Universidade Federal do Pampa

Mateus Henrique Dal Forno

Evolução de Software através de Reengenharia: um processo didático

Alegrete

2014

Mateus Henrique Dal Forno

Evolução de Software através de Reengenharia: um processo didático

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Prof. Me. Sam da Silva Devincenzi

Coorientador: Prof[a]. Ma. Andréa Sabedra Bordin

Alegrete

2014

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

D136e Dal Forno, Mateus Henrique
Evolução de Software através de Reengenharia: um processo didático / Mateus Henrique Dal Forno.
118 p.

Trabalho de Conclusão de Curso(Graduação)-- Universidade Federal do Pampa, ENGENHARIA DE SOFTWARE, 2014.
"Orientação: Sam da Silva Devincenzi".

1. Engenharia de Software. 2. Processo de Software. 3. Evolução de Software. 4. Reengenharia. I. Título.

Mateus Henrique Dal Forno

Evolução de Software através de Reengenharia: um processo didático

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

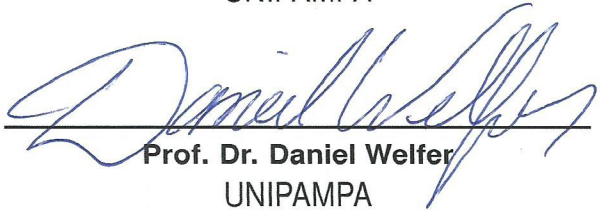
Trabalho de Conclusão de Curso defendido e aprovado em 19 de março de 2014.

Banca examinadora:

Prof. Me. Sam da Silva Devincenzi
Orientador

Profª. Ma. Andréa Sabedra Bordin
Coorientadora

Prof. Dr. Alessandro Gonçalves Girardi
UNIPAMPA



Prof. Dr. Daniel Welfer
UNIPAMPA

Dedico este trabalho aos meus pais,
que sempre me forneceram apoio incondicional
em minha trajetória acadêmica.

Agradecimentos

A Deus em primeiro lugar que me ilumina e me dá forças em minha caminhada.

Ao meu pai Irani e minha mãe Rosana pelo exemplo de vida e amor que diariamente dedicam em sua profissão, que apesar da importância infelizmente está cada vez mais sendo desvalorizada. Agradeço a minha mãe também pelos quatro anos em que fui seu aluno, que com certeza fizeram muita diferença em minha vida. Apesar da distância sempre estiveram presentes em minha vida e apoiando nos momentos de dificuldade.

As minhas irmãs Ana Júlia e Luisa Giovana e demais familiares: Bisavó Ana, meus avós: Francisco, Lurdes, Fiorentino (*sempre lembrado*) e Alzira, meus tios e primos. Nestes quatro anos em que estive distante do convívio de vocês realmente consegui entender o real significado do termo “Família”.

Aos amigos que me distanciei nos últimos anos em função da distância. Aos amigos que conheci aqui em Alegrete. Aos meus colegas de curso: Bruno, Helison, Nasser, Rafhael, Thiago e demais com quem convivi quase que diariamente nestes últimos anos.

Aos multiplicadores do Núcleo de Tecnologia da Educação de Ijuí, professor Gilmar e professora Maristela, com os quais convivi durante o meu estágio em 2009 e que me incentivaram a vir a Alegrete, apesar das dúvidas e incertezas em abandonar um curso de Ciência da Computação no segundo semestre e hoje provavelmente teria me arrependido caso não tivesse seguido suas opiniões.

O agradecimento especial a todas as pessoas envolvidas na realização deste trabalho: aos alunos da disciplina de Evolução de Software do segundo semestre letivo de 2013, por contribuírem executando o processo e desenvolvendo as atividades presentes no mesmo. Ao professor Sam pela paciência, dedicação e auxílio na orientação neste trabalho. A professora Andréa pela dedicação e disponibilidade em auxiliar na construção deste trabalho como co-orientadora, nas várias reuniões para discussão e aprimoramento do processo. Ao professor Cristiano pelas valiosas sugestões e contribuições para o desenvolvimento deste trabalho.

Por fim, gostaria de agradecer aos professores da área de computação do campus Alegrete da Universidade Federal do Pampa, pela dedicação empregada e pelos ensinamentos passados em suas aulas, em especial as professoras Vanessa, Márcia e Amanda, pela fundamental orientação recebida nos projetos de ensino e extensão que contribuíram para a minha formação acadêmica.

“Quando os ventos de mudança sopram,
umas pessoas levantam barreiras,
outras constroem moinhos de vento”.
(Érico Veríssimo)

Resumo

Este trabalho apresenta a construção de uma proposta de processo para a realização de evolução de sistemas legados, utilizando-se a técnica de reengenharia. Buscou-se na elaboração do processo criar estratégias para facilitar o aprendizado das práticas de evolução de software e a compatibilidade do mesmo com a aplicação prática na disciplina de Evolução de Software, ofertada aos alunos no sexto semestre do curso de Engenharia de Software. O processo é composto de uma etapa inicial de avaliação do sistema legado sob perspectiva técnica e de negócio, que possui o objetivo de avaliar a viabilidade da realização de reengenharia ou de outra estratégia de manutenção de sistemas legados. A segunda etapa, baseada no modelo ferradura, é a etapa onde ocorre a análise do sistema legado, a análise e projeto do novo sistema e a implementação e implantação do novo sistema. Optou-se por uma abordagem incremental para o desenvolvimento do novo sistema evoluído, baseada no desenvolvimento completo de pequenos incrementos. O processo foi aplicado na disciplina de Evolução de Software, no segundo semestre letivo de 2013, e utilizou-se o Sistema de Gerenciamento de Concursos Públicos (GCP) para a execução das atividades do processo. Para a avaliação da aplicação do processo na disciplina utilizou-se um questionário composto por questões relacionadas à avaliação do processo e também por questões que avaliavam a aprendizagem obtida pelos alunos através da realização das atividades práticas na disciplina. A análise dos resultados obtidos através da avaliação evidencia que houve uma boa aceitação da proposta e que houve coerência da proposta com a disciplina. Adicionalmente, constatou-se que houve aprendizado significativo de reengenharia através da realização das atividades práticas.

Palavras-chave: Engenharia de software. Processo de software. Evolução de software. Reengenharia.

Abstract

This work presents the construction of a proposed process for performing evolution of legacy systems, using the technique of reengineering. Sought in drafting the process create strategies to facilitate the learning of software development practices and the same compatibility with the application practice in the discipline of Software Evolution, offered to students in the sixth semester of the course of Software engineering. The process consists of an initial stage of evaluation of the legacy system under technical and business perspective, with the objective of evaluating the feasibility of realization of reengineering or another legacy systems maintenance strategy. The second step, based on the horseshoe model, is the step where the legacy system analysis, the analysis and design of the new system and the implementation and deployment of the new system. We opted for an incremental approach to the development of the new system evolved, based on the complete development of small increments. The process was applied in the discipline of Software Evolution, in the second half of 2013 school, and used the Sistema de Gerenciamento de Concursos Públicos (GCP) for the execution of the process activities. For the evaluation of the implementation of the process in the discipline it was used a questionnaire consists of questions related to the evaluation of the process and also for questions that evaluated the learning obtained by students through the provision of practical activities in the discipline. The analysis of the results obtained by evaluating evidence that there was a good acceptance of the proposal and that there was coherence of the proposal with the discipline. Additionally, it was noted that there were significant reengineering learning through the provision of practical activities.

Key-words: Software engineering. Software process. Software evolution. Reengineering.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Na década de 50 as pessoas não imaginavam que o software passaria a ser tão importante e que iria permitir muitos dos avanços em várias áreas, da forma que vemos hoje. O software tornou-se a força motriz para a popularização do computador pessoal (PRESSMAN, 2011).

Inicialmente, as grandes empresas e corporações não possuíam sistemas para o controle e gerenciamento das informações. Sentiu-se necessário a criação de uma tecnologia que tornasse eficiente e adequado o uso de tais informações, surgindo assim os sistemas de informação (BEZERRA, 2007). A partir de então, várias empresas passaram a informatizar os seus processos internos, o que permitiu ganhos de produtividade e um controle mais preciso das informações.

Os sistemas eram construídos de maneira informal, ou seja, não eram submetidos a um processo de desenvolvimento. Importantes projetos sofriam atrasos, as soluções não eram confiáveis, eram de difícil manutenção (devido à inexistência de documentações e de padrões de desenvolvimento) e o desempenho desses sistemas era insatisfatório. Essa situação foi denominada como crise de software (SOMMERVILLE, 2007).

Para contornar essa crise, surgiram técnicas de controle e processos de desenvolvimento de sistemas. No ano de 1968 foi realizada uma conferência para discussão de alternativas, onde surgiu o conceito de engenharia de software. A ideia era encarar o desenvolvimento de sistemas da mesma forma que as demais engenharias, de forma controlável e sistemática. A partir de então passaram a surgir metodologias e processos para o desenvolvimento de software.

Foram grandes os avanços na área de engenharia de software. Existem hoje diversas metodologias para o desenvolvimento de software e várias ferramentas que auxiliam neste processo (SOMMERVILLE, 2007). O programador solitário foi substituído por equipes de especialistas, onde cada membro da equipe desempenha um papel específico no desenvolvimento de sistemas (PRESSMAN, 2011). Apesar disso, ainda existem muitas empresas que não utilizam de forma adequada os processos de desenvolvimento de software.

Com o passar do tempo, o software sofre modificações para se adequar as necessidades do cliente. Porém, muitas vezes o sistema possui projeto de característica não expansível, codificação complexa e confusa, ausência de testes e registro dos mesmos, falta de documentação ou documentação desatualizada (PRESSMAN, 2011).

O software, por natureza, necessita mudar para que continue sendo útil. Podem

ocorrer mudanças de requisitos, pode ser necessária a correção de erros descobertos em fase de operação e podem surgir mudanças para adaptação a plataformas ou para melhorar o desempenho.

A área de evolução de software é responsável pela manutenção de um sistema, pela reengenharia, pela refatoração e pelo gerenciamento de sistemas legados. A evolução de um software é importante, pois as empresas são dependentes dos sistemas que investiram e utilizam (SOMMERVILLE, 2011).

A reengenharia de software, por sua vez, é uma técnica de evolução de software que envolve diversas atividades, que vão desde a atualização de documentação até a reformulação da estrutura do sistema e da codificação.

Apesar da importância que a evolução de software possui, os processos de desenvolvimento de software existentes não foram planejados pensando-se futuramente na necessidade de reengenharia de sistemas, sendo que a aplicabilidade dos mesmos não produz resultados satisfatórios. Devido a isso, a manutenção de um sistema é encarada como sendo uma tarefa adicional, e não como sendo parte de um processo de desenvolvimento de software (PEREZ-CASTILLO et al., 2013).

No documento "*Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*", proposto conjuntamente pela IEEE Computer Society; ACM (2004), definiu-se recomendações de currículos de cursos de graduação em Engenhara de Software, sendo que neste documento é proposto que durante o curso, deve-se dedicar ao menos 10 horas para abordagem de evolução de software.

Especificamente no curso de Engenharia de Software na UNIPAMPA, dedica-se 30 horas ao estudo do assunto, através da disciplina de Evolução de Software, que é oferecida aos acadêmicos no sexto semestre do curso e possui como objetivos "conhecer os fundamentos, técnicas e processos de evolução de software para que seja possível gerenciar a evolução de sistemas legados" (Universidade Federal do Pampa, 2012, p. 59). Tal disciplina é composta por 2 créditos teóricos. Na primeira oferta da disciplina, realizada no segundo semestre letivo de 2012, sentiu-se a necessidade da realização de exercícios práticos de evolução de software, apoiados por um processo que norteasse a prática de evolução de forma didática.

1.1 Objetivos

É crescente a busca por metodologias de ensino-aprendizagem que incentivem a proatividade, a iniciativa, a autoaprendizagem e o trabalho em equipe. Especialmente no curso de Engenharia de Software da Unipampa, há uma preocupação em realizar a interação entre os conteúdos teóricos e as atividades práticas, aproximando o ambiente

acadêmico das situações reais do mercado de trabalho (Dal Forno et al., 2012), (CERA; Dal Forno; Gindri Vieira, 2012).

Neste contexto, o objetivo deste trabalho é a criação de um novo processo de reengenharia de sistemas legados, com base em processos existentes, para apoiar às práticas de reengenharia na disciplina de Evolução de Software.

A prática de evolução de software através da aplicação de reengenharia justificou-se pelo fato de ser uma das técnicas que envolve boa parte das demais atividades de evolução (engenharia reversa, reestruturação de documentação, reestruturação de código, etc.) e pela necessidade dos egressos do curso obterem conhecimentos práticos desta importante área.

Para atingir o objetivo geral proposto neste trabalho é fundamental alcançar os seguintes objetivos específicos:

- Realizar um estudo dos processos e técnicas de reengenharia existentes;
- Elaborar um processo de reengenharia baseado nos processos de reengenharia existentes;
- Validar o processo, através da utilização do mesmo para as práticas de reengenharia na disciplina de evolução de software;
- Avaliar a utilização do processo de reengenharia na disciplina.

1.2 Organização do trabalho

Este trabalho está organizado da seguinte forma: no Capítulo 2 realiza-se a revisão bibliográfica de conceitos relacionados a este trabalho. O Capítulo 3 descreve alguns trabalhos relacionados ao tema deste trabalho e apresenta trabalhos correlatos. O Capítulo 4 descreve a metodologia da realização deste trabalho, incluindo o processo didático proposto, juntamente com o relato da execução do processo na disciplina de Evolução de Software. Os resultados obtidos a partir da aplicação do processo na disciplina são descritos no Capítulo 5. Finalmente, o Capítulo 6 apresenta algumas considerações finais e trabalhos futuros, acompanhado das referências utilizadas.

2 Fundamentação Teórica

Neste capítulo apresenta-se a revisão literária de conceitos e conteúdos, servindo de base para o desenvolvimento deste trabalho. A seção 2.1 descreve conceitos relacionados a processos de software, retomando especificamente os processos "Modelo em Cascata", "Modelo Incremental" e os "Métodos Ágeis". Por fim, a seção 2.2 aborda conceitos relacionados à Evolução de Software.

2.1 Processo de software

De acordo com Sommerville (2011, p. 18), "um processo de software é um conjunto de atividades relacionadas que levam à produção de um produto de software". Sua estrutura orienta as ações da equipe de desenvolvimento de software, desta forma estruturando, elucidando, controlando e aprimorando suas atividades. As atividades que compõem o processo são ordenadas, envolvendo restrições e recursos para que a saída de cada atividade seja conforme o planejado (PFLEEGER, 2004).

Pressman (2011) afirma que

> A base para a engenharia de software é a camada de *processos*. O processo de engenharia de software é a liga que mantém as camadas de tecnologia coesas e possibilita o desenvolvimento de software de forma racional e dentro do prazo. O processo define uma metodologia que deve ser estabelecida para a entrega efetiva de tecnologia de engenharia de software. O processo de software constitui a base para o controle do gerenciamento de projetos de software e estabelece o contexto no qual são aplicados métodos técnicos, são produzidos produtos derivados (modelos, documentos, dados, relatórios, formulários, etc.), são estabelecidos marcos, a qualidade é garantida e mudanças são geridas de forma apropriada. (PRESSMAN, 2011, p. 39, grifo do autor).

Os modelos de processos podem ser definidos de forma abrangente, englobando todas as atividades, e também podem ser definidos com enfoque em atividades específicas como, por exemplo, para desenvolvimento ou teste de software. Pode-se ainda definir subprocessos, tais como levantamento de requisitos, análise, projeto, programação, testes, etc. (REZENDE, 2005).

Um processo deve possuir uma documentação própria, contendo informações sobre que tarefa será realizada pelo processo, como será realizada (os passos necessários para a realização da tarefa), quem irá desenvolver as tarefas (papéis e atribuições), as informações necessárias e o resultado final produzido a partir da execução do processo.

Em relação a estrutura de processo, Sommerville (2011) descreve as seguintes atividades fundamentais como sendo comuns a um processo de engenharia de software:

> 1. *Especificação de software.* A funcionalidade do software e as restrições a seu funcionamento devem ser definidas.
> 2. *Projeto e implementação de software.* O software deve ser produzido para atender as especificações.
> 3. *Validação de software.* O software deve ser validado para garantir que atenda às demandas do cliente.
> 4. *Evolução de software.* O software deve evoluir para atender às necessidades de mudanças dos clientes. (SOMMERVILLE, 2011, p. 18, grifo do autor)

Existem vários modelos de processos de desenvolvimento de software amplamente descritos pela literatura, que possuem características próprias. Dentre estes podem ser citados o “modelo em cascata”, o “modelo incremental” e os “métodos ágeis”.

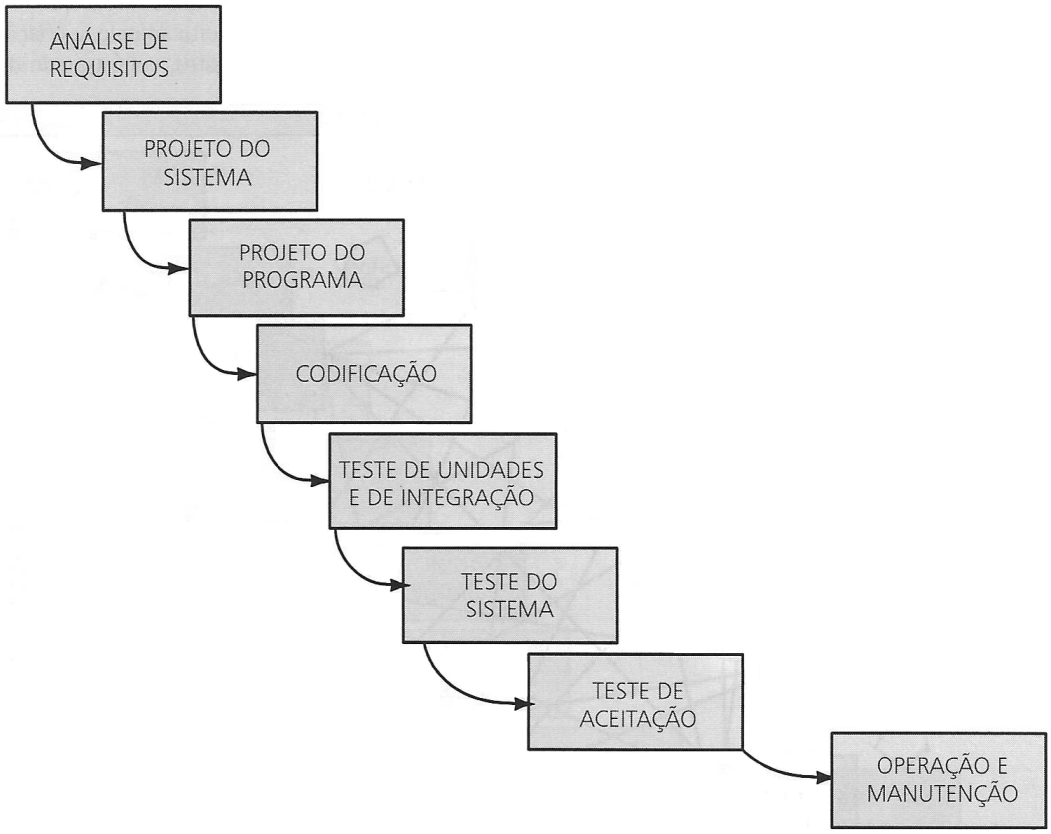
2.1.1 Modelo em cascata

O modelo em cascata possui sua estrutura organizada de forma sequencial e sistemática, ou seja, uma tarefa somente inicia após a conclusão da tarefa anterior. As tarefas envolvem a análise de requisitos, projeto do sistema, projeto do programa, codificação, teste de unidades e integração, teste do sistema, teste de aceitação e operação e manutenção, variando de acordo com o contexto onde encontra-se aplicado. A Figura 1 ilustra o processo.

Dentre as vantagens da utilização do modelo em cascata, Paula Filho (2011) cita a existência de pontos de controle ao final de cada tarefa, o que facilita a gestão do projeto e amplia a confiabilidade. Sommerville (2011) considera o modelo em cascata consistente e facilmente documentável, o que facilita o monitoramento do progresso do desenvolvimento do sistema.

Pfleeger (2004) pondera como desvantagem no modelo em cascata a inexistência de orientação de como tratar mudanças no produto e também no processo, que inevitavelmente irão ocorrer. Pressman (2011) descreve a existência de estados de bloqueio no processo, que ocorrem quando há dependência da conclusão de uma tarefa de um membro da equipe para que outro membro possa prosseguir com seu trabalho. Rezende (2005) afirma que a aplicação do modelo cascata engessa o desenvolvimento do software.

Figura 1 – Modelo em cascata.



Fonte: Pfleeger (2004, p. 39)

2.1.2 Modelo incremental

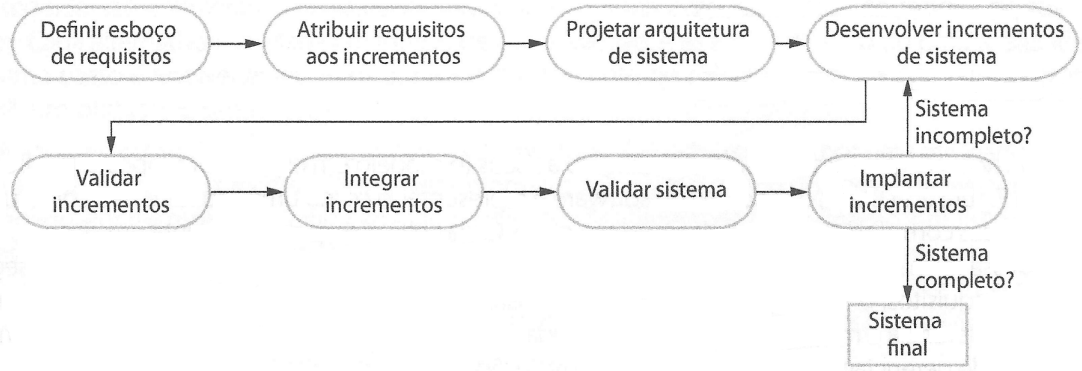
O modelo incremental é baseado em iterações de processo, ou seja, diferentemente do modelo em cascata, ocorrem iterações do processo, onde em cada uma destas iterações ocorre um incremento de funcionalidades no sistema. A Figura 2 ilustra o funcionamento do processo incremental.

Sommerville (2011) descreve o modelo incremental da seguinte forma:

> Em um processo de desenvolvimento incremental, os clientes identificam, em linhas gerais, os serviços a serem oferecidos pelo sistema. Eles identificam quais serviços são mais e menos importantes para eles. Uma série de incrementos de entrega são, então, definidos, com cada incremento proporcionado um subconjunto da funcionalidade do sistema. A atribuição de serviços aos incrementos depende da ordem de prioridade dos serviços – os serviços de mais alta prioridade são implementados e entregues em primeiro lugar. (SOMMERVILLE, 2011, p. 31).

Pressman (2011) considera o desenvolvimento incremental como sendo útil em casos que a equipe é de tamanho reduzido, pois pode-se definir durante o desenvolvimento

Figura 2 – Modelo incremental



Fonte: Sommerville (2011, p. 31)

a necessidade da inclusão de mais membros na equipe de desenvolvimento. Pfleeger (2004) cita como vantagem do processo incremental que uma nova versão do sistema pode incluir novos recursos, mas também pode incluir aprimoramentos nas funcionalidades de versões anteriores, a partir do *feedback* obtido pelo uso do sistema ainda em fase de produção.

2.1.3 Métodos ágeis

Atualmente, o desenvolvimento e entrega rápidos são considerados como fatores críticos para sucesso do desenvolvimento de um sistema, sendo que algumas empresas abrem mão da qualidade e de compromisso aos requisitos em troca da implementação e entrega do software em um menor prazo (SOMMERVILLE, 2011). O conceito de desenvolvimento ágil de software surgiu a partir do **manifesto ágil** no ano de 2001. O surgimento dos métodos ágeis, baseado nos princípios básicos do manifesto ágil, buscou tentar resolver problemas presentes na engenharia de software tradicional (PRESSMAN, 2011). Sommerville (2011) conceitua os métodos ágeis como sendo métodos incrementais de desenvolvimento, onde são realizados pequenos incrementos disponibilizados a cada duas ou três semanas, obtendo *feedback* rápido em relação ao sistema e seus requisitos.

Apesar das vantagens do uso dos métodos ágeis, deve-se avaliar o contexto do sistema antes da aplicação do processo. Os métodos ágeis são indicados para implementação de programas que irão operar em ambientes em que ocorrem mudanças rápidas nos requisitos e que necessitam de agilidade no desenvolvimento e entrega do sistema. Em sistemas críticos e de controle de segurança, que necessitam de especificações detalhadas, recomenda-se uma abordagem de processo direcionada a planos (SOMMERVILLE, 2011).

Sommerville (2011) considera como sendo o método ágil mais conhecido o *eX-*

treme Programming. Este método é baseado em cinco valores bases para o trabalho: *comunicação* efetiva entre a equipe, *simplicidade* (atendimento apenas as necessidades imediatas), *feeedback* do cliente e dos membros da equipe, *coragem* (relacionada à disciplina) e *respeito* entre os membros da equipe e outros membros envolvidos (PRESSMAN, 2011).

Além do *eXtreme Programming*, outro método ágil que vêm sendo utilizado é o *Scrum*. O processo é composto por um desenvolvimento incremental, composto por uma lista de trabalhos pendentes priorizados (*backlog*), que é composta pelos requisitos e alterações com valor estratégico ao cliente. A partir desta lista são selecionadas atividades a serem realizadas no ciclo de trabalho (*sprint*), que geralmente possui 30 dias, e que não será alterado (*backlogs* não são inseridos ou modificados em um *sprint* após o seu início). Após o início do ciclo de trabalho, são realizadas reuniões diárias de 15 minutos para o acompanhamento do andamento das atividades. Através da utilização do *Scrum*, obtém-se para a equipe de desenvolvimento do sistema um ambiente de trabalho de curto prazo e estável (PRESSMAN, 2011).

2.2 Evolução de software

A vida útil de um sistema não acaba com a realização de sua entrega (PFLEEGER, 2004). Um software, durante o seu ciclo de vida, necessita de evolução contínua. Podem surgir erros e falhas, mudanças nas regras de negócios da empresa, novos requisitos, etc.. A evolução de software é importante porque as organizações são muito dependentes dos sistemas que investiram e que utilizam (SOMMERVILLE, 2011).

A evolução de software está diretamente relacionada ao conceito de software legado. Um software pode ser considerado como legado quando encontra-se ainda em operação, foi desenvolvido no passado e sofreu modificações ao longo do tempo sem que recebesse melhorias estruturais e de forma sistemática (De Lucia; FASOLINO; POMPELLA, 2001).

A preocupação com o ciclo de vida de um sistema é foco de estudos desde a década de 60. Aplicações legadas foram desenvolvidas a décadas atrás e sofreram várias modificações a fim de se adequarem a mudanças de requisitos e de regras de negócio. Geralmente quando se fala em software legado a primeira característica que se destaca é a má qualidade. Aplicações legadas geralmente possuem código complexo, documentação pobre ou inexistente, ausência de registros de testes e o controle de modificações não é registrado (PRESSMAN, 2011).

Programas legados muitas vezes são indispensáveis as organizações que os utilizam, pois as mesmas são dependentes das informações ali armazenadas e manipuladas. Quando tal sistema necessita de mudanças contínuas e significativas, deve-se analisar a

viabilidade em seguir realizando manutenções ou então descartar o sistema e construir um sistema novo para substituí-lo. Pfleeger (2004) sugere algumas perguntas como instrumento de avaliação neste caso.

> O custo de manutenção é muito alto?
> A confiabilidade do sistema é inaceitável?
> O sistema não pode mais se adaptar a mudanças adicionais dentro de um período de tempo razoável?
> O desempenho do sistema ainda está fora das restrições prescritas?
> As funções do sistema têm utilidade limitada?
> Outros sistemas fazem o mesmo trabalho melhor, mais rápido ou gerando menos custos?
> O custo de manutenção do hardware é muito alto, a ponto de justificar sua substituição por um hardware novo e mais barato?(PFLEEGER, 2004, p. 384).

Manny Lehman, em seus estudos, observou o comportamento do crescimento e evolução de vários sistemas. A partir desta observação, resumiu seus estudos em 05 leis, conhecidas como **Leis da evolução de software** (PFLEEGER, 2004). A primeira delas se relaciona à mudança contínua. O software em uso, ou irá passar por mudanças contínuas, ou irá, de forma progressiva se tornando menos útil. Sommerville (2011, p. 169) acrescenta: “A primeira lei afirma que a manutenção de sistema é um processo inevitável”.

A segunda lei versa sobre o aumento da complexidade. Ao passo que um sistema passa por manutenções, sua estrutura vai aos poucos sendo degradada e sua complexidade aumenta. Sommerville (2011) considera necessárias as manutenções preventivas, a fim de aprimorar continuamente a estrutura do software, além de simplesmente adicionar funcionalidades. Consequentemente, haverão custos adicionais para a realização de tal manutenção.

A terceira lei indica que a dinâmica da estrutura definida inicialmente para o sistema influencia nas mudanças, podendo em alguns casos até restringir modificações em um sistema. Conforme o sistema vai crescendo, vai ficando mais complexo e difícil de ser compreendido, dificultando as modificações. Sommerville (2011) considera que pequenas mudanças graduais são mais viáveis, pois evitam a redução de confiabilidade do sistema. Com a realização de grandes mudanças, há uma tendência de surgirem muitos defeitos.

A quarta lei de Lehman é relativa a estabilidade organizacional. Lehman considera que, em grandes projetos, a ocorrência de mudanças de recursos ou de pessoal praticamente não afeta a evolução do sistema a longo prazo.

Na quinta lei, Lehman relaciona os incrementos do sistema juntamente com o surgimento de defeitos. Ao adicionar uma nova funcionalidade no sistema, também são adicionados proporcionalmente e de forma inevitável novos defeitos. Sommerville (2011) afirma que desta forma é necessário a cada *release* de sistema um novo *release* apenas para

correção de defeitos. Além disso, ao se realizar grandes incrementos de funcionalidade há a necessidade de se haver um orçamento prevendo correção de defeitos.

Em relação a forma em que se realiza a manutenção de um sistema, Sommerville (2011) define três tipos diferentes de manutenção de software. O primeiro é o de manutenção para reparo de defeitos de Software, cujo objetivo é a correção de erros, que podem ser de codificação (de custo mais barato), de projeto (mais caros por envolver reescrita de componentes) ou de requisitos (onerosos por envolverem o reprojeto do sistema existente). O segundo tipo de manutenção está relacionado a questões de adaptação de um software a um ambiente, que é necessária quando se altera, por exemplo, o hardware, o sistema operacional ou quando outro software de apoio muda. O terceiro tipo de manutenção envolve a modificação do sistema com o objetivo de adição ou modificação de funcionalidades, que ocorre pela alteração de requisitos ou em função de mudanças de negócio da organização.

Em relação ao terceiro tipo de manutenção de um sistema, Pfleeger (2004) exemplifica que ela pode ser necessária por dois tipos de necessidades. A primeira é através de uma necessidade do cliente, motivada por mudanças nas regras de negócio da empresa. A segunda é relativa a natureza do sistema, que é sujeito a fatores externos. Um exemplo deste caso pode ser um sistema que calcula deduções em folha de pagamento para uma empresa. Estes cálculos dependem de leis municipais, estaduais e federais e havendo modificações em alguma destas leis ou caso a empresa se mude para outro município ou abra uma filial, o sistema pode necessitar de manutenção, mesmo funcionando perfeitamente.

Em relação a evolução de sistemas legados, Sommerville (2011) descreve quatro estratégias pelas quais a organização que utiliza o sistema pode optar. A primeira delas é a de descartar o sistema, que deve ser escolhida quando o sistema não contribui de forma efetiva com os processos de negócio da empresa. A segunda opção é a de não alterar o sistema e prosseguir realizando manutenções regulares, que é a melhor escolha quando o sistema é necessário para a empresa, é estável e há poucas solicitações de mudanças. Como terceira alternativa, a empresa pode optar por realizar a reengenharia do sistema e aprimorar sua facilidade de manutenção, que é recomendada quando o sistema já está degradado devido as regulares mudanças e quando ainda há necessidades de mudanças. A última opção é a de substituir o sistema atual por um novo sistema, que é a melhor escolha quando há limitações de hardware no sistema antigo que impossibilitem que o mesmo siga em operação ou quando o custo para a reengenharia do sistema legado é muito superior ao custo de substituição do sistema.

3 Estado da Arte

A área de Evolução de Software é uma área onde ainda há muito o que ser explorado, especialmente no que tange a reengenharia de sistemas legados e também no contexto acadêmico, através de métodos e técnicas que facilitem o aprendizado.

Este capítulo apresenta trabalhos relacionados com a temática e o propósito deste trabalho. Tais trabalhos demonstram e descrevem os estudos que vêm sendo realizados na área de evolução de software e os resultados obtidos. Na seção 3.1 são descritos os processos de reengenharia propostos pela literatura. A seção 3.2 traz trabalhos com enfoque na manutenção de software. Na seção 3.3 estão descritos trabalhos aplicados a reengenharia de software. Por fim, a seção 3.4 demonstra trabalhos com enfoque na reengenharia em cursos de computação, através do uso de processos e ferramentas que apoiem e facilitem o aprendizado.

3.1 Processos de reengenharia

Sistemas legados mais antigos são de difícil compreensão, o que restringe possíveis mudanças. A reengenharia de software é responsável por reimplementar sistemas legados, com o objetivo de facilitar sua manutenção, podendo envolver desde a elaboração de nova documentação, até a reorganização da estrutura do sistema e conversão da linguagem de programação. Geralmente não se altera a funcionalidade do sistema nem a arquitetura.

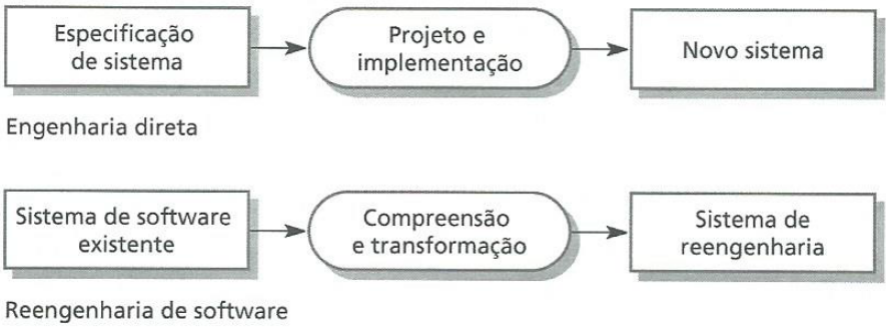
Processo proposto por Sommerville

O processo proposto por Sommerville (2007) cita duas vantagens da realização de reengenharia de sistemas em comparativo a outras abordagens mais radicais: a primeira é a de que o risco é reduzido, pois existe um alto risco no desenvolvimento de um software crítico de negócios. A segunda vantagem é em relação ao custo reduzido: o custo da reengenharia é significativamente menor comparando-se com o custo de um novo software.

Em sua proposta, a principal diferença entre se realizar a reengenharia de um software legado e o desenvolvimento de um novo software é que o próprio software legado passa a ser o ponto de partida para o desenvolvimento, ao invés de uma especificação escrita. A Figura 3 ilustra a diferença existente entre os processos. Na engenharia direta, a especificação do sistema é o ponto de partida e envolve o projeto e a implementação que gera como resultado um novo sistema. Na reengenharia de software o processo inicia-se

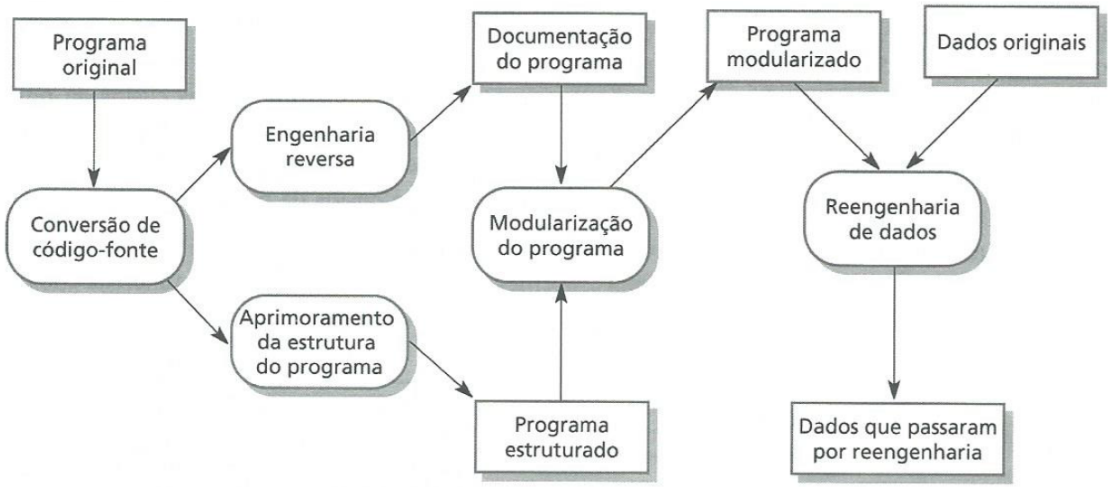
pelo software já existente e o processo de desenvolvimento baseia se na compreensão e transformação desse sistema, gerando como produto final um sistema de reengenharia.

Figura 3 – Diferenciação entre a engenharia direta e a reengenharia.



Fonte: Sommerville (2007, p. 332)

Figura 4 – Processo de reengenharia proposto por Sommerville (2007).



Fonte: Sommerville (2007, p. 332)

O processo de reengenharia proposto por Sommerville (2007) é descrito a seguir. A Figura 4 ilustra o processo de forma detalhada. O processo inicia a partir do sistema legado, que é a entrada para a atividade de **Conversão de código fonte**, onde o programa é convertido de uma linguagem de programação antiga para uma nova versão da mesma linguagem de programação ou então para uma outra diferente.

A partir da conversão do código fonte são realizadas 2 atividades distintas. A primeira delas é a de **Engenharia reversa**, cujo objetivo é extrair informações que facilitem a documentação do sistema. A outra atividade é a de **Aprimoramento da estrutura do**

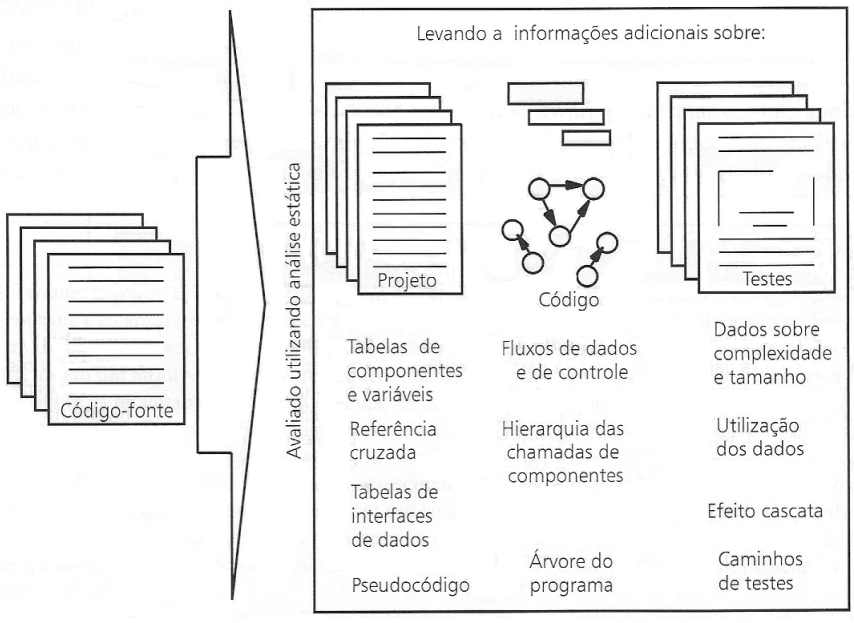
programa, onde ocorrem as modificações da estrutura do software, de forma a facilitar a leitura e a compreensão.

De posse da documentação do programa e do programa devidamente estruturado, passa-se para a etapa de **Modularização do programa**, cujo objetivo é o agrupamento de partes relacionadas do sistema, e a remoção de redundâncias, podendo envolver também transformações na arquitetura do sistema. A última etapa é a **Reengenharia de dados**, onde, a partir do programa modularizado, os dados originais são reorganizados com o objetivo de refletirem as mudanças ocorridas no programa.

Processo proposto por Pfleeger

O processo de rejuvenescimento de software proposto por Pfleeger (2004) considera como um desafio tentar aumentar a qualidade de um sistema já existente. A proposta de processo tem como foco analisar um sistema já existente, obtendo informações e as reformatando de forma compreensível, considerando os seguintes aspectos: redocumentação, reestruturação, engenharia reversa e reengenharia.

Figura 5 – Etapa de redocumentação proposta por Pfleeger (2004)



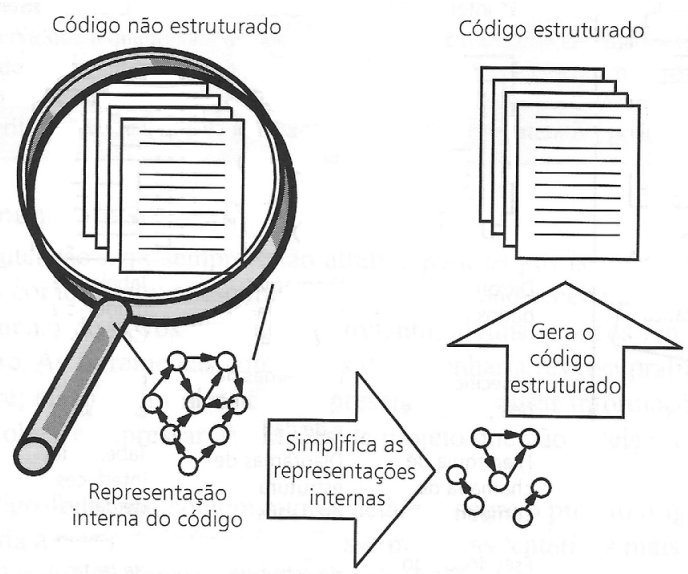
Fonte: Pfleeger (2004, p. 406)

A **redocumentação do sistema** acontece através da análise estática do código-fonte, com o objetivo de levantar informações que auxiliem na compreensão do código. Não há nenhuma transformação nesta etapa, apenas um levantamento de informações. São examinados: a utilização das variáveis, as chamadas de componentes, os caminhos

de controle, componentes, parâmetros, caminhos de teste e outras informações, facilitando o o entendimento de o que determinado código faz e de que forma. As informações obtidas auxiliam na avaliação da necessidade de reestruturação do sistema, refletindo a realidade atual do sistema (e não como o sistema deveria ser). Através da Figura 5 podemos visualizar o funcionamento do processo de redocumentação.

A etapa de **reestruturação** tem por objetivo facilitar o entendimento do software, para realização das modificações. São realizadas 3 atividades, conforme demonstra a Figura 6. Inicia-se pela análise estática, que fornece informações que são representadas como um grafo direcionado. Em uma segunda etapa, através de alguma ferramenta esta representação é refinada e simplificada. O processo de reestruturação é concluído com a geração de código estruturado, que é um código equivalente para o código do sistema já existente.

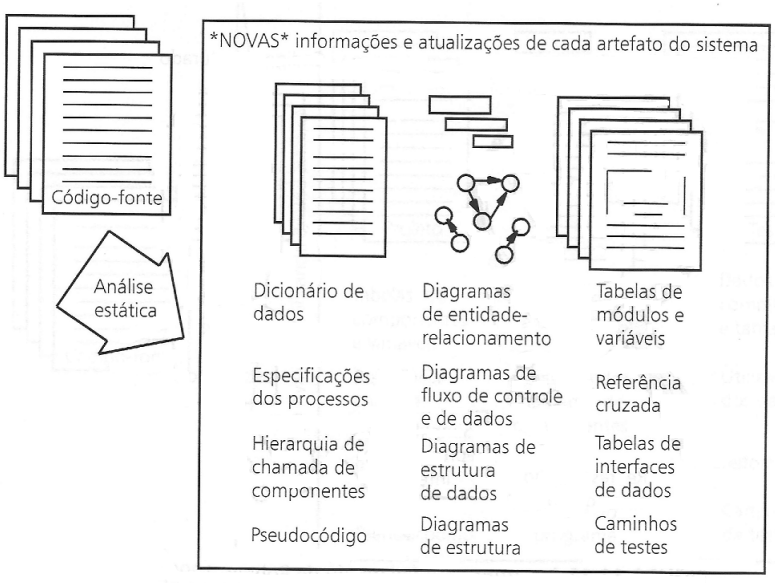
Figura 6 – Etapa de reestruturação proposta por Pfleeger (2004)



Fonte: Pfleeger (2004, p. 407)

Posterior a reestruturação do código é necessária a obtenção de informações relativas a especificação do projeto do software, a partir do código-fonte. Esta etapa é denominada como **engenharia reversa** e as informações obtidas são organizadas de forma que seja possível sua manipulação. Através da Figura 7, ilustra-se o processo de engenharia reversa. A partir do código-fonte a estrutura é interpretada e são identificadas as informações presentes, como por exemplo: dicionário de dados, diagramas, hierarquias, referências cruzadas, etc.

Figura 7 – Etapa de engenharia reversa proposta por Pfleeger (2004)



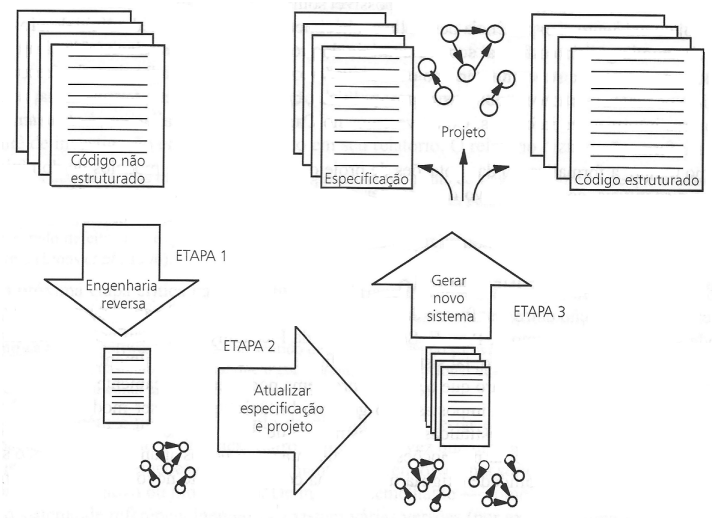
Fonte: Pfleeger (2004, p. 408)

A última etapa do processo proposto por Pfleeger (2004) é a **reengenharia**. Nesta etapa é produzido um novo código-fonte para o software, envolvendo atividades da seguinte forma: a partir da engenharia reversa é gerado o projeto do software. Este projeto é então atualizado, sendo corrigido e complementado a fim de refletir a especificação. A última etapa é a geração do novo sistema a partir do projeto, incluindo-se o software e toda a documentação do novo sistema, a partir das especificações e do projeto. A Figura 8 ilustra o processo de reengenharia.

Pfleeger (2004) acrescenta que o resultado do produto final está relacionado a diversos fatores:

- A(s) linguagem(ns) utilizadas no sistema;
- a interface do banco de dados existente;
- as interfaces que interagem com o usuário;
- as interfaces que se relacionam com os serviços do sistema e com outras linguagens de programação;
- a maturidade e estabilidade do domínio da aplicação;
- as ferramentas disponíveis no apoio ao processo.

Figura 8 – Etapa de reengenharia proposta por Pfleeger (2004)



Fonte: Pfleeger (2004, p. 409)

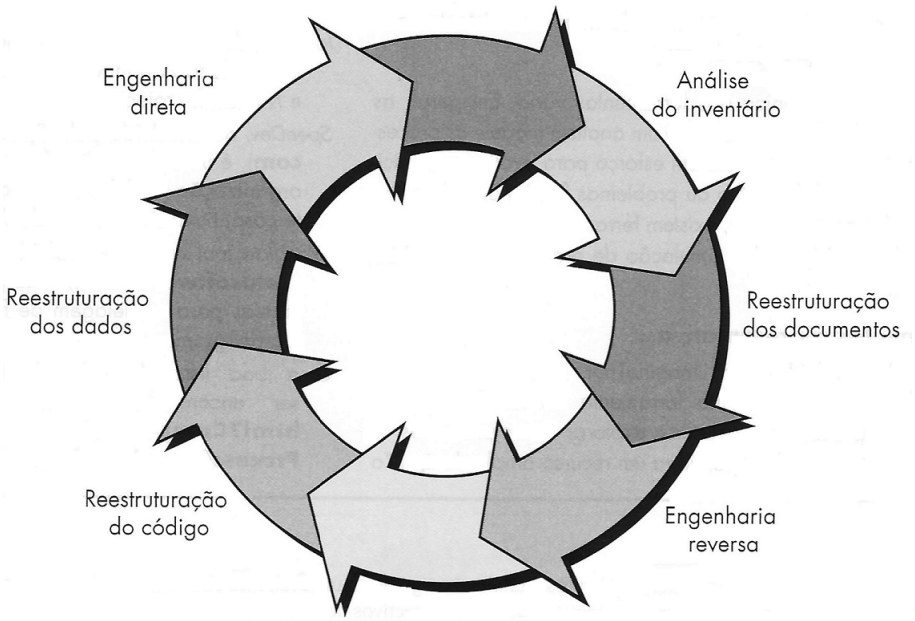
Processo proposto por Pressman

A proposta de processo de reengenharia de Pressman (2011) considera que há a necessidade de uma estratégia pragmática, pois a reengenharia de sistemas de informação tem um custo significativo e absorve recursos de TI durante muitos anos. Dessa forma, o processo é composto por um modelo de seis atividades cíclicas, sendo que cada atividade pode ser revisitada e o processo pode ser encerrado após a realização de qualquer uma das atividades. A Figura 9 representa o processo, que é composto das seguintes atividades: análise do inventário, reestruturação de documentos, engenharia reversa, reestruturação de código, reestruturação dos dados e engenharia direta.

A **análise de inventário** é a etapa em que é analisado o inventário de todas as aplicações utilizadas pela empresa, com o objetivo de analisar a criticidade, longevidade, manutenibilidade, dentre outras características, a fim de identificar possíveis aplicativos que necessitem de reengenharia. O inventário deve ser regularmente analisado, pois o estado das aplicações pode variar com o tempo e em função disso as prioridades podem ser alteradas.

A **reestruturação de documentos** tem como objetivo realizar ações em relação à documentação dos sistemas. Deve-se realizar a avaliação da documentação existente, verificando a viabilidade de realizar a atualização da documentação. Há casos em que o sistema está estático e sua vida útil está chegando ao fim, neste caso não é viável reestruturar a documentação. Também há situações em que um software possui partes que sofrem manutenções frequentes, então é viável atualizar a documentação apenas dessa parte crítica. Em casos em que o sistema é crítico para o negócio e precisa ser totalmente

Figura 9 – Modelo de reengenharia proposto por Pressman (2011)



Fonte: Pressman (2011, p. 668)

redocumentado é recomendável que a documentação seja limitada ao essencial.

A etapa de **engenharia reversa** tem por objetivo a recuperação do projeto do software, através análise da codificação, utilizando-se de ferramentas que extraem informações do projeto, da arquitetura e dos procedimentos existentes no programa.

A atividade de **reestruturação de código** envolve a análise do código-fonte por uma ferramenta de reestruturação, que analisa a estrutura do código a fim de identificar inconsistências. A partir disso o código é reestruturado, revisado e testado, com o objetivo de garantir que anomalias não sejam inseridas neste processo. Nesta etapa a documentação interna do código também é atualizada.

Em muitos casos os programas possuem estruturas de dados fracas, o que torna difícil adaptações e aperfeiçoamentos. Na etapa de **reestruturação dos dados** a arquitetura atual dos dados é minuciosamente analisada, identificando os objetos de dados e atributos, e a estrutura dos dados existentes é revisada com objetivo de ampliar a sua qualidade. Quando a estrutura dos dados é fraca, os dados passam por reengenharia.

A etapa de **engenharia direta** é a etapa do processo em que são recuperadas informações a respeito do software existente, e esta informação é utilizada com o objetivo de alterar ou reconstruir o sistema, aperfeiçoando a sua qualidade global, podendo também incluir a adição de novas funcionalidades e melhorias de desempenho.

Modelo Ferradura

O modelo Ferradura é um dos processos existentes para a reengenharia de sistemas legados. É baseado em três fases (SEACORD; PLAKOSH; LEWIS, 2003):

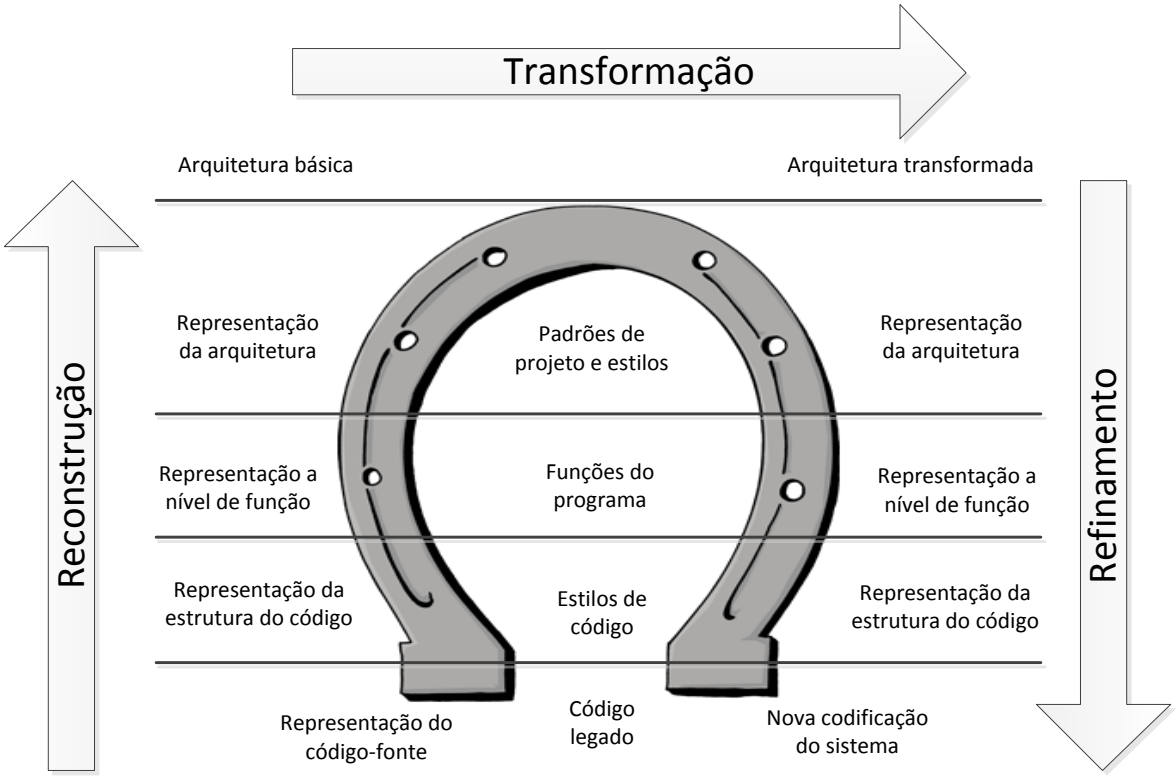
1. Reconstrução: envolve a reconstrução das descrições lógicas do sistema e dos artefatos existentes;

2. Transformação: envolve a transformação e melhoria das descrições lógicas do sistema;

3. Refinamento: envolve a implementação das mudanças na implementação do sistema;

O modelo ferradura é subdividido em três níveis de abstração, sendo que a execução completa perpassa por todos os níveis do processo. Pode-se também realizar apenas a execução de um ou mais níveis, dependendo da necessidade de mudanças no sistema. A Figura 10 ilustra o modelo ferradura.

Figura 10 – Modelo ferradura de reengenharia



Fonte: Adaptado de Seacord, Plakosh e Lewis (2003)

Em relação aos níveis de abstração, Seacord, Plakosh e Lewis (2003) descrevem os níveis de transformações de código, transformações funcionais e transformações de arquitetura da seguinte forma:

As atividades realizadas no nível de **Transformação de código** são geralmente realizadas de forma rápida e "suja". Este nível é normalmente associado com as atividades de manutenção do sistema do que com atividades de reengenharia.

As atividades realizadas no nível de **Transformação funcional** tem por objetivo a mudança de tecnologia, mantendo as funcionalidades já existentes no sistema inalteradas. Como exemplos de transformações funcionais pode ser citados a alteração do sistema legado para um paradigma orientado a objetos ou a mudança de tecnologia de banco de dados do sistema.

As atividades realizadas no nível de **Transformação de arquitetura** reestruturam a arquitetura do sistema, e em muitos casos envolvem a reconstrução completa de um sistema legado. Além disso, a transformação de arquitetura tende a possibilitar uma maior segurança, confiabilidade e desempenho do sistema, melhorando a sua qualidade.

3.2 Trabalhos aplicados a manutenção de software

Para esta seção do trabalho, selecionou-se três artigos que demonstram um pouco do que vem sendo pesquisado na área de manutenção de sistemas legados. Tais trabalhos possuem enfoque na pesquisa e uso de técnicas e/ou ferramentas com o objetivo de aumentar a qualidade da manutenção de sistemas legados.

Em Breivold, Chauhan e Babar (2010) realizou-se uma revisão sistemática, com o objetivo de obter uma visão geral acerca dos estudos existentes relacionados a evolução de programas de código aberto. Adicionalmente, a intenção dos autores com este trabalho era a de obter informações de como os programas de código aberto estão preparados para acomodar as alterações realizadas no decorrer do ciclo de vida. Como resultados desta pesquisa, a maioria dos resultados encontrados concentraram-se em analisar a evolução dos sistemas ao longo do tempo a partir de métricas definidas. Poucos trabalhos focaram em utilizar dados históricos para realizar estimativas futuras que poderiam prever esforço de manutenção e perspectivas econômicas na previsão de evolução de um software *open source*.

Kagdi, Gethers e Poshyvanyk (2011), a partir de estudos, propõem uma abordagem integrada de apoio à evolução de software, com o nome de "SE^2". A abordagem é composta de três atividades. A primeira atividade envolve a identificação da Localização da Funcionalidade a ser alterada ou do ponto de partida para uma mudança (FL). São recuperadas informações do software através do código-fonte e demais artefatos (requi-

sitos, relatório de *bugs*, projeto, etc.) e é realizada a recuperação de informação através de mineração de dados. A segunda etapa é a de Análise do Impacto que a alteração causa no sistema (IA). Esta atividade avalia a entidade em que é proposta a mudança e estima possíveis entidades relacionadas que sejam candidatas à alteração, gerando um conjunto estimado de entidades impactadas pela mudança. Concluindo, a Recomendação de Desenvolvedores também é considerada, obtida dos desenvolvedores experientes na implementação de mudanças no sistema (DR). As informações obtidas passam por uma indexação semântica, com o objetivo de localizar e classificar unidades relevantes na codificação.

A avaliação da aplicabilidade deste modelo foi realizada para as atividades de IA e DR. Em relação a IA, a partir da avaliação de vários sistemas de código aberto, a realização desta atividade gera uma melhoria de até 20%. Em DR, por sua vez, obteve-se que esta abordagem superou outras abordagens semelhantes, de forma substancial. Este estudo ainda necessita de mais investigação, especialmente porque as atividades foram estudadas de forma independente.

Ao se realizar alterações no código-fonte, é necessário que se tenha a garantia de que a alteração é propagada para entidades afins, como forma de evitar a introdução de erros. Malik e Hassan (2008) consideram que a precisão da propagação de mudanças é essencial para que haja sucesso na evolução de aplicativos grandes e complexos. Como alternativa de controle de propagação de mudanças, Malik e Hassan (2008) propõem duas heurísticas adaptativas de propagação de mudanças. A primeira delas está relacionada ao quão recente é determinada manutenção, avaliando a influência da execução desta no sistema. A segunda heurística, por sua vez, está relacionada com a frequência que as manutenções são realizadas no sistema, também avaliando a execução desta em comparativo com as manutenções anteriores. O estudo prático destas heurísticas foi realizado em sistemas de código aberto (GCC, FreeBSD, PostgreSQL e GCluster) e obteve-se 57% de melhora na execução das mudanças realizadas.

3.3 Trabalhos com enfoque em reengenharia de sistemas legados

Para esta seção do trabalho selecionou-se artigos que possuem enfoque na reengenharia de sistemas legados. O trabalho de Junior e Silva (2003) estuda a importância da usabilidade na reengenharia de um sistema legado. No trabalho de Bernhart et al. (2012) há um relato de uma experiência de reengenharia de um sistema aeroportuário com 40 anos de operação. Por fim, o trabalho de Thums e Quante (2012) relata as limitações da reengenharia em um sistema embarcado.

Uma das questões relevantes relacionadas a reengenharia de software é a usabi-

lidade: juntamente com a insatisfação dos usuários com a utilização do sistema, a usabilidade é um dos principais pontos que deve ser levado em consideração na realização da reengenharia de um sistema. Junior e Silva (2003) relatam que é muito importante o estudo da interação do usuário com o sistema, identificando os problemas e realizando o planejamento de reengenharia com base nas necessidades identificadas. Os autores indicam em seu artigo formas de utilização e aplicação das técnicas de usabilidade no planejamento da reengenharia de um sistema. Através das técnicas de teste de usabilidade, *Pluralistic Walkthrough*, avaliação heurística de usabilidade , *Cognitive Walkthrough* e modelo GOMS, os autores ponderam que a avaliação da usabilidade deve ser realizada utilizando-se ambas as técnicas, a fim de garantir a completude do processo. Além disso, a avaliação da usabilidade contribui para o sucesso da realização de reengenharia, pois avalia a insatisfação do usuário no uso do sistema e identifica possíveis falhas de usabilidade, que são problemas recorrentes em sistemas legados.

Além da usabilidade, existem também outros entraves na evolução de um sistema, como, por exemplo, sistemas de alta complexidade e de sistemas que possuem restrições relacionadas ao tempo de inatividade. Bernhart et al. (2012) descrevem em seu trabalho as experiências obtidas na reengenharia de um sistema de operações aeroportuárias que havia na época completado 40 anos de operação. O software era desenvolvido na linguagem COBOL, possuía uma interface baseada em modo texto e não havia nenhuma documentação. Em função do grande risco na substituição total do sistema de forma abrupta, devido aos riscos operacionais e financeiros que podem ser gerados por falhas no sistema e por tempo de inatividade, optou-se por uma abordagem incremental, sendo que aos poucos as funcionalidades foram migrando para o novo sistema durante os 18 meses de trabalho, e ambos os sistemas eram utilizados durante este período.

O processo foi organizado em etapas de planejamento, análise, projeto, implementação e verificação.

Em relação ao planejamento, realizou-se um recorte técnico e organizacional das funcionalidades, e definiu-se 43 incrementos, a maioria destes agregando conjuntos de características funcionais para usuários específicos. Além destes, haviam também incrementos de questões técnicas.

Em relação à análise, devido à inexistência de documentação, realizou-se a interpretação do código-fonte do sistema legado para a extração dos requisitos funcionais.

Em relação ao projeto, houve a preocupação em projetar serviços técnicos que refletissem as exigências funcionais identificadas. Além disso, houve um trabalho aprofundado na elaboração do projeto de interfaces, para que fosse reduzido o risco da falta de aceitação das interfaces pelos usuários. Criou-se interfaces objetivas e eficientes, obtidas através de um design de interação conservador e seguindo-se a estrutura e a lógica do sistema legado.

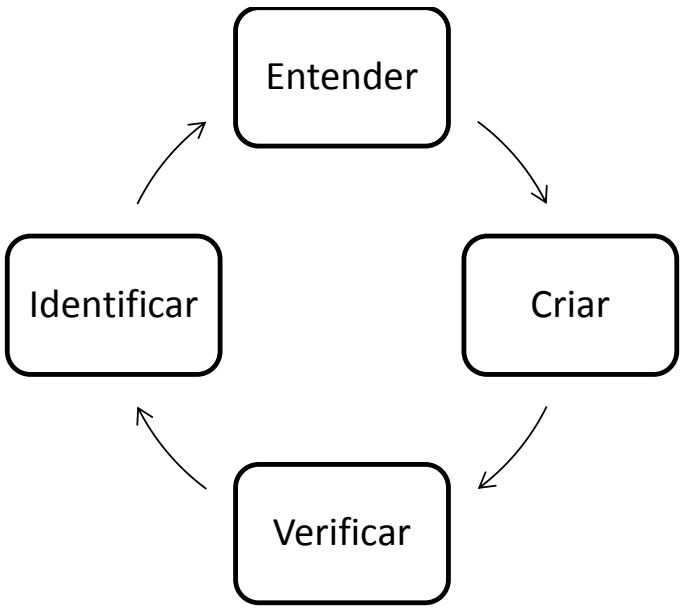
Em relação à implementação do novo sistema, houve um cuidado especial em implementar o código necessário para a comunicação e sincronização entre os 2 sistemas de forma que fosse facilmente removido após a conclusão das migrações de funcionalidades do sistema antigo.

Em relação à verificação, houve um esforço maior na realização dos testes caixa preta, através de testes estruturados e com dados reais. Também foram realizados vários testes com usuários durante o desenvolvimento das funcionalidades, antes do teste formal de aceitação.

A equipe de trabalho foi composta por 31 engenheiros que trabalharam em tempo integral. Dois destes engenheiros foram também desenvolvedores do software legado, e a participação no desenvolvimento do novo sistema foi considerada como sendo um fator crítico de sucesso da realização da reengenharia do software.

Em alguns casos existem limitações físicas em um processo de reengenharia. Thums e Quante (2012) relatam em seu trabalho um projeto de pesquisa realizado pela empresa alemã *Bosch*, que envolvia a reengenharia de um software automotivo embarcado. Tal software possui um grande valor para a empresa, porém, devido as várias modificações, não possuía mais uma estrutura adequada. O processo de reengenharia, baseado no modelo ferradura, envolveu quatro fases: *identify* (identificar), *understand* (entender), *create* (criar) e *verify* (verificar). A Figura 11 ilustra a proposta.

Figura 11 – Modelo de reengenharia utilizado por Thums e Quante (2012)



Fonte: Adaptado de Thums e Quante (2012, p. 495)

Inicialmente são identificados os artefatos em que a realização da reengenharia

trará maior benefício, e que geralmente são os mais complexos. A partir da seleção, é necessário compreender e entender o motivo da complexidade presente. São identificados os padrões “sujos” e os padrões de projeto que geram uma melhor solução. A funcionalidade deve ser totalmente interpretada antes da construção de uma nova implementação com o mesmo comportamento e com complexidade reduzida. Essa atividade geralmente consome 40% dos custos da reengenharia de componentes.

A construção das novas implementações deve respeitar os princípios de design e padrões de projeto, gerando codificação compreensível. A última etapa compreende a verificação da solução implementada, avaliando se a mesma cumpre com os requisitos e se é equivalente a solução anterior.

Neste processo de reengenharia o foco principal foi prevenir o envelhecimento do software e apoiar a compreensão do programa. Algumas características específicas necessárias em um software desse tipo foram observadas na evolução: necessidade do menor tempo de resposta possível, o software é programado em um subconjunto restrito da linguagem C, a comunicação é realizada através de variáveis globais e as funções são chamadas ciclicamente em intervalos definidos, o código possui estruturas que maximizam a velocidade de execução que desestruturam o código, a memória é alocada de forma estática para prevenir erros de execução e a compreensão das funções do programa é difícil porque envolve vários conceitos físicos. Os autores apontam como principal contribuição da reengenharia um melhor entendimento das características do sistema e das tecnologias utilizadas no desenvolvimento do sistema e também um melhor entendimento de como devem ser utilizadas de forma eficaz e eficiente.

3.4 Trabalhos aplicados ao ensino de reengenharia de software

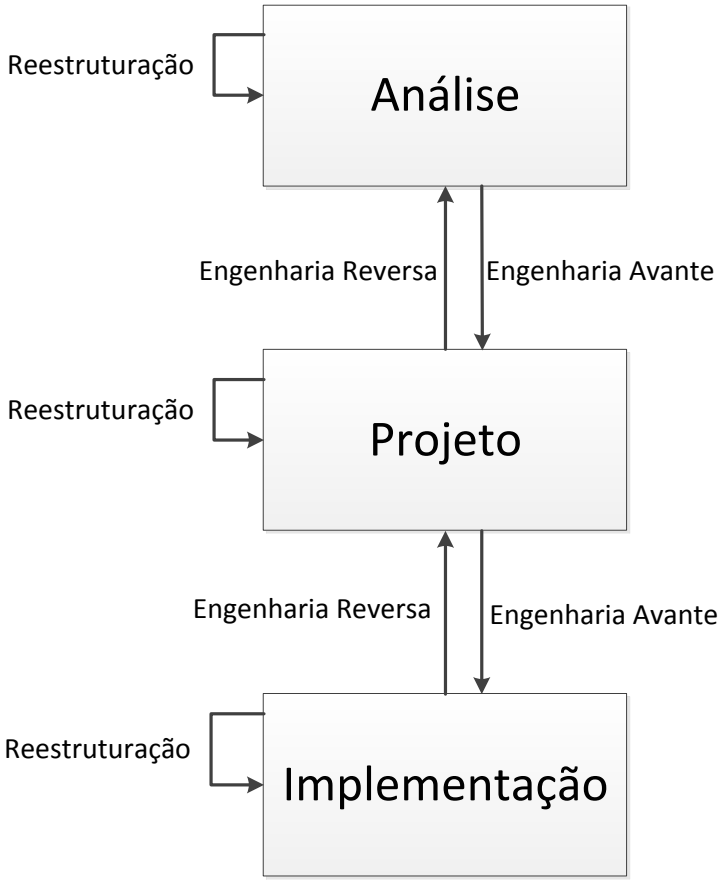
Nesta seção são selecionados dois artigos que descrevem a experiência do ensino de reengenharia de software em universidades distintas. O primeiro trabalho, de Perez-Castillo et al. (2013), relata a experiência de reengenharia com a utilização de ferramentas em cursos de computação de uma universidade espanhola. O segundo trabalho, descrito por Petrenko et al. (2007), descreve a experiência de utilização de um processo de reengenharia de software, onde a realização das atividades práticas de reengenharia é realizada com o uso de sistemas de código aberto (*open source*).

O ensino das práticas de Evolução de software no nível universitário geralmente se restringe apenas ao conteúdo teórico. Perez-Castillo et al. (2013) citam que a manutenção de software é muitas vezes encarada como sendo uma atividade adicional no desenvolvimento de um sistema (e não como parte do processo de desenvolvimento) e também que há pouca pesquisa na área de manutenção em comparativo com outras fases de desen-

volvimento de software.

Os autores realizaram um estudo prático do uso de ferramentas de apoio a reengenharia de software, conciliando momentos teóricos e exercícios práticos com aplicativos de apoio a geração de código e engenharia reversa, no contexto de um curso de Engenharia de Software em uma universidade espanhola. Esta estratégia foi utilizada por dois anos e os participantes foram avaliados antes e após a experiência.

Figura 12 – Modelo de reengenharia utilizado por Perez-Castillo et al. (2013)



Fonte: Adaptado de Perez-Castillo et al. (2013, p. 1284)

A Figura 12 ilustra o processo utilizado durante as aulas práticas, que é baseado no modelo ferradura. O mesmo é composto de três partes:(I) a etapa de análise, que tem por objetivo analisar o software e identificar componentes e suas relações, produzindo uma representação do sistema, (II) a etapa de projeto, onde transforma-se a representação anterior em uma representação melhorada do sistema, preservando os comportamentos do sistema e (III) a etapa de implementação, onde gera-se a implementação física do programa reestruturado.

Em relação à avaliação do uso do processo, os autores relatam a aplicação de 03

questionários distintos:

1. **Pré-questionário**: este questionário, composto por questões teóricas e práticas, foi aplicado antes da exposição teórica sobre o assunto, e teve como objetivo realizar uma avaliação diagnóstica do nível de conhecimento dos alunos em relação à reengenharia;

2. **Pós-questionário**: o pós-questionário foi respondido pelos alunos após a realização das atividades práticas e teve por objetivo avaliar os conhecimentos adquiridos pelos alunos após a realização da experiência;

3. **Exame final**: o exame final teve por objetivo coletar a opinião dos alunos em relação à aplicação da experiência, indagando-os sobre incluir o exercício prático de reengenharia como conteúdo regular futuramente, e também em relação aos aspectos positivos e negativos da experiência.

Como resultados obtidos através da aplicação dos questionários, os autores descrevem que a experiência foi eficaz em ambos os anos em que foi realizada. Além disso, os autores atingiram um alto grau de satisfação da aplicação da experiência. Em relação aos pontos positivos, os estudantes destacaram que aprenderam técnicas de reutilização de código e avaliaram positivamente o uso de novas ferramentas de reengenharia. A necessidade de um maior tempo para a realização das atividades foi apontada como ponto negativo.

Os autores citam como lições aprendidas desta experiência: o desafio para os estudantes em introduzir novas funcionalidades no sistema; confusão das fases de reengenharia; reestruturação a nível de código, ao invés de ser a nível de UML; tempo inábil para um exercício prático de melhor resultado; Aprovação por parte dos estudantes do uso das ferramentas de apoio; e a prática de reengenharia pode ser aplicada em turmas iniciantes e também em turmas próximas ao final do curso.

O segundo trabalho com enfoque em ensino de evolução de software é proposto por Petrenko et al. (2007). O trabalho envolve a realização de atividades práticas de evolução de software utilizando-se aplicativos de código aberto, em uma universidade norte-americana. A experiência desenvolvida foi executada duas vezes, com turmas distintas, e os alunos preencheram um questionário após a realização das atividades. Para a realização das atividades, o trabalho baseou-se em um processo de fácil entendimento, composto pelas seguintes etapas: iniciação, localização de contexto, análise de impacto, pré-fatoração, atualização, pós-fatoração, atualização, propagação de mudanças, pós-fatoração, teste e nova linha de base. A Figura 13 ilustra o processo utilizado para a realização da evolução.

Figura 13 – Modelo de reengenharia utilizado por Petrenko et al. (2007)

Fonte: Adaptado de Petrenko et al. (2007, p. 26)

A atividade de iniciação envolve a solicitação, por parte do cliente, de correções de defeitos em alguma funcionalidade ou o desenvolvimento de uma nova funcionalidade.

A atividade de localização de contexto envolve o levantamento da localização inicial da mudança dentro do código-fonte do sistema. Esta atividade é importante porque ao se localizar o local exato evita-se a compreensão de todo o código e a inserção de defeitos em outras funcionalidades do sistema.

A atividade de análise de impacto determina a extensão de uma mudança no código e delimita o conjunto de impacto no sistema.

A atividade de pré-fatoração refatora o código existente, com o objetivo de facilitar o entendimento do mesmo antes de se inciar as modificações solicitadas.

A atividade atualização envolve a implementação do novo código e a integração do mesmo com o restante do sistema nos locais definidos pela atividade de localização de contexto.

A atividade de propagação de mudanças tem por objetivo apontar se há inconsistências no sistema após as modificações, identificando classes dependentes e de apoio.

A atividade de pós-fatoração tem por objetivo a remoção de código desnecessário no sistema, através de refatoração, com a utilização de ferramentas de apoio.

A atividade de teste ocorre durante as etapas de pré-fatoração, atualização, propagação de mudanças e pós-fatoração, realizando testes unitários e testes funcionais, verificando todas as classes afetadas pela mudança e se as novas funcionalidades estão funcionando corretamente. Os testes também garantem que as refatorações realizadas não inserem defeitos no sistema.

A última atividade do processo é a de criação de uma nova linha de base, onde os programadores criam uma linha de base em um servidor cvs para futuras atualizações.

A dinâmica de utilização do processo ocorreu da seguinte forma: os alunos se organizaram em grupos com 4 a 6 componentes e tiveram o suporte de um gerente de projeto / cliente, papel que foi realizado por um estudante de pós-graduação. Os grupos realizavam reuniões semanais com seu gerente de projeto e também realizavam comunicação com o mesmo através de *e-mail*. A universidade disponibilizou um servidor CVS para a utilização por parte dos alunos.

A execução do processo aconteceu em três fases/iterações. A primeira fase objetivou a experimentação do uso do CVS e a familiarização com o processo. A segunda e terceira fase envolveram a introdução de mudanças complexas e exigiu a comunicação dos estudantes com seus respectivos gerentes. Ao final de cada ciclo os estudantes apresentavam os resultados obtidos para seus gerentes e também elaboraram um relatório das atividades realizadas.

Os autores destacam no trabalho que foram necessárias 20 horas para a seleção dos sistemas que seriam utilizados, para a definição das mudanças que seriam solicitadas e também para a preparação do servidor CVS.

Cada fase de iteração do processo envolveu entre 7 e 12 horas de trabalho, sendo que foi necessário o treinamento para o uso do CVS em função da inexperiência dos estudantes no uso dessa ferramenta e da forma que o código a ser evoluído estava organizado na ferramenta.

Os autores citam no trabalho como vantagem o uso do CVS, pois com o uso do servidor há a identificação da produção e contribuição individual de cada participante no desenvolvimento das mudanças solicitadas no sistema.

Em relação à avaliação da utilização da proposta, foi elaborado um questionário sob duas dimensões distintas: a satisfação do aluno em relação à proposta e a aprendizagem obtida através da proposta.

A partir dos dados obtidos nas duas experiências realizadas, identificou-se que os estudantes conseguiram cumprir com o proposto e esperado, independentemente do tamanho do sistema disponível para a realização da experiência. A avaliação qualitativa respondida pelos estudantes demonstrou que a abordagem utilizada foi positiva e que é possível o uso de sistemas de código aberto (*open source*) para o ensino de evolução de software. Adicionalmente, o uso do servidor CVS permitiu uma avaliação eficaz da equipe e também do desempenho individual de cada integrante.

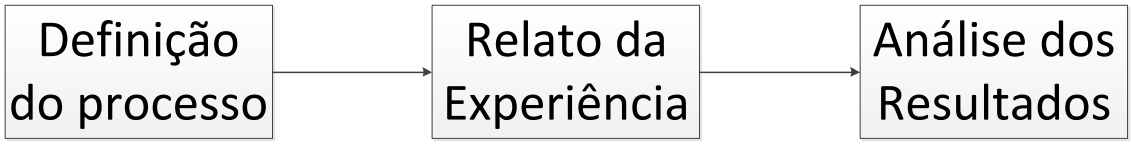
4 Metodologia

O presente capítulo apresenta a metodologia da realização deste trabalho, que envolve a definição do processo, baseada nos processos de reengenharia existentes, a aplicação do processo na disciplina de evolução de software e a análise dos resultados obtidos a partir da execução do processo na disciplina.

A partir do estudo dos processos existentes na literatura, e dos trabalhos que apresentam estratégias da realização de reengenharia de sistemas legados, constatou-se que nos mesmos há apenas uma descrição das etapas envolvidas na execução do processo, não descrevendo claramente quais as atividades envolvidas em cada uma destas etapas. Neste contexto, buscou-se na criação do processo envolver além das etapas, as atividades necessárias para a realização da reengenharia do sistema, o que é a grande contribuição deste trabalho e o grande diferencial desta proposta de processo de reengenharia em relação aos processos existentes.

A Figura 14 apresenta um esquema para a representação da metodologia de realização deste trabalho. A primeira etapa é a definição do processo, a partir do estudo dos processos existentes, que é descrita na seção 4.1. A etapa seguinte é a aplicação do processo na disciplina e a aplicação da avaliação através de um questionário, cujos relatos estão descritos na seção 4.2. Concluindo, a partir das informações obtidas através do acompanhamento da realização das atividades e da avaliação da execução do processo é realizada a análise dos resultados obtidos, descrita no Capítulo 5.

Figura 14 – Metodologia



4.1 Definição do processo

4.1.1 Visão geral

A definição do processo de reengenharia envolve a criação de um conjunto de atividades, sendo que o processo inicialmente possui uma abordagem baseada no modelo em cascata, que é responsável pela avaliação do software legado sob perspectiva de negócio e técnica e definir a estratégia de manutenção do sistema legado. A partir daí optou-se

por uma abordagem incremental, envolvendo a análise técnica do sistema legado, a identificação do contexto do novo sistema, a elaboração do projeto do sistema e a elaboração e priorização da lista de incrementos que serão alocados aos ciclos de desenvolvimento seguintes.

A utilização desta abordagem incremental justifica-se pela rápida entrega de software funcional ao cliente, o que permite que as funcionalidades sejam aos poucos migradas para o novo sistema, além da realização de *feedback* por parte dos usuários, onde as solicitações de melhorias ocorrem ainda na etapa de desenvolvimento.

Adicionalmente, o uso de abordagens incrementais para a realização de reengenharia já foi descrito em Bernhart et al. (2012) e em Petrenko et al. (2007), evidenciando a viabilidade de se utilizar esta forma de abordagem, especialmente no contexto educacional, onde muitas vezes em função do reduzido tempo disponível para atividades práticas, a utilização de abordagens incrementais permite que sejam exercitadas todas as atividades de um processo e ao final da execução deste sejam gerados artefatos funcionais e consistentes.

Optou-se pela modelagem do sistema evoluído a partir da abordagem de orientação a objetos, que uniformiza os modelos utilizados para a análise, projeto e implementação de um sistema. O uso da orientação a objetos facilita a comunicação da equipe e permitem uma maior participação do usuário no processo. Adicionalmente, com o uso de orientação à objetos obtém-se um aumento de produtividade, redução de custos com desenvolvimento, manutenção e reutilização de código (MELO, 2010).

O processo foi modelado através da *Business Process Modeling Notation* (BPMN), que é uma notação de modelagem de processos de negócio. A Figura 15 define o modelo de processo de reengenharia proposto por este trabalho, sendo que nesta figura os subprocessos encontram-se minimizados para que seja possível uma visão geral do processo. No Apêndice A é possível visualizar a versão da modelagem do processo com os subprocessos expandidos.

Figura 15 – Processo de reengenharia.

4.1.2 Detalhamento do processo

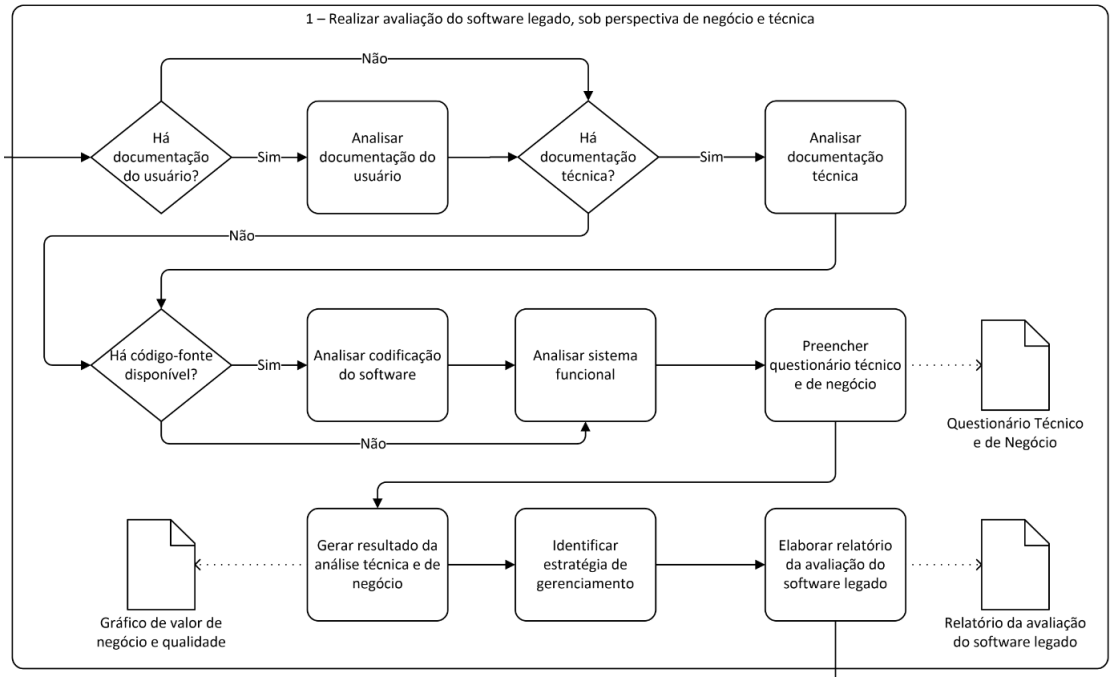
Nesta seção do trabalho cada subprocesso é descrito de forma detalhada. Para cada subprocesso ou atividade isolada são apresentadas as entradas e saídas, facilitando o entendimento do funcionamento do processo. Também estão presentes os artefatos elaborados para o suporte da execução das atividades. Adicionalmente, as técnicas de Evolução de Software são referenciadas em cada atividade as quais estão relacionadas.

Etapa 1 - Avaliação do software legado

A primeira etapa do processo envolve a avaliação do software legado e a definição da estratégia de manutenção para o sistema, avaliando a viabilidade da realização da reengenharia do software legado. Pfleeger (2004) e Sommerville (2011) defendem a necessidade de se realizar um estudo do sistema legado, com o objetivo de avaliar qual a melhor estratégia de manutenção para o mesmo.

4.1.2.1 Subprocesso 1 - Realizar avaliação do software legado

Figura 16 – Subprocesso expandido 1.



O subprocesso “Realizar avaliação do software legado, sob perspectiva técnica e de negócio” é o primeiro grupo de atividades a serem desenvolvidas no processo. Este subprocesso é estratégico porque tem por objetivo avaliar o software legado e definir a estratégia para a manutenção do mesmo. A avaliação do sistema legado é dirigida sob duas perspectivas: a perspectiva de negócio e a perspectiva técnica. Em relação à perspectiva

de negócio, avalia-se a real necessidade do sistema para a organização, e sob a perspectiva técnica examina-se a qualidade do software e sua compatibilidade com as tecnologias mais recentes (SOMMERVILLE, 2011).

De acordo com Sommerville (2011), a partir do cruzamento das perspectivas de negócio e técnica, obtém-se quatro grupos distintos de sistemas: (1) Baixa qualidade e baixo valor de negócio, (2) Baixa qualidade e alto valor de negócio, (3) Alta qualidade e baixo valor de negócio e (4) Alta qualidade e alto valor de negócio. Cada um desses grupos possui correspondência com uma das quatro opções estratégicas para a manutenção do software que estão descritas na seção 2.2: (1) Descartar completamente o sistema, (2) Deixar o sistema inalterado e seguir a manutenção regular, (3) Reestruturar o sistema ou (4) Substituir o sistema por um novo sistema. Abaixo encontra-se a descrição das características do sistema a partir das possíveis combinações de qualidade e valor de negócio, juntamente com a opção estratégica recomendada para cada caso:

1. *Baixa qualidade e baixo valor de negócio*. Devido a essa característica, o uso do software pelo cliente possui pouco retorno, e sua manutenção é cara, o que faz com que sistemas que se enquadram nessa categoria sejam descartados.

2. *Baixa qualidade e alto valor de negócio*. Sistemas que se enquadram nessa categoria são importantes para a organização e não devem ser descartados. A baixa qualidade indica que os custos de manutenção são altos e que o sistema deve ser reestruturado para ampliar a sua qualidade, ou então ser substituído por uma solução de prateleira, caso a mesma esteja disponível.

3. *Alta qualidade e baixo valor de negócio*. Sistemas com essa característica não contribuem de forma efetiva para a organização, porém podem ser submetidos a manutenções regulares se as mudanças necessárias não forem de alto custo, caso contrário o mesmo deve ser descartado.

4. *Alta qualidade e alto valor de negocio*. Sistemas com essas características devem permanecer em operação e a manutenção regular deve ser mantida.

É importante salientar que a execução do processo somente prosseguirá caso a opção escolhida seja a reengenharia como estratégia de manutenção do sistema legado, caso contrário, o processo é encerrado após a conclusão das atividades presentes neste subprocesso.

Entradas

O subprocesso “Realizar avaliação do software legado, sob perspectiva técnica e de negócio” possui as seguintes entradas:

- Documentação técnica do sistema legado (caso ela exista);
- Documentação de usuário do sistema legado (caso ela exista);
- Software legado (executável ou códigos-fonte, caso estejam disponíveis);

Atividades

A partir da análise das informações presentes na documentação técnica e de usuário e do sistema legado, avalia-se o software, utilizando-se um conjunto de questões previamente definidas em função da perspectiva técnica e da perspectiva de negócio. Tais questões foram elaboradas a partir da proposta de gerenciamento de sistemas legados de Sommerville (2011) e estão organizadas em dois conjuntos, um de questões relativas ao valor de negócio e outro em função da qualidade do sistema. Cada conjunto de questões é composto pelo número sequencial da questão, a descrição da questão, o avaliador da questão (responsável ou o grupo de responsáveis por respondê-la) e pela resposta (sim ou não), de forma que a resposta seja quantificável. As tabelas abaixo representam as questões, sob a perspectiva de valor de negócio (Tabela 1) e sob a perspectiva de qualidade (Tabela 2).

Tabela 1 – Questões para avaliação sob perspectiva de valor de negócio

Nº	Questão	Avaliador	Resposta
1	Considerando o contexto onde o software está inserido, o sistema é usado com frequência?	Usuário	
2	Os processos de negócio do sistema estão atualizados?	Gerente	
3	O sistema é confiável (possui pouquíssimos defeitos ou falhas)?	Usuário, gerente	
4	As saídas do sistema são utilizadas? Há importância para o usuário?	Usuário, gerente	

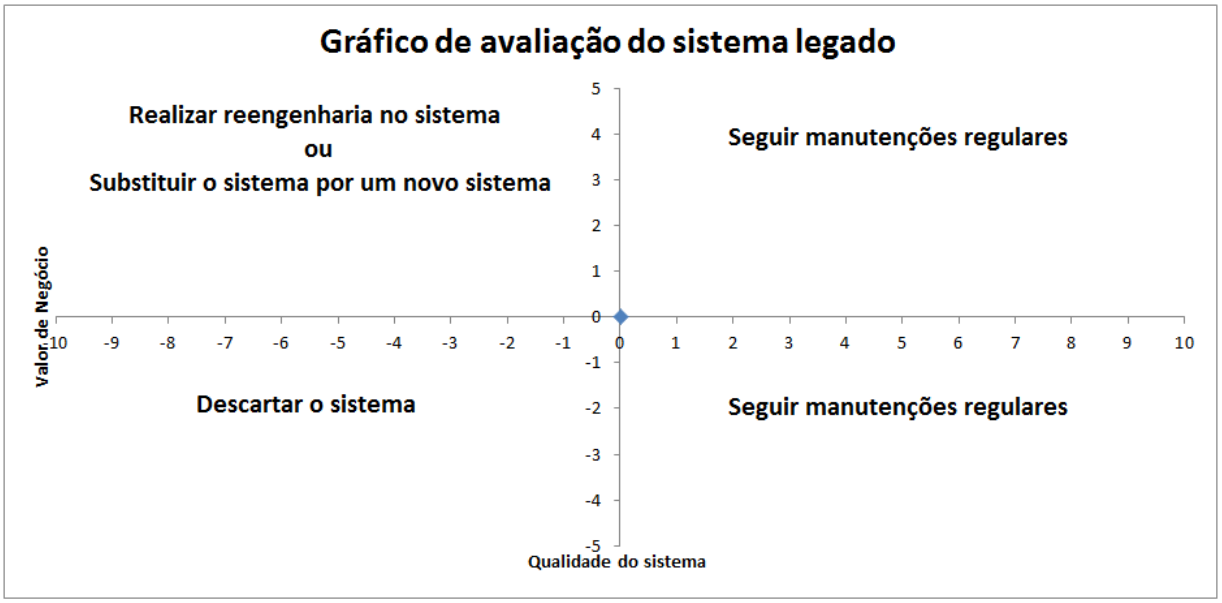
As questões são respondidas com o apoio de uma planilha eletrônica previamente configurada. De acordo com as respostas de cada uma das questões, a planilha representa em um gráfico de dispersão um ponto, sendo que o local desse ponto é definido em função do eixo da abscissa (x), que representa a qualidade do sistema e do eixo da ordenada (y), que representa o valor de negócio do sistema.

Para cada possível localização do ponto, que representa o sistema em um dos quadrantes do gráfico, há uma estratégia recomendada para a manutenção desse sistema. Tal gráfico foi construído a partir de uma adaptação do gráfico de avaliação de sistemas legados proposto por Sommerville (2011), que classifica o sistema em função da perspectiva de valor de negócio e qualidade. A Figura 17 ilustra o gráfico gerado pela planilha.

Tabela 2 – Questões para avaliação sob perspectiva de qualidade

Nº	Questão	Avaliador	Resposta
1	O fornecedor do sistema ainda existe e dá suporte para o mesmo?	Gerente	
2	O sistema é compatível com o hardware e sistema operacional mais recentes?	Área técnica	
3	O sistema foi submetido a várias modificações durante o seu uso?	Área técnica, gerente	
4	O desempenho do sistema é adequado?	Usuário, gerente	
5	O software opera independentemente de manutenção de hardware legado?	Área técnica	
6	O software opera independentemente de manutenção de sistemas de apoio pagos?	Área técnica	
7	O sistema possui custo baixo de manutenção?	Área técnica	
8	O sistema é compatível com diversos sistemas operacionais (windows, linux, etc.)?	Área técnica	
9	O código-fonte do sistema legado é de fácil compreensão?	Área técnica	
10	Há documentação do software legado disponível?	Área técnica	
11	Os dados são armazenados em um banco de dados?	Área técnica	
12	O banco de dados está normalizado?	Área técnica	
13	É adotado um padrão para nomenclatura das tabelas e campos do Banco de Dados?	Área técnica	
14	São realizadas validações nos campos (Ex.: Datas, CPF, e-mail, etc.)?	Área técnica, usuário	
15	O sistema exige que campos obrigatórios sejam preenchidos?	Área técnica, usuário	
16	A linguagem de programação do sistema legado ainda é utilizada no mercado de trabalho?	Área técinca	
17	Existem registros de execução de testes do sistema?	Área técnica	
18	A interface do sistema é implementada em modo gráfico?	Área técnica	
19	Existem profissionais com habilidades necessárias para o suporte da aplicação?	Área técnica	
20	O sistema redimensiona as informações ao maximizar a tela?	Área técnica	
21	Ao solicitar o armazenamento de dados de um formulário, o usuário é sempre informado se os mesmos foram gravados com sucesso ou se ocorreu uma falha?	Área técnica, usuário	
22	O sistema possui um padrão de interfaces? Esse padrão é sempre seguido?	Área técnica	
23	O sistema permite a navegação através do teclado?	Área técnica	

Figura 17 – Gráfico de avaliação do sistema legado.



Adicionalmente, a partir da análise da documentação técnica e de usuário do sistema e da codificação, deve-se identificar as seguintes informações adicionais, que serão utilizadas na etapa seguinte do processo:

- Identificar a linguagem de programação em que o software foi desenvolvido;
- Identificar o propósito do sistema (a que se destina?);
- Identificar o paradigma do software (estruturado ou orientado a objetos?);

Todas essas informações levantadas neste subprocesso deverão ser organizadas em um relatório, onde deverão estar presentes as questões do questionário técnico e de negócio respondidas, o gráfico indicando a estratégia compatível com as respostas dos questionários, as informações adicionais e a decisão de estratégia de manutenção do software legado escolhida pela equipe ((1) Descartar completamente o sistema, (2) Deixar o sistema inalterado e seguir a manutenção regular, (3) Reestruturar o sistema ou (4) Substituir o sistema por um novo sistema).

Saídas

O subprocesso “Realizar avaliação do software legado, sob perspectiva técnica e de negócio” produz como saída o relatório da avaliação do software legado.

Etapa 2 - Reengenharia

É nesta etapa que ocorrem as atividades que compreendem a reengenharia do sistema legado. É composta por 3 grupos de macro atividades: o de análise do sistema legado, o de análise e projeto do novo sistema e o de implementação e implantação do novo sistema. Na Figura 18 é possível visualizar cada um destes grupos com seus respectivos subprocessos. Os grupos de macro atividades agrupam os subprocessos e atividades de acordo com a proposta de processo de reengenharia do modelo ferradura (SEACORD; PLAKOSH; LEWIS, 2003), que é organizado em três etapas: reconstrução, transformação e refinamento.

O uso da proposta do modelo ferradura para o ensino de reengenharia é descrito em Perez-Castillo et al. (2013), que utilizou uma abordagem não incremental para o ensino de reengenharia de software em uma universidade espanhola. No trabalho de Thums e Quante (2012) também realizou-se a reengenharia de um sistema legado, porém neste caso houve uma adaptação do modelo ferradura e a experiência foi realizada no contexto de uma empresa, e não no contexto educacional.

Em relação a macro atividade de análise do software legado, a mesma é composta pelo subprocesso 02 e baseia-se no conceito da engenharia reversa, cujo objetivo é extrair informações relacionadas a estrutura do sistema legado, que facilitam a documentação do sistema (SOMMERVILLE, 2007). Dentro desta macro atividade há o subprocesso de análise técnica do software legado, que tem por objetivo o levantamento de informações relacionadas à estrutura do sistema. Essa macro atividade é relacionada à etapa de reconstrução do modelo ferradura.

A macro atividade de análise e projeto do novo software envolve os subprocessos 03, 04 e 05 e tem por objetivo o planejamento e a identificação do contexto do novo sistema, a elaboração do projeto do novo sistema e a elaboração da lista de incrementos, através da tabulação e priorização dos casos de uso do novo sistema. Pfleeger (2004) define em seu processo de reengenharia que é necessário que a especificação e o projeto do sistema sejam atualizados. Essa macro atividade é relacionada à etapa de transformação do modelo ferradura.

A terceira macro atividade é composta pelas atividades 06, 10 e 12 e pelos subprocessos 07, 08, 09, 11 e 13 e envolve o desenvolvimento das funcionalidades do novo sistema a partir da construção utilizando-se uma abordagem incremental. Essa macro atividade é relacionada à etapa de refinamento do modelo ferradura.

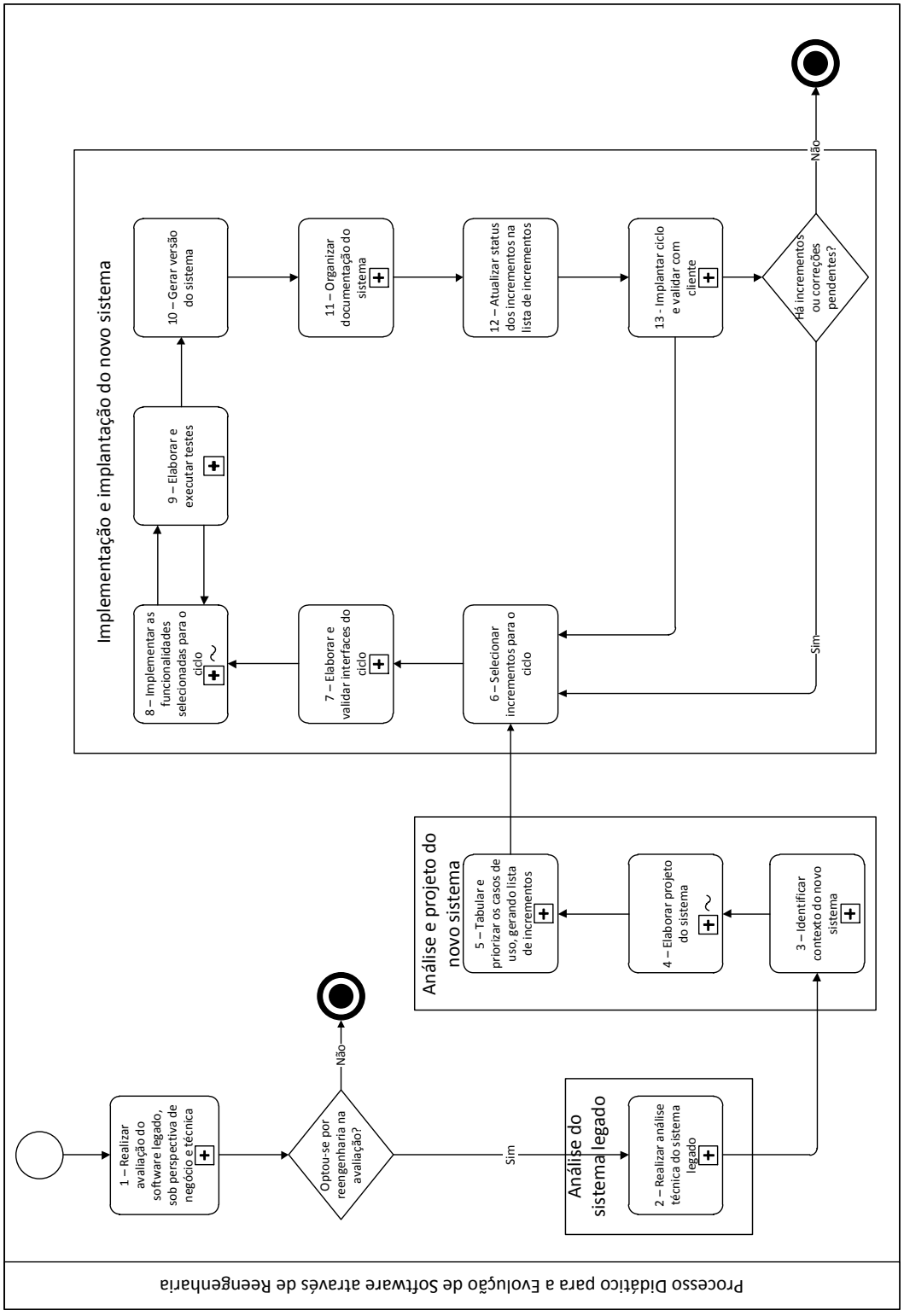
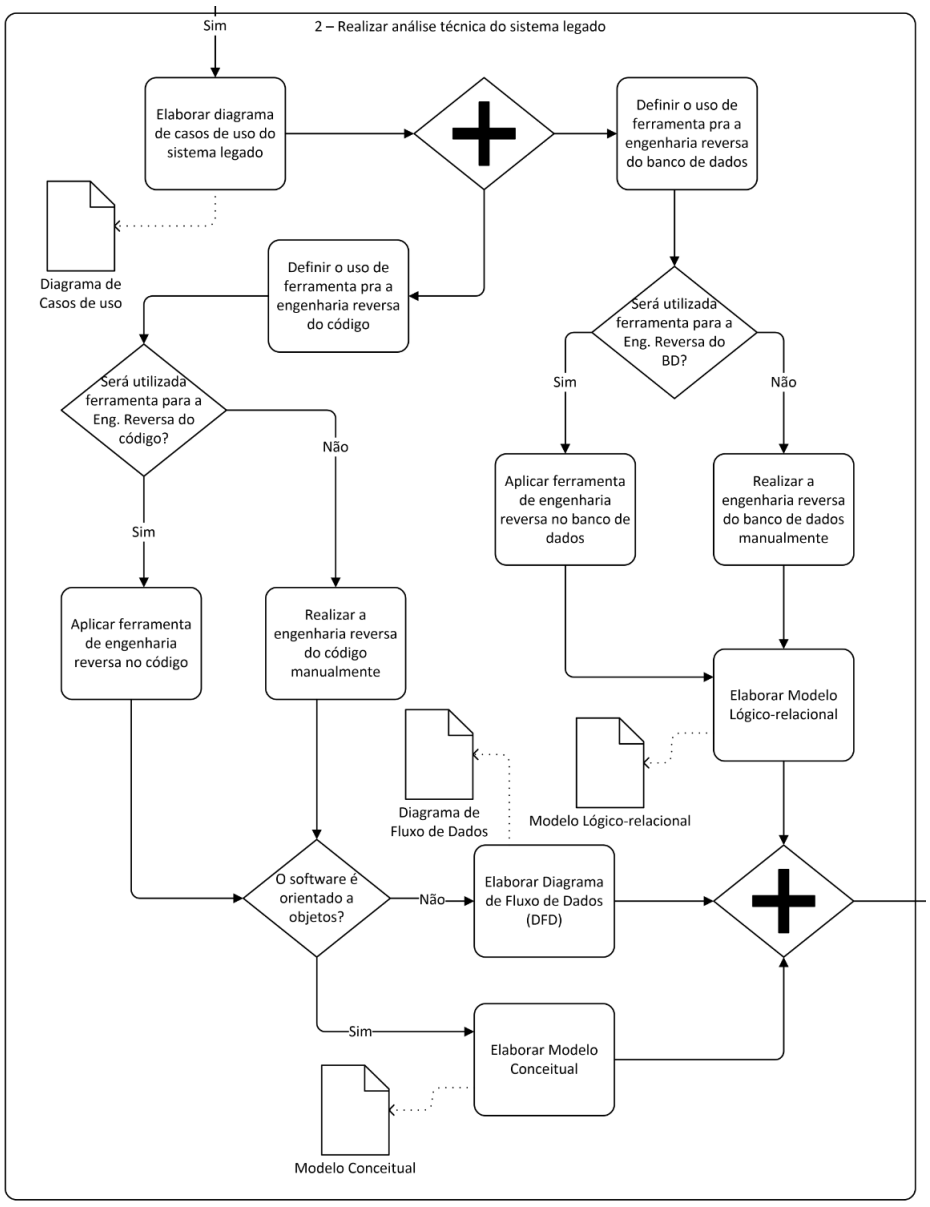


Figura 18 – Indicação das macro atividades no processo

4.1.2.2 Subprocesso 2 - Realizar análise técnica do software legado

Figura 19 – Subprocesso expandido 2.



A análise técnica tem por objetivo levantar informações relacionadas a estrutura do sistema legado, de forma mais aprofundada, envolvendo a elaboração do diagrama de casos de uso do sistema e a obtenção de diagramas através de engenharia reversa. A engenharia reversa é realizada a partir do uso de ferramentas ou de forma manual, caso não existam ferramentas para a linguagem de programação do sistema legado. As informações são organizadas de forma que seja possível a sua manipulação, e a partir de sua representação pode-se identificar inconsistências e violações de padrões de projeto (PFLEEGER, 2004).

Entradas

O subprocesso “Realizar análise técnica do software legado” possui as seguintes entradas:

- Documentação do sistema legado (caso ela exista);
- Software legado (executável ou códigos-fonte, caso estejam disponíveis);
- Informações relativas ao contexto do software legado, presentes no relatório da avaliação do software legado(saída do subprocesso 1);

Atividades

No subprocesso “Realizar análise técnica do software legado” são realizadas as seguintes atividades:

- Elaborar o diagrama de casos de uso do sistema legado;
- Definir se a atividade de engenharia reversa do código e do banco de dados será apoiada por alguma ferramenta ou se será realizada de forma manual;
- Realizar a engenharia reversa do código e do banco de dados;
- A partir das informações obtidas com a atividade de engenharia reversa, elaborar o modelo conceitual do software legado[1] ou diagrama de fluxo de dados do software legado[2];
- A partir das informações obtidas com a engenharia reversa do banco de dados, elaborar o modelo lógico relacional do software legado (caso o software armazene as informações em banco de dados);

Saídas

O subprocesso “Realizar análise técnica do software legado” gera os seguintes artefatos:

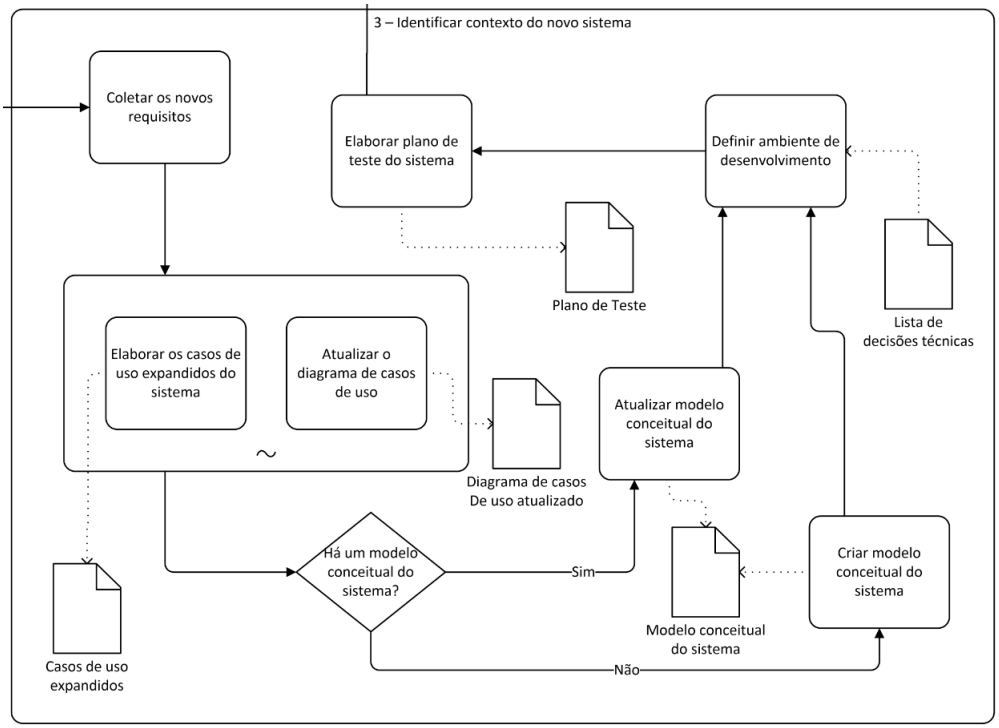
[1] O modelo conceitual representa as informações que o sistema gerencia e que o usuário identifica, representada em notação UML (WAZLAWICK, 2011). Caso seja identificado anteriormente que o sistema legado foi modelado utilizando-se a abordagem orientada a objetos, deve-se representar os atributos, conceitos e associações do sistema através do modelo conceitual.

[2] Através do diagrama de fluxo de dados do sistema, obtém-se uma visão dos fluxos e requisitos de um sistema, através da visão entrada-processo-saída, onde os objetos de dados entram no sistema, sofrem transformação através de processamento e saem do software (PRESSMAN, 2011). O uso do diagrama de fluxo de dados é mais frequente em sistemas que possuem paradigma estruturado, como forma de representação das transições e processamento dos dados do sistema.

- Diagrama de casos de uso do sistema, identificando os atores do sistema e como se relacionam com as funcionalidades;
- Identificação da forma com que os dados são armazenados no sistema (arquivos, banco de dados, etc.), juntamente com o modelo lógico relacional do sistema legado (caso o software armazene as informações em banco de dados);
- Modelo conceitual ou Diagrama de Fluxo de Dados do Software legado.

4.1.2.3 Subprocesso 3 - Identificar contexto do novo sistema

Figura 20 – Subprocesso expandido 3.



O subprocesso “Identificar contexto do novo sistema” envolve a identificação do contexto do novo sistema, elaborado a partir das informações obtidas através da análise do sistema legado e das mudanças solicitadas pelo cliente (caso elas existam).

Entradas

O subprocesso “Identificar contexto do novo sistema” possui as seguintes entradas:

- Informações presentes na análise do software legado, produzidas no subprocesso 2;
- Solicitações de mudança;

Atividades

No subprocesso “Identificar contexto do novo sistema” são realizadas as seguintes atividades:

- Coletar os novos requisitos (caso eles existam);
- Atualizar o diagrama de casos de uso do sistema (inclusão de novos casos de uso e/ou alterações nos casos de uso já existentes);
- Elaborar os casos de uso expandidos do sistema;
- Atualização do modelo conceitual (caso o sistema legado seja de paradigma orientado a objetos) ou elaboração do modelo conceitual do sistema (caso o sistema legado seja de paradigma estruturado);
- Definir o ambiente de desenvolvimento (escolha de linguagem de programação, padrões de projeto, ferramentas a serem utilizadas, banco de dados, forma de migração das informações do sistema antigo para o novo sistema, IDE de desenvolvimento, etc.) e registrá-lo em uma lista das decisões técnicas tomadas pela equipe;
- Definir o plano de teste (definição estratégia de teste do novo sistema, seleção das funcionalidades a serem testadas e funcionalidades que não serão testadas, escolha de ferramentas para o teste, definição de artefatos gerados pelos testes);

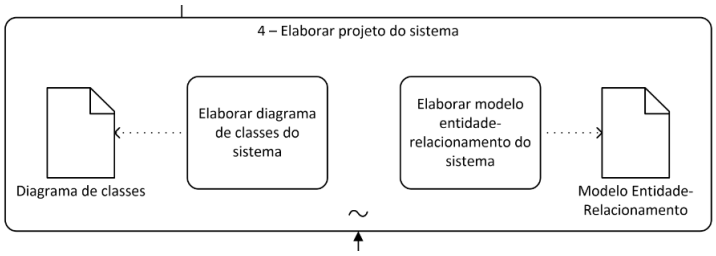
Saídas

O subprocesso “Identificar contexto do novo sistema” possui as seguintes saídas:

- Diagrama de casos de uso do novo sistema;
- Casos de uso expandidos do novo sistema;
- Modelo conceitual do novo sistema;
- Lista de decisões técnicas;
- Plano de teste;

4.1.2.4 Subprocesso 4 - Elaborar projeto do sistema

Figura 21 – Subprocesso expandido 4.



A partir das informações obtidas no subprocesso de identificação do contexto do novo sistema, é necessária a elaboração do projeto do novo sistema, que será modelado inicialmente de forma ampla e será modificado e complementado de acordo com a necessidade durante os ciclos de desenvolvimento.

Entradas

O subprocesso “Elaborar projeto do sistema” possui as seguintes entradas:

- Diagrama de casos de uso do novo sistema (saída do subprocesso 3);
- Casos de uso expandidos do novo sistema (saída do subprocesso 3);
- Modelo conceitual do novo sistema (saída do subprocesso 3);
- Lista de decisões técnicas (saída do subprocesso 3);
- Plano de teste (saída do subprocesso 3);
- Modelo lógico relacional do sistema legado (caso o sistema legado utilize tecnologia de banco de dados para o armazenamento das informações - saída do subprocesso 2);

Atividades

O projeto do sistema envolve a elaboração do diagrama de classes, que representa a estrutura do sistema, e do modelo entidade-relacionamento, sendo que este deve ser gerado a partir da reestruturação dos dados do sistema legado (SOMMERVILLE, 2007) (PRESSMAN, 2006), de forma a refletir as mudanças do software e melhorar a qualidade da estrutura de dados, corrigindo erros, redundâncias, etc.

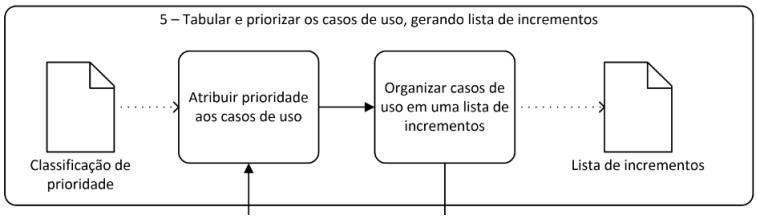
Saídas

O subprocesso “Elaborar projeto do sistema” gera as seguintes saídas:

- Diagrama de classes do sistema;
- Modelo entidade-relacionamento do sistema;

4.1.2.5 Subprocesso 5 - Tabular e priorizar os casos de uso

Figura 22 – Subprocesso expandido 5.



Este subprocesso tem por objetivo definir uma lista de incrementos a serem realizados, a partir da descrição dos casos de uso do novo sistema.

Entradas

O subprocesso “Tabular e priorizar os casos de uso, gerando lista de incrementos” possui as seguintes entradas:

- Diagrama de casos de uso do novo sistema (saída do subprocesso 3);
- Casos de uso expandidos do novo sistema (saída do subprocesso 3);
- Diagrama de classes do sistema (saída do subprocesso 4);
- Modelo entidade-relacionamento do sistema (saída do subprocesso 4);

Atividades

A partir das informações das expansões dos casos de uso do sistema e do projeto, a equipe de desenvolvimento do sistema atribui a cada caso de uso sua prioridade, seguindo o método descrito por Bezerra (2007), que relaciona o risco de desenvolvimento e a prioridade do usuário:

1. *Risco alto e prioridade alta*, são casos de uso críticos e devem ser implementados o quanto antes;

2. *Risco alto e prioridade baixa*, são casos de uso de alto risco, porém sua necessidade de implementação é baixa e deve-se negociar com o cliente a necessidade de implementação;

3. *Risco baixo e prioridade alta*, possuem risco baixo, porém devem ser priorizados após os de alto risco;

4. *Risco baixo e prioridade baixa*, são os casos de uso de menor importância e os passíveis de corte em caso de atrasos no desenvolvimento.

Os casos de uso são então organizados em uma tabela de incrementos, que serão posteriormente selecionados para a realização dos ciclos, conforme ilustrado na Tabela 3. A tabela é composta pelo número sequencial do incremento, a descrição do incremento, a prioridade de implementação e o status atual do incremento. Nesta tabela também serão registradas as correções necessárias após a conclusão de um ciclo, a partir da verificação do cliente.

Tabela 3 – Modelo de lista de incrementos.

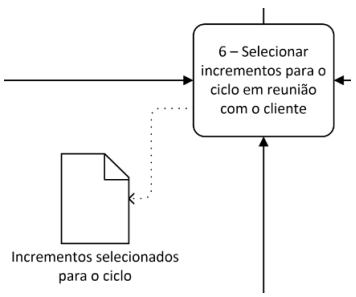
Nº	**Descrição**	**Prioridade**	**Status**
01	Cadastro de Clientes	03	Em aberto
02	Emissão de Nota Fiscal	01	Realizado no ciclo 1
03	Cadastro de Produtos	03	Em implementação

Saídas

O subprocesso “Tabular e priorizar os casos de uso, gerando lista de incrementos” gera como saída a lista de incrementos a serem implementados no novo sistema.

4.1.2.6 Atividade 6 - Selecionar incrementos para o ciclo

Figura 23 – Atividade 6.



Esta atividade tem por objetivo selecionar os incrementos que serão realizados durante o ciclo de desenvolvimento, através de uma reunião com o cliente.

Entradas

A atividade “Selecionar incrementos para o ciclo” possui as seguintes entradas:

- Diagrama de casos de uso do novo sistema (saída do subprocesso 3);
- Casos de uso expandidos do novo sistema (saída do subprocesso 3);
- Diagrama de classes do sistema (saída do subprocesso 4);
- Modelo entidade-relacionamento do sistema (saída do subprocesso 4);
- Lista de incrementos (saída do subprocesso 5);

Atividades

A seleção de incrementos para o ciclo é realizada através de uma reunião entre o cliente e a equipe de desenvolvimento. A definição de quais incrementos serão selecionados para o ciclo é determinada pela equipe, juntamente com o cliente e levando em consideração:

- O tempo de duração do ciclo de desenvolvimento;
- A priorização dos incrementos;
- A dependência entre os incrementos;
- As necessidades do cliente;
- Os incrementos de correção (caso existam) e sua prioridade;

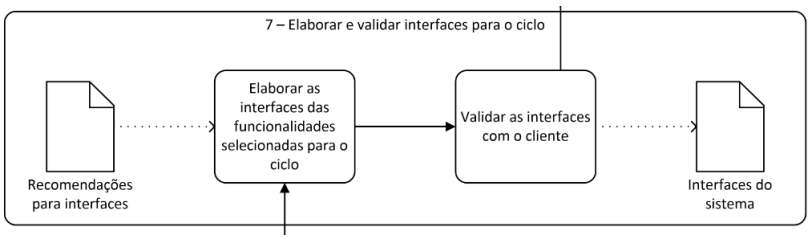
Saídas

A atividade “Selecionar incrementos para o ciclo” gera a atualização dos status dos incrementos a serem realizados no ciclo de desenvolvimento na lista de incrementos do sistema.

4.1.2.7 Subprocesso 7 - Elaborar e validar interfaces do ciclo

Nesta etapa do processo são elaboradas as interfaces das funcionalidades selecionadas para o ciclo, observando mudanças solicitadas pelo cliente e também recomendações para a elaboração de interfaces.

Figura 24 – Subprocesso expandido 7.



Entradas

O subprocesso “Elaborar e validar interfaces do ciclo” possui as seguintes entradas:

- Interfaces anteriores do sistema;
- Diagrama de casos de uso do novo sistema (saída do subprocesso 3);
- Casos de uso expandidos do novo sistema (saída do subprocesso 3);
- Normas e recomendações para a elaboração de interfaces;
- Incrementos selecionados para o ciclo de desenvolvimento (saída da atividade 6);

Atividades

A elaboração das interfaces deve utilizar como referência as interfaces antigas, procurando manter a identidade anterior do sistema e realizando as modificações necessárias, evitando-se assim a perda de identidade ao sistema antigo.

Como forma de validação da usabilidade do sistema, deve-se realizar a análise das interfaces do ciclo utilizando-se ao menos um dos métodos de avaliação de usabilidade descritos por Junior e Silva (2003), a fim de estabelecer um nível adequado de usabilidade do sistema. A utilização dessa proposta se justifica pela necessidade de avaliar a insatisfação do usuário no uso do sistema e a identificação de possíveis falhas de usabilidade. Dessa forma, a avaliação da usabilidade é um fator crítico para o sucesso da reengenharia do sistema legado.

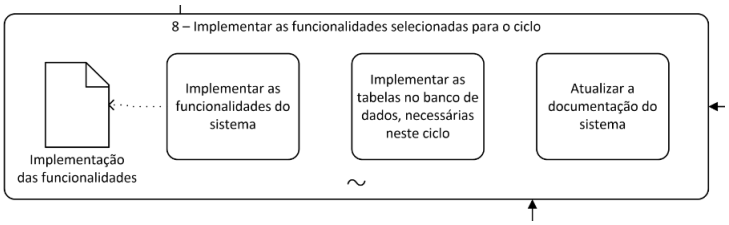
Adicionalmente, as interfaces devem ser validadas pelo cliente do sistema antes da programação das mesmas, a fim de evitar a inclusão de incrementos de correção de interfaces na lista de incrementos.

Saídas

O subprocesso “Elaborar e validar interfaces do ciclo” gera como saída as interfaces dos incrementos selecionados para o ciclo.

4.1.2.8 Subprocesso 8 - Implementar as funcionalidades do ciclo

Figura 25 – Subprocesso expandido 8.



Neste subprocesso é realizada a implementação da funcionalidade selecionada para o ciclo de desenvolvimento, envolvendo a codificação do sistema utilizando-se as ferramentas definidas na lista de decisões técnicas. Adicionalmente, nesta etapa do sistema também são implantadas no Banco de Dados as tabelas necessárias ao sistema neste ciclo.

As inconsistências identificadas no diagrama de classes e no modelo entidade-relacionamento deverão ser corrigidas, pois a documentação deve refletir a implementação realizada.

Entradas

O subprocesso “Implementar as funcionalidades do ciclo” possui as seguintes entradas:

- Diagrama de casos de uso do novo sistema (saída do subprocesso 3);
- Casos de uso expandidos do novo sistema (saída do subprocesso 3);
- Lista de decisões técnicas (saída do subprocesso 3);
- Diagrama de classes do sistema (saída do subprocesso 4);
- Modelo entidade-relacionamento do sistema (saída do subprocesso 4);
- Incrementos selecionados para o ciclo de desenvolvimento (saída da atividade 6);
- Interfaces do sistema (saída do subprocesso 7);

Atividades

No subprocesso “Implementar as funcionalidades do ciclo” são realizadas as seguintes atividades:

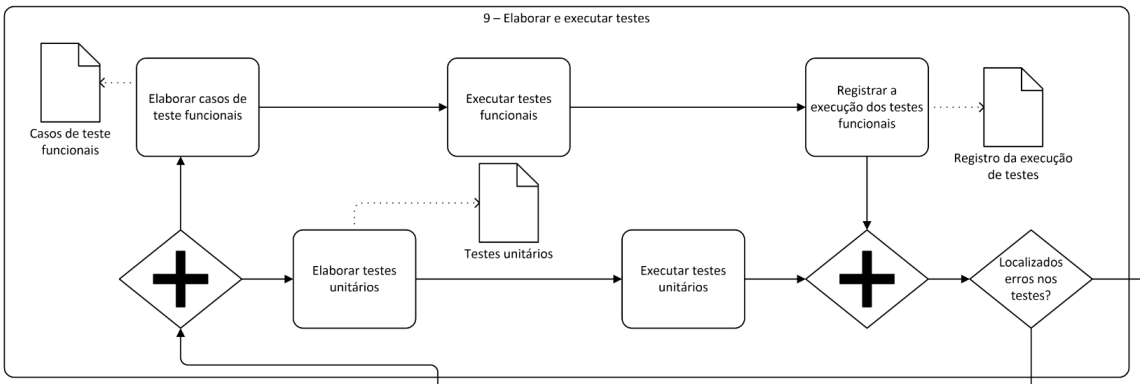
- Implementar as funcionalidades do sistema;
- Implementar as tabelas necessárias, no banco de dados do sistema;
- Atualizar a documentação do sistema;

Saídas

O subprocesso “Implementar as funcionalidades do ciclo” gera como saída as implementações das funcionalidades dos incrementos selecionados para o ciclo de desenvolvimento.

4.1.2.9 Subprocesso 9 - Elaborar e executar testes

Figura 26 – Subprocesso expandido 9.



O subprocesso de elaboração e execução de testes envolve a elaboração e execução de testes funcionais e testes unitários. A realização de testes unitários poderá ser apoiada por ferramentas que automatizem o processo. O controle dos testes funcionais deverá ser realizado através de uma ferramenta que permita a criação dos testes funcionais e registre a execução dos casos de testes e seus resultados.

A realização de testes deve se focar em verificar a funcionalidade do sistema e também em verificar se a implementação da solução corresponde a mesma funcionalidade equivalente do software legado e as modificações solicitadas.

Entradas

O subprocesso “Elaborar e executar testes” possui as seguintes entradas:

- Diagrama de casos de uso do novo sistema (saída do subprocesso 3);
- Casos de uso expandidos do novo sistema (saída do subprocesso 3);

- Lista de decisões técnicas (saída do subprocesso 3);
- Plano de teste (saída do subprocesso 3);
- Diagrama de classes do sistema (saída do subprocesso 4);
- Modelo entidade-relacionamento do sistema (saída do subprocesso 4);
- Incrementos selecionados para o ciclo de desenvolvimento (saída da atividade 6);
- Implementações das funcionalidades dos incrementos selecionados para o ciclo de desenvolvimento (saída do subprocesso 8);

Atividades

No subprocesso “Elaborar e executar testes” são realizadas as seguintes atividades:

- Elaborar e executar testes unitários;
- Elaborar casos de teste funcionais;
- Executar e registrar a execução de casos de teste funcionais;

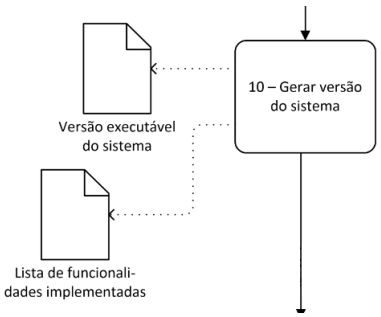
Saídas

A execução do subprocesso “Elaborar e executar testes” gera os seguintes artefatos:

- Implementações das funcionalidades testadas;
- Casos de testes com o registro de execução;

4.1.2.10 Atividade 10 - Gerar versão do sistema

Figura 27 – Atividade 10.



Após a realização dos testes é gerada uma versão do sistema, que incluirá todas as funcionalidades realizadas neste ciclo e também as funcionalidades realizadas nos ciclos anteriores, para posterior implantação e avaliação por parte do cliente.

A versão será identificada pelo número do ciclo que estiver em execução, como forma de manter o histórico das versões do sistema. Ex.: caso esteja ocorrendo o terceiro ciclo de desenvolvimento, ao final deste será gerada a versão 3 do sistema.

Entradas

A atividade “Gerar versão do sistema” possui as seguintes entradas:

- Implementação das funcionalidades testadas (saída do subprocesso 9);
- Funcionalidades implementadas em ciclos anteriores;

Atividades

- Gerar versão do sistema;

Saídas

A atividade “Gerar versão do sistema” gera os seguintes artefatos:

- Versão executável do sistema;
- Lista de funcionalidades implementadas no ciclo;

4.1.2.11 Subprocesso 11 - Organizar documentação do sistema

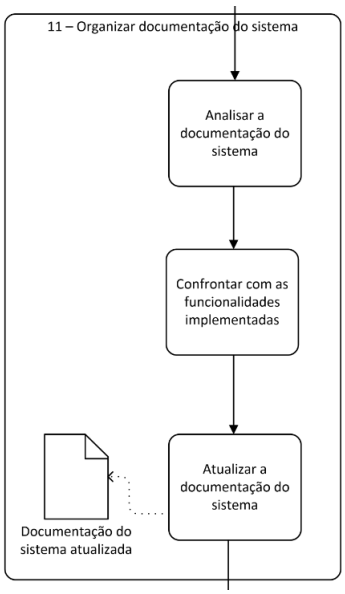
Após a implementação, teste e geração da versão do sistema, a documentação deve ser revisada, com o objetivo de verificar se as inconsistências identificadas nas etapas anteriores foram devidamente corrigidas e também se os diagramas foram complementados com informações omitidas inicialmente.

Entradas

O subprocesso “Organizar documentação do sistema” possui as seguintes entradas:

- Documentação do sistema;
- Implementação das funcionalidades testadas (saída do subprocesso 9);

Figura 28 – Subprocesso expandido 11.



Atividades

No subprocesso “Organizar documentação do sistema” são realizadas as seguintes atividades:

- Analisar a documentação do sistema e da Implementação das funcionalidades;
- Confrontar a documentação do sistema com as funcionalidades do sistema;
- Atualizar a documentação do sistema;

Saídas

O subprocesso “Organizar documentação do sistema” gera como saída a documentação atualizada do sistema.

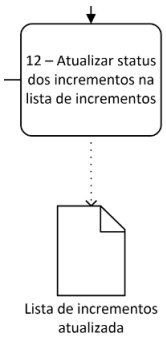
4.1.2.12 Atividade 12 - Atualizar status dos incrementos

Após a conclusão do ciclo, a lista de incrementos deve ser atualizada, alterando os status dos incrementos da lista para “Realizado no ciclo”, acompanhado do número do ciclo em execução, referenciando em qual ciclo e versão do sistema o incremento foi realizado.

Entradas

A atividade “Atualizar status dos incrementos na lista de incrementos” possui as seguintes entradas:

Figura 29 – Atividade 12.



- Lista de incrementos do sistema (saída do subprocesso 5);
- Lista de funcionalidades implementadas no ciclo (saída da atividade 10);

Atividades

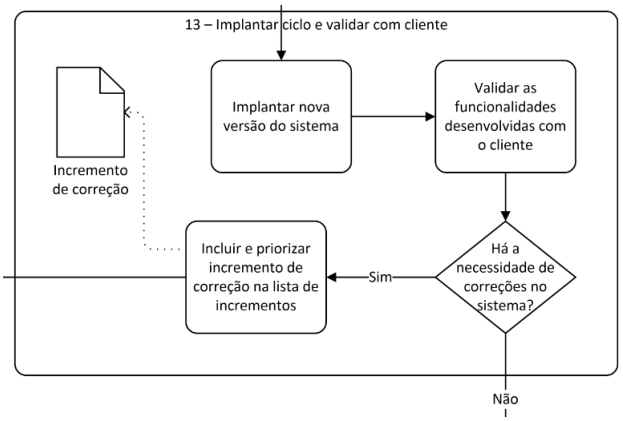
- Atualizar a lista de incrementos do sistema;

Saídas

A atividade “Atualizar status dos incrementos na lista de incrementos” gera como saída a lista de incrementos com os status dos incrementos atualizados.

4.1.2.13 Subprocesso 13 - Implantar ciclo e validar

Figura 30 – Subprocesso expandido 13.



A nova versão do sistema é então implantada no cliente, realizando a migração das funcionalidades que até então eram utilizadas no sistema legado para o novo sistema. O

cliente então valida as novas funcionalidades implementadas no ciclo e avalia a necessidade de correções no sistema, para posteriormente serem incluídas na lista de incrementos do sistema.

Entradas

O subprocesso “Implantar ciclo e validar” possui como entrada a versão executável do sistema (saída da atividade 10);

Atividades

No subprocesso “Implantar ciclo e validar” são realizadas as seguintes atividades:

- Implantar nova versão do sistema no cliente;
- Validar as funcionalidades desenvolvidas e migradas (pelo cliente), gerando incrementos de correção, de acordo com a necessidade. Caso sejam gerados incrementos de correção, os mesmos devem ser priorizados, seguindo o método proposto por Bezerra (2007) para a priorização de casos de uso, e posteriormente inseridos na lista de incrementos. A correção será selecionada e realizada em um ciclo posterior, de acordo com a prioridade definida para o mesmo;

Saídas

A realização das atividades presentes no subprocesso “Implantar ciclo e validar” pode gerar a inclusão de incrementos de correção na lista de incrementos, caso sejam necessárias correções no sistema.

4.2 Relato da experiência

Este seção do trabalho concentra-se no relato da experiência obtida através da aplicação do processo de reengenharia inserido na disciplina de Evolução de Software, ofertada aos acadêmicos do 6º Semestre do curso de Engenharia de Software no segundo semestre letivo de 2013.

4.2.1 Organização da disciplina de Evolução de Software

A disciplina de Evolução de Software faz parte do grupo de disciplinas obrigatórias do curso de Engenharia de Software, sendo ofertada aos discentes no 6º semestre. A disciplina é composta por 2 créditos, totalizando 30 horas. Neste contexto, o objetivo da disciplina é “conhecer os fundamentos, técnicas e processos de evolução de software para

que seja possível gerenciar a evolução de sistemas legados" (Universidade Federal do Pampa, 2012, p. 59).

A turma de Evolução de Software do 2º Semestre letivo de 2013 é composta por 15 alunos matriculados. De forma a organizar a realização do trabalho e permitir um melhor acompanhamento da aplicação do processo, a turma foi dividida em 03 grupos, com equipes compostas por 05 integrantes.

O acompanhamento da execução do processo de reengenharia na disciplina foi realizado pelo proponente deste trabalho, com o apoio e o acompanhamento da professora responsável pela disciplina, que também é co-orientadora neste trabalho. Para a realização do acompanhamento houve a participação nas aulas que ocorriam aos sábados pela manhã e a disponibilidade de horário às terças e quintas-feiras para atendimento extra-classe, caso algum dos grupos manifestasse necessidade. Adicionalmente, disponibilizou-se o endereço de *e-mail* para a resolução de dúvidas e questões pontuais em casos que não fosse necessário o encontro presencial com os grupos.

Para a execução do processo de reengenharia utilizou-se como sistema legado o Sistema de Gerenciamento de Concursos Públicos (GCP) da UNIPAMPA, que é responsável por registrar as informações referentes a realização de concursos públicos de docentes e automatizar a emissão de relatórios por parte da bancas examinadoras dos concursos. Tal sistema foi desenvolvido pelo campus Alegrete da UNIPAMPA.

O sistema de Gerenciamento de Concursos Públicos foi submetido à atividade de avaliação do software legado, descrita anteriormente na subseção 4.1.2.1. As evidências da realização da avaliação do sistema estão disponíveis no Apêndice B. A partir da avaliação realizada, constatou-se que o sistema foi submetido a várias mudanças, foi elaborado através do paradigma de programação estruturada, utiliza o armazenamento dos dados através de arquivos e possui diversos problemas de validação de campos, inconsistência e padronização de interfaces e usabilidade, além de ser compatível apenas com Sistemas Operacionais *Windows*. Por possuir tais características, identificou-se que o sistema possui vários pontos a serem evoluídos, com o objetivo de melhorar a qualidade e facilitar manutenções futuras.

Em relação ao valor de negócio, o sistema possui alto valor de negócio, pois ainda é extremamente necessário para a universidade. Para potencializar sua contribuição para a universidade, o sistema necessita de atualizações nos processos de negócio para que seja compatível com a realização de concursos para professores temporários e substitutos, onde muitas vezes não há a realização de prova escrita, sendo que da forma como o sistema está atualmente, não é possível a utilização do mesmo sem que sejam realizadas todas as provas.

Por possuir tais características, a avaliação do sistema realizada indicou a reenge-

nharia como estratégia de manutenção recomendada, desta forma o sistema é compatível com a realização das práticas de reengenharia presentes no processo.

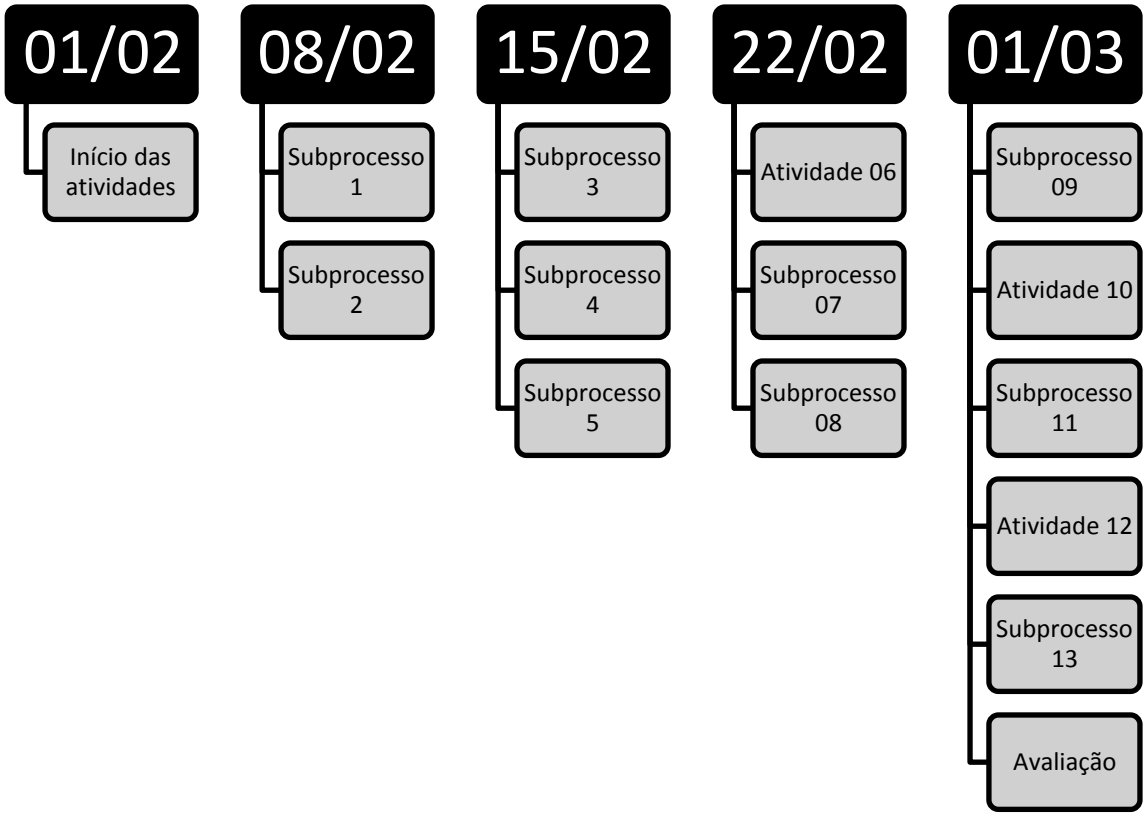
A aplicação do processo na disciplina ocorreu no mês de fevereiro de 2014. Para a execução do processo utilizou-se um período de 04 semanas, sendo que o estabelecimento desse prazo foi determinado pelos seguintes fatores:

1. A necessidade de refinamento e amadurecimento da proposta de processo de reengenharia, de forma que fosse possível a oferta na disciplina;
2. A aplicação da proposta ocorreu no mês de fevereiro em função da compatibilidade com os conteúdos teóricos abordados na disciplina necessários para a aplicação do processo;
3. O prazo necessário para a aplicação do processo, aplicação da avaliação, coleta e análise de resultados e conclusão do presente trabalho de conclusão de curso;

O cronograma de execução de atividades neste período foi organizado da seguinte forma: Na aula de 01 de fevereiro planejou-se o início das atividades, com a explanação da proposta, envolvendo a apresentação do processo e de suas atividades e o cronograma aos alunos da disciplina. Adicionalmente, nesta data também foi prevista a organização dos grupos e a distribuição dos materiais necessários à realização das atividades através do Ambiente Virtual de Aprendizagem (AVA) *Moodle* da UNIPAMPA.

Para a realização das atividades da disciplina organizou-se o cronograma como segue: Para a primeira semana de atividades (de 02 a 08 de fevereiro) planejou-se a realização das atividades presentes nos subprocessos 01 e 02. A segunda semana de atividades (de 09 a 15 de fevereiro) envolve as atividades presentes nos subprocessos 03, 04 e 05. Na terceira semana de atividades (de 16 a 22 de fevereiro), planejou-se a realização da atividade 06 e das atividades presentes nos subprocessos 07 e 08. Concluindo o cronograma de atividades, a última semana (de 23 de fevereiro a 01 de março) foi reservada para a conclusão e fechamento das atividades, envolvendo a realização das atividades 10 e 12 e das atividades presentes nos subprocessos 09, 11 e 13. Adicionalmente, a aula do dia 01 de março foi reservada para o preenchimento da avaliação referente ao uso do processo na disciplina. A Figura 31 ilustra a organização do cronograma de atividades na disciplina.

Figura 31 – Cronograma da aplicação do processo de reengenharia na disciplina.



Para a realização do acompanhamento semanal do andamento das atividades utilizou-se uma tabela, composta pelos grupos e pelos artefatos gerados ao final de cada atividade ou subprocesso (ver Tabela 4). Ao final de cada aula presencial verificou-se o andamento das atividades grupo a grupo, registrando na tabela o andamento semanal dos trabalhos. A notação "✓" indica que a atividade foi realizada conforme o previsto. A notação "✗" indica que a atividade não foi realizada e a notação "✗" acompanhada por "(✓em <data>)" indica que a atividade não foi concluída na semana prevista, porém foi concluída na semana indicada pela data.

Tabela 4 – Tabela de acompanhamento da realização de atividades

Semanas	Atividades	Artefatos Gerados	Grupo 1	Grupo 2	Grupo 3
Semana 1 - 1 a 08/02	**Subprocesso 01**	Relatório de avaliação do software legado	✓	✓	✓
		Diagrama de Casos de Uso	✓	✗(✓em 15/02)	✗(✓em 15/02)
	Subprocesso 02	Modelo Lógico / Relacional	✗(✓em 15/02)	✗(✓em 22/02)	✗(✓em 01/03)
		Diagrama de Fluxo de Dados	✗(✓em 01/03)	✗(✓em 22/02)	✗(✓em 22/02)
Semana 2 - 09 a 15/02	**Subprocesso 03**	Casos de uso Expandidos	✗(✓em 01/03)	✗(✓em 22/02)	✗(✓em 22/02)
		Diagrama de Casos de Uso atualizado	✗(✓em 01/03)	✗(✓em 22/02)	✗(✓em 22/02)
		Modelo Conceitual	✗(✓em 01/03)	✗	✗
		Lista de Decisões Técnicas preenchida	✓	✗(✓em 22/02)	✗(✓em 01/03)
		Plano de Teste	✗	✗	✗(✓em 01/03)
	Subprocesso 04	Diagrama de Classe	✗(✓em 01/03)	✗(✓em 22/02)	✗(✓em 22/02)
		Modelo Entidade / Relacionamento	✗(✓em 01/03)	✗(✓em 22/02)	✗(✓em 01/03)
	Subprocesso 05	Lista de Incrementos	✗(✓em 01/03)	✗(✓em 22/02)	✗(✓em 22/02)
Semana 3 - 16 a 22/02	**Atividade 6**	Incrementos Selecionados para o Ciclo	✓	✓	✓
	Subprocesso 07	Interfaces do Sistema	✗(✓em 01/03)	✓	✗(✓em 01/03)
	Subprocesso 08	Implementação das Funcionalidades	✗(✓em 01/03)	✗	✗(✓em 01/03)
Semana 4 - 23/02 a 01/03	**Subprocesso 09**	Casos de Teste Funcionais	✓	✗	✓
		Testes Unitários	✓	✗	✓
		Registro da Execução dos Testes	✓	✗	✓
	Atividade 10	Versão executável Sistema	✓	✗	✓
		Lista de Funcionalidades Implementadas	✓	✗	✓
	Subprocesso 11	Documentação do sistema atualizada	✓	✗	✓
	Atividade 12	Lista de incrementos atualizada	✓	✗	✓
	Subprocesso 13	Incrementos de Correção	✓	✗	✓

4.2.2 Execução do cronograma de atividades

Nesta seção do trabalho descreve-se a aplicação do processo na disciplina de Evolução de Software, relatando o andamento das atividades semana a semana.

4.2.2.1 01/02 - Apresentação do processo

No primeiro dia de atividades na disciplina de Evolução de Software, explanou-se o processo de reengenharia através de uma apresentação. Tal apresentação era composta pelo processo e a descrição de cada um dos subprocessos e atividades da modelagem BPMN. Adicionalmente, a apresentação incluiu o cronograma de realização das atividades, presente na Figura 31, e os momentos de interação com o cliente, de acordo com as atividades do processo. Houve um momento para que os participantes tirassem dúvidas sobre o processo. Posteriormente, apresentou-se o sistema legado que seria utilizado para a execução do processo e foram organizados os grupos para a execução da reengenharia.

4.2.2.2 02 a 08/02 - Primeira semana

Para a primeira semana de atividades esteve prevista a realização das atividades presentes no subprocesso 01 (Realizar avaliação do software legado, sob perspectiva de negócio e técnica) e no subprocesso 02 (Realizar análise técnica do sistema legado). Os artefatos entregáveis previstos para serem realizados nessa semana eram os seguintes:

1. Subprocesso 01: Relatório da avaliação do software legado;
2. Subprocesso 02: Diagrama de casos de uso do sistema, modelo lógico-relacional e diagrama de fluxo de dados;

Dessa forma, na primeira semana de atividades todos os grupos conseguiram concluir com sucesso as atividades referentes ao subprocesso 01 (conforme registro na Tabela 4), sendo que as atividades relativas ao subprocesso 02, devido ao fato de envolverem a engenharia reversa do sistema legado, necessitaram de um pouco mais de tempo e não foram concluídas na primeira semana de trabalho. Neste contexto, percebeu-se a necessidade de um tempo maior entre a apresentação da proposta na primeira semana de trabalho e a realização das atividades do subprocesso 2, com o objetivo de proporcionar um maior tempo para o estudo do processo e do sistema legado que passará pela reengenharia, facilitando a atividade de engenharia reversa.

4.2.2.3 09 a 15/02 - Segunda semana

Durante a segunda semana de atividades esteve prevista a realização das atividades referentes ao subprocesso 03 (Identificar contexto do novo sistema), ao subprocesso

04 (Elaborar projeto do sistema) e e ao subprocesso 05 (Tabular e priorizar os casos de uso, gerando lista de incrementos). Para tais atividades, foi prevista a elaboração dos seguintes artefatos:

1. Subprocesso 03: Casos de uso expandidos, diagrama de casos de uso atualizado, modelo conceitual, lista de decisões técnicas e plano de teste;
2. Subprocesso 04: Modelo entidade-relacionamento e diagrama de classes do sistema;
3. Subprocesso 05: Lista de incrementos;

Em função do acúmulo de trabalho da semana anterior, e devido ao nível de complexidade das atividades presentes no subprocesso 02, os alunos utilizaram a semana e o período de aula para a realização das atividades do subprocesso 02, sendo que as atividades relativas a esse subprocesso foram concluídas parcialmente por todos os grupos (ver registros do acompanhamento na Tabela 4).

4.2.2.4 16 a 22/02 - Terceira semana

Na terceira semana de execução do processo esteve prevista a realização da atividade 06 (Selecionar incrementos para o ciclo em reunião com o cliente), do subprocesso 07 (elaborar e validar interfaces para o ciclo) e do subprocesso 08 (Implementar as funcionalidades selecionadas para o ciclo). Para a realização das atividades, preveu-se a elaboração dos seguintes artefatos:

1. Atividade 06: Relação dos incrementos selecionados para o ciclo de desenvolvimento;
2. Subprocesso 07: interfaces das funcionalidades selecionadas para o ciclo de desenvolvimento;
3. Subprocesso 08: implementação das funcionalidades do sistema, selecionadas para o ciclo de desenvolvimento;

Durante a terceira semana de execução do processo, os grupos conseguiram realizar boa parte das atividades pendentes das semanas anteriores, porém, das atividades previstas para a semana, realizou-se de forma completa apenas a atividade 06, onde, em contato com o cliente, foi selecionada a funcionalidade que será implementada no ciclo de desenvolvimento.

Para a realização do ciclo de implementação e implantação do novo sistema foi selecionado para todos os grupos o mesmo incremento, que envolvia a criação do concurso,

definindo informações como o edital do concurso, a área, a classe do concurso, dentre outras.

Em relação as atividades dos subprocessos 07 e 08, alguns grupos já estavam trabalhando nesta semana na elaboração de interfaces e na implementação do sistema, porém, não concluíram tais atividades.

4.2.2.5 23/02 a 01/03 - Quarta semana

Para a última semana de execução do processo, programou-se a conclusão das atividades, envolvendo a execução do subprocesso 09 (elaborar e executar testes), da atividade 10 (gerar versão do sistema), do subprocesso 11 (organizar documentação do sistema), da atividade 12 (atualizar status dos incrementos na lista de incrementos) e do subprocesso 13 (implantar ciclo e validar com cliente). A execução das atividades programadas para a semana geram os seguintes artefatos:

1. Subprocesso 09: casos de testes funcionais, testes unitários e registros da execução dos testes;
2. Atividade 10: versão executável do sistema e lista das funcionalidades implementadas;
3. Subprocesso 11: documentação do sistema atualizada;
4. Atividade 12: lista de incrementos atualizada;
5. Subprocesso 13: incrementos de correção;

As atividades relativas ao subprocesso 13 (Implantar ciclo e validar com o cliente) foram realizadas na aula do dia 01/03, e definiram o encerramento do ciclo incremental. Ao final da execução do processo de reengenharia obteve-se duas versões de sistemas funcionais. A funcionalidade produzida pelos grupos no ciclo incremental do sistema está disponível no Apêndice C. Foram identificadas inconsistências nas interfaces de implementação destes 02 sistemas, devido a validação das interfaces não ter sido realizada conforme o esperado no planejamento do processo, o que gerou um incremento de correção na lista de incrementos de cada grupo. Apenas o grupo 2 não conseguiu concluir as atividades previstas para o ciclo de desenvolvimento, e praticamente não evoluiu no desenvolvimento das atividades da terceira semana para a quarta semana, conforme pode ser verificado na Tabela 4.

Adicionalmente, nesta última aula presencial da execução do processo aplicou-se o questionário de avaliação da execução do processo, com o objetivo de avaliar a aplicação do processo sob 2 dimensões: Avaliação do processo e avaliação da aprendizagem. O

relato da aplicação do questionário de avaliação e dos resultados obtidos encontra-se no Capítulo 5.

5 Avaliação da aplicação do processo na disciplina de Evolução de Software

Ao término da realização das atividades do processo de reengenharia no último dia da aplicação do processo, realizou-se a avaliação através de um questionário composto por questões de cunho qualitativo, disponibilizado através de um formulário eletrônico aos alunos. De um universo de 15 alunos, foram registradas as respostas de 10 alunos no questionário.

O questionário era composto por dois grupos de questões, que foram chamados de dimensões. Na primeira dimensão foram elaboradas 07 (sete) questões que tinham o foco de avaliar o funcionamento do processo e a aceitação do uso do mesmo na disciplina. Na segunda dimensão foram elaboradas 06 (seis) questões com o foco de avaliar a aprendizagem obtida pelos alunos com a aplicação do processo na disciplina. A elaboração do questionário de avaliação a partir destas dimensões é baseada na avaliação de uso do processo realizada nos trabalhos de Petrenko et al. (2007) e Perez-Castillo et al. (2013).

Nas seções seguintes são descritos os resultados obtidos. Na seção 5.1 são descritas as questões e os resultados referentes a primeira dimensão do processo. Na seção 5.2 são descritas as questões e os resultados referentes a segunda dimensão do processo.

5.1 Dimensão 1 - Avaliação do processo

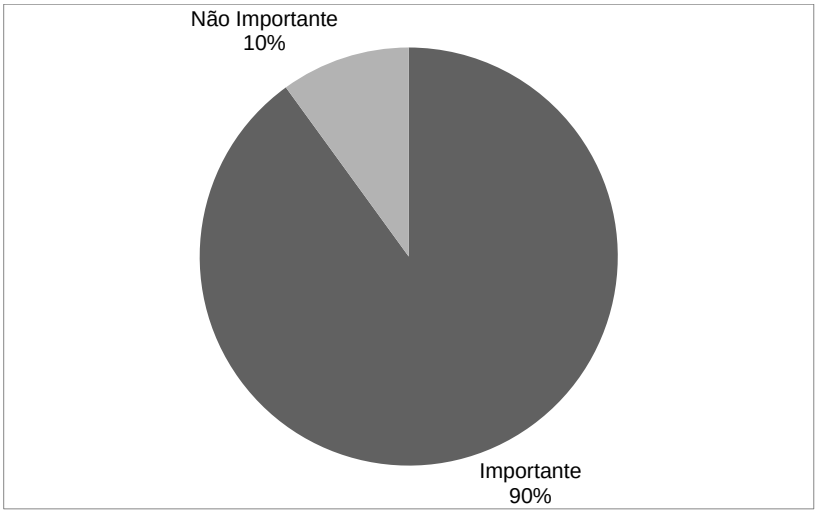
Nesta dimensão da avaliação buscou-se obter informações relativas ao processo e a execução do mesmo na disciplina de evolução de software. Das 07 questões presentes nesta dimensão, 04 eram dirigidas e 03 eram abertas, onde os alunos puderam descrever livremente sugestões e considerações relacionadas à experiência.

Questão 1 - Consideras importante o uso de um processo específico (ou de um processo com adaptações) para a reengenharia de sistemas legados?

A primeira pergunta presente no instrumento de avaliação questionava o aluno em relação a importância de uso de um processo específico (ou de um processo com adaptações) para a realização da reengenharia de sistemas legados. Caso o aluno respondesse que não considerava importante, era convidado a justificar este posicionamento.

Como podemos observar na Figura 32, 09 alunos consideram que é necessária a adoção de um processo específico para a reengenharia de sistemas legados. O aluno que discordou do uso de um processo específico argumentou que em projetos de pequeno

Figura 32 – Importância do uso de processo para a reengenharia



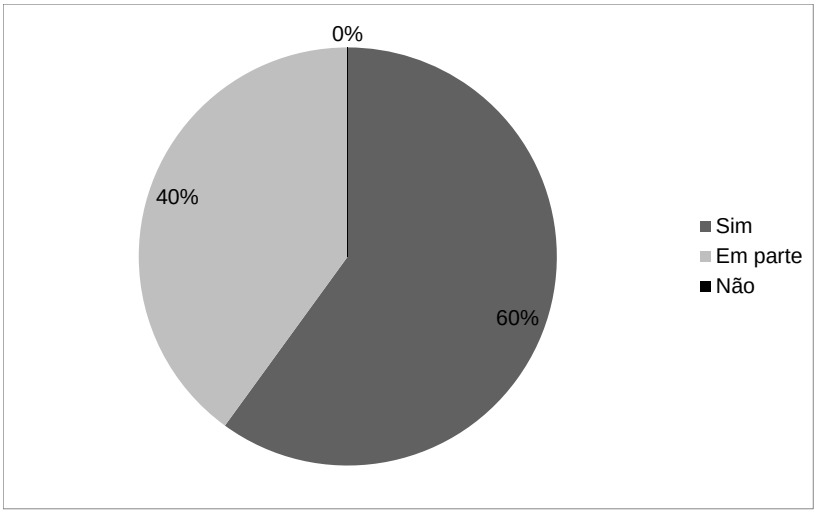
porte, baixo custo e importância não é necessário o uso de um processo específico, porém, considera o uso muito válido no cenário proposto em evolução de software. Tal resultado evidencia que os alunos consideram importante o apoio de um processo para o suporte da realização de reengenharia em um sistema legado.

Questão 2 - Considera que a atividade prática orientada pelo processo forneceu base de conhecimento para a prática profissional de Reengenharia de Software?

A segunda questão desta dimensão do questionário tinha por objetivo avaliar se o aluno considerava que a atividade prática orientada pelo processo forneceu base de conhecimento para a prática profissional de reengenharia de sistemas legados. A Figura 33 ilustra a resposta dos acadêmicos relativa a esta questão.

60% das respostas consideram que foram adquiridos conhecimentos para a prática profissional, enquanto 40 % consideraram que os conhecimentos foram em parte adquiridos. Não foram registradas respostas que indicassem que não houve aprendizado com a realização da atividade prática.

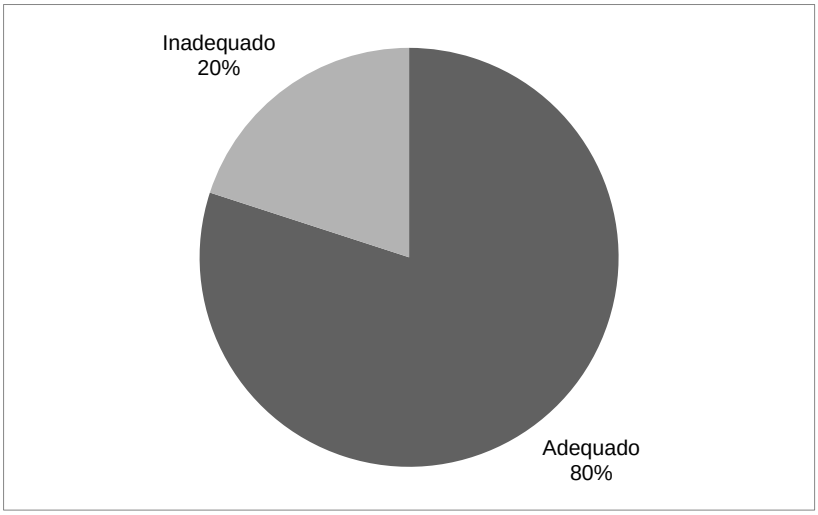
Figura 33 – Atividade forneceu base de conhecimento?



Questão 3 - Considera adequado o número de componentes por grupo?

A terceira pergunta da primeira dimensão questionava se o aluno considerava adequado o número de componentes por grupo de trabalho. Na Figura 34 podemos visualizar as respostas obtidas.

Figura 34 – Número de componentes por grupo



80 % das respostas indicam que o número de alunos por grupo foi adequado, enquanto 20% discordam do número de componentes por grupo de trabalho. Como justificativa por considerar que o número de componentes por grupo não era adequado, uma das respostas considera que 03 pessoas seria um número suficiente de componentes por grupo, devido as características do projeto, e a dificuldade de realizar a divisão das tarefas para todos os componentes do grupo. Outra resposta registrada se opõe a anterior, indi-

cando que o número de componentes por grupo seria adequado se todos os integrantes trabalhassem, o que indica que não houve uma divisão adequada das tarefas no grupo, e alguns componentes ficaram mais sobrecarregados de tarefas do que outros. Tal indicador interfere apenas na aplicação prática do processo, pois o processo foi elaborado sem a limitação de número de componentes para a equipe de trabalho, porém em futuras aplicações do processo é necessário avaliar a possibilidade de se reduzir o número de componentes por grupo.

Questão 4 - Avalie os seguintes aspectos:

A quarta questão desta dimensão do questionário teve por objetivo avaliar alguns aspectos específicos da organização do processo de reengenharia e também avaliar a aplicação do mesmo na disciplina. Para cada critério a ser avaliado eram disponibilizadas as seguintes opções: Péssimo, ruim, regular, bom e ótimo. Percebe-se uma tendência das respostas variando entre bom e ótimo. A Tabela 5 resume as respostas obtidas nesta questão.

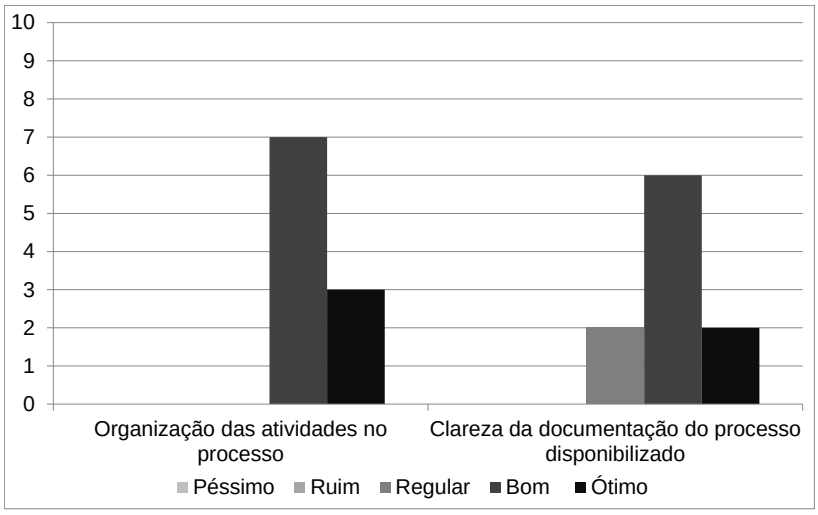
Tabela 5 – Respostas obtidas na questão 4

Critério	Péssimo	Ruim	Regular	Bom	Ótimo
Organização das atividades no processo	0%	0%	0%	**70%**	**30%**
Clareza da documentação disponibilizada	0%	0%	**20%**	**60%**	**20%**
Tempo disponível para a realização das atividades	**10%**	**30%**	**60%**	0%	0%
Aproveitamento do tempo disponibilizado	0%	0%	**40%**	**50%**	**10%**
Coerência com a proposta da disciplina	0%	0%	0%	**40%**	**60%**
Utilização de um sistema legado em operação	0%	0%	**20%**	**60%**	**20%**
Utilização de avaliação de estratégia de manutenção (primeira etapa do processo	0%	0%	0%	**80%**	**20%**
Utilização de proposta de desenvolvimento incremental (segunda etapa do processo)	0%	0%	**10%**	**70%**	**20%**

A Figura 35 representa as respostas relativas aos critérios de organização das atividades do processo e de clareza da documentação do processo disponibilizada.

Em relação à organização das atividades do processo, 70% das respostas consideraram a organização como boa e 30% como ótima e em relação à clareza da documentação do processo que foi disponibilizada aos alunos para que executassem o processo, 20% consideraram a mesma regular, 60% consideraram boa e 20% consideraram como ótima. Os resultados obtidos através destes critérios evidenciam que as atividades do processo

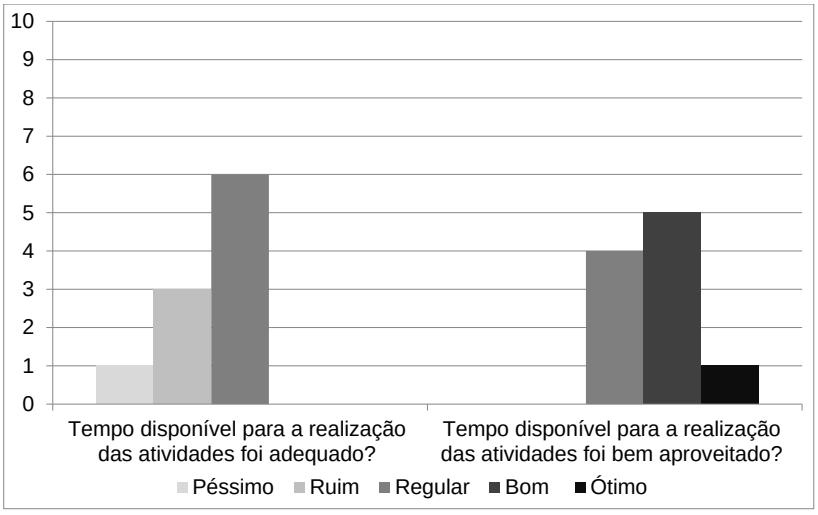
Figura 35 – Organização e clareza do processo.



estão bem organizadas e que o processo possui uma documentação clara e objetiva, o que permite a sua aplicação didática na disciplina.

A Figura 36 agrupa os resultados obtidos em relação aos critérios de adequação e de aproveitamento do tempo disponibilizado para a execução do processo na disciplina.

Figura 36 – Avaliação do tempo disponibilizado



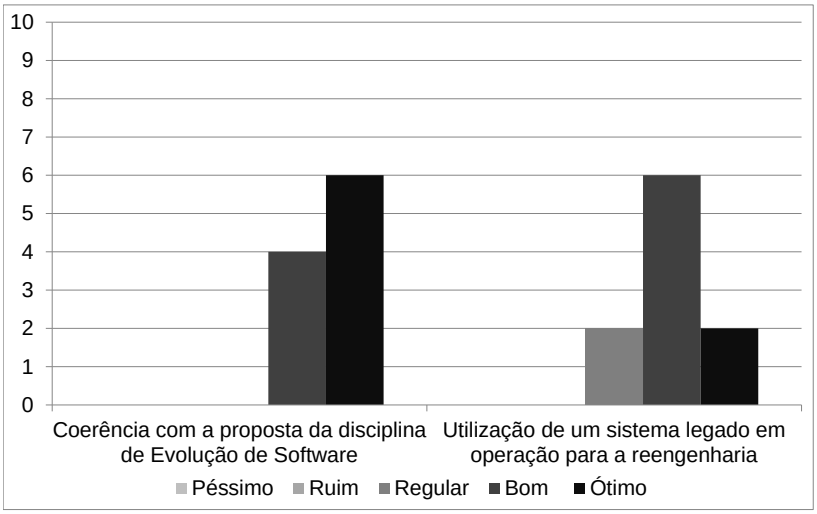
Em relação ao tempo disponível para a realização das atividades, 10% das respostas indicam que o tempo foi péssimo, 30% apontam que foi ruim e 60% indicam que o tempo foi regular. Em relação ao aproveitamento do tempo disponibilizado para a realização das atividades, 40% dos alunos consideram regular, 50% consideram como bom e 10% consideram como ótimo.

Ao se analisar conjuntamente os dois critérios, percebe-se que o tempo disponibilizado para a realização das atividades foi inadequado, sendo necessário um maior prazo para a realização das atividades. Porém, do tempo disponibilizado, os alunos consideram que foi bem aproveitado.

Ao se comparar estas informações com a execução do processo e o acompanhamento realizado pela Tabela 4, identifica-se que o tempo não foi dimensionado adequadamente, pois os grupos não conseguiram acompanhar o planejamento semanal de entregas, e um dos grupos, inclusive, não conseguiu concluir com sucesso as atividades. A estimativa de tempo necessário para a realização das atividades deve ser proporcional ao tamanho e complexidade do sistema legado que será utilizado para as atividades práticas e deve ser revista em utilizações futuras do processo.

A Figura 37 agrupa os resultados obtidos em relação aos critérios de coerência da proposta com a disciplina e do uso de um sistema legado em operação paras as atividades práticas.

Figura 37 – Coerência da proposta e utilização de um sistema legado em operação.

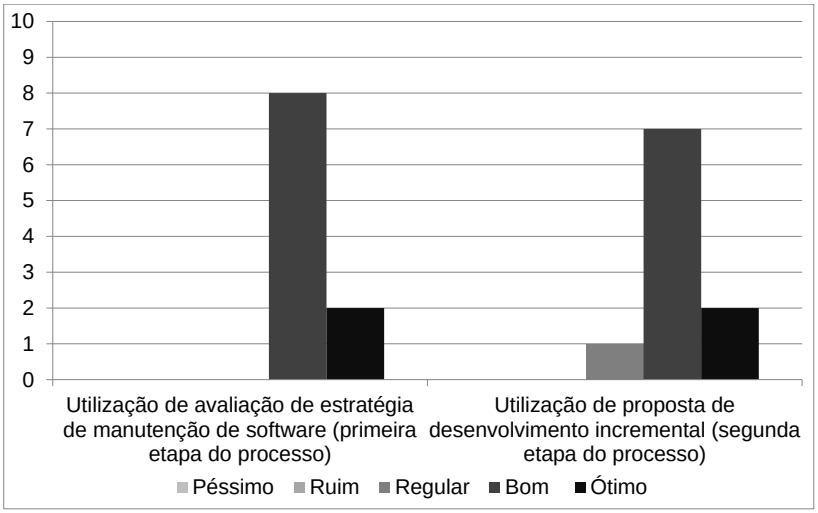


40% das respostas consideram boa e 60% consideram ótima a aderência da proposta de processo de reengenharia com a disciplina de evolução de software. A utilização de um sistema legado em operação para a reengenharia é avaliada por 20% dos alunos como regular, 60% dos alunos como boa e 20% dos alunos como ótima.

A partir da análise destas informações, percebe-se que há um grande entendimento por parte dos alunos de que o exercício prático de reengenharia é compatível com a disciplina de evolução de software. O uso de um sistema legado em operação permite que seja realizado um estudo prático da utilização do mesmo, e possibilita a identificação de melhorias que podem ser realizadas, além de permitir que os estudantes entrem em contato com um sistema em operação e que necessita de melhorias.

O próximo gráfico, representado pela Figura 38, tabula os resultados a partir do questionamento em relação a utilização de avaliação de estratégia de manutenção e da utilização de desenvolvimento incremental no processo.

Figura 38 – Uso de avaliação de estratégia de manutenção e de uso de abordagem incremental



Em relação à utilização de avaliação de estratégia de manutenção de software, 80% das respostas consideram como boa e 20% como ótima. A utilização de proposta de desenvolvimento incremental, por sua vez, registrou uma aceitação de 10% como regular, 70% como boa e 20% como ótima.

Refletindo tais resultados, percebe-se que houve uma boa aceitação por parte dos alunos da realização de atividade de análise da melhor estratégia de manutenção do sistema legado (descrita na subseção 4.1.2.1), pois permite que os alunos avaliem o sistema legado e definam qual a melhor e mais adequada estratégia de manutenção do mesmo. O uso desta abordagem no processo é um diferencial em relação aos demais relatos de experiência de ensino pesquisados.

O uso de uma abordagem incremental também é avaliado de forma positiva pois, devido ao pouco tempo disponível para a realização das atividades, esta abordagem permite que sejam realizadas todas as tarefas do ciclo de reengenharia do sistema, e sejam gerados artefatos funcionais ao final da execução do processo.

Questão 5 - O que você modificaria no processo?

A quinta questão da primeira dimensão do questionário era aberta e solicitava aos estudantes que descrevessem o que modificariam no processo. O preenchimento da pergunta era opcional e foram registradas três respostas.

A primeira resposta sugere que seja modificada a exigência de documentação completa no processo, pois na abordagem do processo as atividades relativas aos subprocessos 02 a 05, que envolvem a realização das atividades de reconstrução e transformação são realizadas para todo o sistema, e não apenas para as funcionalidades selecionadas para o ciclo.

A segunda resposta solicita que seja avaliado o tempo disponível para o desenvolvimento das atividades.

A terceira resposta sugere que seja realizado um monitoramento semanal da realização das atividades por componente e não por grupo, com objetivo de identificar quem efetivamente realiza ou não as atividades.

As respostas obtidas nesta pergunta servem de subsídio para uma avaliação futura de melhorias que podem ser realizadas no processo e na aplicação do mesmo.

Questão 6 - Pontos positivos em relação ao processo

A sexta pergunta desta primeira dimensão do questionário indagava aos alunos que indicassem pontos positivos em relação ao processo. Abaixo estão descritas algumas considerações apontadas como pontos positivos da abordagem.

- O uso do processo abrange todas as etapas da reengenharia e ajuda na organização das tarefas que devem ser realizadas para uma reengenharia com sucesso, direcionando e orientando a realização das atividades;
- O processo permite a atualização de um software, sem a necessidade de descartá-lo e ter que produzir um software novo;
- O modelo de avaliação do software legado provê noções de decisões para qualquer tipo de analista (independente de experiência técnica);
- O processo possibilita a participação constante do cliente;
- A implementação da reengenharia em ciclos e que possibilita a atualização de documentação;
- Documentação gerada é necessária para compreender o software legado. Ou seja, dá subsídio a “subida” no modelo ferradura;
- O processo permite a realização de atividades práticas, permitindo a vivência do processo de uma reengenharia de software em um sistema legado;
- As etapas iniciais de decisão sobre a manutenção do sistema e de entendimento do sistema legado estão muito bem definidas e colaboram bastante com o andamento do processo;

- O processo possibilita uma boa comunicação entre as partes interessadas;

Os pontos positivos evidenciados através das respostas obtidas evidenciam os benefícios do uso de um processo para a organização das atividades de reengenharia e que possibilita a realização de atividades práticas na disciplina.

Questão 7 - Pontos negativos em relação ao processo

A sétima e última pergunta desta primeira dimensão do questionário solicitava aos alunos que indicassem pontos negativos em relação ao processo. Essa pergunta é relevante porque indica fragilidades do processo identificadas pelos alunos que utilizaram na prática a proposta para a execução da reengenharia.

Em relação ao tempo disponível, algumas respostas dos alunos indicaram a necessidade de um tempo maior para a realização das atividades, o que possibilitaria que o trabalho realizado gerasse melhores resultados.

Outra questão relatada pelos alunos foi a falta de conhecimento da tecnologia do sistema legado. A tecnologia presente no sistema foi um obstáculo para a realização das atividades de engenharia reversa, que demandaram um tempo maior para a realização em função disso.

A terceira questão levantada é em relação a dependência de artefatos nas primeiras fases da reengenharia, o que inviabiliza a divisão de tarefas dentro do grupo, devido à dependência que as tarefas possuem.

5.2 Dimensão 2 - Avaliação da aprendizagem

A segunda dimensão da avaliação é formada por 06 questões objetivas, com o objetivo de avaliar o nível de aprendizado relacionado à reengenharia obtido através do uso do processo na disciplina.

Questão 1 - Como avaliar um software legado e definir a estratégia de manutenção para o mesmo?

A primeira questão desta dimensão questionava a forma de se avaliar um software legado e definir a estratégia para a sua manutenção. As alternativas disponíveis eram as seguintes:

1. Perguntar para o cliente qual estratégia de manutenção que ele prefere;
2. Realizar entrevista com várias pessoas envolvidas e deduzir qual é a melhor estratégia de manutenção do sistema;

3. Utilizando-se questionário sob perspectiva técnica e de negócio e analisar as respostas através de um gráfico que indica a melhor estratégia de manutenção do sistema legado;

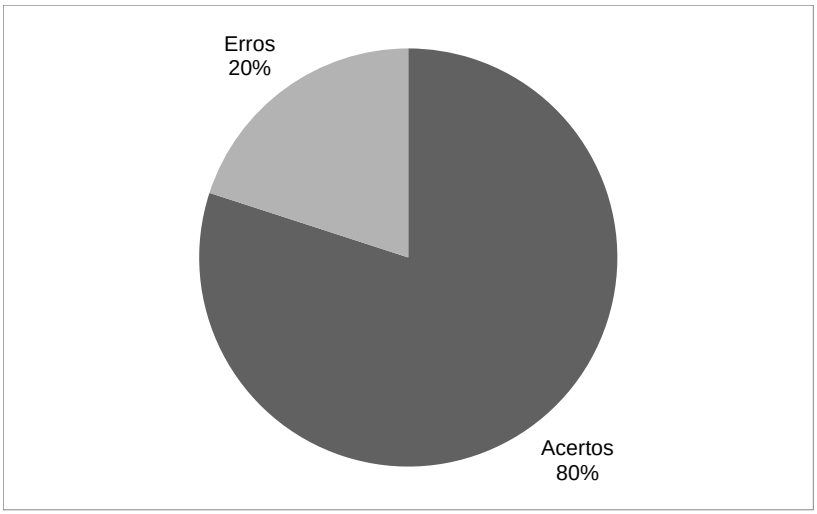
4. Não deve-se perguntar a opinião do cliente, deve-se descartar completamente o sistema e reconstruir o mesmo do zero;

5. Não é necessário substituir o sistema, pois se o mesmo é legado significa que a organização não precisa mais do mesmo;

6. Não é necessário realizar manutenção, deve-se apenas substituir o sistema por um novo.

A Figura 39 apresenta a porcentagem de acerto da resposta da questão. A única alternativa correta é a opção 3. 8 alunos responderam corretamente à questão, o que indica que houve um bom entendimento da forma que se avalia um sistema legado para decidir qual a estratégia de manutenção indicada para o mesmo.

Figura 39 – Acertos questão 1 dimensão 2



Questão 2 - Selecione a alternativa que descreve de forma INCORRETA a estratégia de manutenção de um sistema legado.

A segunda questão desta dimensão do questionário solicitava ao aluno que assinalasse a alternativa incorreta dentre as alternativas de estratégias de manutenção de um sistema legado disponibilizadas. As alternativas disponíveis eram as seguintes:

1. Deve-se DESCARTAR COMPLETAMENTE O SISTEMA quando se identificar que o mesmo é um sistema legado, e por ser legado significa que a organização não precisa mais do mesmo;

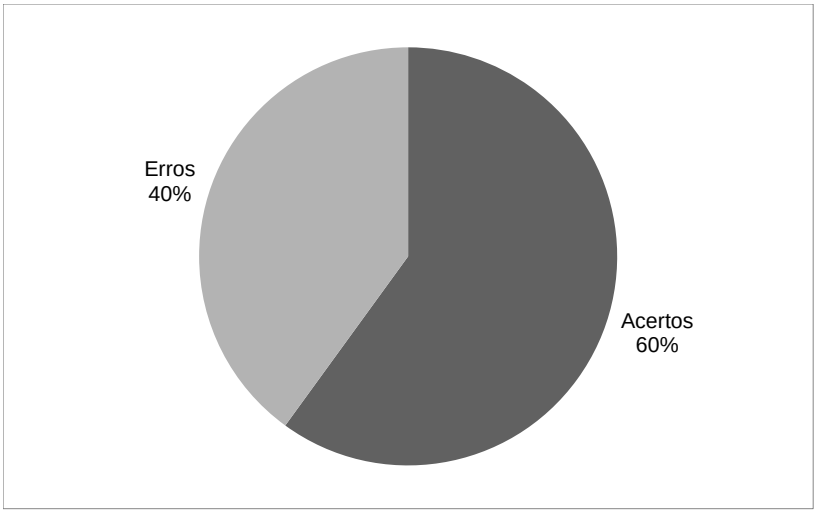
2. Deve-se DEIXAR O SISTEMA INALTERADO E CONTINUAR COM A MANUTENÇÃO REGULAR quando o sistema ainda é necessário, mas é bastante estável e os usuários do sistema fazem poucas solicitações de mudança;

3. Deve-se REESTRUTURAR O SISTEMA quando a qualidade do sistema foi degradada pelas mudanças, e novas mudanças para o novo sistema ainda estão sendo propostas;

4. Deve-se SUBSTITUIR O SISTEMA POR UM NOVO SISTEMA quando existir uma solução de prateleira adequada e a um custo razoável;

A Figura 40 apresenta a porcentagem de acerto da resposta da questão A única alternativa incorreta é a opção 1. 6 alunos responderam corretamente a questão, o que indica que em uma aplicação futura do processo as estratégias de manutenção devem ser melhor exploradas.

Figura 40 – Acertos questão 2 dimensão 2



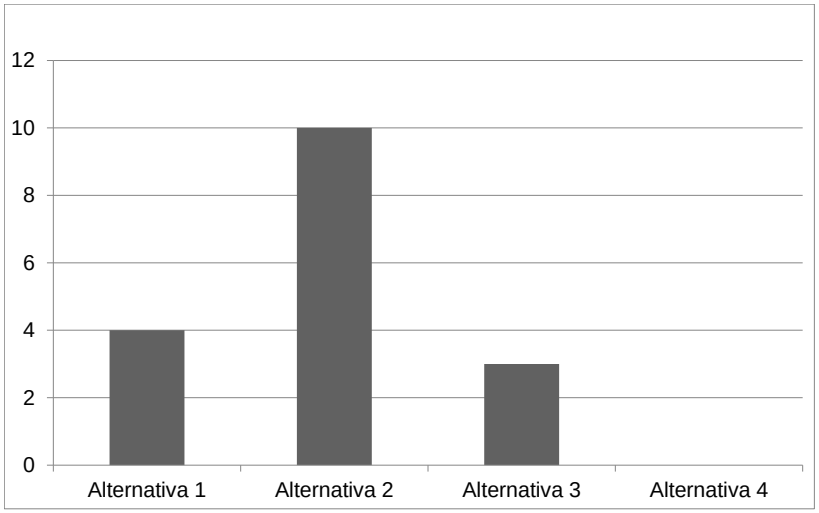
Questão 3 - Marque AS ALTERNATIVAS CORRETAS em relação à reengenharia

A terceira questão desta dimensão do questionário solicitava ao aluno que assinalasse as alternativas corretas em relação ao conceito de reengenharia. As alternativas disponíveis eram as seguintes:

1. A reengenharia de software é responsável por reimplementar sistemas legados, com o objetivo de facilitar sua manutenção;
2. A reengenharia pode envolver desde a elaboração de nova documentação, até a reorganização da estrutura do sistema e conversão da linguagem de programação;
3. Com a realização da reengenharia ocorre a redução de risco de desenvolvimento de um software e também custo reduzido;
4. Sempre há adição de novas funcionalidades ao se realizar a reengenharia de um sistema;

A Figura 41 apresenta o número de marcações que cada alternativa recebeu. Nenhum aluno assinalou as três alternativas no questionário, que era a resposta esperada para esta questão. 04 alunos assinalaram a alternativa 1 e 2, 03 alunos assinalaram a alternativa 2 e 3 e 03 alunos assinalaram apenas a alternativa 2. Apesar disso, nenhum aluno assinalou a alternativa 4, que era a única alternativa incorreta dentre as apresentadas na questão.

Figura 41 – Marcações questão 03 dimensão 2



Questão 4 - Considerando as seguintes afirmações referentes as atividades do processo de reengenharia relacionadas com o modelo ferradura, selecione as afirmações CORRETAS:

A quarta questão desta dimensão do questionário solicitava ao aluno que assinalasse as alternativas corretas que relacionavam as atividades do processo de reengenharia com o modelo ferradura. Eram disponibilizadas 06 opções, sendo que haviam 02 opções

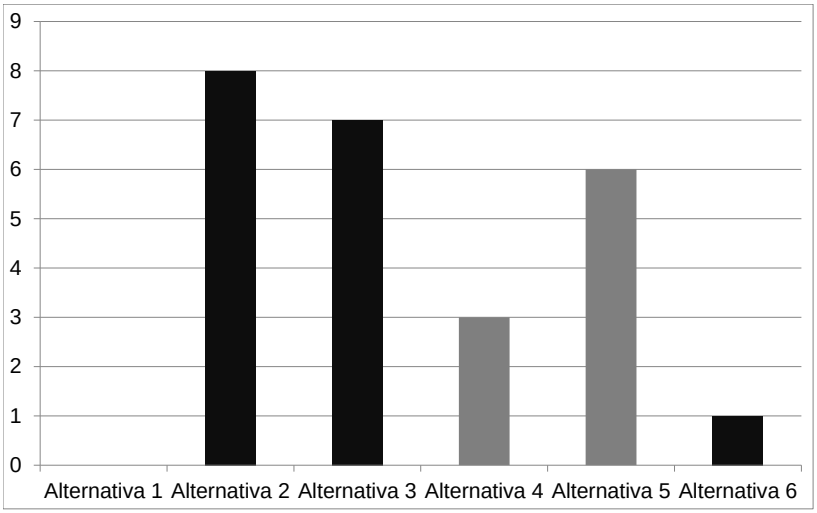
para cada fase do modelo ferradura. Os alunos foram orientados na descrição da questão a assinalarem uma opção de cada fase. As alternativas disponíveis eram as seguintes:

1. A fase de reconstrução é representada pela implementação do sistema de forma igual ao sistema anterior, sem modificar as tecnologias utilizadas no sistema legado;
2. A fase de reconstrução envolve a obtenção das informações lógicas do sistema, obtidas através de Engenharia reversa, gerando a representação da arquitetura do sistema legado;
3. A fase de transformação envolve a transformação e melhoria das descrições do sistema, envolvendo a refatoração da arquitetura do software, para a posterior implementação da mesma;
4. A fase de transformação envolve a implementação, teste e validação do sistema, a partir da nova arquitetura do sistema;
5. A fase de refinamento envolve a transformação e melhoria das descrições do sistema, envolvendo a refatoração da arquitetura do sistema, para a posterior implementação do mesmo;
6. A fase de refinamento envolve a implementação das mudanças do sistema, refletindo a nova arquitetura obtida anteriormente;

A Figura 42 apresenta o número de marcações que cada alternativa recebeu. Apenas um aluno respondeu corretamente a questão, assinalando a alternativa 2, a alternativa 3 e a alternativa 6 (representadas pela cor preta no gráfico). Alguns alunos não assinalaram as 03 opções, conforme foi orientado de forma clara na descrição da questão, o que indica que os mesmos tiveram dificuldades na interpretação da questão.

Apesar disso, a maioria das demais respostas assinalava a alternativa 2 e alternativa 3 de forma correta, e a alternativa 05 que é uma das alternativas incorretas, o que demonstra que os alunos obtiveram um entendimento parcial da relação entre as etapas do processo de reengenharia e o modelo ferradura, confundindo as etapas da reengenharia, sendo que esta é uma questão a ser melhor explorada no processo em aplicações futuras.

Figura 42 – Marcações questão 04 dimensão 02.



Questão 5 - Marque com verdadeiro ou falso as seguintes afirmações:

A quinta questão desta dimensão do questionário solicitava ao aluno que assinalasse com verdadeiro ou falso as afirmações relativas a reengenharia. Esta questão foi proposta no trabalho de Perez-Castillo et al. (2013) e foi adaptada para a aplicação no questionário. As afirmações apresentadas eram as seguintes:

1. A reengenharia pode ser utilizada para a manutenção de sistemas de informação;
2. Reengenharia é um tipo de engenharia aplicada através de ferramentas CASE;
3. Reengenharia pode obter versões melhoradas de sistemas de informação;
4. Reengenharia é somente a realização da Engenharia Reversa;
5. Reengenharia é somente a realização da Refatoração;
6. Reengenharia é somente a realização da Migração;

A Figura 43 e Figura 44 apresentam o número de opções verdadeiro e falso assinaladas para cada afirmação. Apenas a primeira e a terceira afirmação eram verdadeiras, as demais eram falsas. Em relação à primeira afirmação, obteve-se 70% de acerto. A segunda afirmação obteve 70% de acerto. Na terceira e na quarta afirmação houve acerto de 90% e na quinta e sexta afirmação obteve-se 100% de acerto.

Figura 43 – Afirmações relativas à reengenharia (1)

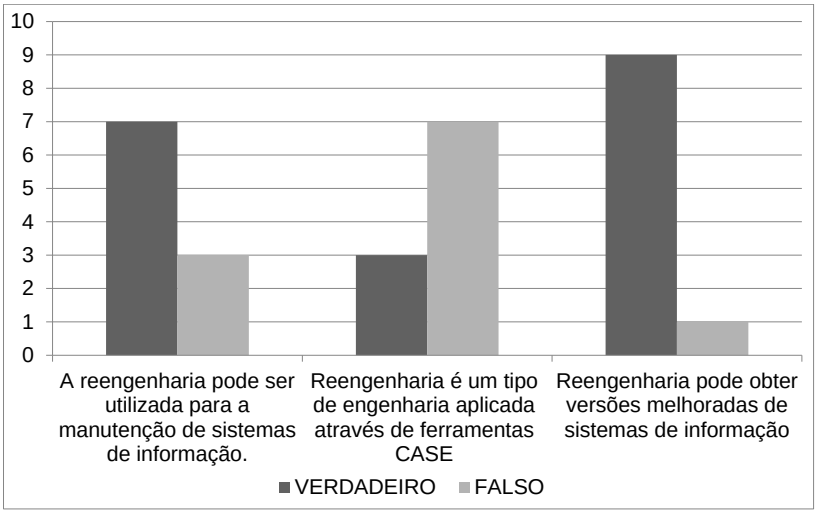
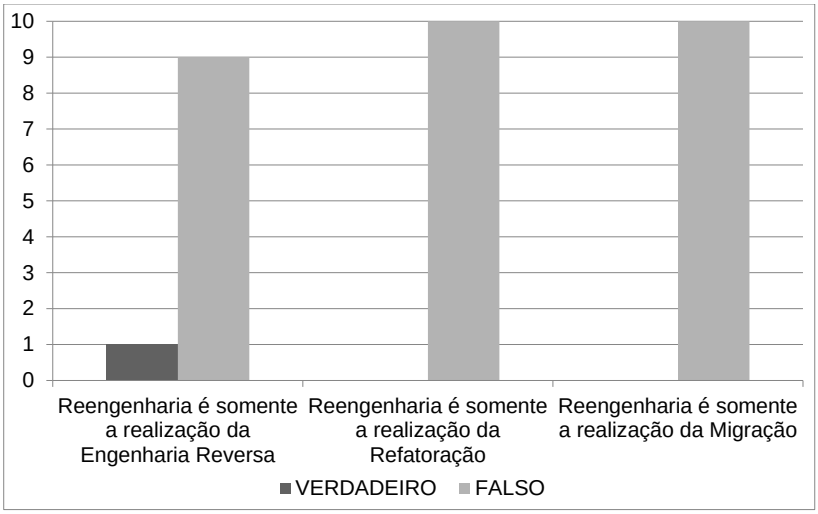


Figura 44 – Afirmações relativas à reengenharia (2)



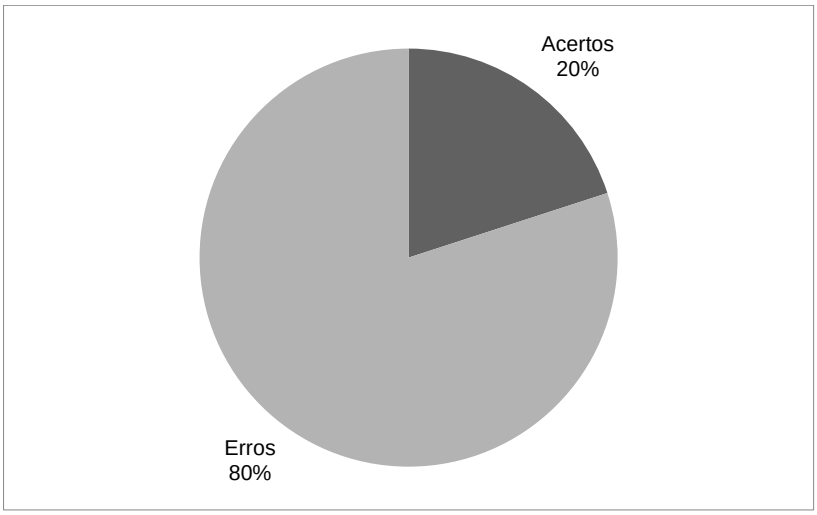
Questão 6 - Indique qual dos seguintes fatores são razão para a aplicação de reengenharia em um sistema de informação.

A sexta e última questão da segunda dimensão do questionário solicitava que o aluno indicasse apenas uma opção relativa a quais dos fatores eram razão para a aplicação de reengenharia em um sistema de informação. Esta questão foi proposta no trabalho de Perez-Castillo et al. (2013) e foi incorporada a este questionário. As opções disponíveis eram as seguintes:

1. Para adaptar o sistema às novas tecnologias, linguagens de programação , etc;
2. Para incorporar novas funcionalidades ou atender às novas exigências do sistema;

3. Para realizar manutenção no sistema (erros e falhas) sem realizar melhorias;
4. As opções A e B;
5. As opções A, B e C;

Figura 45 – Acertos questão 6 dimensão 2



A Figura 45 apresenta a porcentagem de acerto da resposta da questão A única alternativa correta é a opção 4. Apenas 2 alunos (20%) responderam corretamente a questão. Nesta questão houve um entendimento incorreto por parte dos alunos, pois a maioria destes assinalou a alternativa 5 (que considerava as três primeiras alternativas como sendo corretas), porém a alternativa 3 é incorreta, devido a manutenção de erros e falhas sem a realização de melhorias ser característica da estratégia de manutenção de um sistema e não da estratégia de reengenharia.

6 Considerações finais

A evolução de sistemas legados ainda é um desafio: No nível acadêmico é explorada superficialmente em cursos de computação, sendo muitas vezes apresentada como uma tarefa adicional e externa ao desenvolvimento do software. Além disso, a evolução de software é uma área onde ainda há muito a ser explorado devido as poucas pesquisas, especialmente em relação ao ensino de manutenção e reengenharia de sistemas legados (PEREZ-CASTILLO et al., 2013).

Este trabalho teve por objetivo principal a realização de um estudo dos processos de reengenharia existentes, e a partir destes, a elaboração e validação de um processo didático para o apoio às práticas de reengenharia na disciplina de Evolução de Software. Para a realização desta proposta, foi necessária a realização dos seguintes objetivos específicos:

- A realização de um estudo dos processos e técnicas de reengenharia existentes;
- A elaboração do processo de reengenharia baseado nos processos de reengenharia existentes;
- A validação do processo, através da utilização do mesmo para as práticas de reengenharia na disciplina de evolução de software;
- A avaliação da utilização do processo de reengenharia na disciplina.

O estudo dos processos existentes na literatura identificou que os mesmos não possuem um viés prático porque ilustram apenas as etapas a serem realizadas para a realização da reengenharia e não definem claramente os passos e as atividades necessárias para que as etapas do processo sejam realizadas.

As atividades e etapas presentes no processo de reengenharia de software proposto por este trabalho estão devidamente relacionadas com os processos de evolução de software existentes na literatura. O processo inclui como uma das atividades a avaliação do sistema legado sob a perspectiva técnica e de negócio antes da realização da reengenharia, com o objetivo de avaliar o a situação atual do sistema e definir qual a melhor estratégia para a manutenção do mesmo, que pode não ser a reengenharia.

O uso da abordagem incremental no processo possibilitou a realização do desenvolvimento de uma pequena funcionalidade do sistema, dentro do tempo possível para a aplicação do processo na disciplina. Adicionalmente, o processo proposto neste trabalho contempla o uso de algumas técnicas de engenharia de software que não estão presentes

em processos existentes como, por exemplo, a utilização de casos de uso para a representação das funcionalidades do sistema e o uso de diagramas da UML para a modelagem do software evoluído utilizando-se orientação a objetos.

A validação do processo na disciplina de evolução de software ocorreu conforme as expectativas, sendo que ao final das atividades dois dos três grupos de trabalho conseguiram concluir com sucesso as atividades. A avaliação realizada pelos discentes evidenciou os benefícios do uso da prática para o ensino de reengenharia e fornece *feedback* para melhorias futuras no processo.

6.1 Trabalhos futuros

Ao término do presente trabalho, identificou-se algumas questões que podem ser pesquisadas e desenvolvidas em trabalhos futuros. A primeira delas envolve analisar os resultados obtidos pelas respostas presentes no questionário de avaliação da aplicação do processo, respondido pelos alunos, e realizar melhorias no processo. A análise dos resultados evidenciou algumas questões que podem ser melhoradas no processo, como por exemplo a dependência entre os artefatos na etapa de análise do sistema legado e de análise e projeto do novo sistema. Também em relação à estas duas etapas, pode-se estudar a possibilidade da realização do projeto do novo sistema de forma incremental, o que pode gerar um melhor aproveitamento do tempo disponível para a aplicação do processo.

Outro passo importante para a sequência deste trabalho é a realização de nova aplicação do processo de reengenharia, seja na disciplina de evolução de software ou em outra disciplina, utilizando-se o mesmo sistema legado como objeto de realização de atividades práticas. Através desta nova aplicação do processo será possível a realização de avaliação do desempenho e o comparativo com a aplicação do processo realizado neste trabalho.

Adicionalmente, também é relevante a realização de aplicação do processo de reengenharia com outro sistema legado como objeto de realização das atividades práticas, para avaliar o comportamento do processo em outro contexto de trabalho.

Pesquisas futuras também podem avaliar a viabilidade da utilização deste processo para a realização de reengenharia de sistemas legados por empresas de desenvolvimento de software, pois não há restrições no processo que não permitam o uso da proposta em um ambiente profissional.

Referências

BERNHART, M. et al. Incremental reengineering and migration of a 40 year old airport operations system. *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, Ieee, p. 503–510, set. 2012. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6405313>. Citado 3 vezes nas páginas 42, 43 e 52.

BEZERRA, E. *Princípios de análise e projeto de sistemas com UML*. 2. ed. Rio de Janeiro: Elsevier, 2007. ISBN 85-352-1696-0. Citado 3 vezes nas páginas 21, 66 e 76.

BREIVOLD, H. P.; CHAUHAN, M. A.; BABAR, M. A. A Systematic Review of Studies of Open Source Software Evolution. *2010 Asia Pacific Software Engineering Conference*, Ieee, p. 356–365, nov. 2010. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5693212>. Citado na página 41.

CERA, M. C.; Dal Forno, M. H.; Gindri Vieira, V. Uma Proposta para o Ensino de Engenharia de Software a partir da Resolução de Problemas. *Revista Brasileira de Informática na Educação*, v. 20, n. 03, p. 116–129, dez. 2012. ISSN 14145685. Disponível em: <http://www.br-ie.org/pub/index.php/rbie/article/view/1391>. Citado na página 23.

Dal Forno, M. H. et al. Aprendizagem baseada em Problemas aplicada a Engenharia de Software: Suporte a Disciplina de Resolução de Problemas I. *International Conference PBL 2012*, 2012. Citado na página 23.

De Lucia, A.; FASOLINO, A. R.; POMPELLA, E. A Decisional Framework for Legacy System Management Faculty of Engineering - University of Sannio. *Proceedings International Conference on Software Maintenance*, p. 642 – 651, 2001. Citado na página 29.

IEEE Computer Society; ACM. *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. 2004. Citado na página 22.

JUNIOR, S. L. D.; SILVA, J. C. A. Processes of Software Reengineering Planning Supported by Usability Principles. *Proceedings of the Latin American conference on Human-computer interaction*, p. 223–226, 2003. Disponível em: <http://dl.acm.org/citation.cfm?id=944544>. Citado 3 vezes nas páginas 42, 43 e 69.

KAGDI, H.; GETHERS, M.; POSHYVANYK, D. SE^2 Model to Support Software Evolution. *2011 27th IEEE International Conference on Software Maintenance (ICSM)*, Ieee, p. 512–515, set. 2011. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6080820>. Citado na página 41.

MALIK, H.; HASSAN, A. E. Supporting Software Evolution Using Adaptive Change Propagation Heuristics. *2008 IEEE International Conference on Software Maintenance*, Ieee, p. 177–186, set. 2008. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4658066>. Citado na página 42.

MELO, A. C. *Desenvolvendo aplicações com UML 2.2: do conceitual à implementação*. 3ª. ed. Rio de Janeiro: Brasport, 2010. ISBN 978-85-7452-444-3. Citado na página 52.

Paula Filho, W. d. P. *Engenharia de software: fundamentos, métodos e padrões*. Rio de Janeiro: LTC, 2011. ISBN 978-85-216-1650-4. Citado na página 26.

PEREZ-CASTILLO, R. et al. A Teaching Experience on Software Reengineering. In: *2013 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2013. p. 1284–1293. ISBN 978-1-4673-6110-1. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6530272>. Citado 9 vezes nas páginas 15, 22, 45, 46, 59, 85, 98, 99 e 101.

PETRENKO, M. et al. Teaching Software Evolution in Open Source. *Computer*, v. 40, n. 11, p. 25–31, nov. 2007. ISSN 0018-9162. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4385252http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4385252>. Citado 6 vezes nas páginas 15, 45, 47, 48, 52 e 85.

PFLEEGER, S. L. *Engenharia de Software: teoria e prática*. 2. ed. São Paulo: Prentice Hall, 2004. ISBN 978-85-87918-31-4. Citado 15 vezes nas páginas 15, 25, 26, 27, 28, 29, 30, 31, 35, 36, 37, 38, 54, 59 e 61.

PRESSMAN, R. S. *Engenharia de Software*. 6. ed. São Paulo: MCGRAW-Hill, 2006. ISBN 85-86804-57-6. Citado na página 65.

PRESSMAN, R. S. *Engenharia de software: uma abordagem profissional*. 7. ed. Porto Alegre: AMGH, 2011. ISBN 978-85-63308-33-7. Citado 10 vezes nas páginas 15, 21, 25, 26, 27, 28, 29, 38, 39 e 62.

REZENDE, D. A. *Engenharia de software e sistemas de informação*. 3. ed. Rio de Janeiro: Brasport, 2005. ISBN 85-7452-215-5. Citado 2 vezes nas páginas 25 e 26.

SEACORD, R. C.; PLAKOSH, D.; LEWIS, G. A. *Modernizing Legacy Systems: Software Technologies, Engineering Process and Business Practices*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 0321118847. Citado 3 vezes nas páginas 40, 41 e 59.

SOMMERVILLE, I. *Engenharia de Software*. 8. ed. São Paulo: Pearson Addison-Wesley, 2007. ISBN 978-85-88639-28-7. Citado 6 vezes nas páginas 15, 21, 33, 34, 59 e 65.

SOMMERVILLE, I. *Engenharia de Software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011. ISBN 978-85-7936-108-1. Citado 11 vezes nas páginas 22, 25, 26, 27, 28, 29, 30, 31, 54, 55 e 56.

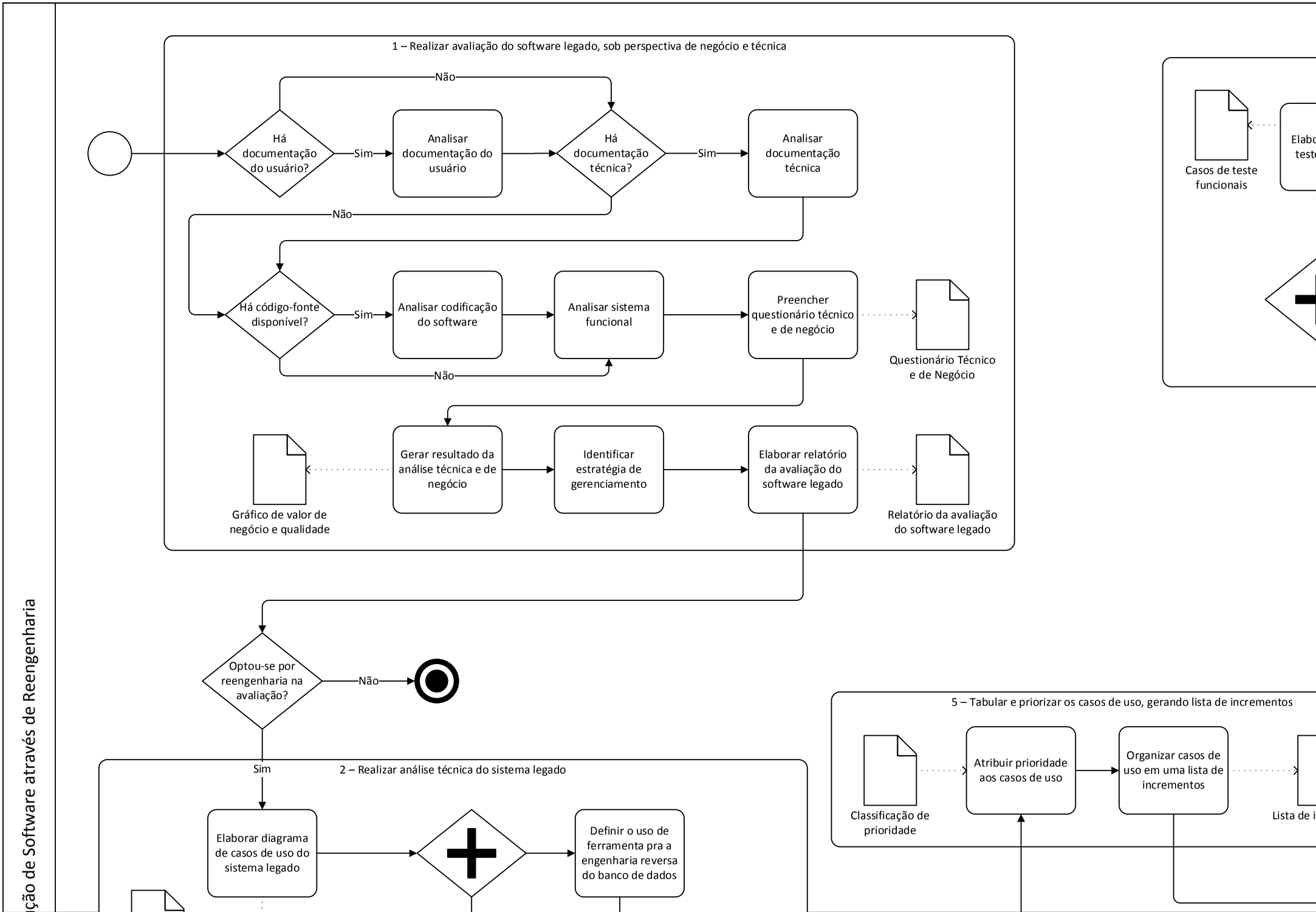
THUMS, A.; QUANTE, J. Reengineering embedded automotive software. *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, p. 493–502, 2012. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6405312>. Citado 4 vezes nas páginas 15, 42, 44 e 59.

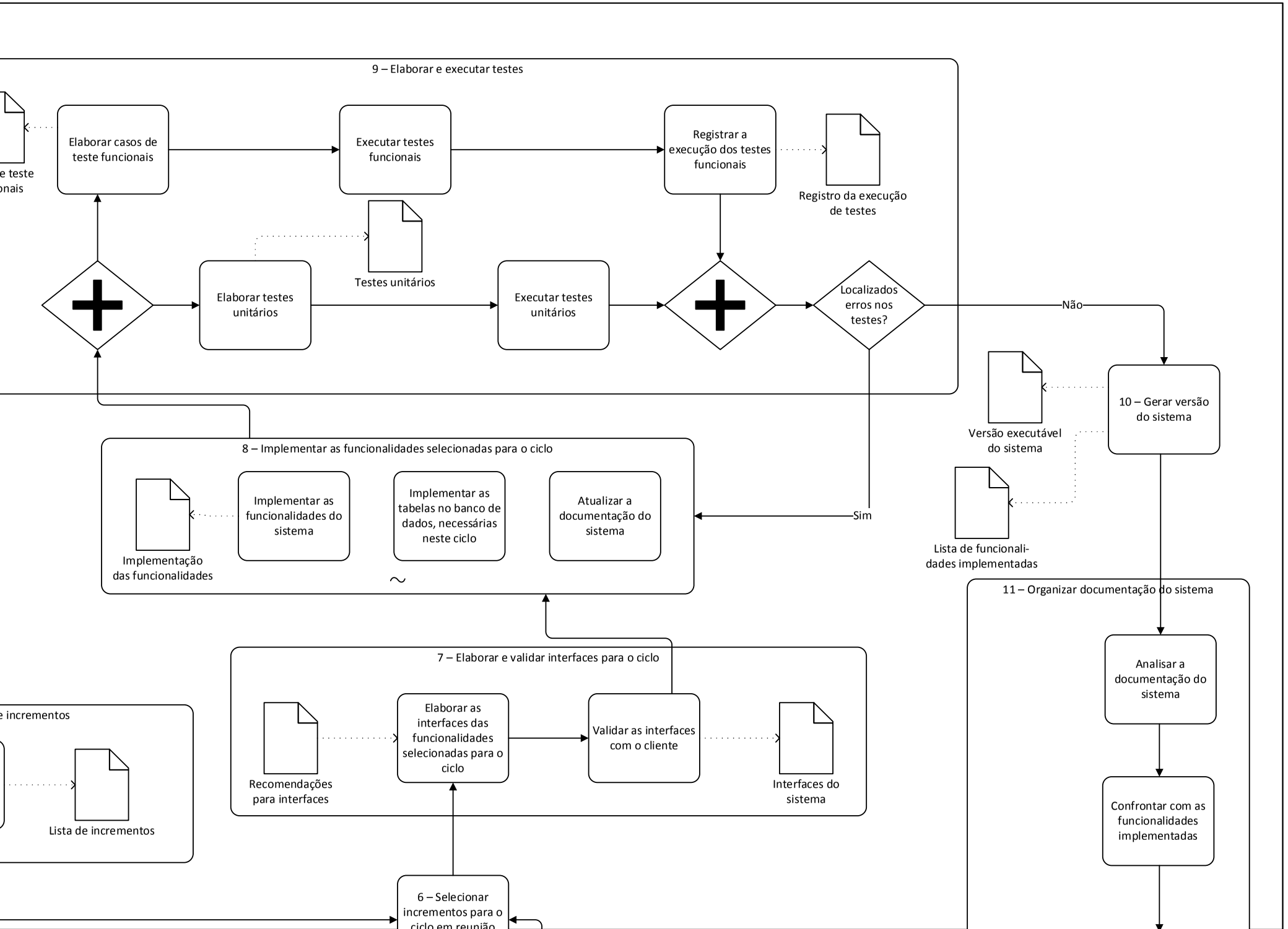
Universidade Federal do Pampa. *Projeto Pedagógico de Curso Engenharia de Software*. 2012. Citado 2 vezes nas páginas 22 e 77.

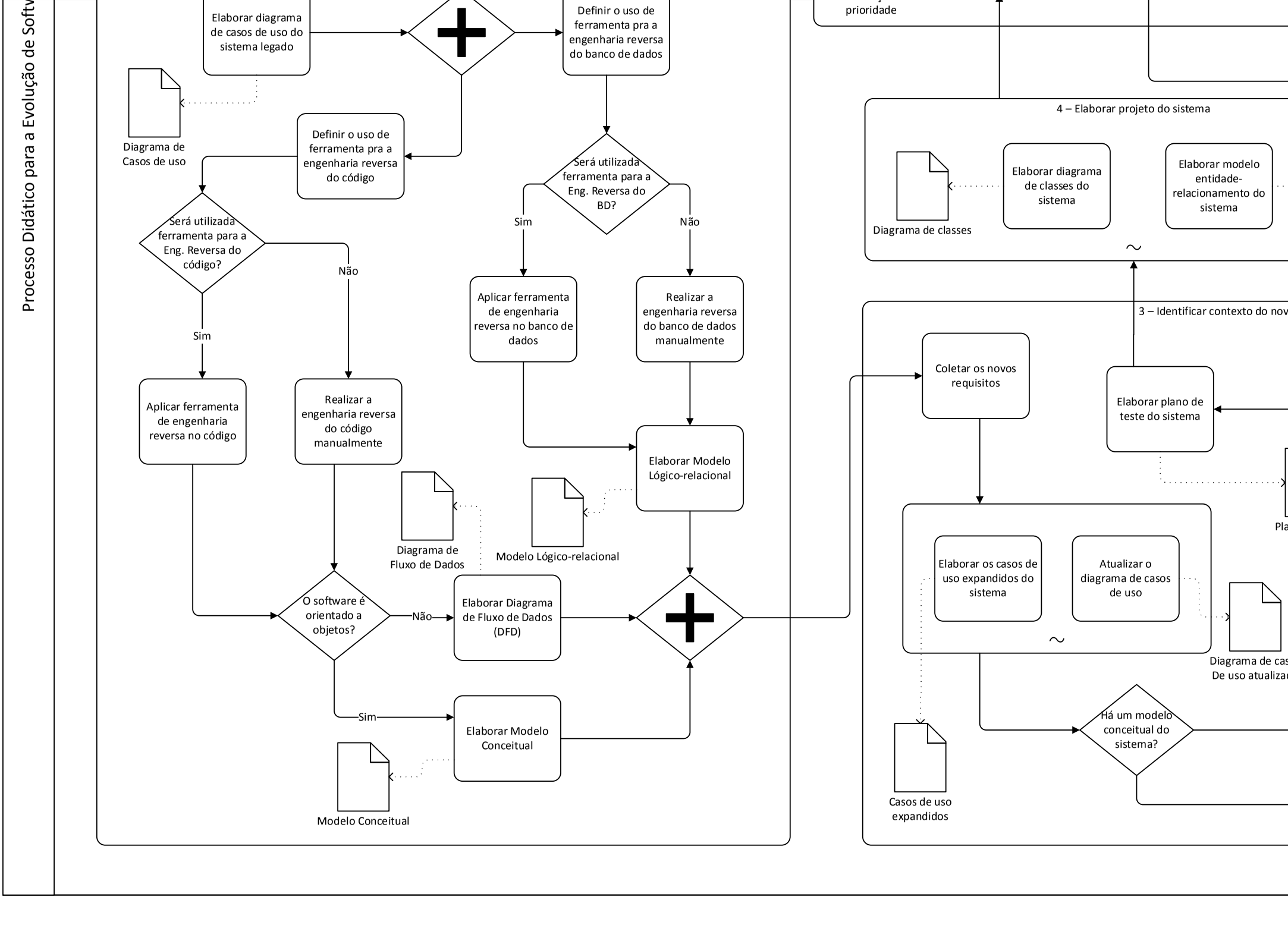
WAZLAWICK, R. S. *Análise e projeto de sistemas de informação orientados a objetos*. 2. ed. Rio de Janeiro: Elsevier, 2011. ISBN 978-85-352-3916-4. Citado na página 62.

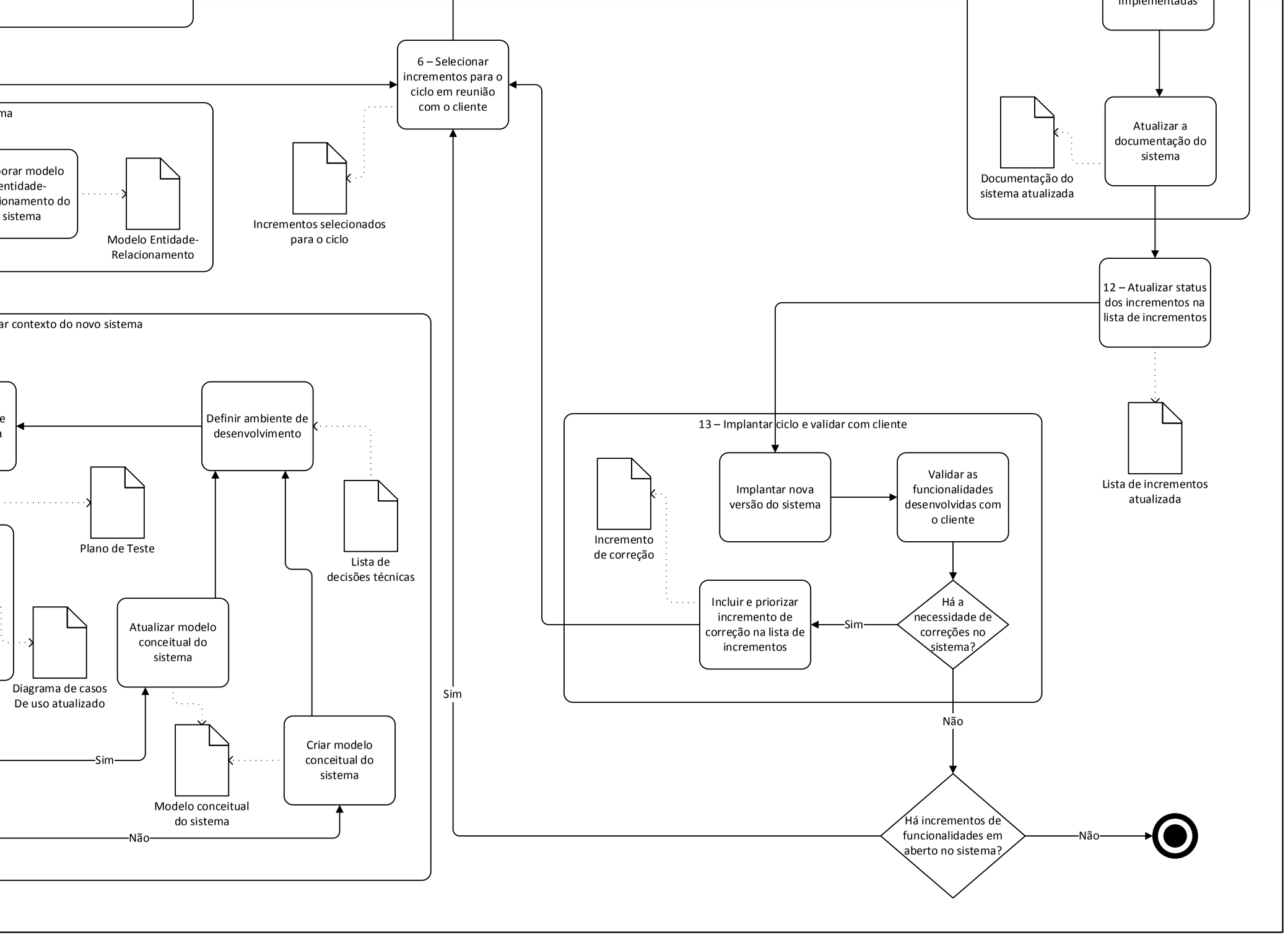
Apêndices

APÊNDICE A – Processo em BPMN com os subprocessos expandidos


ção de Software através de Reengenharia
1 – Realizar avaliação do software legado, sob perspectiva de negócio e técnica
Não
Há documentação do usuário?
Sim
Analisar documentação do usuário
Há documentação técnica?
Sim
Analisar documentação técnica
Não
Há código-fonte disponível?
Sim
Analisar codificação do software
Analisar sistema funcional
Não
Preencher questionário técnico e de negócio
Questionário Técnico e de Negócio
Gráfico de valor de negócio e qualidade
Gerar resultado da análise técnica e de negócio
Identificar estratégia de gerenciamento
Elaborar relatório da avaliação do software legado
Relatório da avaliação do software legado
Casos de teste funcionais
Optou-se por reengenharia na avaliação?
Não
Sim
2 – Realizar análise técnica do sistema legado
Elaborar diagrama de casos de uso do sistema legado
Definir o uso de ferramenta pra a engenharia reversa do banco de dados
5 – Tabular e priorizar os casos de uso, gerando lista de incrementos
Classificação de prioridade
Atribuir prioridade aos casos de uso
Organizar casos de uso em uma lista de incrementos


9 – Elaborar e executar testes
e teste
onais
Elaborar casos de teste funcionais
Executar testes funcionais
Registrar a execução dos testes funcionais
Registro da execução de testes
Testes unitários
Elaborar testes unitários
Executar testes unitários
Localizados erros nos testes?
Não
Sim
10 – Gerar versão do sistema
Versão executável do sistema
Lista de funcionali-dades implementadas
8 – Implementar as funcionalidades selecionadas para o ciclo
Implementação das funcionalidades
Implementar as funcionalidades do sistema
Implementar as tabelas no banco de dados, necessárias neste ciclo
Atualizar a documentação do sistema
11 – Organizar documentação do sistema
Analisar a documentação do sistema
Confrontar com as funcionalidades implementadas
7 – Elaborar e validar interfaces para o ciclo
Recomendações para interfaces
Elaborar as interfaces das funcionalidades selecionadas para o ciclo
Validar as interfaces com o cliente
Interfaces do sistema
e incrementos
Lista de incrementos
6 – Selecionar incrementos para o ciclo em reunião


Processo Didático para a Evolução de Softw
Elaborar diagrama de casos de uso do sistema legado
Definir o uso de ferramenta pra a engenharia reversa do banco de dados
Diagrama de Casos de uso
Definir o uso de ferramenta pra a engenharia reversa do código
Será utilizada ferramenta para a Eng. Reversa do BD?
Sim
Não
Será utilizada ferramenta para a Eng. Reversa do código?
Não
Sim
Aplicar ferramenta de engenharia reversa no banco de dados
Realizar a engenharia reversa do banco de dados manualmente
Aplicar ferramenta de engenharia reversa no código
Realizar a engenharia reversa do código manualmente
Elaborar Modelo Lógico-relacional
Diagrama de Fluxo de Dados
Modelo Lógico-relacional
O software é orientado a objetos?
Não
Elaborar Diagrama de Fluxo de Dados (DFD)
Sim
Elaborar Modelo Conceitual
Modelo Conceitual
prioridade
4 – Elaborar projeto do sistema
Diagrama de classes
Elaborar diagrama de classes do sistema
Elaborar modelo entidade-relacionamento do sistema
3 – Identificar contexto do nov
Coletar os novos requisitos
Elaborar plano de teste do sistema
Pla
Elaborar os casos de uso expandidos do sistema
Atualizar o diagrama de casos de uso
Diagrama de cas
De uso atualiza
Há um modelo conceitual do sistema?
Casos de uso expandidos


6 – Selecionar incrementos para o ciclo em reunião com o cliente
Incrementos selecionados para o ciclo
Modelo Entidade-Relacionamento
Atualizar a documentação do sistema
Documentação do sistema atualizada
12 – Atualizar status dos incrementos na lista de incrementos
Lista de incrementos atualizada
13 – Implantar ciclo e validar com cliente
Implantar nova versão do sistema
Validar as funcionalidades desenvolvidas com o cliente
Incremento de correção
Incluir e priorizar incremento de correção na lista de incrementos
Há a necessidade de correções no sistema?
Sim
Não
Há incrementos de funcionalidades em aberto no sistema?
Não
Sim
Definir ambiente de desenvolvimento
Plano de Teste
Lista de decisões técnicas
Atualizar modelo conceitual do sistema
Diagrama de casos De uso atualizado
Criar modelo conceitual do sistema
Modelo conceitual do sistema
Sim
Não

APÊNDICE B – Avaliação do Sistema de Gerenciamento de Concursos Públicos (GCP), sob a perspectiva de negócio e técnica

Valor de Negócio

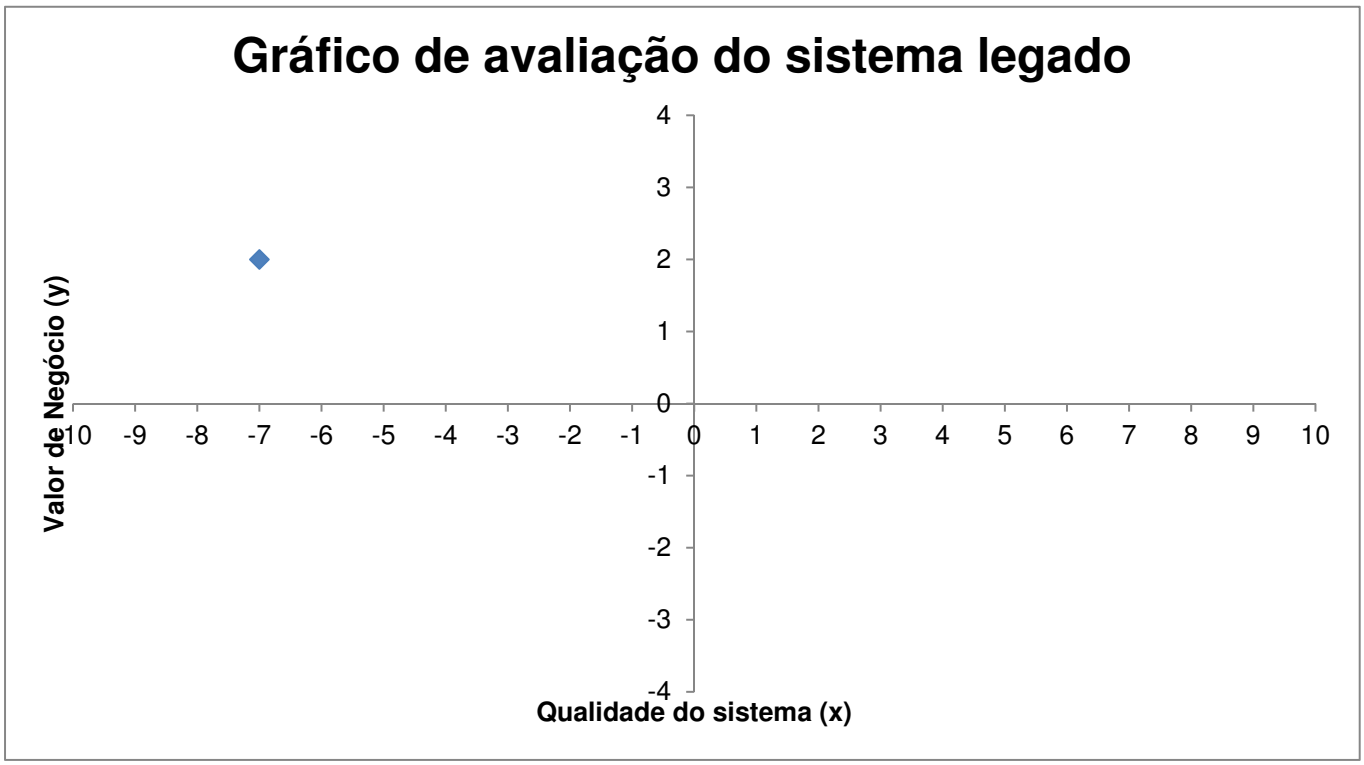
Nº	Questão	Avaliador	Resposta
1	Considerando o contexto onde o software está inserido, o sistema é usado com frequência?	Usuário	1
2	Os processos de negócio do sistema estão atualizados?	Gerente	-1
3	O sistema é confiável (possui pouquíssimos defeitos ou falhas)?	Usuário, gerente	1
4	As saídas do sistema são utilizadas? Há importância para o usuário?	Usuário, gerente	1
	SOMATÓRIO DE VALOR DE NEGÓCIO		2

Qualidade do sistema

Nº	Questão	Avaliador	Resposta
1	O fornecedor do sistema ainda existe e dá suporte para o mesmo?	Gerente	1
2	O sistema é compatível com o hardware e sistema operacional mais recentes?	Área técnica	1
3	O sistema foi submetido a várias modificações durante o seu uso?	Área técnica, gerente	1
4	O desempenho do sistema é adequado?	Usuário, gerente	1
5	O software opera independentemente de manutenção de hardware legado?	Área técnica	1
6	O software opera independentemente de manutenção de softwares de apoio pagos?	Área técnica	-1
7	O sistema possui custo baixo de manutenção?	Área técnica	1
8	O sistema é compatível com diversos sistemas operacionais (windows, linux, etc.)?	Área técnica	-1
9	O código-fonte do sistema legado é de fácil compreensão?	Área técnica	-1
10	Há documentação técnica do software legado disponível?	Área técnica	-1
11	Os dados são armazenados em um banco de dados?	Área técnica	-1
12	O banco de dados está normalizado?	Área técnica	-1
13	É adotado um padrão para nomenclatura das tabelas e campos do Banco de Dados?	Área técnica	-1
14	São realizadas validações nos campos (Ex.: Datas, CPF, e-mail, etc.)?	Área técnica, usuário	-1
15	O sistema exige que campos obrigatórios sejam preenchidos?	Área técnica, usuário	-1
16	A linguagem de programação do sistema legado ainda é utilizada no mercado de trabalho?	Área técnica	1
17	Existem registros de execução de testes do sistema?	Área técnica	-1
18	A interface do sistema é implementada em modo gráfico?	Área técnica	1
19	Existem profissionais com habilidades necessárias para o suporte da aplicação?	Área técnica	-1
20	O sistema redimensiona as informações ao maximizar a tela?	Área técnica	-1
21	Ao solicitar o armazenamento de dados de um formulário, o usuário é sempre informado se os mesmos foram gravados com sucesso ou se ocorreu uma falha?	Área técnica, usuário	-1
22	O sistema possui um padrão de interfaces? Esse padrão é sempre seguido?	Área técnica	-1
23	O sistema permite a navegação através do teclado?	Área técnica	-1
	SOMATÓRIO DE QUALIDADE DO SISTEMA		-7

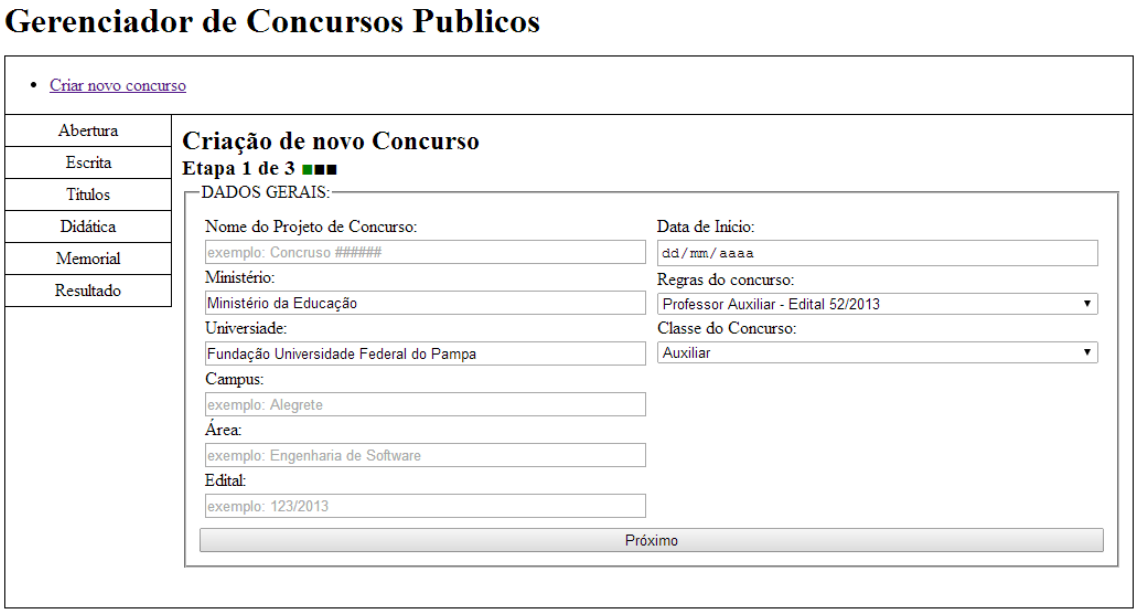
LEGENDA PARA AS RESPOSTAS

Sim	1
Não	-1



APÊNDICE C – Funcionalidades produzidas no primeiro ciclo incremental do processo

Funcionalidade implementada pelo grupo 1



Funcionalidade implementada pelo grupo 3