meio de marcações conhecidas como *tags*, que estruturam os dados armazenados. Também é uma linguagem extensível, pela qual é possível criar *tags* personalizadas (Silva, 2001). Já o *JSON*[3] oferece um padrão estruturado de dados que é facilmente manipulável por meio da linguagem *JavaScript*, sendo uma alternativa interessante em relação a utilização de *XML* (Heilmann, 2006).

Essa relação entre componentes como *JavaScript*, *XML* ou *JSON*, possibilita a criação de aplicações com fluxo de dados diferenciado, introduzindo o conceito de *Asynchronous JavaScript and XML* (*Ajax*). Esse termo foi criado por Jesse James Garret no ano de 2005 e descreve um método de desenvolvimento diferente das abordagens tradicionais (Heilmann, 2006). Esse método é formado por diversas tecnologias *Web*, permitindo criar aplicações que utilizem tráfego de informações de maneira assíncrona, evitando o recarregamento de todo o conteúdo de uma página *Web* ao realizar uma requisição, como ocorre em sites que utilizam apenas *HTML* (Powell, 2008).

Este método tem desempenhado um papel importante na criação de interfaces *Web* ricas, que podem ser construídas por meio de uma relação entre tecnologias como *JavaScript*, *Extensible Hypertext Markup Language* (*XHTML*) ou *HTML* e *Cascading Style Sheets* (*CSS*) (Lauriat, 2007). Esse estilo de aplicação faz uso de *Ajax*, realizando somente o recarregamento de partes específicas da página *Web* que sofreram alterações, tornando possível a interação entre usuário e aplicação durante este processo, assim como é apresentado na Figura 4.2.

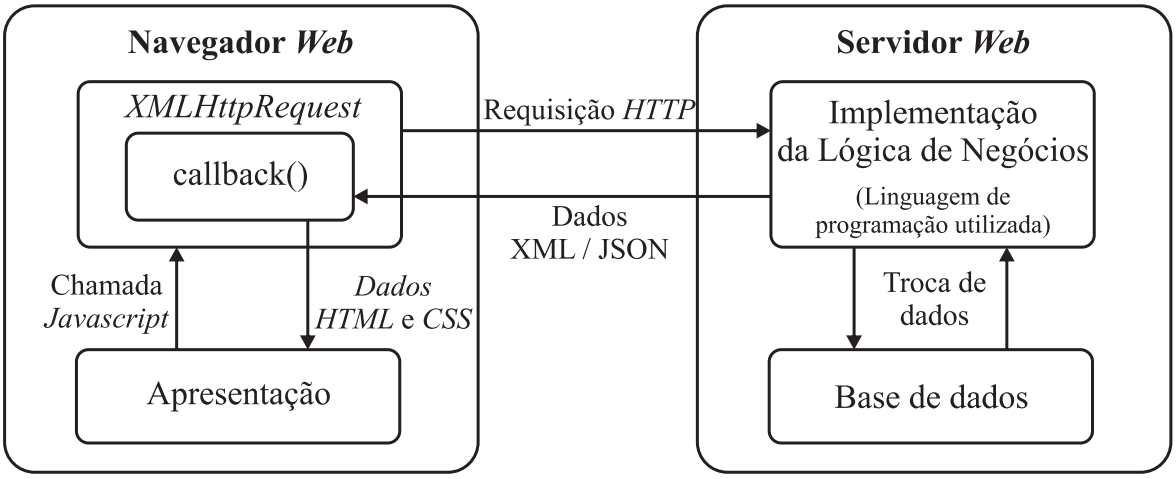

Figura 4.2: O modelo de comunicação utilizado pelo conceito de *Ajax* (modificado de Netbeans.org (2012)).

Para realizar as transações é utilizado um objeto chamado *XMLHttpRequest* (*XHR*), que

[3]http://json.org/

não faz parte do padrão oficial do *World Wide Web Consortium* (W3C)[4], porém é suportado por grande parte dos navegadores *Web*, como *Mozilla Firefox*, *Opera* e *Safari*. O *Microsoft Internet Explorer* não faz uso do *XHR*, utilizando em seu lugar o *ActiveX*, que funciona de maneira semelhante (Heilmann, 2006).

Diferente de *JavaScript*, que pode ser executado somente na máquina cliente, para utilizar *Ajax* torna-se necessária a presença de um servidor *Web*, já que esta tecnologia faz uso do *XHR* para realizar as requisições ao servidor (Heilmann, 2006). Após receber a requisição, o servidor gera a resposta no formato *XML*, encaminhando-a para o cliente que fez a solicitação. Como mencionado, ao invés de *XML*, outros formatos podem ser utilizados, tal como *HTML* ou *JSON* (Powell, 2008).

De modo a simplificar o desenvolvimento com *JavaScript*, foram criados *frameworks* que disponibilizam funções para diversos propósitos comuns no desenvolvimento de aplicações *Web*. Entre elas encontram-se o *Prototype*[5] e o *jQuery*[6].

Atualmente a utilização de *jQuery* tem crescido frente aos demais *frameworks* do gênero. Este *framework* teve origem em 2005, sendo desenvolvido inicialmente por John Resig e atualmente mantido por uma grande comunidade, por ser de código aberto. Além de oferecer diversas funcionalidades, o modo de utilização do *jQuery* é simples, não exigindo do usuário um conhecimento amplo da linguagem *JavaScript* para utilizá-lo (Porteneuve, 2010). Entre as funcionalidades disponibilizadas pelo *framework* destacam-se (Chaffer and Swedberg, 2007):

- Fácil acesso à partes de uma página *HTML* e alteração de seu conteúdo;
- Capacidade de modificar a aparência de uma página por meio do acesso e da aplicação de estilos presentes no *CSS*;
- Capacidade de responder a interações do usuário com a página;
- Possibilidade de adicionar animações gráficas a uma página;
- Capacidade de utilização de operações que envolvam *Ajax*.

Pelas características e funcionalidades que envolvem a linguagem *JavaScript* e *frameworks* relacionados é possível concluir que desempenham um papel importante para o desenvolvimento de sistemas *Web*, principalmente nas questões relacionadas à produção de interfaces gráficas de utilização para o usuário, impactando diretamente na usabilidade do sistema desenvolvido.

[4]Consórcio internacional destinado a desenvolver padrões para a *Web* (http://www.w3c.br).
[5]http://www.prototypejs.org/
[6]http://jquery.com/

4.5 Sistema Gerenciador de Banco de Dados *PostgreSQL*

Uma das maiores contribuições dos sistemas computacionais consiste em gerenciar grandes quantidades de informações, que por sua vez precisam ser armazenadas em alguma estrutura apropriada. Entre as estruturas mais utilizadas encontram-se os Bancos de Dados (BD). Para gerenciar as estruturas e dados relacionados nos BD foram criados sistemas de *software* de propósito geral, que possibilitam a manipulação e o compartilhamento do banco de dados entre diversos usuários e aplicações. Um sistema com esse propósito é denominado de **Sistema Gerenciador de Banco de Dados** (SGBD) (Elmasri and Navathe, 2005).

Atualmente, existem diversos SGBD com características diferentes. Muitos desses sistemas são comerciais e apresentam diversas funcionalidades para manipulação de dados, incluindo linguagens de programação próprias que funcionam internamente ao sistema, como é o caso do SGBD *Oracle*[7] e sua linguagem *Procedural Language/Structured Query Language* (*PL/SQL*). Outros SGBD possuem condições de uso diferenciadas, sendo livres de licenças quando relacionados a projetos de código aberto, como é o caso do MySQL[8], o qual é muito utilizado em *websites* e sistemas *Web*. Outros ainda são projetos livres, com código aberto e mantidos por uma grande comunidade, como é o caso do *PostgreSQL*[9].

Os SGBD podem ser classificados de acordo com diversos critérios, entre os quais destaca-se o modelo de dados base do sistema. Os mais utilizados adotam o modelo de dados relacional, que representa a base de dados como uma coleção de tabelas. Uma evolução do modelo relacional resultou no modelo objeto-relacional, que incorporou conceitos da orientação a objetos no modelo existente (Elmasri and Navathe, 2005).

Dados os conceitos, torna-se mais fácil definir *PostgreSQL* como um SGBD objeto-relacional cujo desenvolvimento iniciou na Universidade da Califórnia, no ano de 1977, com um projeto chamado *Ingres*. Muitos dos conceitos desenvolvidos no *PostgreSQL* foram pioneiros, surgindo em SGBD comerciais somente anos depois. Atualmente é considerado o SGBD de código aberto mais avançado, se equiparado a produtos comerciais de alto nível (Worsley and Drake, 2002; The PostgreSQL Global Development Group, 2011).

Para realizar a manipulação de dados os SGBD fazem uso de uma linguagem declarativa denominada *Structured Query Language* (*SQL*), implementada a partir de um projeto da *International Business Machines* (*IBM*) (Stinson, 2001). As funcionalidades presentes nesta linguagem oferecem grande controle sobre os dados, possibilitando tarefas de definição de dados, consultas e atualizações. Essas funcionalidades são constantemente revisadas, o que resulta em novas atualizações da linguagem (Elmasri and Navathe, 2005).

O *PostgreSQL* permite a manipulação de dados por meio da linguagem *SQL*, assim como

[7] http://www.oracle.com/us/products/database/index.html
[8] http://www.mysql.com
[9] http://www.postgresql.org/

outros SGBD, e também oferece uma linguagem de programação própria, cujo nome é *Procedural Language/PostgreSQL Structured Query Language* (*PL/pgSQL*). Essa linguagem própria permite a implementação de recursos como funções, estruturas de controle, tipos e operadores definidos pelo usuário, além de outros processos computacionais. Os códigos desenvolvidos em *PL/pgSQL* permanecem no SGBD e podem executar, quando for solicitado, um conjunto de computações e retornar seu resultado, diminuindo a sobrecarga de informações trafegadas pela rede. Procedimentos semelhantes com a linguagem *SQL* exigiriam uma quantidade maior de consultas ao banco de dados, aumentando a comunicação entre cliente e servidor e resultando em um menor desempenho da aplicação se comparada a utilização de recursos do *PL/pgSQL* (The PostgreSQL Global Development Group, 2011).

Além da linguagem *PL/pgSQL*, outras funcionalidades destacam o *PostgreSQL* entre os SGBD existentes, como o controle de concorrência, que garante a consistência dos dados acessados de modo concorrente, replicação assíncrona, *backups online*, transações aninhadas, *log* de alterações para restauração caso ocorram erros entre outras (Worsley and Drake, 2002; The PostgreSQL Global Development Group, 2011).

4.6 Considerações Finais

As ferramentas para desenvolvimento de sistemas representam uma escolha importante dentro dos projetos de desenvolvimento de *software*. Para realizar esta escolha deve-se observar as características da linguagem de programação, *frameworks*, SGBD ou outras ferramentas adotadas, assim como foi apresentado nesta seção. Grande parte do tempo despendido para a conclusão do projeto está relacionado à utilização dessas ferramentas durante a fase de implementação. Sendo assim, esse processo influencia diretamente nos prazos de entrega do produto e dificuldades no desenvolvimento das funcionalidades que compõem o sistema.

As ferramentas apresentadas neste capítulo têm se mostrado eficientes para o desenvolvimento de sistemas *Web*, sendo cada vez mais utilizadas pela comunidade tanto em projetos comerciais quanto de código aberto. A linguagem *Ruby*, devido às suas características, favorece a criação de um código legível e ao mesmo tempo capaz de executar tarefas complexas com uma quantidade menor de linhas de código. Essas características somadas à filosofia do *framework Ruby on Rails*, bem como suas funcionalidades adicionais, abstraem muitos requisitos essenciais às aplicações *Web*, reduzindo assim a complexidade durante a implementação do projeto. Além disso, é possível realizar a integração de modo simples com *frameworks* adicionais, como é o caso do *jQuery* que contribui significativamente com a melhora, em questões de usabilidade, da interface gráfica com o usuário. Em relação à persistência dos dados, o SGBD *PostgreSQL* mostra-se robusto e com recursos que garantem a consistência dos dados armazenados, além de integrar-se facilmente com o *framework Ruby on Rails*.

De um modo geral, todas essas características podem contribuir para o desenvolvimento

de sistemas *Web*. Porém, é preciso notar que sistemas biomédicos possuem algumas particularidades, repercutindo, portanto, no protótipo desenvolvido neste trabalho. No Capítulo 5 serão apresentadas as principais características de sistemas biomédicos, as exigências para armazenamento de dados médicos bem como as normas brasileiras que regulamentam os sistemas para registro eletrônico de saúde.

Capítulo 5

Sistemas de Informação para Saúde

5.1 Considerações Iniciais

A quantidade de informação gerada a partir de procedimentos aplicados a pacientes é muito grande e precisa ser registrada em documentos que compõem o histórico médico do paciente, que além de ser utilizado para seu tratamento pode também ser utilizado no desenvolvimento de pesquisas e trabalhos acadêmicos.

Como consequência da necessidade de registros médicos, o volume de prontuários gerenciados por instituições de saúde cresce consideravelmente ao longo do tempo, trazendo consigo problemas de armazenamento, como espaço físico e condições ambientais, que podem comprometer o estado dos documentos. Outra dificuldade refere-se ao levantamento de informações específicas presentes nos prontuários, sendo este um processo que precisa ser executado manualmente.

De modo a simplificar a aquisição e o gerenciamento de dados médicos, a utilização de recursos computacionais tem sido amplamente adotadas, dando origem aos Sistemas para Registro Eletrônico em Saúde (S-RES). Porém, devido à responsabilidade pertinente ao armazenamento de dados médicos, esses sistemas devem estar de acordo com uma legislação específica criada especialmente para regular os requisitos mínimos necessários de modo a manter a informação em formato digital.

Neste capítulo são abordadas as características dos dados médicos e os requisitos que devem estar presentes em um sistema de informação para que ele possa ser certificado como um S-RES.

5.2 Características de Dados Médicos

Relatos históricos revelam que há muito tempo o registro de dados médicos vem sendo empregado. Hipócrates solicitava que os médicos registrassem por escrito seus feitos tanto para

analisar a evolução de uma doença quanto para indicar suas possíveis causas. Os registros eram realizados em ordem cronológica e até o início do século XIX as anotações dos médicos eram baseadas em fatos vistos, sentidos e escutados. A invenção do estetoscópio, por Laennec em 1816, e o surgimento de outros instrumentos contribuíram significativamente com as técnicas de diagnóstico, complementando os registros médicos com novas terminologias desenvolvidas para representar fatos percebidos com o auxílio desses instrumentos (Bemmel et al., 1997).

Uma reflexão sobre as atividades médicas resulta na percepção de que a obtenção dos dados e sua interpretação são fundamentais em todo o processo de tratamento de saúde. O conhecimento provindo da interpretação dos dados proporciona um melhor entendimento dos problemas do paciente e um melhor diagnóstico, fornecendo base para a tomada de decisão pelo médico e para a aplicação de tratamentos.

Para conhecer melhor as características dos dados médicos é interessante entender como o processo de criação do conhecimento médico se desenvolve. Em relação à prática clínica, o médico realiza uma investigação semiológica de modo a coletar os dados relevantes sobre o paciente. Os dados são interpretados de acordo com o conhecimento existente e juntamente com as evidências médicas disponíveis, possibilitam ao profissional tomar decisões de modo mais adequado (Massad et al., 2003).

Assim, um dado médico é definido como qualquer observação sobre um paciente, como a pressão sanguínea, a temperatura, o histórico de uma doença entre outros. Os dados médicos também podem ser vistos como uma composição de quatro elementos (Shortliffe and Cimino, 2006):

- o paciente em contexto;
- a variável observada no paciente;
- o valor da variável observada;
- o momento em que ocorre a observação.

De acordo com a literatura é possível ainda caracterizar os dados médicos de acordo com diversos aspectos, entre eles (Raś and Dardzinska, 2009):

Heterogeneidade: a área médica faz uso de dados provindos de diversas fontes que contém diferentes tipos de dados. Por exemplo, valores de sinais bioelétricos, respostas qualitativas de questionários, medidas numéricas, como peso e altura, textos descritivos de fatos gerados, por médicos ou enfermeiros, entre outros. Os dados também podem ser obtidos de maneira subjetiva, tal como a descrição de um problema pelo paciente, ou objetiva, como a detecção de problemas por meio de algum exame;

Granularidade: os dados utilizados na área médica também se apresentam em diferentes granularidades, sendo classificados como dados brutos, informação processada e co-

nhecimento, como representado na Figura 5.1. A abrangência de diferentes níveis de granularidade por sistemas computacionais é uma tarefa complexa e, caso esteja presente, deve ser claramente especificada durante a modelagem do sistema;

Padronização: a falta de padrões e terminologias bem definidas corroboram com a presença de informações dúbias ou imprecisas, tal como o conceito de saúde cuja definição não apresenta consenso comum.

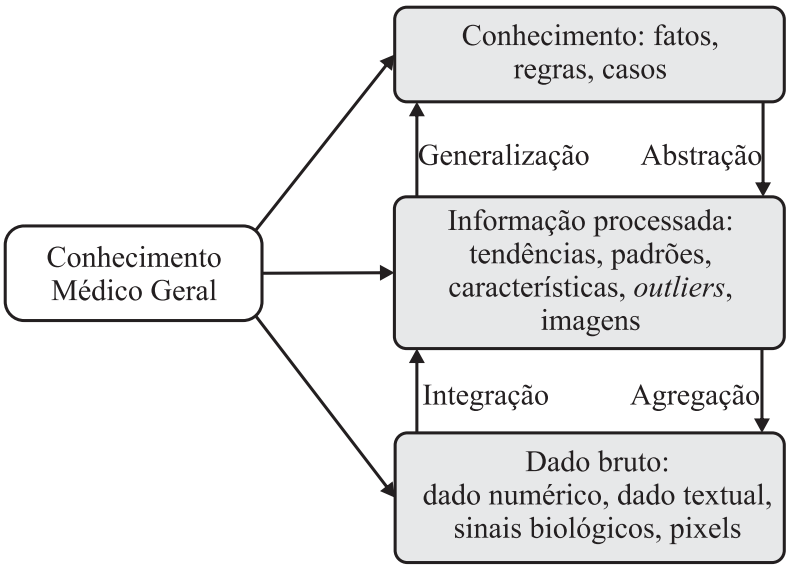

Figura 5.1: Granularidade de dados médicos (modificado de Raś and Dardzinska (2009)).

Para que seja possível representar os dados médicos de maneira consistente e confiável pelos sistemas de informação, órgãos específicos de regulamentação criaram diversas normas que visam monitorar os requisitos mínimos necessários para considerar um sistema computacional como adequado para o gerenciamento de dados médicos. Esse assunto é abordado na sequência.

5.3 Legislação para Sistemas de Registro Eletrônico em Saúde

Na área da saúde os fatos e os dados relacionados ao atendimento médico dos pacientes são registrados em um **prontuário médico**. Este documento é definido pelo Conselho Federal de Medicina[1] (CFM) como um documento único composto por um conjunto de informações e imagens sobre a saúde de um paciente, geradas a partir da assistência médica prestada a ele, possibilitando a continuidade do tratamento de um indivíduo por diferentes profissionais. Esse documento é de caráter legal, sigiloso e científico, sendo o seu zelo de responsabilidade do médico e da instituição em que está presente (Conselho Federal de Medicina, 2002b).

[1] http://www.cfm.org.br

Em relação à propriedade do documento, o elemento físico e seu armazenamento é de responsabilidade da instituição que o detém, porém a informação contida no prontuário é pertencente ao respectivo paciente e uma cópia pode ser solicitada por ele ou por seu responsável legal (Conselho Federal de Medicina, 2007).

Devido a grande quantidade de informações geradas no Brasil por procedimentos médicos, por exemplo, cerca de 360 milhões de consultas médicas por ano, torna-se inviável o armazenamento da grande quantidade de papel acrescida aos prontuários (Conselho Federal de Medicina, 2002a). Porém, como prevê o parecer 30/02 e a resolução 1.821/07 do CFM, o prontuário médico em papel deve ser obrigatoriamente armazenado pelo prazo mínimo de 20 anos a partir do último registro de assistência ao paciente. Após esse período o documento deve ser microfilmado, de acordo com as normas da legislação arquivística, definidas pela Lei 5.433/68 e pelo Decreto 1.799/96, ou digitalizado e assim mantido por tempo indeterminado. Posteriormente a execução desse procedimento o suporte em papel do prontuário pode ser eliminado (Conselho Federal de Medicina, 2002a; Conselho Federal de Medicina, 2007).

Para armazenar os prontuários médicos em formato digital foram definidas diversas normas que estabelecem os requisitos visando à garantia e a segurança dos dados. As normas estão contidas na resolução 1.821/07 do CFM que também estabelece critérios técnicos para o uso de Prontuário Eletrônico do Paciente (PEP), por meio de sistemas de informação específicos, não sendo necessário o armazenamento do prontuário em papel desde que os regulamentos da resolução sejam seguidos.

O PEP apresenta diversas vantagens frente ao uso do prontuário em formato de papel, devido às funcionalidades que podem ser executadas por meio de sistemas computacionais. Entre elas podem ser destacadas (Wechsler et al., 2003; Massad et al., 2003):

- Possibilidade de acesso remoto por diferentes profissionais simultaneamente;
- Legibilidade da informação quando os dados são digitados;
- Apresentação dos dados pode ser personalizada;
- Integração a outros sistemas computacionais;
- Pesquisa facilitada.

Além dos aspectos citados, pode-se acrescentar o fato de que um sistema computacional auxilia no processo de extração e preparação de dados para a aplicação de métodos computacionais de descoberta de conhecimento em bases de dados, como é o caso do processo de Mineração de Dados.

O prontuário médico é composto por um formato muito variado de dados, exigindo do sistema de informação um comportamento diferenciado para lidar com as informações. Como exemplo podem ser citados os resultados de exames laboratoriais, que podem assumir formas

variadas, os exames de imagem, como radiologia e tomografia computadorizada, as observações clínicas em forma de texto livre e os elementos da anamnese, que podem ser definidos em tópicos, entre outros elementos que representam a variabilidade no formato dos dados (Massad et al., 2003).

Para que o PEP possa substituir a utilização do prontuário médico em papel, o CFM expediu a resolução 1.821/07 que compreende diversas orientações e decisões importantes para a área de informática médica, entre as quais estão (Conselho Federal de Medicina, 2007):

- A aprovação do **Manual de Certificação para Sistemas de Registro Eletrônico em Saúde (MCSRES)**, desenvolvido por meio da parceria entre o CFM e a Sociedade Brasileira de Informática em Saúde (SBIS)[2];
- Os arquivos digitais dos prontuários devem ser gerenciados por *software* específico que possua como funcionalidades mínimas:
  - A utilização de uma base de dados adequada para o armazenamento dos dados digitalizados;
  - A capacidade de organizar a informação e a possibilidade de efetuar pesquisa simples e eficiente;
  - A necessidade de estar de acordo com os requisitos do **Nível de garantia de segurança 2 (NGS2)** definidos no MCSRES;
- Os sistemas de informação que obedecem aos requisitos do NGS2 podem substituir a utilização em papel do prontuário médico;
- O CFM e a SBIS tornam-se certificadores dos sistemas de informação, expedindo selos de qualidade dos *software* que estejam de acordo com as especificações do MCSRES.

Como mencionado, o MCSRES foi desenvolvido pela parceria entre CFM e SBIS e surgiu em resposta a diversas consultas sobre a legalidade da utilização dos sistemas de informação para o gerenciamento e a transmissão dos dados no atendimento em saúde. O intuito deste manual é oferecer um processo de certificação para os S-RES baseando-se em conceitos e padrões nacionais e internacionais de informática em saúde (Leão et al., 2009).

Juntamente com outros tipos de sistemas, o PEP se enquadra na categoria de S-RES que compreende todos os sistemas para gerenciamento de informações de um registro eletrônico em saúde. Atualmente, os S-RES que podem ser submetidos à certificação são os que se encaixam em três categorias (Leão et al., 2009):

Assistencial ambulatorial: são S-RES que gerenciam informações sobre o processo de assistência ambulatorial;

[2] http://www.sbis.org.br

GED: são sistemas que pertencem à categoria de Gerenciamento Eletrônico de Documentos (GED), compostos por sistemas destinados ao armazenamento e a visibilização de documentos de saúde;

TISS: são sistemas que adotam o padrão de comunicação denominado Troca de Informação em Saúde Suplementar (TISS), criado pela Agência Nacional de Saúde Suplementar[3] (ANS) objetivando a padronização do registro e da troca de dados entre operadoras de planos privados de assistência à saúde e prestadores de serviços de saúde. Esse padrão é descrito pela resolução normativa 153/07 da ANS.

Os S-RES dessas categoriais podem ser submetidos à certificação da SBIS que contempla dois níveis de segurança: o Nível de Garantia de Segurança 1 (NGS1) e o NGS2, citado anteriormente. Para que o S-RES seja certificado ele deve adequar-se pelo menos ao NGS1. Porém, para considerar o uso do prontuário médico somente em meio eletrônico, o sistema deve adequar-se ao NGS2, cujos requisitos consistem basicamente em estar de acordo com o NGS1 além de possuir funcionalidades para trabalhar com certificados digitais de acordo com o padrão de Infraestrutura de Chaves Públicas Brasileiras (ICP-Brasil) (Conselho Federal de Medicina, 2007). O S-RES também deve ser descrito como de acesso local, podendo ser acessado por somente um usuário, ou de acesso remoto, no caso de permitir o acesso por meio de rede de computadores (Leão et al., 2009).

Assim, a utilização de sistemas certificados com o NGS1 exige que o prontuário seja impresso e assinado pelo responsável, sendo obrigatório seu armazenamento neste formato. O armazenamento do prontuário em formato de papel é necessário, pois este nível de segurança não exige a utilização de certificados digitais, sendo esta a principal diferença em relação ao NGS2

O MCSRES contém a relação de requisitos necessários, bem como o detalhamento desses, para cada categoria de S-RES e também as exigências para os níveis de segurança NGS1 e NGS2. Na Tabela 5.1 são apresentadas as características necessárias a cada nível de segurança (Leão et al., 2009).

Todo o processo para certificação é descrito pelo MCSRES, disponível no *website* do SBIS[4]. A validade da certificação tem a duração de dois anos a partir de sua emissão ou um ano após a publicação de um novo MCSRES pela SBIS. A última versão do MCSRES é a 3.3 datada de maio de 2009. Essa certificação é realizada sobre uma versão específica do S-RES e as alterações no *software*, por exemplo, pela correção de problemas ou melhoria de funcionalidades, podem ser assim classificadas (Leão et al., 2009):

Pequenos ajustes: compreendem as modificações que não afetam os requisitos necessários à certificação;

[3] http://www.ans.gov.br
[4] http://www.sbis.org.br/

Tabela 5.1: Requisitos dos níveis para garantia de segurança da certificação para S-RES.

Nível de Segurança	Requisito
NGS1	- Controle de versão do *software* - Mecanismos de identificação e autenticação dos usuários - Mecanismos para controle de sessões de utilização dos usuários - Mecanismos para autorização e controle de acesso por usuários - Disponibilidade do registro eletrônico de saúde - Mecanismos para comunicação remota segura - Segurança de dados - Suporte para auditoria - Documentação - Registro temporal para auditoria - Mecanismos para notificação de ocorrências, como problemas e sugestões
NGS2	- Certificação digital - Assinatura digital - Autenticação de usuário utilizando certificado digital

Ajustes relevantes: compreendem as modificações que ofereçam risco de afetar os requisitos necessários à certificação.

Assim, ao realizar modificações no S-RES os desenvolvedores podem solicitar a extensão da certificação também para a nova versão por meio do preenchimento de ficha específica, conforme orientações contidas no MCSRES. Caso a nova versão seja composta por ajustes relevantes, o S-RES deve ser submetido a um novo processo de auditoria (Leão et al., 2009).

Recentemente, por meio da Portaria 188 de 15 de março de 2012, o Ministério da Saúde tornou público o Regimento Interno do Comitê de Informação e Informática em Saúde (COINFO/MS), sendo este um órgão com função diretiva, normativa e fiscalizadora das atividades relacionadas aos sistemas de informação e informática em saúde, como descrito pelo Art. 1° do Anexo contido na respectiva Portaria (Brasil, 2012). Esta medida reflete o grande impacto da informática sobre a área da saúde bem como a preocupação do Estado com medidas que assegurem a qualidade e a segurança dos sistemas computacionais e da informação relacionada a saúde.

5.4 Considerações Finais

Como apresentado neste capítulo, os dados médicos possuem fundamental importância em todo o processo de cuidado da saúde de um paciente. Além disso, os registros médicos constituem recursos que possibilitam o desenvolvimento da ciência, fator que agrega motivação para o desenvolvimento deste trabalho.

Assim, um sistema de informação adequado para o registro de dados médicos pode contribuir de diversas formas significativas. Entre elas podem ser citadas a organização, o gerenciamento e a criação de uma base de dados robusta que pode ser utilizada tanto para tarefas de

caráter médico e administrativo quanto para atividades de pesquisa.

Porém, para utilizar um S-RES como substituto do registro em papel, como o PEP, é preciso assegurar que o sistema atenda aos requisitos mínimos que garantam a confiabilidade e a segurança dos dados, conforme descrito no MCSRES. A certificação aplicada pela SBIS assegura que o sistema certificado esteja de acordo com especificações que foram desenvolvidos por pesquisas que analisaram padrões nacionais e internacionais de informática em saúde.

Além de orientar os desenvolvedores em relação ao processo de certificação, o MCS-RES serve também como guia para a especificação de diversas funcionalidades necessárias aos sistemas de informação destinados à área médica, que mesmo sem certificação constituem ferramentas de grande importância para o uso em diversas aplicações. Como exemplo, pode-se citar a pesquisa interdisciplinar envolvendo a computação e a medicina, por meio da aplicação de processos de extração de conhecimento ou padrões interessantes, como a MD.

No Capítulo 6 é apresentado o estudo de caso desenvolvido neste trabalho, no qual estão presentes alguns conceitos e técnicas abordadas neste capítulo.

Capítulo 6

Estudo de Caso

6.1 Considerações Iniciais

Como mencionado, este trabalho tem o objetivo de estudar a aplicação de métodos que auxiliem na garantia da Qualidade de Dados, desde o início do desenvolvimento de um sistema computacional médico. Para atingir este objetivo foi realizado um estudo de caso que envolve a interdisciplinaridade entre as áreas da computação e da medicina.

O estudo de caso envolveu, mais especificamente, a área da Coloproctologia na qual são estudadas e tratadas as enfermidades que acometem o cólon. Nessa especialidade o tratamento de diversas doenças é realizado por procedimentos cirúrgicos, como doença de Crohn, doença hemorroidária, câncer, entre outras (Valarini and Campos, 2008; Torres Neto et al., 2008; Martins et al., 2009; Prudente et al., 2009; Queiroz et al., 2010; Oliveira et al., 2010). Os casos cirúrgicos requerem muitos dados sobre diversos aspectos dos pacientes, de modo que o médico seja capaz de definir o procedimento mais adequado para cada situação. Entre os aspectos envolvidos citam-se o histórico de enfermidades do paciente, o diagnóstico e os resultados de exames. Além disso, um acompanhamento pós-operatório é muito importante para determinar a eficácia do procedimento aplicado ao paciente, bem como a evolução de seu quadro clínico.

Atualmente, no Serviço de Coloproctologia da FCM – UNICAMP é utilizado um sistema desenvolvido com a ferramenta *Microsoft Access*[1] para armazenar digitalmente os dados relacionados aos procedimentos cirúrgicos realizados neste serviço. Porém, esse sistema, neste trabalho denominado de sistema legado, não possui uma estrutura completamente adequada para representar corretamente os fatos relacionados a uma cirurgia, além de possuir limitações e problemas de modelagem que influenciam a qualidade dos dados nele armazenados.

Para contribuir com o gerenciamento desse grande volume de informação proveniente dos períodos de pré, intra e pós-operatório, neste estudo foi desenvolvido um **Protótipo para Gerenciamento de Dados de Cirurgia Coloproctológica** (PGDCC), em uma parceria entre o LABI – UNIOESTE e o Serviço de Coloproctologia da FCM – UNICAMP. Esse protótipo foi

[1] http://office.microsoft.com

desenvolvido visando oferecer uma estrutura computacional que centralizasse os dados necessários às atividades médicas e preparar uma base de dados robusta com vistas à QD, contribuindo para aplicação de processos como a Mineração de Dados e para a utilização dos dados em pesquisas científicas.

Neste capítulo são discutidos o processo de desenvolvimento empregado no estudo de caso e os procedimentos realizados durante a criação do novo PGDCC.

6.2 Sistema Legado

O *Microsoft Access* consiste em uma ferramenta computacional composta por um SGBD próprio e recursos para a criação de formulários, relatórios e interfaces gráficas de utilização para o usuário. Essa ferramenta apresenta-se como uma alternativa para a criação rápida e dinâmica de aplicações simples que não necessitam gerenciar um grande volume de dados. Porém, essa ferramenta é proprietária, sendo necessária a aquisição de licenças para seu uso, e apresenta limitações que não favorecem a sua escolha como ferramenta para o desenvolvimento de sistemas de grande porte. Entre essas limitações citam-se[2]:

- Utilização restrita a sistemas operacionais *Microsoft Windows*;
- Limitação da quantidade de usuários que podem realizar acessos simultâneos;
- Não pode ser acessado via *Web*;
- Não oferece mecanismos de segurança robustos.

Apesar das limitações, esta ferramenta foi utilizada para criar uma estrutura na qual fosse possível realizar o cadastro de dados provenientes de procedimentos cirúrgicos para utilizá-los posteriormente como recursos para pesquisas. Neste trabalho, esse sistema legado foi analisado e sua estrutura de dados foi mapeada, o que possibilitou a identificação do conjunto de informações que eram coletadas. Essa análise resultou na descoberta de falhas na modelagem do sistema legado, além de eventuais problemas nos dados cadastrados.

Na Figura 6.1 é apresentada a tela inicial do sistema legado, na qual é descrita a identificação do sistema e são disponibilizados dois botões que fornecem acesso ao cadastro de pacientes ou saída do sistema. Ao escolher a opção **Cadastro de Pacientes**, o usuário é direcionado a uma tela, na qual são apresentadas as informações básicas de um paciente, como representado na Figura 6.2, contendo dados fictícios. Para realizar uma pesquisa ou navegar entre os registros do sistema, é disponibilizada na parte inferior da tela uma barra auxiliar contendo alguns botões de navegação e um campo para digitar o nome do paciente procurado.

[2]http://www.sql-programmers.com/DisadvantagesofAccess.aspx

A partir da tela **Cadastro de Pacientes** é possível ainda ter acesso às outras partes do sistema por meio de botões localizados na seção de **Informações Clínicas**. Cada uma das telas acessadas por esses botões, possibilita o cadastro de dados provenientes de etapas do acompanhamento de um paciente que será submetido a um procedimento cirúrgico. As opções exibidas referem-se ao período pré-operatório (Exames Pré-Operatórios), ao período intra-operatório (Cirurgia) e ao período pós-operatório (Evolução Hospitalar, Evolução Ambulatorial e Patologia), bem como permitem o cadastro de recorrências cirúrgicas (Reoperação).

Figura 6.1: Tela de entrada do sistema legado.

Figura 6.2: Tela de cadastro de pacientes no sistema legado.

No total, o sistema legado é composto por 108 atributos organizados em sete telas que caracterizam diversos aspectos do paciente. Os atributos também são distribuídos em sete ta-

belas, cujo relacionamento é apresentado na Figura 6.3. A partir do cadastro de um paciente, o sistema permite realizar o registro dos seguintes dados:

- Exames pré-operatórios para um procedimento cirúrgico;
- Uma ocorrência cirúrgica;
- Um ou mais registros de evolução hospitalar;
- Um ou mais registros de evolução ambulatorial;
- Um ou mais registros de patologia;
- Um ou mais registros de reoperação.

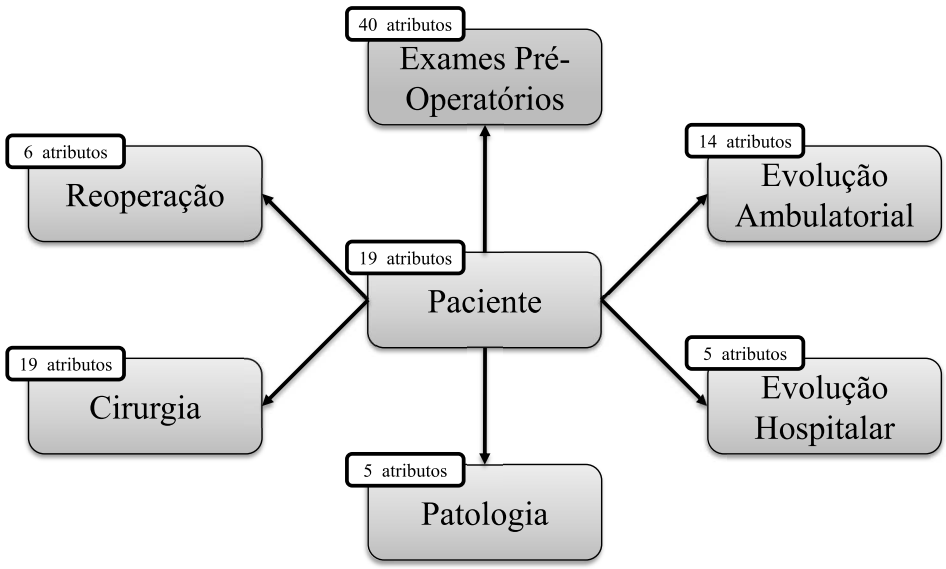

Figura 6.3: Diagrama de relacionamento entre as tabelas do sistema legado (modificado de Lee et al. (2012)).

Como mencionado, a análise do sistema legado possibilitou a identificação de limitações e falhas na modelagem do sistema que podem comprometer a segurança e a qualidade dos dados nele cadastrados, além de não contemplar informações que poderiam ser importantes tanto para pesquisas quanto para aplicação do processo de MD.

Por se tratar de um aplicativo desenvolvido em *Microsoft Access*, a estrutura e a base de dados do sistema legado estão armazenadas em um único arquivo de fácil acesso a todos os usuários que utilizem o respectivo computador. Este arquivo pode ser facilmente copiado para discos externos, como *Pendrives*, e executado em quaisquer computadores que possuam o *software Microsoft Access*. Ainda em relação à segurança, assim como os demais arquivos de um computador, sua estrutura está vulnerável à ação de *software* maliciosos, podendo o arquivo ser corrompido ou até mesmo ter os dados copiados indevidamente.

Ao iniciar o aplicativo, é possível notar a ausência de mecanismos de controle de acesso, como identificação de usuário e senha, fator que agregaria um nível de segurança maior para as informações do paciente, restringindo seu acesso a usuários autorizados.

A abordagem teórica proposta por Wand and Wang (1996) para a definição de QD considera que um sistema de informação é a representação de um sistema do mundo real. A modelagem do sistema legado não reflete, de modo completo, a realidade de uma ocorrência cirúrgica, estando em desacordo com os conceitos de QD. Durante o acompanhamento médico, um paciente pode passar por um ou mais procedimentos cirúrgicos que necessitam de informações sobre diversas características suas, tal como os exames pré-operatórios. Utilizando o sistema legado é possível realizar o cadastro de somente uma cirurgia por paciente e um conjunto de exames pré-operatórios, não sendo esta uma representação adequada. Para minimizar esse problema é possível realizar o cadastro de uma reoperação, porém, os atributos presentes no cadastro de reoperação são limitados frente aos atributos do cadastro de cirurgia, como representado na Figura 6.4 na qual as telas **Cirurgia** e **Reoperação Tardia** são apresentadas.

Outra limitação do sistema legado consiste na ausência de mecanismos que auxiliem a evitar o cadastro em duplicidade de um mesmo paciente, fator que pode repercutir na presença de diferentes registros para um mesmo paciente. Esse problema pode ocasionar o cadastro de casos que apresentem dados contraditórios, anulando a credibilidade dos registros envolvidos. Além disso, devido à falta de atributos específicos em diversas telas, o cadastro de informações complementares é realizado nos campos de observação por meio de linguagem natural. A informação expressa nesse formato é mais propensa a erros e dificulta a análise posterior por meio de processos computacionais como a MD.

Devido aos fatores apresentados, a utilização do sistema legado não permite uma representação completa e consistente dos dados referentes ao acompanhamento médico de um paciente submetido a procedimentos cirúrgicos. Esses fatores depreciam a QD e afetam indiretamente as atividades que fazem uso deles, como pesquisas e métodos de análise computacional.

De modo a solucionar os problemas presentes no sistema legado e formar uma base de dados com qualidade, para ser utilizada em diversas atividades, foi proposto o desenvolvimento de um novo sistema seguindo os preceitos de QD. Esse novo sistema é discutido na seção seguinte.

6.3 Protótipo Proposto: Sistema para Gerenciamento de Dados de Cirurgia Coloproctológica (PGDCC)

Frente às necessidades de ambas as áreas, da saúde e da computação, optou-se por desenvolver um novo sistema que oferecesse uma estrutura mais adequada tanto para o armazenamento dos dados quanto para sua utilização. Sendo assim, as etapas principais do processo de

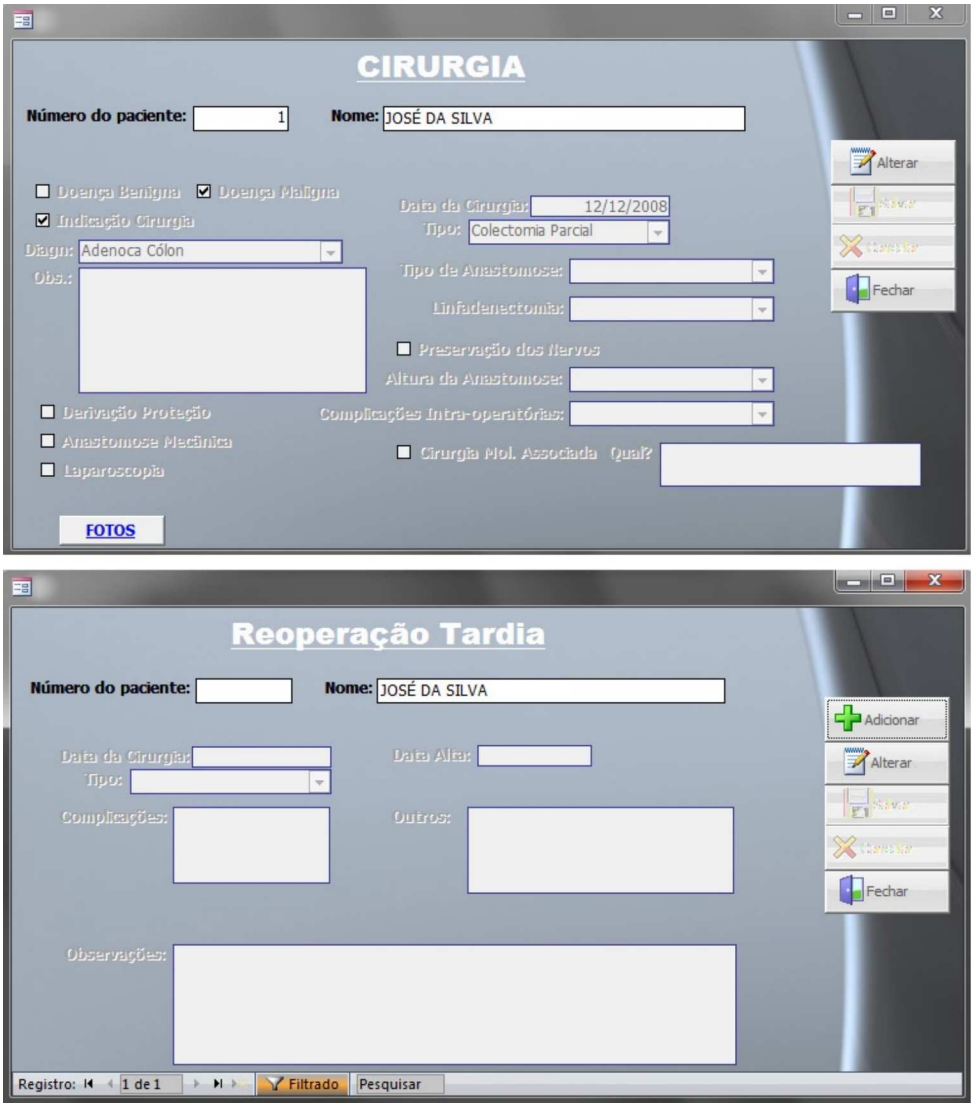

Figura 6.4: Telas de cadastro de cirurgia e de reoperação do sistema legado.

desenvolvimento do PGDCC foram (Lee et al., 2012):

1. Análise do sistema legado;
2. Levantamento dos requisitos;
3. Elaboração de modelos e protótipos de telas;
4. Implementação do protótipo;
5. Avaliação do protótipo.

A análise do sistema legado, discutida na seção anterior, contribuiu com uma visão geral sobre os tópicos que deveriam estar presentes no novo protótipo e com o levantamento dos requisitos. Para ampliar o conhecimento do caso estudado foi realizada uma visita técnica ao

Serviço de Coloproctologia da FCM - UNICAMP que propiciou o entendimento dos procedimentos e processos realizados neste serviço, contribuindo assim para um melhor planejamento dos recursos necessários ao novo sistema. Essa experiência somada às dificuldades percebidas durante a análise do sistema legado serviu de base para a discussão sobre as necessidades que deveriam ser satisfeitas pelo projeto desenvolvido neste estudo de caso.

De modo a entender melhor o domínio da aplicação e para captar as necessidades dos usuários, a etapa de levantamento dos requisitos foi realizada de modo interdisciplinar entre os profissionais da área da saúde e da área computacional. A forte interação entre ambas as áreas se mostrou um fator relevante para um projeto com vistas à QD, pois dessa maneira foi possível melhorar os aspectos que envolvem a utilização do sistema e também auxiliar na estruturação de uma base de dados adequada para futuras aplicações de processos de análise inteligente de dados, como a MD.

A partir dessa interação foi possível definir características desejáveis ao novo sistema, entre elas (Lee et al., 2012):

- Proteção dos dados por meio de mecanismos de controle de acesso, restringindo o uso do sistema a pessoas autorizadas;
- Possibilidade de acesso ao sistema por meio de diferentes computadores em locais distintos;
- Interface gráfica amigável e informação organizada;
- Mecanismos para auxiliar no preenchimento correto dos dados e no monitoramento de registros que representem problemas de QD.

Além das características apresentadas, o levantamento dos requisitos resultou na definição de uma quantidade maior de atributos se comparado ao sistema legado, sendo a maioria de natureza estruturada, fornecendo uma visão mais ampla sobre as características dos pacientes submetidos a procedimentos cirúrgicos. Esses requisitos levantados junto aos especialistas de domínio foram agregados à análise do sistema legado e também à experiência de projetos anteriores, como o Sistema para Gerenciamento de Protocolos de Câncer Colorretal (Lee, da Costa, Ferrero, Coy, Fagundes, Machado and Wu, 2011), o que possibilitou o projeto das entidades que formam o novo protótipo bem como a estrutura de seus relacionamentos, como apresentado na Figura 6.5. Nessa mesma figura é possível observar a quantidade de atributos presentes em cada seção, somando 322 atributos, quantidade aproximadamente três vezes maior que os 108 atributos presentes no sistema legado.

Durante a utilização do sistema legado, muitas informações sobre os pacientes eram registradas em formato de observações, devido à ausência de campos adequados para preenchimento. Com a utilização de atributos estruturados diminui-se a subjetividade no preenchimento do sistema, favorecendo o uso desses atributos por processos computacionais. Outro fator positivo

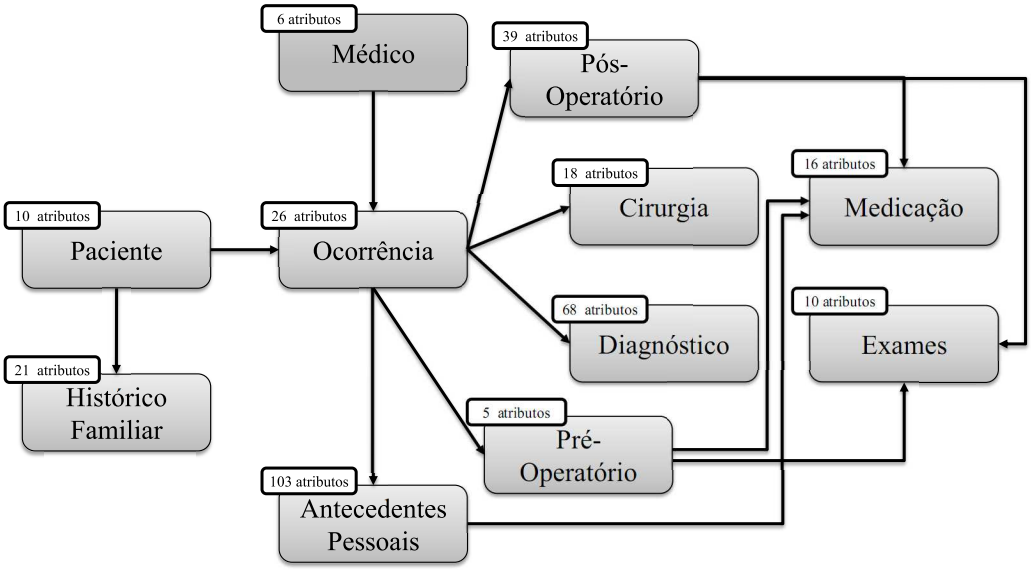

Figura 6.5: Diagrama de relacionamento entre as principais tabelas do PGDCC (modificado de Lee et al. (2012)).

dessa medida consiste na consequente redução de falhas no preenchimento dos dados, como erros de ortografia, já que os possíveis valores dos atributos são pré-definidos.

A estrutura proposta para o novo protótipo tenta representar a realidade que ocorre no ambiente médico no que diz respeito ao acompanhamento de um paciente submetido a uma cirurgia coloproctológica. Por exemplo, a prática médica segue alguns conceitos bem consolidados para guiar o processo de atendimento aos pacientes, organizando a informação de acordo com os elementos clássicos da propedêutica médica (Porto, 1982). Esses mesmos elementos serviram como base para a organização da informação no novo protótipo, tornando mais natural o fluxo da informação para o usuário da área da saúde. Esse conhecimento adquirido por meio da interação com os especialistas resultou, em um primeiro momento, na produção de protótipos de telas que auxiliaram a verificar, de maneira visual, as futuras seções do sistema bem como as informações nelas contidas (Lee, Jung, Silva, Costa and Wu, 2011).

Assim, os campos para cadastro das informações médicas de um paciente estão distribuídos em três sessões principais:

Identificação do paciente: nesta seção são registrados os dados que identificam um paciente de maneira única, como o nome e o código identificador denominado de HC, o tipo sanguíneo e o fator RH bem como a nacionalidade, a naturalidade, a religião, a cor e o sexo;

Ocorrências: nesta seção são registrados os dados médicos relacionados a um único procedimento cirúrgico, porém é possível cadastrar tantas ocorrências quantas forem necessárias para cada paciente. Cada ocorrência cirúrgica também está relacionada a um profissional médico responsável. A partir do cadastro de uma ocorrência é possível registrar

as informações características dos períodos pré, intra e pós-operatórios, como antecedentes pessoais, exames, diagnóstico, procedimento cirúrgico realizado, evolução hospitalar e evolução ambulatorial;

Histórico familiar: nesta seção são registradas informações sobre os dados familiares do paciente, como a quantidade de irmãos, filhos e se o paciente é gemelar.

O projeto de desenvolvimento do sistema também incluiu a criação de protótipos de telas que foram analisados por especialistas da área médica e computacional, auxiliando a determinar a melhor disposição da informação e a estrutura da interface gráfica de utilização. Apesar de não gerar resultados funcionais, essa abordagem possibilita simular a aparência e a estrutura do sistema, facilitando o entendimento por parte dos profissionais da saúde e possibilitando a revisão dos requisitos, caso necessário.

O MCSRES também auxiliou na orientação de requisitos necessários ao desenvolvimento adequado de um S-RES, sendo muitas de suas orientações incorporadas ao projeto, como (Leão et al., 2009):

- Utilização de ferramentas para controle de versão de *software*, possibilitando a verificação do histórico de alterações realizadas no sistema bem como a possibilidade de alternar entre diferentes versões;
- Mecanismos para autenticação de usuários e controle de acesso, protegidos por algoritmos de *hashing*;
- Utilização de um SGBD para armazenamento dos dados;
- Preservação de relacionamento entre dados;
- Registro sobre pessoa para contato e responsável legal na seção de identificação do paciente;
- Suporte ao registro de anamnese, hábitos sociais, histórico familiar, alergias, imunizações, diagnóstico entre outras informações.

Assim, apesar do objetivo do desenvolvimento desse sistema não incluir a substituição do prontuário do paciente em formato de papel, o planejamento do protótipo adotou medidas que podem facilitar a evolução do sistema para processos de certificação, em conformidade com o MCSRES.

Depois de efetuado o planejamento do sistema, iniciou-se a etapa de implementação do protótipo, na qual foram aplicados os conceitos de prototipagem e desenvolvimento incremental propostos pela engenharia de *software*, caracterizando a interatividade entre os domínios da saúde e da computação. Assim, os especialistas acompanharam o desenvolvimento do protótipo em todas as suas fases e auxiliaram nos ajustes necessários ao longo das revisões realizadas sobre o sistema.

A linguagem de programação utilizada para implementação foi a *Ruby* versão 1.9.2, juntamente com o *framework* para desenvolvimento *Web Ruby on Rails* versão 3.0.11, apresentadas no Capítulo 4. Nestas ferramentas estão presentes funcionalidades que agilizam diversas etapas do desenvolvimento de sistemas *Web* e estão em constante desenvolvimento, mantidas por uma comunidade ativa e aberta, na qual é possível encontrar auxílio para a resolução de problemas relacionados a diversos domínios. Para dar suporte ao armazenamento de dados o SGBD *PostgreSQL* versão 9 foi integrado ao *Ruby on Rails*, formando uma estrutura completa para manipulação e armazenamento de dados.

A escolha das ferramentas para desenvolvimento de sistema também mostrou-se uma etapa importante na busca por qualidade. O *framework* de desenvolvimento escolhido adota uma filosofia que favorece o desenvolvimento de um sistema com qualidade, possuindo convenções que orientam a criação de uma base de dados adequada e oferecendo mecanismos pré-definidos que auxiliam em diversas tarefas de verificação e manipulação de dados.

Apesar das características oferecidas pelas ferramentas escolhidas, a fase de implementação revelou-se desafiadora em virtude da intersecção de diversas tecnologias que convergem para a produção de uma aplicação *Web*. Grande parte dessas tecnologias, como a linguagem *HTML*, *CSS* e *JavaScript*, destinam-se ao desenvolvimento de interfaces gráficas de utilização, tópico que recebeu uma atenção especial durante o desenvolvimento do PGDCC, devido ao fato de que interfaces gráficas mal elaboradas e confusas estão entre as causas mais comuns do surgimento de problemas de QD (Maydanchik, 2007).

Como resultado da fase de implementação do PGDCC, pode ser observado na Figura 6.6 a primeira tela do protótipo na qual é solicitado ao usuário que realize a autenticação para poder utilizar as funcionalidades do sistema. Mecanismos de segurança são fundamentais em sistemas médicos, pois garantem o caráter de confidencialidade dos dados dos pacientes, mantendo o sistema em conformidade com a dimensão de QD de Segurança.

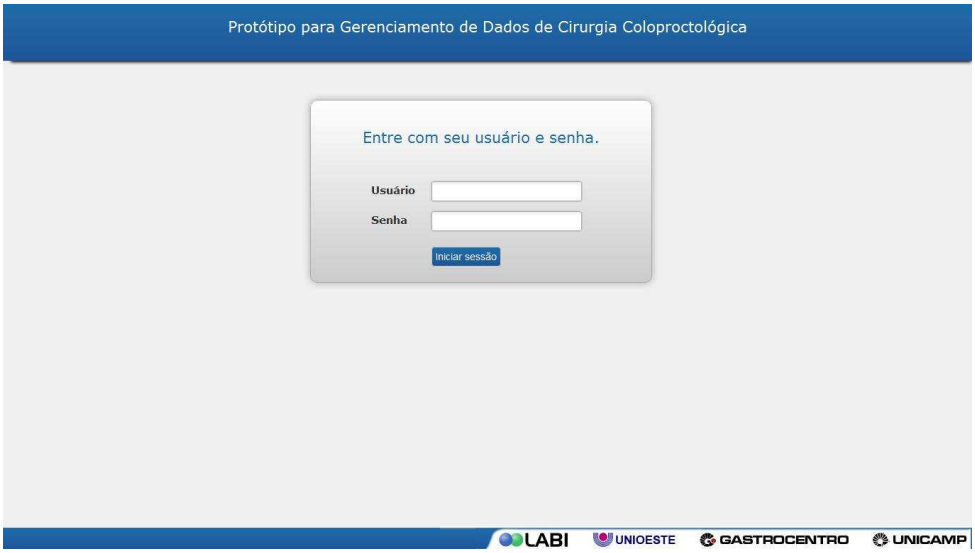

Figura 6.6: Tela de acesso ao sistema.

Após realizada a autenticação no sistema, o usuário é redirecionado para uma página, representada pela Figura 6.7[3], na qual é exibida a relação de pacientes cadastrados. A partir dessa página o utilizador pode acessar as demais funcionalidades do sistema, como ver o registro de um paciente, acessar a relação de médicos por meio da opção presente no menu esquerdo da página ou até mesmo acessar o relatório de problemas de QD por meio da opção **Monitor QD**, também presente no menu esquerdo da página. Para evitar o excesso de informação, a exibição da relação de pacientes está limitada a 10 registros por página. Caso exista uma quantidade maior de cadastros, é disponibilizado, na parte inferior da página, um mecanismo que possibilita a navegação entre os registros não exibidos. Ainda é possível pesquisar de maneira pontual por um paciente, tanto por meio de seu nome quanto pelo seu HC, bastando inserir a informação procurada no campo de pesquisa presente na parte superior da tela.

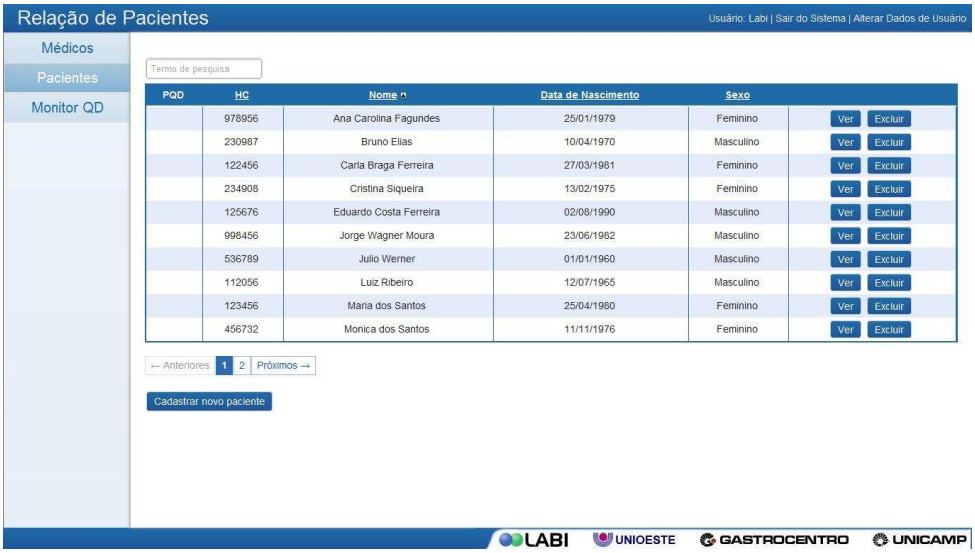

Figura 6.7: Tela exibindo a relação de pacientes do PGDCC.

Na exibição da referida tela é possível notar a presença do botão **Cadastrar Novo Paciente**, cuja ação associada redireciona o usuário para a tela de cadastro de um novo paciente, representada na Figura 6.8. Além do formulário de cadastro, que apresenta o padrão utilizado em todo o sistema, é possível observar a presença de mensagens de erro. Essas mensagens de erro são geradas a partir de um processo de validação executado pela tentativa de armazenar um registro que não está de acordo com as restrições declaradas no modelo da entidade que está sendo cadastrada. As restrições presentes neste exemplo referem-se aos atributos **HC** e **Nome**, cuja presença é necessária para possibilitar o cadastro de pacientes. Esses mecanismos de validação presentes no *framework Ruby on Rails* contribuíram significativamente para o objetivo deste trabalho, sendo discutidos em mais detalhes na Seção 6.4

Na tela de **Relação de Pacientes** (Figura 6.7), ao selecionar o registro de um paciente e acionar a opção **Ver**, o usuário é redirecionado para uma página, representada na Figura 6.9, na qual são exibidas informações sobre as três sessões anteriormente mencionadas: identificação

[3]Todas as informações apresentadas nas telas do sistema são fictícias.

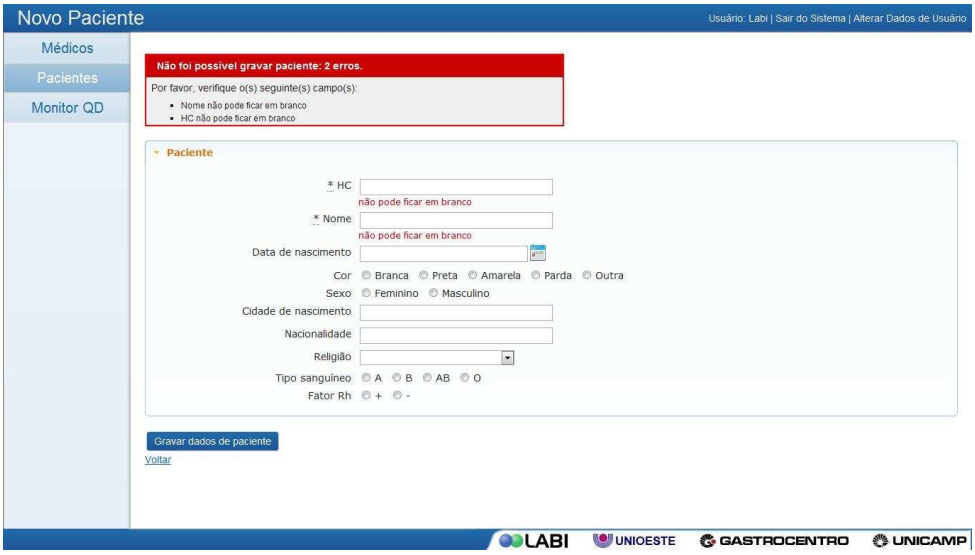

Figura 6.8: Tela de cadastro de pacientes com a exibição de mensagens de erros de validação.

do paciente, ocorrências e histórico familiar. As ocorrências cirúrgicas do paciente são exibidas em formato tabular e ao executar a ação **Ver** de uma determinada ocorrência, é fornecido acesso aos dados dos períodos pré, intra e pós-operatórios da respectiva ocorrência. Essa tela também oferece a opção para realizar o cadastro de novas ocorrências e a partir destas o cadastro do procedimento cirúrgico aplicado ao paciente, como representado na Figura 6.10.

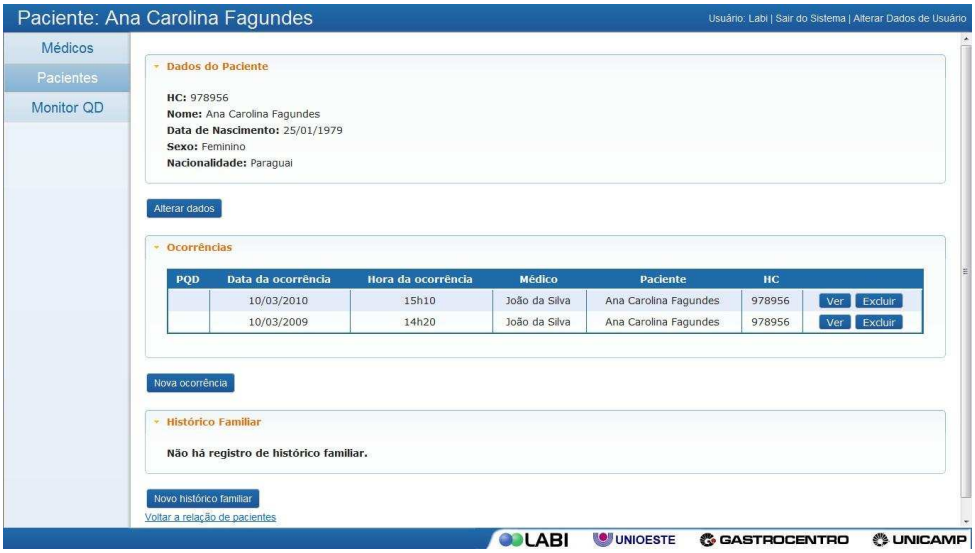

Figura 6.9: Tela de exibição de dados de um paciente.

Na Figura 6.10 é possível destacar algumas das funcionalidades que também estão presentes nas demais sessões do protótipo, como a exibição de listas com opções de valores para os atributos, agilizando o preenchimento dos formulários por parte dos utilizadores e reduzindo a presença de valores redundantes para os atributos, assim como erros ortográficos provindos da inserção de valores textuais. Caso a opção desejada não esteja presente na lista, pode-se cadastrá-la, sem que a sessão seja encerrada, por meio do acionamento de botões próximos a

essas listas que exibem janelas flutuantes nas quais podem ser inseridos os valores desejados. Para evitar o acúmulo de informações na tela, alguns atributos são exibidos após serem ativados por ações específicas, por exemplo, os atributos **Método de Reconstrução Intestinal** e **Observação**, que são exibidos somente se a opção **Sim** de **Reconstrução Intestinal** for marcada. Esta funcionalidade contribui com a melhora da usabilidade do sistema que possui um número elevado de atributos, reduzindo a sobrecarga de informações na interface. Nesta tela ainda é possível observar a exibição de alertas sobre incompletudes e inconsistências, funcionalidade discutida na Seção 6.4.

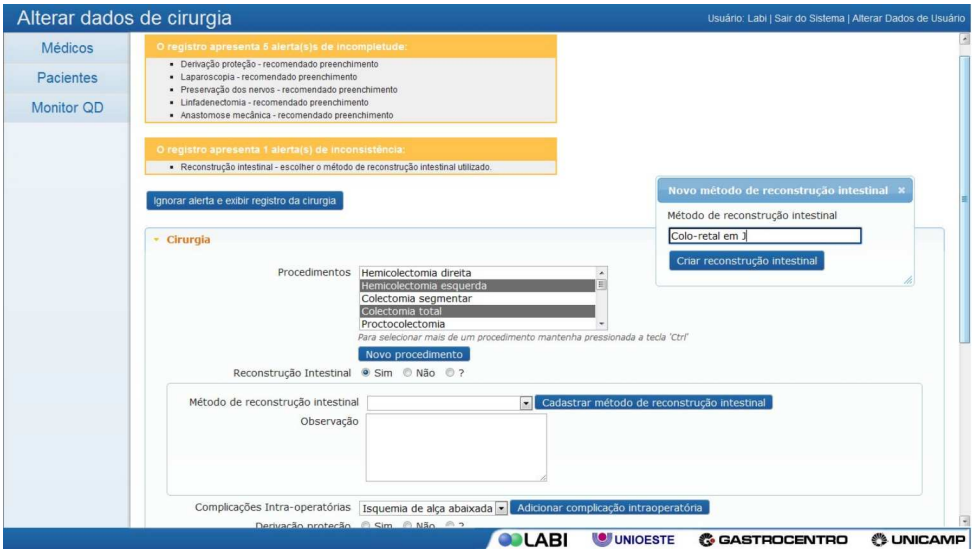

Figura 6.10: Tela de edição de um registro de cirurgia, com exibição de alertas.

Ao final, é possível observar os dados provenientes do cadastro finalizado de uma cirurgia, representados na Figura 6.11[4]. Os dados exibidos pertencem ao período intra-operatório e estão posicionados dentro da sessão de ocorrência, ao lado de informações referentes aos demais períodos de um tratamento cirúrgico, como pode ser observado nas abas exibidas na parte superior da imagem. É possível observar ainda que o sistema oferece suporte para a adição de fotografias da cirurgia, exibindo suas miniaturas e possibilitando ao usuário vê-las em um tamanho maior ao selecionar a imagem desejada.

Apesar do elevado número de atributos que compõem o sistema, nem sempre é possível cobrir todos os aspectos relacionados à realidade médica, sendo necessário muitas vezes que os profissionais de saúde utilizem os campos de observação, em formato texto, em tais situações. Porém, neste trabalho essa necessidade foi minimizada se comparada à utilização do sistema legado. É interessante citar que métodos para o mapeamento, de informação em linguagem natural para formatos estruturados, podem ser usados como apoio para a posterior transformação da informação, contida nos campos de observação, para bases de dados estruturadas (Costa et al., 2010).

[4]Fotos cirúrgicas extraídas das páginas *Web*: http://commons.wikimedia.org/wiki/File:Stomach_surgery_in_Kosovo,_2000.JPEG e http://commons.wikimedia.org/wiki/File:Stomach_surgery,_2000.jpeg.

Figura 6.11: Tela de exibição do registro da cirurgia de uma ocorrência.

Para auxiliar os profissionais de saúde a cadastrar os dados provenientes de pacientes que possuam mais de uma ocorrência cirúrgica foram implementados mecanismos assistentes que realizam o prévio preenchimento de determinados formulários, com base em registros anteriores do paciente. Essa funcionalidade além de otimizar o trabalho do profissional de saúde, bastando que o mesmo modifique os campos necessários, colabora com o adequado armazenamento de dados que já foram preenchidos em uma etapa anterior. Caso haja a presença de erros, os mesmos podem ser verificados por meio de regras específicas definidas no módulo de monitoramento de QD proposto neste trabalho e discutido na seção seguinte.

Durante a utilização do sistema legado houve o cadastro dos dados de muitos pacientes. De modo a não inutilizar o trabalho realizado pelos profissionais de saúde, os atributos do sistema legado, considerando os problemas de projeto identificados, foram mapeados em relação aos atributos do novo protótipo, para posterior auxílio no processo de migração dos dados.

Como mecanismo de prevenção e auxílio na identificação de problemas, neste trabalho foi proposto um processo de monitoramento de QD que visa auxiliar em uma identificação mais ágil de possíveis fatores que causem a degradação do nível de QD, de modo a não comprometer a usabilidade e confiabilidade da informação gerenciada pelo sistema. Tal proposta é discutida na seção seguinte.

6.4 Processo de Monitoramento de Qualidade de Dados

O domínio da aplicação que abrange este estudo de caso contempla muitas informações que são representadas no PGDCC por uma grande quantidade de atributos, entre os quais muitos são interrelacionados ou dependentes uns dos outros. O mapeamento de todas as situações de preenchimento dos dados mostra-se inviável devido ao número elevado de casos possíveis.

Como mencionado no Capítulo 2, um fator que resulta na deterioração da qualidade dos dados consiste em falhas ou faltas no preenchimento da informação pelos usuários. Um método que pode contribuir com a redução de campos não preenchidos contempla a validação dos atributos presentes em formulários de cadastro, persistindo os dados somente após o preenchimento de campos marcados como obrigatórios. Apesar desse método contribuir com o preenchimento dos atributos, muitos usuários podem preencher valores errôneos de modo a burlar o sistema, fator que requer a adoção de medidas que visem a conscientização sobre a importância de um preenchimento correto das informações.

Apesar do método de obrigatoriedade de preenchimento contribuir com a redução de problemas de completude, existem circunstâncias em que a aplicação desta solução não é aconselhada, tal como o caso estudado neste trabalho. O preenchimento dos dados do PGDCC é realizado por diferentes profissionais de saúde, como enfermeiros, residentes e médicos, cujo ambiente de trabalho e a dinâmica do serviço de atendimento à saúde pode muitas vezes impossibilitar, momentaneamente, a conclusão do cadastro de dados de um paciente. Assim, para evitar a perda de dados de um cadastro incompleto, um dos requisitos pertencentes ao protótipo consiste no fato de que os usuários possam armazenar os dados mesmo quando um formulário está incompleto, salvo alguns casos em que informações específicas não podem estar ausentes, como o nome do paciente e seu número de identificação.

Apesar desse método evitar a possível perda de dados, esse procedimento pode elevar o nível de incompletudes e inconsistências da base de dados. Para contornar esse efeito negativo, a proposta de desenvolvimento do PGDCC inclui a implementação de um módulo auxiliar para verificação e marcação de registros que apresentem problemas, de acordo com especificações descritas nos modelos de cada entidade, denominado de **módulo de monitoramento** (Jung et al., 2011). O processo decorrente da utilização do módulo de monitoramento pode ser dividido em três etapas representadas pelos números um, quatro e cinco do fluxo de registro de dados, apresentado na Figura 6.12.

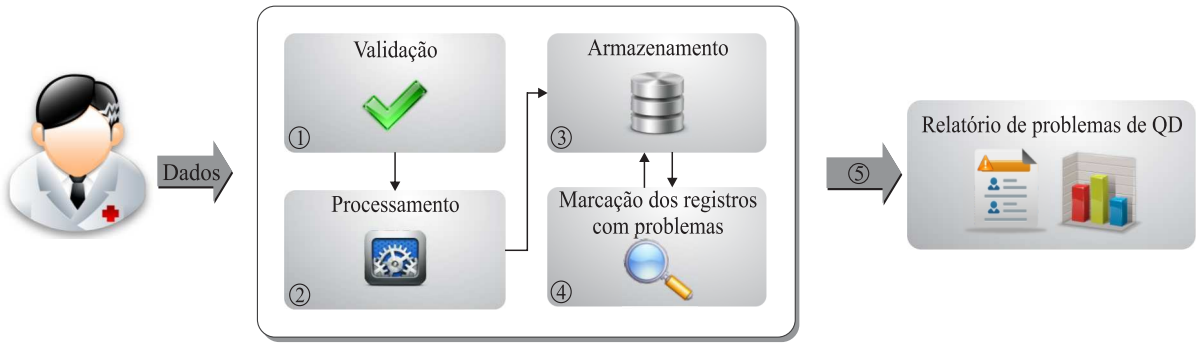

Figura 6.12: Fluxo do processo de registro de dados (modificado de Jung et al. (2011)).

A implementação do módulo de monitoramento faz uso de algumas funcionalidades disponibilizadas pelo *framework Ruby on Rails*. Como mencionado no Capítulo 4, em *Ruby on*

Rails estão presentes recursos que auxiliam na validação[5] dos atributos pertencentes a cada entidade do sistema, sendo estes recursos fundamentais para a primeira etapa do processo de monitoramento. Os mecanismos de validação consistem na declaração de expressões pré-definidas, que podem ser consultadas na documentação do *framework*, cuja sintaxe se assemelha a linguagem natural. Cada expressão representa uma ou mais restrições ou condições de conformidade para que os atributos sejam mantidos na base de dados. Um exemplo de validação de atributo é apresentado na Figura 6.13, cuja sintaxe pertence à linguagem de programação *Ruby*.

```
validates_presence_of :nome, :message => "O nome do paciente deve ser preenchido."
```

Figura 6.13: Validação da presença do atributo nome da entidade **Paciente**.

A formação de cada expressão requer parâmetros que incluem a restrição que será aplicada, os atributos monitorados e a mensagem que será exibida ao usuário caso a restrição seja violada. Em alguns casos particulares, é possível aplicar a validação com base no resultado de uma função na qual podem ser desenvolvidos inúmeros procedimentos. Em relação à Figura 6.13, a restrição estabelecida é a obrigatoriedade da presença do atributo "nome" para que o registro de um paciente seja gravado e, caso a restrição não seja satisfeita, uma mensagem personalizada é exibida ao usuário. Assim, durante a utilização do sistema, quando um usuário realiza o preenchimento de um campo cujo atributo possui validação, ao executar a ação para armazenar os dados o mecanismo de validação é ativado, verificando se o atributo obedece à restrição estabelecida, condição que determinará a sequência da operação. Para melhor representar o conceito explanado, o resultado da execução do mecanismo de validação pode ser observado na Figura 6.8 (Página 64).

Porém, a lógica relacionada à implementação dos mecanismos de validação do *framework Ruby on Rails* não permite salvar os dados que estejam em desacordo com as restrições. Como citado, há situações em que são necessárias validações, porém deve-se permitir ao usuário que os dados sejam armazenados mesmo que as restrições não sejam cumpridas. Para atingir esse objetivo uma das soluções possíveis é aproveitar a funcionalidade já existente no *framework*, que pode ser alterada devido ao fato de que *Ruby on Rails* possui seu código aberto.

Entre as vantagens presentes em projetos de código aberto, em geral, destaca-se a presença de uma comunidade de desenvolvedores ativa, dentro da qual muitos implementam funcionalidades destinadas à solução de problemas específicos. Na comunidade *Ruby on Rails*, muitas dessas funcionalidades são disponibilizadas no formato de *gems*, como mencionado no Capítulo 4. Um exemplo pode ser representado pela *gem* ***validation_scopes***[6], que foi desenvolvida com a finalidade de criar mecanismos de alerta que aproveitem as características das validações presentes no *framework*, porém, possibilitem ao usuário armazenar os dados, mesmo com a presença de um registro que não esteja em conformidade com a validação. Deste modo,

[5] http://guides.rubyonrails.org/active_record_validations_callbacks.html
[6] https://github.com/dasil003/validation_scopes

a funcionalidade vem ao encontro das necessidades do módulo de monitoramento.

A partir do código pertencente à *gem validation_scopes* foram implementadas novas funções que culminaram no módulo de monitoramento. A utilização do módulo de monitoramento tem início a partir da definição e da declaração de escopos específicos nas entidades do protótipo que serão monitoradas. Dentro de cada escopo, que recebe um identificador, devem ser adicionados os atributos que serão avaliados de acordo com uma função de validação associada.

Portanto, cada formulário do sistema pode estar associado a funções de verificação de erros e de alertas. Quando um registro contiver atributos cujos mecanismos de validação acusem erros, o usuário será notificado sobre a presença de tais erros e não poderá armazenar o cadastro antes de corrigi-los. De outro modo, quando o registro incluir alertas, o usuário será informado sobre a presença de tais violações, entretanto poderá armazenar o cadastro, se assim o desejar. Tais mecanismos formam a **primeira etapa** do processo de monitoramento de QD.

Para avaliar o funcionamento do módulo de monitoramento foram consideradas duas dimensões, a da completude e da consistência, sendo que cada dimensão está relacionada a um escopo definido no modelo de uma entidade, no qual são inseridas as funções de validação e os atributos sobre os quais são executadas. Como problemas de completude, foram considerados os campos monitorados que não foram preenchidos. Já como problemas de consistência, foram considerados o não preenchimento de campos interrelacionados, tal como o atributo **Reconstrução Intestinal** que se relaciona com o atributo **Método de Reconstrução Intestinal**. A não definição do segundo atributo, quando o usuário indica que houve reconstrução intestinal, é considerada um problema de inconsistência neste trabalho.

Após a Etapa 01 do processo de monitoramento, que como citado contempla os mecanismos de validação, os dados são processados e armazenados em uma base de dados, atividades essas representadas, respectivamente, pelas Etapas 02 e 03 da Figura 6.12 (Página 67). Após o armazenamento na base de dados, um mecanismo é iniciado para marcar os registros dos pacientes que não estão de acordo com as dimensões de QD avaliadas, auxiliando assim em uma posterior revisão dos registros.

De modo a facilitar a localização do problema por parte dos usuários, no registro de um paciente, foram consideradas duas classificações identificadas por elementos gráficos, como representado na Figura 6.14.

O elemento “A” da Figura 6.14 identifica que uma entidade possui um **problema direto**, ou seja, o modelo de sua entidade possui atributos que violaram as especificações de uma ou mais dimensões de QD, de acordo com os escopos definidos. Por sua vez, o elemento “B” da Figura 6.14 identifica um **problema indireto**, ou seja, a entidade marcada não possui problemas diretamente em seus dados, porém relaciona-se com outras entidades que possuem problemas. Desse modo, o usuário do sistema poderá facilmente navegar entre os registros de maneira pontual para localizar a origem do problema. Na Figura 6.15 é possível observar a tela do

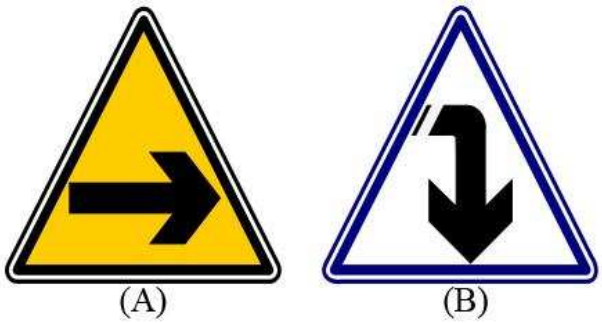

Figura 6.14: Elementos gráficos de auxílio à localização de problemas. (A): problema direto; (B) problema indireto.

protótipo na qual é exibida a relação de pacientes, cuja presença do sinal gráfico na coluna **PQD** identifica os registros que apresentam problemas diretos ou indiretos.

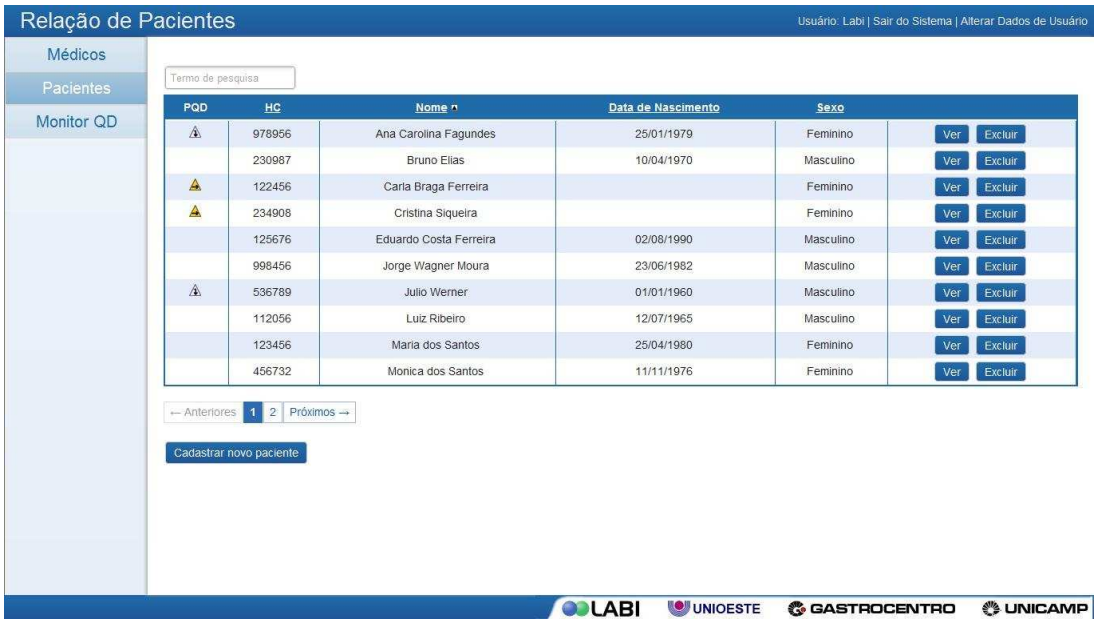

Figura 6.15: Tela de exibição da relação de pacientes com destaque para registros com problemas de QD.

Para realizar a marcação dos registros do protótipo de acordo com os problemas existentes, torna-se necessário verificar se há violação das restrições após a criação, atualização ou remoção de cada entidade monitorada. Assim, após realizada a respectiva operação sobre a entidade monitorada, o estado das entidades ancestrais deve ser atualizado, fazendo alusão à estrutura de uma árvore.

De modo a avaliar o conceito proposto, o conjunto de entidades monitoradas foi, inicialmente, composto pelas entidades **Paciente**, **Ocorrência** e **Cirurgia**. A arquitetura da aplicação propõe que um **Paciente** pode estar relacionado a uma ou várias **Ocorrências**, e cada **Ocorrência** relaciona-se com uma **Cirurgia**. Portanto, quando o cadastro de uma **Cirurgia** apresenta problemas em seus dados, seus registros ancestrais de **Ocorrência** e **Paciente** são marcados

como detentores de problemas indiretos. Esse conceito é representado graficamente pela Figura 6.16.

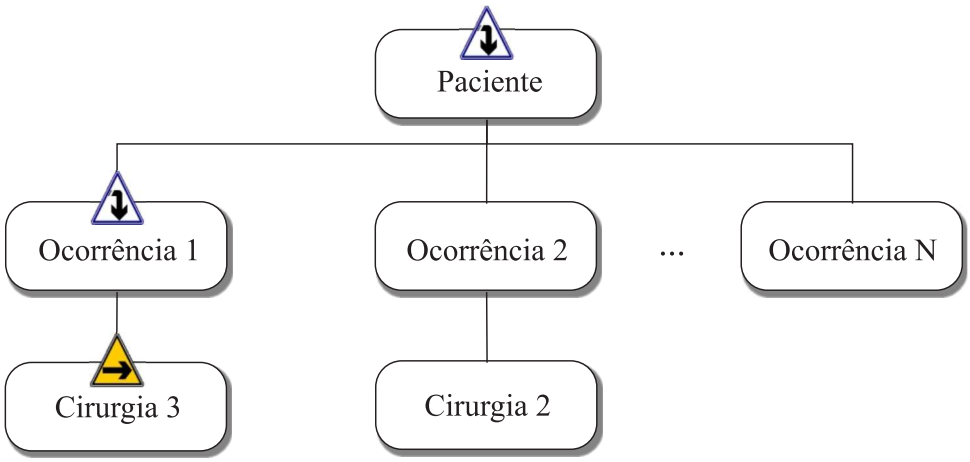

Figura 6.16: Propagação de problemas de QD entre entidades relacionadas.

Para realizar a atualização do estado das entidades e seus ancestrais foi implementado um algoritmo que mantém armazenadas informações sobre todos os registros que possuam tanto problemas indiretos como diretos. A aplicação deste algoritmo contempla a **segunda etapa** do processo de monitoramento de QD. Ele atua recursivamente, sendo executado após operações que envolvam a manipulação de uma entidade monitorada. Seu pseudocódigo pode ser observado no Algoritmo 1.

De maneira a facilitar ainda mais a utilização do sistema, ao abrir para edição um registro que contenha problema direto, uma mensagem é exibida indicando os campos que necessitam de correção. Um exemplo desta funcionalidade pode ser observado na Figura 6.10 (página 65), na qual estão presentes na parte superior da imagem dois quadros relatando os atributos que não estão de acordo com os requisitos dos escopos de incompletude e de inconsistência.

A **terceira etapa** do processo de monitoramento de QD consiste em apresentar ao usuário a síntese dos registros de pacientes marcados com problemas de QD, sendo exibida de maneira tabular e gráfica, como representado na Figura 6.17. Na tabela são apresentados o número de identificação do paciente, seu nome e o tipo de problema relacionado, possibilitando ao usuário ser redirecionado ao registro do paciente por meio de um clique com o *mouse* sobre a respectiva linha na tabela. Outra contribuição apresentada na Figura 6.17 consiste no gráfico de distribuição de dados dos pacientes cadastrados, pelo qual é possível verificar a porcentagem de registros de pacientes que foram acometidos por problemas.

Da mesma maneira como é exibido um relatório contendo os registros de pacientes que possuam problemas diretos e indiretos, a codificação do sistema pode ser adaptada para apresentar os registros que contenham problemas de acordo com as dimensões de QD monitoradas.

Algoritmo 1: Monitoração de Entidades

```
Entrada: self ← entidade_em_contexto
if self.possui_pqd_direto = true then
    // Caso a entidade possua um problema direto, porém, não esteja registrada
    // na base de dados, a operação de registro é executada.
    if self.pqd_criado = false then
        self.criar_pqd_direto
    else
        // Caso o registro da entidade já exista e esteja marcado como problema
        // indireto, altera-se o status para problema direto.
        self.pqd_direto ← true
    end if
    // Executa a atualização do estado dos ancestrais
    self.atualiza_status_pais
else
    // Caso a entidade esteja registrada na base de dados e não possua problema
    // direto, porém, existam filhos com problema direto, altera-se o status para
    // problema indireto.
    if self.pqd_existe = true AND self.possui_pqd_indireto = true then
        self.pqd_direto ← false
    else
        // Caso não existam filhos com problema direto, remove-se o registro dessa
        // entidade da base de dados de problemas de QD.
        self.remove_pqd
    end if
    // Executa a atualização do estado dos ancestrais.
    self.atualiza_status_pais
end if
```

6.5 Considerações Finais

O trabalho desenvolvido neste estudo de caso possibilitou o planejamento de medidas para auxiliar na criação de uma estrutura computacional visando a QD durante o processo de desenvolvimento de um sistema computacional médico.

Neste estudo de caso foi possível aliar os objetivos deste trabalho com as necessidades do Serviço de Coloproctologia da FCM – UNICAMP, que já fazia uso de um sistema legado para registrar digitalmente os dados provenientes de procedimentos cirúrgicos, e do LABI – UNIOESTE. A análise desse sistema legado revelou uma série de fatores que podem comprometer a qualidade da informação e consequentemente sua usabilidade, além de dificultar a correta representação das fases envolvidas no acompanhamento de um paciente submetido a procedimentos cirúrgicos.

A partir da análise do sistema legado, foram identificados os requisitos necessários ao novo sistema junto aos especialistas das áreas da saúde e da computação, culminando no de-

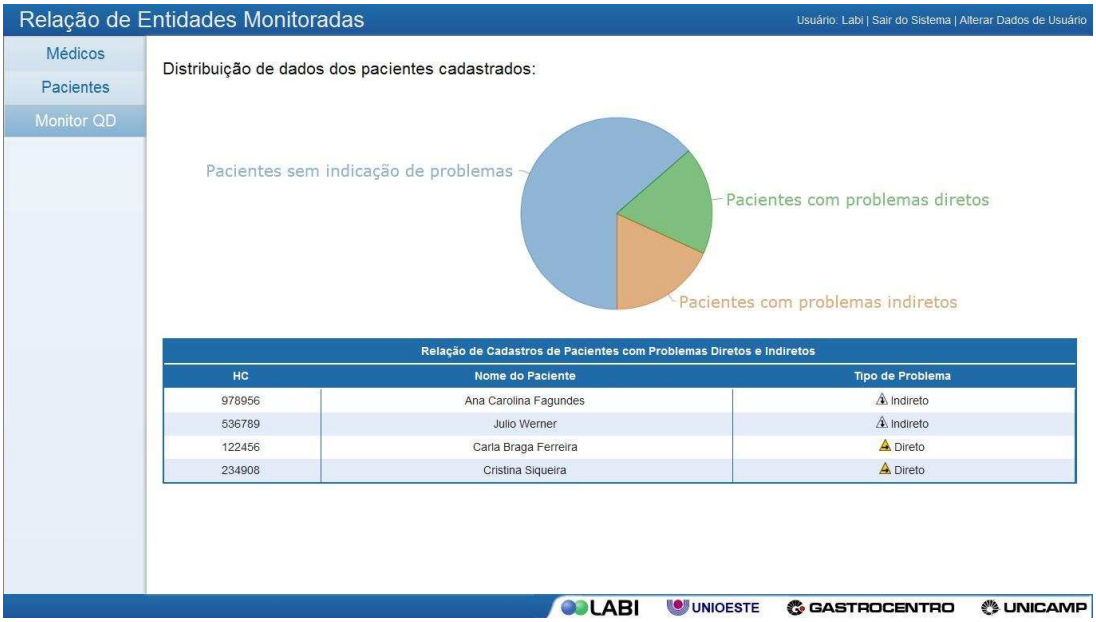

Figura 6.17: Tela do PGDCC contendo a relação de entidades monitoradas.

senvolvimento do novo PGDCC. A forte interação com os especialistas foi considerada como fator predominante na busca por qualidade sob a perspectiva dos usuários. Já a preocupação com QD sob a perspectiva dos dados obteve a contribuição de fatores como o planejamento e a modelagem do sistema com o acompanhamento dos especialistas e as funcionalidades presentes nas ferramentas de desenvolvimento, como a linguagem *Ruby* e o *framework Ruby on Rails*.

Tendo em vista que a preocupação com a QD deve ser contínua, o processo de monitoramento de QD proposto neste projeto auxilia o usuário durante a utilização do sistema, emitindo alertas que indicam a não conformidade dos dados em relação a regras estabelecidas, e possibilitando o acompanhamento do nível de QD dos registros de paciente. Tal procedimento não é comum a sistemas médicos e constituiu uma grande contribuição para com as atividades dependentes do PGDCC.

Neste trabalho também se fez presente a preocupação com a criação de uma interface gráfica interativa e intuitiva, visando à melhoria da experiência de utilização pelos profissionais da saúde. Este aspecto foi alcançado devido a fatores como a organização da informação de acordo com os elementos clássicos da propedêutica médica, grande quantidade de atributos estruturados, campos com opções de resposta pré-definidas e mecanismos que possibilitam ocultar os atributos que não estão sendo utilizados no momento.

De modo a analisar a experiência dos usuários em relação ao contato com o PGDCC, bem como a visão de cada utilizador quanto aos dados gerenciados por meio do sistema, foi realizada uma pesquisa inicial com diversos colaboradores que caracterizou a avaliação subjetiva da QD e do sistema. Tal discussão é abordada no Capítulo 7.

Capítulo 7

Resultados e Discussão

De modo a avaliar a usabilidade do sistema bem como a perspectiva dos usuários em relação à qualidade da informação nele inserida, o PGDCC foi disponibilizado para o acesso e a utilização de colaboradores da área da computação e da área da saúde. Após a realização de um breve tutorial sobre o protótipo, os usuários foram convidados a utilizar o sistema e a responder um formulário de pesquisa, elaborado em conjunto com especialistas, com base em um método já consolidado na literatura.

Essa etapa, chamada de Avaliação Qualitativa de QD, representa um processo importante para a identificação do ponto de vista dos usuários que podem indicar fatores que necessitam de melhorias, não percebidos durante o processo de desenvolvimento. A seguir são descritas as etapas do processo de preparação e aplicação da avaliação, bem como a discussão dos resultados obtidos.

7.1 Avaliação Qualitativa de Qualidade de Dados

Como explanado no Capítulo 2, um dos aspectos que envolvem a avaliação da QD está relacionado à perspectiva dos usuários em relação aos dados oriundos dos sistemas de informação (Wang, 1998; Eppler, 2006). Para avaliar essa abordagem, o método *AIMQ* mostrou-se eficiente quando aplicado a diversas organizações por pesquisadores relacionados ao *MIT Total Data Quality Management* (*TDQM*)[1], programa que possui grande experiência na pesquisa sobre QD.

Como mencionado, um dos recursos utilizados pelo método *AIMQ* consiste no formulário *IQA*, a partir do qual foi derivado o instrumento de pesquisa utilizado neste trabalho. O *IQA* é composto por 65 questões relacionadas a 15 dimensões de QD, possuindo em média quatro a cinco questões por dimensão. Muitas das questões existentes neste formulário são redundantes, de modo a reduzir a subjetividade das respostas dos entrevistados (Lee et al., 2002).

[1] http://web.mit.edu/tdqm

A pesquisa qualitativa adotada neste trabalho visa avaliar a QD e extrair informações sobre os aspectos de usabilidade do sistema de informação, além de algumas características específicas dos entrevistados. Em conjunto com os especialistas da área da saúde e da computação, o *IQA* foi analisado e dele foram extraídas questões que melhor representavam as características relacionadas ao domínio no qual o PGDCC está inserido. Este processo resultou na elaboração de um Formulário para Avaliação Qualitativa do Protótipo (FAQP), apresentado no Apêndice A, cuja estrutura está organizada em quatro seções:

Seção A: é composta pelas questões derivadas do *IQA* que visam capturar a percepção do usuário quanto à qualidade dos dados gerenciados pelo PGDCC;

Seção B: é composta por questões que visam extrair informações sobre a experiência do usuário com o sistema de informação e sua importância em relação a seus objetivos;

Seção C: é composta por questões que caracterizam o entrevistado e sua área de atuação, bem como o tempo despendido no preenchimento do formulário;

Seção D: é composta por um espaço para sugestões e críticas.

A **Seção A** do FAQP é formada por 36 questões relacionadas a 12 dimensões de QD extraídas do *IQA*, sendo cada dimensão composta por três perguntas que visam minimizar a subjetividade do formulário. Como descrito por Lee et al. (2006), cabe à organização identificar as dimensões de QD consideradas importantes no contexto onde será realizada a avaliação. Assim, a seleção das dimensões de QD se deu por meio de um processo de análise e discussão sobre o *IQA* entre especialistas da saúde e da computação. Durante esse processo foram selecionadas as questões que melhor representavam o domínio estudado, sendo algumas adaptadas para o contexto deste trabalho, já que o *IQA* é direcionado a área empresarial.

A dimensão de Representação Concisa foi agregada à dimensão de Interpretabilidade, devido à similaridade dos aspectos investigados pelas duas dimensões. A dimensão da Pontualidade não foi selecionada, pois está relacionada ao grau de atualidade dos dados após uma mudança no mundo real e se esse grau é apropriado para a tarefa em questão. Como o prontuário médico do paciente é a fonte a partir da qual os dados são inseridos no sistema e o prontuário é atualizado durante o atendimento de cada paciente, não há a necessidade de avaliar esta fonte. A dimensão da Reputação também não foi selecionada, pois avalia a fonte da informação que como mencionado consiste no prontuário do paciente, não sendo este o objeto de estudo.

Sendo assim, na Tabela 7.1 são representadas as questões que formam a **Seção A** do FAQP, agrupadas de acordo com a dimensão de QD a que representam. Algumas questões foram ordenadas aleatoriamente no FAQP, de maneira a evitar a proximidade de questões redundantes.

Para realizar a avaliação qualitativa do PGDCC foram convidados diversos colaboradores da área da saúde e da computação, possibilitando obter uma melhor visão sobre aspectos técnicos percebidos pela área da computação e também a usabilidade, a adequação, a organização e

Tabela 7.1: Questões relacionadas às dimensões de QD presentes no questionário de avaliação qualitativa.

Dimensão de QD	Questões
Acessibilidade	A informação é facilmente recuperável A informação é facilmente acessível A informação é rapidamente obtida
Quantidade Adequada	A quantidade de informações é suficiente para nossas necessidades A quantidade de informação é alta A quantidade de informação é baixa
Credibilidade	A informação é confiável A informação é de credibilidade duvidosa A informação é verossímil
Completude	A informação é completa para nossas necessidades A informação é suficientemente completa para nossas necessidades A informação tem profundidade suficiente para nossas necessidades
Representação Consistente	A informação é representada em um formato consistente As informações sobre um mesmo tópico são agrupadas de forma consistente A informação está adequadamente organizada
Facilidade de Operação	A informação é facilmente inserida A informação é facilmente removida A informação é facilmente alterada
Livre de Erros	A informação obtida por meio do sistema é correta A informação é inserida de forma adequada no sistema A informação obtida por meio do sistema é precisa
Interpretabilidade	A informação é apresentada de modo conciso A informação é facilmente interpretada As unidades de medida para a informação são claras
Objetividade	A informação é baseada em fatos A informação é objetiva A informação apresenta uma visão imparcial
Relevância	A informação é útil para nosso trabalho A informação é essencial para nosso trabalho A informação é apropriada para nosso trabalho
Segurança	A informação é protegida contra acessos não autorizados O acesso a esta informação é suficientemente restrito A informação é passível de dano
Entendimento	A informação é fácil de entender A informação é fácil de compreender O significado da informação é fácil de entender

a intuitividade do protótipo, percebidos pela área da saúde, para a qual o sistema é destinado. A atividade de avaliação teve início com uma reunião na qual foram abordados os objetivos do projeto no qual está inserido o protótipo, o PGDCC desenvolvido, sua estrutura, os recursos e um exemplo de utilização. Em seguida, foi explanado sobre o procedimento de avaliação e foi disponibilizado um guia com exemplos de dados a serem preenchidos, de modo a auxiliar a utilização de todos os recursos oferecidos pelo sistema.

Parte dos entrevistados participou em modo presencial, enquanto outros participaram por meio de videoconferência, com o auxílio da ferramenta *Google Hangout*[2].

De modo a facilitar a coleta das respostas e também facilitar o acesso dos colaboradores

[2]http://www.google.com/tools/dlpage/res/talkvideo/hangouts/

em diferentes localizações, o formulário de avaliação foi disponibilizado em formato digital e na *Web*, com o auxílio da ferramenta *Google Docs*[3], sendo parte do questionário exibido na Figura 7.1.

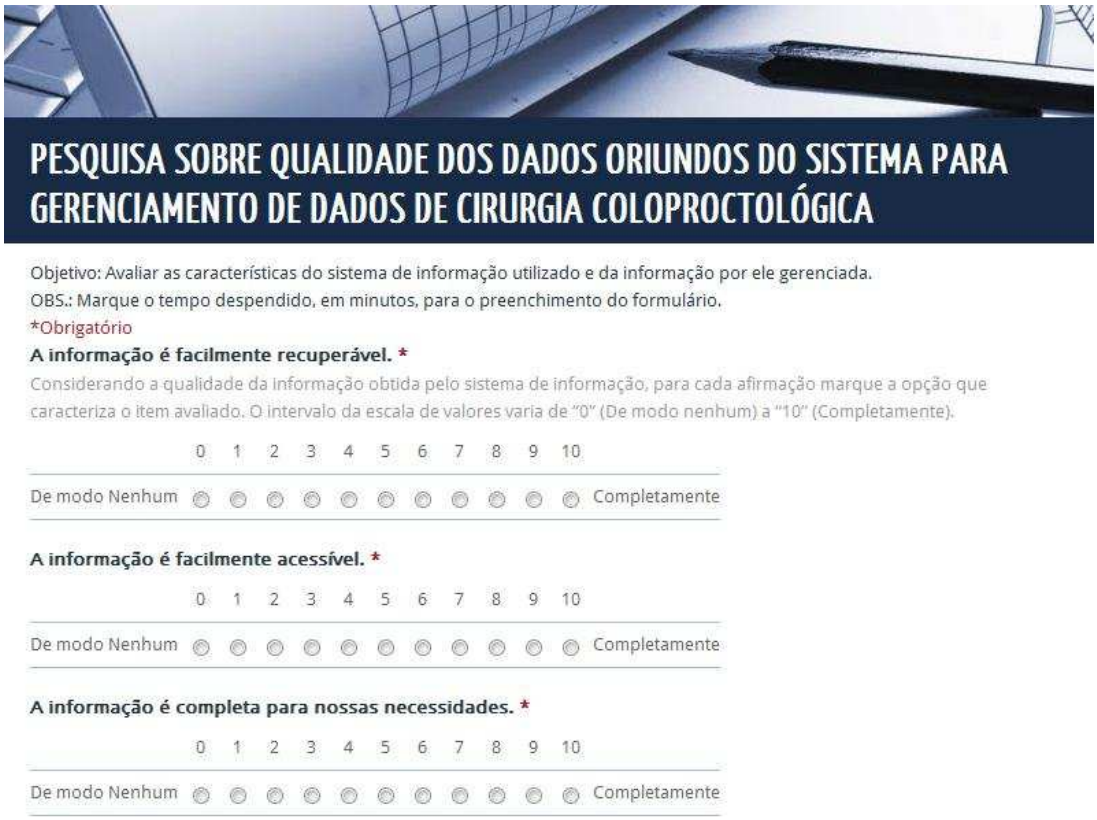

Figura 7.1: Formulário de avaliação qualitativa em formato digital disponibilizado na *Web*.

A avaliação do sistema proposta aos entrevistados consistiu em simular o preenchimento de dados fictícios, possibilitando ao usuário utilizar todos os recursos oferecidos pelo sistema e a partir dessa experiência responder ao questionário de avaliação qualitativa. De acordo com os dados obtidos por meio da **Seção C** do FAQP, o grupo de entrevistados foi composto por 12 pessoas, sendo três do sexo feminino e nove do sexo masculino, com intervalo de idade entre 19 e 26 anos. Dentre os entrevistados, oito eram da área da computação e quatro da área da saúde, com tempo de experiência variado, como representado na Tabela 7.2.

Para possibilitar a utilização do sistema pelos entrevistados, o protótipo foi disponibilizado para acesso por meio da *Web* em dois servidores em diferentes locais. Um dos servidores está instalado no LABI – UNIOESTE[4], enquanto o outro servidor[5] pertence à empresa Heroku[6] que disponibiliza sua plataforma para utilização de maneira gratuita durante 750 horas, mediante a realização de cadastro. A implantação da aplicação em duas estruturas se fez necessária devido a algumas dificuldades para acesso ao servidor presente no LABI quando o usuário es-

[3]https://docs.google.com
[4]http://200.134.24.164:3000
[5]http://sgdcc-labi.herokuapp.com/
[6]http://www.heroku.com/

Tabela 7.2: Perfil dos entrevistados.

Identificação	Área	Idade (Anos)	Sexo	Tempo de Atuação
Entrevistado 01	Computação	20	Masculino	4 anos
Entrevistado 02	Computação	26	Masculino	5 ou mais anos
Entrevistado 03	Saúde	19	Masculino	2 anos
Entrevistado 04	Computação	25	Masculino	5 ou mais anos
Entrevistado 05	Computação	25	Masculino	5 ou mais anos
Entrevistado 06	Saúde	24	Feminino	Menos de 1 ano
Entrevistado 07	Computação	21	Masculino	4 anos
Entrevistado 08	Saúde	24	Feminino	2 anos
Entrevistado 09	Computação	21	Masculino	4 anos
Entrevistado 10	Computação	24	Masculino	5 ou mais anos
Entrevistado 11	Computação	23	Masculino	5 ou mais anos
Entrevistado 12	Saúde	22	Feminino	4 anos

tivesse conectado por meio da rede interna do PTI e também devido a algumas instabilidades que acometeram a plataforma Heroku, como relatado pela própria empresa[7].

Após a explicação e a exemplificação da utilização do sistema, foi solicitado aos usuários que avaliassem o protótipo de acordo com a disponibilidade de cada participante, o que ocorreu no período de três a seis de março de 2012. De posse dos resultados da avaliação, os dados foram analisados fornecendo uma visão geral sobre o PGDCC e sobre a informação por ele gerenciada.

Em um primeiro momento foi realizada a análise dos dados pertencentes à **Seção A** do FAQP, que como mencionado consiste em questões extraídas do *IQA*, sendo algumas delas adaptadas, que visam avaliar a QD pertencentes ao sistema de informação.

O cálculo do resultado da avaliação de cada entrevistado foi realizado com base na média dos valores definidos para as três questões de cada dimensão de QD, bem como o cálculo do desvio-padrão de cada média. Após realizados os cálculos, a análise foi iniciada pelas avaliações dos entrevistados da área computacional, cujo grupo é formado por uma quantidade maior de integrantes.

A representação gráfica do resultado da avaliação realizada pelos entrevistados da área da computação pode ser observada na Figura 7.2, na qual o eixo x representa as dimensões de QD avaliadas e o eixo y o valor de cada dimensão. Ainda nesse gráfico é possível observar que a área hachurada indica o desvio-padrão da respectiva dimensão.

A análise dos dados representados na Figura 7.2 indicou que os aspectos pertencentes a algumas dimensões de QD necessitam de maior atenção. Algumas situações indicaram uma possível interpretação equivocada de questões específicas, devido a respostas contraditórias de

[7]https://status.heroku.com/incident/308

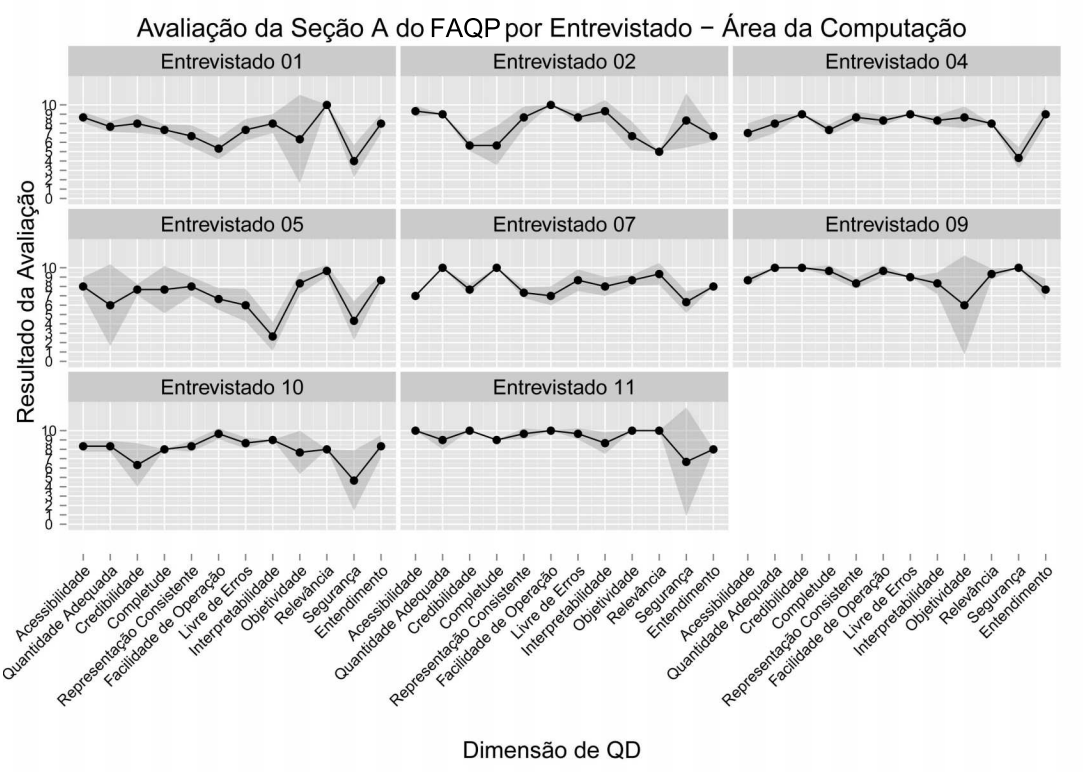

Figura 7.2: Representação gráfica do resultado da avaliação da **Seção A** do FAQP pelos entrevistados da área da computação.

alguns entrevistados em questões pertencentes à mesma dimensão, como observado nos pontos do gráfico que possuem um desvio-padrão acentuado. Este fato pode representar um indicativo da necessidade de um melhor esclarecimento sobre o método de avaliação qualitativa aplicado, bem como o significado das perguntas ou uma possível reformulação de algumas questões.

A média geral da avaliação dos entrevistados de computação pode ser observada na Figura 7.3. Nessa figura também está representada graficamente a média obtida por meio da avaliação dos entrevistados da área da saúde, que será posteriormente discutida nesta seção. Em relação ao resultado da avaliação da área computacional, é possível notar que os valores são inferiores ao resultado da avaliação realizada pela área da saúde. Esse fato pode ser justificado pelo conhecimento mais apurado sobre a representação da informação em formato digital, assunto relacionado a essa área. Já na área da saúde, é possível notar que a avaliação foi focada na utilidade e na usabilidade do protótipo proposto que é destinado à respectiva área.

Na Tabela 7.3 são apresentadas as médias, desvios-padrão, mínimo e máximo referentes ao resultado da avaliação das dimensões de QD realizada pela área de computação. Os valores inferiores a oito e desvios-padrão com valores superiores a dois foram destacados, por se diferenciarem da maior parte dos resultados obtidos.

Por meio da análise do gráfico que representa o resultado da avaliação da computação é possível observar que as dimensões de Interpretabilidade, Objetividade e Segurança obtiveram as menores médias, bem como um desvio-padrão acentuado em relação às demais dimensões.

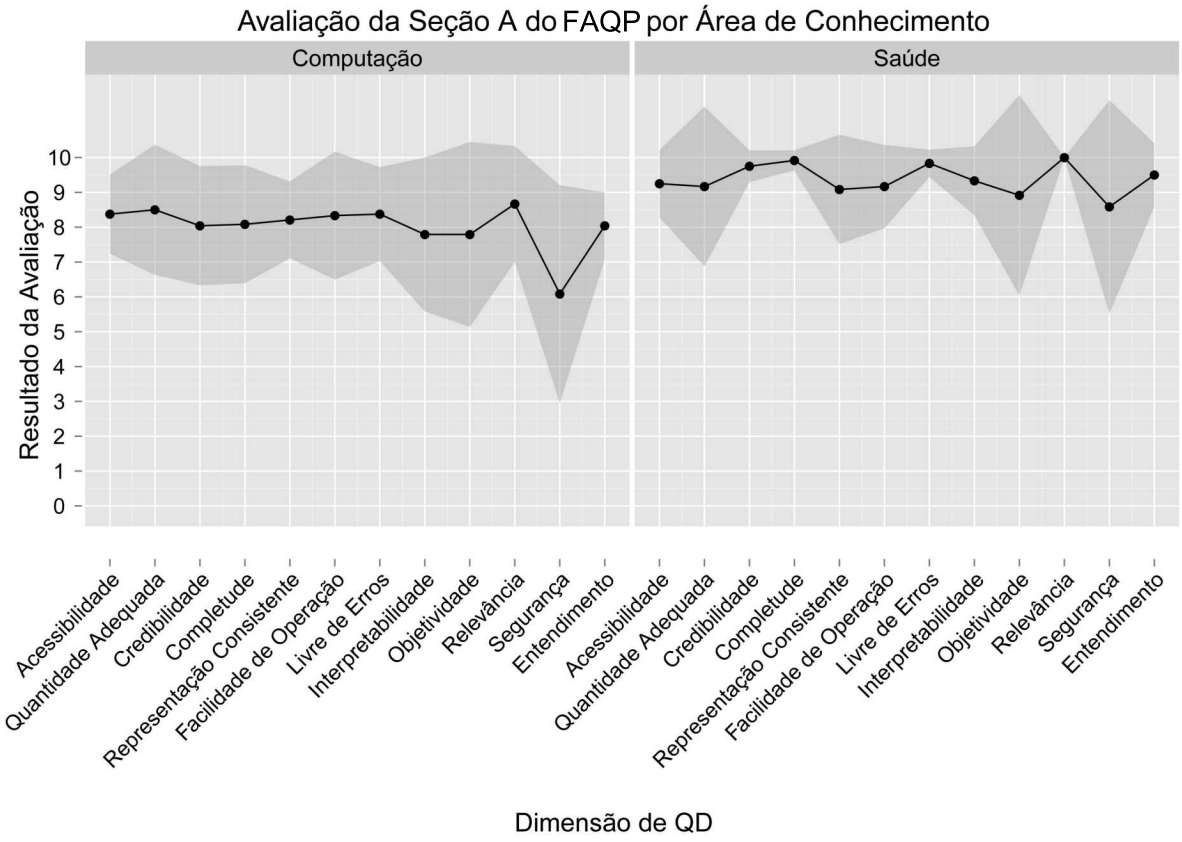

Figura 7.3: Representação gráfica da média geral da avaliação da **Seção A** do FAQP de acordo com as áreas da computação e da saúde.

Tabela 7.3: Resultado da avaliação das dimensões de QD pela área da computação.

Dimensões de QD	Média	desvio-padrão	Mínimo	Máximo
Acessibilidade	8,375	1,13492	6	10
Quantidade Adequada	8,5	1,86501	1	10
Credibilidade	8,041667	1,706233	5	10
Completude	8,083333	1,691839	4	10
Representação Consistente	8,208333	1,102533	6	10
Facilidade de Operação	8,333333	1,833663	4	10
Livre de Erros	8,375	1,345282	4	10
Interpretabilidade	**7,791667**	**2,206299**	1	10
Objetividade	**7,791667**	**2,653614**	0	10
Relevância	8,666667	1,659404	5	10
Segurança	**6,083333**	**3,119597**	0	10
Entendimento	8,0416667	0,9545847	6	10

De acordo com os conceitos anteriormente apresentados, a dimensão da Interpretabilidade visa mensurar se os dados estão representados em linguagem apropriada. A pouca familiaridade dos colaboradores da área da computação com os dados pertencentes ao domínio da aplicação pode ter dificultado a avaliação dessa dimensão. De qualquer modo, mesmo que o sistema seja

destinado principalmente a usuários da área da saúde, para facilitar a correta interpretação dos campos presentes no protótipo podem ser adicionadas notas explicativas às telas do sistema que são exibidas a partir da interação do usuário com os formulários de cadastro.

Já a dimensão da Objetividade visa mensurar se os dados são imparciais e não tendenciosos. Os dados que são inseridos no sistema têm origem nos prontuários dos pacientes e não em um processo de amostragem, portanto são considerados imparciais. Entretanto, os valores obtidos em algumas avaliações, por exemplo, o resultado dos entrevistados 01 e 09 na Figura 7.2, apresentam valores baixos para essa dimensão, sendo uma das hipóteses a interpretação equivocada do significado de questões relacionadas a esta dimensão, como "**A informação apresenta uma visão imparcial**".

Por sua vez a dimensão de Segurança visa mesurar a restrição de acesso aos dados. Apesar do sistema apresentar mecanismos de autenticação, podem ser adicionadas medidas mais restritivas, por exemplo, o uso de biometria ou até mesmo o uso de assinatura digital, sendo este último recurso indicado no MCSRES como exigência para a obtenção da certificação NGS2, como explicado no Capítulo 5. Além de recursos mais avançados de autenticação, é interessante adicionar mecanismos que registrem as operações realizadas pelos usuários, de modo a identificar a origem de possíveis falhas ou erros durante a manipulação dos dados.

Parte das sugestões apresentadas anteriormente foi adicionada pelos entrevistados na **Seção D** do FAQP, na qual foi disponibilizado um campo para tal fim.

Em relação à área da saúde, o resultado individual das avaliações dos entrevistados, assim como apresentado para a área da computação, pode ser visibilizado graficamente na Figura 7.4. Nessa figura também é possível observar que as dimensões de Quantidade Adequada, Objetividade e Segurança possuem valores de desvio-padrão consideráveis. Os valores de cada dimensão podem ser observados na Tabela 7.4.

Assim como na área da computação, o resultado das avaliações da área da saúde também indicou que as dimensões de Segurança e Objetividade apresentaram uma média inferior se comparado com as outras dimensões de QD. Algumas medidas adicionais de segurança foram sugeridas, como citado nos parágrafos anteriores. Também percebeu-se que a dimensão de Objetividade requer um melhor esclarecimento para os usuários.

Já o desvio-padrão da dimensão de Quantidade Adequada foi influenciado pela avaliação do entrevistado 08, que não considerou a quantidade de informação como alta ou baixa, mas sim como adequada para as necessidades do domínio, de modo semelhante ao entrevistado 05 da área de computação.

Algumas críticas especificadas na **Seção D** do FAQP indicaram dificuldades no preenchimento de dados, como o cadastro de diagnóstico, e na navegação entre diferentes seções, como em ocorrências. Devido à quantidade elevada de campos nas seções relacionadas à ocorrência, a avaliação revelou uma dificuldade maior da utilização do sistema pelos usuários da saúde,

fator que pode ser minimizado pela realização de um estudo mais aprofundado para a adição de campos informativos e estruturas mais claras que direcionem o fluxo de preenchimento dos dados.

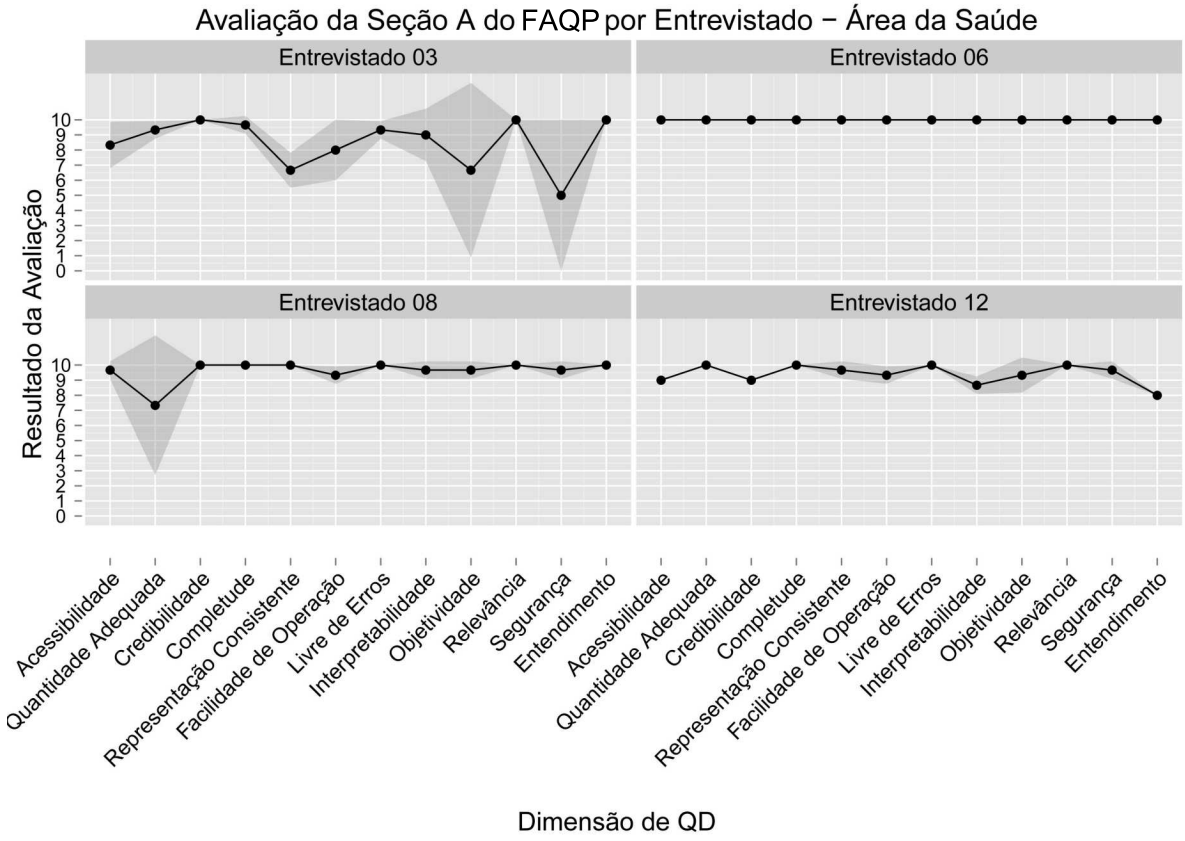

Figura 7.4: Representação gráfica do resultado da avaliação da **Seção A** do FAQP pelos entrevistados da área da saúde.

Tabela 7.4: Resultado da avaliação das dimensões de QD pela área da saúde.

Dimensões de QD	Média	Desvio-Padrão	Mínimo	Máximo
Acessibilidade	9,25	0,9653073	7	10
Quantidade Adequada	9,166667	**2,289634**	2	10
Credibilidade	9,75	0,452267	9	10
Completude	9,9166667	0,2886751	9	10
Representação Consistente	9,083333	1,564279	6	10
Facilidade de Operação	9,166667	1,193416	6	10
Livre de Erros	9,8333333	0,3892495	9	10
Interpretabilidade	9,3333333	0,9847319	7	10
Objetividade	8,916667	**2,874918**	0	10
Relevância	10	0	10	10
Segurança	8,583333	**3,058768**	0	10
Entendimento	9,5	0,904534	8	10

A média geral da **Seção A**, que se refere à avaliação da QD sob a perspectiva do usuário,

pode ser observada na Figura 7.5. Nesse gráfico é possível observar que o resultado da avaliação de cada dimensão é satisfatório para a fase em que se encontra o protótipo, apresentando pequeno declínio em relação à dimensão de Segurança.

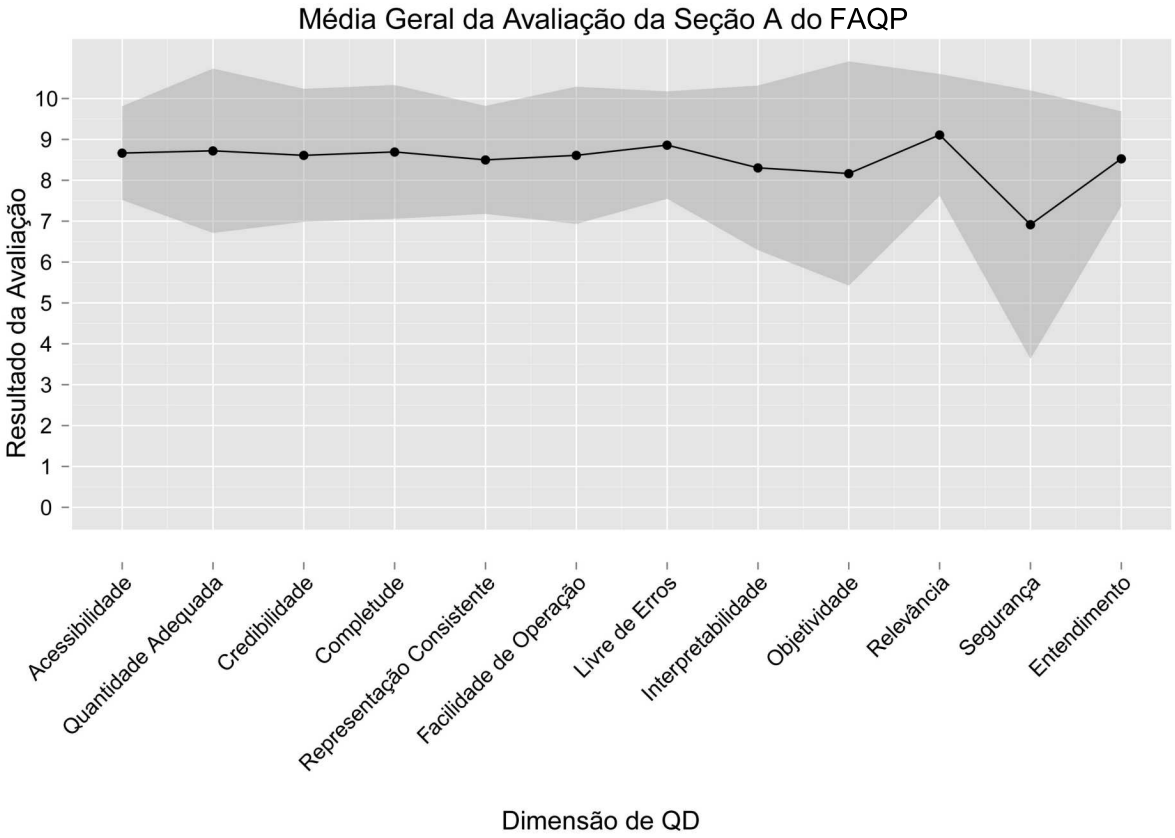

Figura 7.5: Representação gráfica da média geral da avaliação da **Seção A** do FAQP.

Como citado, a **Seção B** do FAQP é composta por questões que visam extrair informações sobre as características de usabilidade e importância do protótipo proposto, que podem ser visibilizadas no Apêndice A. Na Figura 7.6 é possível observar graficamente a média do resultado de cada questão da avaliação da **Seção B** do FAQP realizada pelos entrevistados da área da computação e da área da saúde. Neste gráfico o eixo x representa as questões da respectiva seção do FAQP e o eixo y o valor de cada questão.

Entre os aspectos do PGDCC que necessitam de melhoras, de acordo com a avaliação dos entrevistados da computação, destaca-se o aprimoramento de mecanismos de busca para localização de informações de maneira mais abrangente, como indicado pela média das respostas da questão 40. Também se faz necessária uma validação mais abrangente dos campos dos formulários de cadastro, atualmente presente em algumas partes do sistema. Este fato é indicado pelas respostas da questão 39, que também foi avaliada como deficiente pelos entrevistados da área da saúde. Outro aspecto cujo resultado da avaliação é semelhante em ambos os grupos, representado pela questão 37, consiste na melhoria da interface de algumas sessões relacionadas ao cadastro de ocorrências cirúrgicas, tornando a utilização do sistema mais intuitiva, fato que foi melhor percebido pela área da saúde.

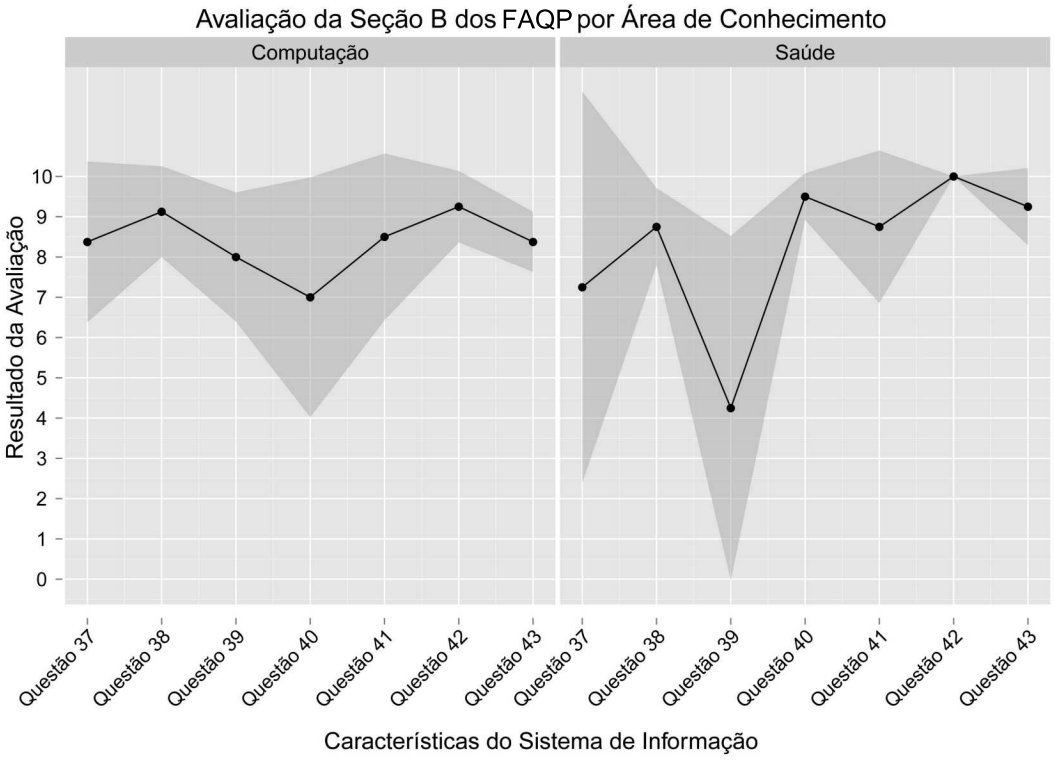

Figura 7.6: Média do resultado da avaliação da **Seção B** do FAQP por área de conhecimento.

De modo mais abrangente, a média geral da avaliação da **Seção B** em conjunto das áreas de saúde e computação, que pode ser graficamente observada na Figura 7.7, reafirma as melhorias necessárias ao protótipo discutidas no parágrafo anterior.

7.2 Considerações Finais

A avaliação qualitativa de QD revelou uma boa aceitação dos usuários em relação ao protótipo proposto, indicando que o processo de desenvolvimento adotado no estudo de caso foi adequado. Esse resultado foi favorecido principalmente pela forte interação com os especialistas da saúde e da computação durante as fases de levantamento de requisitos, projeto e desenvolvimento.

O processo de avaliação aplicado também possibilitou a identificação de aspectos do PGDCC e de seus dados que requerem uma melhor estruturação de maneira a tornar mais simples e intuitiva a sua utilização. Assim, o FAQP mostrou-se como uma ferramenta de grande importância, não somente na avaliação dos dados de um sistema de informação em uso, mas também em etapas que antecedem a fase de implantação do *software*.

Entretanto, o monitoramento do nível de QD deve ser realizado de modo contínuo, pois ao longo do tempo a experiência do usuário com o sistema será aprimorada e serão adicionados dados com diferentes características, fatores que podem revelar necessidades adicionais.

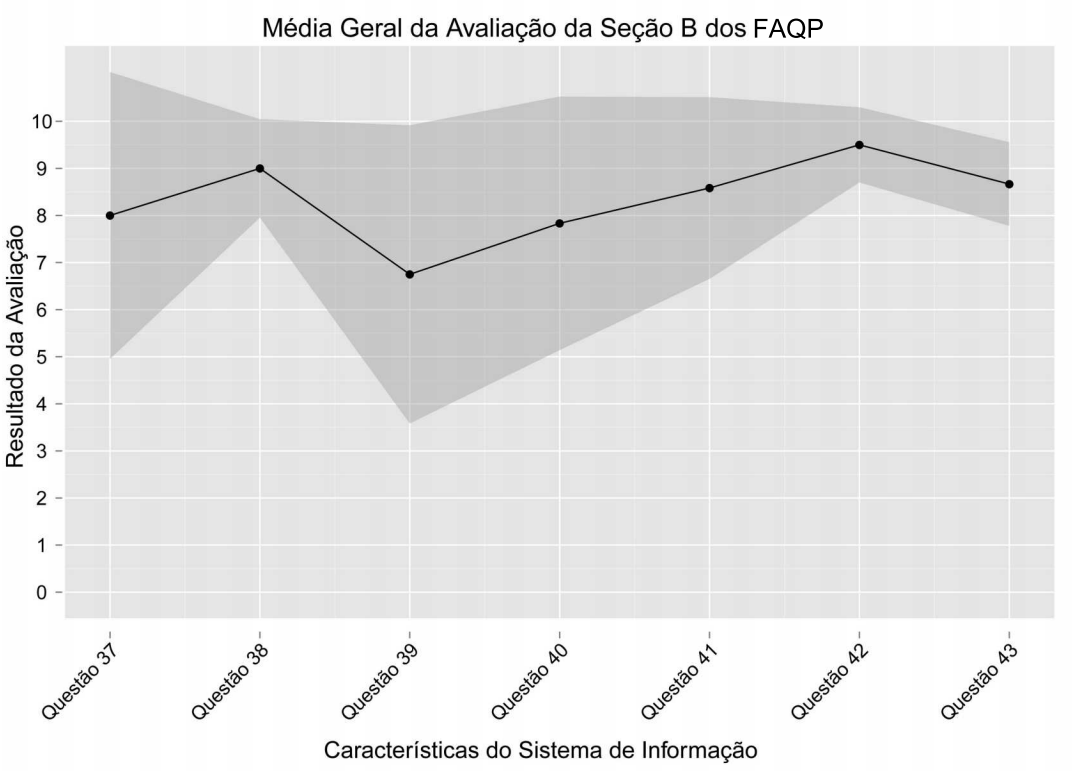

Figura 7.7: Média geral da avaliação da **Seção B** do FAQP.

Por meio da avaliação aplicada e por interações posteriores com os entrevistados notou-se dificuldades na interpretação de algumas questões referentes à **Seção A**, destinada à avaliação de QD. Também podem ser realizados estudos futuros comparando as diferenças na aplicação do *IQA*, proposto na literatura, e do formulário de pesquisa proposto neste trabalho, derivado com base no *IQA*, a partir de discussões e análise junto a especialistas.

No próximo capítulo são apresentadas as conclusões obtidas no desenvolvimento deste trabalho de pesquisa.

Capítulo 8

Conclusão

Neste trabalho foi realizado o estudo da aplicação de princípios de Qualidade de Dados (QD) durante o desenvolvimento de um sistema de informação para a medicina, visando à criação de uma estrutura computacional que auxilie no correto armazenamento dos dados e disponibilize uma interface gráfica apropriada para o manuseio da informação. A construção de uma base de dados robusta e adequada contribui tanto para as atividades relacionadas ao respectivo *software* médico quanto para futuras pesquisas e aplicações de processos computacionais que façam uso desses dados.

O impacto de problemas de QD, como relatado em diversos casos na literatura, pode resultar em perdas econômicas e sociais. Assim, a pesquisa na área de QD investiga as causas de problemas nos dados bem como métodos de avaliação e processos de melhoria da qualidade. O enfoque de grande parte das pesquisas nesta área está relacionado à identificação e a correção de problemas existentes. Apesar da importância de métodos corretivos, a adoção de uma abordagem preventiva pode minimizar o esforço necessário, e consequentemente os custos, para a garantia do nível de qualidade. Além disso, a preocupação com a qualidade deve ser contínua, o que requer processos que realizem o monitoramento constante do nível de qualidade.

Na área médica, todos os procedimentos aplicados durante o atendimento de saúde de um paciente são registrados em um prontuário, tradicionalmente, mantido em papel. Atualmente, muitas instituições de saúde têm optado por gerenciar eletronicamente os dados dos pacientes. Sendo assim, os Sistemas para Registro Eletrônico em Saúde devem oferecer uma estrutura confiável e segura para a gestão da informação relativa aos cuidados de saúde dos pacientes. Esta preocupação motivou a Sociedade Brasileira de Informática em Saúde, juntamente com o Conselho Federal de Medicina, a criar um processo de certificação que assegure o correto funcionamento dos sistemas de informação submetidos à avaliação. O trabalho dessas duas entidades também resultou na criação de um manual, como apresentado no Capítulo 5, que além de orientar para o processo de certificação serve também como guia para a especificação das funcionalidades necessárias e recomendadas a sistemas que gerenciam dados médicos.

Assim, os conceitos discutidos neste trabalho foram aplicados durante a criação do protótipo de um sistema para gerenciamento de dados de cirurgia coloproctológica, que partiu da

análise de um sistema legado relacionado ao estudo de caso. Neste estudo de caso também foi proposto o modelo de um processo de monitoramento de QD para auxiliar tanto no correto preenchimento do sistema quanto na identificação dos registros que não estejam de acordo com especificações previamente definidas. Esse modelo proposto também pode auxiliar no processo de revisão dos dados, indicando os locais nos quais estão presentes campos de dados que necessitem de correção.

Um dos fatores de importância percebidos durante o processo de desenvolvimento do sistema consiste na estreita relação com os especialistas de domínio em todas as etapas do projeto. Desse modo foi possível o levantamento mais preciso dos requisitos necessários ao sistema, o refinamento dos modelos definidos durante o projeto do sistema e a revisão das funcionalidades implementadas de modo incremental. Para esta última fase, as ferramentas escolhidas possuem recursos que contribuíram significativamente com a criação do protótipo em função das necessidades de QD, como os mecanismos de validação de atributos pertencentes a cada entidade, utilizado pelo processo de monitoramento de QD.

Durante o projeto e o desenvolvimento do sistema, uma das preocupações consistiu na definição de interfaces gráficas de simples utilização, já que o meio pelo qual os dados são inseridos representa uma das causas mais comuns do surgimento de problemas nos dados. Também foi realizada a modelagem de uma base de dados visando o correto armazenamento da informação cadastrada, repercutindo em futuras aplicações de processos de análise inteligente de dados, como o processo de Mineração de Dados (MD). Como mencionado no Capítulo 2, este processo é composto por diversas etapas, cada qual com características e responsabilidades diferentes. A preocupação com a qualidade do conjunto de dados pode auxiliar na redução do tempo despendido na fase de pré-processamento e em uma menor influência de ruídos sobre o modelo obtido, por meio da etapa de extração de padrões.

Após implementados os requisitos do protótipo verificou-se o parecer inicial dos usuários em relação ao sistema proposto. Uma das abordagens adotadas para definir dados com qualidade consiste no fato de que os dados devem estar de acordo com as necessidades dos usuários. Assim, de modo a analisar essa perspectiva, em relação ao sistema e sua informação, foi realizada uma avaliação qualitativa de QD juntamente a colaboradores da área da saúde e da área computacional, com base em métodos definidos na literatura, como descrito no Capítulo 7. Essa avaliação indicou pontos para melhoria tanto em relação à estrutura e a organização dos dados do protótipo quanto em relação a usabilidade do mesmo, mostrando-se como um método eficiente, não somente para a avaliação da informação de um sistema já em uso como também em etapas anteriores a sua implantação.

Desse modo, considerando o protótipo desenvolvido e o resultado da avaliação realizada, pode-se concluir que:

- O protótipo proposto oferece uma estrutura adequada para o registro de dados de cirurgia coloproctológica, preenchendo as lacunas existentes em relação às funcionalidades

disponibilizadas pelo sistema legado;

- A preocupação com a qualidade dos dados de um sistema computacional médico deve iniciar durante a concepção do sistema e acompanhá-lo em todo o tempo de atividade;
- A relação estreita com especialistas dos domínios envolvidos na criação de um sistema computacional médico é um fator essencial para seu desenvolvimento adequado;
- O processo de monitoramento de QD proposto neste trabalho pode auxiliar no correto preenchimento e manutenção dos campos de dados presentes nos formulários de cadastro e consequentemente na construção de uma base de dados com qualidade;
- A avaliação qualitativa de QD mostra-se como um recurso de grande importância para averiguar a adequação tanto do sistema quanto dos dados por ele gerenciados;
- As ferramentas de desenvolvimento de *software* selecionadas oferecem recursos que favorecem e auxiliam no desenvolvimento de sistemas de informação com qualidade, contribuindo, consequentemente, para a qualidade dos dados gerenciados por esses sistemas.

8.1 Principais Contribuições

As principais contribuições deste trabalho podem ser organizadas da seguinte maneira:

- Proposta de um processo computacional para monitoramento da QD pelo qual são gerenciados os registros que apresentam problemas nos dados;
- Criação de uma estrutura computacional adequada, por meio do protótipo de sistema computacional desenvolvido, para o armazenamento e o gerenciamento de dados de cirurgia coloproctológica, cujas principais funcionalidades são:
  - Mecanismos para controle de acesso restringindo a operação somente a usuários autorizados;
  - Organização da informação de acordo com os elementos clássicos da propedêutica médica;
  - Disponibilização de grande quantidade de atributos com valores pré-definidos em listas;
  - Possibilidade de adição de novos valores às listas por meio de janelas flutuantes sem que a sessão seja encerrada;
  - Implementação de mecanismos para ocultar campos de dados não utilizados;
  - Mecanismos assistentes para o preenchimento semi-automatizado de formulários específicos, com base em registros anteriores do paciente;
- Criação de uma base de dados adequada para futuras aplicações do processo de MD;

- Publicação de resultados por meio de:
  - Trabalho publicado e apresentado no 60º Congresso Brasileiro de Coloproctologia;
  - Trabalho publicado na Revista Brasileira de Coloproctologia;
  - Trabalho publicado e apresentado na Conferência Brasileira de Dinâmica, Controle e Aplicações (DINCON) – 2011;
  - Relatório técnico.

Dadas as limitações existentes no sistema legado, que foram apresentadas no Capítulo 6, foi realizado o planejamento de um novo sistema disponibilizado via *Web*, oferecendo uma melhor estrutura computacional cuja informação foi organizada de modo a facilitar a utilização pelo profissional de saúde. Também foram implementados recursos na interface gráfica visando facilitar o preenchimento correto dos atributos. Entre esses elementos destacam-se os atributos que possuem listas de valores pré-definidos e a possibilidade de adição dinâmica de novos valores a essas listas por meio de janelas flutuantes, sem a necessidade de encerrar a respectiva sessão. Também foram disponibilizados mecanismos para ocultar dados não utilizados de um formulário, melhorando a usabilidade e reduzindo a exibição, em excesso, de campos de dados desnecessários. Outro recurso que otimiza a tarefa de cadastro dos dados de pacientes recorrentes, por parte dos profissionais de saúde, consiste no preenchimento semi-automatizado de determinadas seções do sistema, tendo como base o último registro do paciente. Assim, o usuário necessita apenas realizar a verificação dos registros preenchidos além de eventuais correções.

Conforme explanado no Capítulo 6, e mencionado neste capítulo, o modelo de monitoramento proposto neste trabalho auxilia na etapa de inserção dos dados no sistema e no monitoramento dos registros de pacientes que não estão de acordo com regras definidas em escopos presentes nas entidades monitoradas. Em um primeiro momento, durante o processo de criação de um novo registro, é realizada a análise dos dados inseridos no formulário pelo usuário. Caso exista a violação de alguma regra pertencente à entidade que está sendo cadastrada, o usuário é alertado por meio de mensagens de texto indicando os atributos que necessitam de correção. Assim, o usuário pode optar por corrigir os respectivos atributos ou ignorar a mensagem de alerta, dando continuidade à utilização do sistema. Caso seja escolhida a última opção é executada a segunda etapa do processo de monitoramento que consiste na marcação da entidade cadastrada de modo a identificar a existência de problemas no registro do respectivo paciente e auxiliar em uma posterior correção. Também são atribuídos sinais gráficos ao registro do paciente com a finalidade de facilitar a navegação do usuário no sistema para localização da origem do problema.

O conjunto de dados pertencente ao protótipo proposto foi modelado de maneira a possibilitar o cadastro adicional de informações sobre o paciente, como hábitos sociais e alimentares entre outros. Estas informações complementares foram definidas junto aos especialistas de domínio por representarem características que podem contribuir com o reconhecimento de

padrões interessantes em futuras aplicações do processo de MD. Além disso, a estrutura de armazenamento dos dados foi definida com o auxílio do SGBD *PostgreSQL*, sendo este um banco de dados robusto que oferece mecanismos para garantir a integridade da informação, não possuindo as limitações existentes no sistema legado.

Durante o desenvolvido deste projeto também foi possível compartilhar com o meio acadêmico o trabalho realizado por meio da publicação de relatório técnico e de artigos. No relatório técnico são apresentados os protótipos de telas criados durante o desenvolvimento do sistema, bem como a descrição das mesmas. Para a Conferência Brasileira de Dinâmica, Controle e Aplicações – 2011 foi submetido um artigo relacionado à contribuição do processo de monitoramento de QD proposto, integrado ao protótipo desenvolvido. Já para o 60º Congresso Brasileiro de Coloproctologia e para a Revista Brasileira de Coloproctologia foram submetidos trabalhos que relatam de maneira mais abrangente o processo de desenvolvimento do protótipo proposto neste trabalho, visando a qualidade dos dados.

8.2 Limitações

Algumas limitações encontradas durante o desenvolvimento deste trabalho incluem:

- Custo computacional elevado do algoritmo de monitoramento de QD;
- Definição em código das regras para monitoramento de QD;
- Registro limitado de problemas de QD;
- Realização de uma avaliação qualitativa inicial restrita a um grupo de usuários das áreas de computação e saúde.

Para efetuar a marcação dos registros de pacientes que apresentam problemas nos dados, o algoritmo de monitoramento atualiza o estado das entidades relacionadas ao respectivo registro. Esse processo também é realizado após a criação, atualização ou remoção de um registro monitorado que apresenta problemas, necessitando de um estudo que avalie o impacto dessas operações sobre a escalabilidade do sistema e vise a otimização dos procedimentos executados pelo módulo de monitoramento.

De modo a realizar a customização de regras para o monitoramento de novas dimensões de QD ou modificação das regras existentes é necessário realizar alterações no código da entidade. Apesar da simplicidade deste processo, caso o usuário deseje monitorar uma nova condição dos dados, torna-se necessário solicitar ao desenvolvedor alterações na codificação do sistema.

Atualmente o módulo de monitoramento de QD não exibe estatísticas sobre os problemas de acordo com a dimensão de QD cujos requisitos não foram atendidos. São necessárias implementações adicionais para suprir tal necessidade e assim auxiliar em futuras aplicações de

processos de avaliação objetiva de QD, por meio do fornecimento de informações mais detalhadas sobre os problemas.

A avaliação qualitativa realizada neste trabalho, em uma etapa preliminar à implantação do sistema, contou com um grupo restrito de colaboradores cuja distribuição foi composta por uma quantidade diferente de entrevistados das áreas de computação e da saúde. Após a fase de implantação e início da utilização do sistema poderão ser realizadas novas avaliações, buscando ampliar o grupo de colaboradores, possivelmente de diferentes áreas, de modo a manter uma avaliação contínua da QD relacionada ao PGDCC.

8.3 Trabalhos Futuros

Durante o desenvolvimento deste trabalho foram identificadas diversas questões que podem ser tratadas em trabalhos futuros. Algumas delas incluem:

- Desenvolver casos de teste mais amplos para posterior implantação da primeira versão do sistema;
- Avaliar o nível de qualidade dos dados presentes no sistema legado e realizar a correção de problemas existentes;
- Adaptar e migrar os dados do sistema legado para o novo sistema;
- Aperfeiçoar a heurística utilizada no módulo de monitoramento de QD e armazenar informações adicionais sobre os problemas registrados;
- Disponibilizar uma interface para customização de regras e definição dos atributos que serão monitorados;
- Possibilitar o envio de relatórios de problemas por e-mail;
- Estudar a aplicação de algoritmos de inteligência computacional para auxiliar na identificação de problemas de QD;
- Integrar o sistema proposto neste trabalho com o Sistema de Gestão de Protocolo de Câncer Colorretal desenvolvido pelo LABI;
- Realizar avaliação contínua, com colaboradores de diferentes áreas e experiências, para o monitoramento da QD.

Referências Bibliográficas

Batini, C. and Scannapieca, M. (2006). *Data quality: concepts, methodologies and techniques*, Springer, New York.

Bemmel, J., Musen, M. and Helder, J. (1997). *Handbook of medical informatics*, Handbook of Medical Informatics, Springer, Heidelberg.

Blake, R. and Mangiameli, P. (2011). The effects and interactions of data quality and problem complexity on classification, *J. Data and Information Quality* **2**: 8:1–8:28.

Borrás, C. (2006). Overexposure of radiation therapy patients in Panama: problem recognition and follow-up measures, *Revista Panamericana de Salud Pública* **20**(2-3): 173–187.

Brasil (2012). *Portaria N° 188, de 15 de Março de 2012*. Torna público o Regimento Interno do Comitê de Informação e Informática em Saúde (CIINFO/MS), na forma do Anexo. Diário Oficial [da] República Federativa do Brasil, Poder Executivo, Brasília, DF, 16 mar. 2012. Seção 1, p. 171. Disponível em: http://www.brasilsus.com.br/legislacoes/secretaria-executiva/112524-188.html. Acesso em: 19 mar. 2012.

Braude, E. (2005). *Projeto de Software*, Bookman, São Paulo.

Carneiro Junior, C. and Barazi, R. A. (2010). *Beginning Rails 3*, Apress, New York.

Carvalho, D. R., Bueno, M. and Neto, W. (1999). Ferramenta de Pré e Pós-processamento para Data Mining, *XII Seminário de Computação* . Disponível em: http://www.inf.furb.br/seminco/2003/artigos/97-vf.pdf. Acesso em: 28 nov. 2010.

Chaffer, J. and Swedberg, K. (2007). *Learning jQuery*, Packt Publishing, Birmingham.

Chan, P., Fan, W., Prodromidis, A. and Stolfo, S. (1999). Distributed data mining in credit card fraud detection, *IEEE Intelligent Systems* **14**(6): 67–74.

Chapman, A. (2005). Principles of data quality, *Relatório técnico*, Global Biodiversity Information Facility, Queensland. Disponível em: http://niobioinformatics.in/pdf/workshop/Data%20Quality.pdf. Acesso em: 10 fev. 2012.

Collingbourne, H. (2008). *The Little Book of Ruby*, 2 edn, Sapphiresteel Software. Disponível em: http://www.sapphiresteel.com/The-Little-Book-Of-Ruby. Acesso em: 02 fev. 2012.

Conselho Federal de Medicina (2002a). *Parecer 30/02*. Processo consulta sobre Prontuário Eletrônico. Disponível em: http://www.portalmedico.org.br/pareceres/CFM/2002/30_2002.htm. Acesso em: 15 fev. 2012.

Conselho Federal de Medicina (2002b). *Resolução 1.638/02*. Define prontuário médico e torna obrigatória a criação da Comissão de Revisão de Prontuários nas instituições de saúde. Disponível em: http://www.portalmedico.org.br/resolucoes/CFM/2002/1638_2002.htm. Acesso em: 15 fev. 2012.

Conselho Federal de Medicina (2007). *Resolução 1.821/07*. Normas técnicas para a digitalização e uso dos sistemas informatizados para a guarda e manuseio dos documentos dos prontuários dos pacientes. Disponível em: http://www.portalmedico.org.br/resolucoes/CFM/2007/1821_2007.htm. Acesso em: 15 fev. 2012.

Cooper, P. (2007). *Beginning Ruby: From Novice to Professional*, Apress, New York.

Costa, L. H. D. d., Ferrero, C. A., Lee, H. D., Coy, C. S. R., Fagundes, J. J. and Wu, F. C. (2010). Mapeamento de laudos médicos de endoscopia digestiva alta apoiados por ontologias, *Anais do X Workshop de Informática Médica*, Belo Horizonte, pp. 1479–1488.

Crespo, A. N., de Barros, C. P., Jino, M., Junior, M. A. and Bueno, P. M. S. (2011). Piloto de testes, *Linux Magazine* pp. 20–22.

Criptografia (2010). Priberam Informática, Brasil. Disponível em: http://www.priberam.pt/dlpo/dlpo.aspx?pal=criptografia. Acesso em: 7 dez. 2010.

Das, S. and Saha, B. (2009). Data Quality Mining using Genetic Algorithm, *International Journal of Computer Science and Security* **3**(2): 105–112.

Denning, R. and Elizabeth, D. (1982). *Cryptography and data security*, Addison-Wesley Longman Publishing Co., Inc., Boston.

Elmasri, R. and Navathe, S. B. (2005). *Sistemas de banco de dados*, Pearson Addison, São Paulo.

English, L. P. (1999). *Improving data warehouse and business information quality: methods for reducing costs and increasing profits*, John Wiley & Sons, Inc., New York.

Eppler, M. J. (2006). *Managing information quality: increasing the value of information in knowledge-intensive products and processes*, Springer, Germany.

Fadel, F. D. Q., Malucelli, A. and Bastos, L. C. (2006). Prontuário Eletrônico Para Pacientes De Hanseníase Via Web, *X Congresso Brasileiro de Informática em Saúde*, Florianópolis, pp. 1462–1467.

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. and Others (1996). Knowledge discovery and data mining: Towards a unifying framework, *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR*, pp. 82–88.

Fioravanti, C. (2012). Pontes entre disciplinas, *Pesquisa FAPESP online* . Edição impressa 191. Disponível em: http://revistapesquisa.fapesp.br/2012/01/16/pontes-entre-disciplinas/. Acesso em: 25 mar. 2012.

Fisher, C. (2001). Criticality of data quality as exemplified in two disasters, *Information & Management* **39**(2): 109–116.

Flanagan, D. (2006). *JavaScript: the definitive guide*, O'Reilly, Sebastopol.

Freitas, A. A. (1999). On rule interestingness measures, *Knowledge-Based Systems* **12**(5–6): 309–315.

Ge, M. and Helfert, M. (2007). A Review of Information Quality Research, *Proceedings of the 12th ICIQ*, MIT, Cambridge, pp. 76–91.

Gopalakrishnan, V., Lustgarten, J. L., Visweswaran, S. and Cooper, G. F. (2010). Bayesian rule learning for biomedical data mining., *Bioinformatics (Oxford, England)* **26**(5): 668–75.

Graves, K. (2007). *CEH: official certified ethical hacker review guide*, Wiley Pub., Indianapolis.

Griffiths, D. (2008). *Head First Rails: A Learner's Companion to Ruby on Rails*, O'Reilly, Sebastopol.

Guedes, G. T. A. (2009). *UML 2 - Uma Abordagem Prática*, Novatec Editora, São Paulo.

Guerra-García, C., Caballero, I. and Piattini, M. (2010). A systematic literature review of how to introduce data quality requirements into a software product development, *5th International Conference on Evaluation of Novel Approaches to Software Engineering*, Athens, pp. 12–19.

Han, J. and Kamber, M. (2006). *Data Mining: Concepts and Techniques*, 2 edn, Elsevier, San Francisco.

Hassine, S. B., Clément, D. and Laboisse, B. (2008). Using Association Rules to Detect Data Quality Issues, *Proceedings of the 13th ICIQ*, pp. 184–197.

Heilmann, C. (2006). *Beginning Javascript with Dom Scripting and Ajax*, Apress, New York.

Hellsten, C. and Laine, J. (2006). *Beginning Ruby on Rails E-Commerce*, Apress, New York.

Herzberg, S., Rahbar, K., Stegger, L., Schäfers, M. and Dugas, M. (2011). Concept and implementation of a computer-based reminder system to increase completeness in clinical documentation., *Int J Med Inform* **80**(5): 351–358.

Initiative, O. S. (2010). Open Source Initiative OSI - The MIT License:Licensing. Disponível em: http://www.opensource.org/licenses/mit-license.php. Acesso em: 16 nov. 2010.

Jung, W., Lee, H. D., da Silva, A. C., da Costa, L. H. D., Espindola, B., Coy, C. S. R., Fagundes, J. J. and Wu, F. C. (2011). Adoção de Medidas de Qualidade de Dados para o Desenvolvimento de Sistemas Biomédicos, *X Conferência Brasileira de Dinâmica, Controle e Aplicaçoes - DINCON 2011*, Águas de Lindóia, pp. 751–754.

Kwak, C. and Yih, Y. (2004). Data-Mining Approach to Production Control in the Computer-Integrated Testing Cell, *IEEE Transactions on Robotics and Automation* **20**(1): 107–116.

Larose, D. T. (2005). *Discovering Knowledge in Data*, Wiley-Interscience, Hoboken.

Lauriat, S. M. (2007). *Advanced Ajax*, Prentice Hall, Boston.

Lee, H. D. (2005). *Seleção de atributos importantes para a extração de conhecimento de bases de dados*, Tese de doutorado, Universidade de São Paulo (USP), São Paulo.

Lee, H. D., da Costa, L. H. D., Ferrero, C. A., Coy, C. S. R., Fagundes, J. J., Machado, R. B. and Wu, F. C. (2011). Protótipo de um sistema de gerenciamento de protocolos de câncer colorretal, *Revista Brasileira de Coloproctologia* **31**(1): 1–7.

Lee, H. D., da Costa, L. H. D., Ferrero, C. A., Coy, C. S. R., Fagundes, J. J., Machado, R. B. and Wu, F. C. (2012). Protótipo de um Sistema para Gerenciamento de Dados de Cirurgia Coloproctológica, *Revista Brasileira de Coloproctologia (aceito para publicação)* .

Lee, H. D., Jung, W., Silva, A. C., Costa, L. H. D. d. and Wu, F. C. (2011). Descrição do Protótipo de Telas para o Sistema de Gerenciamento de Protocolo de Cirurgia Coloproctológica, *Relatório técnico*, Universidade Estadual do Oeste do Paraná - Laboratório de Bioinformática, Foz do Iguaçu. Disponível em: http://200.134.24.164:8080/labi/documentos/RTLabi/Huei.pdf. Acesso em: 25 mar. 2012.

Lee, Y. W., Pipino, L. L., Funk, J. D. and Wang, R. Y. (2006). *Journey to Data Quality*, MIT Press, Cambridge.

Lee, Y. W., Strong, D. M., Kahn, B. K. and Wang, R. Y. (2002). AIMQ: a methodology for information quality assessment, *Information & Management* **40**(2): 133–146.

Lehman, A. (2005). *Jmp For Basic Univariate And Multivariate Statistics: A Step-by-step Guide*, SAS Press, Cary.

Leão, B. d. F., Costa, C. G. A. d., Silva, M. L. d. and Galvão, S. d. C. (2009). *Manual de Certificação para Sistemas de Registro Eletrônico em Saúde (S-RES)*. Disponível em: http://www.sbis.org.br/certificacao/Manual_Certificacao_SBIS-CFM_2009_v3-3.pdf. Acesso em: 13 fev. 2012.

Leveson, N. and Turner, C. (1993). An investigation of the Therac-25 accidents, *Computer* **26**(7): 18–41.

Martins, J. F., Rocha, J. G., Miranda, E. F., Sartor, M. C., Steckert, J. S., Steckert Filho, A., Guimarães, P. R. B. and Kotze, P. G. (2009). Análise da prevalência de entidades coloproctológicas nos pacientes idosos do serviço de coloproctologia de um Hospital Universitário, *Revista Brasileira de Coloproctologia* **29**(2): 145–157.

Massad, E., Marin, H. d. F. and Azevedo Neto, R. S. d. (2003). *O Prontuário Eletrônico do Paciente na Assistência, Informação e Conhecimento Médico*, São Paulo. Disponível em: http://www.sbis.org.br/site/arquivos/prontuario.pdf. Acesso em: 13 fev. 2012.

Maydanchik, A. (2007). *Data Quality Assessment*, Technics Publications, LLC, Bradley Beach.

McClanahan, C. (2008). Cleaning a formulation database using rule discovery technique, *Proceedings of the 13th ICIQ*, pp. 176–183.

Miles, R. and Hamilton, K. (2006). *Learning UML 2.0*, O'Reilly, Sebastopol.

Mitchell, T. M. (1997). *Machine Learning*, McGraw-Hill, New York.

Monteiro, J. (2010). Ruby on Rails Brasil. Disponível em: http://www.rubyonrails.pro.br/. Acesso em: 16 nov. 2010.

Netbeans.org (2012). Introdução ao Ajax para aplicações Web Java. Disponível em: http://netbeans.org/kb/docs/web/ajax-quickstart_pt_BR.html. Acesso em: 23 jan. 2012.

Oliveira, R. G., Faria, F. F., Lima Junior, A. C. B., Rodrigues, F. G., Andrade, M. M. d. A., Gomes, D. M. B. M., Neves, P. M., Constantino, J. R. M., Braga, u. C. G., Ferreira, R. M. S., Alvarenga, I. M., Lanna, D. d., Teixeira, R. G., Valle Junior, H. N., Leite, S. M. O., Costa, L. M. P., Silva, I. G. and Cruz, G. M. G. d. (2010). Cirurgias êntero-colorretais: abordagem cirúrgica de 129 pacientes do SUS no programa de pós-graduação Sensu Lato em coloproctologia, *Revista Brasileira de Coloproctologia* **30**(3): 333–343.

Olson, J. (2003). *Data quality: the accuracy dimension*, Morgan Kaufmann, San Francisco.

Omara, E., El Said, T. and Mousa, M. (2011). Employing Neural Networks for Assessment of Data Quality with Emphasis on Data Completeness, *ICGST International Journal on Artificial Intelligence and Machine Learning, AIML* **11**(1): 21–28.

Oppliger, R. (2009). *SSL and TLS: theory and practice*, Artech House, Norwood.

Perrotta, P. (2010). *Metaprogramming Ruby: Program Like the Ruby Pros*, Pragmatic Bookshelf, Raleigh.

Pfleeger, S. (2004). *Engenharia de Software: Teoria e Prática*, 2 edn, Prentice Hall, São Paulo.

Pipino, L. L., Lee, Y. W. and Wang, R. Y. (2002). Data Quality Assessment, *Communications of the ACM* **45**(4): 211–218.

Pollock, J. (2004). *JavaScript: a beginner's guide*, 2 edn, McGraw-Hill, New York.

Porteneuve, C. (2010). *Pragmatic Guide to Javascript*, The Pragmatic Bookshelf, Raleigh.

Porto, C. C. (1982). *Exame Clínico*, Guanabara Koogan, Rio de Janeiro.

Powell, T. A. (2008). *Ajax: the complete reference*, McGraw-Hill, New York.

Pressman, R. S. (2011). *Engenharia de software: uma abordagem profissional*, 7 edn, AMGW Editora Ltda, Porto Alegre.

Prudente, A. C. L., Torres Neto, J. d. R., Santiago, R. R., Mariano, D. R. and Vieira Filho, M. C. (2009). Cirurgias proctológicas em 3 Anos de serviço de coloproctologia: série histórica, *Revista Brasileira de Coloproctologia* **29**(1): 71–76.

Pyle, D. (1999). *Data preparation for data mining*, Morgan Kaufmann, San Francisco.

Queiroz, F. L., Côrtes, M. G. W., Rocha Neto, P., Alves, A. C., Freitas, A. H. A., Lacerda Filho, A., Neiva, A. M., Hanan, B., Côrtes, B. G. W., Bechara, C. d. S., Maia Junior, C. L. S., Fernandes, C. K. M., Mansur, E. S., Cruz, G. M. G. d., Silva, H. A., Mendonça, I. A., Vasconcelos, J. B., Figueiredo, J. A., Sena, K. A., Maciel, L., Costa, L. P., Luz, M. M. P. d., Santos, M. A. M. d., Carmona, M. Z., Maranhão, R. P., Paiva, R. d. A., Silva, R. G. d., Leite, S. M. d. O., Oliveira, T. A. d. N., Silva, T. B. d., Alves Filho, V. and Lamounier, P. C. d. C. (2010). Resultados do registro de cirurgias colorretais videolaparoscópicas realizadas no estado de Minas Gerais - Brasil de 1996 a 2009, *Revista Brasileira de Coloproctologia* **30**(1): 61–67.

Raś, Z. and Dardzinska, A. (2009). *Advances in Data Management*, Studies in Computational Intelligence, Springer, Heidelberg.

Redman, T. C. (1997). *Data Quality for the Information Age*, 1 edn, Artech House, Inc., Norwood.

Rezende, D. A. (2005). *Engenharia de software e sistemas de informação*, 3 edn, Brasport, Rio de Janeiro.

Ruby, S., Thomas, D. and Hanson, D. H. (2010). *Agile Web Development with Rails*, 4 edn, Pragmatic Bookshelf, Raleigh.

Ruby Visual Identity Team (2010). Ruby: A Programmer's Best Friend. Disponível em: http://www.ruby-lang.org/pt/sobre-o-ruby/. Acesso em: 15 nov. 2010.

Shortliffe, E. and Cimino, J. (2006). *Biomedical informatics: computer applications in health care and biomedicine*, 3 edn, Springer, New York.

Silva, O. J. (2001). *XML: Aplicações Práticas*, Érica, São Paulo.

Sommerville, I. (2007). *Software engineering*, 2 edn, Addison-Wesley, São Paulo.

Stallings, W. (2002). *Cryptography and Network Security: Principles and Practice*, 3 edn, Prentice Hall, Englewood Cliffs.

Stanger, J., Lane, P. and Crothers, T. (2002). *CIW: security professional : study guide*, Sybex, Alameda.

Steiner, M. T. A., Soma, N. Y., Shimizu, T., Nievola, J. C. and Steiner Neto, P. J. (2006). Abordagem de um problema médico por meio do processo de KDD com ênfase à análise exploratória dos dados, *Gestão & Produção* **13**(2).

Stewart, J., Tittel, E. and Chapple, M. (2008). *CISSP: Certified Information Systems Security Professional Study Guide*, John Wiley & Sons, Indianapolis.

Stinson, B. (2001). *PostgreSQL essential reference*, New Riders, Indianapolis.

Strong, D. M., Lee, Y. W. and Wang, R. Y. (1997). Data quality in context, *Commun. ACM* **40**(5): 103–110.

The Mozilla Foundation (2007). Mozilla Developer Center. Disponível em: https://developer .mozilla.org/en/Introduction_to_SSL. Acesso em: 06 dez. 2010.

The Mozilla Foundation (2010a). Mozilla Developer Center. Disponível em: https://developer .mozilla.org/en/About_JavaScript. Acesso em: 16 nov. 2010.

The Mozilla Foundation (2010b). Mozilla Developer Center. https://developer.mozilla.org /en/JavaScript/Guide/Details_of_the_Object_Model#Class-Based_vs._Prototype-Based_Languages. Acesso em: 16 nov. 2010.

The PostgreSQL Global Development Group (2011). PostgreSQL 9.1.2 Documentation, *Technical documentation*, The PostgreSQL Global Development Group. Disponível em: http://www.postgresql.org/docs/manuals. Acesso em: 02 fev. 2012.

Thomas, D., Fowler, C. and Hunt, A. (2009). *Programming Ruby 1.9: The Pragmatic Programmers' Guide*, Pragmatic bookshelf, Raleigh.

Thomas, S. (2000). *SSL & TLS essentials: securing the Web*, Wiley, New York.

Torres Neto, J. d. R., Souza, M. C. A. d. S., Santiago, R. R. and Prudente, A. C. L. (2008). Cirurgias colorretais no Hospital Universitário da Universidade Federal de Sergipe: três anos de criação do serviço de coloproctologia (série histórica), *Revista Brasileira de Coloproctologia* **28**(1): 77–83.

Tremblay, M. C., Dutta, K. and Vandermeer, D. (2010). Using Data Mining Techniques to Discover Bias Patterns in Missing Data, *Journal of Data and Information Quality* **2**(1): 1–19.

Valarini, R. and Campos, F. G. C. M. d. (2008). Resultado do registro nacional brasileiro em vídeo-cirurgia colorretal em 2007, *Revista Brasileira de Coloproctologia* **28**(2): 145–155.

Veiga, A. K., Saraiva, A. M. and Cartolano, E. A. (2011). Methods and tools to improve data quality in biodiversity specimens-occurrence data, *European Federation for Information Technologies in Agriculture, Food and the Environment (EFITA)*, Prague, pp. 471–480.

Voltolini, R. F. (2006). *Discretização e geração de gráficos de dados em aprendizado de máquina*, Master's thesis, USP - São Carlos.

Wand, Y. and Wang, R. Y. (1996). Anchoring data quality dimensions in ontological foundations, *Commun. ACM* **39**(11): 86–95.

Wang, R. Y. (1996). Beyond accuracy: What data quality means to data consumers, *J. Manage. Inf. Syst.* **12**(4): 5–33.

Wang, R. Y. (1998). A product perspective on total data quality management, *Commun. ACM* **41**(2): 58–65.

Wechsler, R., Anção, M. S., de Campos, C. J. R. and Sigulem, D. (2003). A Informática no Consultório Médico, *Jornal de Pediatria* **79**: S3–S12.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, 2 edn, Elsevier, San Francisco.

Worsley, J. and Drake, J. (2002). *Practical PostgreSQL*, O'Reilly & Associates, Sebastopol.

Zalewski, W., Lee, H. D., Caetano, A. M. J. F., Lorena, A. C., Maletzke, A. G., Fagundes, J. J., Coy, C. S. R. and Wu, F. C. (2008). Evaluation of models for the recognition of hadwritten digits in medical forms, *Advances in Bioinformatics and Computational Biology*, Vol. 5167, Springer, Germany, pp. 178–181.

Apêndice A

Avaliação Qualitativa de Qualidade de Dados

PESQUISA SOBRE QUALIDADE DOS DADOS ORIUNDOS DO SISTEMA PARA GERENCIAMENTO DE DADOS DE CIRURGIA COLOPROCTOLÓGICA

Objetivo: Avaliar as características do sistema de informação utilizado e da informação por ele gerenciada.

Seção A: Qualidade da Informação

	Considerando a qualidade da informação obtida pelos sistemas de informação, para cada afirmação marque a opção que caracteriza o item avaliado. O intervalo da escala de valores varia de "0" (De modo nenhum) a "10" (Completamente).	**De modo nenhum ←→ Completamente**										
		0	1	2	3	4	5	6	7	8	9	10
01	A informação é facilmente recuperável.											
02	A informação é facilmente acessível.											
03	A informação é completa para nossas necessidades.											
04	A informação é rapidamente obtida.											
05	A quantidade de informações é suficiente para nossas necessidades.											
06	A informação é confiável.											
07	A informação é de credibilidade duvidosa.											
08	A quantidade de informação é baixa.											
09	A informação tem profundidade suficiente para nossas necessidades.											
10	A informação é representada em um formato consistente.											
11	As informações sobre um mesmo tópico são agrupadas de forma consistente.											
12	A quantidade de informação é alta.											
13	A informação é verossímil.											
14	A informação está adequadamente organizada.											
15	A informação é facilmente inserida.											
16	A informação é facilmente removida.											
17	A informação é facilmente alterada.											
18	A informação obtida por meio do sistema é correta.											
19	A informação é suficientemente completa para nossas necessidades.											
20	A informação é inserida de forma adequada no sistema.											
21	A informação obtida por meio do sistema é precisa.											
22	A informação é apresentada de modo conciso.											
23	A informação é facilmente interpretada.											
24	As unidades de medida para a informação são claras.											
25	A informação é útil para nosso trabalho.											
26	A informação é apropriada para nosso trabalho.											
27	A informação é protegida contra acessos não autorizados.											
28	A informação é passível de dano.											
29	A informação é essencial para nosso trabalho.											

30	A informação é fácil de entender.											
31	O significado da informação é fácil de entender.											
32	O acesso a esta informação é suficientemente restrito.											
33	A informação é baseada em fatos.											
34	A informação é objetiva.											
35	A informação é fácil de compreender.											
36	A informação apresenta uma visão imparcial.											

Seção B: Características do Sistema de Informação

	Considerando a usabilidade do sistema de informação, para cada afirmação marque a opção que caracteriza o item avaliado. O intervalo da escala de valores varia de "0" (De modo nenhum) a "10" (Completamente).	**De modo nenhum ←→ Completamente**										
		0	1	2	3	4	5	6	7	8	9	10
37	O sistema é intuitivo.											
38	O sistema é facilmente acessado por meio de diferentes computadores.											
39	O sistema relata erros no preenchimento dos formulários.											
40	O sistema oferece mecanismos de busca facilmente utilizáveis.											
41	O sistema é de fácil uso.											

	Considerando a importância do sistema de informação para auxiliar na atividade a que foi destinado, marque a opção que caracteriza o item avaliado.	**De modo nenhum ←→ Completamente**										
		0	1	2	3	4	5	6	7	8	9	10
42	O sistema de informação pode contribuir significativamente com o gerenciamento de dados de cirurgia coloproctológica.											

	Considerando o sistema em termos gerais, marque a opção que caracteriza o item avaliado.	**Péssimo ←→ Excelente**										
		0	1	2	3	4	5	6	7	8	9	10
43	Como você avalia o sistema de informação?											

Seção C: Informações do Entrevistado

Informações pessoais	
44	Área de atuação: ( ) Computação ( ) Saúde
45	Tempo de atuação na respectiva área: ( ) Menos de 1 ano ( ) 1 ano ( ) 2 anos ( ) 3 anos ( ) 4 anos ( ) 5 ou mais anos
46	Idade:
47	Sexo: ( ) Masculino ( ) Feminino

Tempo despendido no preenchimento do formulário: _____ min.

Seção D: Sugestões e críticas

Sugestões e críticas