MAÍNNE TARCILA PROFETA E SILVA

PROCESSOS DE RECUPERAÇÃO DE REQUISITOS DE SISTEMAS LEGADOS

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para conclusão do curso de MBA em Tecnologia de Software.

São Paulo
2011

MAÍNNE TARCILA PROFETA E SILVA

PROCESSOS DE RECUPERAÇÃO DE REQUISITOS DE SISTEMAS LEGADOS

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador: Prof. Dr. Kechi Hirama

São Paulo
2011

FICHA CATALOGRÁFICA

Silva, Maínne Tarcila Profeta e
Processos de recuperação de requisitos de sistemas legados / M.T,P. e Silva. – São Paulo, 2011
69p.

Monografia (MBA em Tecnologia de Software) – Escola Politécnica da Universidade de São Paulo. Programa de Educação Continuada em Engenharia.

1. Manutenção de software 2. Reengenharia de software I. Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II.t.

DEDICATÓRIA

Dedico este trabalho aos meus pais
que me incentivam a cada passo.

AGRADECIMENTOS

À minha família pela paciência e apoio.

Ao meu orientador Prof. Dr. Kechi Hirama pela atenção, incentivo e diretrizes essenciais para a elaboração deste trabalho.

À minha amiga Juliana por ter me incentivado e escutado todas as minhas reclamações.

RESUMO

A longo prazo quase todos os aspectos de um negócio são passíveis a mudanças que, impreterivelmente, refletem em novas demandas para o ambiente tecnológico. Surge, então, a necessidade de manutenções a fim de permitir que os sistemas vigentes atendam às novas exigências do mercado.

Independentemente do tipo de manutenção adotado, é primordial um conhecimento amplo do funcionamento do sistema em produção, englobando a funcionalidade existente, quais requisitos devem ser alterados e quais serão impactados com a adequação à nova regra a ser implementada.

Diante deste cenário destaca-se a compreensão do software como a principal atividade do processo de manutenção. Ela representa mais de 50% dos esforços (CANFORA e CIMITILE, 2000) e mobiliza a comunidade acadêmica para criação de processos que viabilizem a recuperação das funções dos sistemas legados com a finalidade de facilitar o entendimento dos mesmos.

Este trabalho apresenta alguns processos de recuperação de requisitos encontrados na literatura que não fazem uso de algum artefato além do código executável. O trabalho apresenta também a aplicação de dois processos estudados em um sistema legado com o intuito de levantar os seus requisitos para melhorar a qualidade das manutenções de software.

ABSTRACT

For the long term, most of the business aspects are likely to change. In order to enable existing systems to achieve new market requirements. Unavoidably, these changes results in new demands in a technological environment arising the need for system maintenance.

Regardless of maintenance type, to be able to accomplish the requirement, it is essential to have a broad knowledge of the system that is running in production including existing processes, which requirements must be changed, and what will be impacted.

In this scenario, the comprehension of the application stands as the main activity of the maintenance process. It represents over 50% of efforts (CANFORA e CIMITILE, 2000), mobilizing the academic community to create processes that allow functions legacy systems to be recovered in order to support the understanding.

This paper presents some recovery requirement processes found in the literature that do not use any device beyond the executable code. The paper also presents the application of two processes studied in a legacy system in order to raise their requirements for improving the quality of software maintenance.

LISTA DE ILUSTRAÇÕES

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

AMBOLS – Analysing and Modelling the Bahaviour of Legacy Systems

CelLEST – CEL Legacy Enhancement Software Technologies

LeNDI – Legacy Navigation Domain Identifier

IPM – Interaction Pattern Mining

SGA – Sistema de gerenciamento de ativos

EPS – Empresa prestadora de serviço

SUMÁRIO

Pág.

1. INTRODUÇÃO

Neste capítulo é apresentada uma breve introdução do trabalho proposto, assim como as motivações, o objetivo, as justificativas e a estrutura do trabalho.

1.1 Motivações

Com o passar dos anos o mercado tornou-se cada vez mais competitivo exigindo das corporações expansões de seus negócios. A fim de acompanhar tal evolução, as empresas se adaptaram às novas regras e seus sistemas que suportam essas operações não ficaram para trás.

Este é somente um dos cenários comuns que exige a manutenção de software. Seja com foco em melhorias, adaptações ou em correções, esta atividade ganha destaque no dia-a-dia das empresas e, segundo Canfora e Cimitile (2000), é onde se concentra mais de 70% dos esforços, considerando que desta fatia a maior parte, cerca de 80%, são gastos em ações não-corretivas.

Assim como no processo de desenvolvimento, o ponto de partida para a manutenção é o entendimento das funções do sistema, porém para esta última se faz necessária a compreensão não só das novas funções a serem implementadas, mas também daquelas que ditam as regras de negócio do sistema em operação, uma vez que são cruciais para que o engenheiro de software identifique onde realizar a manutenção no sistema e contribuem de forma substancial para o planejamento dos testes e análise de impacto.

O entendimento dessas funções não é uma tarefa trivial na manutenção. Quando se diz respeito aos sistemas legados, geralmente não se pode contar com documentação consistente e confiável e/ou com os principais envolvidos no desenvolvimento. Dessa forma surgem diversas dificuldades de entendimento do software e de suas principais funções comprometendo significativamente a qualidade da manutenção.

Neste contexto, o uso de processos que visam a recuperação de requisitos de um sistema legado pode contribuir de forma positiva, visto que tornam viável o entendimento do sistema em período menor e com menos esforço, atividade esta

que consume de 50% a 90% do tempo da manutenção (CANFORA e CIMITILE, 2000).

1.2 Objetivo

O objetivo deste trabalho é apresentar processos de recuperação de requisitos de sistemas legados e aplicar dois desses processos em um sistema legado real a fim de identificar o grau de aderência dos requisitos recuperados à sua funcionalidade.

1.3 Justificativas

Os esforços associados à manutenção destacam a importância da adoção de processos que tornem viável a execução desta atividade de forma mais produtiva. Nesta linha, um caminho é o investimento no melhor entendimento das funções do software, atividade essa que pode ser auxiliada com a aplicação de processos que identifiquem as funções do sistema em produção.

Trabalhos encontrados na literatura, por exemplo, Liu et al. (1998), Cohen (1994), Ward e Bennett (2003), El-Ramly, Stroulia e Sorenson (2001) entre outros, introduzem algum processo específico com o intuito de mostrar como este pode ser útil na realização da manutenção. O presente trabalho busca a partir destas publicações, contribuir com a análise da aplicabilidade de alguns processos de recuperação de requisitos.

A escolha por processos que usam a análise comportamental para foco deste trabalho justifica-se pelos mesmos terem como recursos somente o executável e o usuário do sistema, facilitando a aplicação em uma gama maior de sistemas legados.

1.4 Estrutura do Trabalho

Capítulo 01 - Introdução

Neste capítulo são apresentadas as motivações, o objetivo, as justificativas e a estrutura do trabalho.

Capítulo 02 – Manutenção de Software

No segundo capítulo são apresentados conceitos importantes na manutenção e a relação das principais dificuldades enfrentadas neste processo, além de identificar como a recuperação de requisitos de forma consistente ajuda no processo como um todo.

Capítulo 03 – Processos de Recuperação de Requisitos

Neste capítulo são abordados alguns processos de recuperação de requisitos, como eles funcionam e o porquê estão sendo estudados.

Capítulo 04 – Aplicação dos Processos

Este capítulo relata a aplicação dos processos AMBOLS e CelLEST e quais os resultados obtidos.

Capítulo 05 – Considerações Finais

O último capítulo apresenta as contribuições deste trabalho e tece considerações finais e trabalhos futuros.

Referências

Por fim são listadas as referências utilizadas neste trabalho.

2. MANUTENÇÃO DE SOFTWARE

Este capítulo apresenta uma visão geral dos conceitos básicos relacionados à Manutenção de Software.

2.1 Sistemas Legados

Muito mais que a definição “Um software legado é um software que foi escrito ontem” (SEACORD, PLAKOSH e LEWIS, 2003), um sistema legado caracteriza-se na maioria das vezes, além de antigo, como um sistema que fornece serviços essenciais para a organização (CANFORA e CIMITILE, 2000). Este é um dos principais motivos pelo qual se justifica tantas corporações, mesmo diante da evolução crescente do mundo tecnológico, manter suas principais operações suportadas por sistemas legados.

Conforme já descrito por Lehman (1985) em sua primeira lei formulada ainda na década 80 – “Um sistema de informação que é usado deve ser continuamente adaptado, caso contrário se torna progressivamente menos satisfatório” – o investimento em melhorias nesses sistemas é contínuo, porém, como evidenciado na Figura 1 abaixo, não se mostra eficaz indefinidamente.

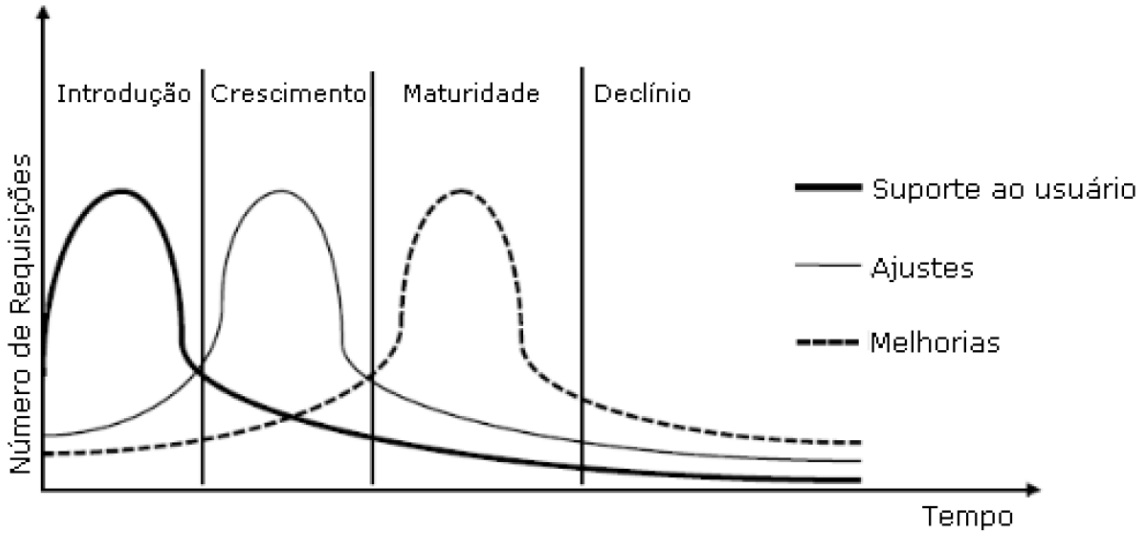

Figura 1. Ciclo de vida de manutenção de software (PADUELLI, 2007)

O ciclo de vida de manutenção de software ilustrado comprova que a primeira lei de Lehman (1985) não se aplica durante toda a vida do software. Segundo

descrito em Paduelli (2007), o software em produção comporta-se de tal forma que em determinado momento atinge a fase de maturidade, ou seja, adquire estabilidade podendo, então, ser classificado como um sistema legado, porém, conforme as mudanças externas ocorrem, ele deixa de atender às necessidades dos usuários passando à fase de declínio.

Durante este ciclo, os tipos de mudanças solicitadas variam de acordo com a fase na qual o sistema se encontra. No estágio inicial, as demandas caracterizam-se por atender o suporte aos usuários, à medida que o sistema cresce, são feitos ajustes que atendam aos novos critérios que surgem, já na fase de maturidade iniciam-se as melhorias para manter o software em produção, até que em certo ponto a complexidade gerada pelas mudanças faz com que o sistema entre na fase de declínio exigindo reestruturação ou substituição.

Tomar a decisão de manter um software ou substituí-lo não é uma tarefa simples, faz parte do dia a dia das empresas definir o custo benefício dessa manutenibilidade, e em sua maioria, o investimento e a rentabilidade que esses sistemas representam destacam-se diante das necessidades de melhorias e mudanças que ocasionariam uma substituição.

Nesse contexto, é cada vez mais comum que a atividade de manutenção se destaque, porque mesmo atendendo às funções vitais da empresa e funcionando de maneira adequada é inquestionável a deterioração do software, no sentido de os objetivos de suas funções cada vez menos se adequarem ao ambiente externo que está em constante evolução (PADUELLI, 2007).

2.2 Manutenção de Software

A manutenção presente no ciclo de vida do software pode ter diversas finalidades, o que se comprova a partir da definição dada pela norma IEEE-1219 (1998) que diz que manutenção representa “qualquer modificação de um produto de software já entregue ao cliente, para a correção de eventuais erros, melhora em seu desempenho, ou qualquer outro atributo, ou ainda adaptação desse produto a um ambiente modificado”.

Para uma melhor caracterização, a manutenção pode ser classificada em quatro categorias de acordo com sua natureza: adaptativa, na qual, alterações são feitas para adaptar o sistema a novos ambientes; perfectiva, que visa atender novas funções; corretiva que tem a finalidade de corrigir erros; e preventiva que foca na melhoria da manutenibilidade do software, normalmente realizada por meio de uma reengenharia (PRESSMAN, 2006).

Embora os dados variem, diversas pesquisas indicam que os esforços com a manutenção são em grande parte devido às melhorias (geralmente 75% a 80% focadas em manutenções adaptativas ou perfectivas) ao invés de correções (PRESSMAN, 2006), o que destaca o investimento das corporações em aperfeiçoamento dos seus sistemas legados.

Porém, mesmo com essa constatação o processo de manutenção ainda não pode ser considerado um processo difundido e maduro. Existem algumas normas, como por exemplo, ISO/IEC 12207 (ISO, 2008), que definem diretrizes do processo, contudo é fato que a abordagem mais usada é a de correção rápida que foca primeiro no código a ser alterado e posteriormente na documentação, quando esta existe (CANFORA e CIMITILE, 2000). Isto se justifica pela expectativa que a maioria das manutenções deve ser uma alteração rápida e econômica, o que acaba por negligenciar o planejamento, a análise de impacto e os testes.

2.3 Reengenharia de Software

A falta de preocupação com a arquitetura do software como um todo durante a manutenção ocasiona muitas vezes a deterioração da estrutura do sistema, acompanhada por documentação desatualizada e codificações de má qualidade, o que por fim acaba por inviabilizar sucessivas manutenções.

Quando um sistema de grande importância para o negócio chega neste nível, isto é, não suporta mais manutenções que garantam que a funcionalidade geral permanecerá intacta após sua execução, surge a necessidade de reestruturação a fim de garantir sua continuidade e viabilizar futuras manutenções (PIEKARSKI e QUINÁIA, 2000).

Definida por Arnold (1993) como "... qualquer atividade que amplia a compreensão do software, ou prepara ou melhora o software em si, geralmente por aumento de manutenibilidade, reusabilidade ou capacidade de evolução" a Reengenharia de Software destaca-se como uma técnica de manutenção que permite dar uma sobrevida a um sistema legado em declínio.

Quando é confirmada sua necessidade, existem duas atividades básicas que a compõem: engenharia reversa e engenharia progressiva. A primeira fornece uma visão alternativa do sistema em um diferente nível de abstração, a fim de permitir uma melhor compreensão da lógica usada, enquanto a segunda tem como objetivo reconstruir o sistema a partir da visão identificada na fase de análise (IEEE-1219, 1998).

Já, segundo Jacobson e Lindström (1991), no processo de Reengenharia de Software há um elemento a mais:

Reengenharia = Engenharia Reversa + Δ + Engenharia Progressiva

Onde Δ pode ser: alterações parciais de funcionalidade ou alterações de implementação. Em resumo, a reengenharia contempla a fase de análise, de alteração e por fim a atividade de recriar o sistema, que por sua abrangência contempla a atividade anterior.

Assim como todo processo que implica mudanças, aplicar a reengenharia é uma tarefa difícil, pois leva tempo, tem um custo significativo em dinheiro, e absorve recursos que poderiam, por outro lado, ser usados em necessidades imediatas (PRESSMAN, 2006). São por estes motivos que a tomada de decisão por uma reestruturação deve ser feita de forma consciente e cuidadosa, pois o sistema em questão normalmente suporta uma atividade crítica da empresa.

2.4 Engenharia Reversa

Como anteriormente apresentada, a engenharia reversa pode ser encarada como uma atividade da reengenharia, porém sua utilidade vai além, pois também pode servir de apoio à manutenção de software ou simplesmente para recuperação de documentação.

Na manutenção, a engenharia reversa destaca-se diante dos inúmeros casos que não se pode contar com qualquer tipo de especificação para o entendimento do sistema a ser modificado, tarefa essencial no processo de manutenção e que tem sua importância destacada por evidências de que se gasta de 50% até 90% de tempo somente na compreensão do software (CANFORA e CIMITILE, 2000).

Chikofsky e Cross (1990) definem engenharia reversa como o processo de análise de um sistema a fim de identificar os componentes e suas inter-relações e criar representações em outra forma ou em um nível maior de abstração. Estas representações têm como objetivo mostrar diferentes perspectivas do sistema e podem ser classificadas como: visão implementacional, abstração das características da linguagem e específicas da implementação; visão estrutural, foca nas dependências entre os componentes do sistema; visão funcional que identifica as funções dos componentes; e visão de domínio que indica o porquê do sistema no contexto em que opera (HARANDI e NING, 1990).

É importante ressaltar que na maioria dos casos a visão obtida após a engenharia reversa não é uma cópia fiel da representação original, pois esta última reflete a compreensão do problema pelo analista de desenvolvimento, enquanto que o produto final da engenharia reversa é a representação do código fonte tomado como partida inicial do processo (PIEKARSKI e QUINÁIA, 2000).

De acordo com Canfora e Di Penta (2008), o processo mais difundido de engenharia reversa engloba duas atividades macros: extração de informação e abstração. Na primeira é feita análise dos artefatos disponíveis para a extração de dados brutos, enquanto na segunda são criados documentos orientados ao usuário e visões do sistema.

Ainda segundo esses autores, a análise para extração de informações pode ser estática, dinâmica e/ou histórica e esta classificação pode ser feita de acordo com a natureza do artefato a ser analisado, que varia de código fonte, documentação e *traces* até experiências pessoais e conhecimentos gerais sobre o problema e o domínio de aplicação (CHIKOFSKY e CROSS, 1990). Na maioria dos casos usa-se análise estática por ser mais rápida e barata, porém, muitas vezes por particularidades de implementação faz-se necessário o emprego da análise dinâmica

gerando assim uma terceira classificação: a análise híbrida (CANFORA e DI PENTA, 2008).

Por fim, Chikofsky e Cross (1990) ainda ressaltam que a engenharia reversa pode focar na tarefa de redocumentar uma visão representando-a de forma diferente, porém no mesmo nível de abstração, ou em diferentes tipos de recuperação: recuperação de arquiteturas, de diagramas UML (UML, 2004), de requisitos e da rastreabilidade entre código fonte e documentação de alto nível.

2.5 Desafios na Manutenção

Ainda que existam esforços na literatura que se dedicam ao estudo do processo de manutenção de software, esta área ainda apresenta desafios que surgem na sua aplicação. Há uma gama de fatores que se destacam como determinantes para o sucesso da manutenção e que falham a cada nova tentativa de modificação no sistema.

Entre os problemas que interferem na atividade de manutenção em sistemas legados podem-se destacar (PADUELLI e SANCHES, 2006):

- Ausência de um processo de manutenção de software: não há procedimento padrão a ser seguido, o foco principal é realizar a atividade com rapidez a fim de atender o prazo e expectativa do cliente.

- Instabilidade da equipe: uma vez que a falta de documentação é um fato, os custos seriam menores se a mesma equipe envolvida no desenvolvimento executasse a manutenção, pois o conhecimento adquirido previamente no desenvolvimento contribuiria de forma crucial na manutenção.

- Sobrecarga de tarefas: falta uma equipe específica de manutenção, é comum os profissionais alocados para manutenção terem outra tarefa em paralelo.

- Falhas de comunicação com o usuário: nem sempre o usuário compreende totalmente o que está solicitando, muitas vezes acredita que uma determinada manutenção será o suficiente para adequar o software à sua nova necessidade, o que nem sempre é verdade.

- Documentação insuficiente e de baixa qualidade: com a falta de um processo e a necessidade de atender a curtos prazos a atividade de documentar o que está sendo alterado passa para segundo plano.

- Falta de responsabilidade contratual: uma vez que não há o compromisso formal com a manutenção do sistema, não existe a preocupação com um suporte a mudança futura, os sistemas são criados sem atenção a manutenibilidade.

- Habilidade e experiência da equipe: são prejudicadas pelo conhecimento limitado do domínio. Na maioria das vezes não há treinamento adequado.

- Compreensão do sistema: o entendimento das funções do sistema em operação é a base para identificar onde as modificações serão realizadas e quais os impactos que essa alteração pode refletir no sistema como um todo. Normalmente esta atividade é agravada pela alta complexidade e tamanho do sistema.

- Análise de impacto: conhecendo o sistema que sofrerá a alteração é possível determinar quais os efeitos da manutenção no sistema, porém, como na maioria dos casos o mantenedor desconhece o software fica inviável mensurar quais os impactos que poderão surgir no decorrer da manutenção.

- Testes: esta atividade é intimamente ligada às duas atividades anteriores, uma vez que através do conhecimento do sistema e do impacto que a modificação pode causar fica mais fácil definir os testes que realmente devem ser realizados para garantir a integridade não só do que foi alterado, mas também dos módulos do sistema que não sofreram modificação

2.6 Compreensão do Sistema

Conforme descrito na seção anterior a compreensão do sistema é encarada como um desafio no processo de manutenção. Dentre todos os fatores expostos ela destaca-se, pois é crucial o entendimento das funções do software a ser desenvolvido, atentando-se ao fato, porém, que quando se trata de manutenção o entendimento do que deve ser implementado não é o suficiente, se faz necessária a compreensão das funcionalidades já existentes, a fim de identificar os pontos que

devem ser alterados e quais os impactos que a modificação pode causar no sistema como um todo.

Segundo Tilley (1998), os mecanismos que auxiliam a compreensão do sistema podem ser classificados em três categorias:

- Navegação sem ajuda: o engenheiro de software sem nenhum auxílio navega pelo sistema propriamente dito, ou analisa o código.

- Aproveitamento do conhecimento corporativo e da experiência: pode ser realizado por meio de orientação ou através de entrevistas informais com o pessoal qualificado. Considerando a disponibilidade de uma equipe que conhece o sistema como profundidade, e que tenha participado de sua evolução, pode ser uma abordagem de grande utilidade, porém vale lembrar que os detentores de conhecimento corporativo nem sempre são acessíveis, visto que a rotatividade dentro do ambiente empresarial pode ser considerável.

- Técnicas assistidas por computador: um exemplo claro desta categoria é a engenharia reversa que auxilia na extração de informações de alto nível partindo de uma abstração de baixo nível, como por exemplo, o código fonte.

Dentre as categorias propostas, Tilley (1998) destaca que é comum que a navegação sem ajuda seja executada em todos os processos de compreensão de software, uma vez que é uma tarefa característica de processos de análise. As demais possibilidades são definidas de acordo com os recursos disponíveis para o processo de manutenção.

2.7 Recuperação de Requisitos

Para qualquer reengenharia de um sistema legado como alteração, modificação ou completa reconstrução, entender os requisitos originais pode ser importante e muitas vezes crítico (LIU, ALDERSON e QURESHI, 1999). Porém, esta atividade sofre com alguns obstáculos que repercutem no processo de manutenção como um todo, uma vez que a compreensão do sistema a ser mantido é o passo inicial do processo.

Entre as muitas dificuldades encontradas destaca-se a complexidade do sistema, que, a cada nova modificação que o software sofre, aumenta de forma considerável, pois em sua maioria essas intervenções acontecem de forma desorganizada, sem um processo consistente, contribuindo para a deterioração da estrutura do mesmo.

Outro ponto a se considerar é a falta de documentação, que quando existe normalmente se encontra desatualizada raramente refletindo o sistema implementado. A maior parte dos casos é reflexo de alterações realizadas em função da urgência requerida pela dinâmica do negócio, o que acaba por realimentar o caos para futuras mudanças. Nessas situações se destacam as práticas de engenharia reversa que possuem meios de buscar a funcionalidade do sistema sem o apoio da documentação.

Por fim, outro dificultador no processo de recuperação de requisitos é a ausência dos *stakeholders* originais, pessoas envolvidas no desenvolvimento do sistema, que normalmente não são recursos 100% disponíveis no momento da manutenção, o que faz com o que o conhecimento sobre o software por eles adquiridos seja um bem perdido para a organização.

É neste cenário que surge a possibilidade do emprego de processos para a recuperação de requisitos em sistemas em produção, uma vez que as únicas fontes de informações são o código-fonte e aqueles que o usam. Porém, a recuperação a partir deles não é uma tarefa fácil, visto que é preciso reconhecer decisões tomadas pelos projetistas e distinguir estas decisões dos requisitos reais (ESPINDOLA, MAJDENBAUM e AUDY, 2004).

2.8 Considerações do Capítulo

A importância dos sistemas legados nos ambientes corporativos evidencia como o processo de manutenção é tarefa essencial e presente no dia a dia das empresas. É a partir dela que o sistema é adaptado e melhorado a fim de continuar a manter as principais regras de negócio que suportam as organizações.

A manutenção pode ter alto custo e ser trabalhosa, isso porque ainda não existem processos maduros e bem difundidos que permitem contornar com

facilidade os desafios encontrados, como a instabilidade da equipe, falhas na comunicação com os usuários, documentação insuficiente ou de baixa qualidade e a compreensão do sistema.

Técnicas de Engenharia Reversa e a Reengenharia de Software fazem parte do contexto de manutenção e dão sua contribuição permitindo o entendimento dos sistemas e a reestruturação destes quando não atendem mais às necessidades dos usuários.

Considerando os diversos níveis de abstração que a Engenharia Reversa pode englobar, este trabalho consolida-se tratando da recuperação de requisitos a partir de sistema legados, atividade que tem sua importância justificada na relevância do entendimento do sistema na manutenção. Em tempo, o termo requisitos é usado para fazer referência à funcionalidade do sistema implementado.

3. PROCESSOS DE RECUPERAÇÃO DE REQUISITOS

Este capítulo apresenta alguns processos de recuperação de requisitos difundidos na literatura.

3.1 Processos de Recuperação de Requisitos

Na manutenção de software, a compreensão do sistema é considerada tarefa essencial e a de maior complexidade (Liu et al., 1998). Por este motivo, existem vertentes nesta disciplina que dedicam suas pesquisas à viabilização desta atividade de forma mais produtiva, através da construção de processos que permitem a recuperação dos requisitos para melhor compreendê-lo.

No processo de recuperação podem ser empregadas duas abordagens distintas: *top-down* e *bottom-up*. A primeira é considerada uma abordagem comportamental que trata o sistema como uma caixa-preta examinando entradas e saídas do software dentro do contexto de operação. Por sua vez, a abordagem *bottom-up* é considerada funcional e foca na compreensão de partes internas do sistema, considerando-o como uma caixa branca, partindo de um baixo nível de abstração (WEIDERMAN et al., 1997).

Neste capítulo são apresentadas algumas publicações que atingem esses dois grupos, porém como o foco deste trabalho são os processos de análise comportamental, são explorados mais profundamente os projetos AMBOLS - Analysing and Modelling the Bahaviour of Legacy Systems (Liu et al. 1998) e CelLEST - CEL Legacy Enhancement Software Technologies (El-Ramly, Stroulia e Sorenson, 2001). A escolha pelo AMBOLS e CelLEST, foi devida a ambos aplicarem o conceito *top-down* para a recuperação, ou seja, fazem análise comportamental tratando o sistema como uma caixa preta. Esta abordagem destaca-se diante da do tipo *bottom-up* porque evita problemas de compreensão da estrutura interna do sistema legado, que não é fundamental no foco da compreensão do sistema para manutenção (WEIDERMAN et al., 1997).

3.2 Processos *Bottom-up*

Muitos dos projetos que tratam recuperação de requisitos usam o código fonte como recurso para extração das informações, isto ocorre porque em sistemas

legados é muito comum que o único produto disponível seja o código, fazendo-se necessário o uso do mesmo para a obtenção de um nível de abstração mais alto que permita a compreensão do software como um todo dentro do contexto.

Em Cohen (1994), é usado o conceito de Programação Lógica Indutiva (ILP – Inductive Logic Programming) para a descoberta da especificação em códigos da linguagem C, isto é, utilizam-se técnicas de aprendizado de máquina para a geração de especificações em Datalog (Prolog sem símbolos de função) a partir da extração de bancos de dados relacionais que tem visões expressas em códigos em C. Foi relatado no estudo que o processo construído descobriu dois terços da especificação com cerca de 60% de precisão em um código com mais de um milhão de linhas. Tal resultado é garantido, desde que os dados de treinamento usados sejam suficientes, caso contrário, os resultados englobariam inúmeras especificações inconsistentes.

Considerando a idéia de transformação formal – operação que modifica um programa em uma forma diferente, porém com o mesmo comportamento externo, Ward e Bennett (2003), também elaboraram um processo de recuperação de requisitos com foco na utilização do código fonte. O método proposto resume-se a partir de um programa P, selecionar uma transformação previamente analisada de uma biblioteca para gerar uma representação S do sistema. Podem ocorrer transformações subseqüentes para apurar a representação encontrada, sendo que o ponto final depende da necessidade do usuário que pode querer só realizar uma reestruturação simples ou precisar gerar uma especificação bem definida. Conforme descrito por Ward e Bennett (2003), para o sucesso do processo são necessárias competências não só em engenharia de software, como também no domínio da aplicação, pois este conhecimento é decisivo na interpretação das representações geradas que darão subsídios para a criação da especificação de requisitos.

Di Lucca, Fasolino e De Carlini (2000) também parte do código fonte para obter o entendimento do sistema, essa abordagem foca em códigos que implementam o paradigma orientado objetos e modela o sistema a partir da execução de *threads*. *Thread* é uma seqüência de execuções de métodos ligados por mensagens trocadas entre os objetos, é desencadeada por eventos de entrada e finalizada por eventos de saída. A idéia principal da proposta é recuperar o modelo de casos de uso que

segundo Jacobson (1987) representa os requisitos funcionais implementados. É de se considerar que esta abordagem é limitada, uma vez que a maior parte dos sistemas legados estão implementados de acordo com o paradigma estruturado.

3.3 Processos *Top-Down*

Nessa seção serão explorados os projetos AMBOLS, que consiste em atividades de investigação que usa conjunto de técnicas de análise e representação (LIU, 2005), e CelLEST que busca a funcionalidade do sistema a partir do comportamento em tempo real de seus usuários (El-Ramly, Stroulia e Sorenson, 2001). Ambos utilizam-se do conceito *top-down* para a construção de processos de recuperação de requisitos, e foram escolhidos por focarem a extração a partir da interação entre usuários e o software não fazendo uso de qualquer artefato além do executável.

3.3.1 AMBOLS

O projeto de pesquisa Analysing and Modelling the Bahaviour of Legacy Systems – AMBOLS (Liu et al., 1998) criou um processo que visa à recuperação de requisitos baseando-se em entrevistas sobre as capacidades que o sistema provê a seus usuários e na observação destes em interação com o sistema. A motivação dessa abordagem está relacionada a questões de como as regras organizacionais, culturais e contextuais governam o comportamento das pessoas. Alguns ramos da filosofia, lingüística e psicologia cognitiva formaram sistemáticas respostas a estas perguntas e, por este motivo o processo AMBOLS adota a semiótica organizacional como fundamento teórico.

Os conceitos que servem de base para os métodos semióticos remontam do trabalho de Charles Peirce (1960), que define semiótica como a “doutrina formal dos signos”. Signo, por sua vez, é definido pelo estudioso como “algo que, sob certo aspecto ou de algum modo, representa alguma coisa para alguém” (Peirce, 1960).

Ao adotar a semiótica organizacional como fundamento teórico, a proposta de Liu et. al. (1998) vê a organização como um sistema onde sinais são criados para permitir a comunicação necessária ao negócio. Três conceitos importantes da psicologia são abordados nessa teoria: agente, padrão de comportamento e normas.

O primeiro diz respeito ao responsável pela tarefa, o segundo termo representa a capacidade, padrão de ação ou comportamento de um agente, e por fim a norma estabelece como e quando a instância de um padrão de comportamento ocorrerá. Abaixo a Figura 2 relaciona estes conceitos.

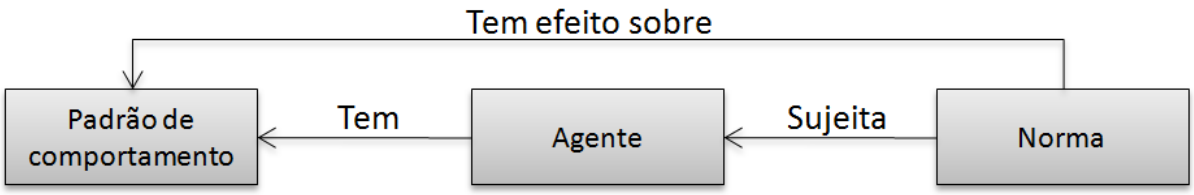

Figura 2. Relacionamento dos conceitos semióticos (Adaptado de Liu et. al 1998)

O processo AMBOLS fundamenta-se nesses princípios para melhor entender a relação entre o sistema e o contexto de trabalho, ou seja, entender como as pessoas interagem com software, a fim de identificar quais são as necessidades dos usuários supridas por ele. A análise é realizada então, utilizando-se dos métodos semióticos de análise semântica e análise de normas.

No processo ilustrado na Figura 3 é possível reconhecer três estágios principais: captura de comportamento, modelagem dinâmica do comportamento e derivação dos requisitos.

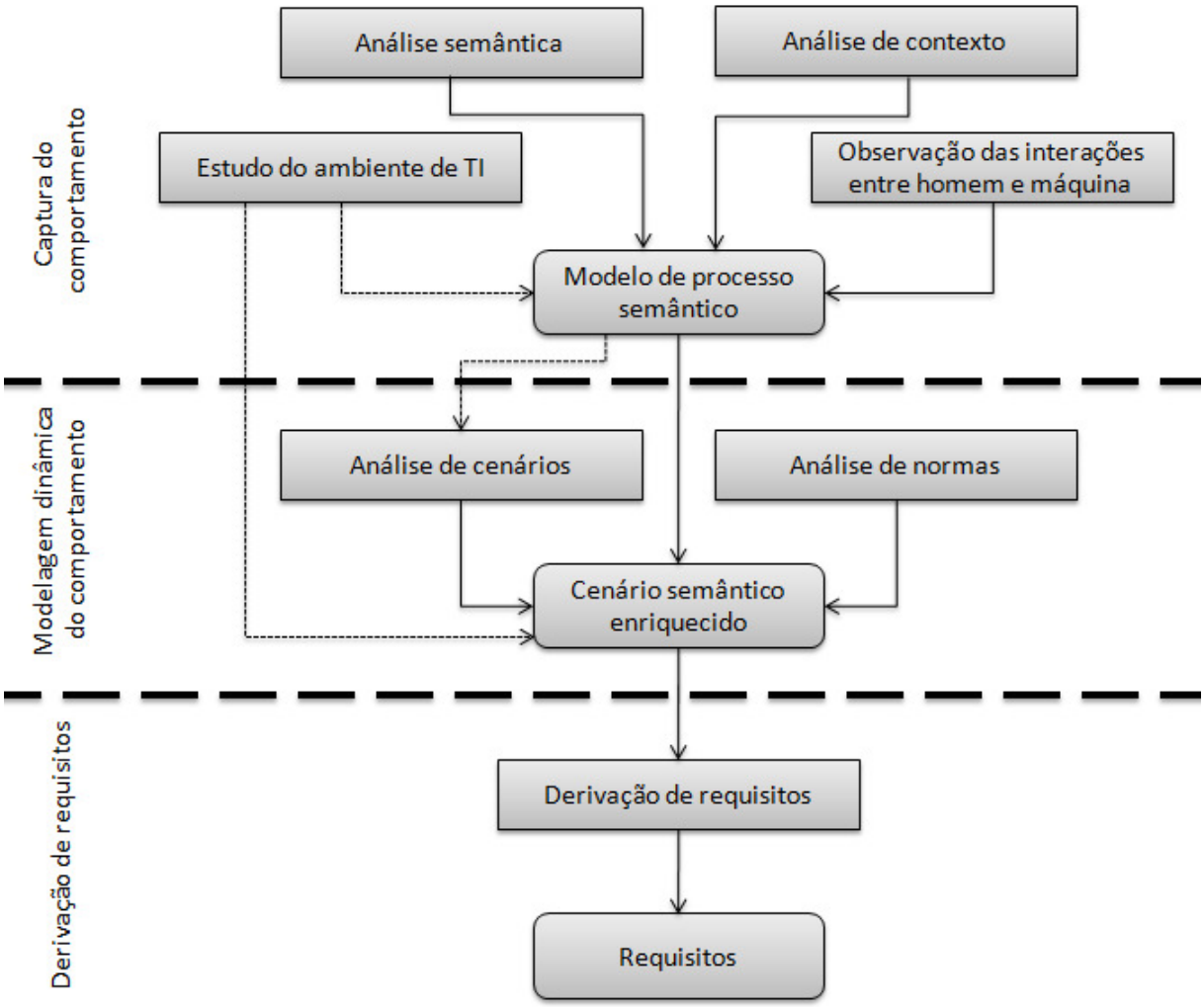

Figura 3. Abordagem AMBOLS (LIU, ALDERSON e QURESHI, 1999)

3.3.1.1 Captura do Comportamento

Este estágio é responsável por identificar as funções do negócio, a relação entre elas e quais são as funções do sistema que as atende. Seu objetivo é alcançado através da análise do domínio, da estrutura organizacional e do negócio. O resultado esperado é a interação do usuário e sistema descrita em um modelo de processo, representado por um diagrama de atividades da UML (UML, 2004), e conceitos e termos usados neste processo definidos num modelo semântico. É desmembrado em três fases principais:

a) Análise de Contexto

Examina o ambiente organizacional e de negócio no qual o sistema opera. Sua meta é entender a natureza do negócio e suas principais funções, além de identificar

stakeholders, através de entrevistas e questionários tipo *checklists*. Por fim, os relacionamentos identificados entre *stakeholders* e as funções são descritas a partir de casos de uso da UML (UML, 2004).

b) Análise Semântica

A principal função da análise semântica é criar, a partir do objeto de estudo, uma interpretação deste dentro do contexto em que se encontra. No cenário da manutenção de software, a análise semântica é usada então para interpretar as operações e normas do sistema dentro do cenário em qual atua. Em termos da semiótica organizacional, o objetivo é alcançado considerando o conceito do padrão de comportamento do agente aplicado ao ambiente.

Esta análise é feita a partir do estudo do significado de conceitos, termos e funções para identificação de padrões de comportamento que são descritos em um modelo semântico. No processo AMBOLS, a representação desse modelo foi feita a partir de um gráfico de ontologia que descreve os padrões de comportamento dos atores e sistemas, e define as relações de conceitos e termos que correspondem a esses padrões.

O gráfico de ontologia foi escolhido por representar as dependências ontológicas entre padrões de comportamento dos agentes, pois nele é expresso que entre os antecedentes da esquerda e dependentes, à direita, existe uma condição prévia para que o padrão de comportamento no lado direito da linha seja reconhecido.

A Figura 4 abaixo ilustra um exemplo desta representação. Conforme o gráfico, quando um usuário da biblioteca faz um empréstimo ele abre a possibilidade da ação de devolver, logo a devolução é dependente do empréstimo, que por sua vez tem como pré-requisito a existência de uma pessoa e da biblioteca para se tornar significativo. Nesta representação as elipses indicam os agentes, que podem ser classificados de forma diferente de acordo com seus atributos e os retângulos são os padrões de comportamento destes agentes.

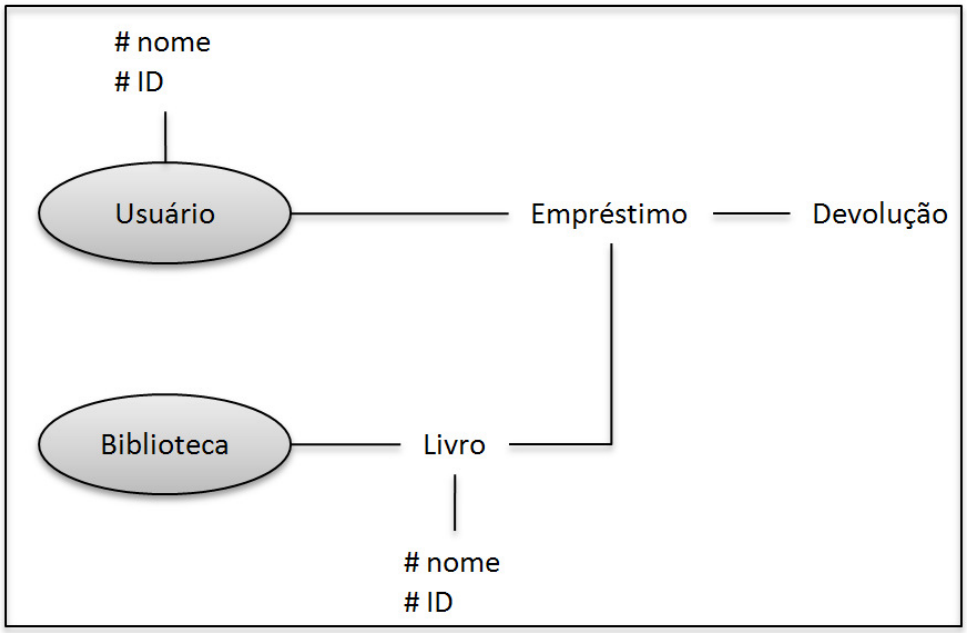

Figura 4. Exemplo de um gráfico de ontologia

Diante do exposto fica claro que o papel da análise semântica é especificar quem faz o que e em quais circunstancias os padrões de comportamento entram em vigor. No primeiro momento estas análises podem parecer triviais, porém quando se considera um sistema de grande porte e complexo este modelo permite a criação de representação rigorosa viabilizando a compreensão do sistema.

c) Observação das Interações Entre Homem e Máquina

É realizada a partir do estudo de como o sistema é utilizado por todos os grupos de usuários. Notas detalhadas são tomadas e, muitas vezes, explicações adicionais por parte dos usuários são necessárias sobre os significados, os propósitos e a razão para certas operações. Cabe salientar que as interações não correspondem somente a telas e entradas de teclado, também podem ser considerados quaisquer artefatos que representem a interação do usuário com o sistema.

d) Estudo do Ambiente de TI

No estudo apresentado por Liu et. al (1998) não foi abordada de forma explícita a atividade de estudo do ambiente de TI, porém, no decorrer das demais atividades que compreendem a captura e modelagem do comportamento fica clara a

importância de considerar o ambiente tecnológico no qual o sistema está inserido. Seja durante a observação da interação ou a partir da análise do domínio, da estrutura organizacional ou do negócio.

3.3.1.2 Modelagem Dinâmica do Comportamento

O segundo estágio foca na seqüência de ações e interações entre o usuário e o sistema, considerando a análise de cenários (seqüência de operações *versus* regras de negócios), normas e detalhes como modelo de dados e telas. O propósito final é representar o domínio do problema com cenários enriquecidos semanticamente, consistentes com os modelos de processos e com clara conexão com as normas que regem as operações.

Assim como o estágio anterior é divido em fases:

a) Análise de Cenários

As operações realizadas no sistema de acordo com as regras de negócio são analisadas diante alguns pontos de vista específicos:

- Ponto de vista de processo: cenários descrevem as seqüências de ações ou eventos, não fatos individuais.

- Ponto de vista de situação: cenários representam a situação ou episódio com um componente temporal

- Ponto de vista de escolha: cenários representam alternativas entre projeto ou implementação.

- Ponto de vista de uso: cenários permitem usuário centrar-se sobre o artefato alvo.

A perspectiva escolhida pelo AMBOLS é a de processos enriquecida por um componente temporal, permitindo a representação das partes interessadas, ações e eventos segundo as operações do sistema baseando-se no tempo.

b) Análise de Normas

Na modelagem de processos de negócio a maioria das regras, sejam formais, informais ou regulamentos, é classificada como normas de comportamento, uma vez que prescrevem o que as pessoas devem, podem e não podem fazer.

A presença de normas viabiliza então padrões de comportamento, conhecidos também por padrões normativos, que são caracterizados por começo e fim e se mantém invariáveis durante o período entre os dois marcos. As normas controlam quando ocorre cada ocorrência de um padrão de comportamento, ou seja, indicam o início, o término ou se o padrão vai realmente existir.

Uma norma comportamental pode ser representada por quatro elementos-chave: elemento temporal, condicional, de permissão ou obrigação, e de ação, conforme exemplo:

Sempre *condição,* ***se*** *afirmação* ***então*** *agente* ***é*** *operador deôntico* ***a fazer*** *ação.*

Onde, o operador deôntico representa obrigação, permissão ou proibição, e o agente é um tipo particular que pode assumir responsabilidade por suas próprias ações e as ações de outros. Abaixo segue exemplo de norma:

Sempre que o operador da célula de cadastro requisitar cadastro de novo estabelecimento **se** o cadastro for realizado com sucesso **então** o sistema **pode solicitar** ordem de serviço para instalação de um meio de captura.

A análise das normas se mostra eficaz, pois uma vez conhecidos os agentes e sob quais normas executam suas atividades, então pode ser deduzir o que o indivíduo ou grupo vai precisar e o que produzem para outras pessoas usarem, isto porque as regras governam o comportamento dos agentes e as operações que o sistema executa.

Considerando os conceitos descritos pode-se dizer que a análise semântica e de normas se completam para compreender o contexto de aplicação do sistema, pois de nada adiantaria reconhecer os padrões de comportamento se não se sabe

quais as regras que os regem, não sendo capaz, então, de falar sobre a realização de um padrão específico.

3.3.1.3 Derivação de Requisitos

O último estágio tem como produto final o documento de requisitos que define as funções do sistema em operação. Tal resultado é obtido a partir das saídas das duas fases anteriores, ou seja, do modelo de processos com conexões claras com as normas que regem as mudanças de estados e controle de ações. O processo para produzir o documento é composto por duas atividades que geralmente são executadas em iterações: a síntese e derivação, e a validação.

Usam-se os conceitos semióticos de dedução, indução e abdução para sintetizar e checar os requisitos levantados. Segundo Peirce (1960), dedução parte do geral para o particular, a indução parte de uma premissa menor para uma maior e a abdução afirma um caso a partir de uma regra ou resultado.

É considerando estes três tipos de raciocínio que os requisitos são derivados do conhecimento do domínio, que é representado por termos (ontologias) usados pelos usuários para descrever o sistema e expressos no modelo semântico construído nos dois primeiros estágios do processo.

Por fim ocorre a validação, fase na qual o analista juntamente com usuários, *stakeholders* e engenheiros de manutenção (todos os envolvidos disponíveis) analisam se os requisitos extraídos estão coerentes com as operações realizadas pelo sistema. É importante ressaltar que não há garantia que o resultado seja os requisitos iniciais que deram origem ao sistema, pois o que se busca com o processo é a recuperação dos requisitos funcionais dos usuários do sistema em produção. O sucesso do processo é a aceitação do usuário a partir de testes sobre esses requisitos descobertos.

3.3.2 CelLEST

Motivado por investigar o problema de migração de sistemas legados para plataforma Web e, mais especificamente, o problema de recuperação de requisitos funcionais do aplicativo legado, uma vez que há a necessidade de mantê-los através do processo de migração, um grupo de estudo da Universidade de Alberta, Canadá

criou o processo CelLEST - CEL Legacy Enhancement Software Technologies (El-Ramly, Stroulia e Sorenson, 2001), cujo objetivo não é documentar os requisitos que levaram ao desenvolvimento da aplicação inicial, mas capturar os que ditam o uso do sistema atual.

A abordagem usada baseia-se na mineração de dados para descobrir padrões seqüenciais a partir das tarefas freqüentes dos usuários, expressas em *traces* do comportamento do sistema legado durante a operação. O termo *trace*, no contexto do trabalho pode ser interpretado como log de execução, mais especificamente captura de telas que evidenciem as ações realizadas pelo usuário durante interação com o sistema.

Para atender a formulação do problema de recuperação de requisitos como uma instância do problema de mineração de padrões seqüenciais foi desenvolvido um algoritmo para a descoberta de padrões de interação, chamados assim porque são subseqüências da interação com o sistema legado, que identificam as operações realizadas pelo usuário e, portanto, a funcionalidade do sistema usado por ele.

Assim como ilustrado na Figura 5, o processo é desmembrado em duas etapas: na primeira é construído um mapa da interface do sistema considerando os *traces* da interação com o mesmo, e na segunda é produzido um modelo abstrato da tarefa de forma declarativa (requisitos) que servirá para a geração de uma nova interface gráfica adaptada para a tarefa em questão.

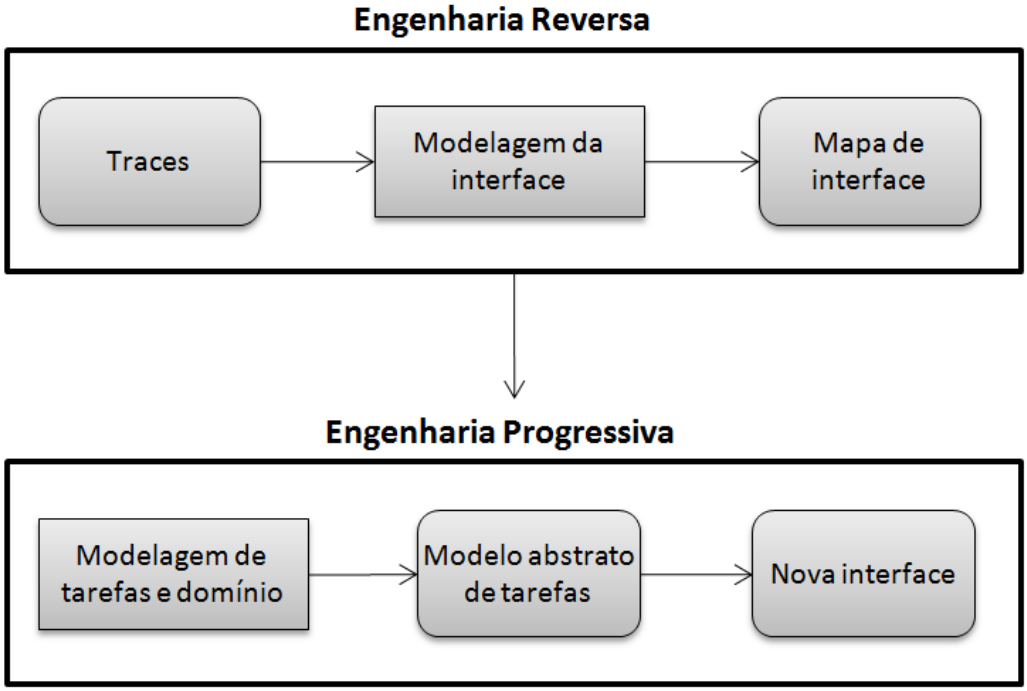

Figura 5. Processo CelLEST

3.3.2.1 Engenharia Reversa

A etapa de engenharia reversa (Figura 6) é caracterizada pela modelagem da interação entre o sistema e o usuário. Em resumo pode-se dividi-la em três passos principais:

- Identificação das telas: consiste na captura dos *traces* de interação entre sistema e usuários.

- Reconhecimento das interações entre sistema e usuário: são identificadas seqüencias de interações comuns que posteriormente são representadas em modelos de transição de estados.

- Extração de padrões de interação: o modelo de transição de estados é explorado a fim de identificar padrões de interação, que por sua vez representam a funcionalidade que o sistema atende.

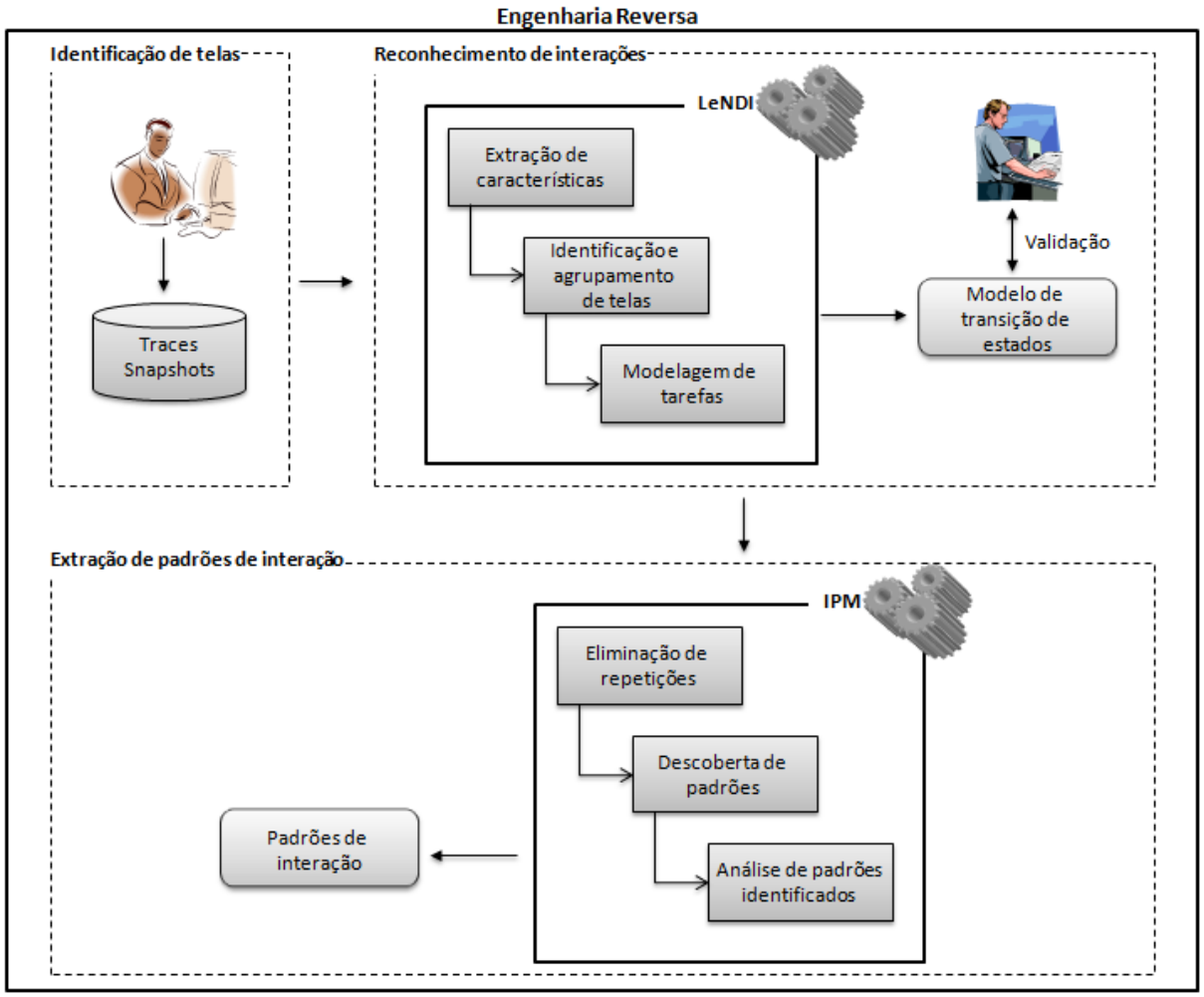

Figura 6. Etapa de Engenharia Reversa do processo CelLEST

a) Identificação de Telas

O primeiro passo para a implantação do processo CelLEST é a captura de *traces* do comportamento do sistema legado. Como supracitado o *trace* é identificado como a captura de telas que mostram a interação entre usuário e sistema. Cada conjunto de *traces* pode representar múltiplas execuções de uma única tarefa do usuário, isto é, para atender a uma funcionalidade o usuário pode executar diversas operações em uma mesma tela que tem vários estados ou em mais de uma tela, gerando assim um conjunto de *traces* que representa uma única tarefa.

A coleta de informações pode ser feita de duas maneiras, quando se trata de um sistema simples a captura pode acontecer de forma manual, porém quando o sistema é de grande porte e as ações do usuário são complexas faz-se necessário o uso de ferramenta que auxilie essa captura. Para o CelLEST foi criada a ferramenta

LeNDI (Legacy Navigation Domain Identifier) que também é capaz de realizar esta atividade para algumas plataformas específicas, porém como não se trata de tarefa obrigatoriamente executada pela ferramenta é possível tratá-la como um passo inicial do processo e independente dos demais.

b) Reconhecimento das Interações

Esta fase é marcada pelo uso do protótipo LeNDI que permite a criação de um modelo de transição de estados (mapa de interface) a partir dos *traces* de interação. Neste modelo gerado cada estado corresponde a uma tela diferente e cada transição a uma ação do usuário que pode levar a uma nova tela ou não. Para fins de execução a cada tela é atribuído um identificador, e uma seqüencia de execução é representada pela seqüencia de identificadores. Por fim, ocorre a validação do modelo por um analista e um usuário especialista, que contribui com explicações de como as tarefas representadas ocorrem.

A Figura 6 ilustra o processo aplicado pela ferramenta LeNDI, identificando três atividades macros:

- Extração de características: pré-requisito para o agrupamento que acontece na atividade seguinte, esta extração pode considerar diversos critérios. A proposta abordada por El-Ramly, Stroulia e Sorenson (2001) sugere como características significativas palavras-chave, recursos de layout e características específicas da aplicação. A saída desse processo é um vetor de características para cada tela capturada.

- Identificação e agrupamento de telas: partindo das características extraídas as telas são agrupadas de acordo com a similaridade visual e classificadas em telas de interfaces distintas, onde cada grupo corresponde a um tipo, por exemplo, telas de menu, telas de cadastro etc.

- Modelagem da tarefa: o foco desta atividade é identificar as ações disponíveis para que um usuário possa navegar de uma tela para outra, dessa forma modelando quais transições são necessárias para executar uma tarefa a partir das telas mapeadas. Nesse contexto cada tela passa a ser considerada um estado, sendo que algumas telas têm embutidas nelas mais de um estado, e a transição

entre elas têm origem das ações que o usuário faz durante a tarefa. O resultado dessa operação é o modelo de transição de estados, também chamado de mapa de interface que é validado pelo usuário para servir de entrada para a extração de padrões de interação.

c) Extração de Padrões de Interação

O passo seguinte consiste na descoberta dos padrões de interação a partir do modelo gerado. Como mencionado o fundamento teórico parte da mineração de padrões seqüenciais, cuja proposta é a descoberta de episódios freqüentes de seqüência de eventos. O processo CelLEST foca em identificar padrões que sejam totalmente solicitados, ou seja, que atendam totalmente ao critério especificado pelo usuário. No modelo de estados cada estado recebe um identificador, e cada tarefa então é representada por uma seqüência de identificadores.

O critério de busca usado parte de seqüências menores ou menos ambíguas que quando atendem ao suporte especificado, se tornam padrões candidatos, e então são estendidas para formar seqüências mais longas ou ambíguas. Este processo continua até que não tenham mais padrões a serem descobertos, então se obtém o conjunto de padrões qualificados que atendem exatamente ao critério especificado pelo usuário.

No processo CelLEST todo este procedimento é executado pelo algoritmo IPM (Interaction Pattern Mining) que engloba três etapas:

- Pré-processamento: nessa etapa ocorre a eliminação de repetições das seqüências de entrada resultantes de diversos acessos consecutivos a mesma tela, uma vez que podem mascarar um candidato favorável. O tratamento é feito por um algoritmo que codifica a seqüência substituindo as repetições por uma contagem seguida pela identificação, como exemplo: {4,5,6,6,6,6,6,6,7} → {4,5,(6)6,7}.

- Descoberta de padrões: nessa etapa ocorre a busca exaustiva por padrões qualificados que atendam ao critério especificado pelo usuário. Este critério, segundo o algoritmo IPM é composto por quatro variáveis que indicam respectivamente, o tamanho mínimo que a seqüência deve ter, o limiar calculado para indicar o quanto a seqüência suporta um padrão, o número máximo de erros

permitidos na inserção e a pontuação mínima para classificar os padrões descobertos. Atingindo estes valores a seqüência então é classificada como qualificada e faz parte do conjunto final que representa a saída do algoritmo.

- Análise dos padrões identificados: nesta análise é feito ajustes no critério de qualificação para identificar padrões de interação que realmente correspondam a requisitos funcionais do sistema legado, um usuário com conhecimento do domínio da aplicação deve decidir quais dos padrões minerados correspondem a funcionalidades do sistema.

3.3.2.2 Engenharia Progressiva

Com os padrões qualificados que identificam a tarefa, uma ferramenta é usada para criar a especificação declarativa (requisitos) da tarefa modelada, o resultado desta etapa é um executável por um conjunto de componentes específicos que por fim geram uma interface que servirá de “front-end” para o sistema original, só que disponível em múltiplas plataformas. A idéia desta etapa é representar a funcionalidade do sistema de forma declarativa e a partir dela criar um sistema de entrada com as mesmas funções só que compatível com o novo ambiente.

3.4 Considerações do Capítulo

A necessidade de compreender a funcionalidade do sistema a ser mantido faz com que processos que foquem a descoberta dos requisitos que ditam o sistema legado ganhem destaque na literatura. Considerando os tipos de recuperação *top-down* e *bottom-up* esses processos variam basicamente em relação ao recurso utilizado para a extração das informações, que pode ser tanto de baixo nível quanto de alto nível de abstração, de acordo com a realidade do sistema legado.

No presente capítulo são apresentados processos que abordam os dois grupos de recuperação, porém respeitando o escopo deste trabalho que se limita a estudar processos de recuperação que foquem a análise comportamental foram expostos de forma mais profunda dois processos específicos, AMBOLS e CelLEST.

O processo AMBOLS fundamenta-se nos métodos semióticos de análise semântica e análise de normas para buscar a lista de requisitos que ditam o sistema

legado. Para tal, parte da compreensão do comportamento do sistema diante da interação com usuários.

Já a técnica CelLEST usa os *traces* de execução para extrair informações que dêem subsídio para a identificação das funções do software. A base teórica para esta extração está no problema de mineração de padrões seqüenciais que busca padrões que atendam a um critério previamente estabelecido. No caso da recuperação de requisitos as interações mais freqüentes entre sistema e usuário são encaradas como padrões de interação que identificam as operações realizadas pelo usuário e, portanto, a funcionalidade do sistema usada por ele.

Após uma breve descrição dos processos o próximo capítulo explorará a aplicação destes em um sistema comum. O objetivo é verificar a aplicabilidade dos processos identificando a sua aderência à funcionalidade de um sistema legado.

4. APLICAÇÃO DOS PROCESSOS

Este capítulo descreve os resultados da aplicação dos processos AMBOLS e CelLEST detalhados anteriormente em um sistema legado real. É feito um estudo comparativo a partir dos resultados obtidos de cada processo.

4.1 O Sistema Legado

A fim de validar a aplicação dos processos apresentados no capítulo anterior foi eleito um sistema no qual os mesmos pudessem ser empregados. O sistema escolhido é um sistema legado real cujos requisitos eram desconhecidos, o mesmo atende ao gerenciamento de ativos de uma operadora de cartão de crédito. No contexto deste negócio, ativos caracterizam equipamentos que fazem a leitura do cartão de crédito, também conhecidos como meios de captura.

Considerando esta definição, entenda-se gerenciamento de ativos como o controle do parque de equipamentos por estabelecimento, além de controle de ordem de serviço para instalação, manutenção e retirada. Para fins didáticos da aplicação dos processos é considerado somente o módulo que gerencia a instalação do ativo e que corresponde ao macro fluxo exposto na ilustração abaixo.

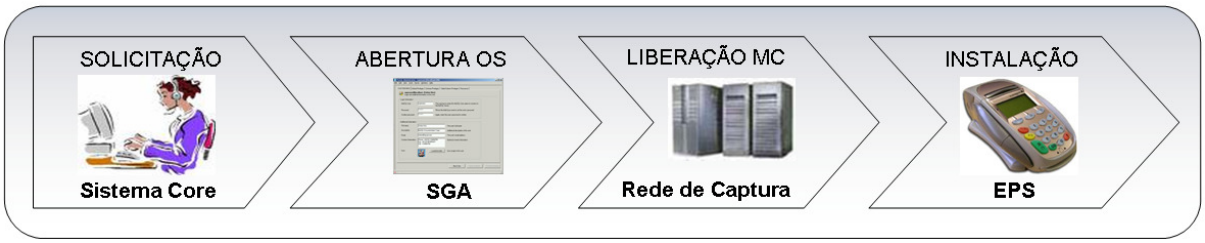

Figura 7. Macro fluxo da instalação de um ativo

Conforme Figura 7 o processo de instalação de um ativo possui quatro atores principais: Sistema Core, Sistema de Gerenciamento de Ativos (SGA), Rede de Captura e Empresa Prestadora de Serviço (EPS).

O Sistema Core é responsável pelo gerenciamento dos estabelecimentos e faz interface direta com o SGA a fim de solicitar uma nova ordem de serviço para a instalação do equipamento para seu cliente. O SGA, por sua vez, gerencia as ordens de serviços controlando todo o fluxo, desde o cadastro do novo cliente em sua base até a instalação física do equipamento. O papel da Rede de Captura é

validar as informações do estabelecimento e gerar um número de controle para associar o estabelecimento ao meio de captura. Por fim, a EPS representa a empresa conveniada responsável por instalar o equipamento fisicamente.

4.2 Metodologia de Aplicação AMBOLS

Considerando o processo AMBOLS ilustrado na Figura 3, nesta seção é executada cada atividade do processo a fim de extrair os requisitos do SGA. É importante considerar que para a aplicação dos estágios descritos usam-se técnicas disponíveis na engenharia de software e na engenharia de requisitos, podendo estas sofrer adaptações ou extensões para melhor atender ao modelo.

4.2.1 Captura de Comportamento

O resultado do primeiro estágio é a interação entre usuário e sistema descrita em um modelo de processo, representado por um diagrama de atividades da UML (UML, 2004) conforme Figura 8, e um modelo semântico que o completa retratando regras, papéis e responsabilidades dos *stakeholders* do sistema.

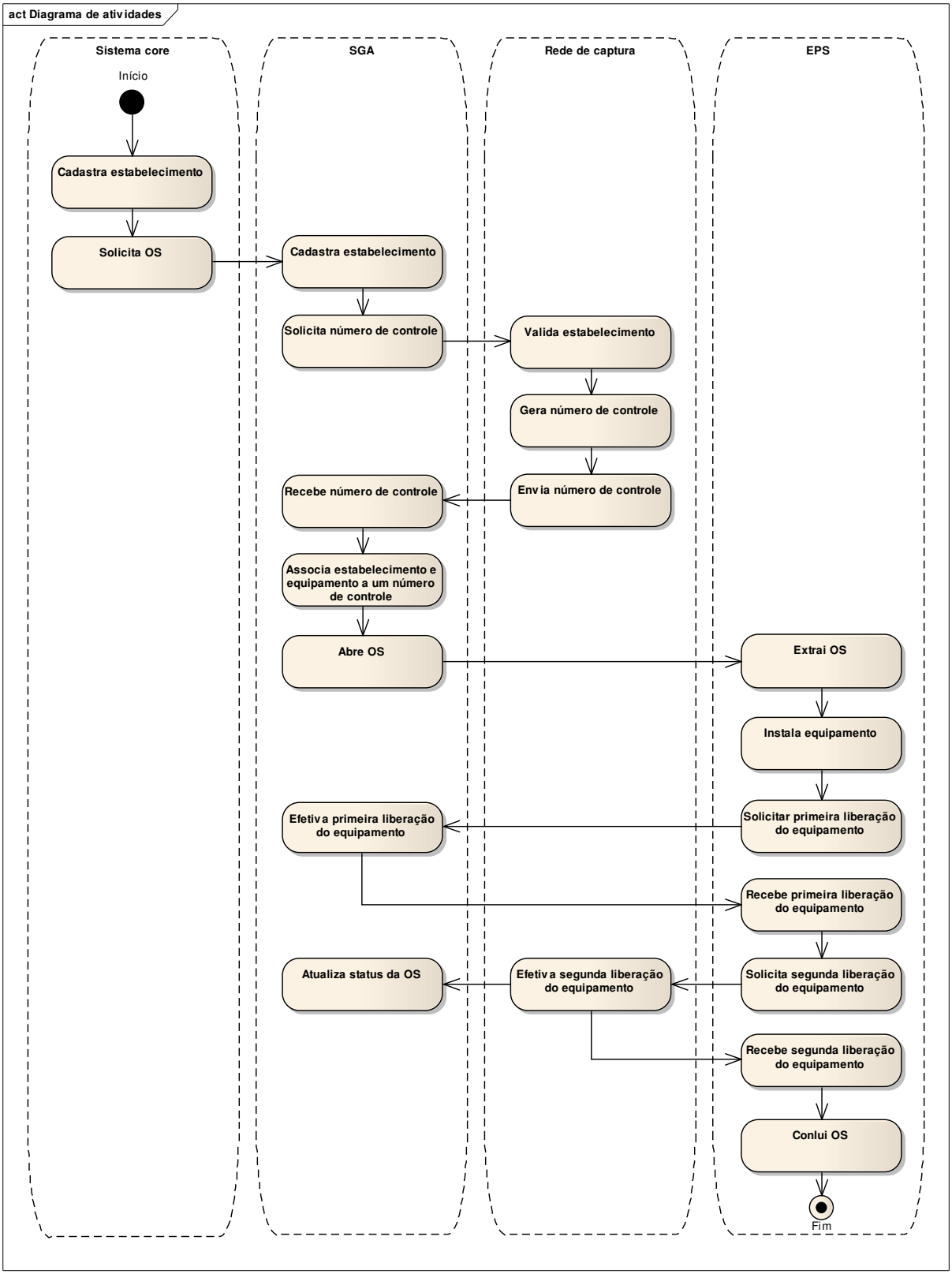

Figura 8. Diagrama de atividades SGA

Para construir o diagrama acima foram realizadas algumas etapas descritas a seguir.

4.2.1.1 Análise de Contexto

Na análise de contexto são identificados os *stakeholders* do sistema, suas responsabilidades e as principais funções desempenhadas pelo software. A representação dessa análise é ilustrada na Tabela 1, onde são listados os *stakeholders*, suas responsabilidades e a que categoria eles pertencem. Assim como no processo AMBOLS, para classificar os *stakeholders* de acordo com sua influência sobre o sistema foram consideradas as categorias: principal, ator, cliente, fornecedor, colaborador, concorrente, expectador e legislador.

Tabela 1. *Stakeholders* e responsabilidades do sistema SGA

Stakeholder	Categoria	Responsabilidade
Central de atendimento Célula de cadastro	Colaborador	Cadastro de estabelecimentos no sistema core
Sistema Core	Ator	Gerencia estabelecimentos e solicita a instalação de equipamento
Rede de Captura	Ator	Valida dados do estabelecimento, gera número de controle e efetiva a segunda liberação do equipamento
Operador EPS	Ator	Extrai ordem de serviço para execução, instala e solicita a liberação do equipamento e conclui a ordem de serviço
Central de atendimento Célula de baixa	Ator	Solicita primeira liberação do equipamento
Estabelecimento	Cliente	Solicita meio de captura

A partir das responsabilidades é possível relacionar as principais atividades e identificar as funções do sistema que as apóia, a representação escolhida para esse nível de abstração foi o diagrama de caso de uso da UML (UML, 2004), que para o módulo de gerenciamento da instalação de ativos está representado na Figura 9.

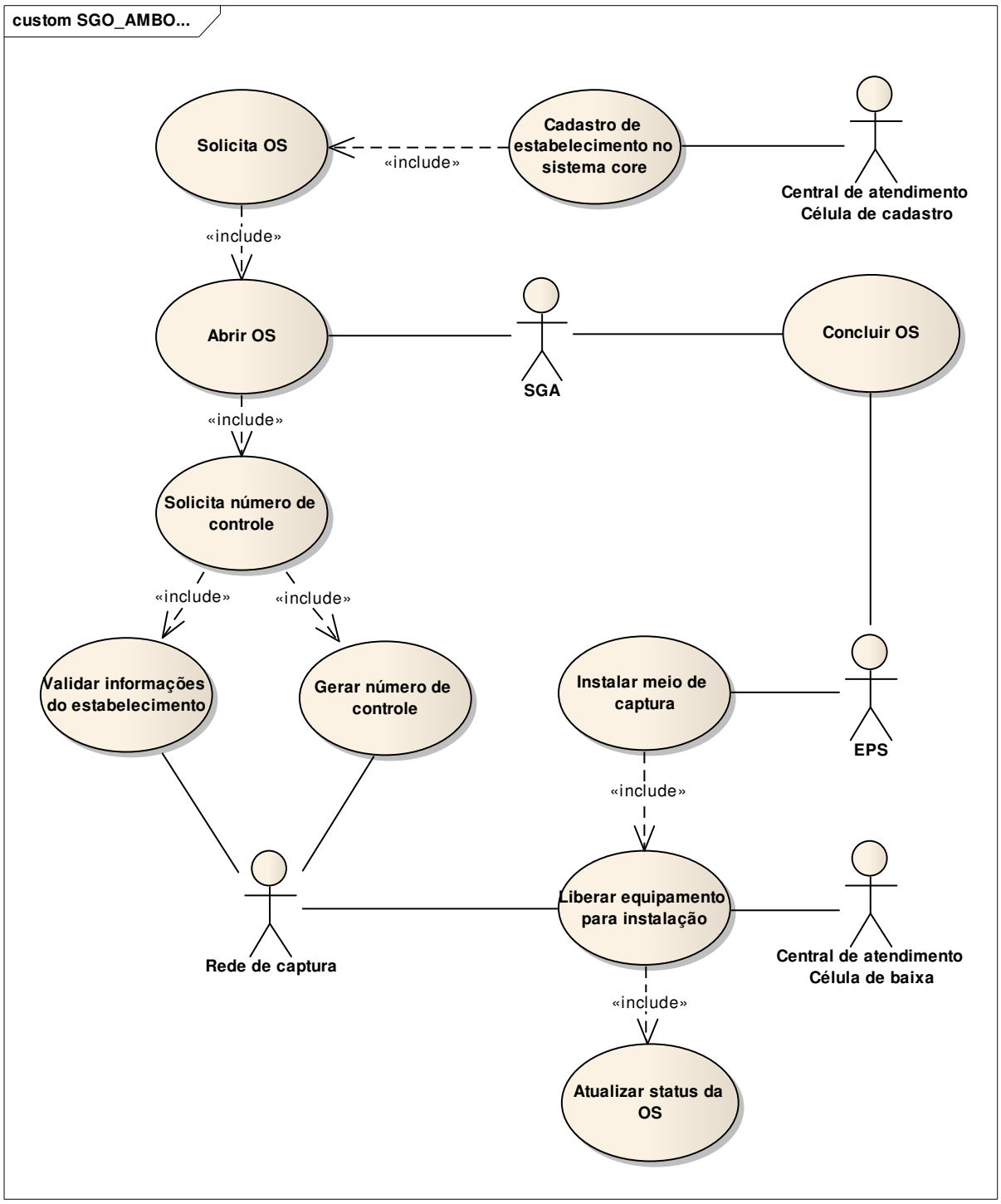

Figura 9. Casos de uso do módulo de gerenciamento de ativos

4.2.1.2 Observação das interações entre homem e máquina

A identificação das principais funções exercidas pelo sistema apóia-se na observação da interação entre usuário e sistema. Outro artifício são entrevistas realizadas para melhor compreender a operação que o usuário está executando e o que ele espera como resposta do sistema. No estudo de caso além desses recursos

foram capturadas as telas na mesma seqüência de execução realizada pelo usuário, a fim de facilitar a visualização do fluxo como um todo.

4.2.1.3 Análise Semântica

A análise semântica consiste no estudo do significado de conceitos, termos e funções a fim de definir a relação entre eles e identificar a que padrão de comportamento correspondem. Conforme exposto por Liu et. al (1998) esta análise pode ser representada em um gráfico de ontologia, que para o estudo do SGA pode ser visualizado na Figura 10.

Nele identificam-se elipses, que representam os agentes que tomam ações e podem ser responsáveis por elas e retângulos com seus padrões de comportamento determinados pelas atividades que os agentes são capazes de exercer. Outra característica importante está no fato da ordem de apresentação representar dependência entre as partes, onde o padrão de comportamento do lado direito só é reconhecido diante da existência do antecedente a sua esquerda.

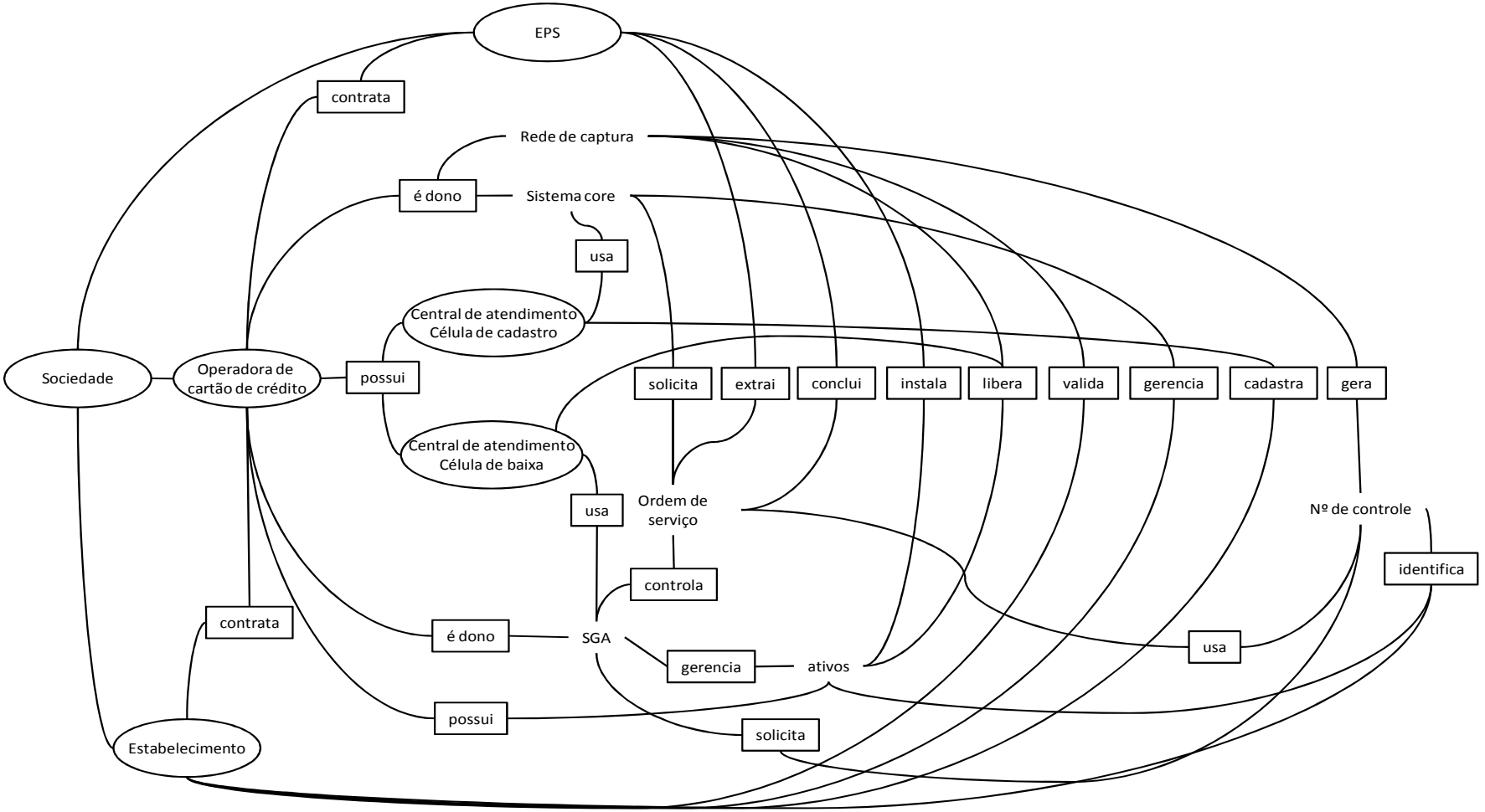

Figura 10. Gráfico de ontologia para o SGA

4.2.2 Modelagem Dinâmica do Comportamento

O segundo estágio do processo AMBOLS caracteriza-se pelo foco na seqüência de ações e interações entre usuário e sistema enfatizando a representação via cenários.

4.2.2.1 Análise de Cenários

Foi usado o ponto de vista de processo para a construção do cenário, cujo objetivo é descrever as seqüências de eventos e não fatos individuais, sempre considerando um componente temporal. O processo AMBOLS fez uso do diagrama de seqüência da UML (UML, 2004) para representar esta etapa do processo, o resultado para o sistema SGA pode ser analisado abaixo.

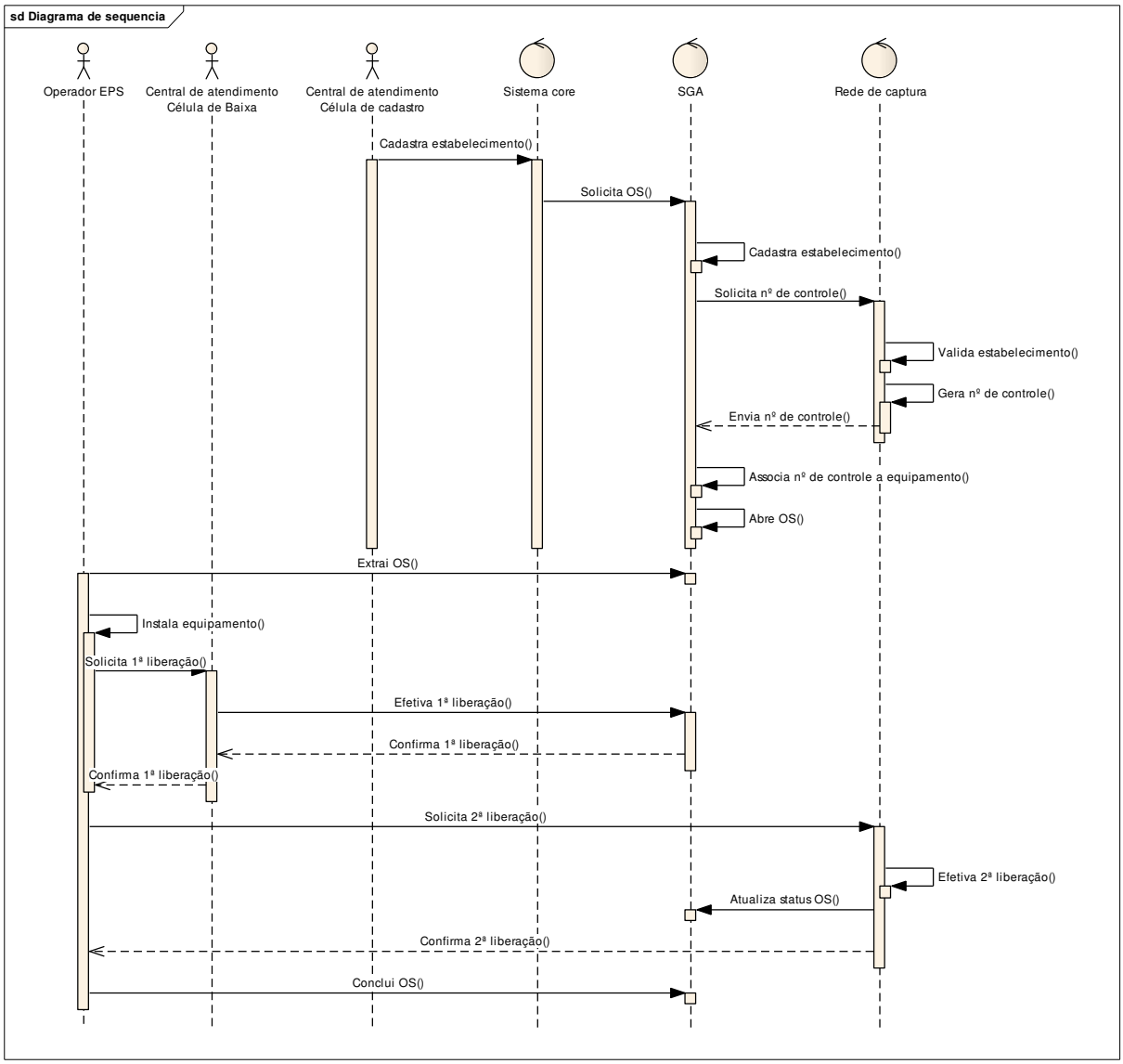

Figura 11. Cenário na visão de processo do fluxo de instalação de ativo

4.2.2.2 Análise de Normas

O processo AMBOLS descreve a análise de normas como a identificação das regras que regem os padrões de comportamento descritos no modelo semântico. Dessa forma, a entrada para esta análise é o gráfico de ontologia construído a partir da observação da interação entre usuário e sistema.

A seguir são listadas algumas regras extraídas a partir do modelo semântico criado para o estudo do SGA:

- Sempre que o operador da célula de cadastro requisitar cadastro de novo estabelecimento se o cadastro for realizado com sucesso então o sistema pode solicitar ordem de serviço para instalação de um meio de captura.

- No cadastro de estabelecimento, o Sistema Core só deve permitir acesso aos usuários da central que atendam ao perfil de célula de cadastro;

- Sempre que o cadastro de um estabelecimento for efetivado pelo Sistema Core fica sob sua responsabilidade o gerenciamento dos dados deste cliente;

- Todos os ativos da operadora de cartão de crédito devem ser gerenciados pelo SGA;

- Durante um cadastro de um novo estabelecimento o SGA deve solicitar um número de controle a Rede de Captura;

- Sempre que for solicitado novo número de controle a Rede de Captura deve validar as informações do estabelecimento;

- Nunca deve ser dado andamento em uma ordem de serviço de instalação se a Rede de Captura não retornar o número de controle para associar ao estabelecimento e ao equipamento;

- No processo de instalação física do equipamento a operadora de cartão de crédito pode terceirizar o serviço contratando uma EPS;

- Uma ordem de serviço só pode ser extraída pela EPS quando a mesma apresentar status em aberto;

- A EPS deve sempre solicitar a primeira liberação para a central de atendimento - célula de baixa para iniciar a instalação física do equipamento;

- Sempre que for requisitada a primeira liberação o operador da central de atendimento – célula de baixa deve utilizar o SGA para efetivar a liberação;

- A primeira liberação deve ser efetivada e confirmada pela central para que a EPS solicite a segunda liberação;

- Após confirmação da primeira liberação a EPS deve solicitar a segunda liberação a Rede de Captura através de transação;

- A Rede de Captura deve efetivar a segunda liberação do equipamento sempre que receber o script de transação que identifique essa ação;

- Um equipamento só pode ser efetivamente utilizado pelo cliente após a confirmação da segunda liberação; e

- Uma ordem de serviço deve sempre ser concluída pela EPS.

4.2.3 Derivação de Requisitos

Por fim é executada a atividade de derivação de requisitos através das atividades de raciocínio de dedução, abdução e indução, que só é possível a partir dos termos apresentados no modelo semântico quando analisados juntamente com o modelo de processo representado pelas atividades e os cenários que ilustram o processo de forma dinâmica.

A seguir são listados os requisitos obtidos a partir da análise destes artefatos gerados durante o processo e do conhecimento de domínio adquirido pela observação e estudo do negócio:

1. Para ter o meio de captura instalado o estabelecimento deve ter um cadastro na base do Sistema Core;

2. Somente usuários com perfil da célula de cadastro conseguem cadastrar novos estabelecimentos no Sistema Core;

3. A etapa final de um cadastro de estabelecimento bem sucedido no sistema core gera a demanda de solicitação de ordem de serviço para o SGA;

4. Para realizar a abertura de ordem de serviço de instalação o SGA deve receber as informações do estabelecimento do sistema core para realizar cadastro interno;

5. O cadastro interno do SGA corresponde ao cadastro do estabelecimento como cliente e o cadastro de um novo equipamento para ele;

6. Após realizar o cadastro em sua base o SGA deve solicitar a Rede de Captura um número de controle que relacionará o equipamento ao estabelecimento solicitante;

7. A Rede de Captura deve validar as informações do estabelecimento antes de gerar um número de controle para ele a partir das informações enviadas na solicitação de número de controle;

8. O número de controle deve ser um identificador único que servirá para associar o estabelecimento àquele equipamento;

9. Após receber o número de controle o SGA deve associá-lo ao estabelecimento e ao equipamento, e efetivar a abertura da ordem de serviço;

10. Durante o processo de abertura de ordem de serviço o estabelecimento e o equipamento cadastrado para ele devem ser associados à ordem de serviço pelo número de controle;

11. A EPS deve ter acesso ao SGA respeitando o nível de permissão referente às suas atividades dentro do sistema;

12. Para efetivar a instalação física do equipamento a EPS deve extrair a ordem de serviço do SGA;

13. Uma ordem de serviço só deve ser extraída para execução pela EPS quando estiver com status em aberto;

14. Para iniciar a instalação o operador da EPS deve requisitar a central de atendimento a primeira liberação do equipamento;

15. Somente usuários com perfil célula de baixa conseguem realizar a primeira baixa do equipamento no SGA;

16. A segunda baixa do equipamento só pode ser requisitada após a confirmação da primeira liberação do equipamento;

17. O usuário da central de atendimento deve acessar o SGA para efetivar a primeira liberação;

18. Após confirmação da primeira liberação o operador EPS deve solicitar a segunda liberação para conseguir efetivar a instalação do equipamento;

19. A segunda liberação é feita pela Rede de Captura a partir de um script de execução feita pelo operador da EPS diretamente no equipamento;

20. Após a confirmação da segunda liberação o equipamento está instalado e pronto para uso;

21. O operador da EPS deve acessar o SGA para concluir a ordem de serviço;

22. A conclusão da ordem de serviço poderá ser realizada manualmente por baixa ou de forma agrupada a partir de uma carga das ordens de serviços de equipamentos instalados; e

23. A conclusão da ordem de serviço deve sempre acontecer independentemente das efetivações das liberações e instalação física.

A validação dos requisitos descritos e até mesmo a complementação foi realizada com o suporte do usuário que identificou se os mesmos estavam aderentes à suas atividades do dia a dia.

4.3 Metodologia de Aplicação CelLEST

Esta seção aborda a aplicação do processo CelLEST no mesmo módulo de instalação de equipamento do SGA. Assim como mencionado no capítulo anterior, este processo igualmente faz uso da observação da interação entre usuário e sistema, porém busca os requisitos baseando-se nos *traces* dessa operação. Abaixo são descritas as etapas realizadas usando a abordagem CelLEST para a recuperação dos requisitos do SGA.

4.3.1 Engenharia Reversa

A engenharia reversa no processo CelLEST caracteriza-se pela criação de um mapa de interface, também conhecido por modelo de transição de estados, que representa a interação do usuário com o sistema. Nele é possível identificar quais as telas utilizadas durante a execução da tarefa e as ações necessárias nestas

interfaces para alcançar o objetivo final. Esta etapa é dividida em três atividades descritas a seguir.

4.3.1.1 Identificação das Telas

Os *traces* usados como entrada para a etapa de engenharia reversa são caracterizados pelas telas que representam as operações que o usuário realiza no sistema. No estudo para o SGA, a captura das telas foi feita manualmente durante a observação da interação entre usuário e sistema e serviram como produto para reconhecimento das interações.

É importante ressaltar que mesmo explorando o fluxo de instalação de equipamento, não foram consideradas atividades executadas em sistemas que fazem interface com o SGA, como por exemplo, a geração de número de controle para o equipamento que é feita pela Rede de Captura.

4.3.1.2 Reconhecimento das Interações

O reconhecimento das interações é realizado através de três atividades: extração de características das telas, identificação e agrupamento delas de acordo a similaridade entre as características e, por fim, a modelagem da tarefa que gera gráficos de estados que representam a execução de uma operação. Assim como feito pela ferramenta adotada no processo CelLEST, as telas foram numeradas de forma seqüencial e as atividades modeladas pela seqüência destes identificadores. A seguir, é apresentado o modelo de estados para o fluxo de ordem de serviço de instalação, que representa as atividades de gerenciamento de uma ordem de serviço dentro do SGA.

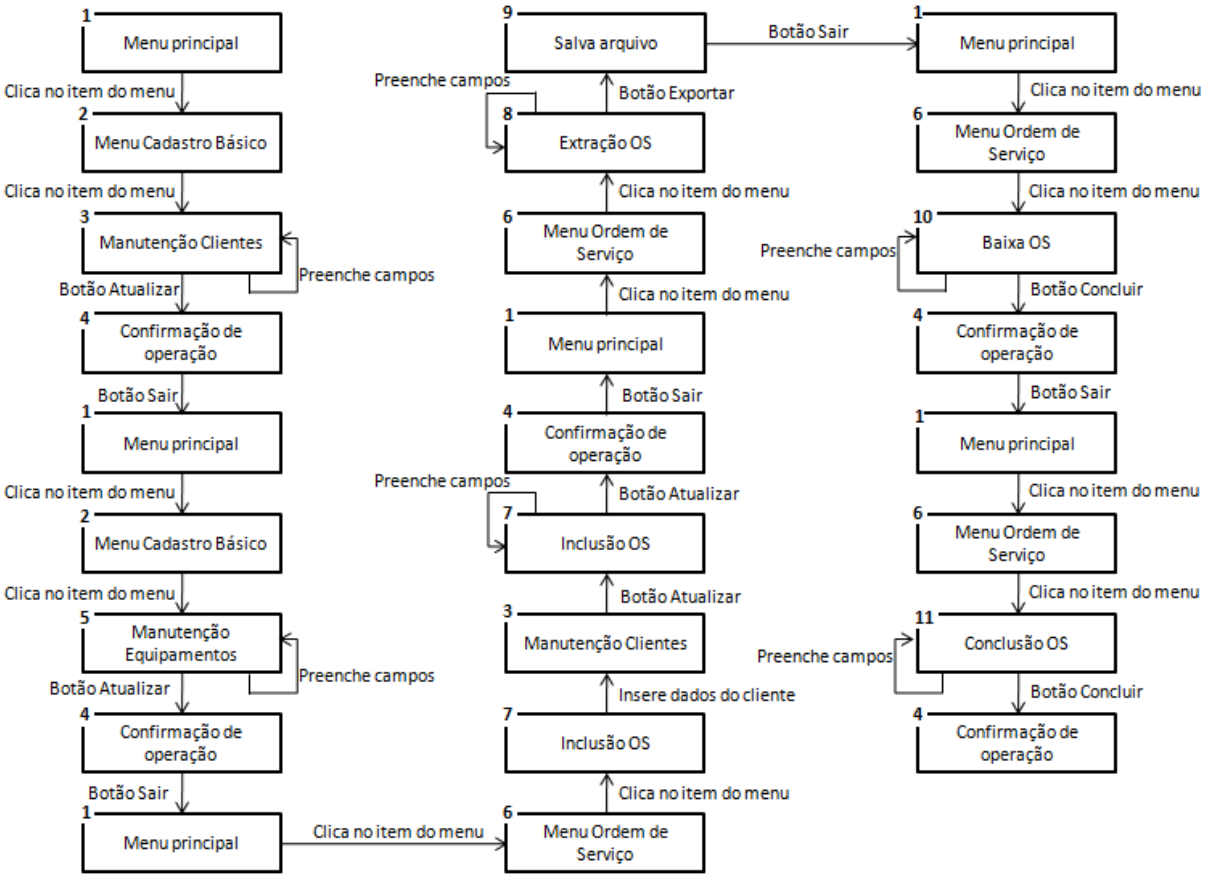

Figura 12. Modelo de transição de estados para o fluxo de ordem de serviço de instalação

Na ilustração acima é possível identificar as telas para a execução da tarefa, que estão representadas pelos retângulos e numeradas de acordo com o levantamento feio inicialmente para mapear as telas que compõem o sistema. Para o estudo em questão a transição entre os estados foi representada pela ação tomada pelo usuário em texto declarativo a fim de facilitar a compreensão do modelo.

4.3.1.3 Extração dos Padrões de Interação

Esta atividade tem como objetivo identificar o padrão de interação necessário para a execução de uma tarefa atendendo às necessidades do usuário, isto é factível quando são consideradas possibilidades de execução de uma mesma tarefa, pois a partir delas é traçado o melhor caminho para a realização da atividade.

Considerando esta premissa para o estudo do SGA foram capturados seis grupos de *traces* que representam a tarefa de instalar um novo meio de captura para um determinado estabelecimento, ou seja, representam toda a gestão de uma ordem

de serviço de instalação que vai desde o cadastro do estabelecimento até a conclusão da ordem de serviço, a Figura 12 representa um desses grupos de *traces*.

Assim como no processo CelLEST para a execução desta etapa do processo foi usado a ferramenta IPM, que por sua vez também é desmembrado em atividades-chaves:

4.3.1.4 Pré-processamento

O pré-processamento tem como objetivo eliminar repetições das seqüências de entrada resultantes do diversos acessos consecutivos a mesma tela, segundo o trabalho exposto por El-Ramly, Stroulia e Sorenson (2001) as repetições devem ser substituídas por uma contagem seguida pela identificação do item repetido, conforme exemplo: {4,5,6,6,6,6,6,6,7} → {4,5,(6)6,7}. Porém, não fica clara na abordagem como esta seqüência com o controle de repetição é tratada posteriormente, por este motivo para o estudo de caso foi adotada a seqüência de interação sem as repetições, uma vez que estas podem mascarar um candidato a padrão qualificado.

4.3.1.5 Descoberta e análise de padrões

A descoberta de padrões corresponde à atividade principal do algoritmo IPM, é nesta fase que ocorre uma busca exaustiva que identifica padrões que atendam ao critério definido pelo usuário, que engloba a quádrupla: tamanho mínimo que a seqüência deve ter (minLen), o limiar calculado para indicar o quanto a seqüência suporta um padrão (minSupp), o número máximo de erros permitidos na inserção (maxError) e a pontuação mínima para classificar os padrões descobertos (minScore).

Assim como mencionado a entrada do algoritmo foram seis seqüências representadas pelos identificadores das telas utilizadas para a execução da tarefa. A tabela abaixo lista a descrição das telas utilizadas, seus identificadores (ID) e a freqüência (Freq.) em que aparecem nos traces capturados.

Tabela 2. Identificadores, descrição e freqüências das telas do SGA

ID	Descrição	Freq.		ID	Descrição	Freq.
0	Menu principal	33		7	Extração OS	12
1	Menu Cadastros Básicos	09		8	Salva arquivo	07
2	Manutenção de clientes	37		9	Baixa OS	18
3	Confirmação da operação	27		10	Conclusão OS	04
4	Manutenção de equipamentos	14		11	Erro na operação	02
5	Menu ordem de serviço	24		12	Upload de baixa	09
6	Inclusão OS	13		13	Visualização do arquivo de baixa	04

As seqüências iniciais foram exploradas em duas fases, na primeira foram identificados todos os padrões candidatos de tamanho dois que atendiam ao critério, e na segunda fase, o algoritmo de forma recursiva estendeu os padrões candidatos definidos, avaliando se atendiam ao critério de entrada. Quando se esgotaram as possibilidades de gerar mais candidatos o algoritmo retornou os padrões qualificados.

Ao fim de cada execução foi realizada uma análise dos padrões qualificados juntamente com o usuário para identificar se os mesmos correspondiam às funcionalidades do sistema e se havia necessidade de ajustes no critério definido inicialmente para a descoberta de novos padrões.

No processo foram explorados oito critérios distintos em busca de um resultado gerenciável e compreensível. E dentre eles foi eleito o critério representado por c = (minLen, minSupp, maxError, minScore) = (5, 2, 2, 3), que buscava por um padrão composto por no mínimo cinco telas; que tivesse dois suportes, isto é, aparecesse pelo menos duas vezes no conjunto de seqüências utilizado como entrada; que suportasse no máximo dois erros de inserção na seqüência; e que entre o ranque de candidatos apresentasse pontuação mínima de três. O critério considerado produziu cinco padrões qualificados, e destes o usuário elegeu três que representam sua tarefa de maneira muito próxima.

4.3.2 Engenharia progressiva

A engenharia progressiva para o processo CelLEST é representada pela identificação da funcionalidade do sistema e modelagem de um *front-end* que atenda à migração do sistema. Para fins desse estudo não será aplicado o processo de

construção de *front-ends* para o sistema em questão. Esta decisão justifica-se no fato do escopo do trabalho se restringir a recuperação de requisitos.

Abaixo estão listados os requisitos identificados a partir dos padrões qualificados:

1. Um estabelecimento só pode ter uma ordem de serviço de instalação após ser cadastrado no SGA;

2. Todo estabelecimento para solicitar uma ordem de serviço deve ter cadastrado um equipamento;

3. Durante a abertura de uma ordem de serviço é necessária a associação do estabelecimento e do equipamento a esta ordem;

4. Após a abertura da ordem de serviço a mesma deve ser extraída para a instalação física do equipamento;

5. O processo de extração de ordem de serviço permite que o usuário exporte todas as ordens pendentes de instalação em um arquivo;

6. Durante a instalação do equipamento o mesmo deve ser liberado através da tela de Baixa de OS;

7. A conclusão da ordem de serviço pode ser feita através de um *upload* do arquivo com as ordens baixadas, para processamento batch;

8. A conclusão da ordem de serviço pode ser feita diretamente na tela de Conclusão de OS;

9. A conclusão da ordem de serviço não depende da baixa da mesma;

10. Todas as operações devem apresentar tela do status da realização da tarefa, se executada com sucesso ou não;

11. O menu de cadastro básicos deve fornecer acesso a telas de manutenção de cliente e equipamento;

12. Também deve ser possível cadastrar um novo equipamento a partir da tela de abertura de ordem de serviço; e

13. O menu de ordem de serviço deve fornecer acesso as telas de inclusão, extração, baixa e conclusão de ordem.

4.4 Estudo comparativo

O documento original de requisitos funcionais do sistema estudado abrangia 54 requisitos referentes ao módulo de solicitação de ordem de serviço de instalação. Como apresentado nas seções anteriores foram capturados 23 requisitos pelo processo AMBOLS e 13 requisitos pelo processo CelLEST.

Para estimar quão próximo os requisitos capturados estavam dos originais foi feita uma comparação entre cada grupo de requisitos com os requisitos do SGA, nessa comparação chegou-se aos percentuais de 65% e 84%, respectivamente. A discrepância entre os números obtidos justifica-se pelos focos distintos que os processos apresentam. Dessa forma, não é válido comparar esses percentuais e os mesmos serão tratados separadamente para avaliação dos processos.

Mesmo recuperando quase metade do número de requisitos do documento original o processo AMBOLS não se mostrou 100% aderente quando se usa o filtro de requisitos funcionais no universo de requisitos capturados. Esse baixo nível de aproximação explica-se no nível de detalhamento atingido se referir ao fluxo geral de solicitação incluindo, inclusive, informações referentes aos demais sistemas envolvidos na atividade.

A abrangência alcançada com o AMBOLS fundamenta-se no fato do processo fazer uso do conhecimento do usuário em nível mais aprofundado, diferentemente do processo CelLEST que se baseia somente em capturas de telas. Diante desta evidência nota-se que o conhecimento humano é um recurso valioso, mesmo que na avaliação nem todos os requisitos identificados a partir desse domínio fossem classificados como funcionais na documentação original.

Á nível de aplicação, o uso do AMBOLS mostrou-se mais didático e fácil, considerando, porém, que há necessidade que o analista domine técnicas de modelagem, uma vez que o processo baseia-se em modelos para descrever cada etapa executada.

O processo CelLEST, por sua vez, recuperou 13 requisitos, porém por fazer uso de telas específicas para o módulo testado o nível de aderências desses requisitos alcançou um percentual considerável, uma vez que em sua maioria tratavam de requisitos funcionais.

Apesar do alto nível de aderência o processo em questão requer mais habilidade técnica do analista que precisa implementar os procedimentos a fim de obter o resultado, mas vale ressaltar que uma vez vencida esta etapa o processo é automatizado requerendo apenas a calibragem do algoritmo de acordo com a análise dos padrões feita pelo usuário, isto é, o esforço de análise após implementação é pequeno.

Como mencionado, mesmo como um objetivo comum, recuperação de requisitos, e fazendo uso da interação do usuário com o sistema como ponto de análise, os processos divergem quanto ao foco, o primeiro caracteriza-se em entender a regra de negócio mapeando papéis e responsabilidades dos envolvidos, enquanto o segundo processo busca os requisitos funcionais de forma metódica, sem dar ênfase ao ambiente em qual o sistema opera ou para qual fim é executado no âmbito comercial.

Por fim, em ambos os processos fica evidente a necessidade de um usuário conhecedor do domínio e que tenha familiaridade com o sistema, este ator tem papel fundamental, pois sem o mesmo não seria possível, em nenhuma das abordagens, alcançar o objetivo de construir um documento com os requisitos funcionais do sistema.

Diante destas evidências, mesmo quando o produto fim é único, neste contexto, requisitos funcionais, é necessário ser criterioso no processo adotado para a extração dos mesmos. Muitas vezes a manutenção do sistema requer muito mais que uma simples análise técnica, pois o sistema em operação possui uma bagagem de regras de negócio que só estão expressas nele e que precisam ser analisadas

para que as novas alterações não acarretem impacto nessas que são de certa forma o alicerce da organização.

Porém, não é difícil deparar-se com a necessidade de alterações emergenciais e pontuais que requerem conhecimento técnico do que o sistema faz e o porquê. Geralmente este tipo de atividade faz parte do dia a dia das organizações e atendendo às exigências do mercado sempre são realizadas com urgência, inviabilizando a utilização de processos morosos que levam tempo e tem custo elevado para a organização.

O estudo dos processos mostra que ambos são aplicáveis para a recuperação de requisitos, porém ponderando-se qual é o objetivo de obter a funcionalidade do software, isto é, qual tipo de manutenção a ser aplicada e sua finalidade. Uma vez que determinada essa premissa torna-se mais fácil a opção por um ou outro processo, sempre considerando os recursos disponíveis e variáveis de custo e prazo para atender a demanda de compreensão do sistema.

4.5 Considerações do Capítulo

Neste capítulo foi apresentado um sistema de gerenciamento de ativos de uma operadora de cartão de crédito que serviu como base para a aplicação dos processos AMBOLS e CelLEST.

Com foco em recuperação de requisitos a escolha por estes processos justifica-se no fato de usarem princípios da recuperação *top-down* que faz uso de uma abstração de alto nível para chegar a um detalhamento do sistema. Outro ponto em comum dos processos é que ambos apóiam-se na observação da interação do usuário com o sistema como ponto de partida para a compreensão do mesmo.

O primeiro processo aplicado faz uso dos princípios teóricos da semiótica organizacional a partir da aplicação dos métodos de análise semântica e de normas, já o segundo fundamenta-se no problema de mineração de padrões seqüenciais para obter o resultado almejado.

Após a aplicação dos processos foi feito um estudo comparativo dos resultados obtidos que sintetizou o quão próximo cada processo chegou aos requisitos reais do

sistema. A partir dessa análise também foi ilustrado as dificuldades e vantagens da aplicação dos processos no contexto apresentado.

5. CONSIDERAÇÕES FINAIS

A adoção de processos para a recuperação de requisitos pode ser eficaz dentro da manutenção de software, uma vez que contribui para uma das mais importantes atividades dentro do ciclo de manutenção, a compreensão do software.

Diversos trabalhos na literatura abordam processos de Engenharia Reversa que buscam otimizar a recuperação das funções de um sistema legado a fim de amenizar os esforços gastos nesta atividade, visto que é fundamental para determinação do escopo da manutenção e seus impactos no sistema como um todo.

A escolha por qual processo usar deve ser feita de forma criteriosa considerando, custo, prazo e os recursos disponíveis para a análise, atentando-se sempre ao fato de não comprometer a qualidade da manutenção em pró da urgência de uma exigência do mercado.

5.1 Contribuições

Este trabalho agregou informações sobre o cenário da manutenção, os desafios enfrentados nessa atividade e alguns processos que podem facilitar a compreensão do sistema legado objeto da manutenção.

Focando em processos que não fazem uso de qualquer artefato além do executável e que têm como ponto de entrada a interação com usuário, foram estudados dois processos em um nível de detalhamento que possibilitasse a aplicação dos mesmos.

A aplicação dos processos evidenciou que cada processo tem particularidades que podem satisfazer a diversos tipos de manutenção, é indispensável entender a necessidade da manutenção considerando os recursos disponíveis e as variáveis de prazo e custo que são essenciais no contexto da atividade de manutenção.

5.2 Trabalhos Futuros

Como trabalhos futuros, poderia ser proposto um processo híbrido que viabilizasse a recuperação de requisitos de forma equilibrada, considerando tanto ao

lado técnico, capturado pelo estudo da interface, quanto o lado amplo da execução do processo de negócio, obtido do conhecimento de domínio do usuário do sistema.

Este estudo apresentaria a vantagem de associar o conhecimento humano obtido a partir do usuário a procedimentos de análise dos módulos do sistema, incluindo interface e código fonte, quando disponíveis, para recuperar as funções do software legado. A intenção de compreender o sistema para identificar o ponto ser adequado às novas regras de negócio, quais os impactos dessa adequação e como testar os componentes alterados ou não justifica o benefício dessa proposta.

REFERÊNCIAS

ARNOLD, R. S. **A Road Map to Software Re-engineering Technology**. Software Reengineering - a tutorial, IEEE Computer Society Press, Los Alamitos, CA, 1993, pp.3-22.

CANFORA, G.; CIMITILE A. **Software Maintenance**. In Proc. 7th Int. Conf. Software Engineering and Knowledge Engineering, 2000.
Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.25.1678>
Acessado em: 05/10/2010

CANFORA, G.; DI PENTA, M. **Frontiers of Reverse Engineering: a Conceptual Model**. In Int. Conf. on Software Maintenance, IEEE Computer Soc. Press., 2008.
Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.145.1773>
Acessado em: 30/11/2010

CHIKOFSKY, E.; CROSS, J. J. **Reverse engineering and design recovery: A taxonomy**. IEEE Software 7, 1, 1990
Disponível em: <http://portal.acm.org/citation.cfm?id=624902>
Acessado em: 30/11/2010

COHEN, W. **Recovering Software Specifications with Inductive Logic Programming**. In Proc. 12th National Conf. on Artificial Intelligence, AAAI Press, 1994, vol. 1, 142-148.

DI LUCCA, G.; FASOLINO, A.; DE CARLINI, U. **Recovering Use Case Models from Object-oriented Code: a Threadbased Approach**. In Proc. 7th Working Conf. on Rev. Eng., 2000, IEEE Computer Soc. Press,pp.108-117.

ESPINDOLA, R. S.; MAJDENBAUM, A.; AUDY, J. L. N. **Uma Análise Crítica dos Desafios para Engenharia de Requisitos em Manutenção de Software**. In Workshop em Engenharia de Requisitos, 2004.
Disponível em:
<http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER04/Rodrigo_Espindola.pdf>
Acessado em: 16/08/2010

EL-RAMLY, M.; STROULIA, E., SORENSON, P. **Recovering Software Requirements from System-user Interaction Traces**. In Proc. 8th Working Conf. on Rev. Eng., IEEE Computer Soc. Press, Oct. 2001, 208-217

HARANDI, M. T.; NING, J. Q. **Knowledge-Based Program Analysis**. IEEE Software v. 7, n. 1, 1990, p. 74-81.

ISO 1219 - INTERNATIONAL STANDARD FOR SOFTWARE MAINTENANCE - IEEE Std 1219 - 1998. Pg. 34 – 38
Disponível em: <http://ieeexplore.ieee.org/servlet/opac?punumber=5832>
Acessado em: 19/06/2010

INTERNATIONAL STANDARD - ISO/IEC 12207 – IEEE Std 12207-2008. Systems and software engineering — Software life cycle processes. Second edition 2008-02-01. Pg. 53 – 55
Disponível em: <http://ieeexplore.ieee.org/servlet/opac?punumber=4475822>
Acessado em: 02/11/2010

JACOBSON, I., **Object Oriented Development in an Industrial Environment**.In: ACM SIGPLAN Intl. Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'87), pp. 183-191, Orlando, Florida, USA, October, 1987

JACOBSON, I.; LINDSTRÖM, F. **Re-engineering of old systems to an object-oriented architecture**. In Conference proceedings on Object-oriented programming systems, languages, and applications (OOPSLA '91), New York, NY, USA, 1991
Disponível em: <http://www.cs.uccs.edu/~chamillard/cs534/Jacobson91.pdf>
Acessado em: 13/12/2010

LEHMAN, M. M.; **Programs, cities, students – limits to growth?**, In: Imperial College of Science Technology, London, 1974
Disponível em: <http://www.mannymlehman.com/articles/mml68.pdf>
Acessado em: 14/11/2010

LIU, K.; ALDERSON, A.; QURESHI, Z. **Requirements Recovery from Legacy Systems by Analysing and Modelling Behaviour**. 1999. In Proc. Int. Conf. on Software Maintenance. IEEE Computer Soc. Press.
Disponível em: <http://www.cs.uccs.edu/~chamillard/cs534/Liu.pdf>
Acessado em: 06/07/2010

LIU, K. **Requirements Reengineering from Legacy Information Systems Using Semiotic Techniques**. 2005 In **Systems, Signs & Actions -** An International Journal on Communication, Information Technology and Work. Vol. 1 (2005), No. 1, pp. 38–61

LIU, K.; ALDERSON, A.; SHAH, H.; SHARP, B; DIX, A. **Applying Semiotic Methods to Requirements Recovery**, 1998. In: 6th International Conference on Information Systems Methodologies.

LIU, K.; ALDERSON, A.; SHARP, B.; SHAH, H.; DIX, A. **Using Semiotic Techniques to Derive requirements from Legacy Systems**. 1998
Disponível em: <www.ais.reading.ac.uk>
Acesso em: 20/12/2010

PADUELLI, M. M.; SANCHES, R. **Problemas em manutenção de software: caracterização e evolução**. III Workshop de Manutenção de Software Moderna – Brasil – 2006.

PADUELLI, M. M. **Manutenção de Software: problemas típicos e diretrizes para uma disciplina específica**. ICMC – USP – São Carlos, 2007.
Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-21062007-154606/pt-br.php>

Acessado em: 19/10/2010

PIEKARSKI, A. E. T.; QUINÁIA, M. A. **Reengenharia de software: o que, por quê e como**. UNICENTRO – Paraná – Brasil, 2000.
Disponível em:
<http://www.unicentro.br/editora/revistas/recen/v1n2/Reengenharia.pdf>
Acessado em: 30/11/2010

PEIRCE, C. S. Collected Papers of C. S. Peirce 1931-58, In Hartshorne, C. and Weiss, P. (eds.), Harvard University Press Cambridge, Mass, 1960

PRESSMAN, R. S. **Engenharia de Software**. 6ª Ed.- Mc Graw Hill - 2006

SEACORD, R. C.; PLAKOSH, D.; LEWIS, G. A. **Modernizing legacy systems: software technologies, engineering processes, and business practices**. SEI Series in Software Engineering. Addison Wesley, 2003.
Disponível em: <http://flylib.com/books/en/3.52.1.1/1/>
Acessado em: 19/10/2010

STROULIA, E.; EL-RAMLY, M.; KONG, L.; SORENSON, P., **Reverse Engineering Legacy Interfaces: An Interaction-Driven Approach**. In Proceedings of the Sixth Working Conference on Reverse Engineering (WCRE '99). IEEE Computer Society, Washington, DC, USA, 292-. 1999

TILLEY, S. ***A Reverse-Engineering Environment Framework***, Technical Report CMU/SEI-98-TR-005, Software Engineering Institute, Carnegie Mellon University, 1998
Disponível em: <http://www.sei.cmu.edu/reports/98tr005.pdf>
Acessado em: 25/01/2011

UML. ***Unified Modeling Language*** - UML 2.0, 2004
Disponível em: <http://www.uml.org/>

WARD, M. P.; Bennet, K. H. ***Formal Methods for Legacy Systems***. Journal of Software Maintenance: Research and Practice, 1995.

WEIDERMAN, N., BERGEY, J., SMITH, D., DENNIS, B. and TILLEY, S.***Approaches to Legacy System Evolution***, Technical Report CMU/SEI-97-TR-014, Software Engineering Institute, Carnegie Mellon University, 1997
Disponível em: <http://www.sei.cmu.edu/reports/97tr014.pdf>
Acessado em: 26/01/2011