PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

ALEXANDRE ROBERTO DOS PASSOS

MODELAGEM E ANÁLISE DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE UTILIZANDO REDES DE PETRI

Curitiba
Junho, 2008

ALEXANDRE ROBERTO DOS PASSOS

MODELAGEM E ANÁLISE DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE UTILIZANDO REDES DE PETRI

Dissertação apresentada ao programa de Pós-Graduação em Engenharia de Produção e Sistemas - PPGEPS do Centro de Ciências Exatas e de Tecnologia – CCET / PUCPR, como requisito parcial para obtenção ao título de mestre.

Orientadores:
Eduardo Alves Portela Santos
Marco Antonio Busetti de Paula

Curitiba
Junho, 2008

Passos, Alexandre Roberto dos
P289m Modelagem e análise do processo de desenvolvimento de software
2008 utilizando redes de Petri / Alexandre Roberto dos Passos ; orientadores, Eduardo Alves Portela Santos, Marco Antonio Busetti de Paula. – 2008.
99 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2008
Bibliografia: f. 92-98

1. Software - Desenvolvimento. 2. Petri, Redes de. I. Santos, Eduardo Alves Portela. II. Paula, Marco Antonio Busetti de. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Engenharia de Pordução e Sistemas. IV. Título.

CDD 20. ed. – 005

A história do homem livre nunca é escrita pela sorte, mas pela escolha – a escolha dele.

Eisenhower

AGRADECIMENTOS

Ao meu amado e generoso DEUS, que faz de sua luz um pouco minha, que me abre as portas até então desconhecidas, fazendo com que meu caminho pela incalculável trajetória da vida seja passível de ser percorrida com honradez, caráter e dignidade.

À minha amada e afável esposa Manuela, que sempre me apóia e incentiva ao alcance de meus objetivos.

À minha querida mãe Gessi, que envolta às dificuldades, conseguiu, com muito caráter e dignidade, dar seguimento em nossas vidas.

Ao Prof. Portela, que com seu discernimento, seu incentivo e sua ajuda esclarecida, me conduziu dos conceitos fundamentais à conclusão deste trabalho.

Ao Prof. Busetti, que com seu pensamento reflexível e congruente, auxiliou significativamente para a boa definição e conclusão deste trabalho.

Ao Prof. Loures, que compartilhou seu admirável conhecimento para o alinhamento de importantes conceitos deste trabalho.

À empresa Desbravador, que incentivou meu estudo neste mestrado, partilhando de sua base de conhecimento para a elaboração desta dissertação.

A todos que, direta ou indiretamente, me auxiliaram no transcorrer de meus estudos.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

O mercado de *software* está a cada dia se aprimorando, trazendo consigo novas ferramentas e novas formas de se administrar empresas. Com foco centrado no Processo de Desenvolvimento de *Software* (PDS), principalmente no que tange a sua estrutura e ao seu fluxo funcional, observam-se lacunas, que tendem a retardar e a prejudicar a liberação do *software*. Tais lacunas podem ser visualizadas sob distintas visões, tanto na parte organizacional do PDS, quanto na forma dos resultados extraídos do PDS, considerando benefícios e tempo de processamento.

No atual contexto administrativo em que as empresas vivem, Silva (2001) forma um conceito sobre administração que se aplica bem à necessidade de melhorias sobre o processo de desenvolvimento de *software* nas organizações: "Administração é um conjunto de atividades dirigidas à utilização eficiente e eficaz dos recursos, no sentido de alcançar um ou mais objetivos ou metas organizacionais".

Assim sendo, diante da demanda crescente por sistemas informatizados (*softwares*), observa-se que o processo de desenvolvimento de *software* pode ser fator decisivo para que as organizações consigam atender a tal demanda, sendo que a qualidade do produto seja tratada, permitindo obter certa vantagem competitiva junto à concorrência.

Estudos têm focado diretamente sobre o processo de desenvolvimento de *software*, buscando gerenciá-lo (Bezzerra, 2004) e melhorá-lo. A abordagem de gestão do conhecimento é apresentada por Parreiras et al. (2004) como um mecanismo para melhorar o PDS. Kantorski & Kroth (2004) apresentam fatores considerados importantes, como medição de atividades e o gerenciamento e relacionamento entre workflows, que devem ser decisivos na melhoria de processos de desenvolvimento de *software*. Da mesma forma, ainda em busca por melhorias sobre o PDS, Tamaki & Hirama (2007) buscam demonstrar também como é possível projetar e implantar melhorias sobre um processo existente, aplicando *process patterns*.

Desse modo, constata-se na literatura que esforços vêm sendo dispensados em busca de melhorias no PDS. Por outro lado, ainda existem lacunas, relacionadas ao tempo e à qualidade de processamento das atividades do PDS, que podem ser tratadas objetivando o aperfeiçoamento no PDS.

O PDS é considerado como sendo um processo de negócio, no qual suas atividades estão direcionadas para processar requisitos, tendo como resultado o produto em forma de *software*. Dessa forma, podemos observar que o PDS também precisa ser estudado na busca de melhores resultados. Tal estudo necessita um nível de detalhamento que pode ser encontrado através da modelagem e de sua análise. A modelagem possibilita representar graficamente como o PDS está estruturado em suas atividades. A análise possibilita que possíveis problemas possam ser encontrados no PDS, facilitando sua resolução. Buscando obter dados para representar um ambiente real de PDS, este trabalho empregará o formalismo de redes de *petri* (RdP) para modelar e analisar o processo.

Dentre os trabalhos existentes com RdP, encontra-se o de Aalst et al. (1994), que demonstra como workflows podem ser modelados utilizando RdP. Pádua et al. (2004) corrobora com a modelagem de workflows de processos de negócio em RdP.

Demonstrando como modelar em RdP, Jiao & Cheung (2006) apresentam como as propriedades de integridade (*soundness*) podem ser preservadas ao realizar a composição e o refinamento de atividades em uma modelagem.

A importância de ferramentas computacionais para o emprego de redes de petri, assim como as principais características de edição, modelagem e análise, são fatores abordados por Suraj et al. (2006).

Buscando avaliar processos de análise de sistemas, Kawabata et al. (2005) demonstra que várias visualizações podem ser obtidas de um determinado sistema, através da representação gráfica em redes de *petri*.

Da mesma forma, mas focando na educação de engenharia de software e na análise de sistemas, Kawabata & Itoh (2006) apresentam a colaboração que as redes de *petri* podem proporcionar através da representação visual (modelagem em RdP), o que facilita a identificação de distintos componentes do *software*.

Uma representação abstrata das atividades do *software* é realizada por Schmerl et. al. (2006), buscando demonstrar em alto nível, as operações que o *software* (baixo nível) contempla, permitindo assim, análises com maior precisão e eficácia sobre o *software*.

Assim sendo, este trabalho buscará demonstrar como uma ferramenta formal pode auxiliar na modelagem e na análise de processos de desenvolvimento de *software*. Não pretende-se aqui demonstrar ferramentas de baixo nível, utilizadas para modelagem ou desenvolvimento de *software* (ex.: UML, Delphi, C++, Java), tampouco o ciclo do *software*. Pretende-se sim, demonstrar como uma ferramenta formal pode ser utilizada para modelagem e análise de **processos de desenvolvimento de *software***, a qual tende a ser útil para aprimorar o fluxo das atividades e da alocação de recursos em uma organização que desenvolve software.

Dessa forma, visto a grande aplicabilidade da ferramenta de redes de *petri* para modelagem e análise de processos, pode-se objetivar seu emprego também em processo de desenvolvimento de *software*.

Assim sendo, tendo em vista a grande necessidade por processos de desenvolvimento de *software* bem estruturados e funcionais, este trabalho pretende explorar o formalismo de “redes de *Petri*” como meio propiciador para efetuar a modelagem e a analise do **processo de desenvolvimento de *software*** de uma organização real. A modelagem e a análise realizadas em “redes de *Petri*” serão utilizadas para propor melhorias para o PDS.

1.1. Problema

Empresas prestadoras de serviço que desenvolvem e personalizam seus *softwares* estão em constante aprimoramento para atender às necessidades de seus clientes. Em busca crescente pela obtenção de vantagem competitiva em seu mercado de atuação, acabam direcionando atenção para seus processos internos, os quais, muitas vezes, são defeituosos e precários.

O principal problema observado para propor este trabalho está relacionado à ineficácia no processo de desenvolvimento de *software*. Esta ineficácia pode ser

melhor caracterizada pela demora na liberação de versões do *software* no processo de desenvolvimento.

Desse modo, este trabalho busca propor uma contribuição para as organizações que necessitam aprimorar seus processos de desenvolvimento de *software*. Neste contexto, este trabalho apresentará como uma ferramenta formal, denominada "Redes de *Petri*", pode auxiliar na modelagem e na análise de processos de desenvolvimento de *software*.

Assim sendo, busca-se avaliar, sob uma abordagem formal, o processo de desenvolvimento de *software* como sendo o tema problemático a ser tratado. Dessa forma, surge a seguinte questão:

Como a ferramenta "Redes de *Petri*" pode auxiliar a propor novos cenários para o processo de desenvolvimento de *software*?

1.2. Objetivos

Os objetivos são aqui apresentados como "Objetivo Geral" e como "Objetivos Específicos", como segue nos tópicos 1.2.1 e 1.2.2.

1.2.1. Objetivo Geral

Propor novos cenários para o processo de desenvolvimento de software através da modelagem e da análise utilizando o formalismo de redes de *Petri*.

1.2.2. Objetivos Específicos

- Estudar a estrutura e o fluxo do processo de desenvolvimento de *software*, de uma organização produtora de *software* real;
- Aplicar o formalismo de redes de *Petri* para desenvolver a modelagem e a análise do processo de desenvolvimento de *software*;

- Propor novos cenários, com base na modelagem e na análise, ao processo de desenvolvimento de *software* de uma organização real.

1.3. Justificativas

Devido à grande quantidade de requisições por melhorias e personalizações que um dado sistema de informação (*software*) venha a receber, uma maior atenção deve ser direcionada para o processo ao qual irá atender estas requisições, desde a entrada da requisição até sua conclusão.

Dependendo do nível de abrangência que um processo de desenvolvimento de *software* venha a ter, este pode se tornar completamente mutável e adaptável às necessidades provindas da demanda. Tal dinamismo se torna indispensável em ambientes que estão constantemente se aprimorando, buscando melhorias e personalizando suas rotinas. Nesta questão, a flexibilidade tende a melhorar significativamente a capacidade de produção e aprimoramento de um dado sistema de informação, o que a torna uma característica fundamental (Wadhwa et al, 2005).

Desse modo, um ambiente constantemente mutável como a gestão de negócios requer aparatos tecnológicos flexíveis e adaptáveis, que sejam hábeis para acompanhar suas mudanças. Em se tratando de *software*, a diferença competitiva estará proporcionalmente interligada à capacidade de atender em tempo hábil e com qualidade aos requisitos que forem surgindo no ambiente de atuação. Assim sendo, este trabalho pretende demonstrar como o formalismo de "redes de *Petri*" pode auxiliar na modelagem e na análise de processos de desenvolvimento de *software* para a proposição de cenários.

1.4. Estudo de Caso

Como estratégia de pesquisa utilizar-se-á o **estudo de caso**, aplicando-o sobre o **processo de desenvolvimento de *software* (PDS)**. Devido aos objetivos propostos, o estudo deve ser direcionado em face a propiciar a modelagem, a análise e a proposição de novos cenários sobre o PDS.

Tendo foco no estudo de um PDS, pode-se avaliar que o mesmo pode conter atividades refinadas, que se caracterizam por **unidades incorporadas** (Yin, 2005). Da mesma forma, pode-se verificar que o PDS é considerado um caso típico, por ser encontrado em distintas organizações. Assim sendo, pode-se qualificá-lo de **caso único** (Yin, 2005). Dessa forma, este trabalho se caracteriza por projeto incorporado de caso único. A figura 1 ilustra estas características.

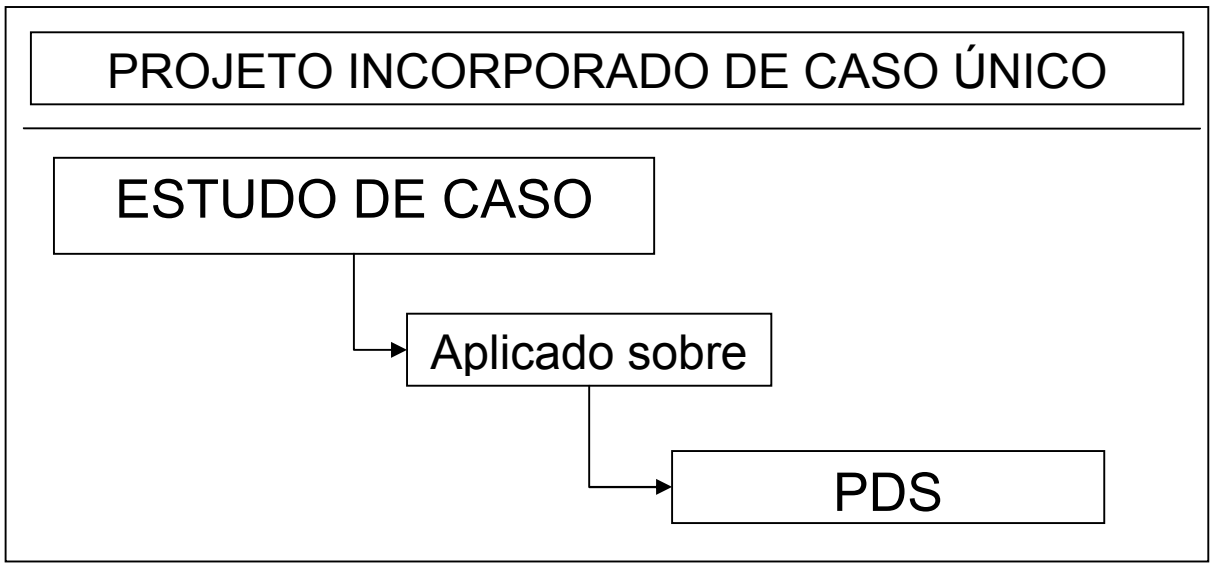

Figura 1: Caracterização da metodologia do projeto
Fonte: Autor.

1.5. Estrutura do Trabalho

A presente dissertação está estruturada da seguinte forma: no capítulo 2 são apresentados os fundamentos teóricos que norteiam o projeto de pesquisa proposto. Este capítulo apresenta duas abordagens principais:

1. Processos: abrange *Workflow*, Processos de negócios, Processos de desenvolvimento de *software*, Modelos de processos de desenvolvimento de *software*, Engenharia de requisitos e Trabalhos com abordagem em PDS ;
2. Formalismo: abrange as categorias Redes de *Petri* (RdP), Análise de processos em Modelos de RdP e Trabalhos com RdP;

Inicialmente, no capítulo 2, são apresentados conceitos de *workflow*, de processos de negócio, de processos de desenvolvimento de *software*, assim como são descritos os principais modelos de processos de desenvolvimento de *software* e características de engenharia de requisitos. Em seguida, é apresentado o formalismo utilizado: as redes de *Petri*. Tal formalismo é apresentado visando utilizá-lo na modelagem conceitual dos processos de desenvolvimento de *software*. Na seqüência são apresentadas algumas técnicas de análise disponíveis em redes de *Petri*. Para finalizar este capítulo, alguns trabalhos com abordagem em processo de desenvolvimento de *software*, e alguns trabalhos que utilizam RdP, são apresentados.

No capítulo 3, é apresentada a estrutura de como se pretende realizar este trabalho, através de uma proposição metodológica. Em seguida, no capítulo 4, apresenta-se o desenvolvimento do trabalho com o estudo do cenário atual da empresa Desbravador Automação Hoteleira, para então, na seqüência, efetuar a modelagem e a análise do cenário atual.

No capítulo 5 é proposto um novo cenário, embasado na análise realizada no capítulo 5. Na seqüência, é efetuada a modelagem e a análise sobre este novo cenário, realizando-se um comparativo entre o cenário atual e o cenário proposto.

Por fim, no capítulo 6, a conclusão sobre todo o trabalho é apresentada, fazendo-se um breve resumo do conteúdo deste, assim como, uma avaliação sobre o alcance dos objetivos propostos. As limitações deste trabalho e perspectivas de trabalhos futuros também são apresentadas.

2. REVISÃO DE LITERATURA

Este capítulo está dividido em duas grandes áreas: processos e formalismo. A primeira área é composta por conceitos de *workflow*, de processos de desenvolvimento de *software*, assim como dos principais modelos de processos de desenvolvimento de *software* e de engenharia de requisitos. A segunda parte é composta pelo formalismo que se pretende empregar através de conceitos de redes de *petri*. Tal formalismo é apresentado visando utilizá-lo na modelagem conceitual modular dos processos de desenvolvimento de *software,* e em algumas técnicas de análise disponíveis em redes de *Petri*. Para finalizar este capítulo, são apresentados alguns trabalhos que contemplam estas duas abordagens (processos e formalismo). São apresentados trabalhos sobre PDS e sobre redes de *Petri* para modelagem e análise de processos de negócio.

A figura 2 ilustra como este capítulo está estruturado.

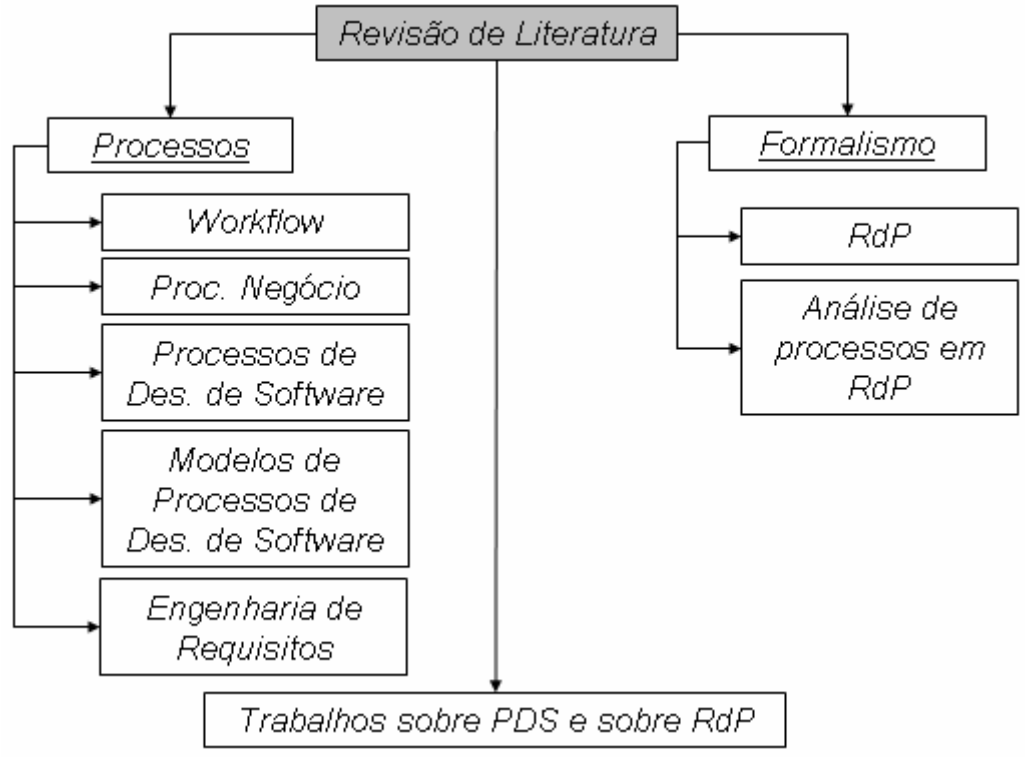

Figura 2: Estrutura da revisão de literatura
Fonte: Autor.

2.1. Workflow

De acordo com o WFMC (*Workflow Management Coalition*) (WFMC, 1999), para o termo *workflow*, caracteriza este como a “automação de um processo de negócio, todo ou em parte, no qual documentos, informações e tarefas são passados de um participante para outro através de ações, de acordo com um conjunto de regras procedimentais”.

Assim sendo, *workflow* é um conjunto de etapas e de trabalhos (parcialmente) ordenadas. Um processo representa a execução de uma série de atividades, que estão de acordo com um dado procedimento. Um procedimento, por sua vez, é um conjunto ordenado de atividades de controle e (sub) procedimentos com conjuntos de classes de recursos. O controle de atividade tem por função especificar a rota das atividades no procedimento, assim como, à sincronização do trabalho no mesmo (Aalst et al. 1994).

Conforme os autores, um trabalho é uma parte da atividade que deve ser executada por um ou mais recursos em um intervalo de tempo pré-determinado. Um recurso pode executar um trabalho isoladamente ou em conjunto com outros recursos. Uma classe de recursos é um conjunto de recursos. Um documento pode ser utilizado como uma entrada ou uma saída de um trabalho.

Para tornar possível modelar um *workflow* de forma a obter um modelo conciso e formal, que possibilite uma melhor análise, far-se-á o uso da ferramenta redes de *Petri*, que possibilita efetuar um modelo gráfico, contendo uma semântica formal e ainda ter a vantagem de possuir técnicas de análise e ferramentas que possibilitam analisar processos de *workflow* (Aalst, 1996; Aalst,1998).

2.2. Processos de Negócio

Processo de negócio é caracterizado como um conjunto de procedimentos ou atividades que, respeitando as regras, a estrutura e as funções empresariais de

determinada organização, são realizadas para se alcançar o objetivo do negócio (WFMC, 1999).

Neste contexto, observa-se que um sistema *workflow* pode gerenciar o fluxo de trabalho e organizar a rota dos dados de um determinado processo através de recursos humanos e de aplicativos automatizados. Todavia, para poder gerenciar distintos *workflows*, de distintos processos, faz-se necessário adotar um Sistema Gerenciador de *Workflows* (WFMS). Tal sistema, por ter uma série de funções que podem ser usadas para definir e graficamente direcionar *workflows*, torna a análise e a revisão de um *workflow*, com seu fluxo, clara e mais fácil de se efetuar (Aalst et al., 2002).

2.3. Processos de Desenvolvimento de *Software*

O processo de desenvolvimento de *software* é contemplado pela área de engenharia de *software* como sendo composto por três grandes fases: definição, desenvolvimento e manutenção. Estas fases estão contidas em, basicamente, qualquer processo de desenvolvimento de *software*. Um processo de desenvolvimento de *software* pode ser caracterizado como um conjunto de processos compartilhados, o qual contém um número de atividades que são aplicados para todos os projetos de *software*. As atividades são compostas por um número de conjunto de tarefas (trabalhos) que viabilizam a adaptação das atividades a qualquer característica do projeto de *software*, assim como, aos requisitos provindos da equipe de projeto (Pressman, 1997).

A figura 3 abaixo retrata o exposto acima:

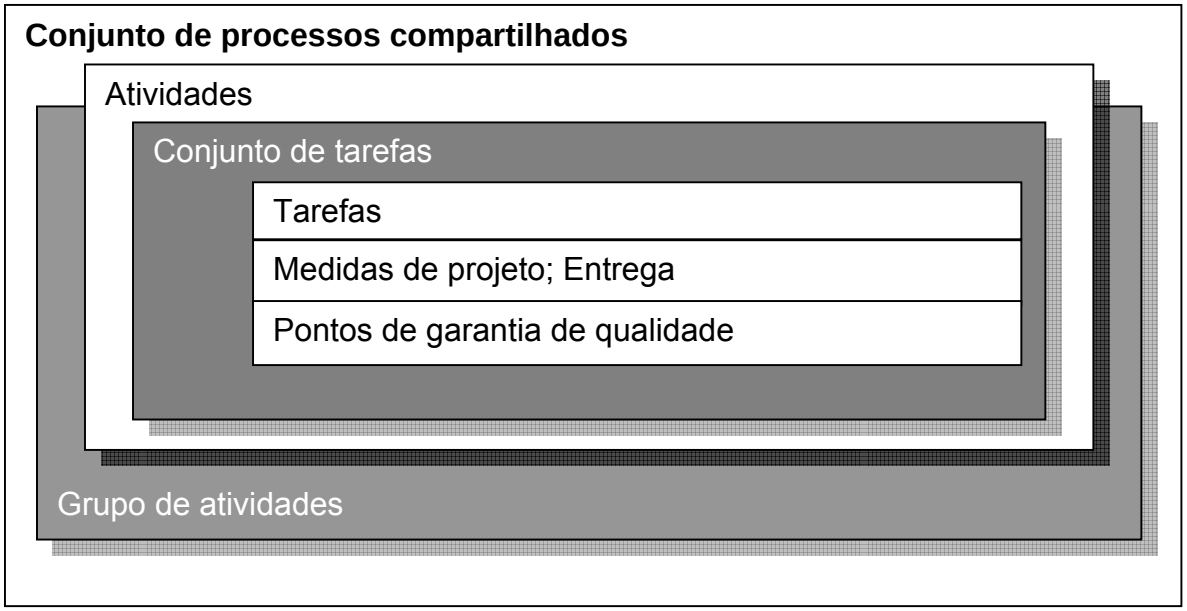

Figura 3: Conjunto de processos de desenvolvimento de *software*
Fonte: Pressman, 1997.

2.4. Modelos de Processos de Desenvolvimento de *Software*

Sendo parte integrante da engenharia de *software*, os processos de desenvolvimento de *software* podem ser caracterizados como roteiros de elaboração de *software*, que devem contemplar fases, subfases, produtos externados e pontos de avaliação de qualidade. Neste contexto, processos podem ser definidos para atividades como projeto, desenvolvimento e manutenção de *software*. Da mesma forma, sub-processos podem ser definidos para estas atividades, como análise de requisitos funcionais, programação e testes (Rezende, 2002).

Não existe um roteiro único (modelo), ou considerado correto, para se aplicar em um ambiente de desenvolvimento de *software*. Empresas e seus processos diferem em tamanho em habilidades de seus recursos, o que requer muitas vezes adaptações para poder suprir as necessidades (Rezende, 2002). Assim sendo, mesmo que não existam modelos únicos a serem aplicados aos processos, vários modelos podem ser encontrados na literatura para definir os roteiros dos processos de desenvolvimento de *software*. Os principais são (Pressman, 1997):

- Seqüencial linear: Também conhecido como modelo cascata ou modelo clássico. Basicamente é constituído, seqüencialmente, pelas etapas de análise, projeto, codificação e teste;
- Prototipação: Tem como filosofia o desenvolvimento de acordo com os requisitos do cliente. Tão logo esteja pronto, o *software* é apresentado para o cliente para que o mesmo teste e valide o protótipo. Dessa maneira o *software* toma forma de maneira iterativa;
- RAD (*Rapid Application Development*): Modelo no qual o ciclo de desenvolvimento tem um período de tempo considerado curto. É considerado um modelo seqüencial linear de alta velocidade, onde o período de tempo de seu desenvolvimento pode variar em torno de 90 dias;
- Evolucionário: Modelo que tende a atender um crescimento no *software* ao decorrer do tempo, de forma iterativa. Tem em seu escopo uma sub-classificação, da qual pode-se referenciar como elementos principais:
    - Incremental: Combina os modelos “seqüencial linear” e “prototipação”. A cada linha de desenvolvimento linear, uma iteração é efetuada, e como resultado é gerado um incremento a ser liberado para o cliente;
    - Espiral: Modelo que utiliza a filosofia iterativa do modelo de “prototipação” unido com características do “seqüencial linear”. Cada incremento de protótipo é composto por seis etapas: Comunicação com cliente, planejamento, análise de riscos, engenharia, construção/liberação e validação do cliente;
    - Orientado por componentes: Baseia-se na construção de incrementos através da reutilização de componentes. Utiliza conceitos de orientação a objetos para construção de classes que encapsulam dados e algoritmos para sua manipulação. Incorpora características do modelo em espiral como base para gerar seus incrementos de *software*.

As figuras 4, 5, 6 e 7 apresentam alguns dos modelos apresentados.

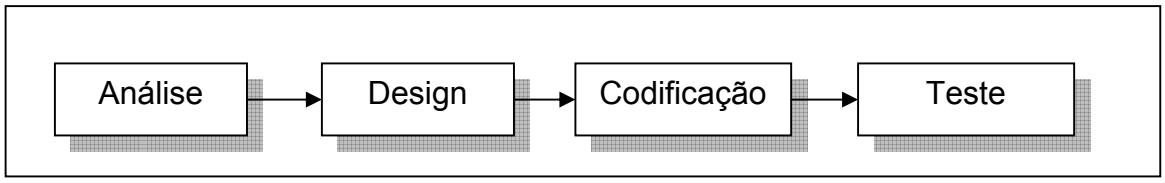

Figura 4: Modelo seqüencial linear
Fonte: Pressman, 1997.

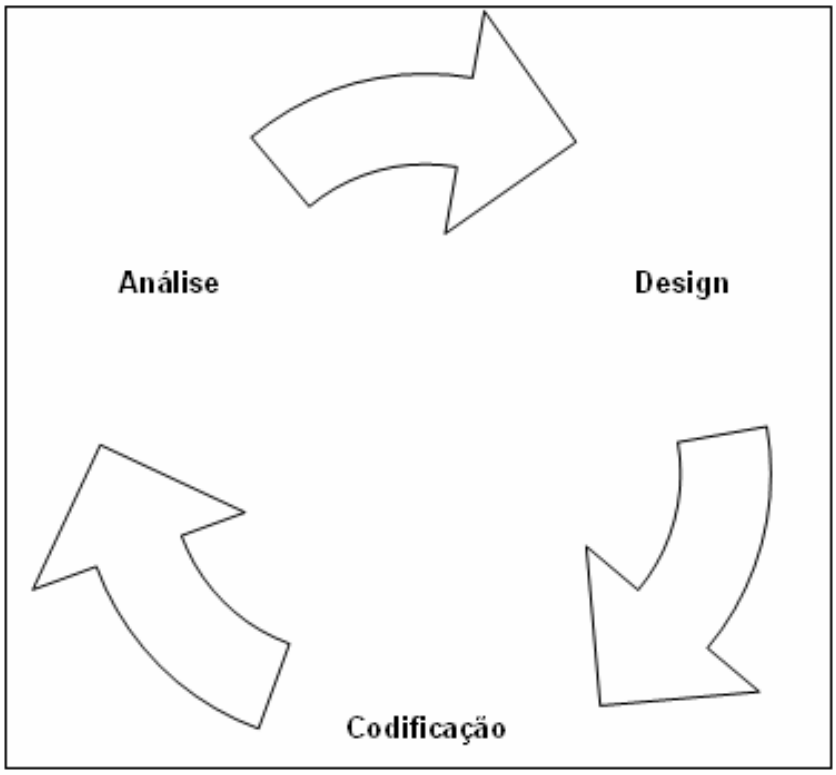

Figura 5: Modelo de prototipação.
Fonte: Pressman, 1997.

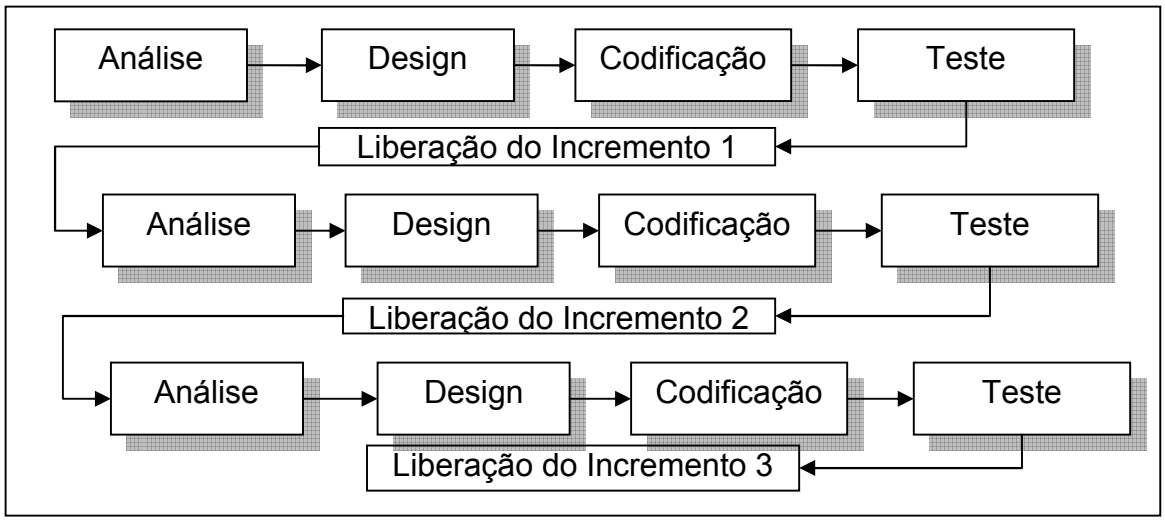

Figura 6: Modelo incremental
Fonte: Pressman, 1997.

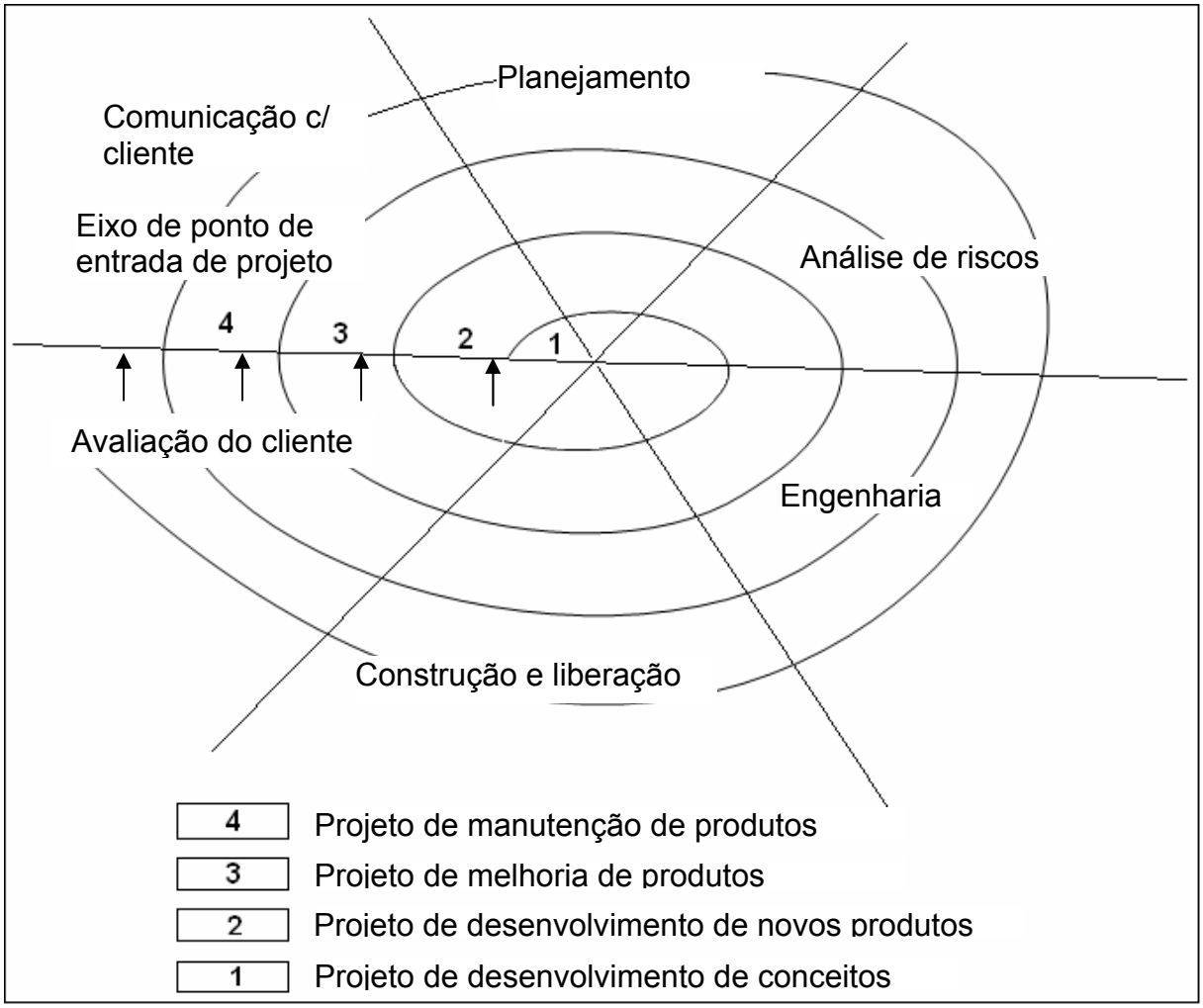

Figura 7: Modelo espiral
Fonte: Pressman, 1997.

2.5. Engenharia de Requisitos

O processo de descobrir, analisar, documentar e verificar as funções e restrições de um determinado sistema informatizado é denominado de engenharia de requisitos (Sommerville, 2003).

Segundo o autor, a engenharia de requisitos contempla requisitos de usuário, de sistema e especificação de projeto de *software*. Os requisitos de usuário designam requisitos abstratos de alto nível. Os requisitos de sistema se caracterizam por indicar uma descrição detalhada do que o sistema deve fazer. A especificação de projeto de *software* é uma descrição abstrata para embasar o projeto e a implementação detalhada do *software*. Esta especificação acrescenta mais detalhes para os requisitos de sistema.

A má implementação de atividades que tratam distintamente desses três tipos de requisitos são fatores que podem causar vários problemas nas fases posteriores de desenvolvimento. Direfentes níveis de especificação de requisitos são úteis porque comunicam informações para distintos tipos de leitores (Sommerville, 2003).

Assim sendo, segundo o autor, o documento de requisitos de *software* é a declaração estabelecida dos requisitos de sistema. Ele deve ser organizado de modo que possa ser utilizado pelos clientes de sistema e pelos desenvolvedores de *software*.

Conforme Espindola et al. (2004), a falta de documentação é um dos principais problemas que são frequentemente encontrados na etapa de levantamento de requisitos para a manutenção de software.

Dessa forma, faz-se necessário validar os requisitos após sua definição. Sua validação pode ser implementada através de um esboço do documento de requisitos, que demonstre o que o cliente deseja. Ignorar a validação do documento de requisitos pode gerar inconsistências ao *software* e grandes custos relacionados ao retrabalho (Sommerville, 2003). Zanlorenci & Burnett (2003) também sugerem que o usuário do software deve ter participação na validação dos requisitos.

2.6. Redes de *Petri*

Rede de *petri* (*Place/Transition nets*: *PT-nets*) é uma ferramenta gráfica que propicia estudar e descrever sistemas que processam informações, cujas características contemplam operações concorrentes, assíncronas, distribuídas, paralelas, não determinísticas e/ou estocásticas. Sua aplicação pode se dar em múltiplas áreas, como compiladores, sistemas de informação e linguagens formais (Murata, 1989).

De acordo com Murata(1989), uma rede de *Petri* é definida por uma quíntupla PN = (P, T, F, W, Mo), sendo:

P = {p1, p2, ..., pm} é um conjunto finito de lugares;

T = {t1, t2, ..., tn} é um conjunto finito de transições;

$F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos (Relação de fluxo);

$W: F \rightarrow \{1,2,3,...\}$ é uma função de pesos dos arcos;

$Mo: F \rightarrow \{0,1,2,3,...\}$ é a marcação inicial;

$P \cap T = \emptyset$ e $P \cup T \neq \emptyset$.

Uma rede de *Petri* sem nenhuma marcação inicial especificada, com estrutura N = (P, T, F, W), é denotada por N. Quando uma marcação inicial estiver especificada, sua denotação dar-se-á por (N, M0) (Murata, 1989). Ainda segundo o autor, dois tipos de propriedades podem ser estudadas na modelagem de redes de *Petri*:

1. As que dependem da marcação inicial (Propriedades comportamentais). Ex.: Acessibilidade; K-Limitadas; Vivacidade e Reinicializáveis;
2. As que não dependem da marcação inicial (Propriedades estruturais). Ex: Vivacidade estrutural; Controlabilidade; Limitação estrutural; Repetitividade.

O formalismo de redes de *Petri* dispõe de algumas técnicas para análise de processos de negócio (Aalst et al., 2002), que podem ser utilizadas como base para a proposição de novos cenários. As técnicas de análise de *workflow*, segundo os autores, se caracterizam em qualitativas e quantitativas. As qualitativas visam à correção lógica de um determinado processo, tratando de problemas como *deadlocks* (quando ocorre um bloqueio, não permitindo que o processo evolua) e *livelocks* (quando ocorre um *loop* sem fim em determinado ponto do processo). As quantitativas têm foco na performance de um dado processo, buscando indicadores de performance, como média de tempo de execução, nível de serviço e capacidade de utilização.

A técnica de análise qualitativa pode ser assim sub-classificada (Aalst et al., 2002):

- Análise de acessibilidade: Determina, a partir de um estado inicial, quais estados serão alcançados e qual a ordem de sua acessibilidade. Tal aplicação se dá com a utilização de "grafos de acessibilidade";
- Análise estrutural: Preocupa-se com a composição dos lugares, das transições e seus arcos de entrada e saída. Busca corrigir possíveis problemas na estrutura de um dado processo.

A análise quantitativa pode ser sub-dividida em duas áreas:

- Análise de performance: Utilizando o método de simulação, busca-se aprimorar a performance do processo em um dado *case*. Neste contexto, procura-se reproduzir o processo real em um *software* computacional, onde os resultados tendem a retratar a realidade, para que a análise possa se dar de forma mais precisa.
- Planejando a capacidade: Tem foco na classe de recursos, assim como, na sua capacidade de execução das tarefas. Objetiva efetuar a alocação das tarefas para um determinado recurso.

2.7. Trabalhos com abordagem em Processos de Desenvolvimento de *Software*

Muitos trabalhos estão relacionados ao processo de desenvolvimento de *software*, buscando focar distintos pontos na qualidade do mesmo. Estudos de caso, como o de Bezerra (2004), vêm sendo desenvolvidos para avaliar a aplicabilidade e a continuidade da implantação de gestão de projetos no processo de desenvolvimento de *software*. Melhoria no processo é outro elemento de estudo de caso, no qual Parreiras et al. (2004) visam avaliar os impactos exercidos pelo processo de desenvolvimento de *software* (PDS) sobre a gestão do conhecimento (GC) nas organizações, assim como os impactos exercidos pela GC sobre a qualidade do PDS.

Da mesma forma, Subramanian et al. (2006), demonstram em seu trabalho que estratégias de implementação de sistemas de informação podem ser aplicadas com o uso do modelo de maturidade e capacidade (CMM) em empresas que desenvolvem *software*. Apresentam também um comparativo de que os altos níveis de CMM retratam a performance no projeto e a qualidade no *software*.

Mantendo foco na melhoria de processos, Kantorski & Kroth (2004) apresentam a necessidade de efetuar medições para que seja possível efetuar melhorias sobre o processo. O gerenciamento e relacionamento entre *workflows* também são apresentados como fatores decisivos na melhora do processo. Algumas normas, como a ISO/IEC 15504 apresentada por Silva et al. (2003), também são empregadas como meios para se obter melhorias sobre o processo desenvolvimento de *software*. Tamaki & Hirama (2007) buscam demonstrar também como é possível projetar e implantar melhorias sobre um processo existente, aplicando conceitos de propriedade sobre processos (*process patterns*).

De forma geral, algumas questões, expostas por Kettunen & Laanti (2005), são colocadas em pauta na escolha de um novo modelo de processo de desenvolvimento de *software*. Neste contexto, para organizações que pretendem adotar um novo processo, assuntos relativos à agilidade no desenvolvimento, assim como, à área de mercado a ser atendida pelo *software*, são expostos para serem levados em consideração para selecionar um novo modelo.

Uma abordagem com relação à agilidade no PDS é realizada por Qureshi & Hussain (2007), onde é proposto um modelo de processo adaptativo para PDS,

baseada na programação extrema (XP). Tal modelo tende a atender a pequenos, médios e complexos projetos de software, com agilidade em seu processo.

Buscando especificar as distintas fases do PDS, Rauterberg & Strohm (1992) demonstram que o processo de desenvolvimento de software deve ser separado em sub-processos, de modo a dar especial atenção para as fases preliminares, que contemplam a análise e a especificação de requisitos. Tais fases tendem, dessa forma, a melhorar o custo do desenvolvimento e da aplicação. Segundo os autores, quanto maior for o esforço dedicado às fases iniciais, menos problemas serão encontrados nas fases posteriores de manutenção do produto.

Neste contexto, em que o processo de desenvolvimento de *software* vem sendo foco para conquista de diferencial competitivo, este trabalho buscará propor uma abordagem que visa proporcionar uma nova contribuição para melhoria no processo de desenvolvimento de *software*. Tal contribuição fará uso do formalismo de redes de *Petri* para modelar processos de desenvolvimento de *software* reais. Tal modelagem será utilizada então para possibilitar a realização de análises sobre o atual processo, e deste modo, propor novos cenários para o processo em questão.

2.8. Trabalhos com redes de *Petri* com abordagem em processo de negócio

Buscando analisar a aplicabilidade da ferramenta redes de *Petri* sobre processos de desenvolvimento de *software*, busca-se neste tópico apresentar alguns trabalhos que, embora não sejam relacionados diretamente ao processo de desenvolvimento *software*, utilizam redes de *Petri* para modelar e analisar processos de negócios.

Para modelar e analisar *workflows* e sistemas gerenciadores de *workflows*, Aalst et al. (1994) apresentam uma abordagem baseada em Redes de *Petri* de alto nível (RdP coloridas). Nesta abordagem, são apresentados conceitos de workflows, assim como utilizar RdP de alto nível na sua modelagem. Com base nas análises em RdP de alto nível, os autores desenvolveram um protótipo de um sistema gerenciador de *workflows* utilizando a ferramenta *ExSpect* (ferramenta baseada em RdP de alto nível).

Da mesma forma, Pádua et al. (2004), apresenta como redes de *Petri* podem ser aplicadas na modelagem de workflows de processos de negócio. Os autores apresentam um exemplo de modelagem e como efetuar a análise sobre os modelos concebidos. Em conseqüência, conclui-se que, além de possuir representação matemática formal, juntamente com um ambiente gráfico de boa visualização, a grande conveniência de se utilizar redes de *Petri* na modelagem de processos de negócio, é de conseguir efetuar um "rastreamento minucioso e não-ambíguo de cada etapa da operação".

No contexto de processos de negócio, Yi et al. (2004), efetuam a modelagem e a análise em redes de *Petri* coloridas, de um modelo de workflow para gerenciamento de autorizações para execução ou restrição de tarefas. O trabalho apresenta a utilização de várias técnicas de análise disponíveis, em especial a técnica de "grafos de acessibilidade" para analisar se os estados de autorização podem ser alcançáveis.

Em busca de uma maior flexibilidade na alocação de processos e sub-processos modulares para produção de distintos produtos e buscando atender mercados específicos distintos, Jiao et al. (2004) utilizam as RdP orientadas a objetos e as RdP coloridas para representar e modelar uma estrutura mutável de produção. Neste contexto, as RdP orientadas a objeto são utilizadas para acomodar a representação de objetos físicos, assim como para representar a alocação dos componentes genéricos para instâncias específicas no sistema flexível. As RdP coloridas são utilizadas para encontrar uma configuração válida para cada especificação de produto a ser produzido no ambiente flexível. Assim sendo, distintos processos podem ser encontrados e tratados em um processo genérico com estrutura flexível.

Considerando processos de *workflows* complexos, que podem ser de difícil compreensão, Zerguini (2004) apresenta meios para se melhorar e simplificar as redes de *Petri* sem perder as características iniciais do processo. Para isso é proposta uma transformação, da rede que representa o *workflow*, em redes de *Petri* hierarquicamente decompostas. Tal transformação, segundo o autor, tende a facilitar na análise da rede.

Buscando considerar e tratar de recursos em uma rede de *Petri workflow*, Jiao & Cheung (2006) demonstram como as propriedades de integridade (*soundness*) podem ser preservadas ao realizar a composição e o refinamento de atividades em uma modelagem. Trabalhos, como o de Suraj et al. (2006), também são direcionados para demonstrar o quão importante são as ferramentas computacionais que vêm a auxiliar no trabalho com redes de *Petri*. Em sua abordagem, é indicada que uma ferramenta deve conter três requisitos essenciais, sendo um bom editor, um simulador e um poderoso mecanismo de análise.

Técnicas de modelagem e simulação de processos em CPN (Redes de *Petri* coloridas) também foram utilizadas, com sucesso, como base para sistemas de apoio a decisão (SAD), descrito por Hennemann et al.(2006). Buscando descrever processos de análise de sistemas, Kawabata et al. (2005) demonstram a aplicação de redes de *Petri* para possibilitar várias visualizações sobre um determinado sistema, através de sua representação gráfica.

Da mesma forma, Kawabata & Itoh (2006), demonstram que redes de *Petri* também podem ser utilizadas em processos de educação sobre engenharia de software e de análise de sistemas, em que sua colaboração, através do ambiente visual, favorece no entendimento do sistema e na identificação de componentes reusáveis.

Buscando focar o produto concebido pelo processo de desenvolvimento de *software*, ou seja, o próprio *software*, e criar um mecanismo que represente estruturalmente o que o *software* executa, Schmerl et. al. (2006) utilizam RdP coloridas para efetuar a integração teórica que possibilite sua representação estrutural. A integração é realizada buscando demonstrar em alto nível, numa visão estrutural abstrata, as operações que o *software* (baixo nível) contempla, permitindo assim, análises com maior precisão e eficácia sobre o *software*.

Deste modo, com base nos trabalhos acima apresentados, observa-se que a representação gráfica e as técnicas de análise, do formalismo de Redes de *Petri,* tem sido muito explorada em distintos processos de negócio. Da mesma forma, este trabalho buscará demonstrar como Redes de *Petri* pode auxiliar na modelagem e na análise do processo de desenvolvimento de *software*.

2.9. Considerações Finais

Buscou-se neste capítulo apresentar características de *workflow,* de processos de negócios e de engenharia de requisitos com foco em desenvolvimento de *software*. Com base nestes conceitos, e em busca de uma abordagem formal para sua representação, foi apresentado, na seqüência, fundamentos de redes de *Petri*, assim como, seus métodos de análise sobre processos. Para finalizar foram apresentados alguns trabalhos, que embora não tratem somente de *software*, ou de processos de desenvolvimento de *software*, utilizam redes de *Petri* como formalismo para modelar e analisar seus processos de negócio.

3. PROPOSTA METODOLÓGICA DESTE TRABALHO

Para agregar e direcionar o conteúdo apresentado até agora, de forma a direcionar a obtenção de resultados, este capítulo tem por finalidade propor um *framework* para o desenvolvimento do trabalho.

Inicialmente será apresentada a forma com que se pretende realizar o levantamento informacional do processo de desenvolvimento de *software* (PDS). Na seqüência será apresentado como a modelagem será implementada. Um novo tópico, relativo à análise, dá seqüência a este capítulo, demonstrando como se pretende efetuar a análise sobre o processo de desenvolvimento de *software* modelado.

Buscando propor novos cenários, em seguida é apresentado um tópico sobre a proposição e análise de cenários.

Por fim, as considerações finais são expostas, buscando resumir o exposto neste capítulo.

A figura 8 ilustra a proposta metodológica para o desenvolvimento deste trabalho, demonstrando de forma cíclica os itens relevantes a serem tratados.

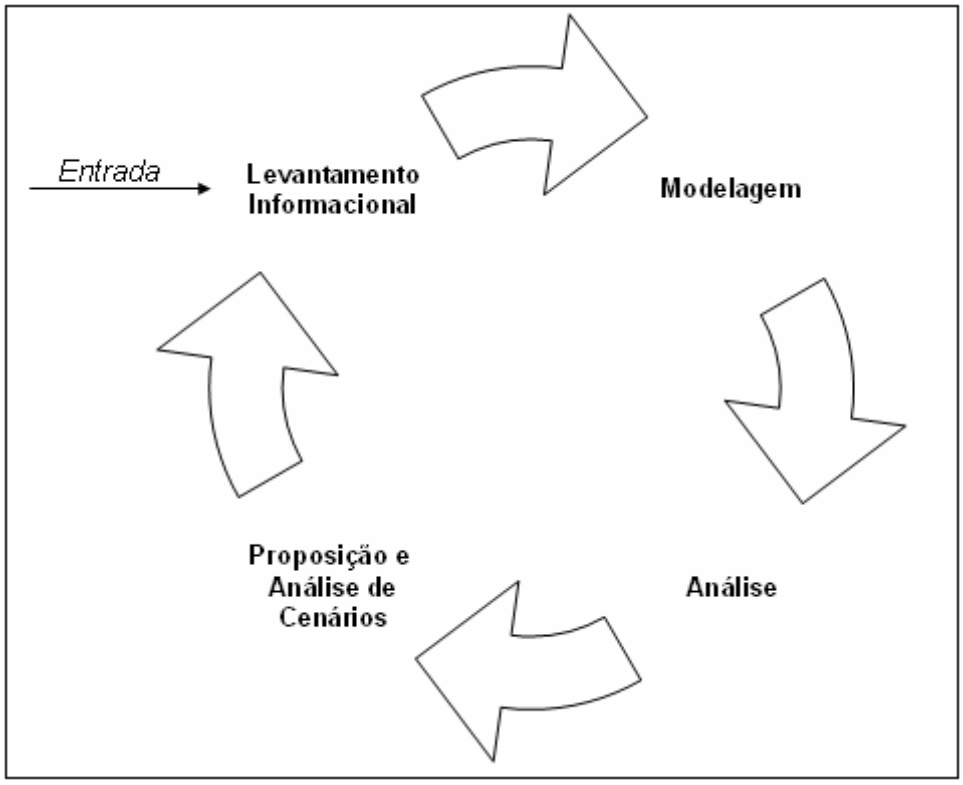

Figura 8: Estrutura da proposta metodológica
Fonte: Autor.

3.1. Levantamento informacional

Para realizar o levantamento informacional, em uma empresa desenvolvedora de *software*, buscar-se-á num primeiro momento delimitar alguns itens, como tema de estudo, elemento de estudo, recursos a serem estudados, assim como, as variáveis que serão consideradas para o estudo. Em seguida será apresentado, de forma seqüencial, o roteiro de realização do levantamento informacional.

3.1.1. Itens a serem considerados no levantamento informacional

Os itens abaixo têm objetivo de servir como delimitadores, buscando direcionar e esclarecer os pontos relevantes para este trabalho.

1. Tema de estudo

- ✓ Processo de desenvolvimento de *software*

2. Elemento de estudo
   - ✓ Ordem de serviço (OS): Ordem de serviço será considerada neste trabalho como sendo o elemento (objeto) a ser tratado pelo tema de estudo, onde a mesma (OS) tende a resultar em forma de *software* (Incremental ou totalmente).
3. Atividades a serem estudadas
   - ✓ Setor: Dentro de uma empresa que desenvolve *software*, buscar-se-á identificar as distintas atividades que os setores da empresa contemplam (Ex.: Manutenção; Projetos; Desenvolvimento; Testes) e que fazem parte do tema de estudo.
   - ✓ Humano: Buscar-se-á identificar as atividades humanas que fazem parte dos setores.
4. Variáveis de estudo
   - ✓ Tempo de execução: Tempo em que as OSs são executadas, por recurso;
   - ✓ Capacidade de execução: Capacidade de OSs a serem executadas em determinados intervalos de tempo.

3.1.2. Roteiro de realização do levantamento informacional

Abaixo será apresentada a seqüência com que o levantamento informacional será efetuado em uma organização real desenvolvedora de *software*. Neste contexto, a observação e a conversação guiada por temas serão técnicas constantes neste levantamento.

1. Estudo geral do tema de estudo (Processo de desenvolvimento de *software*): Estudo no qual se pretende entender a estrutura e o fluxo geral do processo de desenvolvimento de *software* da organização.
2. Estudo do elemento de estudo (OSs): Será estudada a forma com que a organização classifica as OSs.

Ex.: Complexidade: Baixa, Média, Alta...

Tipicidade: Corretiva, Adaptativa, Melhoria...

3. Estudo das atividades: Estudo das atividades setoriais e humanas, onde se pretende identificar sua inter-relação.
4. Estudo das variáveis de estudo: Buscar-se-á estudar, com base nos documentos históricos da empresa, assim como, na conversação guiada por temas, as variáveis “Tempo de execução” e “Capacidade de execução” para cada recurso.

Desta forma, pretende-se então levantar dados com base no tempo de execução e na capacidade de execução realizada na empresa.

Assim sendo, procurar-se-á quantificar a capacidade de execução em um dado período para o processo de desenvolvimento de software, para na seqüência, relacionar o elemento de estudo (OSs), na mesma perspectiva da capacidade de execução por classificação das OSs no período. De forma similar, os dados serão agrupados em busca de quantificar a capacidade de execução por recurso.

Buscando contrapor os dados com a realidade estrutural do processo de desenvolvimento de software da empresa, buscar-se-á informações que permitam avaliar a estrutura de dependências entre as atividades dos recursos. Tal informação será de extrema necessidade para efetuar a modelagem do cenário atual.

A figura 9 representa o que deve ser abordado para o levantamento informacional.

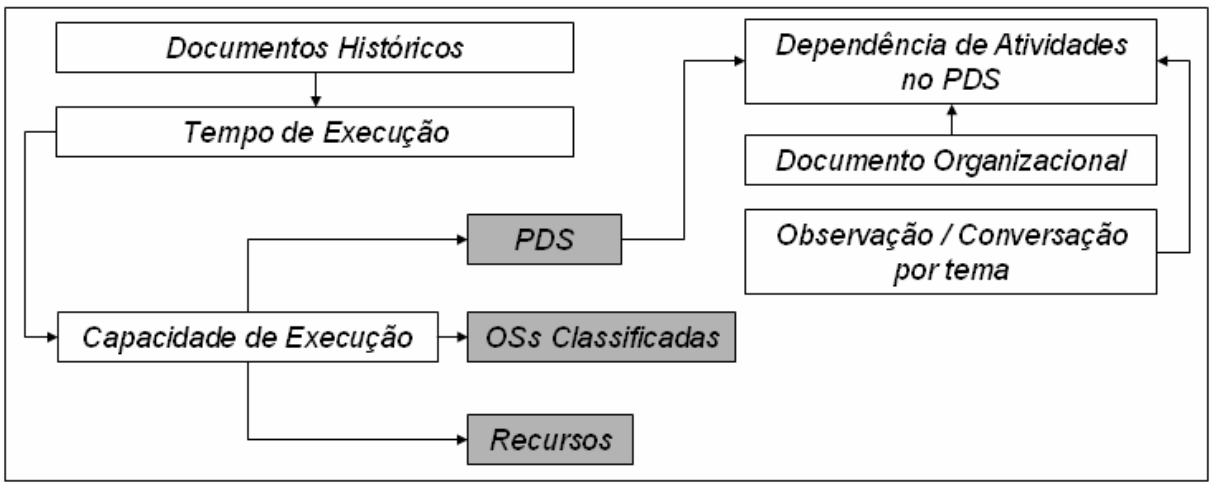

Figura 9: Levantamento informacional
Fonte: Autor.

De forma a esclarecer a imagem acima, podemos considerar que a capacidade de execução será levantada através de documentos históricos que retratem um determinado intervalo de tempo de execução. Dessa forma, buscar-se-á quantificar a capacidade de execução no âmbito de todo o processo de desenvolvimento de software. Buscando detalhar, outras visões serão abordadas, sendo que a capacidade de execução será levantada para as OSs e suas classificações e para as atividades setoriais e humanas.

De forma similar, a dependência das atividades do PDS será levantada, para que seja possível dar início a modelagem do cenário atual. Para tal, documentos organizacionais, que retratem o processo de desenvolvimento de software, serão analisados. Paralelamente, e buscando coletar informações que não estejam documentadas a respeito do PDS, será necessário efetuar observações *in loco*, onde pode ser necessário efetuar conversas, de acordo com o tema que está sendo observado, para obter esclarecimentos sobre as atividades do PDS.

3.2. Modelagem do cenário atual

Com base nos dados extraídos no levantamento informacional, buscar-se-á modelar o processo de desenvolvimento de *software*, com seus sub-processos, em busca de representar e simular a realidade em um ambiente virtual.

Para efetuar a modelagem, utilizar-se-á a ferramenta formal de redes de *petri*, utilizando o *software* Income como ambiente virtual.

Neste contexto, o termo “OS” demonstra o elemento de estudo, sendo a representação de uma solicitação dentro do processo de desenvolvimento de *software*.

O termo “atividade” representa parte, ou todo, de um trabalho. Neste contexto, conforme Aalst et al. (2002), uma atividade deve ser entendida pelo trabalho a ser exercido, independente da situação ou resultado.

Dessa forma, os seguintes itens demonstram como a modelagem será implementada:

1. Modelagem geral: Modelagem do processo de desenvolvimento de *software* de forma abstrata, contendo:
   - ✓ Seleção e ordenação das atividades;
   - ✓ Relacionamento entre as atividades;
   - ✓ Mapeamento de dependência de atividades;
   - ✓ Disposição das atividades em forma de fluxo;
   - ✓ Tratamento das OSs por um contexto geral.
2. Modelagem refinada: Modelagem que busca refinar as atividades principais dos setores da empresa, contendo:
   - ✓ Seleção e ordenação das atividades;
   - ✓ Relacionamento entre as atividades;
   - ✓ Mapeamento de dependência de atividades;
   - ✓ Disposição das atividades em forma de fluxo;
   - ✓ Tratamento das OSs por atividades refinadas;

3.3. Análise do cenário atual

Com base na modelagem efetuada sobre o processo de desenvolvimento de *software*, análises serão efetuadas buscando considerar aspectos qualitativos e quantitativos, como segue:

1. Análise qualitativa: Análise que pretende observar a correta disposição estrutural do fluxo de atividades no processo de desenvolvimento de *software*. Segundo Aalst (2002), para que alguns erros simples sejam solucionados, a análise qualitativa deve tratar das seguintes avaliações:
   - ✓ Atividades desnecessárias: Quando uma atividade não tem uma entrada no fluxo de atividades, que a condicione ser ativada, ou mesmo, não tem uma saída, no fluxo de atividades, para qual enviar resultados, a atividade não tem razão de existir;

- ✓ Atividades mortas: Quando uma atividade, apesar de estar interligada no fluxo de atividades do processo com lugares de entrada e saída, não pode ser acionada por falta de elementos de entrada;
- ✓ Vivacidade: Definido pela análise de deadlocks e livelocks;
  - I. Deadlock: Quando a evolução do processo torna-se, em um determinado momento, bloqueada, onde a atividade seguinte fica esperando infinitamente para que suas entradas tenham disponibilidade para que o processo evolua, sendo que, não tenha outra alternativa para evoluir o processo. Neste caso o fluxo de atividades no processo fica travado.
  - II. Livelock: Ocorre com um laço de repetição entre duas ou mais atividades, sendo que tal laço se torna infinito no processo, fazendo com que o processo fique bloqueado por este laço.
- ✓ Soundness (Fortemente modelada): Três são os requisitos para que um modelo em rede de *petri* seja considerado como “forte”:
  - I. Para cada entrada de ficha no processo, somente uma saída, eventualmente, será gerada no final do processo;
  - II. Quando uma ficha aparecer no final do processo, todos os demais lugares no processo deverão estar limpos (sem resíduos relativos à ficha de saída);
  - III. Para cada atividade, esta deve ser capaz de transportar a ficha do lugar de entrada para o lugar de saída.

2. Análises quantitativas: Análise com vista a quantificar a performance atual do processo de desenvolvimento de *software*. Os seguintes itens serão considerados na análise:
   - ✓ Tempo de execução: Tempo que o processo, assim como suas atividades, leva para executar determinada OS.
   - ✓ Qualidade de execução: Capacidade de execução para as atividades.

Dessa forma, embasado nessas análises, será possível efetuar um diagnóstico a respeito da situação atual do processo de desenvolvimento de *software* da empresa, no qual serão apontados alguns pontos que podem ser tratados na proposição de um novo cenário.

3.4. Proposição e Análise de Cenários

Com vista na análise efetuada sobre a modelagem do processo de desenvolvimento de *software*, serão propostos novos cenários, os quais tendem a proporcionar uma melhor alocação dos recursos, assim como, um melhor aproveitamento do tempo por parte dos mesmos na execução de suas atividades.

Embasado na análise qualitativa, e nos quesitos da análise quantitativa, a questão “tempo de execução” e “qualidade de execução” serão abordados para buscar um novo cenário.

Assim sendo, com base na modelagem do ambiente real do processo de desenvolvimento de *software*, assim como nos novos cenários a serem propostos, será possível efetuar uma análise buscando avaliar, de um lado, o esforço para alteração do processo, e de outro, os benefícios que tal alteração trará para a organização.

A análise comparativa do cenário atual com o cenário proposto será estruturada tabularmente, com as seguintes informações:

- ✓ Alteração: Descrição da alteração a ser efetuada dentro do processo de desenvolvimento de *software*;
- ✓ Esforço necessário: Dificuldade e obstáculos que a empresa terá para efetuar a alteração;
- ✓ Efeitos: Efeitos derivados da alteração;
- ✓ Benefícios: Benefícios esperados pela alteração;
- ✓ Impactos negativos: Relação de alguns dos possíveis impactos que poder surtir negativamente para o processo.

3.5. Considerações Finais

Neste capítulo foi apresentada uma proposta metodológica para a realização deste trabalho.

O levantamento informacional do PDS foi apresentado inicialmente, sendo que, na seqüência, em tópicos distintos, se apresentou como a modelagem e a análise serão realizadas.

De forma similar, para finalizar este capítulo, buscou-se demonstrar um mecanismo para a proposição de novos cenários, assim como, para sua modelagem e análise.

A figura 8, apresentada no início deste capítulo, ilustra de forma cíclica como a proposta metodológica está estruturada.

4. CENÁRIO ATUAL DA EMPRESA

Sediada em Chapecó/SC, a empresa Desbravador Automação Hoteleira desenvolve *software* para o setor hoteleiro, atendendo a mais de mil hotéis na América Latina (Brasil, Argentina, Paraguai, Uruguai e Bolívia), nos Estados Unidos e em Portugal. O produto da empresa é o *software* Desbravador, cujas versões são três:

1. Desbravador *Light*: Disponível para atender pequenos hotéis;
2. Desbravador 3.0: Atender médios hotéis;
3. Desbravador 4.0: Atender médios e grandes hotéis, incluindo redes de hotéis.

A sua estrutura de atendimento é descentralizada, composta por empresas terceirizadas exclusivas para atender aos clientes da empresa Desbravador. A sua equipe central, disposta na sede da empresa, em Chapecó/SC, é composta por 43 pessoas.

Este capítulo tem por objetivo realizar um estudo direcionado sobre o processo de desenvolvimento de *software* na empresa. O estudo terá início demonstrando a estrutura da empresa. Na seqüência o processo de desenvolvimento de *software* da empresa será apresentado, de modo a levantar os dados necessários para este trabalho. Com base nos dados levantados sobre o processo de desenvolvimento de *software* da empresa, será realizada a modelagem e a análise, buscando, na seqüência, diagnosticar o cenário atual.

A figura 10 apresenta a estruturação do presente capítulo.

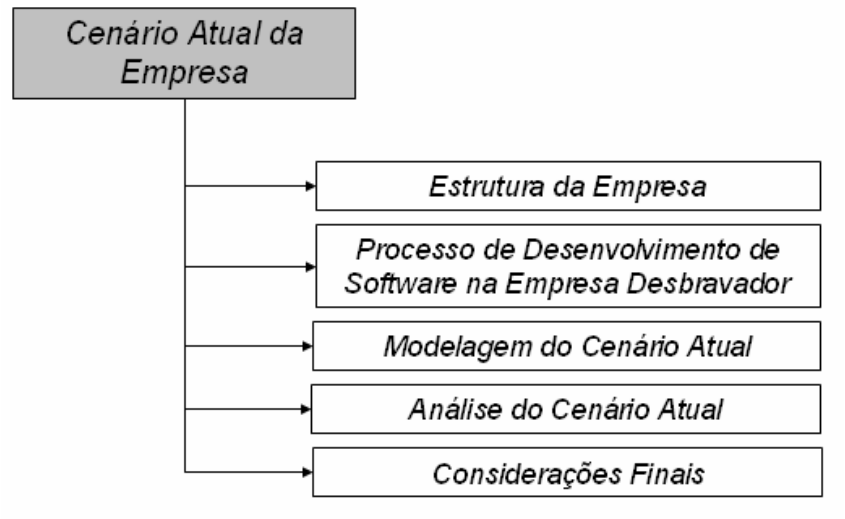

Figura 10: Estrutura do Cenário Atual da Empresa
Fonte: Autor.

4.1. ESTRUTURA DA EMPRESA

Buscando melhor delimitar suas atividades, a empresa organizou-se em setores, cada qual com função bem definida, sendo:

- Administrativo: Atende questões financeiras, de recursos humanos e questões administrativas, necessárias para o bom andamento da empresa;
- Comercial: Responsável por questões relativas à comercialização do produto;
- Projetos: Responsável pela elaboração de novos projetos a serem desenvolvidos na empresa;
- Programação: Responsável pela atividade de programação e manutenção do *software*;
- Testes: Setor cuja responsabilidade é validar o produto final da empresa;
- Suporte: Setor responsável para dar suporte às empresas terceirizadas, sendo que, em última instância, poderá dar suporte direto ao cliente.

- Idiomas: Setor com a função de traduzir o sistema para os distintos idiomas dos países no qual o sistema está em funcionamento.

A tabela 1 demonstra a capacidade de recursos humanos para cada setor:

Tabela 1: Capacidade de recursos humanos por setor
Fonte: Empresa Desbravador, 2007

Setor	Pessoas
Administrativo	*7*
Comercial	*3*
Projetos	*5*
Programação	*11*
Testes	*5*
Suporte	*8*
Idiomas	*4*
Total	**43**

4.2. PROCESSO DE DESENVOLVIMENTO DE *SOFTWARE* NA EMPRESA DESBRAVADOR

Buscando entender como a informação é tratada pelo processo de desenvolvimento de *software*, contido na empresa Desbravador, buscou-se elencar quais são as atividades principais, por setor, assim como suas inter-relações. Sabendo que o termo “OS” representa uma solicitação dentro do processo de desenvolvimento de *software,* faz-se necessário primeiramente analisá-la. As OSs são os elementos a serem tratados pelo fluxo de atividades através de recursos, os quais são apresentados na seqüência, como sendo **recursos humanos** e **recursos setores**.

Dando seqüência ao estudo sobre o processo de desenvolvimento de *software*, buscar-se-á apresentar os dados históricos relativos ao processo na empresa. Estes dados serão apresentados objetivando demonstrar a situação atual da empresa, considerando o **tempo de execução** e **capacidade de execução**.

4.2.1. ORDEM DE SERVIÇO (OS)

Ordem de serviço (OS) é tratada neste trabalho como elemento principal a ser analisado no decorrer do processo de desenvolvimento de *software*.

Por ter natureza abrangente e certas especificidades, ou seja, por tratar de diferentes níveis de complexidade para sua implementação no processo de desenvolvimento de *software*, as OSs têm distintas classificações na empresa. A tabela 2 demonstra como as OSs são classificadas.

Tabela 2: Classificação das OSs
Fonte: Empresa Desbravador, 2007

Classificação	Opções
Congelada	*Sim* *Não*
Tipo	*Correção* *Alteração* *Implementação*
Severidade	*Assunto crítico* *Assunto importante* *Assunto de menor importância* *Pedido de mudança de aplicação*
Prioridade	*Urgente* *Alta* *Normal* *Baixa*
Complexidade	*Alta* *Normal* *Baixa*

A classificação de “Congelada” indica que a OS deve ser implementada (desenvolvida pelo setor de programação) em uma versão anterior (última versão liberada) à versão atual. Dessa forma, a OS será implementada duas vezes, sendo uma na última versão liberada (denominada de congelada) e outra na versão atual.

O “Tipo” caracteriza um indicador da origem da OS. Assim sendo, se a OS tem origem em uma requisição do cliente, e a mesma é oriunda de uma OS anterior, que saiu com defeito, a mesma passa a ser considerada como “Correção”. Se uma OS tem origem em uma requisição do cliente, sendo que a mesma não é um defeito do sistema, e tal requisição sugere para que o *software* se comporte com distinção em certas circunstâncias, então a OS passa a ser do tipo “Alteração”. No entanto, se

uma requisição provinda do cliente diz respeito a um novo recurso, que até então não era contemplada pelo *software*, a OS é considerada como sendo do tipo "Implementação".

A classificação de "Severidade" tem intuito de especificar um grau de importância para as OSs. A classificação de "Prioridade" objetiva indicar qual OS deve ser desenvolvida com maior ou menor urgência. A "Complexidade" indica o nível de esforço que deve ser dispensado para desenvolver tal OS. A classificação da prioridade passa a ser irrelevante quando se tratando de OSs que estejam com a classificação *Congelada* com valor de *Sim*.

4.2.2. ESPECIFICAÇÃO DAS ATIVIDADES

Para tratar as OSs, no escopo do processo de desenvolvimento de *software*, a empresa executa uma série de atividades, vinculadas a setores e a pessoas. Nem todos os setores apresentados na seção 5.1, fazem parte do fluxo do processo de desenvolvimento de *software*. Assim sendo, abaixo são apresentados, tabularmente, os setores que estão incluídos no processo.

Tabela 3: Setores pertencentes ao processo
Fonte: Empresa Desbravador, 2007

Setor/Atividade	Pessoas
Projetos	*5*
Programação	*11*
Testes	*5*
Suporte	*8*
Total	**29**

Deste modo, verifica-se que as atividades **setores** são compostas de atividades internas, denominadas **pessoas**. Pode-se ainda caracterizar, conforme sugere Aalst et. al. (2002), que os setores realizam atividades de forma generalizada e as pessoas refinam estas atividades. Neste contexto identificam-se dois níveis de fluxo para o processo de desenvolvimento de *software*. O primeiro nível é identificado pelo fluxo de atividades decorrente dos setores, nível este que pode ser considerado como nível generalizado, ou abrangente. O outro nível é um

refinamento dos setores, que tende a demonstrar o processo em um nível mais detalhado, através das atividades das pessoas que compõem os setores.

4.2.3. VARIÁVEIS DE ESTUDO

Para poder avaliar quantitativamente o processo de desenvolvimento de *software*, faz-se necessário obter dados históricos relativos à execução das OSs na evolução do processo. Neste levantamento são consideradas duas variáveis chave para estudo: **Tempo de execução** e **Capacidade de execução**. Para obter os dados de tempo de execução relativos às OSs históricas da empresa, foi utilizado um período de 18 meses, de 01/01/2006 até 30/06/2007.

Para quantificar os dados, foi realizado o enquadramento de acordo com as classificações adotadas pela empresa, acima descritas na tabela 2. Para determinar o tempo de execução, assim como, da capacidade de execução das OSs, foi considerado o intervalo decorrente da data de registro da OS e da data de liberação da OS para implantação. Assim sendo, pôde-se definir os parâmetros iniciais para a coleta de dados históricos referentes à variável tempo de execução, como mostra a tabela 4.

Tabela 4: Parâmetros da variável tempo de execução
Fonte: Empresa Desbravador, 2007

Parâmetros da variável: ***Tempo de Execução***	
Período de levantamento de dados	18 meses (01/01/2006 até 30/06/2007)
Tempo de cada OS	Intervalo da data de registro até a data de liberação da OS
Grupamento de OSs	As OSs serão agrupadas pelas cinco classificações da empresa (Congelada, Tipo, Severidade, Prioridade, Complexidade)

Da mesma forma, para coleta de informações referente à variável capacidade de execução, os seguintes parâmetros são observados, de acordo com a tabela 5.

Tabela 5: Parâmetros da variável capacidade de execução
Fonte: Empresa Desbravador, 2007

Parâmetros da variável: ***Capacidade de Execução***	
Período de levantamento de dados	18 meses (01/01/2006 até 30/06/2007)
Capacidade de execução por recurso	Separado por recurso setor e humano
Grupamento	A capacidade de execução será agrupada por recurso, generalizando pelo recurso setor e especializando pelo recurso humano

Considerando o exposto acima, podem-se relacionar os dados históricos referente à variável tempo de execução das OSs, considerando o período de 18 meses, através da capacidade de execução por classificação das OSs, conforme apresentado na tabela 6.

Tabela 6: Capacidade de execução das OSs por classificação
Fonte: Empresa Desbravador, 2007

Capacidade de execução por classificação: (01/01/2006 até 30/06/2007)		
Número de OSs no período: 3522		
Média de OSs/mês no período: 195,67		
Classificação	***Sub-Classificação***	**Qtde OSs**
Congelada		
	Sim	1380
	Não	2142
Tipo		
	Correção	1402
	Alteração	1301
	Implementação	819
Severidade		
	Ass. Crítico	390
	Ass. Importante	1095
	Ass. De Menor Importância	427
	Pedido de Mudança da Aplicação	93
	Não definido	1517
Prioridade		
	Urgente	2133
	Alta	1144
	Normal	91
	Baixa	29
	Não definido	125

Complexidade		
	Alta	378
	Normal	590
	Baixa	199
	Não definido	2355

Da mesma forma, pode-se relacionar a capacidade de Execução ocorrida no período. Para tanto, devemos considerar que, ao quantificar a capacidade de execução, devemos observar que a empresa considera que uma OS é tratada por uma ou mais operações. As operações são atividades que processam a OS.

Observa-se também que os dados estão sendo analisados pelo período de registro da OS na empresa, até sua efetiva liberação para o cliente. Ao especificar os dados por setor, foi acumulado somente o período que a OS ficou em domínio do mesmo, uma vez que a OS nunca fica sem um setor na evolução do processo de desenvolvimento de *software* da empresa.

Uma outra visão interessante pode ser obtida pelo fluxo, de entradas e saídas, mensal de OSs no processo de desenvolvimento de *software*. A tabela 7 apresenta este fluxo.

Tabela 7: Fluxo de entradas e saídas
Fonte: Empresa Desbravador, 2007

Mês	Entradas	Saídas
1	204	57
2	150	72
3	320	61
4	234	116
5	300	360
6	255	117
7	265	75
8	262	82
9	233	192
10	257	102
11	193	46
12	216	425
13	186	150

14	161	140
15	120	836
16	97	480
17	55	137
18	13	73

Deste modo os dados podem ser mapeados de acordo com a tabela 8. Nesta tabela o fluxo de entradas e saídas é organizado para demonstrar o processamento realizado por setor.

Tabela 8: Capacidade de execução das OSs por recurso setor
Fonte: Empresa Desbravador, 2007

Capacidade de Execução: (01/01/2006 até 30/06/2007)					
Tempo total das OSs (Dias acumulados): 368.910,76 Média de dias/OS no período: 104,74					
Setor	**Núm. de OS**	**Operações das OS/Setor**	**Qtde de Dias (Acumulado das OS)**	**Média (Dias/operação da OS)**	**Média (Dias/OS)**
Projeto	*0*	*0,00*	*0,00*	*0,00*	**0,00**
Programação	*3.500*	*5.080,00*	*63.951,78*	*12,59*	**18,27**
Testes	*3.500*	*18.372,00*	*270.140,50*	*14,70*	**77,18**
Suporte	*3.502*	*4.007,00*	*32.499,48*	*8,11*	**9,28**

Observando a tabela 8, verifica-se que o setor “projeto” não tem dados especificados no período. Isso se deve pelo fato de que o setor fora recentemente criado.

A coluna contendo o número de OS corresponde à quantidade de OSs que passou por determinado setor. Tendo como total a quantidade de OSs no período de 18 meses (01/01/2006 até 30/06/2007), esta quantidade pode variar de setor para setor, dependendo do caminho que a OS siga na evolução do processo de desenvolvimento de *software*.

A coluna que define as “Operações das OS/Setor” identifica o montante relativo às operações que as OSs tiveram no período requisitado. Estas operações são derivadas do andamento que uma OS pode ter durante sua evolução no processo de desenvolvimento de *software*. A coluna “Qtde de dias” demonstra o total

de dias, somados, relativos ao início até o término do processo de desenvolvimento de *software*.

Com estas colunas iniciais, as duas últimas colunas podem ser definidas. Na "média de dias/operação das OS", é obtida a quantidade de dias que uma operação, na evolução do processo, demora para ser concluído. Na "média de dias/OS", é calculada a quantidade de dias que uma OS demora a ser desenvolvida pelo processo de desenvolvimento de *software*, desde seu registro até sua liberação final para o cliente.

Para ter uma visão específica para os recursos humanos, pode-se detalhar a tabela 8 de acordo com a tabela 9.

Tabela 9: Capacidade de execução das OSs por recurso humano
Fonte: Empresa Desbravador, 2007

Setor	**Recurso Humano**	**Núm. de OS**	**Operações das OS**	**Qtde de Dias (Acumulado das OS)**	**Média (Dias/operação das OS)**	**Média (Dias/OS)**
Projeto	***5 RHs***		***0,00***	***0,00***	***0,00***	**0,00**
Programação	***11 RHs***	***3.500***	***5.080,00***	***63.951,78***	***12,59***	**18,27**
	P1	*352*	*523,00*	*4.423,50*	*8,46*	12,57
	P2	*787*	*1.156,00*	*7.705,97*	*6,67*	9,79
	P3	*404*	*513,00*	*25.949,33*	*50,58*	64,23
	P4	*273*	*360,00*	*2.781,49*	*7,73*	10,19
	P5	*383*	*575,00*	*5.650,64*	*9,83*	14,75
	P6	*360*	*468,00*	*4.932,68*	*10,54*	13,70
	P7	*462*	*737,00*	*4.679,67*	*6,35*	10,13
	P8	*452*	*610,00*	*6.521,19*	*10,69*	14,43
	P9	*103*	*127,00*	*1.285,19*	*10,12*	12,48
	P10	*9*	*11,00*	*22,27*	*2,02*	2,47
	P11	*0*	*0,00*	*0,00*	*0,00*	0,00
Testes	***5 RHs***	***3.500***	***18.372,00***	***270.140,50***	***14,70***	**77,18**
	T1	*571*	*1.278,00*	*40.611,04*	*31,78*	71,12
	T2	*635*	*1.226,00*	*56.553,79*	*46,13*	89,06
	T3	*246*	*589,00*	*10.263,37*	*17,43*	41,72
	T4	*107*	*219,00*	*1.927,59*	*8,80*	18,01
	T5	*0*	*0,00*	*0,00*	*0,00*	0,00
	Sem Resp.	*3468*	*11.106,00*	*78.907,78*	*7,10*	22,75
	Realoc.	*N/D*	*3.954,00*	*89.876,93*	*22,73*	N/D
Suporte	***8 RHs***	***3502***	***4.007,00***	***32.499,48***	***8,11***	**9,28**
	S1	*1480*	*1510,00*	*13.379,00*	*8,86*	9,04
	S2	*100*	*109,00*	*1.350,00*	*12,39*	13,50
	S3	*467*	*497,00*	*2.950,00*	*5,94*	6,32
	S4	*472*	*501,00*	*2.124,00*	*4,24*	4,50
	S5	*800*	*805,00*	*9.790,00*	*12,16*	12,24
	S6	*290*	*314,00*	*1750,25*	*5,57*	6,04
	S7	*87*	*90,00*	*527,21*	*5,86*	6,06
	S8	*140*	*281,00*	*629,04*	*2,24*	4,49

Os dados acima, refinados por pessoa, demonstram que a quantidade por coluna, especificada por setor, não equivale, necessariamente, com o total dos recursos humanos (pessoas). Essa diferença acontece pelo fato de que uma OS pode passar por um ou mais recursos do mesmo setor ao decorrer do fluxo de atividades do processo de desenvolvimento de *software*.

Para dar seguimento ao trabalho, assim como facilitar o entendimento dos dados acima coletados, será apresentado na seqüência como está estruturada a empresa com relação ao fluxo de atividades.

4.2.4. FLUXO DO PROCESSO DE DESENVOLVIMENTO DE *SOFTWARE*

Para possibilitar um melhor entendimento de como o processo de desenvolvimento de *software* da empresa está estruturado, faz-se necessário entender quais são os possíveis caminhos que uma OS pode percorrer na evolução do processo.

Deste modo, apresenta-se nesta seção quais são os caminhos existentes, para uma OS, através de duas visões, uma geral, contendo as atividades dos setores, e outra refinada, abrangendo as atividades humanas de cada setor. Para tal, o anexo 1, representando o organograma das atividades no PDS da empresa Desbravador, foi analisado para servir como base de conhecimento.

No fluxo de atividades sobre os setores, a vida útil de uma OS, no processo de desenvolvimento de *software*, é composta por três etapas:

1. Criação: etapa na qual a OS é efetivada no processo;
2. Processamento: Na etapa de processamento a OS estará sendo trabalhada pelas atividades setoriais e humanas;
3. Finalização: Nesta etapa a OS é dada por concluída, findando o ciclo do processo de desenvolvimento de *software* na empresa.

Uma OS (requisição do cliente) pode ser criada, no PDS, por qualquer recurso humano ou setor. Após ser criada, a OS pode permanecer até sua finalização em poder do recurso criador, caso o recurso criador seja pertencente ao

setor de testes ou de suporte, mas também pode ser transferida indefinidamente entre os demais recursos para o processamento da OS.

Para validar ou cancelar uma OS, somente os setores de testes e de suporte podem fazê-lo. Para liberar uma OS para o cliente, mesmo que esta esteja plenamente validada pelo setor de testes, esta deve aguardar a liberação geral que foi previamente estipulada. As liberações gerais são previamente estipuladas com um intervalo de três a cinco meses, onde são reunidas todas as OSs desenvolvidas e dessa forma é gerada uma nova versão do *software* Desbravador. Existe a possibilidade de uma OS ser desenvolvida em uma versão anterior, onde esta é definida com status de congelada. Neste caso a liberação é realizada em um intervalo que varia de um a quatro dias após ser validada pelo setor de testes.

A empresa procura, na maioria dos casos, buscar um caminho ordenado para a OS. Este caminho está assim estruturado:

1. Criação da OS pelo setor de projetos ou suporte;
2. Envio da OS para o setor de programação;
3. Envio da OS para o setor de testes;
4. Finalização da OS pelo testes e liberação para implantação no cliente.

Durante este caminho ordenado podem ocorrer desvios da OS, para que a mesma possa ter maiores esclarecimentos. Desse modo, uma OS que foi criada na etapa 1 pode ser transferida para a etapa 2. Na etapa 2 a OS pode ser devolvida para a etapa 1, para obter esclarecimentos, assim como, pode ser transferida para a etapa 3 ou 4. Da etapa 3, pode ser transferida para a etapa 2 ou para a 4. Nos casos em que a OS for enviada para a etapa 4, a mesma pode estar sendo negada ou implementada, sendo que, se estiver implementada, esta deve estar testada e validada.

Assim sendo, verifica-se que existem certas dependências entre as atividades, de modo que algumas atividades só ocorrerão se outras lhe derem seqüência. Assumindo como atividades aquelas realizadas pelos recursos setores, observa-se que os setores de suporte e de projetos são dependentes de uma requisição inicial para serem ativadas no processo de desenvolvimento de *software*,

mas também podem ser ativados pelo setor de programação, quando este devolver a OS para maiores esclarecimentos. A figura 11 demonstra graficamente a dependência dos setores de suporte e de projetos, através de um diagrama de blocos.

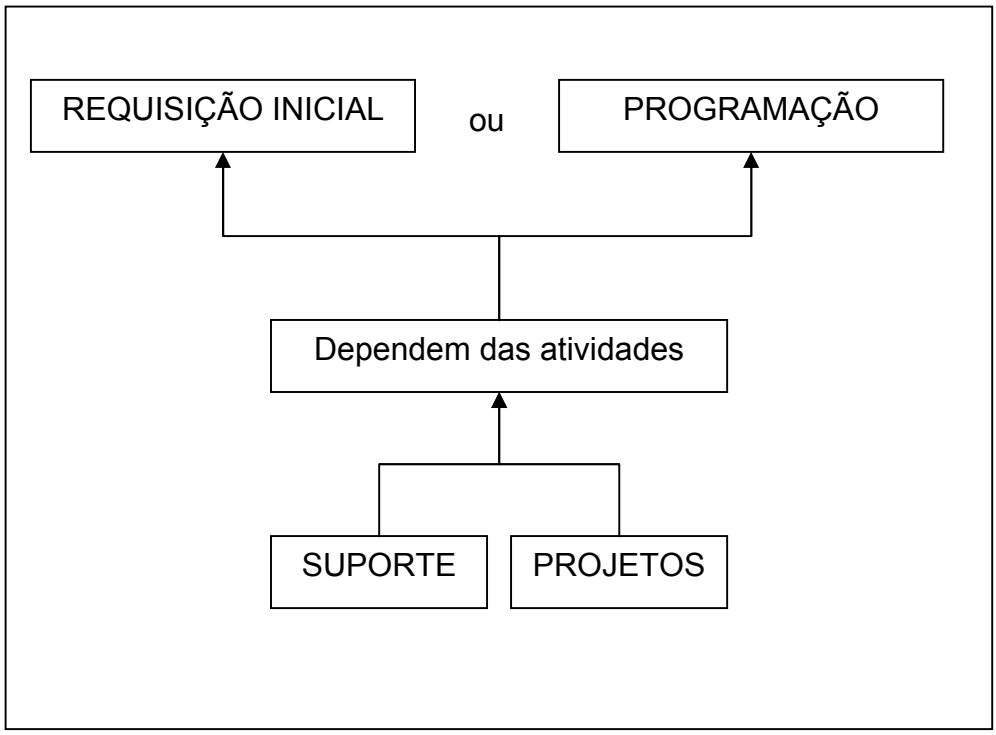

Figura 11: Dependência das atividades dos setores de Suporte e Projetos
Fonte: Autor

A atividade do setor de desenvolvimento, por conseguinte, procederá somente quando for ativada pelas entradas dos setores de suporte, de projetos e de testes. No caso da sensibilização ocorrer pelos setores de suporte ou de projetos, a mesma será por uma nova OS ou por uma OS devolvida da programação para maiores esclarecimentos. Provinda de testes, será somente devido a um retorno para efetuar um re-trabalho. A figura 12 ilustra as dependências do setor de programação.

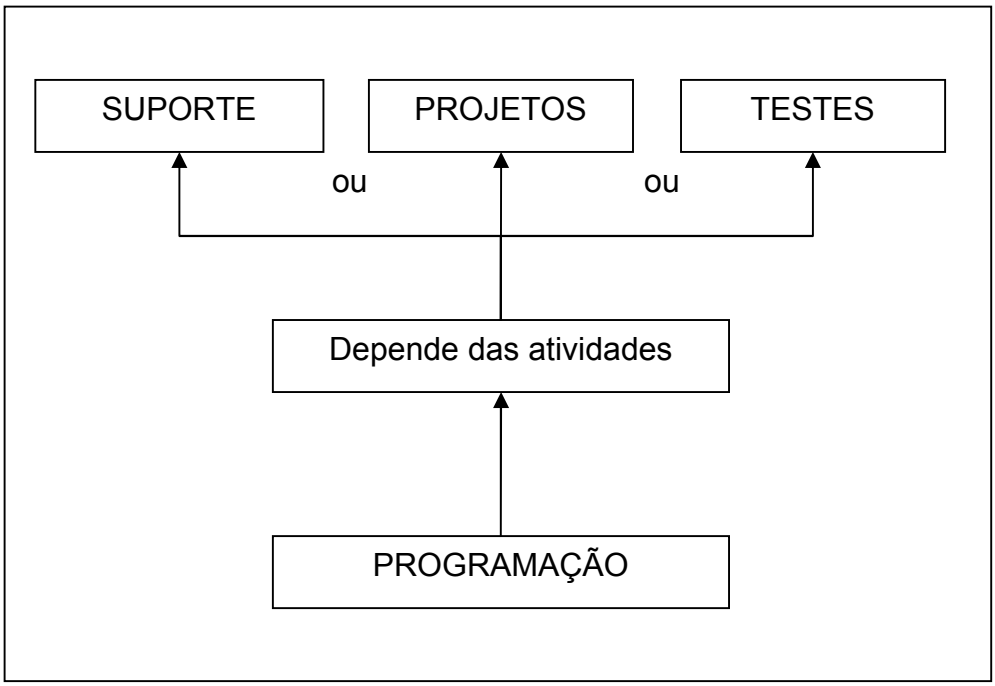

Figura 12: Dependência da atividade do setor de Programação
Fonte: Autor

A atividade de testes, por sua vez, tem seu processo ativado somente se o setor de programação lhe der seguimento, seja devido a uma OS desenvolvida ou devido a uma OS que teve retorno do próprio setor de testes para re-trabalho. A figura 13 ilustra a dependência da atividade de testes.

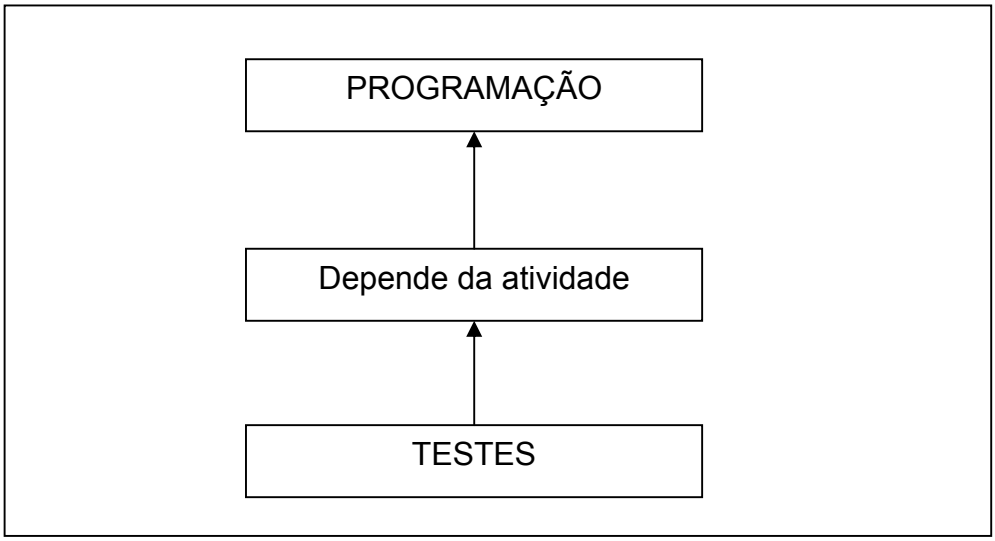

Figura 13: Dependência da atividade do setor de Testes
Fonte: Autor

Deste modo, pode-se observar que estas dependências devem servir como parâmetro para a modelagem formal das atividades do processo de desenvolvimento de *software*.

Passando a avaliar o fluxo de atividades sobre os recursos humanos, pode-se verificar que uma OS pode ser alocada para qualquer recurso humano disponível naquele setor. O único setor que prioriza recursos humanos para determinadas OSs é o setor de programação.

No setor de programação, cada programador é responsável direto por um ou mais módulos do *software*. Quando uma OS chega ao setor, a mesma vem acompanhada de um indicador que define para qual módulo esta OS tem impacto direto. Com esta informação, o responsável pelo setor de programação direciona a OS para o responsável pelo módulo em questão, ou para alguém que esteja com maior disponibilidade. Um programador não pode transferir OSs para outro programador. Pode somente devolver a OS para quem a enviou (setores de Suporte ou Projetos), ou implementá-la e enviá-la para teste.

4.3. MODELAGEM DO CENÁRIO ATUAL

Com base nas informações descritas na seção anterior, pode-se dar início aos modelos que representam o processo de desenvolvimento de *software* da empresa Desbravador.

Para realizar a modelagem utilizar-se-á o software Income. A tabela 10 ilustra como o software Income representa os elementos de redes de *Petri* em seu ambiente de modelagem.

Tabela 10: Simbologia das atividades.
Fonte: Autor

Símbolo	Significado
	Atividade que tem uma entrada e uma saída
	Atividade que tem uma entrada e pode ter vários destinos alternativos (OU Exclusivo), sendo que somente um será ativado.
	Repositório que demonstra o estado do modelo.
	Atividade que tem uma entrada e uma saída. Este símbolo demonstra que a atividade tem refinamento.

Deste modo, este capítulo apresenta o modelo do processo contemplando as atividades dos recursos setores e, em seguida, os modelos refinados, abrangendo os recursos humanos.

4.3.1. MODELAGEM DOS PROCESSOS: ATIVIDADES DOS RECURSOS SETORES

A modelagem apresentada nesta seção foi realizada buscando demonstrar a real situação do fluxo de atividades dos recursos setores, no processo de desenvolvimento de *software* da empresa Desbravador.

Como elemento de estudo, a OS é representada por uma ficha, que vai evoluindo pelo modelo através do fluxo, para chegar até um determinado lugar. Uma ficha, representando uma OS, contém várias propriedades, que tem informações relativas à vida da OS no seu fluxo. Deste modo, as informações contidas nas propriedades da ficha, podem ser informadas, ou alteradas, de acordo com a evolução da OS entre as distintas atividades que o fluxo contempla.

Abaixo, na modelagem do processo de desenvolvimento de *software*, pode-se perceber que uma OS, após uma requisição inicial provinda do cliente, pode ter duas origens. Uma tem origem pelo setor de Projetos e a outra pelo setor de Suporte. Observa-se também que, após ser criada, a mesma pode ser rejeitada, sendo finalizada na seqüência, ou pode ser transferida para o setor de programação, para que este desenvolva a OS. Após ser processada pelo setor de programação, a OS

pode ser transferida como devolução, para os setores de suporte ou de projetos, ou pode transferir para o setor de testes, que terá função de validar o trabalho efetuado pela equipe de programação. Neste último setor, após ser efetuada as devidas validações, a OS pode ser devolvida para o setor de programação, para possíveis ajustes, ou pode ser finalizada.

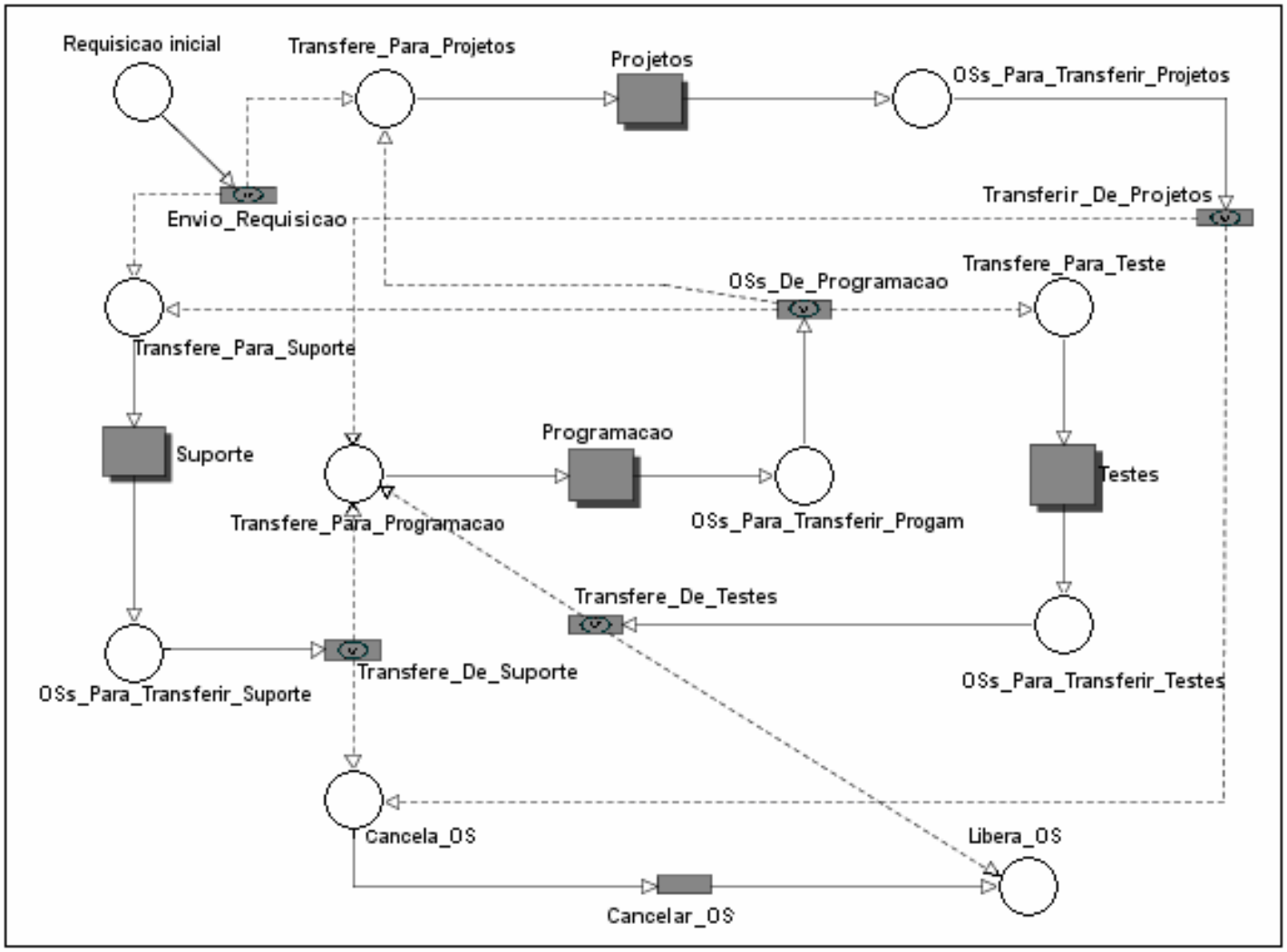

Figura 14: Modelagem do processo de desenvolvimento de *software*
Fonte: Autor

4.3.2. MODELAGEM DOS PROCESSOS: ATIVIDADES DOS RECURSOS HUMANOS

Para facilitar a visualização, e o entendimento do modelo concebido na seção anterior, algumas atividades são refinadas em novos modelos. Estes modelos são correspondentes ao refinamento das atividades dos setores Suporte, Projetos, Desenvolvimento e Testes, apresentados respectivamente nas figuras 15, 16, 17 e 18.

O primeiro modelo apresentado corresponde ao refinamento da atividade do recurso do setor Suporte. O modelo mostra que, chegando ao setor de suporte, a OS fica inicialmente em espera, até que algum recurso humano, disponível no setor, efetue a análise sobre os dados presentes na OS e dê seguimento à mesma. Após analisar a OS, o recurso humano encarregado pode rejeitá-la, registrando os motivos para tal, ou aceitá-la, registrando mais especificações para que as mesmas possam ser avaliadas pelos setores que a OS passar. Ao dar seqüência à OS, rejeitando ou transferindo para o setor de programação, o recurso humano que a estava manipulando torna-se disponível para analisar outra OS.

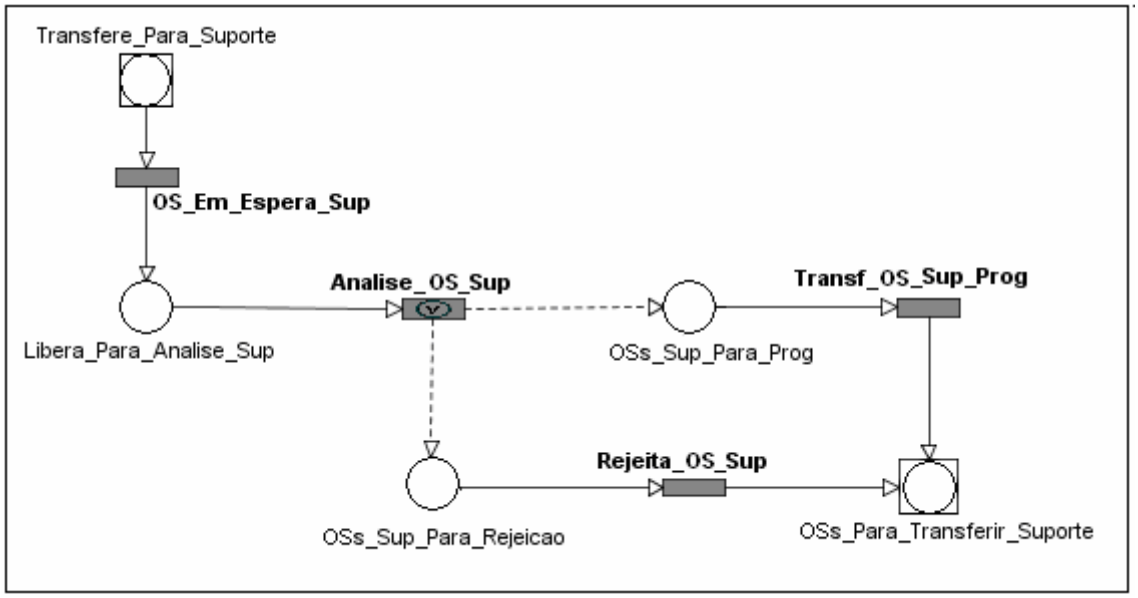

Figura 15: Modelagem refinada da atividade Suporte
Fonte: Autor

Da mesma forma, a modelagem de refinamento para a atividade do recurso projetos tem início com a OS em modo de espera, para ser avaliada por um recurso humano que esteja disponível no setor de projetos. Tão logo um recurso humano esteja liberado para analisar a OS, este o faz e pode rejeitá-la ou transferi-la para o setor de programação. Após liberar a OS o recurso humano fica disponível para analisar outra OS no setor de projetos. Este modelo é apresentado na figura 16.

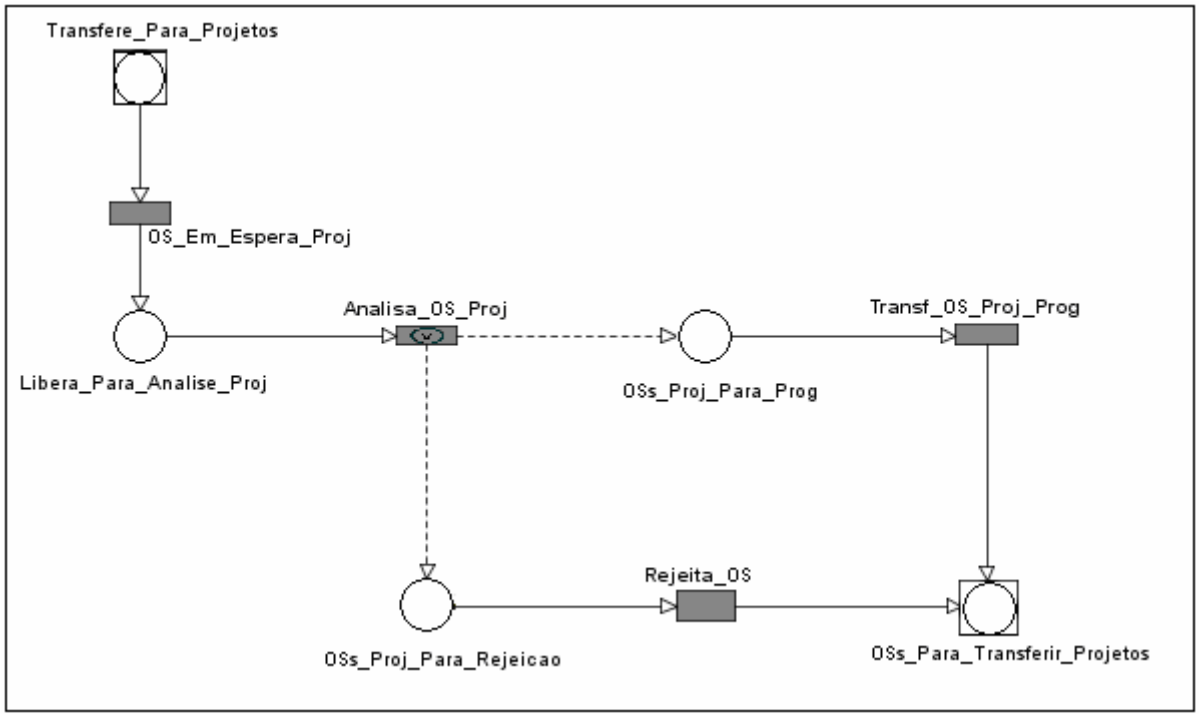

Figura 16: Modelagem refinada da atividade Projetos
Fonte: Autor

Em ambos os modelos refinados anteriormente (atividade Suporte e Projetos), ao rejeitar uma OS, esta se torna inativa, saindo do processo de desenvolvimento de *software*.

Tendo seqüência para o setor de programação, a OS pode ter destino diretamente para um programador (recurso humano do setor de programação) ou seguir sua rota específica de processamento ficando em espera, até que o supervisor analise os dados que a acompanhem. Após analisar a OS, o supervisor pode dar seqüência à mesma devolvendo para o setor de projetos ou de suporte,

para coletar mais informações sobre a mesma, ou pode também transferi-la para um programador que tenha disponibilidade para processá-la. Também o programador, caso ache conveniente, pode devolver a OS, que esteja em seu poder, para o setor de projetos ou de suporte, para aquisição de mais especificidades. Caso o programador tenha todas as informações necessárias para processar favoravelmente a OS, o mesmo o faz e prossegue transferindo a OS para o setor de testes.

Dando seqüência ao fluxo de atividades do processo de desenvolvimento de *software*, o setor de testes pode validar a OS, finalizando-a, ou devolve-la para o setor de programação, para que este reveja a OS em questão. Este modelo é apresentado na figura 17.

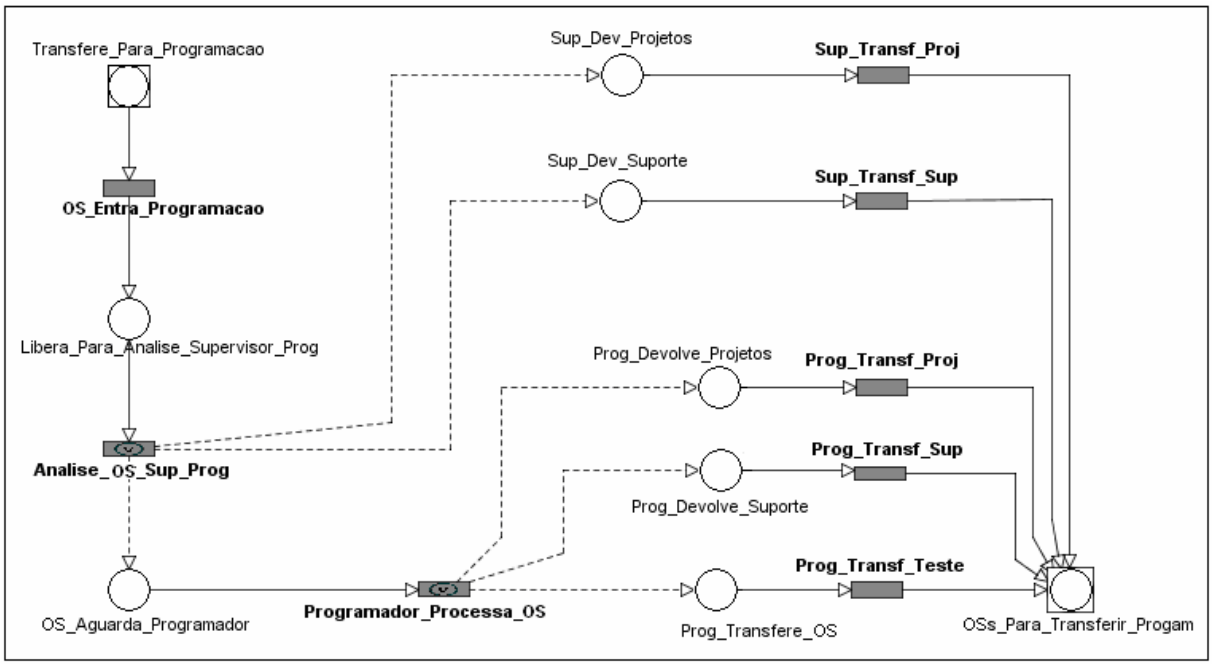

Figura 17: Modelagem refinada da atividade Programação
Fonte: Autor

Igualmente aos demais setores, ao ter entrada no setor de testes, a OS fica aguardando um recurso humano dos testes ficar disponível para começar a ser processada. Tão logo um recurso humano esteja liberado, a OS passa então para o processo de teste, sendo na seqüência finalizada ou devolvida para o setor de

programação. Ao liberar a OS, o recurso humano fica a disposição para tratar outra OS. Este modelo é apresentado na figura 18.

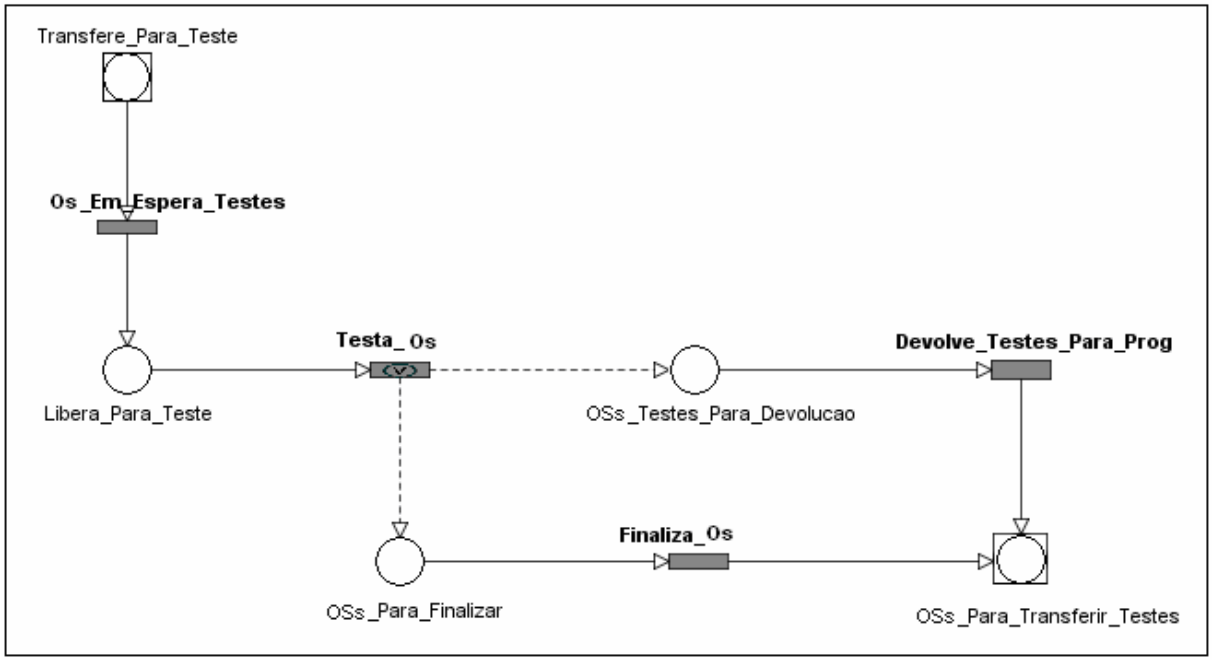

Figura 18: Modelagem refinada da atividade Testes
Fonte: Autor

4.4. ANÁLISE DO CENÁRIO ATUAL

Com base nos modelos descritos na seção anterior, parte-se para a análise para se averiguar a estrutura de forma qualitativa e quantitativa. Na análise qualitativa busca-se analisar se os modelos estão corretos em sua estrutura, observando possíveis problemas que possam estar, ou na modelagem, ou no próprio processo de desenvolvimento de *software* da empresa. Assim sendo, as propriedades de vivacidade, integridade (*soundness*) e a limitação da rede se fazem importantes nessa análise.

Com os modelos devidamente aprovados em sua forma estrutural, pode-se prosseguir efetuando a análise quantitativa, considerando os dados históricos que foram obtidos de um período de 18 meses de atividades, no intervalo das datas de 01/01/2006 até 30/06/2007. Deste modo, buscar-se-á demonstrar tais dados em correlação aos modelos apresentados acima, de forma a propiciar um melhor entendimento sobre os dados quantitativos do fluxo de atividades sobre o processo

de desenvolvimento de *software*. Para tal, serão consideradas questões como tempo de execução, capacidade de execução e ociosidade para verificação de performance e de qualidade.

4.4.1. ANÁLISE QUALITATIVA DOS CENÁRIOS ATUAIS

A análise qualitativa buscará avaliar se as estruturas dos modelos, apresentados no capítulo de modelagem, estão corretas.

Deste modo, os modelos serão dispostos sequencialmente, com suas respectivas avaliações a respeito da análise qualitativa. A modelagem do processo de desenvolvimento de *software* caracteriza-se como não tendo atividades desnecessárias nem mortas, sendo que por não ter ocorrência de *deadlocks*, nem de *livelocks*, tal modelo pode ser considerado como vivo (livre de bloqueios).

Avalia-se que para cada ficha que tiver entrada no processo, somente uma ficha sairá no final, sendo que quando a ficha chegar ao final, os demais lugares estarão limpos, sem ocorrência de fichas remanescentes. Verifica-se também que todas as atividades do modelo transportam a ficha do lugar de entrada para um lugar de saída. Pode-se considerar então que o modelo é fortemente modelado (*soundness*).

Na análise das modelagens refinadas considera-se, para todos os modelos refinados das atividades da modelagem do processo de desenvolvimento de *software*, que suas características qualitativas são equivalentes. Caracterizam-se, dessa forma, como não tendo atividades desnecessárias nem mortas, sendo que por não ter ocorrência de *deadlocks*, nem de *livelocks*, tais modelos refinados podem ser classificados como vivos (livre de bloqueios).

Verifica-se que para cada ficha que tiver entrada na atividade refinada, somente uma ficha sairá no final, sendo que quando a ficha chegar ao final, os demais lugares estarão limpos, sem ocorrência de fichas remanescentes. Observa-se também que todas as atividades dos refinamentos transportam a ficha do lugar de entrada para um lugar de saída. Pode-se considerar então que o modelo é fortemente modelado (*soundness*).

4.4.2. ANÁLISE QUANTITATIVA DOS CENÁRIOS ATUAIS

Para efetuar a análise quantitativa, sobre o cenário atual, utilizar-se-á os dados coletados no estudo realizado na empresa Desbravador. Deste modo, para se ter uma visão mais amigável dos dados, os mesmos são demonstrados em forma de gráficos, os quais serão dispostos com explicações sobre seu conteúdo para posteriores comparações.

Os dados dos gráficos de “Tipagem das OSs”, de “Severidade das OSs”, de “Prioridade das OSs” e de “Complexidade das OSs”, apresentados respectivamente nas figuras 19, 20, 21 e 22, representam os dados, de forma separada por classificação, da tabela 6, apresentada no início deste capitulo.

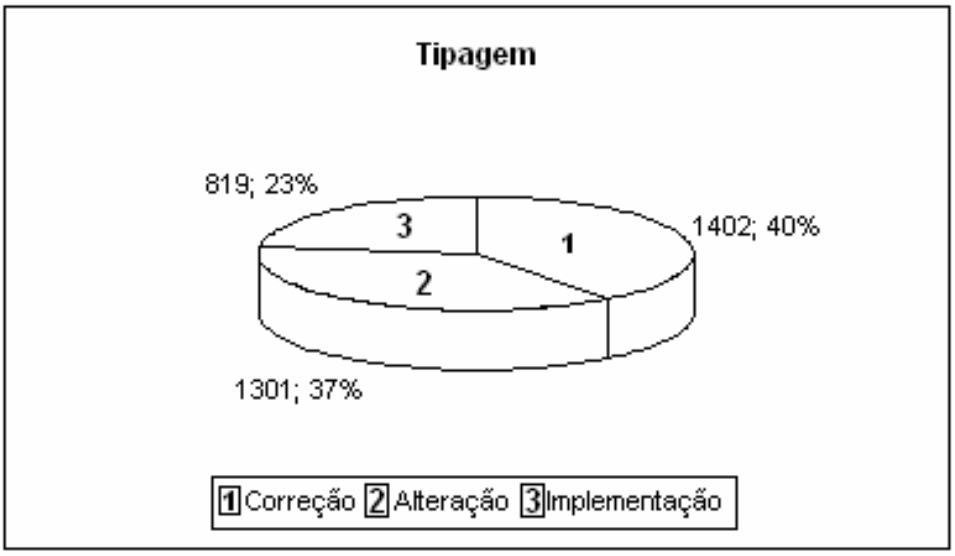

Figura 19: Gráfico de tipagem das OSs
Fonte: Autor

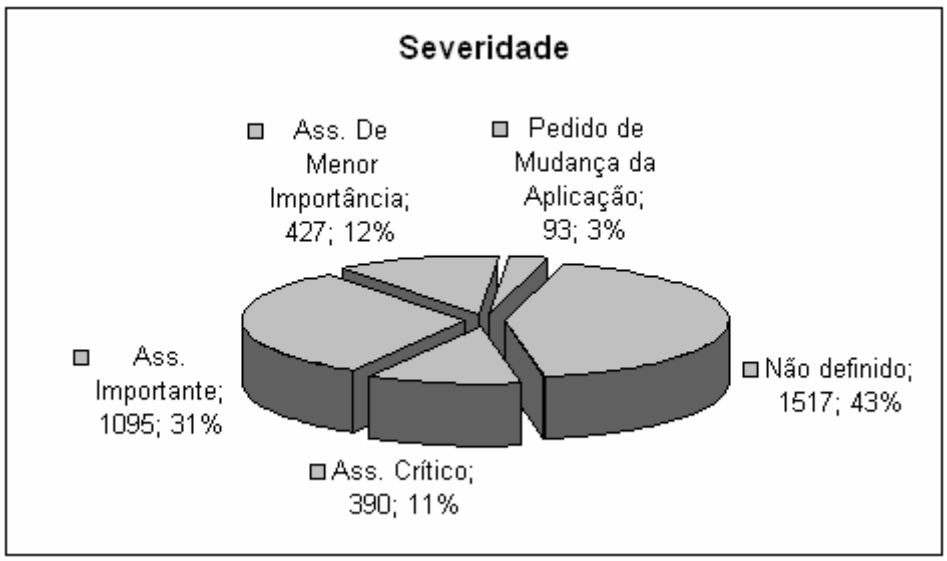

Figura 20: Gráfico de severidade das OSs
Fonte: Autor

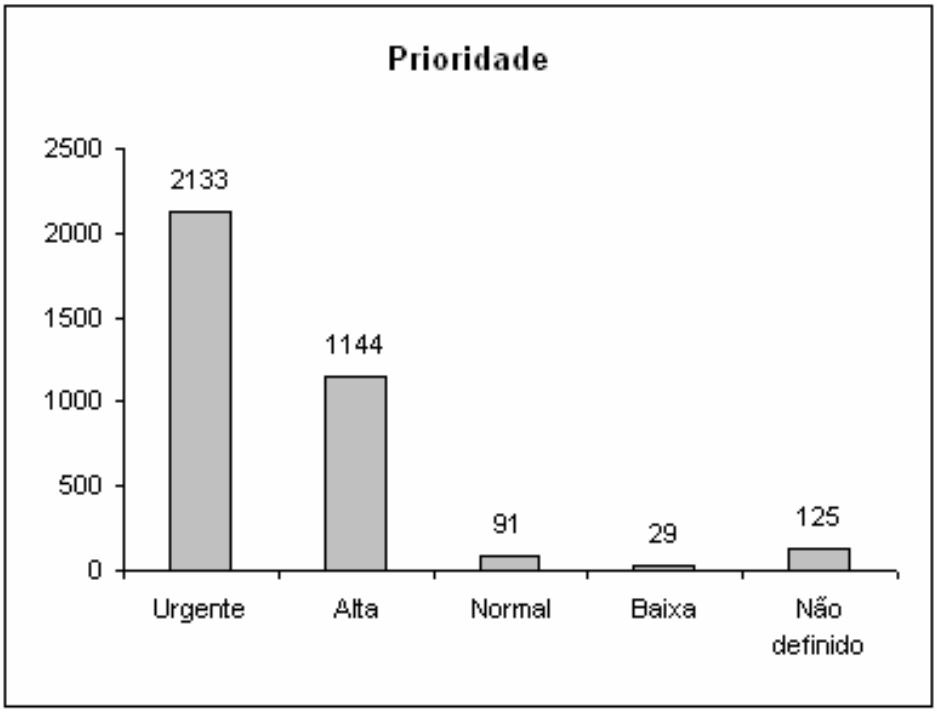

Figura 21: Gráfico de prioridade das OSs
Fonte: Autor

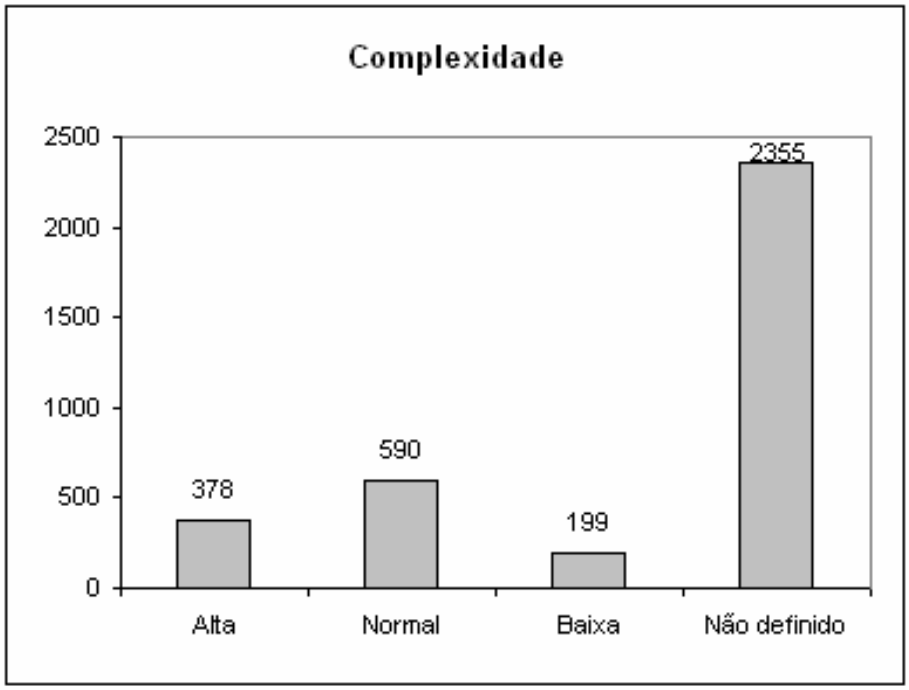

Figura 22: Gráfico de complexidade das OSs
Fonte: Autor

Visualiza-se na figura 23 o gráfico do fluxo de entradas e saídas de OSs por mês, conforme a tabela 7, de “fluxo de entradas e saídas”.

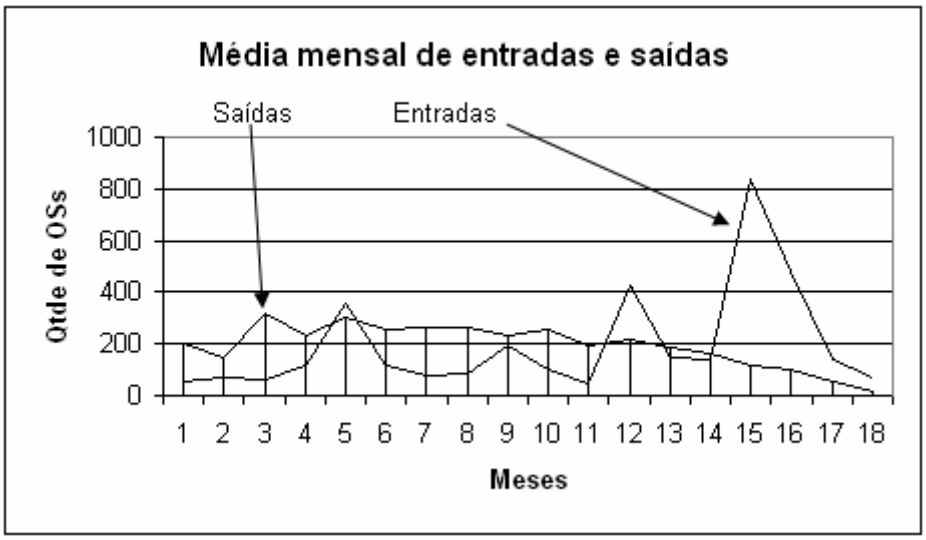

Figura 23: Gráfico de entradas e saídas mensal
Fonte: Autor

Da mesma forma, a tabela 8, de “Capacidade de execução das OSs por recurso setor”, pode ser visualizada a partir do gráfico apresentado na figura 24. Considerando a capacidade de processamento por setor, na coluna de “Média (Dias/OS)” da tabela 8, pode-se observar também como está o cenário atual, graficamente, para os recursos setores.

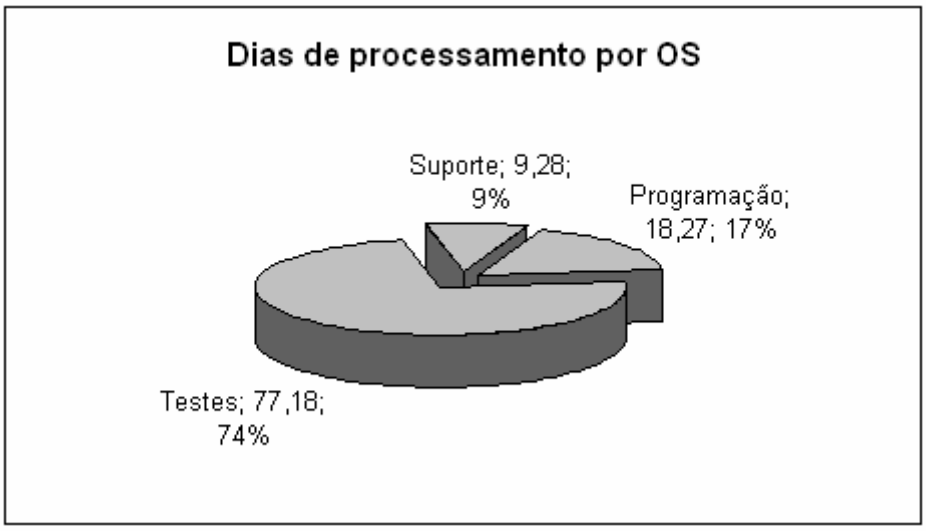

Figura 24: Gráfico de dias de processamento por OS
Fonte: Autor

Este capítulo apresentou em forma de gráficos os dados coletados na empresa Desbravador. Desta forma, a análise quantitativa se restringe a apresentar os dados históricos de forma visual, para facilitar a avaliação do cenário atual, a ser apresentada na seção 4.4.3.

4.4.3. AVALIAÇÃO DOS DADOS DO CENÁRIO ATUAL

Tendo visto em forma de gráfico os dados coletados, no estudo realizado na empresa Desbravador, pode-se realizar uma avaliação visando aprimorar o processo de desenvolvimento de *software*. Num primeiro momento realizar-se-á um comparativo visando classificar o processo de desenvolvimento de *software* real da empresa com a literatura abordada no capítulo de “Modelos de Processos de Desenvolvimento de *Software*” na “Revisão de literatura”. Neste contexto, observa-se, segundo a classificação apresentada anteriormente na tabela de “Classificação

das OSs", que as OSs atendidas pela empresa em seu processo podem ser do tipo: Correção, Alteração e Implementação.

Com tais tipos de classificação sendo tratadas pelo processo de desenvolvimento de *software*, pode-se também verificar a natureza para cada tipo de classificação, sendo:

- Correção: Tem origem em uma requisição do cliente, sendo que a mesma é oriunda de uma OS anterior, que saiu com defeito;
- Alteração: Tem origem em uma requisição do cliente, sendo que a mesma não é um defeito do sistema, e tal requisição sugere para que o *software* se comporte com distinção em certas circunstâncias;
- Implementação: Provinda de uma requisição do cliente que diz respeito a um novo recurso, que até então não era contemplada pelo *software*.

Através destas classificações, observa-se que o *software* da empresa está constantemente evoluindo, tanto corretiva (Correções em geral) quanto evolutivamente (Alterações e Implementações em geral). Caracteriza-se então que o *software* tem um processo evolutivo cíclico, como mostrado na figura 25.

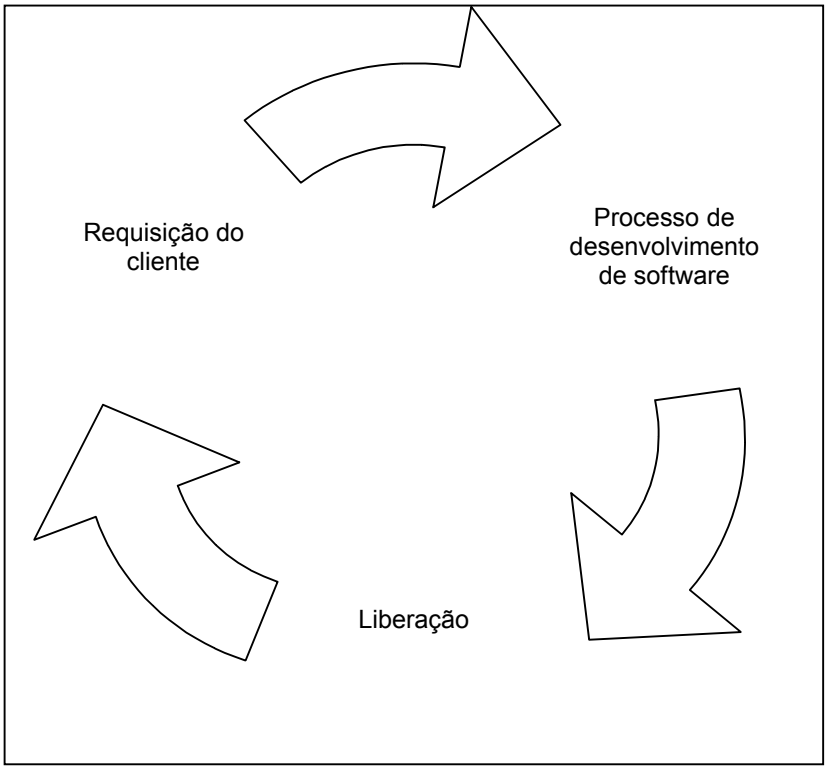

Figura 25: Processo evolutivo do *software* da empresa Desbravador
Fonte: Autor

Nesse sentido, pode-se verificar segundo Pressman (1997), que o processo real da empresa se equipara ao modelo **evolucionário incremental** dos modelos de processos de desenvolvimento de *software*. Tal modelo pode ser assim equiparado porque após passar pelo processo de desenvolvimento da empresa, um incremento à versão é gerado através da liberação. Não obstante a tal classificação como evolucionário incremental, faz-se necessário avaliar este processo de forma a verificar possíveis falhas no mesmo.

Assim sendo, uma análise pode ser realizada com base no gráfico de tipagem de OSs (figura 19). Verifica-se neste gráfico que a maior incidência representativa acontece na correção, ou seja, dos três tipos de OSs que a empresa registra a maior ocorrência é o re-trabalho.

Em outra avaliação, pode-se visualizar os gráficos de severidade (figura 20), de prioridade (figura 21) e de complexidade (figura 22). Assim sendo, o gráfico de prioridades define qual OS deveria ser prioritariamente desenvolvida, sendo que em

sua maioria absoluta verifica-se que as OSs são classificadas como urgente (sendo a maior prioridade). Verifica-se no gráfico de severidade a maioria das OSs classificadas se encontra como assunto importante. Na questão de complexidade, das OSs classificadas, a maior parte está com status de normal.

Dessa forma, efetuando-se um comparativo que visa cruzar os dados apresentados nos gráficos acima com as redes de *Petri* modeladas sobre o processo de desenvolvimento de *software* da empresa, observa-se que não existe nenhum tratamento diferenciado para as OSs que são classificadas distintamente. Assim sendo, verifica-se que uma OS com uma dada configuração específica não está sendo distintamente processada sobre o processo de desenvolvimento de *software*.

Avaliando também a performance de dias por OS do processamento por setor, visivelmente se torna verificável que o setor de testes tem seu tempo muito superior aos demais setores. Observa-se que o processo realizado pelo setor de programação, o qual efetivamente desenvolve as requisições no *software* da empresa, tem seu tempo de processamento bem inferior ao setor de testes.

Os dados coletados apresentam também um fluxo de entradas e saídas mensal de OSs, no qual a quantidade de entradas é normalmente superior às saídas. Tal superioridade é superada em alguns pontos, em que uma maior quantidade de OSs é liberada (saídas) do que a quantidade de OSs que é cadastrada (entradas).

Assim sendo, verifica-se que apesar de estar parcialmente controlado e estruturado, o processo de desenvolvimento de *software* da empresa Desbravador contém alguns gargalos que podem fazer a diferença na performance do processo como um todo.

4.4.4. DIAGNÓSTICOS PARA O CENÁRIO ATUAL

Em face das avaliações supracitadas neste tópico de análise do cenário atual, podem-se selecionar alguns elementos que dêem embasamento para a proposição de novos cenários. Assim sendo, abaixo são apresentados os elementos que podem ser considerados também como possíveis problemas para o processo.

1. ***Lead-time* para entrega**: Avaliando o fluxo de atividades como um todo, e buscando diretamente considerar o período médio decorrente para o desenvolvimento de uma OS, verifica-se que uma OS leva em torno de 105 (cento e cinco) dias para sua efetiva liberação ao cliente. Para um melhor entendimento, desde que uma requisição é realizada - e consequentemente registrada no processo de desenvolvimento de *software* da empresa Desbravador -, até sua efetiva liberação para utilização, o cliente fica esperando, em média, três meses e meio.
2. **Correções**: Outro ponto representativo, exposto na análise do atual cenário, relata que a maior incidência de OSs diz respeito a correções no *software*. Ou seja, as OSs com status de correção são derivadas de um trabalho previamente realizado (Outra OS), cujo cliente não aprovou, ou achou algum defeito, ou por questões legais o *software* passou a estar incorreto.
3. **Classificações**: Um terceiro e considerável ponto diz respeito às classificações e sua falta de tratamento pelo processo de desenvolvimento de *software* da empresa. Apesar de ter várias classificações, uma OS não tem tratamento diferenciado na evolução do processo, sendo que uma OS simplória tem o mesmo período de liberação de uma OS considerada complexa.

Considerando o primeiro elemento (lead-time para entrega), verifica-se, pela análise anteriormente exposta sobre o gráfico de dias de processamento por setor, que o setor de testes é o que mais despende tempo por OS, chegando a levar mais de oito vezes do tempo do setor de suporte e mais do que quatro vezes do tempo do setor de programação.

Com esta avaliação propõe-se como objetivo primário:

- **Reduzir o tempo de processamento por OS do setor de testes.**

Dando seqüência no segundo item, pode-se verificar que as correções, em sua maioria, são derivadas de outras OSs que foram liberadas com defeitos ou sem atender a contento as necessidades do cliente. Conforme visto em Sommerville

(2003), o retrabalho pode ser visualizado quando a colota de requisitos é defeituosa, não analisando e validando de acordo os requisitos de cliente.

Desta forma, um outro objetivo a ser almejado pela proposta de novos cenários é:

- **Melhorar a atividade de coleta de requisitos no processo de desenvolvimento de *software*.**

Assim sendo, melhorando as atividades de entrada de OSs e de testes, uma maior garantia será agregada ao desenvolvimento das OSs como um todo, o que pode impactar em uma diminuição sobre o número de OSs corretivas, pois aquelas que antes eram liberadas com defeitos, outrora serão liberadas com maior precisão.

Dando atenção também às classificações, pode-se perceber que se uma OS tiver uma classificação que seja considerada prioritária, esta poderia ter uma rota alternativa no fluxo de atividades sobre o processo de desenvolvimento de *software*. Deste modo certas OSs terão uma liberação antecipada às demais.

Deste modo, pode-se definir um terceiro objetivo ao propor novos cenários:

- **Viabilizar rotas alternativas para OSs consideradas prioritárias.**

Contudo, faz-se necessário observar os três objetivos, abaixo agrupados para melhor visualização:

1. **Reduzir o tempo de processamento por OS do setor de testes;**
2. **Melhorar a atividade de coleta de requisitos no processo de desenvolvimento de *software*;**
3. **Viabilizar rotas alternativas para OSs consideradas prioritárias.**

Objetivando diminuir o tempo de liberação das OSs, e após ter identificado os possíveis pontos problemáticos no processo, pode-se dar seqüência com a proposição de cenários. Para tal, deve-se analisar o impacto que cada objetivo pode surtir para os setores no escopo do processo. Assim sendo, ao avaliar-se o primeiro objetivo pode-se também observar que, para liberar uma OS do testes, implicitamente, uma versão do *software* deve ser gerada para liberação. Uma

geração de versão do *software* é um processo em que todas as OSs desenvolvidas, pelo setor de programação, e que estão no setor de testes, são agrupadas em uma versão definitiva para ser enviada ao cliente.

Cabe relembrar que para gerar uma versão, a empresa adota uma metodologia de liberações que tem um intervalo de tempo que varia entre três a cinco meses. Desse modo, para atender aos objetivos na proposição de um novo cenário, onde o setor de testes teria seu tempo de processamento reduzido, o intervalo de liberações também deve ser reduzido. O fato do setor de testes estar com seu tempo dilatado para entregar uma OS não representa que as OSs realmente levem esse tempo para serem testadas, mas boa parte desse tempo pode ser decorrente da espera em que a OS fica aguardando a liberação geral.

Reduzindo o tempo de liberações gerais, pode-se aplicar também o terceiro objetivo para novos cenários, pois aplicando rotas alternativas, as OSs prioritárias não ficarão esperando no setor de testes para serem liberadas para o cliente.

Da mesma forma, pode-se aplicar o segundo objetivo, que terá função de manter a qualidade no fluxo de atividades do processo de desenvolvimento de *software*. Para isso, deve-se reforçar a análise de requisições provindas do cliente, para ter certeza de que o que será desenvolvido é realmente o que o cliente deseja, assim como, melhorar a qualidade dos testes, para que uma OS não seja liberada com defeitos.

4.5. CONSIDERAÇÕES FINAIS

Buscou-se neste capítulo demonstrar o cenário atual da empresa Desbravador Automação Hoteleira. Inicialmente foi realizado um estudo para levantar os dados relativos ao processo de desenvolvimento de *software* da empresa. Com base neste estudo foi realizada a modelagem e a análise do processo como um todo, assim como, de suas atividades refinadas.

Para finalizar, fora apresentada uma relação dos possíveis problemas e suas possíveis soluções. A tabela 11 demonstra os diagnósticos levantados e as soluções propostas.

Tabela 11: Resumo dos diagnósticos e suas soluções propostas
Fonte: Autor

DIAGNÓSTICO	PROPOSIÇÃO DE SOLUÇÃO PARA O DIAGNÓSTICO
Lead-time para entrega muito dilatado	Reduzir o tempo de processamento por OS do setor de testes
Muitas ocorrências de correções	Melhorar a atividade de coleta de requisitos no processo de desenvolvimento de *software*
Classificações não têm funcionalidade no processo de desenvolvimento de *software*	Viabilizar rotas alternativas para OSs consideradas prioritárias

5. PROPOSIÇÃO DE CENÁRIOS

A proposição de novos cenários, neste capítulo apresentada, estará embasada nos diagnósticos e em suas soluções propostas, anteriormente apresentadas no capítulo 5. Buscar-se-á atender as seguintes proposições de soluções:

1. **Reduzir o tempo de processamento por OS do setor de testes;**
2. **Melhorar a atividade de coleta de requisitos no processo de desenvolvimento de *software*;**
3. **Viabilizar rotas alternativas para OSs consideradas prioritárias.**

Desse modo, buscar-se-á aqui preservar o máximo da estrutura atual, buscando maior eficiência com o menor número de alterações possíveis no processo de desenvolvimento de *software* da empresa. Dessa forma, conforme sugerido por Sneed & Brössler (2003), pretende-se pelo menos garantir que a qualidade do *software* seja preservada, se não aumentada, no PDS.

O presente capítulo descreve inicialmente a modelagem dos cenários propostos, buscando atender as proposições acima apresentadas. Para tal, os modelos do cenário atual serão aproveitados, adaptando-os para possibilitar alcançar as proposições deste capítulo. Na seqüência, com base nos modelos propostos para um novo cenário, a análise qualitativa e quantitativa é realizada, dando seguimento a uma análise comparativa entre o cenário proposto ao cenário atual. Por fim as considerações finais são apresentadas, buscando sintetizar o exposto neste capítulo.

A figura 26 demonstra como este capítulo está estruturado.

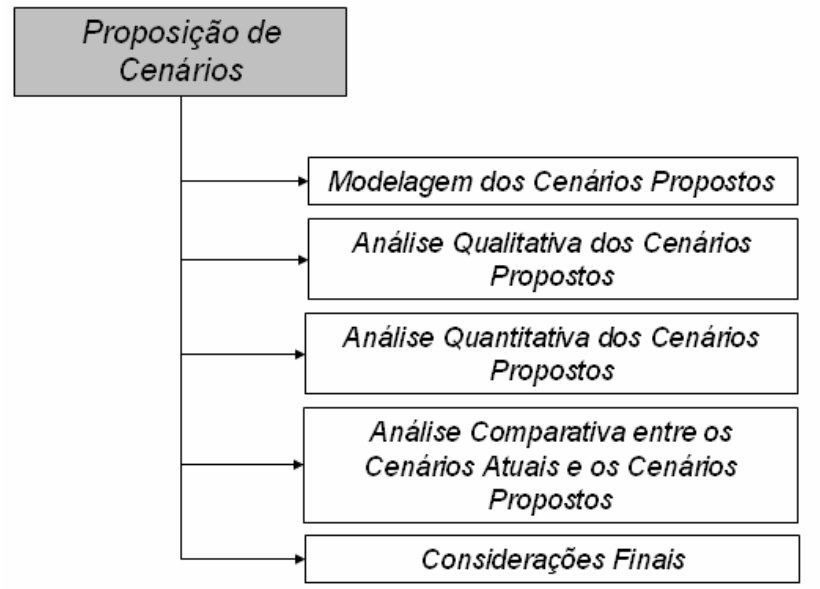

Figura 26: Estrutura da Proposição de Cenários
Fonte: Autor

5.1. MODELAGEM DOS CENÁRIOS PROPOSTOS

No processo de desenvolvimento de *software* atual da empresa estudada, verifica-se que o tempo médio para desenvolvimento de uma OS é de aproximadamente 105 dias (três meses e meio). Como objetivo primário, buscar-se-á reduzir o tempo de processamento por OS desse processo, buscando, como sugerido no diagnóstico do cenário atual, reduzir especificamente o processo do setor de testes.

Objetivo 1: Reduzir o tempo de processamento por OS do setor de testes

Pode-se, para este objetivo, reduzir o tempo de processamento do setor de testes reduzindo o tempo de liberação. Tempo de liberação é o tempo de processamento da OS em todo o PDS, considerando como início a criação da OS e como fim a finalização da OS.

Assim sendo, dentre as atividades refinadas para o setor de testes, verifica-se que a atividade “Finaliza_OS” é uma atividade que tem um retardo em sua

execução, de forma que esse retardo pode impactar negativamente na média de dias para processamento de uma OS. A figura 27 apresenta o modelo atual do setor de testes para melhor visualização.

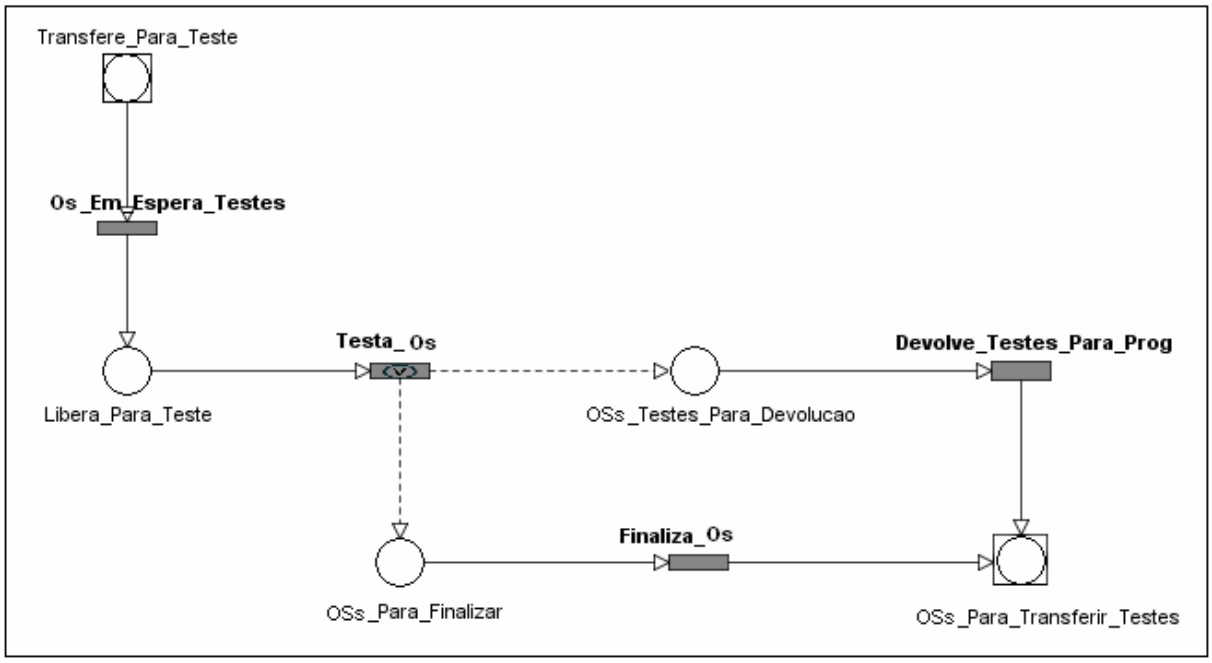

Figura 27: Modelagem refinada do setor de Testes: Cenário atual
Fonte: Autor

Como visto no diagnóstico do cenário atual, a atividade do setor de teste despende muito mais tempo do que as demais atividades setoriais, sendo que sua média de dias por OS, exposta na tabela 8, é de aproximadamente 77 dias.

Com o objetivo a reduzir o tempo dessa atividade, cabe aqui observar que o processo de testes tem um retardo (espera para finalizar a OS) em seu refinamento de atividades, que pode ser relacionado à atividade “Finaliza_OS”. Essa atividade tem por função manter a OS parada, em espera, para sincronizar todas as OSs em uma liberação única. Dessa forma, pode-se propor que esse retardo seja menor. Para isso, os seguintes itens devem ser observados:

- Efetuar liberações gerais do *software* em um intervalo de tempo (dias) menor;
- Estipular um período máximo para a atividade Testa_OS;

- Estipular, com base no prazo das liberações, um retardo para a atividade Finaliza_OS.

Pode-se, então, propor valores para o intervalo de tempo das liberações gerais (Tempo de liberação geral), para o tempo máximo da atividade de testes e para o retardo na atividade “Finaliza_OS”, como segue na tabela 12:

Tabela 12: Proposição de valores para novo cenário de testes
Fonte: Autor

VARIÁVEL	VALOR ATUAL / OS em dias	VALOR PROPOSTO / OS em dias
Tempo de liberação geral	104,74	60
Tempo máximo de teste	N/D	15
Tempo de retardo para “Finaliza_OS”	N/D	17,45

A tabela 12 mostra que no atual cenário da empresa, não se tem meta estipulada para efetuar a atividade de testes, tampouco da atividade Finaliza_OS. Considerarando que o tempo médio que o setor de programação leva para processar uma OS é de 18,27 dias, e que o setor de suporte, em média, leva aproximadamente 9,28 dias, pode-se considerar que o setor de testes teria, com base no tempo de liberação geral proposto de 60 dias, uma disponibilidade de tempo equivalente a um total de 32,45 dias para executar sua atividade.

Ao estipular-se um limite de tempo inferior ao tempo atual da liberação geral, assim como tendo ciência do tempo que as demais atividades demandam no cenário atual, é possível definir um tempo máximo para a atividade de teste, assim como para a atividade Finaliza_OS. Observa-se que o retardo contido na atividade Finaliza_OS acaba por se tornar uma conseqüência das demais atividades, definida, principalmente, pela variável tempo de liberação geral.

No cenário proposto para o setor de testes, verifica-se que estruturalmente o modelo não teve alterações, porém, a atividade Finaliza_OS, que antes tinha um tempo de retardo desconhecido, definido pela liberação geral, agora está definida de acordo com alguns parâmetros, conforme demonstrado acima na tabela 12.

Atingido o primeiro objetivo, pode-se dar seqüência no tratamento do segundo objetivo, sendo:

Objetivo2: Melhorar a atividade de coleta de requisitos no processo de desenvolvimento de *software*

Conforme Palyagar & Richards (2005), um dos principais problemas encontrados ao se coletar requisitos é a falta de comunicação. Da mesma forma, Zanlorenci & Burnett (2003) reforçam que o usuário de sistema deve ter participação na validação dos requisitos.

Desta forma, para possibilitar uma melhora na qualidade na coleta de requisitos e sua validação, propõe-se uma interação com o cliente no momento da análise da solicitação. Quando o setor de suporte, ou de projetos, recebe uma nova requisição, este deve proceder de forma a quantificar ao máximo as informações sobre a nova requisição, minimizando, assim, a possibilidade de que a requisição seja mal interpretada nas demais atividades do processo de desenvolvimento de *software*.

Assim sendo, as atividades refinadas dos setores de suporte e de projetos terão atenção nesta proposição. Para tal, os seguintes itens devem ser tratados:

- Criar atividade, de vínculo direto entre a análise da OS e o cliente, para a atividade refinada do setor de suporte;
- Criar atividade, de vínculo direto entre a análise da OS e o cliente, para a atividade refinada do setor de projetos;

A figura 28 apresenta o cenário proposto da atividade refinada do setor de suporte.

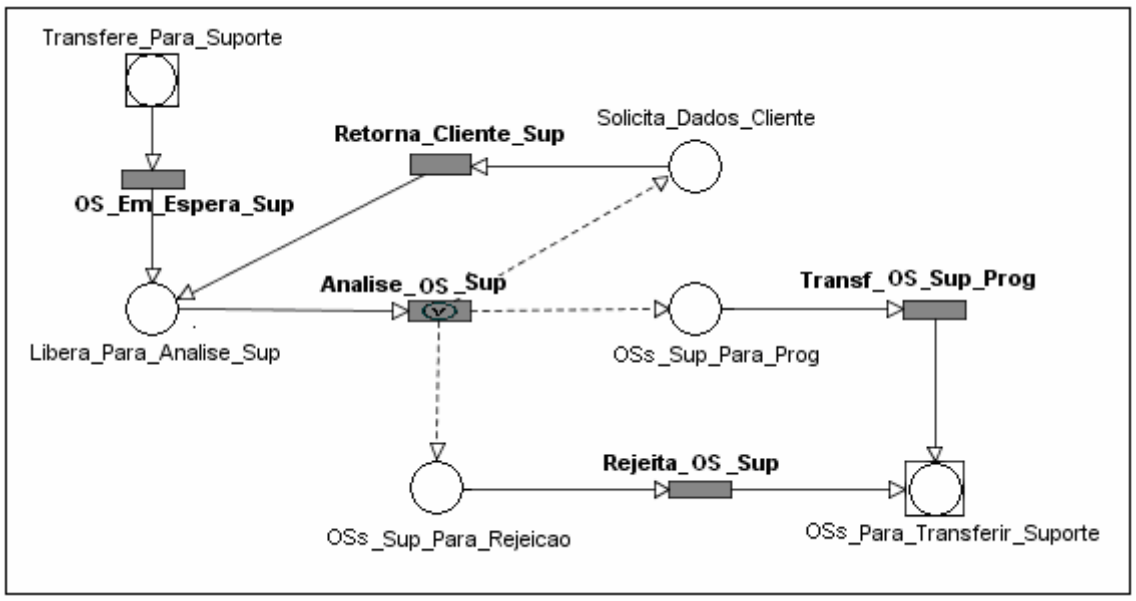

Figura 28: Proposição de modelagem refinada do setor Suporte
Fonte: Autor

Observa-se que na proposição refinada do setor suporte, o que difere do cenário atual é a interação explicitada pela ligação da atividade “Analisa_OS_Sup” com o cliente que realizou a requisição, através da atividade “Retorna_Cliente_Sup”. Da mesma forma, a figura 29 apresenta o cenário proposto para a atividade refinada do setor de projetos.

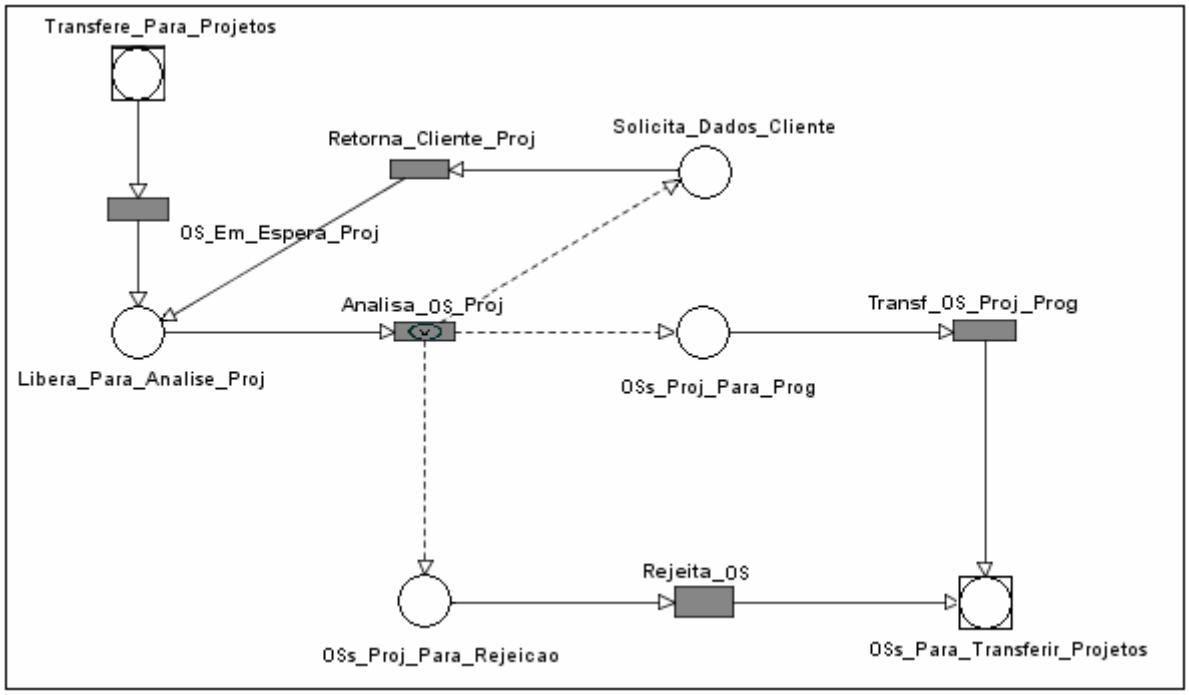

Figura 29: Proposição de modelagem refinada do setor Projetos
Fonte: Autor

Observa-se que, da mesma forma que a proposição de cenário para a atividade refinada de suporte, a proposição refinada do setor projetos, difere do cenário atual na interação explicitada pela ligação da atividade “Analisa_OS_Proj” com o cliente que realizou a requisição, através da atividade “Retorna_Cliente_Proj”.

Após efetuar as alterações para alcançar o segundo objetivo, podemos avaliar o terceiro.

Objetivo 3: Viabilizar rotas alternativas para OSs consideradas prioritárias

Uma OS pode ser classificada com cinco itens classificadores, demonstrado na tabela 2. Cada item classificador pode ser valorado de acordo com suas opções. Considerando que cada item classificador pode ser combinado com os demais itens classificadores, observa-se que uma OS pode conter um vasto número de combinações em sua classificação. Devido à vasta gama de combinações que uma OS pode conter em sua classificação, e buscando possibilitar um tratamento

diferenciado para uma determinada classificação, buscaremos aqui modelar uma rota alternativa à atual, a qual deverá servir como exemplo para trabalhos futuros.

Dessa forma, pretende-se selecionar uma classificação que deve ter um tratamento diferenciado para que possa ser liberada com maior antecedência às demais.

Assim sendo, na tabela 13 é selecionada a classificação específica para a proposição de uma rota alternativa. A opção em negrito é a que deve ser considerada para a rota alternativa.

Tabela 13: Classificação selecionada para proposição de cenário
Fonte: Autor

Classificação	Opções
Congelada	*Sim* *Não*
Tipo	***Correção*** *Alteração* *Implementação*
Severidade	*Assunto crítico* *Assunto importante* *Assunto de menor importância* *Pedido de mudança de aplicação*
Prioridade	*Urgente* *Alta* *Normal* *Baixa*
Complexidade	*Alta* *Normal* *Baixa*

Dessa forma, a OS enquadrada como sendo do tipo “Correção”, deve ter um tratamento diferenciado no processo de desenvolvimento de *software*. Pode-se interpretar que caso uma OS esteja classificada como correção, independente das demais classificações, esta deve ter uma rota alternativa que lhe dê preferência diante das demais. Isso porque, como visto no levantamento dos dados, existe um grande número de re-trabalho no processo de desenvolvimento de *software* atual, sendo 1402 OSs corretivas sobre o total de 3522. Por conseguinte, verifica-se que estas OSs, por serem um re-trabalho, devem ser processadas preferencialmente.

Dessa forma, o modelo apresentado na figura 30 representa uma proposição de uma rota alternativa para a classificação acima apresentada na atividade de programação.

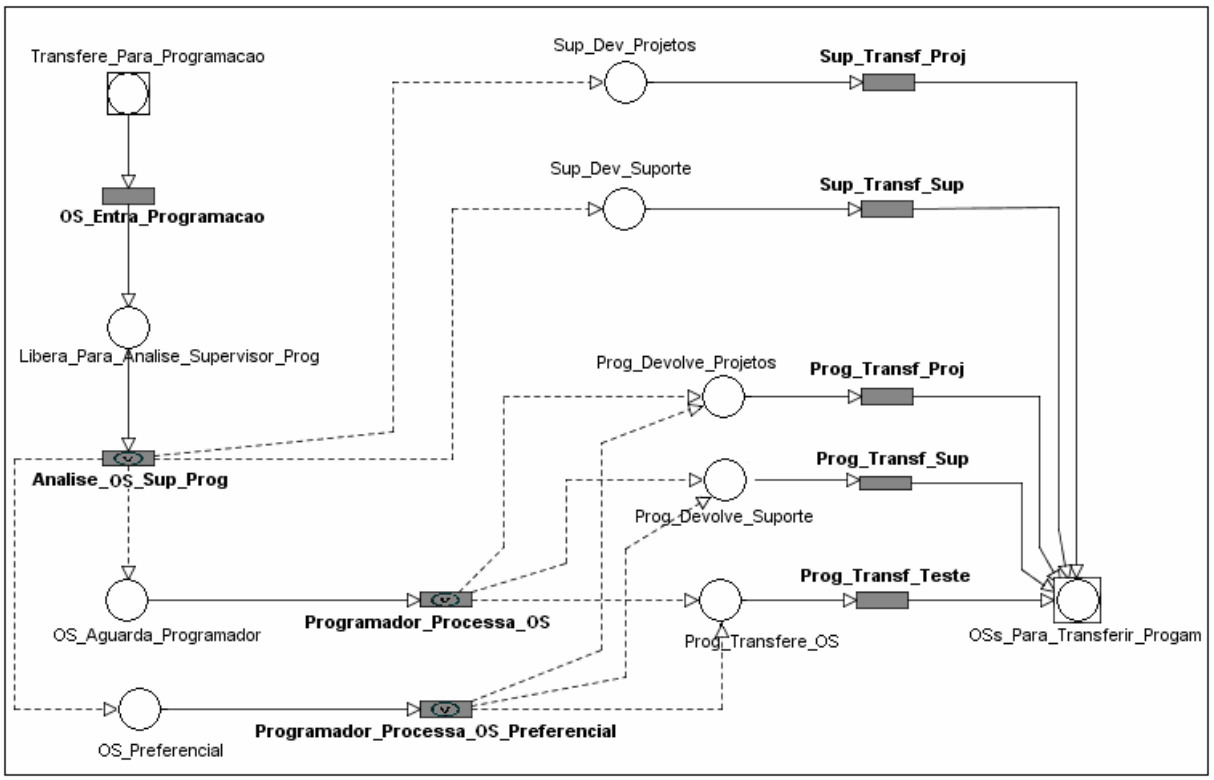

Figura 30: Proposição de rota alternativa refinada da atividade Programação
Fonte: Autor

Como proposta de rota alternativa no refinamento da atividade do setor de programação, é criada a atividade “Programador_Processa_OS_Preferencial”, que é executada com preferência sobre a atividade “Programador_Processa_OS”, que antes era executada indistintamente para todas as OSs. Dessa forma, ao aparecer uma OS no repositório “OS_Preferencial”, esta deve ser processada pelos recursos humanos do setor de programação antes das que estão no repositório “OS_Aguarda_Programador”. Dessa forma, as OSs consideradas preferenciais poderão ter sua liberação antecipada, de acordo com o prazo de liberações proposto como primeiro objetivo deste capítulo.

5.2. ANÁLISE QUALITATIVA DOS CENÁRIOS PROPOSTOS

Neste tópico será apresentada a análise qualitativa do modelo proposto para o processo de desenvolvimento de *software*, e na seqüência, a análise qualitativa sobre os modelos refinados das atividades do processo de desenvolvimento de *software*.

5.2.1. Análise da modelagem proposta do processo de desenvolvimento de *software*

A modelagem proposta do processo de desenvolvimento de *software* caracteriza-se como não tendo atividades desnecessárias nem mortas, sendo que por não ter ocorrência de *deadlocks*, nem de *livelocks*, tal modelo pode ser considerado como vivo (livre de bloqueios).

Verifica-se também que para cada ficha que tiver entrada no processo, somente uma ficha sairá no final, sendo que quando a ficha chegar ao final, os demais lugares estarão vazios, sem a presença de fichas remanescentes. Pode-se verificar também que todas as atividades do modelo transportam a ficha do lugar de entrada para um lugar de saída. Pode-se considerar então que o modelo é fortemente modelado (*soundness*).

5.2.2. Análise dos modelos refinados propostos

Pode-se considerar, para todos os modelos refinados das atividades da modelagem proposta do processo de desenvolvimento de *software*, que suas características qualitativas são equivalentes. Caracterizam-se, dessa forma, como não tendo atividades desnecessárias nem mortas, sendo que por não ter ocorrência de *deadlocks*, nem de *livelocks*, tais modelos refinados podem ser classificados como vivos (livre de bloqueios).

Observa-se também que para cada ficha que tiver entrada na atividade refinada, somente uma ficha sairá no final, sendo que quando a ficha chegar ao final, os demais lugares estarão vazios, sem a presença de fichas remanescentes. Pode-se verificar também que todas as atividades dos refinamentos transportam a ficha do

lugar de entrada para um lugar de saída. Considera-se então que o modelo é fortemente modelado (*soundness*).

5.3. ANÁLISE QUANTITATIVA DOS CENÁRIOS PROPOSTOS

Buscando quantificar o tempo de execução do modelo proposto, realiza-se aqui a análise baseada em uma simulação, considerando o período de 18 meses e a quantidade de OSs de 3522, os quais correspondem ao estudo realizado na empresa.

Assim sendo, para realizar a simulação, foi utilizado o software Income, com os modelos do cenário proposto (tópico 5.1), e os dados da tabela 14 como dados de entrada para execução. A tabela 14 apresenta o tempo de execução por setor a ser utilizado como proposição para ser simulado sobre o modelo proposto.

Tabela 14: Proposição de capacidade de execução das OSs por recurso setor
Fonte: Autor, 2007

Capacidade de Execução: (01/01/2008 até 30/06/2009)		
Setor	**Núm. de OSs**	**Média (Dias/OS)**
Projeto	*0*	**0,00**
Programação	*3.500*	**18,27**
Testes	*3.500*	**32,45**
Suporte	*3.502*	**9,28**
Média de dias/OS no período: 60,00		

Com base nestes dados, observa-se que para desenvolver uma OS no processo proposto, a média de dias seria de 60 dias. Da mesma forma, se consideradas as 3522 OSs, verifica-se que o tempo acumulado de processamento médio das OSs será de 211.320 dias.

A proposição de melhoria na qualidade através de uma maior interação entre o setor de suporte, ou o setor de projetos, com o solicitante (cliente), não deve ter impacto no tempo de processamento desse setor, pois sua intenção é de somente agregar qualidade e confiabilidade aos requisitos da OS. Tal atividade deve ser aplicada no momento da análise da OS pelo recurso humano que estiver processando a OS.

Entretanto, pode-se avaliar que o impacto se dará no número de OSs corretivas no decorrer do tempo. Se a OS estiver melhor definida, com maior rigor para determinar detalhadamente as necessidades do solicitante, consequentemente a atividade de programação terá maior êxito em sua atividade, sendo que a liberação final passará a ter maior probabilidade de acerto sobre as OSs desenvolvidas. Dessa forma, verifica-se uma tendência de diminuição de OSs corretivas através do tempo.

Ao realizar a simulação à alteração proposta para considerar uma rota alternativa para OSs corretivas, apresentada na figura 30, pode-se verificar que a capacidade de liberação no cenário proposto terá um maior número de OSs que poderão ser liberadas no prazo estipulado de 60 dias. Nessas condições, considerando a tabela 14 como dados de entrada, o total de OSs corretivas de 1402, apresentada na tabela 6, poderia ser processada em uma média de 60 dias por OS, sendo que totalizaria um acumulado de dias de 84.120 dias de processamento.

5.4. ANÁLISE COMPARATIVA ENTRE OS CENÁRIOS ATUAIS E OS CENÁRIOS PROPOSTOS

Para poder-se ter um maior entendimento sobre os benefícios expostos na análise do cenário proposto, faz-se necessário contrapor os dados obtidos na simulação do cenário proposto com os dados estudados no cenário atual da empresa.

Assim sendo, analisando o tempo de processamento do processo, pode-se verificar que cada uma das OSs leva uma média de 60 dias (tabela 14) para seu processamento no cenário proposto, contrapondo com os 104,74 dias (tabela 8) do cenário atual do processo de desenvolvimento de *software* da empresa. As 3522 OSs seriam processadas em um total acumulado de 211.320 dias, em contrapartida aos 368.910,76 dias que demandam o cenário atual. No cenário proposto, avaliando o período de 18 meses, teríamos uma economia média de 44,74 dias por OS, sendo que o acúmulo de economia para todas as 3522 OSs seria de 157.590,76 dias. As figuras 31 e 32 apresentam dois gráficos que ilustram o comparativo de dias entre o cenário atual e o proposto. A figura 31 considera dados da tabela 8 para o cenário

atual e tabela 14 para o cenário proposto. A figura 32 utiliza-se de dados apresentados neste parágrafo.

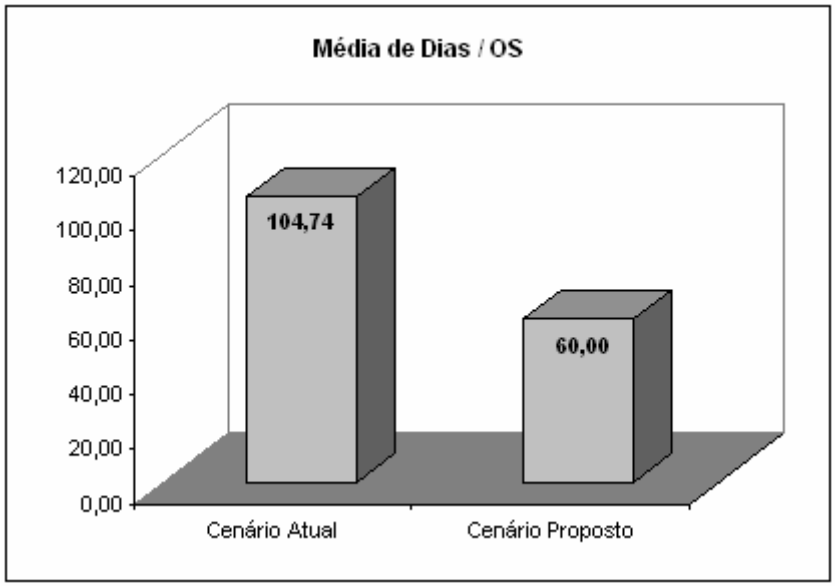

Figura 31: Comparativo de média de dias / OS
Fonte: Autor

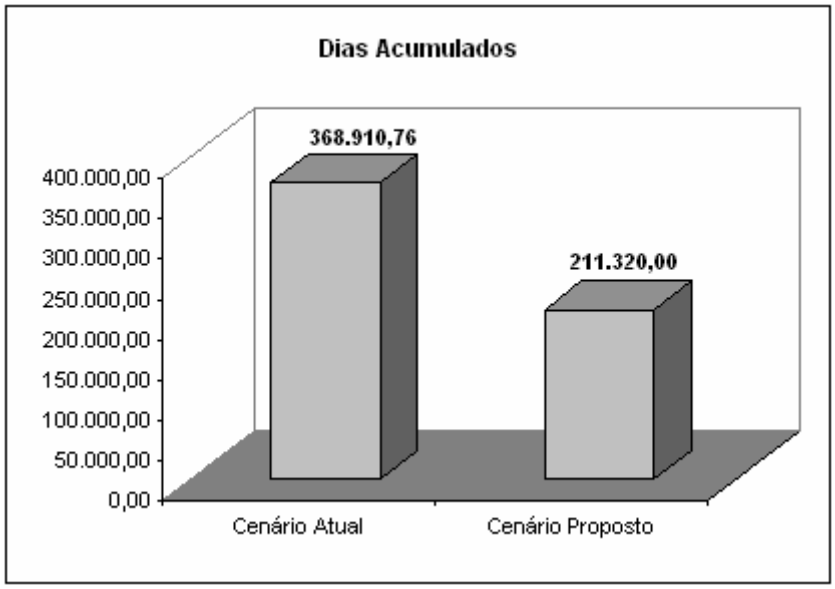

Figura 32: Comparativo de dias acumulado para desenvolvimento
Fonte: Autor

A diferença apresentada nas figuras 31 e 32, relativas ao cenário atual para o cenário proposto, não representam de fato que os recursos devam ficar ociosos,

mas pode-se afirmar que o solicitante (cliente que efetuou a solicitação) poderá ser atendido em um período menor de tempo.

Na qualidade do processo, pode-se verificar que no cenário proposto foi dado ênfase, nos setores de suporte e de projetos, a uma atividade de interação com o solicitante, para obter maior garantia no decorrer da OS pelo processo de desenvolvimento de *software*. O cenário atual da empresa não tem uma ênfase para tal atividade, de modo que a solicitação, ao chegar à empresa, é avaliada e repassada adiante sem ter uma garantia sobre o que realmente o solicitante deseja.

Assim sendo, pode-se caracterizar que no cenário atual não existe um controle preventivo para minimizar a ocorrência de OSs corretivas. Devido à proposição do novo cenário buscar informações junto ao solicitante para garantir qualidade às OSs, observa-se que surge uma tendência favorável a redução de ocorrência de OSs corretivas ao longo do tempo. A figura 33 demonstra tal tendência de forma comparativa, entre o cenário atual e o cenário proposto.

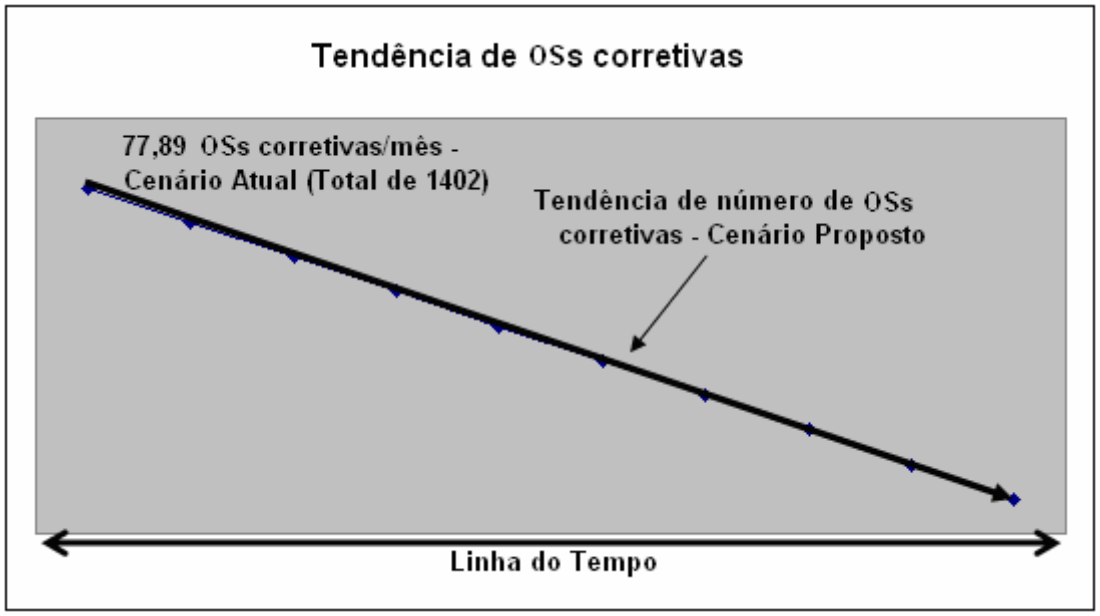

Figura 33: Tendência comparativa de ocorrência de OSs corretivas
Fonte: Autor

De forma similar, no cenário atual, pode-se verificar que as classificações não têm maior importância para a evolução do processo. Assim sendo, ao ser considerado um cenário em que as classificações das OSs são tratadas de forma

diferenciada no PDS, ditando como a OS é processada pelas atividades do processo, percebe-se que o processo como um todo se beneficia em agilidade.

Sabendo que o prazo estipulado de 60 dias terá maior eficiência com um fluxo de atividades alternativo para OSs preferenciais, observa-se que haverá uma minimização no processamento de dias acumulado de 146.845,48 do cenário atual para 84.120,00 do cenário proposto. A figura 34 demonstra esta minimização.

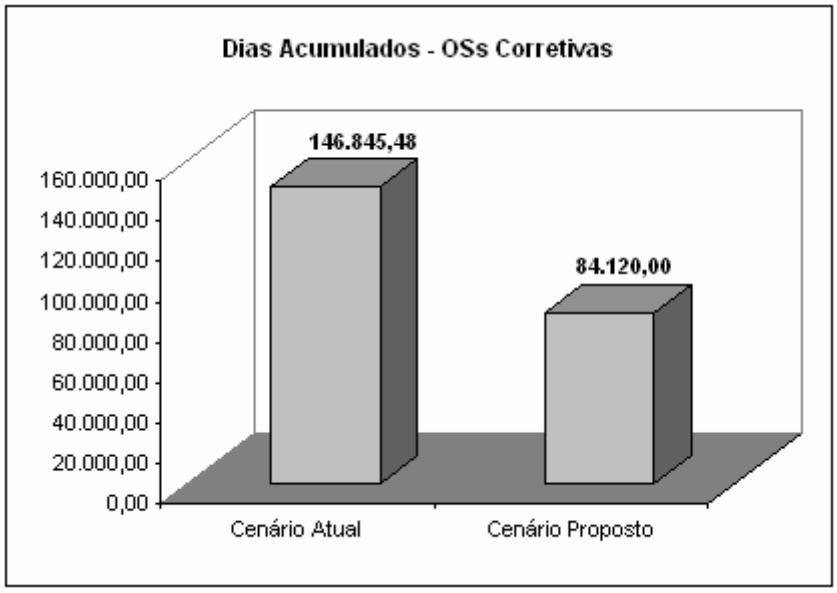

Figura 34: Comparativo de dias acumulados para processamento de OSs corretivas
Fonte: Autor

Na tabela 15 pode ser observado tabularmente o comparativo entre o cenário atual e o cenário proposto. Pode-se visualizar três colunas, sendo o assunto, o cenário atual e o cenário proposto. Nesta tabela as proposições foram colocadas como assunto, os dados do cenário atual, relativo ao assunto, foram apresentados buscando realizar um comparativo com o cenário proposto para o assunto.

Dessa forma, pode-se visualizar que para o tempo de processamento médio (em dias), o cenário atual apresenta 104,74 dias, enquanto que no cenário proposto esses dias foram reduzidos para 60. Na garantia de qualidade, enquanto o cenário atual está centrado somente no seu processo interno da empresa, no cenário proposto busca-se realizar interações com os clientes, buscando melhorar e garantir a qualidade da solicitação (se o que o cliente quer é o que será desenvolvido). Na questão de flexibilidade, o cenário atual busca ser flexível em sua estrutura e no seu

fluxo de atividades, enquanto o cenário proposto apresenta uma flexibilidade que busca considerar as classificações das OSs.

Tabela 15: Comparativo entre cenários
Fonte: Autor, 2007

Assunto	Cenário atual	Cenário proposto
Tempo de processamento médio (em dias)	104,74	60
Garantia de qualidade	Centrado apenas no processo interno da empresa	Além de estar focado no processo da empresa, busca-se obter interações com o solicitante (cliente) para certificação de suas necessidades.
Flexibilidade	Flexível em sua estrutura e no seu fluxo de atividades	Além da flexibilidade atual, uma rota alternativa fora proposta às OSs com determinadas classificações, abrangendo não somente a estrutura do fluxo de atividades, mas também a forma com que as atividades processam as OSs.

Pode-se também classificar as alterações propostas para o novo cenário de acordo com seus benefícios, em relação ao cenário atual, apresentada na tabela 16. Nesta tabela é apresentada a alteração realizada sobre o cenário atual para chegar ao cenário proposto, sendo que os benefícios são apresentados para cada alteração. Pode-se visualizar que na alteração de tempo de processamento, um menor tempo de processamento para cada OS será obtido, sendo que o cliente se benificiará, pois sua solicitação será atendida em menor tempo. Na alteração de garantia de qualidade, uma maior garantia de que a OS será implementada com sucesso, e um menor número de Correções futuras serão resultantes. Na alteração

de flexibilidade o processo terá o benefício de ser mais ágil e confiável em sua evolução.

Tabela 16: Benefício agregado ao cenário proposto
Fonte: Autor, 2007

Alteração	Benefício
Tempo de processamento	• Menor tempo de processamento para cada OS; • Solicitante (Cliente) tem sua solicitação atendida em menor tempo.
Garantia de qualidade	• Maior garantia de sucesso na evolução do processo de desenvolvimento de *software*; • Diminuição de ocorrências de solicitações do tipo "Correção" (Retornos de OSs anteriores que foram liberadas com defeito).
Flexibilidade	• Torna o processo mais ágil e confiável.

Em contrapartida faz-se necessário relacionar alguns pontos que podem sofrer impactos negativos no cenário proposto, conforme ilustrado na tabela 17. Esta tabela demonstra as alterações e seus impactos negativos sobre o PDS. Na primeira alteração, de tempo de processamento, o software terá como impacto negativo o fato de ter que ser liberado em um prazo menor, de 60 dias, o que requer um controle sobre as liberações de versões mais apertado. Na alteração de garantia de qualidade, pode-se visualizar que um maior comprometimento será exigido dos recursos humanos dos setores de suporte e de projetos, para que a interação com o cliente se realize de forma satisfatória. Na alteração de flexibilidade, o fator negativo também é o fator humano, ou seja, os recursos humanos do setor de programação devem ter maior atenção ao processar as OSs, para que as classificações das OSs direcionem qual OS deve ser tratada prioritariamente.

Tabela 17: Impactos negativos do cenário proposto

Fonte: Autor, 2007

Alteração	**Impactos negativos**
Tempo de processamento	• O *software* deve ter sua versão liberada no prazo proposto de 60 dias. Isso requer um controle de versão mais apertado, o que deve ser cuidadosamente avaliado pelo setor de programação, que realiza o empacotamento de todos os fontes alterados em uma única versão para ser liberada.
Garantia de qualidade	• Requer um maior comprometimento dos recursos humanos dos setores de suporte e de projetos, para que a coleta de dados com o cliente seja efetuada de forma proveitosa.
Flexibilidade	• Os recursos humanos do setor de programação devem ter maior atenção ao selecionar as OSs que devem ser processadas, ou retardadas, de acordo com sua classificação (Pode ter auxílio de um *software* para esta atividade).

5.5. CONSIDERAÇÕES FINAIS

Neste capítulo buscou-se propor novos cenários para o processo de desenvolvimento de *software* da empresa Desbravador. Para isso, a análise efetuada sobre o processo atual da empresa foi tomada como ponto de partida, através dos diagnósticos e suas soluções propostas.

Na seqüência foi realizada a análise qualitativa e quantitativa do cenário proposto. Verificou-se na análise qualitativa que os modelos continuam sendo

seguros e fortemente modelados. Na análise quantitativa observou-se que o setor de testes teve sua atividade regrada por tempo, para que o PDS como um todo possa ser efetuado em menor tempo.

Por fim, um comparativo entre o cenário proposto e o cenário atual foi realizado. Neste comparativo pode-se observar uma significativa melhora de determinados pontos do processo. Por outro lado, alguns fatores negativos foram apresentados, visando servir como alerta, para que o cenário proposto tenha êxito.

6. CONCLUSÃO

Observamos neste trabalho que, efetuando um estudo sobre um processo de desenvolvimento de *software* de uma empresa real, o formalismo de redes de *petri* pôde ser utilizado para busca de melhorias sobre o processo. Dessa forma, ao efetuarmos a modelagem e a análise sobre o processo, foi possível propor melhorias ao cenário atual da empresa analisada.

Buscou-se, inicialmente, estipular o objetivo de modelar e analisar processos de desenvolvimento de *software* utilizando o formalismo de redes de *petri*, para a proposição de novos cenários.

Na seqüência, no capítulo de proposta metodológica, foi proposto um roteiro para o desenvolvimento do trabalho, buscando organizar e direcionar o estudo do cenário atual, assim como sua modelagem e análise. Também foi proposto um roteiro para a proposição de novos cenários.

O desenvolvimento do trabalho teve início no capítulo 5, com o cenário atual da empresa. Neste capítulo a empresa Desbravador foi apresentada e seu processo de desenvolvimento de *software* foi estudado. Foi realizada então a modelagem e a análise com o formalismo de redes de *petri*, utilizando o *software* Income como ferramenta computacional.

No capítulo 6, de proposição de cenários, buscou-se propor um novo cenário, com base na análise efetuada no capítulo 5. Questões relacionados ao tempo de processamento das OSs, à engenharia de requisitos (Sommerville, 2003) e à flexibilidade no tratamento das OSs foram consideradas para proposição do cenário.

De forma geral, ao se analisar e propor um novo cenário, fora considerado o ambiente atual, buscando não mexer em assuntos considerados críticos, como os recursos setores e os recursos humanos. Desta forma o novo cenário poderá ser aplicado com maior facilidade e em um menor período de tempo.

Não obstante ao sobredito, e avaliando os objetivos iniciais deste trabalho, pode-se verificar que a modelagem e a análise do processo de desenvolvimento de *software* pôde ser realizada utilizando o formalismo de redes de *petri*. Da mesma

forma, também foi possível propor um novo cenário utilizando redes de *petri*, embasado na análise do cenário atual.

Ao avaliarmos os resultados, verifica-se que ao compararmos o cenário proposto com o cenário atual da empresa, vários benefícios puderam ser constatados, tanto no tempo quanto na qualidade do processo de desenvolvimento de *software*.

O tempo de processamento médio das OSs foi reduzido significativamente, de aproximadamente três meses e meio para dois meses. A qualidade, no processo de desenvolvimento de *software*, também teve ênfase, sendo tratada no registro de solicitações. Como terceira e última alteração proposta, uma maior flexibilidade foi agregada ao PDS ao se propor que OSs corretivas fossem tratadas de forma preferencial pelo setor de programação.

Não obstante a tais benefícios, verificou-se também que alguns impactos negativos podem surgir no processo de desenvolvimento de *software*, e devem ser avaliados com maiores detalhes para que o novo cenário seja aplicado com sucesso.

De forma geral, pode-se avaliar que, não obstante a existirem muitos trabalhos sobre processo de desenvolvimento de *software*, este tende a proporcionar um mecanismo para a proposição de novos cenários, com a ferramenta de redes de *petri*, que neste tipo de processo pode ser muito bem aproveitada.

Buscando demonstrar como a ferramenta redes de *petri* pode auxiliar a modelar e analisar processos de desenvolvimento de *software*, este não explorou com muitos detalhes conceitos relativos às atividades específicas dos recursos humanos, até mesmo porque sua modelagem poderia ser realizada da mesma forma que na modelagem do refinamento das atividades dos setores.

Da mesma forma, este trabalho se limitou à modelagem e análise de processos de desenvolvimento de *software* (PDS), não relevando aspectos financeiros nem administrativos / gerenciais, contidos em empresas que desenvolvem *software*. Tais assuntos, relacionando PDS com questões financeiras o/ou administrativas / gerenciais são de extrema importância, e podem servir como base para novos trabalhos.

Abordando questões estratégicas, de empresas que desenvolvem software, para futuros trabalhos, pode-se também efetuar análises aprofundadas com distintas visões. Considerando objetivos empresariais, fatores de riscos e fatores críticos para obtenção de sucesso, pode-se mapear e visualizar variações do real estado do processo, e dependendo da variação, pode-se realizar ações que visem aprimorar o processo.

Não obstante, verifica-se que existem várias ramificações que poderiam ser proveitosas para um processo de desenvolvimento de *software*. Objetivos que buscam a melhoria contínua, muito abordada por conceitos como CMMI (Integração de Modelo de Maturidade e Capacidade), tornam-se uma extensão promissora deste trabalho, principalmente no que tange a avaliação das atividades que compõe as fases do CMMI em determinada empresa.

Problemas relacionados ao produto *software*, com seus recursos, também podem ser alvo de estudos futuros, onde a ferramenta redes de *petri* pode ser adotada com sucesso para auxiliar na análise e na construção de soluções.

Em suma, o formalismo de redes de *Petri* pode vir a somar muito em distintas áreas relacionadas ao processo de desenvolvimento de *software* e ao próprio *software*, viabilizando análises que podem ser utilizadas para obtenção de melhorias.

REFERÊNCIAS

AALST, W. M. P. V. D.; HEE, K. M. V.; HOUBEN. G. J.; Modelling and Analysing Workflow using a Petri-net based approach. **Proceedings of Second Workshop on Computer Supported Cooperative Work, Petri nets related formalisms.** p. 31-50, 1994.

AALST, W. M. P. V. D.; HEE, K. M. V.; Framework for business process redesign, wetice. **4th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE'95)**, p. 36, 1995.

AALST, W. M. P. V. D.; Three Good reasons for Using a Petri-net-based Workflow Management System. **Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96)**. Camebridge, Massachusetts, 1996, p. 179–201.

AALST, W. M. P. V. D.; The Application of Petri Nets to Workflow Management. **The Journal of Circuits, Systems and Computers**. 1998, p. 21–66.

AALST, W. M. P. V. D.; HEE, K. M. V.; Workflow Management: Models, Methods and Systems (in Dutch). **Academic Service.** Schoonhoven. 2002.

BEZERRA, C. A. A qualidade do processo de desenvolvimento de software a partir da gestão de projetos: um estudo de caso. **Sociedade Brasileira de Computação**. 2004.

ESPINDOLA, R. S.; MAJDENBAUM, A.; AUDY, J. L. N. Uma Análise Crítica dos Desafios para Engenharia de Requisitos em Manutenção de Software. **VII Workshop on Requirements Engineering**. Tandil, Argentina. 2004.

HENNEMANN, F. A., SANTOS, J. V. C.; RABELO, R.; CURY, J. E. R. Um Sistema Híbrido de Apoio à Decisão Formado por Redes de Petri, Simulação e Sistema Especialista. **Sba: Controle & Automação, Sociedade Brasileira de Automação**, v. 17, n. 1, 2006.

JIAO, J.; ZHANG, L.; PRASANNA, K. Process Variety Modeling for Process Configuration in Mass Customization: An Approach Based on Object-Oriented Petri Nets with Changeable Structures. **International Journal of Flexible Manufacturing System**: Vol. 16, p335-361, 2004.

JIAO, L.; CHEUNG, T. Y. Composition Verification for Workflow Nets. **Journal of Circuits, Systems & Computers**. Vol. 15, Issue 4, 2006, p551, 20p.

KANTORSKI, G. Z., Kroth, M. L. Controle de Métricas no Processo de Desenvolvimento de Software através de uma Ferramenta de Workflow. **I Workshop de Computação - WORKCOMP-SUL**, 2004, Florianópolis. Anais do I WORKCOMP-SUL, 2004.

KAWABATA, R.; SENUMA, Y.; MARUYAMA, J.; ITOH, K. Evolutional Systems Analysis and Its Tool by Effective Use of Chart Representations. **Journal of Integrated Design & Process Science.** Vol. 9, issue 1, 2005, p. 17, 14p.

KAWABATA, R.; ITOH, K. Diagrammatic Education for Software Engineering. **Journal of Integrated Design & Process Science.** Vol. 10, issue 1, 2006, p. 79, 14p.

KETTUNEN, P., LAANTI, M. How to Steer an Embedded Software Project: Tactics for Electing the Software Process Model. **Information and Software Technology**. Vol. 47, 2005, pp. 587–608.

MURATA, T. Petri Nets: Properties, Analysis and Applications. **Proceedings of the IEEE**: Vol. 77, n. 4, 1989.

PÁDUA, S. I. D.; SILVA, A. R. Y.; PORTO, A. J. V. INAMUSO, R. Y. O Potencial das Redes de Petri em Modelagem e Análise de Processos de Negócio. **Gestão e Produção**: Vol. 11. p.109-119. 2004.

PALYAGAR, B.; RICHARDS, D. A Communication Protocol for Requirements Engineering Processes. **Proceedings of the 11th Interna-tional Workshop on Requirements Engineering - Foundation for Software Quality**. Porto, Portugal. 2005.

PARREIRAS, F. S., OLIVEIRA, G. S. Análise comparativa de processos de desenvolvimento de software sob a luz da gestão do conhecimento: um estudo de caso de empresas mineiras. **SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE**. 2004, Brasília. Anais. Acessado em <http://www.fernando.Parreiras. com.br/publicacoes/WGC_Parreiras04.pdf>.

PRESSMAN, R. S.; **Software Engeneering, A Practitioner's Approach – Fourth ed**. The McGraw-Hill Companies. New York, USA. 1997.

QURESHI, M. R. J.; HUSSAIN, S. A. An Adaptive Software Development Process Model. **Advances in Engineering Software**, 2007.

RAUTERBERG, M., STROHM, O. Work Organization and Software Development. **Annual Review of Automatic Programming.** Vol. 16, 1992, pp. 121-128.

REZENDE, D. A. **Engenharia de Software e Sistemas de Informação.** 2ª ed. Rio de Janeiro: Brasport. 2002.

SCHMERL, B., ALDRICH, J., GARLAN, D., KAZMAN, R. YAN, H. Discovering Architectures from Running Systems. **IEEE Transactions on Software Engineering**: Vol. 32, no. 7. 2006.

SILVA, O. J., BORGES, C. A., SALVIANO , C. S., CRESPO, A. N., SAMPAIO, A. L., ROULLIER, A. C. Aplicação da ISO/IEC TR 15504 na Melhoria do Processo de Desenvolvimento de Software de uma Pequena Empresa. **V Simpósio Internacional de Melhoria de Processo de Software**. Novembro de 2003. Recife-PE.

SILVA, R. O. da. **Teorias da Administração**. São Paulo: Pioneira Thomson Learning, 2001. 523 p.

SNEED, H.M. BRÖSSLER, P. Critical Success Factors in Software Maintenance. **International Conference on Software Maintenance**. 2003.

SURAJ, Z.; FRYC, B.; MATUSIEWICZ, Z.; PANCERZ, K. A Petri Net System – An Overview. **Fundamenta Informaticae**. Vol. 71, Issue 1, 2006, p101, 20p.

SOMMERVILLE, I. **Engenharia de Software**. São Paulo: Pearson Addison Wesley. 2003. 592 p.

SUBRAMANIAN, G. H.; JIANG, J. J; KLEIN, G. Software Quality and IS Project Performance Improvements from Software Development Process Maturity and IS Implementation Strategies. **The Journal of Systems and Software**. 2006.

TAMAKI, P. A. O., HIRAMA, K., Melhoria de Processos de Desenvolvimento de Software Aplicando *Process Patterns.* **InfoComp: Journal of Computer Science.** Vol. 6, n. 1, 2007.

WADHWA, S.; RAO, K. S.; CHAN, F. T. S. Flexibility-enabled Lead-time Reduction in Flexible Systems. **International Journal of Production Research.** Vol. 43. No. 15, 2005, p. 3131–3162.

WFMC. **Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011**). Technical Report, Workflow Management Coalition, Brussels, 1999.

YI, Z.; YONG, Z.; WEINONG, W. Modeling and Analyzing of Workflow Authorization Management. **Journal of Network and Systems Management**: Vol. 12. 2004.

YIN, R. K. **Estudo de Caso, Planejamento e Métodos**. 3. ed. Bookman. 2005.

ZANLORENCI, E. P.; BURNETT, R. C. Abordagem de Engenharia de Requisitos em Software Legado. **Workshop em Engenharia de Requisitos.** Piracicaba, Brasil. 2003.

ZERGUINI, L. A Novel Hierarchical Method for Decomposition and Design of Workflow Models. **Journal of Integrated Design & Process Science**; Vol. 8, Issue 2, 2004, p65, 10p.

ANEXO 1: ORGANOGRAMA DE ATIVIDADES

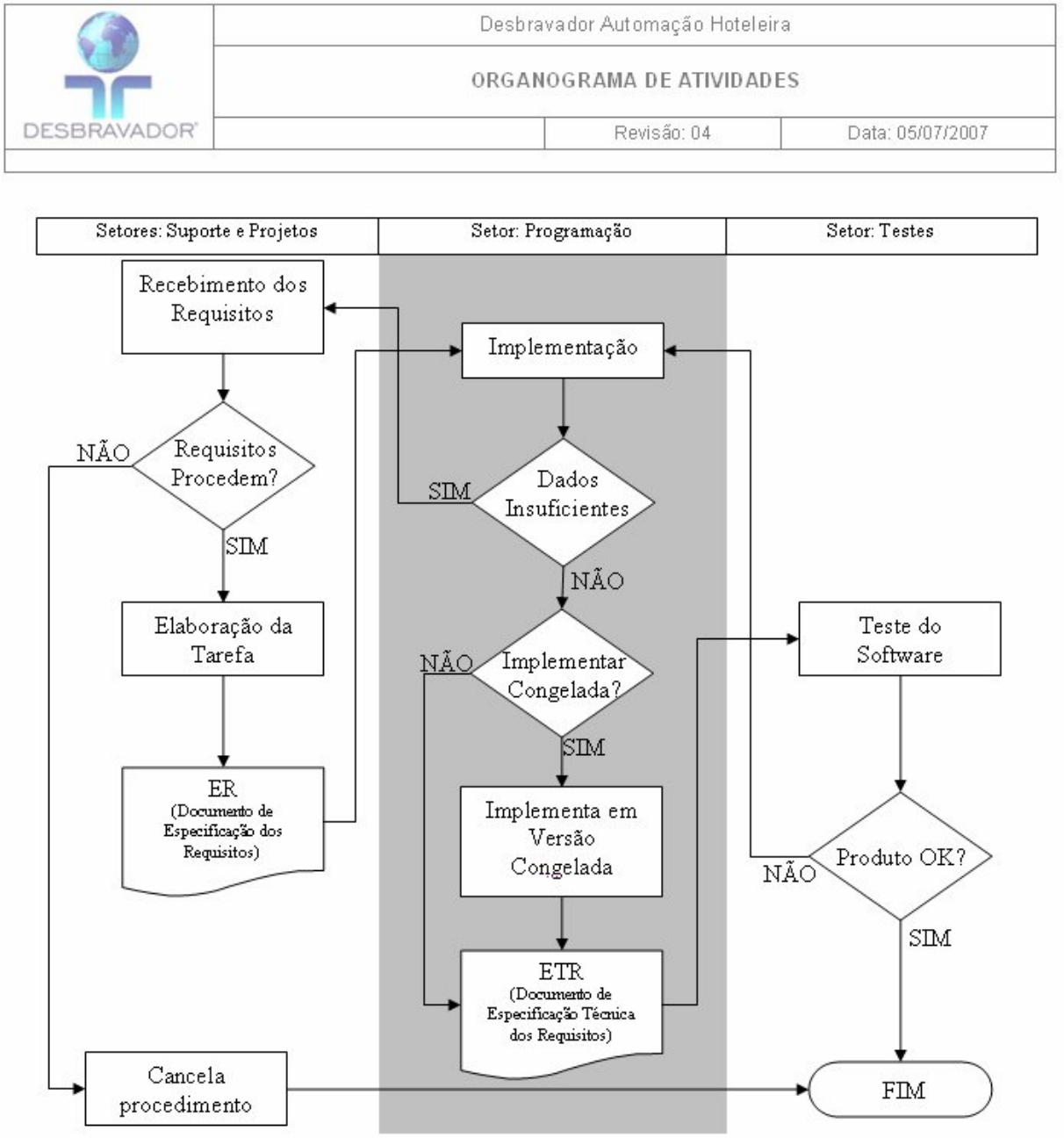

Figura 35: Organograma de atividades da empresa Desbravador
Fonte: Empresa Desbravador, 2007