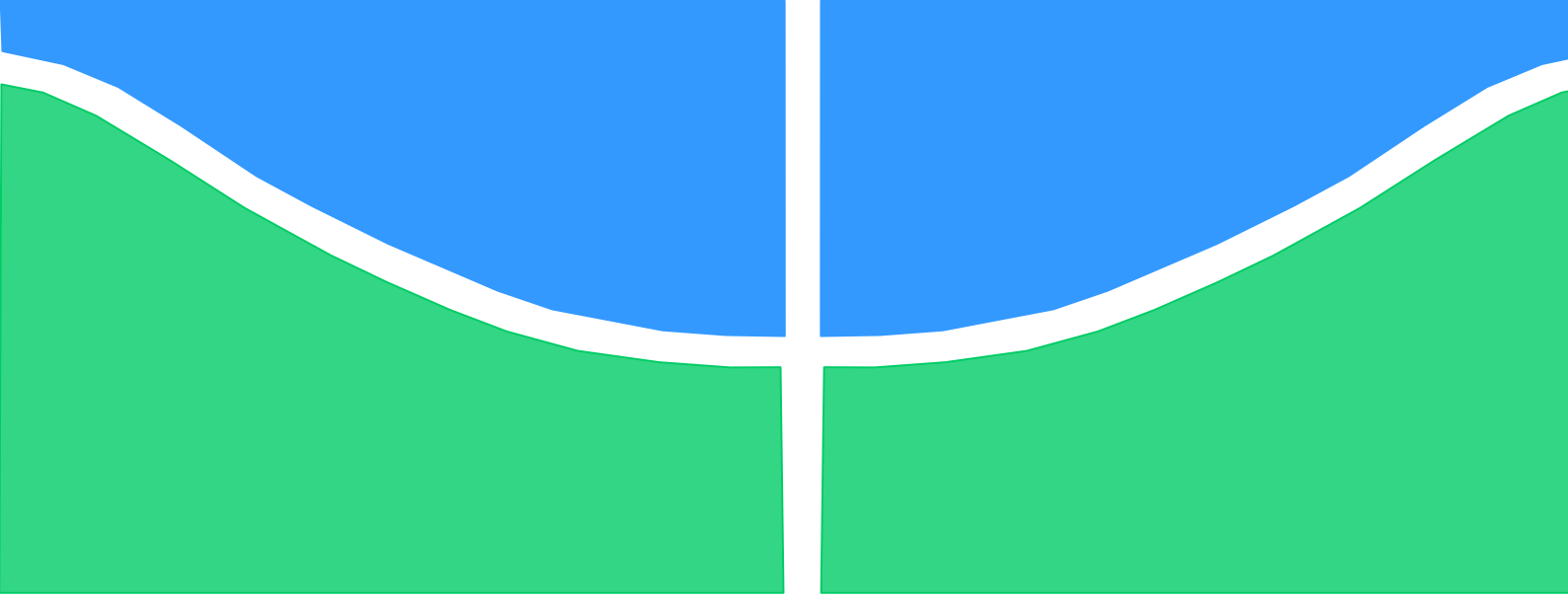

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Curso de Engenharia de Software

Uso do Kanban em um Processo de Gestão de Demandas de Manutenção de Software por Terceiros para um Órgão Público Federal Brasileiro

Autor: Jads Victor Paiva dos Santos

Orientadora: Dr.ª Rejane Maria da Costa Figueiredo

Brasília, DF

2015

Jads Victor Paiva dos Santos

Uso do Kanban em um Processo de Gestão de Demandas de Manutenção de Software por Terceiros para um Órgão Público Federal Brasileiro

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

Universidade de Brasília – UnB

Faculdade do Gama - FGA

Orientadora: Dr.ª Rejane Maria da Costa Figueiredo

Brasília, DF

2015

CIP – Catalogação Internacional da Publicação

Santos, Jads Victor Paiva dos.

Uso do Kanban em um Processo de Gestão de Demandas de Manutenção de Software por Terceiros para um Órgão Público Federal / Jads Victor Paiva dos Santos. Brasília: UnB, 2015. 115 p.: il.; 29,5 cm.

Trabalho de Conclusão de Curso – Universidade de Brasília

Faculdade do Gama, Brasília, 2015. Orientação: Rejane Maria da Costa Figueiredo.

1. Gestão de Demandas de Manutenção. 2. Kanban. 3.Processo de Manutenção. I. Figueiredo, Rejane Maria da Costa. II. Uso do Kanban em um Processo de Gestão de Demandas de Manutenção de Software por Terceiros para um Órgão Público Federal Brasileiro

CDU Classificação

Uso do Kanban em um Processo de Gestão de Demandas de Manutenção de Software por Terceiros para um Órgão Público Federal Brasileiro

Jads Victor Paiva dos Santos

Monografia submetida como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software da Faculdade UnB Gama - FGA, da Universidade de Brasília, em 24/06/2015, apresentada e aprovada pela banca examinadora abaixo assinada:

Prof. Dr.ª Rejane Maria da Costa Figueiredo, UnB/ FGA
Orientadora

Prof. Msc.: Elaine Venson, UnB/ FGA
Membro Convidado

Prof. Dr. Luiz Carlos Miyadaira
Coordenador Geral de Tecnologia da Informação do Ministério das Comunicações
Membro Convidado

Brasília, DF

2015

Dedico este trabalho à minha família, especialmente aos meus pais, João e Cátia, que sempre deram o melhor deles, para que eu pudesse alcançar o meu melhor.

AGRADECIMENTOS

Primeiramente a Deus, por sua infinita graça e misericórdia sobre minha vida, pelas bênçãos e livramentos e por sempre estar ao meu lado. Sem Ele, eu nada seria.

Aos meus queridos pais (João e Cátia), pelos cuidados, ensinamentos, paciência, compreensão e, principalmente, pelo amor incondicional. Sem eles é difícil dizer se alcançaria meus objetivos.

À minha família, mesmo que longe, tão amada. Em especial a minha avó, por ser essa senhora batalhadora e guerreira, de ternura incomparável e espírito sempre disposto a ajudar. Ao meu primo Caio por ser meu irmão, sempre me acobertando os erros. E a minha namorada, por ter me compreendido e amado.

À minha orientadora, Prof.ª Dr.ª Rejane, por sua contribuição acadêmica, dedicação e empenho. Mas, principalmente por, mesmo com o tempo avançado, ter me acolhido como orientando.

Aos meus amigos, que sempre me motivaram e me fizeram rir, independentemente da situação. Em especial a Maria, por me tratar como irmão. E a sua família, por me tratar como um membro dela.

À toda equipe do órgão que tive o privilégio de estagiar pela primeira vez e conviver por dois anos. Agradeço aos ensinamentos e por toda experiência que me foi proporcionada.

Ao Ministério, contexto deste trabalho, por possibilitar a prática deste estudo. E a todos que contribuíram para minha formação acadêmica e pessoal.

Muito obrigado.

“Aquele, pois, que sabe fazer o bem e não o faz, comete pecado. ”

(Tiago 4:17)

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANS	Acordo de Nível de Serviço
CGIIE	Coordenação-geral de Informação e Indicadores Educacionais
CGIS	Coordenação-geral de Infraestrutura e Serviços
CGSI	Coordenação-geral de Sistemas de Informação
CGTI	Coordenação-Geral de Tecnologia e Informação
DISOL	Diretorias de Soluções de Tecnologia da Informática
DTDIE	Diretoria de Tecnologia e Disseminação de Informações Educacionais
GeDDAS	Gestão de Demandas de Desenvolvimento Ágil de Software
IN 04/2014	Instrução Normativa Nº 04, de 11 de setembro de 2014
INEP	Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira
OS	Ordem de Serviço
PDTI	Plano Diretor de Tecnologia da Informação
PETI	Plano Estratégico de Tecnologia da Informação
PF	Pontos de Função
PO	*Product Owner*
SD	*Service Desk*
Sesol-3	3º Serviço de Soluções de TI
SLA	*Service Level Agreement*
SPOA	Subsecretária de Planejamento, Orçamento e Administração
STI	Secretária de Soluções de TI

TCU	Tribunal de Contas da União
TI	Tecnologia da Informação
TP	TargetProcess
WIP	*Work in Progress*

RESUMO

A publicação do manifesto ágil, em 2001, possibilitou o surgimento de novas metodologias de gerenciamento e desenvolvimento menos engessadas e mais dinâmicas, a exemplo temos os *framework*s *Kanban* e *Scrum*. Ambos *frameworks* têm como princípio a entrega constante de produto e a participação constante do cliente no processo. Carente dessa mudança, a indústria de desenvolvimento de software não demorou para avaliar as novas propostas, inclusive órgãos públicos federais. O *Kanban* e o *Scrum* podem e devem ser adaptados às necessidades específicas de uma organização. O processo de gestão de manutenção de software se difere do processo de desenvolvimento e, entre suas particularidades, destaca-se o fato desse necessitar de maior flexibilidade para agregar as frequentes mudanças, tanto no escopo da requisição, quanto na sequência de implementação. O *framework Kanban* vem sendo empregado para auxiliar o processo de gestão, possibilitando otimizar o fluxo de trabalho e permitindo determinar iterações, sem o limite de tempo, o que possibilita alterações nas demandas de acordo com a aceitação da instituição. O Principal objetivo deste trabalho foi apoiar a definição de um processo de gestão de demandas de manutenção de software por terceiros para um órgão do governo federal brasileiro, utilizando o *Framework Kanban*. A pesquisa empregada foi descritiva. Quanto aos procedimentos, realizou-se pesquisas na literatura de empresas que tenham implementado o *Kanban*, também se realizou o estudo de duas instituições Públicas (TCU e INEP) que utilizam este *framework*. Afim de aplicar o *Kanban* no Ministério X, primeiro foi necessário caracteriza-lo e somente então desenvolver as atividades propostas para implementação. Ao fim, foi possível propor uma modelagem do processo de manutenção, e um quadro *kanban* adaptado as necessidades do Ministério. A Homologação destes e a conclusão da adaptação do framework está atrelada a trabalhos futuros.

Palavras-chave: Gestão de Demandas de Manutenção; Kanban; Processo de Manutenção.

ABSTRACT

The agile manifest published in 2001, enabled the surface of new maintenance and development methodologies, less dense and more dynamic. It is possible quote Kanban and Scrum as examples. Both frameworks have as a principle the constant delivery of product and the client's constant participation in the process. Lacking this change, the software development industry did not take to evaluate the new proposals, including federal government agencies. Kanban and Scrum can and should be adapted to the organization's specific needs. The software maintenance management process is different from the development process and, among its singularities, highlights the fact that the process needs greater flexibility to aggregate the frequent changes both in the scope of the request, as in the implementation order. Kanban framework has been used to assist the management process, allowing to optimize the workflow and allowing to define iterations, without a time limit, which allows changes in demands according to the acceptance of the institution. The mainly objective is *support the definition of a software management process for maintenance demands by third parties in Brazilian federal government, using the Kanban Framework*. This research is classified as descriptive. As the procedures, held research in the literature of companies that have implemented the Kanban. Also was conducted a study in two public institutions (TCU and INEP) that use this framework. In order to apply the Kanban in the Ministry X, first was necessary characterizes it and only then carry out the activities proposed for implementation. In the end, was possible to propose a modeling of the maintenance process, and kanban board adapted the Ministry of needs. The approval of these and the completion of the adjustment of the framework is linked to future work

***Keywords**:* Management Maintenance Demands; Kanban; Maintenance process.

CAPÍTULO 1 - INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAS DO CAPÍTULO

Neste primeiro capítulo apresentam-se o contexto no qual o trabalho está inserido, os problemas e objetivos motivadores do estudo, em seguida, a justificativa do tema, a metodologia adotada, e como este trabalho está organizado.

1.2 CONTEXTO

De acordo com Naur e Randell (1968) o termo engenharia de software surgiu no fim da década de 60 em Garmisch, Alemanha, durante uma conferência focada nos principais problemas enfrentados durante o desenvolvimento de software. Na visão da engenharia de software, é possível definir um ciclo de vida para o software.

Sommerville (2007) apresenta o ciclo de vida de um software composto por cinco fases: Análise e Definição de Requisitos; Projeto de Sistema e Software; Implementação e Teste de Unidade; Integração e Teste de Sistema e; por último, Operação e Manutenção.

A fase de manutenção apresenta um grau de complexidade maior que as demais fases (SOMMERVILLE, 2007) (PRESSMAN, 2011). Vários autores ressaltam problemas em etapas dessa fase devido: à falta de documentação e a dificuldade da equipe de manutenção em *entender* a arquitetura e o funcionamento do software (APRIL; ALAN, 2008); à gestão de demandas (WANG, CONBOY E CAWLEY, 2012); e à duração da fase (TAN; MOOKERJEE, 2005).

Tan e Mookerje (2005) atribuíram a duração da fase ao fato dos softwares serem desenvolvidos para funcionarem em conjunto com outros sistemas. Essa interação aumenta o tempo dedicado à fase de manutenção do software, consequentemente aumentando os custos. Segundo Chapin (2009), os custos também aumentam de acordo com a dependência da empresa em relação a determinado software.

No intuito de suportar os custos gerados pela combinação desses fatores, acaba-se investindo na fase de manutenção a maior parte dos recursos destinados ao projeto, tornando-a a fase mais onerosa do ciclo de vida (TAN; MOOKERJEE, 2005) (WEBSTER; OLIVEIRA; ANQUETIL, 2005) (RASHID; WANG; DORNER, 2009) (SERNA; SERNA, 2014).

A fase de manutenção apresenta desafios na execução das demandas, assim como apresenta na gestão das mesmas. O cenário relatado pelo Wang, Conboy e Cawley (2012) expõe uma constante mudança nas requisições de manutenção, e ressalta a dificuldade de gerenciá-las com metodologias tradicionais. Em 2001, publicaram o Manifesto Ágil (HIGHSMITH, 2002), possibilitando que visões dinâmicas de gerenciamento possam ser incorporadas ao desenvolvimento de software.

A metodologia ágil para gerenciamento mais pesquisada é o *Framework Scrum* (DINGSØYR; DYBÅ; ABRAHAMSSON, 2008). O *Scrum* estabelece papéis, artefatos e eventos, assim como regras que regem o funcionamento do conjunto. Tudo ocorre em intervalos de tempo chamados de *Sprints* e a

sua duração, também chamado de *time-box*, pode ser de duas a quatro semanas (COHN, 2010). Apesar da popularidade do *Scrum*, outros *frameworks* estão ganhando espaço, como é o caso do *Kanban*.

O *framework Kanban* tem sua origem na linha de produção da Toyota (OHNO, 1997) e baseia-se na otimização do fluxo de trabalho. Para alcançar esse objetivo, sua principal ferramenta é a limitação do trabalho em progresso. Com o foco no *fluxo de trabalho*, o *Kanban* não faz uso da prática de estipular iterações baseadas em períodos de tempo, permitindo facilmente alterações nas demandas a serem desenvolvidas (ANDERSON, 2010). O emprego do *Kanban* na gestão de demandas de software é apoiado por estudos de caso que comparam o *Kanban* ao Scrum, apresentando maior capacidade do *Kanban* em adaptar-se as frequentes mudanças nas requisições de manutenção (SJOBERG; JOHNSEN; SOLBERG, 2012) (MAASSEN; SONNEVELT, 2010) (GRAVES, 2011).

Na Administração Pública Federal Brasileira – APF, a contratação de bens e serviços comuns, categoria na qual se encaixa a manutenção de software (BRASIL, 2014), deve seguir a Instrução Normativa 04/2014 publicada pelo Governo Federal Brasileiro, por meio da Secretaria de Logística e TI (SLTI) (BRASIL, 2014). Essa é a principal norma referente ao regulamento das contratações de serviços de TI pelas entidades e órgãos do Sistema de Administração dos Recursos de Tecnologia da Informação (BRASIL, 2014).

1.3 PROBLEMA

No cenário da Administração Pública Federal Brasileira – APF, os órgãos têm iniciado seus estudos e análise de viabilidade do uso do *Kanban* para gestão de demandas de serviços de manutenção. A questão é como adaptar o *Kanban* par definir um processo de gestão de demandas de serviços de manutenção de software às características dos órgãos, num contexto de número de servidores insuficientes, falta de capacitação dos servidores, dependência excessiva dos fornecedores, e alinhado a legislação brasileira. Neste trabalho, foi selecionado um órgão como estudo de caso, denominado *Ministério X*. O órgão necessita de um processo de gestão de demandas de manutenção de software para gerenciar adequadamente as requisições feitas e visualizar o progresso da mesma durante a sua implementação pela empresa contratada. Com isso, a questão de pesquisa deste trabalho é:

Como empregar o Framework Kanban na definição de processo de gestão de demandas de manutenção de software por terceiros para um órgão do governo público federal brasileiro.

1.4 OBJETIVOS

O objetivo desse trabalho é apoiar a definição de um processo de gestão de demandas de manutenção de software por terceiros para um órgão do governo federal brasileiro, utilizando o *Framework Kanban.*

Os objetivos específicos definidos foram:

- Identificar e analisar empregos do *Framework Kanban* por organizações;
- Identificar e analisar a legislação vigente para as contratações de serviços de TI por órgãos públicos federais brasileiros;
- Caracterizar o órgão do governo federal brasileiro definido como objeto de estudo de caso;
- Identificar e adequar práticas para a definição do processo de gestão de demandas de manutenção de software por terceiros para o órgão do governo federal brasileiro selecionado.

1.5 METODOLOGIA

Este trabalho compreende o estudo da fase de manutenção, considerando seus benefícios, dificuldades e modelos utilizados para estruturar essa fase, utilizando um estudo de caso para o emprego do *framework Kanban* e sua adequação ao processo de gestão de demandas de manutenção de software de um órgão, a fim e propor um processo de gestão de demandas utilizando o *framework Kanb*an para gerenciar as demandas de serviços de manutenção de software do órgão.

De acordo com Gil (2008), essa pesquisa é classificada como uma pesquisa de natureza aplicada e a abordagem qualitativa. No intuito de descrever o estudo de elaboração de um *Framework Kanban,* adaptado as necessidades do *Ministério X*, essa pesquisa é classificada, quanto ao tipo, como descritiva. Quanto aos procedimentos, revisão bibliográfica, mas também são realizados estudos do emprego de *Kanban* por organizações públicas, emprega-se a pesquisa bibliográfica e documental. Finalizando, o estudo é sobre um órgão brasileiro, isto é, o emprego da técnica de estudo de caso único.

Segundo a metodologia de André (2005), essa pesquisa pode ser realizada em três etapas distintas, descritas a seguir:

- **Planejamento:** nessa etapa, são definidas a questão de pesquisa e os objetivos, seguido da metodologia a ser adotada, com a definição dos procedimentos e das técnicas para a coleta de dados**;**
- **Coleta dos Dados:** nessa etapa são empregadas as técnicas de coleta de dados que atendam aos objetivos especificados.

  As técnicas empregadas são descritas a seguir:

  - **Revisão Bibliográfica**: pesquisas nos principais bancos de dados científicos, como revistas e conferências de engenharia de software, buscando caracterizar a fase de

manutenção, o *Framework Kanban*, assim como, caracterizar o cenário de contratação de serviços de TI por órgãos públicos federais brasileiros;

- **Análise dos Documental:** caracterizar o objeto do estudo de caso selecionado, denominado de *Ministério X* neste trabalho, assim como, identificação e análise do emprego *Framework Kanban* pelas organizações;
- **Visita Técnica**: reconhecer e analisar os diferentes empregos do *Framework Kanban* por órgãos públicos;
- **Entrevista semiestruturada**: realizar entrevistas aos responsáveis pela implantação do *framework Kanban* nas organizações identificadas para visitação técnica, assim como, os envolvidos nesse cenário do *Ministério X*, objeto de estudo de caso. Nesse tipo de entrevista segue um roteiro idealizado previamente, mas permite flexibilidade ao entrevistador para realizar alterações de acordo com o andamento da entrevista.
- **Análise dos Dados**: Dados os levantamentos realizados, serão analisadas as informações obtidas, buscando apoiar *a definição de um processo de gestão de demandas de manutenção de software por terceiros para um órgão do governo federal brasileiro, utilizando o Framework Kanban.*

Na Figura **1Error! Reference source not found.** apresenta-se uma proposta da metodologia a ser adotada neste trabalho.

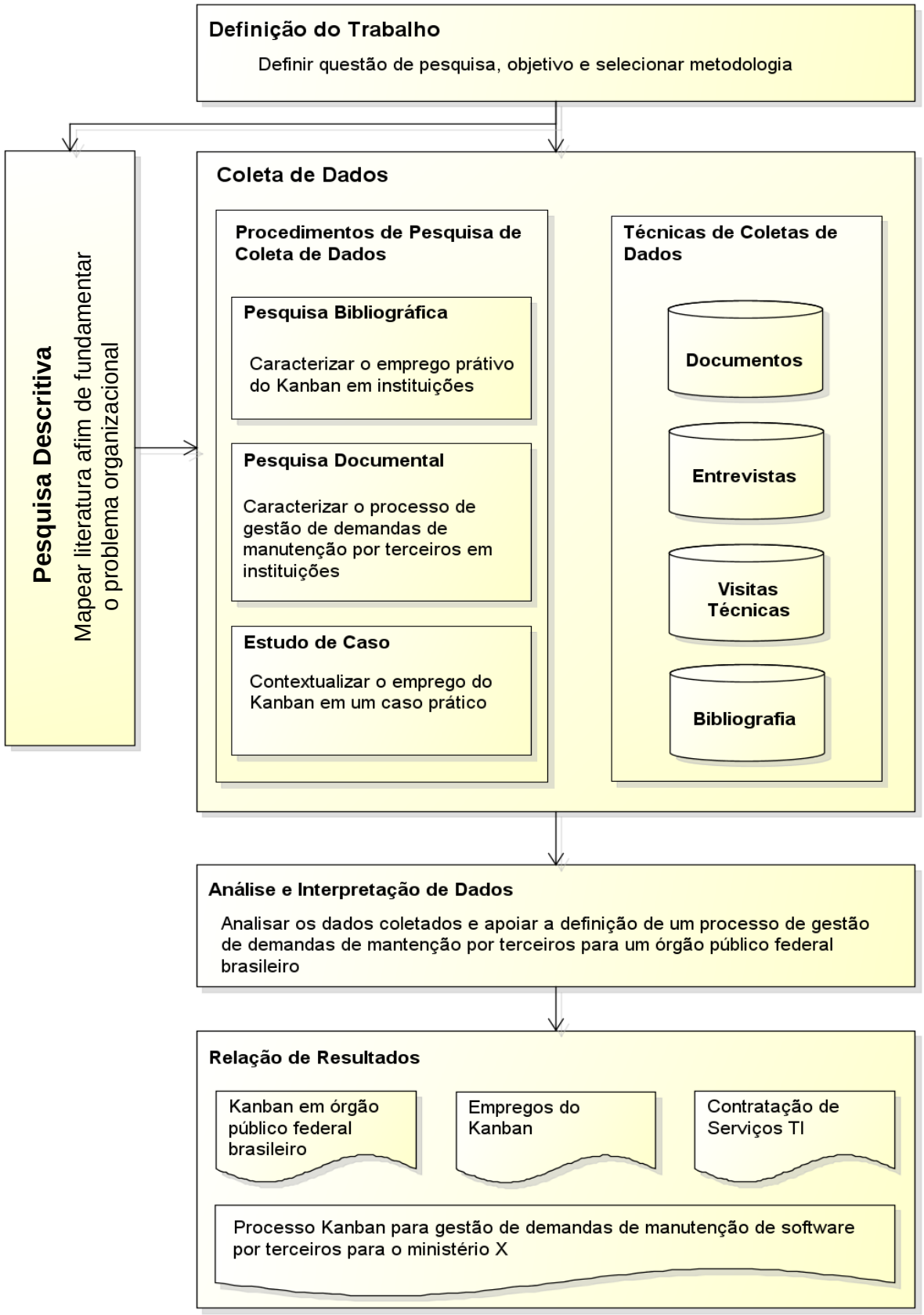

Figura 1: Metodologia de Pesquisa. Fonte: Autor

1.6 ORGANIZAÇÃO DO TRABALHO

O trabalho aqui apresentado está organizado em capítulos, totalizando quatro. Neste Capítulo 1 – *Introdução* são abordados o contexto do trabalho, o problema, os objetivos, e a maneira como o trabalho está organizado

No Capítulo 2 – Manutenção caracteriza-se a atividade do ciclo de vida de software: tipos de manutenção, as dificuldades, e finalizando, apresentam-se alguns modelos de manutenção de software.

No Capítulo 3 – *Framework Kanban* são apresentados um breve histórico sobre a criação do *Kanban*, as atividades do *framework* e as métricas que podem ser utilizadas. Em seguida, alguns benefícios proporcionados pelo *Kanban* à gestão de demandas de manutenção, assim como os benefícios com o emprego de ferramentas *Kanban*. Finalizando, apresenta-se uma comparação entre ferramentas.

No Capítulo 4 – *Proposta de Trabalho* são apresentadas a proposta do trabalho, a metodologia adotada, o cronograma a ser seguido, os resultados esperados com o emprego do *framework Kanban* para a definição de um processo de gestão de demandas de manutenção de software por terceiros para o *Ministério X*. Finalizando, apresentam-se as Considerações Finais.

No Capítulo 5 – *Estudos de Caso* são apresentados o estudo de caso de dois órgãos públicos Brasileiro que fazem uso do Framework *Kanban* na gestão do processo de manutenção de demandas. O primeiro estudo de caso apresentado é do Tribunal de contas da União e o segundo é o do Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira.

No Capítulo 6 – *Materiais e Métodos* são apresentados a caracterização do *Ministério X*, assim como a atual metodologia de gestão de demandas de manutenção e a modelagem do processo como um todo. Com base nessas informações também se realiza um comparativo entre o atual processo e o *Framework Kanban*, assim como identifica-se focos de melhoria.

No Capítulo 7 – *Adaptações do Framework Kanban para um órgão Público Brasileiro* são apresentadas as propostas de modelagem do processo de manutenção, aplicando soluções para os focos de melhorias identificados no capítulo anterior. Também se apresenta uma proposta de quadro kanban baseado na necessidade do *Ministério X*.

No Capítulo 8 – *Conclusões e Trabalhos Futuros* são apresentados as conclusões obtidas com o desenvolvimento deste trabalho, finalizando com sugestões de trabalhos a serem realizados e que darão continuidade a este.

CAPÍTULO 2 – CONTRATAÇÃO DE SERVIÇOS DE TI POR ÓRGÃOS PÚBLICOS FEDERAIS

2.1 CONSIDERAÇÕES INICIAIS DO CAPÍTULO

Neste capítulo apresenta-se um cenário geral da contratação de serviços pelo setor público brasileiro, dando ênfase aos serviços de TI. Apresenta-se também as principais normas e guias referentes a área de TI.

2.2 CONTRATAÇÃO DE SERVIÇOS PELO SETOR PÚBLICO

No Brasil, a atividade de terceirização de serviços teve início na década de 50, juntamente com a vinda das primeiras multinacionais. Essa era empregada exclusivamente para proporcionar a redução dos custos de mão-de-obra, ignorando aspectos como qualidade, produtividade, entre outros (CAMPOS,2006).

O primeiro diploma normativo a abordar a terceirização foi o Decreto-Lei nº 200 de 1967, sendo permitido a contratação de empresas para realizarem tarefas executivas (AMORIM, 2009). De acordo com o Artigo 6º, as atividades da Administração Federal deverão obedecer cinco princípios fundamentais (BRASIL, 1967), sendo eles:

- **Planejamento:** Define a obrigatoriedade das ações governamentais obedecerem e acompanharem um planejamento onde são definidos objetivos, prazos e recursos financeiros.
- **Coordenação:** Propõe a criação de coordenadorias descentralizadas, reafirmando a hierarquia e a necessidade de obedece-la.
- **Descentralização:** Institui a execução das atividades da Administração Federal de maneira amplamente descentralizada e constituída em três planos, sendo eles: Dentro dos quadros da Administração Federal, da Administração Federal para a das unidades e da Administração Federal para a órbita privada.
- **Delegação de Competência:** Determina que esta deverá ser usada no intuito de promover a descentralização administrativa, assegurando que pessoas próximas possam resolver problemas com rapidez e objetividade
- **Controle:** A Administração Federal deverá ter o controle de suas atividades dividida em três níveis, sendo eles: pela chefia competente, pelos órgãos próprios de cada sistema e pelos órgãos próprios do sistema de contabilidade e auditoria.

Em 1970, no intuito de exemplificar as diversas atividades passíveis de terceirização, criou-se a Lei nº 5.645/70. Em 1986 publicou-se a DL nº 2.300, onde as licitações e contratos da Administração Federal foram disciplinadas, prevendo também a possibilidade de execução indireta de obras e serviços (Patrícia Pinheiro Silva, Terceirização nos serviços públicos). Logo mais, em 1988, a CF/88 tornou expresso a obrigatoriedade da licitação na contratação de serviços, inclusive para as entidades da Administração Indireta). Para atender à CF/88, a Lei nº 8.666 foi alterada para contemplar o regime de

execução indireta de obras e serviços (PIETRO, 2005), esta também prevê que tanto as contratações de serviços, quanto a de produtos, pelo governo brasileiro devem atender ao princípio da isonomia. Em 1993, o Tribunal Superior do Trabalho, por meio da edição do Enunciado nº331, realizou a distinção entre mão-de-obra e a terceirização, permitindo a terceirização nas atividades-meio (BRASIL, 1993). Em 1997, o Decreto nº 2.271 recomenda que os órgãos públicos terceirizem os serviços que não se relacionam diretamente com a finalidade da instituição (BRASIL, 1997).

Em 2010, o Tribunal de Contas da União (TCU), juntamente com o Senado Federal, publicou o livro Licitações & Contratos. Este contempla as boas práticas e orientações para as licitações e contratos administrativos. Baseando-se na Lei nº 8.666, relativa aos contratos administrativos e as licitações, e na Lei nº 10.520, conhecida como Lei do Pregão, essa publicação faz um apanhado das orientações e jurisprudência do TCU sobre o tema.

De acordo com o livro e as leis no qual este se baseia, existem nove princípios básicos para o processo de licitação, sendo eles: Princípio da Legalidade, Princípio da Isonomia, Princípio da Impessoalidade, Princípio da Moralidade e da Probidade Administrativa, Princípio da Publicidade, Princípio da Vinculação ao Instrumento Convocatório, Princípio do Julgamento Objetivo, Princípio da Celeridade, Princípio da Competição. E a licitação, propriamente dita, pode ser dividida em quatro modalidades, sendo elas: concorrência, tomada de preços, convite e pregão. Sendo a última obrigatória para realizar a contratação de bens e serviço comuns, incluindo os de tecnologia da informação. Atualmente a contratação de serviços terceirizados para a área de TI é legislada pela Instrução Normativa nº 4.

2.3 INSTRUÇÃO NORMATIVA Nº 4

A Instrução Normativa nº4 é referente a Legislação e ao processo de contratação de soluções para a área de TI. A primeira IN4 foi publicada em 2008 e até então todas as informações pertinentes às contratações de TI estavam dispersas em documentos legais. No intuito de facilitar esse processo e buscar o alinhamento estratégico da área de TI com as áreas fins, a IN4/2008 reuniu ordenadamente pela primeira vez a legislação referente as contratações de TI. Estruturada como um processo lógico de contratação, ela define fases, requisitos e atores, incluindo suas responsabilidades e atribuições.

Após a publicação da IN nº4, a primeira revisão foi feita em 2010, nela aprimorou-se a formalização das solicitações de contratação de soluções de TI, criou-se a Equipe de Planejamento da Contratação e a divisão do trabalho por competência e instrui-se a respeito da definição de sanções. Após 4 anos, em 2014, realizou-se a segunda revisão, consolidando a IN nº 4 como marco para a regulamentação na contratação de TI.

A Secretária de Logística e Tecnologia da Informação (SLTI) publicou a Instrução Normativa (IN) Nº 4, de 11 de setembro de 2014 (BRASIL, 2014a), no intuito de substituir a IN 4/2010 e tornar-se a principal norma regulamentadora de contratações de soluções de TI pelos integrantes do Sistema de Administração dos Recursos de Informação e Informática (SISP). As principais melhorias em relação a sua antecessora estão relacionadas com a consolidação desta como um marco regulatório, a dinamização e a aproximação do processo a legislação (BRASIL, 2014b).

A IN 4/2014 está organizada em 41 artigos, divididos em três capítulos: Disposições Gerais, Processo de Contratação e Disposições Finais. A Figura 2 apresenta o diagrama estrutural da IN 4/2014.

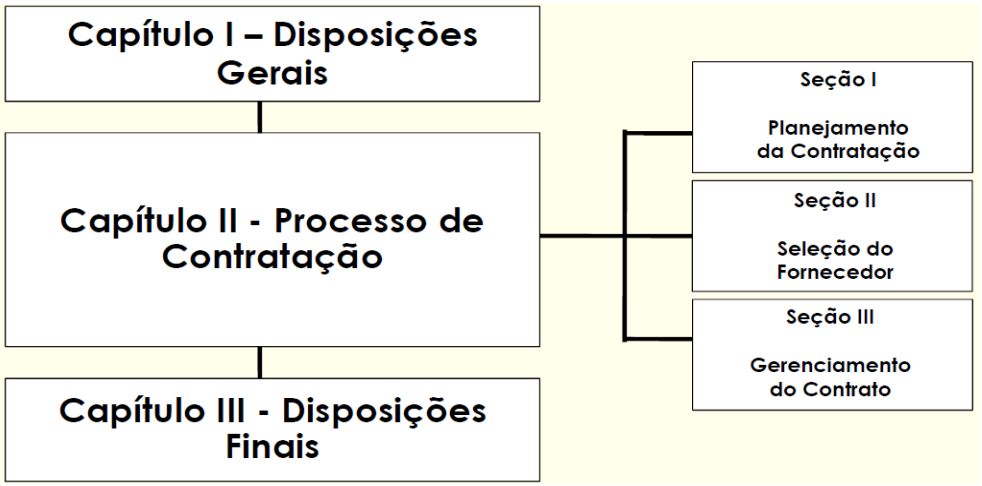

Figura 2: Estrutura da IN 4/2014. Fonte (BRASIL, 2014b)

Disposições Gerais: Este capítulo é composto por seis artigos e, de maneira geral, estes abordam as áreas onde a IN nº 4 é aplicável e aonde não aplicável. Também apresenta os atores, artefatos, o que não é permitido no processo de contratação e a necessidade do Plano Diretor de Tecnologia da Informação (PDTI) estar alinhado com a Estratégia Geral de Tecnologia da Informação e Comunicação (EGTIC), e com o plano estratégico institucional.

Processo de Contratação: Este capítulo é composto por vinte artigos e de acordo com o Artigo 8º, o processo de contratação de Soluções de TI pode ser dividido em três seções: Planejamento de Contratação, Seleção de Fornecedor e Gestão de Contrato. A modelagem do processo de contratação pode ser vista na Figura 3.

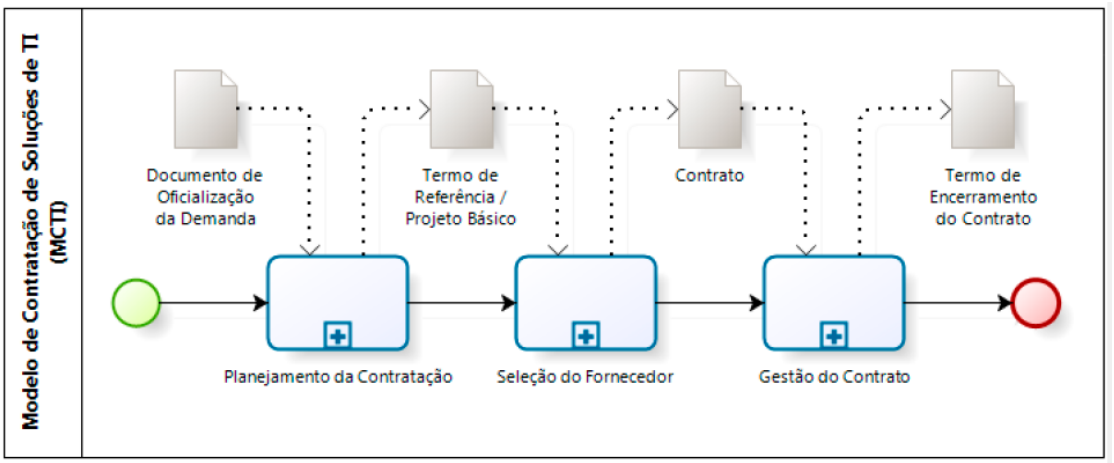

Figura 3: Modelagem do Processo de Contratação. Fonte: (BRASIL, 2014b)

Planejamento de Contratação (PCTI): O planejamento de contratação é composto por quatro etapas, sendo elas:

Instituição da Equipe de Planejamento da Contratação: A equipe de Planejamento da Contratação é a responsável por realizar o planejamento e é composta pelo integrante Técnico, Administrativo e o Requisitante.

Estudo Técnico Preliminar da Contratação: Responsável por Elaborar o documento que ateste a viabilidade econômica e técnica da contratação em si.

Análise de Riscos: Responsável por Elaborar o documento com a descrição, tratamento e análise dos riscos e ameaças

Termo de Referência ou Projeto Básico: Documento responsável por apresentar a justificativa pela qual uma Solução de TI deve ser prestada pelos possíveis contratados. O documento é responsabilidade da Equipe de Planejamento de Contratação e é elaborado com base no Estudo Técnico Preliminar.

Na Figura 4 é possível visualizar a modelagem do Planejamento de Contratação com suas quatro etapas e a ordem de execução.

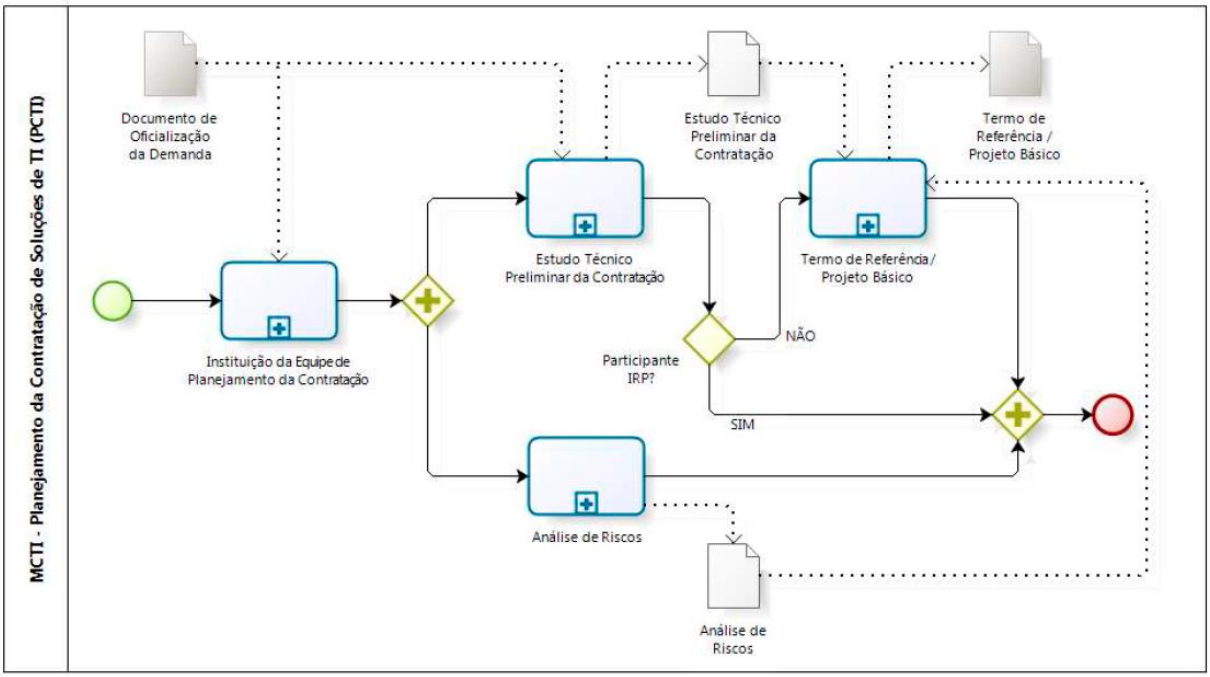

Figura 4: Modelagem do Planejamento de Contratação. Fonte: (BRASIL, 2014b)

Seleção do Fornecedor (SFTI): Responsável por nomear o Gestor do Contrato, o Fiscal Técnico do Contrato, o Fiscal Requisitante do Contrato e o Fiscal Administrativo do Contrato. O marco final é a assinatura do contrato.

Gestão de Contrato (GCTI): Responsável por definir procedimentos adequados ao fornecimento da Solução de TI, servindo de insumo para realizar a elaboração do Modelo de Gestão. Este por sua vez, no intuito de minimizar os riscos no fornecimento da Solução, é responsável pelos processos de gestão e fiscalização da Solução de TI,

Disposições Finais: Este capítulo é composto por quatro artigos e, de maneira geral, abordam a aplicação na nova norma aos contratos já existentes e futuros. Estabelece que a IN 4/2014 entrará em vigor no dia 2 de janeiro de 2015, e que após essa data, as prorrogações contratuais devem estar de acordo com a nova norma, salvo exceções.

2.4 Guia de Boas Práticas em Contratação de Soluções de TI - TCU

De acordo com Brasil, 2012 é árdua a tarefa de vislumbrar um serviço desenvolvido por algum órgão ou entidade da Administração Pública Federal (APF) que não careça diretamente ou indiretamente da Tecnologia. E no intuito de apoiar o uso da TI para dar suporte na automatização de processos de trabalho, na estrutura de informações para dar suporte à gestão e transformação do negócio das organizações públicas, faz-se necessário a contratação de produtos e serviços relacionados a TI. Este

cenário motivou o Tribunal de Contas da União (TCU) a criar Secretaria de Fiscalização de Tecnologia da Informação (SEFTI) (BRASIL, 2015a).

Criada em 2006, a SEFTI é a secretária responsável por fiscalizar a gestão e o uso de recursos de TI na Administração Pública Federal. Também elabora manuais, notas técnicas, metodologias e procedimentos para planejamento e execução de fiscalizações de tecnologia da informação (BRASIL, 2015b). Dentro de suas competências, a SEFTI elaborou o Guia de Boas Práticas em Contratação de Soluções de TI (BRASIL, 2012).

O Guia de Boas Práticas em Contratação de Soluções de TI, publicado em 2012, foi elaborado com a finalidade de contribuir com o planejamento das contratações de bens e serviços dos órgãos da APF. Focado em auxiliar os gestores públicos a com planejamento, o guia indica as melhores práticas, a legislação e a jurisprudência sinalizam sobre o planejamento das contratações.

O guia elaborado pelo TCU aponta sessenta e seis riscos e sugestões para controle interno e destaca o fato do processo de planejamento de contratação sofrer influencias externas de processos relacionados.

Como é possível visualizar na Figura 5, o planejamento do órgão (1) é realizado com base no planejamento do órgão governante superior (2), caso exista, como desdobramento do planejamento do órgão, é feito o de TI (3). Por sua vez, é necessário utilizar os planos de TI do órgão governante superior (4) como entrada do processo de planejamento de TI. Baseando-se no planejamento de TI, o planejamento conjunto das contratações de TI é executado de modo que as contratações estabelecidas estejam atreladas às estratégias do órgão e de TI (5). As contratações devem ser realizadas em função dos planejamentos citados acima, logo, inicialmente é realizado o planejamento de contratação (6). Estando este finalizado, ocorre a fase de seleção do fornecedor (7) e, em seguida, inicia-se a fase de gestão de contrato (8). Se durante o planejamento da contratação de uma solução for averiguado a possibilidade de dividi-la em soluções complementares, seleciona-se dois ou mais fornecedores e é feita a gestão dos respectivos contratos.

Paralelamente a esses processos, há a execução dos processos relativos a governança de TI, tendo este como meio, a alta administração emite diretrizes e realiza o acompanhamento da implementação das mesmas. Devido a expectativa de que essas envolvam contratações de TI, parte do acompanhamento da alta administração pode ser por intermédio de auditorias na área de TI (9). Também deve ser realizado, pelos gestores, controles internos ao longo dos processos de trabalho e que podem tratar os riscos envolvidos. Novamente podem ser realizadas auditorias, sendo a unidade de auditoria interna do órgão responsável por esta ação (9). Por fim, diversas instâncias de controle externas ao órgão podem atuar sobre os processos descritos (10).

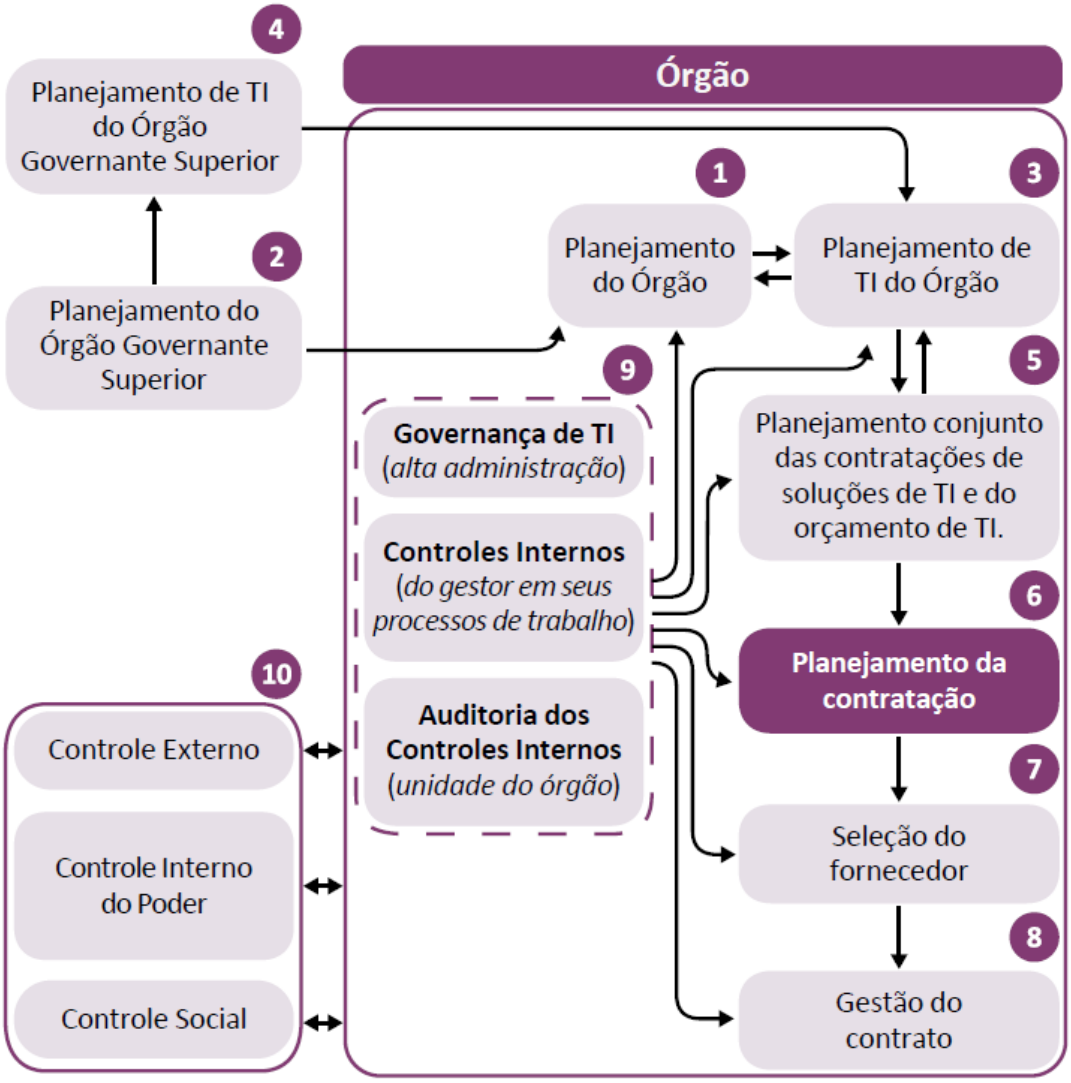

Figura 5: Contexto do planejamento das contratações de soluções de TI. Fonte: (BRASIL, 2012a)

2.5 Considerações Finais

Neste capítulo apresentou-se um breve histórico da contratação de serviços, em geral, terceirizados pelo setor público brasileiro. Logo mais focou-se na contratação de serviços de TI, sendo necessário abordar a Instrução Normativa nº4 que atualmente é a responsável por este setor. Ainda no contexto de TI, também se apresentou o guia de boas práticas elaborado pelo TCU.

CAPÍTULO 3 – MANUTENÇÃO

3.1 CONSIDERAÇÕES INICIAIS DO CAPÍTULO

Neste Capítulo são apresentados os conceitos relativos à atividade de manutenção, em seguida explana-se a respeito dos desafios encontrados durante a manutenção de um software de software. Ao fim, apresenta-se três processos de manutenção.

3.2 MANUTENÇÃO DE SOFTWARE

A definição de Engenharia de Software elaborada pela (IEEE, 1993) apresenta que: (1) é a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software; isto é, a aplicação da engenharia ao software. (2) O estudo de abordagens como definido em (1) ". Para Pfleeger (2007), a engenharia de software inicia-se na conversa com o cliente para definir os requisitos. E o término é fase de Evolução e Manutenção, onde ocorre a entrega do software implementado O software implementado é entregue ao cliente (SOMMERVILLE,2007).

De acordo com a IEEE (1998), a manutenção limita-se à alteração do software após sua entrega, sendo que essas mudanças podem ter como objetivo a correção de erros, a melhora de performance, ou qualquer outra alteração necessária para adaptar o produto a um ambiente modificado (SOUZA *et al*.; 2005) (VORA, SARDA; 2006). Devido a esta abrangência de atividades e objetivos, Swanson (1976), Lientz e Swanson (1980), Alain e Alain (2008) e Grubb e Takang (2011) concordam que a manutenção pode ser dividida em quatro tipos. Sendo elas Corretiva, Adaptativa, Evolutiva e Preventiva. Em seguida, uma breve descrição dos tipos:

- A manutenção corretiva é realizada quando o software apresenta algum defeito, sendo que este pode ser oriundo tanto da lógica do programa, quanto do código ou até mesmo da interface. Esse tipo de erro é nominado como *bug*. De acordo com Sommerville (2007), esse tipo representa 17% do esforço total do esforço empregado na manutenção;
- As mudanças do tipo adaptativa são necessárias quando há alguma modificação no ambiente, *hardware*, leis, regras e negócios, entre outros, e que tenha impacto no *software*, sendo necessárias alterações para manter o funcionamento adequado. De acordo com Sommerville (2007), esse tipo representa 18% do esforço total do esforço empregado na manutenção;
- A manutenção evolutiva é bem delicada, pois pode facilmente ser confundida com a implementação de novas funcionalidades, todavia se encaixa no contexto de manutenção evolutiva as alterações em funcionalidades que já existam, seja para otimizar ou agregar novos requisitos. De acordo com Sommerville (2007), esse tipo representa 65% do esforço total do esforço empregado na manutenção;
- A manutenção preventiva é realizada para prevenir uma possível má funcionalidade do software ou até mesmo adaptar o código tornando-o mais claro, facilitando manutenções futuras.

Sommerville (2007) não apresenta nenhuma estatística relativa ao esforço empregado para esse tipo de manutenção.

A tabela 1 apresenta a estimativa de seis autores quanto à distribuição das demandas de manutenção em relação aos tipos.

Tabela 1: Distribuição das demandas de manutenção em relação à categoria da manutenção.
Fonte: (Hatton, 2007, adaptado)

Referência	Adaptativa	Corretiva	Evolutiva	Outras
Helms e Weiss (1984)	29%	19%	28%	24%
Dekleva (1990)	46%	18%	25%	11%
Glass (1996)	42%	37%	23%	0%
Sneed (1996)	52%	9%	35%	4%
Kemerer e Slaughter (1997)	83%	9%	35%	0%
Sommervile (2007)	18%	17%	65%	0%
Paduelli (2007)	2%	31%	67%	0%

É consenso entre os autores que a manutenção de software é uma fase complicada do ciclo de vida (KERNAHAN *et al*.; 2005) (DAS et al.; 2007) (MOHAMED; 2010) e demanda a maior porcentagem de esforço (LIENTZ; SWANSON, 1981) (MAMONE, 1994) (CANFORA; CIMITILE, 2001) (FORWARD, 2002) (HUANG; TILLEY, 2003) (O'KEEFFE; Ó CINNÉIDE, 2008). Sendo que essa varia de 40% a 80% de todo o esforço empregado durante o ciclo de vida (SCHACH,1994) (AZIZ; AHMED; LAGHARI, 2009). Chan e Ho (1996) realizaram um estudo apontando que na década de 80, somando os gastos de todas as empresas de software, o valor gasto anualmente na manutenção de software é de U$ 30 bilhões. Um estudo publicado por Tan e Mookerjee (2005) revelou que o montante gasto pelas empresas dos Estados Unidos anualmente foi de U$ 70 bilhões. Rashid *et al*. afirma que nas últimas décadas o custo da manutenção de software aumentou de 35% a 90%. De acordo com Serna e Serna (2014), a fase de manutenção continua sendo a que demanda a maior quantidade de recursos.

3.3 PROBLEMAS NA MANUTENÇÃO

Bhatt (2004) indica que o número de sistemas que entram na fase de manutenção aumenta rapidamente. Mesmo com aumento significativo, Pressmann (2014) alerta sobre a marginalização dessa atividade pela literatura e enfatiza a importância da manutenção para a Engenharia de Software.

De acordo com April e Abran (2008) os problemas enfrentados no processo de manutenção de software podem ser divididos de acordo com o ponto de vista analisado, resultando em problemas externos e internos.

Os problemas externos são referentes ao ponto de vista dos clientes, e eles relatam como principais problemas: o alto custo; a demora na entrega; e a falta de clareza quanto a priorização das demandas (PIGOSKI, 1997).

April e Alan (2008) afirmam que o problema do alto custo referente a manutenção realizada não passa de uma má comunicação entre a equipe responsável por realizar a manutenção e o cliente. Nos relatórios e cobranças, comumente as manutenções evolutivas são agrupadas a manutenções corretivas, e não fica claro ao cliente a totalidade do serviço, levando-o a acreditar que foi realizada apenas uma manutenção de preço elevado.

O processo de manutenção de um software pode ocasionar em uma deterioração do código, gerando problemas para implementar novas mudanças e até mesmo a inserção de novos erros (KREMER; 1983) (OHBA, CHOU; 1989), novamente esse fator não fica claro ao cliente que acaba associando o prazo para entrega à ineficiência e má qualidade do serviço. A dificuldade que muitas equipes possuem em priorizar as demandas também influenciam negativamente no tempo de espera (APRIL; ALAN, 2008). Para os autores, no geral, a ideia negativa do cliente em relação a manutenção de software é oriunda principalmente da falta de comunicação entre o responsável pela manutenção e o cliente, assim como pelo processo de manutenção em um software ser completamente diferente do realizado em objetos concretos.

Os problemas internos são referentes aos pontos de vista dos engenheiros de softwares e gerentes, e eles relatam como principais problemas a falta de documentação e a má arquitetura do software devido a um processo precário (GLASS, 1992). Boehm (1976), assim como Ammann e Cameron (1994) veem a estruturação do código do sistema como um aspecto importante no quesito qualidade do código desenvolvido. De acordo com Hill *et al.* (2007) a primeira tarefa do processo de manutenção é entender o software, e normalmente, tem-se um sistema sem documentação (BRIAND; 2003) (HUANG, TILLEY; 2003) ou com a documentação desatualizada (AGGARWAL *et al.*; 2002) (FORWARD, LETHBRIDGE; 2002) (POOLE, et al.; 2001) (THOMAS, TILLEY; 2001**)**. O responsável pela manutenção é o principal prejudicado, ele deveria ser capaz de entender rapidamente o código para então

implementar as mudanças solicitadas, mas gasta de 40% a 60% do tempo dedicado a manutenção no entendimento do sistema (PIGOSKI; 1996) (PFLEEGER; 2002).

Em segundo plano existe o desmerecimento da função de mantenedor, essa geralmente é designada a novos funcionários e que não possuem muita experiência. Ao juntar os dois fatores o processo de oferecer manutenção a um software torna-se crítico (APRIL; ALAN, 2008).

O terceiro problema amplamente relatado é a má elaboração do software, April e Alan (2008) afirmam que devido ao tempo limitado e a carência de um bom processo de desenvolvimento de software é comum pular ou gastar um esforço menor que o ideal em atividades de *design* de software e teste. Segundo Boehm (1976) um bom design auxilia na leitura e entendimento do código, Feathers (2004) complementa afirmando que um bom design deve ser limpo. A pressão para terminar um projeto dentro dos recursos disponíveis também ocasiona a não implementação de algumas funcionalidades, gerando um *backlog* de demandas a serem desenvolvidas nos estágios iniciais da manutenção.

Estes três problemas já foram relatados por Lientz e Swanson (1980) na década de 80. Ambos considerados pioneiros no estudo dos problemas enfrentados, publicaram um livro referente aos problemas de manutenção. Após doze anos, Dekleva (1992) levantou os principais problemas enfrentados na manutenção do software, obtendo um resultado semelhante ao de Lientz e Swanson (1980). Em abril de 2005 foi realizado o *workshop Challenges on a Software Evolution (ChaSE 2005),* com trinta e sete participantes, o foco era identificar os principais desafios encontrados durante a evolução do software. Ao fim foram identificados dezoito desafios (MENS *et al*, 2005):

- **Preservar e melhorar a qualidade do software:** Carência de ferramentas e técnicas que auxiliem a preservar ou melhorar a qualidade do software;
- **Plataforma padrão para evolução do software:** Dificuldade em determinar um ambiente de trabalho comum a todos envolvidos no projeto;
- **Suporte a modelagem de evolução:** Carência de ferramentas que auxiliem na modelagem de diagramas de evolução;
- **Suporte a coevolução:** Dificuldade em manter os diversos artefatos atualizados;
- **Suporte formal a evolução:** Carência de um método formal que dê suporte a evolução do software;
- **Evolução da linguagem de programação:** As linguagens de programação e modelagem deveriam prover, de maneira direta e explicita, suporte a evolução do software;
- **Suporte a sistemas multi-linguagens:** É comum encontrar nas empresas sistemas com linguagens diferentes, logo o suporte e técnicas voltados sistemas multi-linguagens deveria ser melhorado;

- **Integrar mudanças ao ciclo de vida do software:** Identificar quando as mudanças podem ser inseridas no desenvolvimento do software;
- **Melhorar a percepção gerencial:** É necessário desenvolver entre os executivos a consciência da importância da evolução do software e de que essa é inevitável;
- **Necessidade de um sistema de versionamento melhor:** Ferramentas de versionamento é crucial para a evolução do software;
- **Integrar dados provenientes de diversas fontes:** Integrar os dados provenientes de relatório de erros, requisições de mudanças, logs de erros, entre outros;
- **Analisar a grande quantidade de dados:** Devido a quantidade de dados obtidos é necessário técnicas e ferramentas que auxiliem na rápida manipulação desses.
- **Pesquisas empíricas:** O contexto de evolução de software é carente de pesquisas empíricas. Estudos comparativos são necessários urgentemente.
- **Necessário melhorar os modelos preditivos**: Os modelos preditivos são primordiais para os gestores. Por intermédio desses é possível prever uma variedade de informações cruciais. Os atuais modelos preditivos estão longe de ser adequados.
- **Evolução de *Benchmark*:** No intuito de comparar e validar as técnicas, métodos e ferramentas a serem desenvolvidas, faz-se necessário comparar os resultados destas com um banco de dados contendo os *benchmarks* das já existentes e avaliadas. Todavia este banco de dados ainda não se encontra factível.
- **Ensinar a atividade de evolução do software:** Como inserir as problemáticas de um cenário real em uma sala de aula;
- **A teoria da evolução do software:** É necessário desenvolver novas teorias para diminuir a distância entre entender o que é a evolução do software e controlar/dar suporte a evolução;
- **Implementação de mudanças em sistemas que estão sendo executados:** Há uma necessidade urgente em dar suporte a implementar mudanças em um sistema em execução, sem ter necessidade de pará-lo.

Em uma tentativa de lidar com esses problemas que desgastam o processo de manutenção e geram despesas, foram propostos modelos de manutenção de software, tais como a ISO/IEC 14764 e o MANTEMA (POLO et al., 2003).

3.4 PROCESSO DE MANUTENÇÃO

A fase de manutenção possui peculiaridades que a torna diferente da fase de desenvolvimento, por exemplo, durante o desenvolvimento de um software existem requisitos que guiam o desenvolvedor. Na fase de manutenção, as mudanças são feitas por intermédio de requisições registradas pelos usuários (WEBSTER; OLIVEIRA; ANQUETIL, 2005). De acordo com Basili *et al* (1996), essa metodologia

utilizada na manutenção representa um risco ao projeto, uma vez que se gasta mais tempo analisando as solicitações do que efetivamente implementando-as.

Essas diferenças obrigam os desenvolvedores a procurar ferramentas diferentes das utilizadas no desenvolvimento para auxiliar na fase de manutenção. Os processos específicos para manutenção se enquadram nessas ferramentas auxiliadoras.

3.4.1 ISO/IEC 14764

Baseando-se no processo de manutenção apresentado na norma ISO/IEC 12207 de 1995 (ISO/IEC, 2008), a norma ISO/IEC 14764 descreve um passo a passo das atividades e tarefas que devem ser adotadas para realizar a implantação do processo de manutenção. O principal objetivo é dar suporte ao planejamento, gerenciamento e execução das atividades contidas no processo de manutenção (ISO/IEC, 2006).

A ISO/IEC 14764 apresenta seis atividades distintas, apresentadas na **Error! Reference source not found.**. Cada atividade com suas entradas, tarefas, controles, suporte e saídas. As entradas são os insumos para as tarefas, por sua vez, as tarefas são instruções atômicas e que devem ser seguidas durante a execução da atividade. O controle são um conjunto de orientações que visam garantir saídas corretas. O suporte são recursos utilizados para auxiliar o desenvolvimento das tarefas. Por fim, as saídas são os resultados finais da atividade.

De acordo com a ISO/IEC 14764, as seis atividades são descritas a seguir e em ordem de execução:

Implementação do processo: Durante essa atividade deve-se definir as estratégias e os procedimentos que serão utilizados no desenvolvimento do processo de manutenção. Para alcançar esse objetivo, é necessário averiguar os limites da manutenção, levantar os recursos disponíveis, realizar uma estimativa do custo total, avaliar o grau de manutenabilidade do software. Também é realizado uma análise de impacto e riscos.

A atividade tem como entradas a *baseline* do projeto, a documentação do sistema e uma solicitação de mudança. E as saídas são plano de manutenção inicial, *feedback* para os usuários e a gerência de configuração;

Análise dos problemas e modificações: Durante essa atividade deve-se entender o problema, definir as ações que serão realizadas durante a manutenção e obter a aprovação do plano desenvolvido.

A atividade tem como entradas a *baseline* atualizada, repositório, a documentação do sistema e os artefatos obtidos na atividade anterior. As saídas são a análise de impacto, solução sugerida, aprovação da requisição de mudança e da manutenção e a documentação atualizada;

Implementação das modificações: Durante essa atividade as modificações aprovadas na atividade anterior são desenvolvidas e testadas. Para isso, a equipe responsável deve identificar todos elementos que serão alterados, desde o código a documentação.

A atividade tem como entradas a solicitação de mudança já aprovada, a *baseline* atualizada e os artefatos obtidos na atividade anterior. As saídas são o software alterado, a documentação atualizada e os testes executados;

Revisão e aceitação das modificações: Durante esta atividade a equipe responsável pela manutenção deve certificar-se que as alterações realizadas no software estão corretas e de acordo com o planejado. Também é necessário assegurar que a documentação foi alterada corretamente.

A atividade tem como entradas o software alterado e os artefatos obtidos na atividade anterior. As saídas são: *baseline* atualizada, documento aprovando a manutenção e o resultado dos testes realizados;

Migração: Durante essa atividade a equipe responsável pela manutenção do software deve migrar o software alterado, substituindo a versão antiga. Para alcançar este objetivo, é necessário identificar os artefatos modificados e que requerem a migração, também é necessário criar um plano de migração e realizar um *backup* dos dados antigos;

A atividade tem como entradas o novo ambiente do software, o ambiente antigo, a *baseline* antiga e a nova. As saídas são o *backup* dos artefatos antigos, a notificação de conclusão da migração, o plano de migração, as ferramentas utilizadas para realizar a migração e o software migrado.

Retirada: Todo software possui seu ciclo de vida, e quando esse chega ao final, o software deve ser descontinuado. A equipe de manutenção responsável pelo software deve realizar uma análise para verificar se o software ainda é capaz de competir no mercado, caso a conclusão seja negativa é necessário executar as ações necessárias para efetivar sua retirada. Antes de retirar o software, a equipe de manutenção deve criar um plano de retirada, notificar todos os usuários que utilizam o software e manter o *backup* dos dados e produtos antigos.

A atividade tem como entradas o software antigo, o software novo e os ambientes antigos e novo. As saídas são o plano de retirada, a notificação de conclusão, o *backup* dos dados e produtos e o software descontinuado.

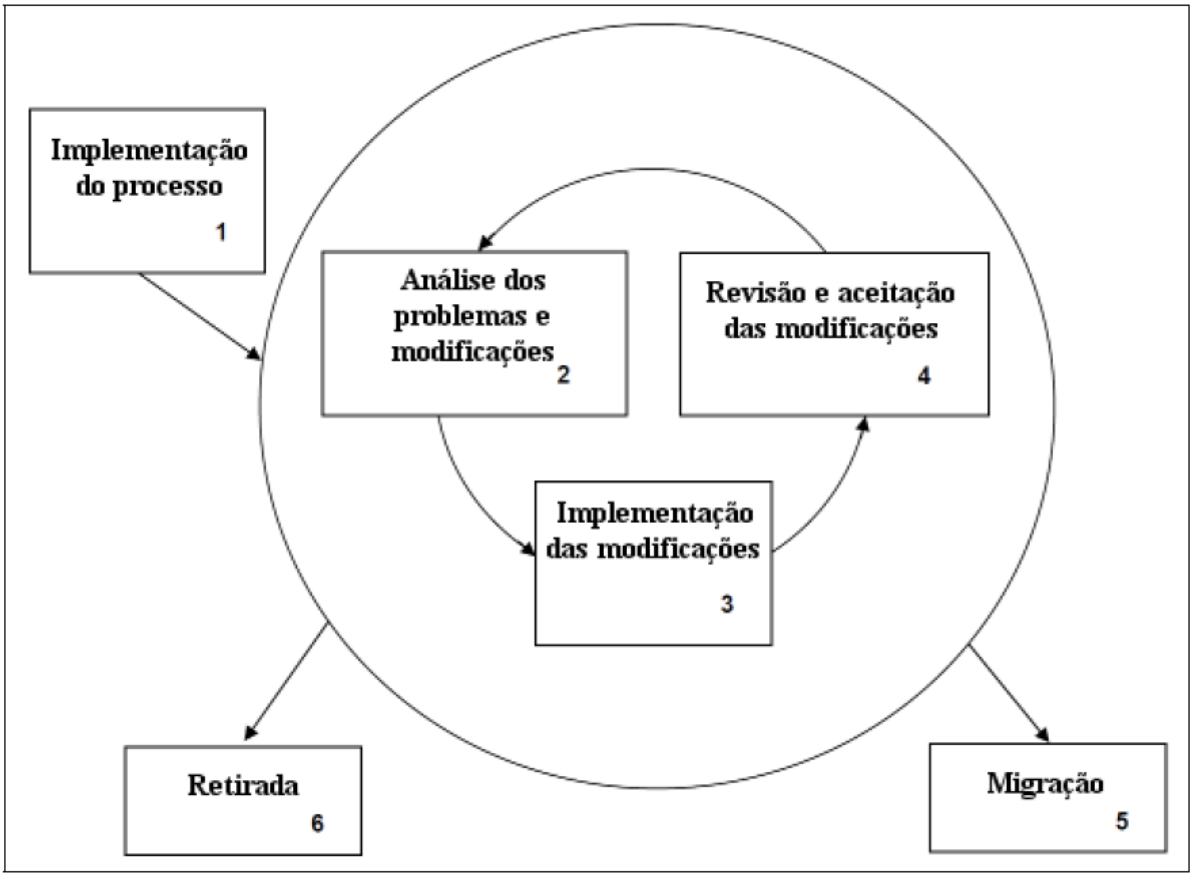

Figura 6: Modelo do Processo de Manutenção da ISO/IEC 14764. Fonte: (ISO/IEC 14764, adaptado).

3.4.2 MANTEMA

A norma ISO/IEC 12207 apresenta um conjunto de atividades e tarefas que devem ser seguidos ao implementar um processo de manutenção (ISO/IEC 12207). Entretanto a ISO informa as atividades que devem ser utilizadas no processo, mas explica como colocar em prática. A carência de detalhamento das atividade e tarefas acarreta na dificuldade dos mantenedores em aplica-las (POLO et al., 1999a). O MANTEMA é uma metodologia elaborada a partir da ISO/IEC 12207 e visa preencher a falta de detalhes que não foi contemplada. Como é possível visualizar na Figura 7**Error! Reference source not found.**, o MANTEMA foi dividido em três grupos de atividades descritas a seguir (POLO et al., 1999a):

Atividades Iniciais: Esse grupo abrange as atividades realizadas antes do atendimento das solicitações de manutenção, englobando as seguintes atividades apresentadas em ordem de execução (POLO et al., 1999b):

- Iniciação e Agrupamento de informação;
- Preparar a proposta de manutenção;

- Contrato;
- Planejar as relações com os clientes e fornecedores;
- Aquisição de conhecimento;
- Desenvolver o plano de manutenção;
- Definir o procedimento de requisição de mudanças;
- Implementar o processo de gerenciamento de configuração;
- Preparar o ambiente de testes;
- Recepcionar a requisição de mudança;
- Identificar o tipo de manutenção necessária.

Atividades de Atendimento: Esse grupo abrange as atividades realizadas com o propósito de executar as mudanças e/ou as correções requisitadas pelos usuários. Diferentemente dos outros dois grupos, as atividades desse são definidas baseando-se no tipo de manutenção necessária. Polo *et al.* (1999b) propõem para a manutenção corretiva urgente as seguintes atividades:

- Investigar e analisar as causas do erro;
- Realizar urgentemente as ações corretivas;
- Documentar as alterações;
- Verificar as alterações;
- Documentar os testes realizados e seus resultados;
- Migrar a nova versão do software para o ambiente de produção;
- Realizar *backup* da documentação gerada.

Atividades Finais: Esse grupo abrange as atividades realizadas com o propósito de encerrar o atendimento da requisição de mudança. Polo *et al.* (1999b) recomenda realizar uma avaliação da necessidade de retirada ou migração do software ao término do atendimento. Também recomenda realizar um *backup* dos dados antigos.

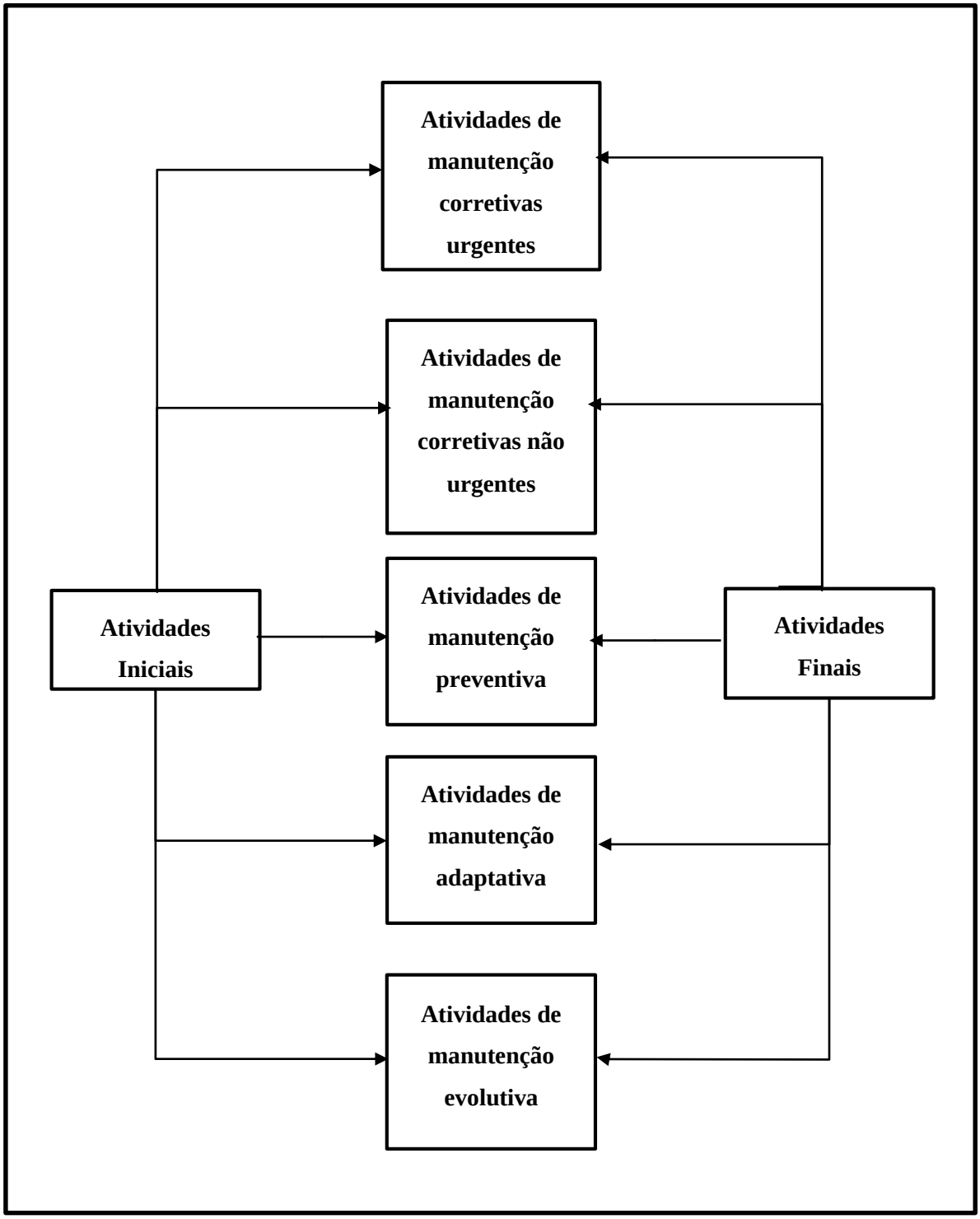

Figura 7: Macroestrutura do Processo de manutenção MANTEMA.

Fonte: (POLO et al., 1999a, traduzido):

3.4.3 MODELO DE MANUTENÇÃO DE TAUTE

O modelo de Taute, ao contrário dos *frameworks* e modelos apresentados anteriormente, é um modelo desenvolvido para ser prático e de fácil entendimento. Possui oito fases sucintas e bem definidas, essas podem ser visualizadas na

Figura **8** e são descritas a seguir (PETER; PEDRYCZ, 2001):

Solicitar: A requisição de mudança é criada e recebida pela equipe responsável pela manutenção do software;

Estimar: Define-se os custos, o tempo, os recursos e o impacto da requisição de mudança;

Agendar: Define-se a data para implantar a nova versão do software, também é nessa fase onde a documentação é planejada e preparada;

Programar: Cria-se uma cópia da versão atual do software e é implementada as alterações requisitadas;

Testar: Testa-se a versão do software com as mudanças implementadas;

Documentar: Com a aprovação da nova versão do software, a documentação é atualizada;

Liberar: A nova versão do sistema e a documentação atualizada são disponibilizadas e inicia-se os testes de aceitação;

Operar: Finalmente o software é migrado para o ambiente real, permitindo o usuário final utiliza-lo efetivamente.

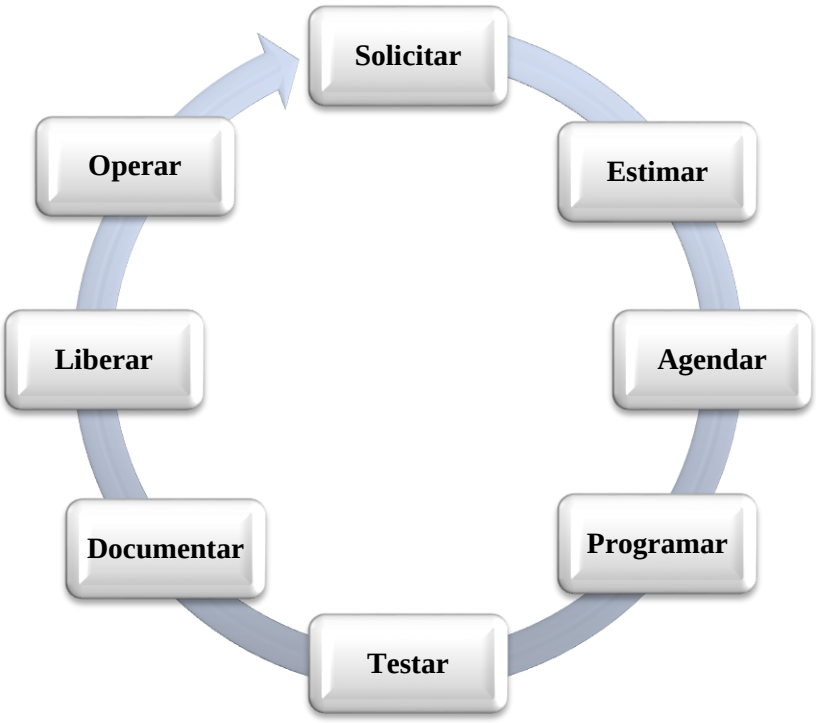

Figura 8: Modelo de manutenção de Taute. Fonte: (PETER; PEDRYCZ, 2001)

3.5 CONSIDERAÇÕES FINAIS

Neste capítulo apresentou-se, inicialmente, uma breve contextualização do cenário de manutenção e dos impactos causados no ciclo de vida como um todo. Em seguida foram apresentados os problemas enfrentados durante essa fase e modelos específicos para o processo de manutenção.

O processo de manutenção necessita lidar com frequentes mudanças no escopo de suas demandas e na ordem de execução. No intuito de agrega-las é necessário adotar um processo de gestão de demandas capaz de lidar adequadamente com essas mudanças. Portanto, no Capítulo 3, será caracterizado um *framework* que apresenta capacidade de atender essa necessidade.

CAPÍTULO 4 – FRAMEWORK KANBAN

4.1 CONSIDERAÇÕES INICIAIS DO CAPÍTULO

Neste Capítulo apresenta-se uma síntese do *Framework Kanban*, características do *framework Kanban*, sua criação, seu uso como metodologia de gestão, sua aplicabilidade no processo de manutenção de software. Também apresenta variações do *Kanban* Tradicional, descreve as vantagens do *Kanban* eletrônico, assim como algumas ferramentas *online* que podem ser utilizadas.

4.2 METODOLOGIAS ÁGEIS

A metodologia tradicional de desenvolvimento é amplamente criticada devido à dificuldade de pensar em todos os requisitos de um software logo no início de um projeto (LARMAN; BASILI, 2003) (POPPENDIECK; POPPENDIECK, 2003) (RAMASUBU; BALAN, 2009). Pressman (2011) complementa essa crítica expondo a incerteza do desenvolvimento do projeto em relação ao impacto de possíveis mudanças no mercado. Ambler (2002) também acredita que documentação excessiva apenas prejudicava o projeto, o ideal seria documentar apenas o necessário para iniciar o desenvolvimento e estender a atividade de documentação para toda duração do projeto. Em face dessas dificuldades, as metodologias ágeis ganharam destaque tanto na prática (AGILE ALLIANCE, 2011), quanto na literatura (BOEHM, 2003).

A metodologia ágil é definida por Larman (2003) e Hunt (2005) afirmam que ser ágil é ser aberto a mudanças e respondê-las de maneira rápida. Em contrapartida Shore e Warden (2008) afirmam que para ser ágil é necessária uma mudança de cultura e consequentemente aumenta-se a produtividade. Consolidando a visão de Shore e Warden, Âgefalk *et al*. (2005) define metodologias ágeis como um conjunto de práticas para desenvolvimento de software que quando aplicadas a um projeto, auxiliam a lidar com mudanças ocorridas durante o desenvolvimento, trazendo agilidade e flexibilidade (COHN, FORD; 2003) (QUMER, HERDERSON-SELLERS; 2008) (CHENG, JANSEN, REMMERS; 2009). Também possibilitam produzir softwares com maior qualidade em um curto período de tempo (LIVERMORE; 2007) e permitem um feedback rápido e constante (CAO, RAMESH; 2008). As metodologias ágeis e suas práticas devem estar alinhadas com os pilares estabelecido no Manifesto Ágil, publicado em 2001 (FOWLER, 2001).

O Manifesto Ágil é um marco para o paradigma ágil, consolidando a essência das práticas ágeis em quatro pilares, sendo eles (HIGHSMITH, 2002):

- Indivíduos e iterações são mais importantes do que processos e ferramentas;
- Software funcional é mais importante do que documentação detalhada;
- A colaboração do cliente é mais importante do que a negociação de contratos

- Responder às mudanças é mais importante do que seguir um plano.

Apesar de sua importância para consolidação desse novo paradigma, as metodologias ágeis antecedem a publicação do manifesto. O primeiro modelo ágil foi apresentado em 1995, e é conhecido como *Dynamic Systems Development Method* (STAPLETON; 2003). Ainda no mesmo ano, foi publicado o primeiro artigo referente ao *Scrum* para o gerenciamento do desenvolvimento de software (SCHAWBER, BEEDLE; 2001) (SCHWABER,1995), e também foi apresentado o *Extreme Programming* (BECK; 2000). Até então as metodologias ágeis eram referentes ao desenvolvimento de software, foi em 1998 que o processo se tornou um dos focos desse novo paradigma, permitindo a utilização dos conceitos *Lean*, *Kanban*, entre outros (COCKBURN; 2004) (PALMER, FELSING; 2002) (POPPENDIECK, POPPENDIECK; 2003) (HIGHSMITH; 1999).

4.3 CRIAÇÃO DO KANBAN

O *Kanban* é um *framework* desenvolvido por *Taiichi Onho*, em meados da década de 50 (OHNO, 1997). Segundo Anderson (2010), fazendo uso dos conceitos do sistema de administração *Just in Time*, o *Kanban* foi implementado para apoiar o controle de produção da fábrica Toyota e tornou-se indispensável ao Sistema Toyota de Produção (Anderson, 2010).

Gross e Mcinnes (2003) ressaltam que a Toyota obteve sucesso ao atingir a redução de custos e o gerenciamento efetivo de seus recursos, mas foi com a "Recessão Global", em 1970, que as demais indústrias compreenderam que era possível realizar um gerenciamento da quantidade de trabalho em progresso, reduzindo os custos associados ao estoque de materiais excedentes

Cinco décadas após a implementação na fábrica da Toyota, Anderson (2010) visualizou o potencial oferecido pelo *framework Kanban* para sanar antigos problemas de desenvolvimento de software e foi, oficialmente, o primeiro a utilizar o *Kanban* nessa área.

4.4 O FRAMEWORK KANBAN

O *framework Kanban* não se impõe aos costumes já empregados, ao contrário, respeita papéis, responsabilidades e cargos já definidos, visando sempre a qualidade e a busca por mudanças incrementais e evolucionárias. O Kanban se baseia em cinco pilares (ANDERSON, 2010):

- Foco na qualidade;
- Redução do trabalho em progresso, permitindo entregas frequentes;
- Redução da variação do fluxo de processo;
- Priorização de demandas;
- Processo Puxado.

Os cinco pilares constituem a base para definir a adoção dos processos e dos artefatos que serão empregados para alcançar o objetivo principal, isto é, a otimização do fluxo de trabalho a partir da exposição do processo de desenvolvimento e identificação dos gargalos.

O *framework Kanban* não possui uma ordem correta de implementação, todavia, para facilitar sua utilização, os autores comumente elaboram um passo a passo, ordenando sequencialmente as atividades que devem ser realizadas (ANDERSON, 2010) (GROSS; MCINNIS, 2003) (BOEG, 2011). Apesar das abordagens realizadas tanto por Gross e Mcinnis (2003), quanto por Boeg (2011) terem sido baseadas na de Anderson (2010), a de Boeg (2011) é a mais detalhada.

4.4.1 ATIVIDADES DO KANBAN - CONCEITOS

De acordo com Boeg (2011), o *Kanban* pode ser implementado seguindo um conjunto de dez atividades. A primeira atividade é visualizar o fluxo, ou seja, para se melhorar um processo é necessário entendê-lo antes e mapeá-lo. Após mapeado, é necessário reproduzi-lo no Quadro *kanban*, como o apresentado na Figura 9.

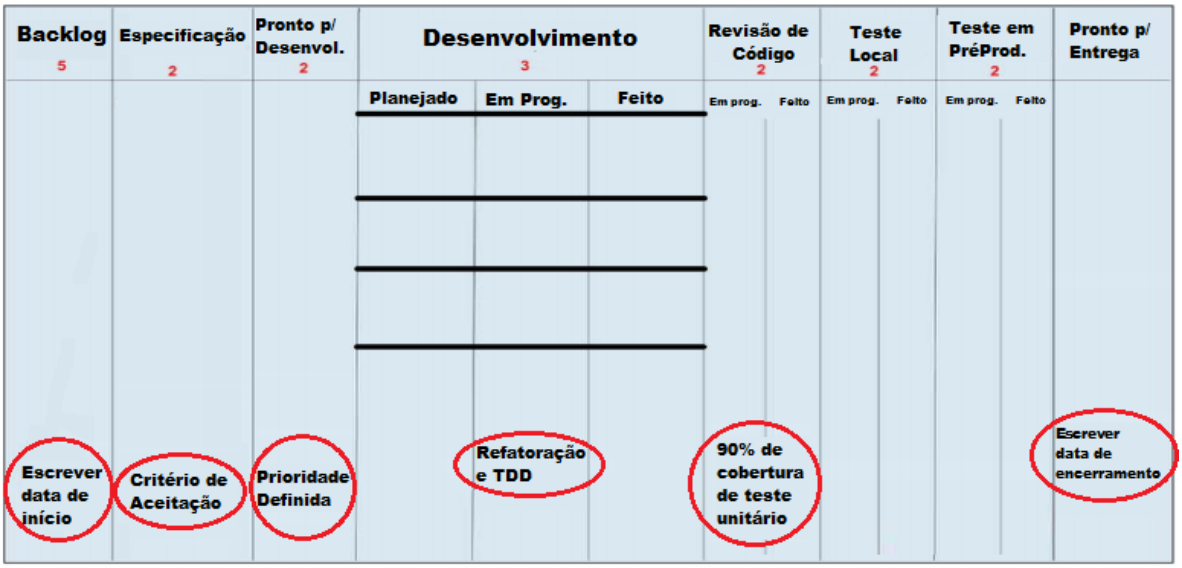

Figura 9: Quadro Kanban com as fases, WIP e Políticas de Qualidade. Fonte: (BOEG, 2011, traduzido).

Essa atividade proporciona uma transparência no fluxo, permitindo verificar como seu trabalho impacta outras pessoas e até mesmo as fontes de desperdícios. Uma vez conhecida as fases, essas deverão conter um limite de atividades, também conhecido como WIP (*Work in Progress)* e políticas de qualidade.

Boeg (2011) reforça que limitar o WIP é uma pratica para estabelecer uma quantidade máxima de trabalho em progresso que poderá ser desenvolvido em cada fase. Ao se definir um número é necessária cautela, pois um limite poderá causar ócio na equipe, e um limite alto vai contra a política de

otimização). Para estabelecer um número ótimo, Anderson (2010) sugere que não seja gasto tempo tentando estimar números, o ideal é arriscar um valor e ajustá-lo aos poucos. Recomenda-se que o limite do WIP esteja localizado no início da coluna que representa a fase, como é exibido na Figura 9.

Segundo Boeg (2011), as políticas de qualidade são um dos principais focos do *Kanban. E*las visam otimizar a qualidade do software, e um esforço significativo é realizado para torná-las explicitas. As políticas mais conhecidas ou praticadas são relativas a fase de desenvolvimento, mas é recomendável que todas as etapas do processo tenham suas políticas. Ao analisar a Figura 9, é possível observar que as políticas de cada etapa estão explicitas no rodapé de cada fase do processo.

Após visualizar o fluxo, delimitar os WIP e definir as políticas de qualidade, é necessário ajustar o ritmo de trabalho, também conhecido como cadência (ANDERSON, 2010). No sistema *Kanban* não é necessário estabelecer uma cadência para o processo de desenvolvimento como um todo. Pode ser definida para cada fase do processo, todavia, as mais importantes são as cadências de planejamento e entrega (BOEG, 2011). O importante é equilibrar tanto a cadência interna, quanto a externa. Quanto mais funções forem entregues, menor será o custo transacional, e quanto mais demorar para entregar, maior será o custo de espera.

A Figura 10 apresenta o custo total, o custo relativo entre o custo transacional e o de espera, como sendo uma curva do tipo "u". Mesmo que não se encontre o ponto ótimo, com um erro de 10 a 15%, ainda é possível obter bons resultados.

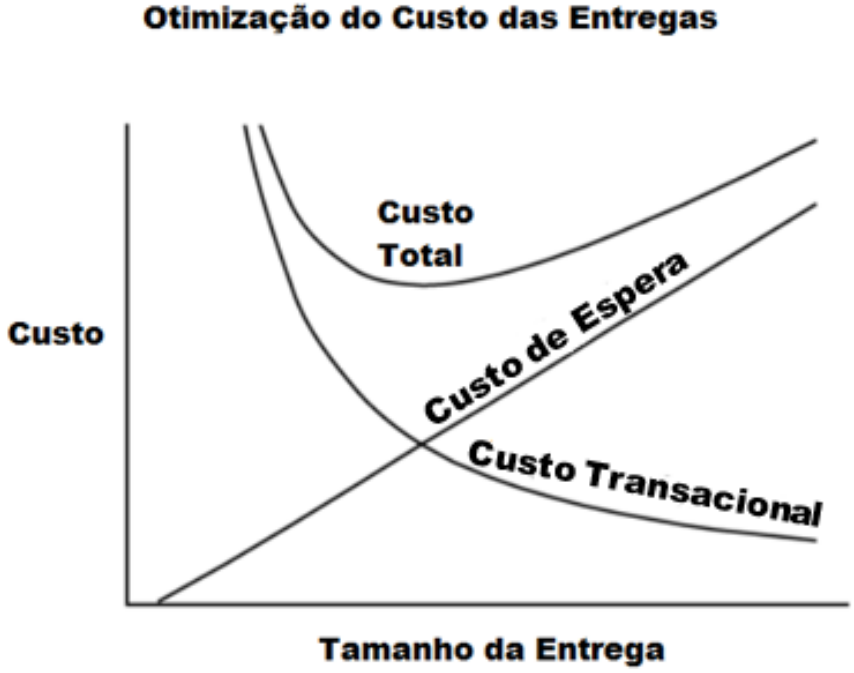

Figura 10: Otimização do custo das entregas. Fonte: (REINERTSEN, 2009, traduzido).

Em paralelo as atividades citadas, também se deve realizar a priorização das demandas. Para isso é necessário analisar a viabilidade técnica de cada demanda, assim como os riscos e o custo que será gerado caso ela não seja atendida. A priorização é exibida no Quadro *kanban* por meio da ordem dos cartões, quanto mais a cima, maior será sua prioridade (ANDERSON, 2010).

As próximas atividades colaboram na priorização das demandas, assim como Anderson (2010), Boeg (2011) sugere quatro classes e para cada classe deve ser estabelecido um acordo de nível de serviço (ANS – *Service Level Agreement* /SLA).

Classe Padrão e os ANS: A classe padrão engloba as demandas relativas aos defeitos cosméticos e histórias de usuário. Não apresentam custo adicional ou prioridade pré-estabelecida. Normalmente essa classe possui uma média de quinze dias para conclusão das demandas, 90% estão dentro do prazo de vinte e um dias e todas são atendidas em no máximo trinta dias.

Classe Prioritária e os ANS: A classe prioritária engloba defeitos críticos e histórias de usuário prioritárias, apresentam um custo adicional em relação as demandas da classe padrão devem ser priorizadas independente da etapa que se encontram no fluxo. Comumente essa classe possui uma média de oito dias para a conclusão da demanda, 90% estão dentro do prazo de treze dias e todas são atendidas em o máximo dezoito dias.

Classe de Prazo Fixo e os ANS: A classe de prazo fixo assemelha-se a classe padrão, exceto quando as demandas estão perto de atingir o prazo de implementação. Caso isso ocorra, ela se torna prioritária e pode gerar um custo adicional superior a classe padrão e a prioritária. Comumente 98% das demandas dessa classe são atendidas dentro do prazo limite.

Classe Urgente e os ANS: A classe Urgente é reservada para a resolução de defeitos impeditivos. Possui um custo adicional superior a qualquer outra classe, assim como é prioridade sobre qualquer outra demanda. Inclusive para atender esta classe é viável desconsiderar o limite do WIP. Normalmente essa classe possui uma média de dois dias para conclusão das demandas, 90% estão dentro do prazo de três dias e todas são atendidas em no máximo quatro dias.

Essas classes são apenas modelos a serem seguidos, provavelmente ao adaptar o *Framework Kanban* à um projeto, é necessário implementar classes específicas que atendam melhor as necessidades identificadas.

As últimas duas atividades são o gerenciamento de fluxo e a otimização contínua. Estas se interpolam em algumas ocasiões, mas possuem objetivos e metodologias diferentes.

No **Gerenciamento de Fluxo**, o limite do WIP é aprimorado e deve sanar os gargalos, se existirem. Os gargalos podem ser gerados devido, principalmente, a algumas dificuldades da equipe em lidar com alguma fase do processo ou até mesmo pelo limite de WIP que ainda necessite de ajustes. No

primeiro caso deve-se criar *buffers* como medida de contingência, e paralelamente é necessário identificar a causa desse atraso no fluxo. No segundo caso, o WIP ainda não condiz com a realidade de trabalho da equipe e é necessário ajusta-lo até alcançar um número ótimo. E mesmo que o WIP não seja o causador do gargalo, deve-se avalia-lo de qualquer maneira, a fim de verificar se é possível otimiza-lo (ANDERSON, 2010).

A **Otimização Contínua** é realizada a partir de *feedbacks* e avalia todo o *Framework Kanban* implementado. Ao se analisar os dados coletados por intermédio das métricas estabelecidas, é possível identificar alvos de melhorias. Esses podem ocorrer em qualquer atividade, desde os ajustes no fluxo do processo de desenvolvimento até o ajuste das métricas utilizadas. Boeg (2011) ressalta que que jamais se deve quebrar algo estabelecido, caso seja necessário, se deve muda-lo.

Como foi dito no início desta seção, o *Kanban* é um *framework* e não um processo, as dez atividades aqui explicadas, no geral, não possuem uma ordem correta. Elas podem seguir o roteiro atividade por atividade ou podem ser implementadas ao mesmo tempo. O que decidirá será o plano estratégico e a capacidade da equipe.

Após introduzir os conceitos acima, finalmente é possível estabelecer as algumas métricas que serão úteis no processo de avaliação e melhoria continua.

4.4.2 ATIVIDADES DO KANBAN - MEDIÇÃO

Boeg (2011) ressalta que se os resultados obtidos, a partir das medições, não forem utilizados para melhorar o processo, então a medição não deve ser realizada. Com base nessa premissa, Boeg (2011) sugere quatro técnicas: fluxo cumulativo; tempo de ciclo; índice de defeitos; e itens bloqueados, detalhados em seguida.

Fluxo Cumulativo: O Gráfico de Fluxo Cumulativo exibe as atividades para cada etapa do processo em relação ao tempo. Ao analisar a Figura 11, percebe-se que além do trabalho realizado é possível identificar outras informações essenciais. Quando o espaço entre duas linhas na área de trabalho em progresso aumenta, pode ser um sinal de gargalo. Se a linha de *backlog* for mais inclinada que a de *done*, a quantidade de trabalho adicionado é maior que a capacidade atual. Ao projetar a linha de *backlog* e a de *done* é possível estimar a data final de entrega. Por fim também é possível extrair o tempo médio do ciclo, assim como a quantidade de itens na fila.

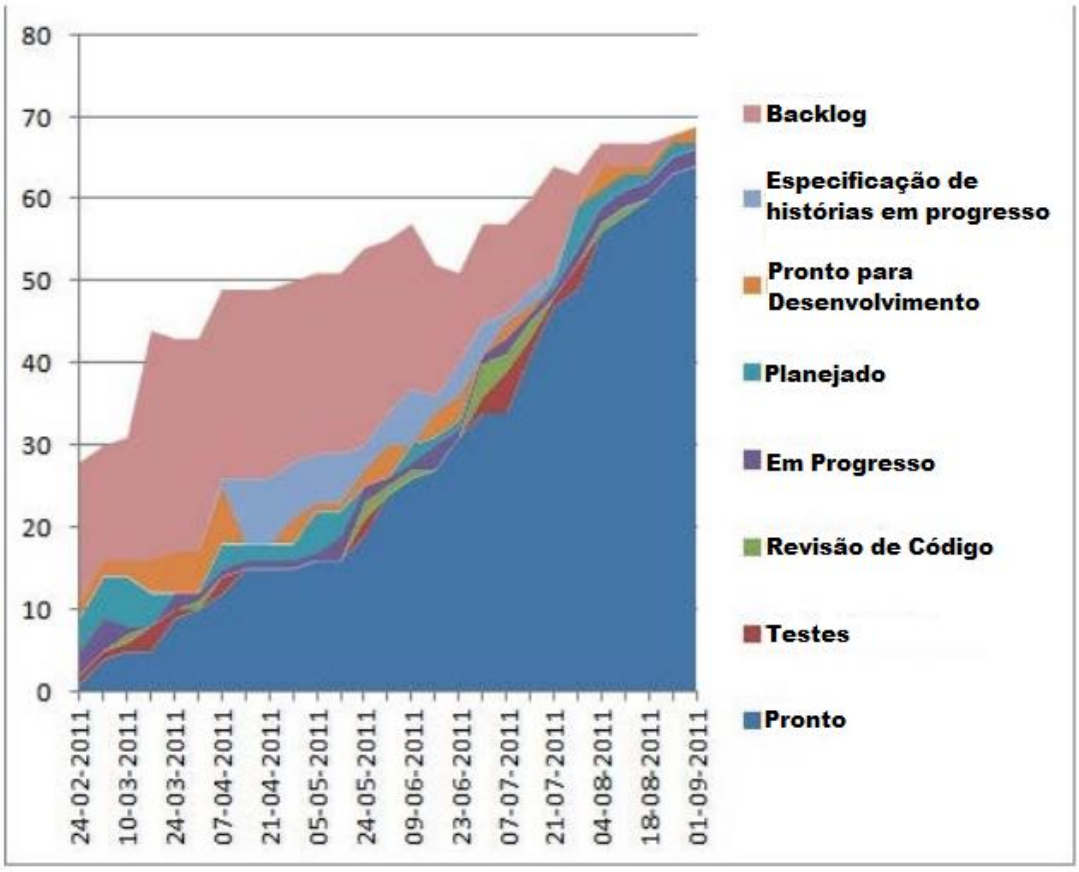

Figura 11: Gráfico de Fluxo Cumulativo. Fonte: (BOEG, 2011, traduzido)

Tempo de Ciclo: O gráfico de Tempo de Ciclo, Figura 12, apresenta o tempo estimado para cada fase do processo, com um grau de confiabilidade maior do que a média geral exibida no gráfico de Fluxo Cumulativo. Inicialmente se deve criar um gráfico para cada fase do processo e contabilizar a quantidade de dias gastos. Após algumas marcações é possível estabelecer uma reta de tendência e então dizer ao cliente uma faixa de tempo na qual, provavelmente, a demanda será concluída. A Figura 12 mostra um exemplo de como ficará o gráfico após sete atividades marcadas, lembrando que o Eixo X representa o Número da tarefa e o Y a Quantidade de dias.

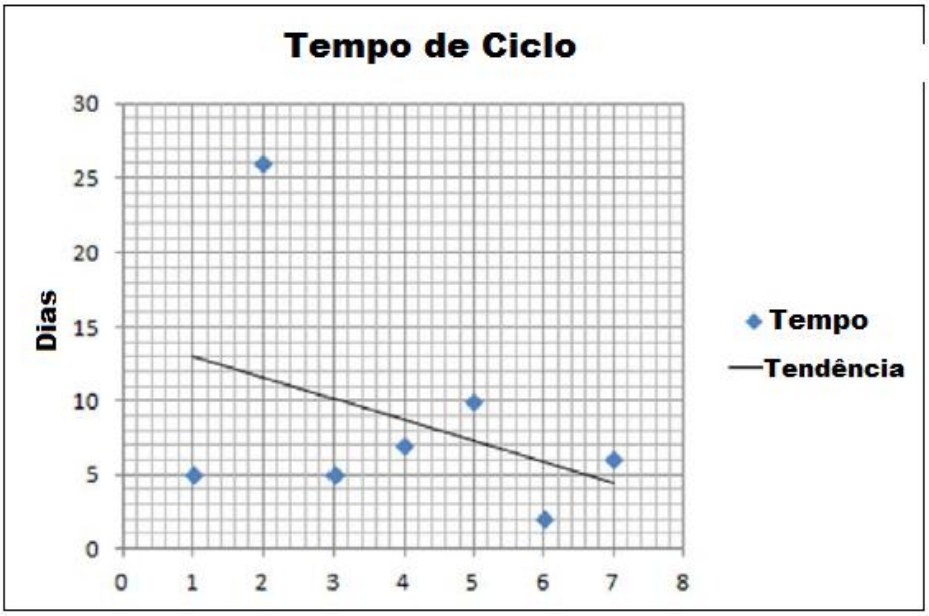

Figura 12: Gráfico de Tempo de Ciclo. Fonte: (BOEG, 2011, traduzido)

Índices de Defeitos: Problemas de qualidade são demasiadamente caros. Medir o índice e o número total de defeitos é uma boa prática para evitar que os problemas saiam do controle. Deve-se medir o total de itens não resolvidos e verificar quais são novos, permitindo identificar as reincidências. No geral o número de defeitos deve-se manter abaixo de vinte, caso esse valor seja ultrapassado, será necessário gastar mais tempo, que o considerado ideal, para controlar a situação. Na Figura 13 é possível visualizar um exemplo do gráfico utilizado para essa medição, onde o Eixo X representa o número de semanas e o Y representa a Quantidade de defeitos.

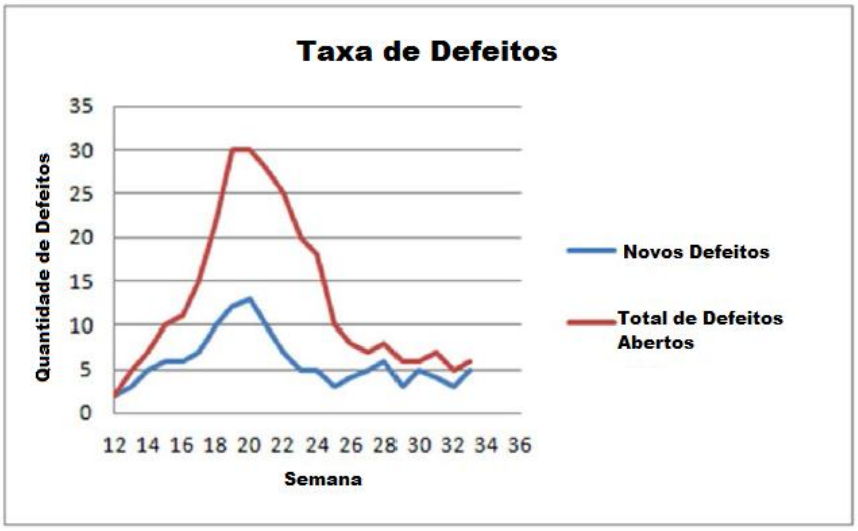

Figura 13: Gráfico de Índice de Defeitos. Fonte: (BOEG, 2011, traduzido)

Itens Bloqueados: As demandas podem ser bloqueadas por diversos motivos, o que importa é a capacidade da equipe em resolver esses empecilhos e como o fluxo é impactado. O gráfico apresentado na Figura 14 exibe a quantidade de itens bloqueado e a média do tempo de bloqueio, onde o eixo X representa a quantidade de semanas e o Y a quantidade de bloqueios. Também é necessário identificar, no Quadro *kanban*, as demandas que estão bloqueadas. Normalmente é anexado um alerta ao cartão, explicando o motivo do bloqueio.

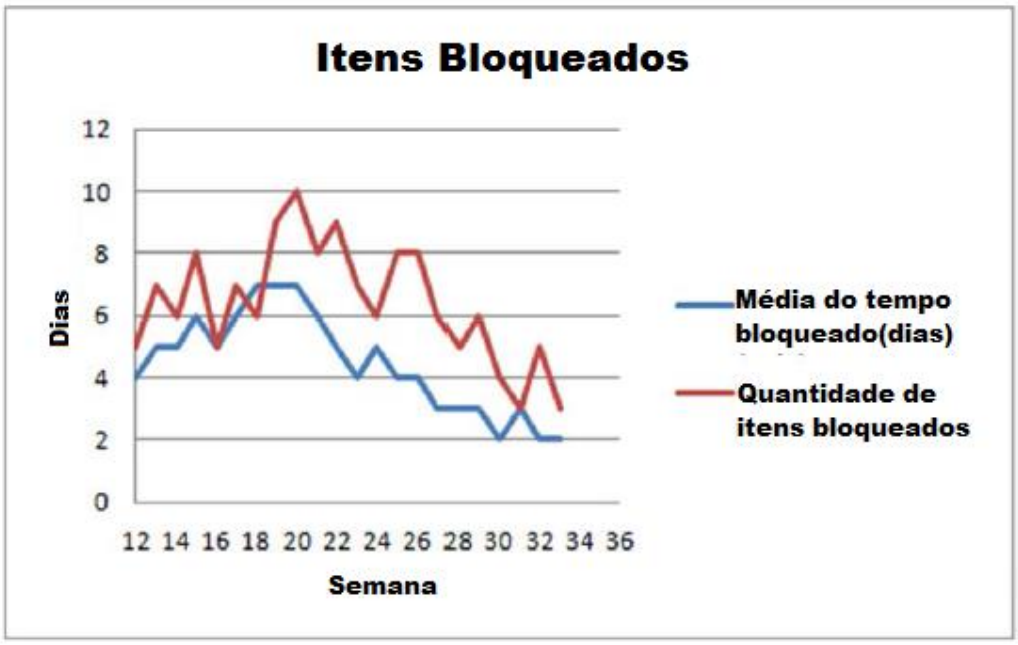

Figura 14: Gráfico de itens bloqueados. Fonte: (BOEG, 2011, traduzido)

4.5 KANBAN NO PROCESSO DE MANUTENÇÃO DE SOFWARE

De acordo com Wang, Conboy e Cawley (2012), o processo de desenvolvimento de software permite que equipe planeje e decida como será definido a próxima entrega, e até mesmo organizar tudo em um cronograma. O mesmo não acontece no processo de manutenção de software, constantemente surgem novas demandas de manutenção e devido os seus impactos, não é possível esperar o atual ciclo acabar e planejando-as para a próxima entrega.

O *framework Kanban* trabalha com o conceito de cadência e não estipula entregas fechadas para um determinado período de tempo (Anderson, 2010), atendendo a necessidade de alocar rapidamente as demandas de manutenção e prioriza-las, bastando coloca-las no topo da lista de atividades a serem desenvolvidas.

Um estudo de caso realizado por Maassen e Sonnevelt (2010) analisou uma das três maiores empresas de seguros dos Países Baixos e concluiu que a gestão de demandas de manutenção estava em colapso por causa do processo de gestão que foi empregado. Esse tinha um ciclo que

durava duas semanas para ser concluído e durante este período dificilmente uma nova tarefa era alocada. Após a troca pelo *Kanban*, a empresa tornou seu processo mais dinâmico, sem períodos fechados de entrega e conseguiu dinamizar a alocação das demandas de manutenção.

Outro estudo foi feito na empresa Fundamo, por Graves (2011). O setor de tecnologia atendia demandas frequentes de manutenabilidade dos diversos sistemas utilizados. Com o intuito de implementar uma metodologia ágil e tratar as requisições da maneira mais adequada possível, optou-se pelo uso do *Kanban*. Resultando em um processo que admitia a inclusão de novas demandas a qualquer momento, desde que o WIP não fosse extrapolado.

4.5.1 PROCESSO DE MANUTENÇÃO E O WIP

O WIP é basicamente o que torna a dinâmica do *Framework Kanban* em uma dinâmica "puxada". Quando o limite é alcançado, deve-se progredir com as demandas para que outras possam ser iniciadas, evitando que os membros da equipe fiquem trocando de tarefa e desperdiçando tempo (*CONCAS et al*., 2013). Para provar que o WIP é, de fato, importante, Concas et al (2013) realizou alguns experimentos com e sem limites de trabalho.

O primeiro experimento foi realizado na Microsoft e o segundo foi realizado em uma empresa Chinesa. Durante quatro anos foram realizadas simulações para verificar o comportamento do processo de manutenção com o WIP limitado e com o WIP sem limite.

Ao analisar a Figura 15 e a
Figura 16, é possível visualizar que o estudo de caso feito com WIP limitado apresentou no dia 1084 uma quantidade de 4179 incidentes fechados, enquanto o sem limite de WIP apresentou no mesmo dia 3879 incidentes fechados, uma diferença de 7%.

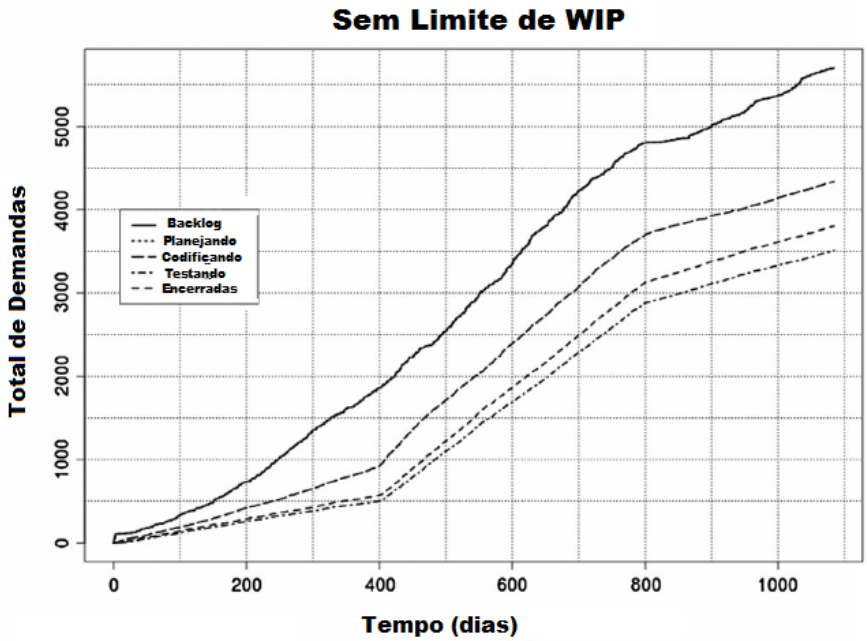

Figura 15: Gráfico de demandas por tempo com o WIP sem limites.

Fonte (Concas et al., 2013, traduzido).

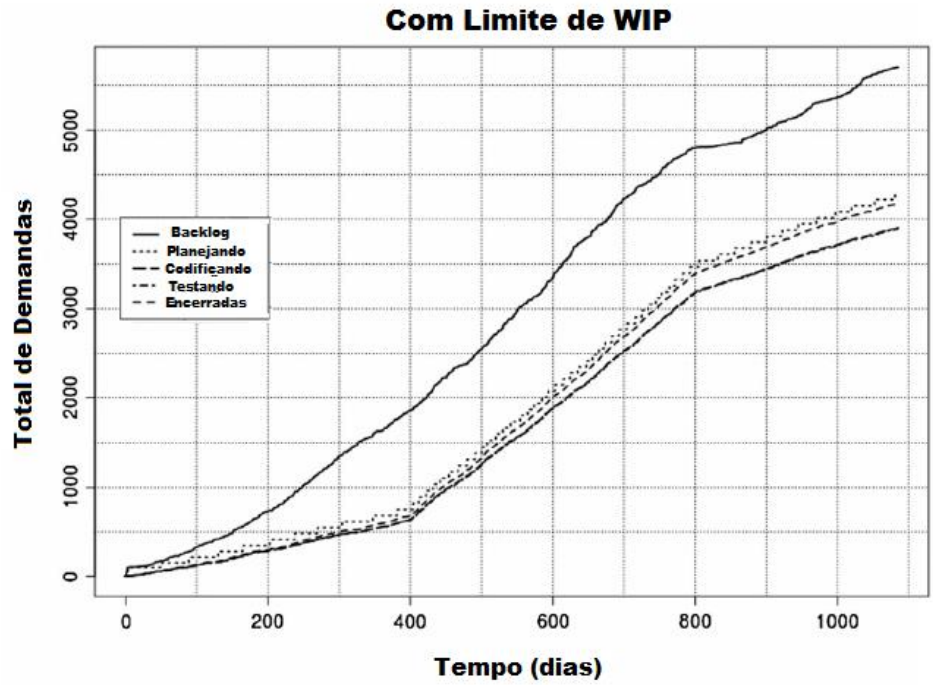

Figura 16: Gráfico de demandas por tempo com o WIP limitado.

Fonte (Concas et al., 2013, traduzido)

Concas et al (2013) afirma que esta diferença ocorre pois com o limite, os integrantes da equipe focam sua atenção na demanda sob sua responsabilidade, evitando a troca constante de uma demanda para outra. Isso gera um processo mais eficiente e com um custo menor.

4.6 KANBAN OU E-KANBAN

Implementar o *framework Kanban* utilizando um quadro físico e papéis é uma maneira fácil, eficiente e que requer o investimento de poucos recursos (WAN; CHEN, 2008). De acordo com Oza, Fagerholm e Munch (2013) a utilização do *kanban* tradicional, nas iterações iniciais, causa um impacto positivo na comunicação e colaboração da equipe como um todo. Após um determinado tempo, o *kanban* passa a ser uma ferramenta secundária nesse processo.

Visto as vantagens do *kanban* tradicional, é preciso ressaltar que este modelo possui algumas desvantagens que pode colocar em risco todo o processo. Wan e Chen (2008) explicita três problemas, relatados a seguir:

A utilização de papel para representar a atividade implica em um risco real de perde-los conforme o tempo, ou até mesmo por algum acidente que possa ocorrer. Gerando um custo extra ao identificar a demanda faltosa, ou até mesmo gerar um software defeituoso (DRICKHAMER, 2005) (OSBORNE, 2002);

O rastreio e o monitoramento das demandas também são prejudicados, pois não há uma maneira trivial de realiza-los utilizando o método tradicional. Ao apresentar este *kanban* à um terceiro perde-se toda a visibilidade do processo, devido a falta do rastreio (WAN, 2006) (DRICKHAMER, 2005);

A escalabilidade do *kanban* é comprometida devido a limitação física do ambiente, impedindo que ele evolua conforme o crescimento da empresa ou a evolução do processo (CUTLER, 2005).

Para sanar estes problemas diversos autores (OSBORNE, 2002) (CUTLER, 2005) (DRICKHAMER, 2005) (WAN, 2006) (WAN; CHEN, 2008) sugerem o uso de uma ferramenta eletrônica para otimizar o *Kanban*. Assim como no k*anban* tradicional, o investimento financeiro para a implementação online é zero, caso uma ferramenta gratuita seja escolhida.

O *kanban online*, ou *e-kanban*, também promove a comunicação e colaboração da equipe, em escala menor, as principais vantagens citadas por Drickhamer (2005) e Wan e Chen (2008) são justamente as fraquezas do *Kanban* Tradicional. Essas são descritas a seguir:

Elimina o problema de perda de cartões, ou acidentes que danificam os cartões, proporcionando um grau de confiabilidade maior e evitando futuros problemas com demandas perdidas;

Permite realizar o monitoramento das demandas em relação ao fluxo, assim como é possível realizar o rastreio do caminho percorrido durante todo o processo.

Por não depender de um espaço físico, o *e-kanban* não possui restrições quanto a sua escalabilidade, exceto no caso de alguma restrição imposta pela ferramenta utilizada.

Dependendo da ferramenta utilizada, o *e-kanban* permite visualizar os índices e gráficos gerados a parir das informações coletadas.

Muitas são as vantagens do *e-kanban* sobre o *kanban* tradicional, para extrair ao máximo o potencial oferecido por este tipo é necessário escolher com cautela a ferramenta que melhor se adequa a necessidade. Uma pesquisa aprofundada é necessária para analisar as ferramentas e então decidir a melhor.

Embora não seja escopo deste trabalho, apresenta-se na seção seguinte um pequeno levantamento de ferramentas que possam apoiar a adoção do processo utilizando *Kanban*

4.7 FERRAMENTAS

Neste trabalho foram levantadas as principais ferramentas gratuitas disponíveis e selecionadas três para uma análise mais detalhada. As ferramentas selecionadas foram analisadas relacionando-as as características de um quadro *Kanban* físico e o cartão de demanda.

Dadas as características quadro *kanban* físico e do cartão de demanda, analisou-se a condição da ferramenta em apoiar a adição de colunas/*swinlanes*, assim como ordená-las. O segundo quesito analisado foi a definição do limite do WIP e a sua visibilidade no quadro. E a possibilidade de incluir políticas de qualidade, interface e a disponibilização de gráficos e métricas.

Dados os quesitos, a melhor classificação de ferramenta *e-kanban* nesse processo de análise foi o Kanbanize. Essa ferramenta apresenta uma interface simples e intuitiva, exibe gráficos, permite a criação de *swinlanes*, entre outros pontos positivos. Dados da análise é apresentado na Tabela 2

Tabela 2: Análise comparativa entre ferramentas *online* para *Kanban*. Fonte: Autor

Itens de Análise	**Ferramentas**		
	Kanbanize	**KanbanFlow**	**Trello**
Métricas/Gráficos	Fluxo Cumulativo, Tempo de ciclo, Distribuição das demandas, Tempo de bloqueio e Demandas criadas vs finalizadas	Apenas na versão paga	Não
Interface	Ótima	Razoável	Razoável
É possível adicionar/renomear colunas.	Sim	Sim	Sim
É possível adicionar subcolunas.	Sim	Não	Não
É possível adicionar *swinlane.*	Sim	Apenas na versão paga	Não
É possível limitar o WIP.	Sim	Sim	Não
O WIP fica visível.	Sim	Sim	Não

Os cartões possuem os campos: Nome, descrição, tipo, *tag*, responsável e anexo.	Sim	Sim	Sim
Os cartões possuem sinalizadores de bloqueio.	Sim	Não	Não
Os cartões exibem o histórico da demanda.	Sim	Apenas na versão paga	Sim
Os cartões exibem o atual responsável.	Sim	Sim	Sim
É possível definir políticas de qualidade.	Apenas na versão paga	Apenas na versão paga	Não
Possui um arquivo para demandas antigas/encerradas.	Sim	Não	Não
Possui coluna *backlog* previamente estabelecida.	Sim	Não	Não

4.8 CONSIDERAÇÕES FINAIS

Neste capítulo apresentou-se o *Framework Kanban*. Inicialmente apresentou-se um breve histórico do *Kanban*, seguido de atividades e métricas que compõem o *framework*.

Ao analisar as atividades e princípios, observou-se a condição do *framework Kanban* em adequar-se as constantes mudanças que ocorrem durante o processo de gestão de manutenção. Por fim, apresentou-se uma comparação entre o *kanban tradicional* e o k*anban eletrônico*, assim como foi realizado uma análise de três ferramentas gratuitas para facilitar a implementação do e-kanban.

CAPÍTULO 5 – ESTUDO DE CASOS

5.1 CONSIDERAÇÕES INICIAIS

Alguns órgãos têm buscado adotar *Kanban*. Neste capítulo apresentam-se alguns empregos do *Kanban* por órgãos governamentais, como metodologia de gerenciamento de demandas de manutenção. Apresenta-se o caso do Tribunal de Contas da União (TCU) e do Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP)

5.2 TRIBUNAL DE CONTAS DA UNIÃO

O Tribunal de Contas da União é responsável por julgar os gastos dos administradores públicos e os demais envolvidos na utilização de bens e valores públicos federais, assim como qualquer pessoa que der causa a perda, extravio ou irregularidade de que resulte prejuízo ao erário. Suas funções básicas são classificadas como: Fiscalizadora, consultiva, informativa, judicante, sancionadora, corretiva, normativa e de ouvidoria. Para desempenhar esses papéis, o TCU dispõe de sistemas próprios e que necessitam de uma constante manutenção (BRASIL, 2015a).

Os sistemas de propriedade do TCU são desenvolvidos e mantidos pela Secretária de Soluções de TI (STI), e como poder ser visto na Figura 17, essa é dividida em três Diretorias de Soluções de Tecnologia da Informática (DISOL). A DISOL-2 é a responsável pelo 3º Serviço de Soluções de TI (Sesol-3), onde o estudo de caso foi realizado (BRASIL, 2015c).

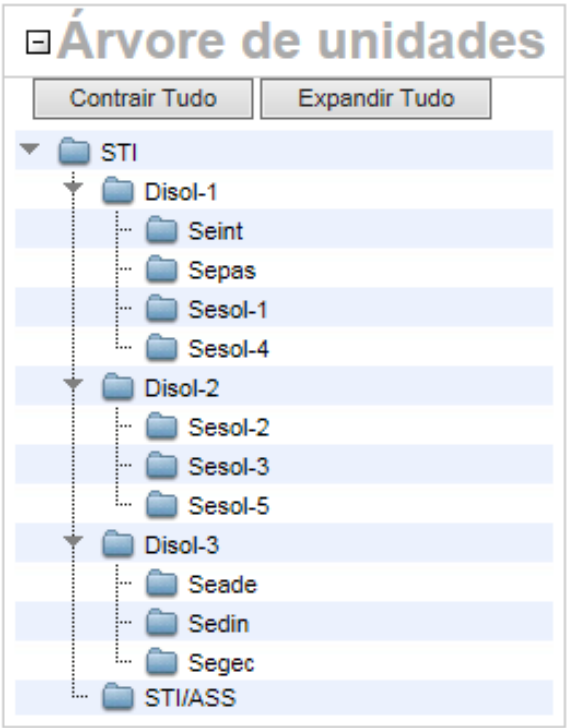

Figura 17: Árvore de Unidades da STI. Fonte: (BRASIL, 2015c).

No intuito de promover monitoria constante e melhoria em seus processos de desenvolvimento e manutenção, o Sesol-3 optou por aplicar o *framework Kanban* em ambas frentes de trabalho. Mesmo possuindo um processo estabelecido, o mesmo não era devidamente mapeado, logo a primeira etapa foi mapear as atividades. Com o processo mapeado em mãos, o chefe do setor sugeriu um WIP para cada fase e gerou-se o protótipo apresentado na Figura 18.

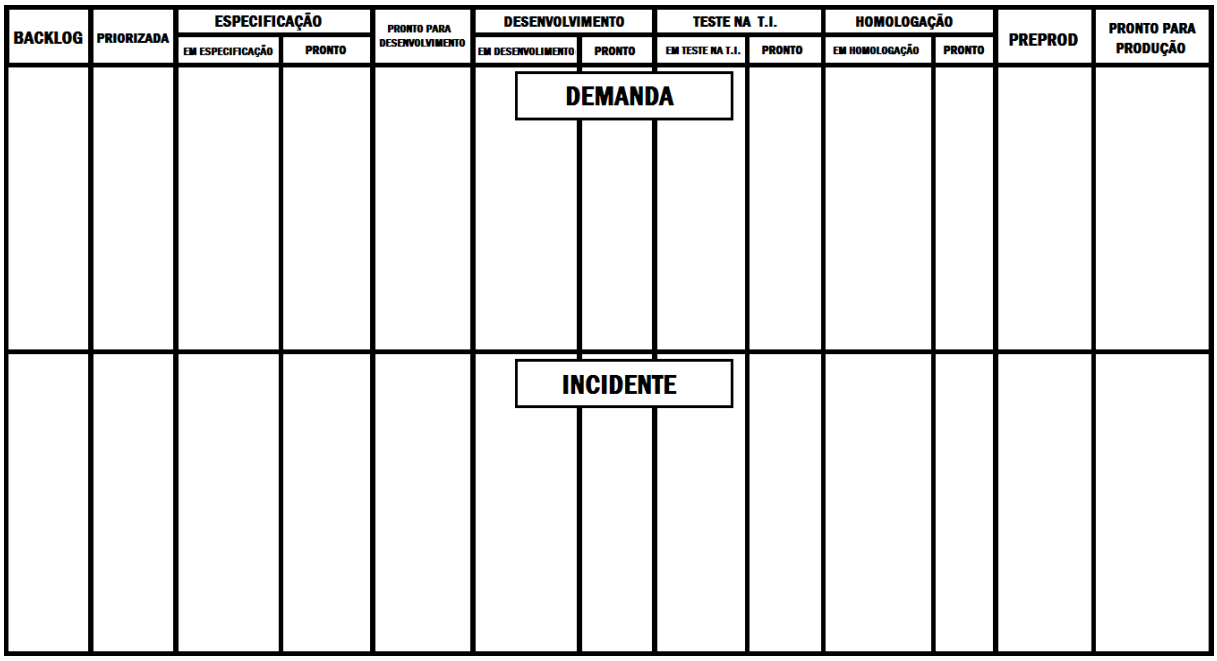

Figura 18: Protótipo do quadro *kanban*. Fonte (BRASIL, 2015e)

O processo mapeado dividi as solicitações em duas categorias, sendo elas: Demanda e Incidentes. As Demandas são solicitações oriundas de superiores, podendo ser categorizadas como manutenção evolutiva, preventiva ou adaptativa. Apesar do *framework Kanban* não fazer uso da prática de estipular iterações baseadas em período, há um acordo com unidades superiores de realizar a entrega das Demandas quinzenalmente. Os incidentes são demandas corretivas proveniente de erros encontrados pelos usuários dos sistemas. Os usuários ao encontrarem algum comportamento inesperado, abrem um chamado no *Service Desk* (SD) e então a equipe do Sesol-3 realiza a inserção no quadro kanban.

A próxima etapa realizada foi definir as políticas de qualidade, essas focaram-se no tempo de atendimento dos incidentes, no prazo de entrega das Demandas e nos testes realizados após o desenvolvimento da solicitação. As seguintes políticas foram identificadas:

- Demandas são cadastradas quinzenalmente;
- Demandas são entregues quinzenalmente;

- Os incidentes devem ser atualizados todos os dias;
- Os incidentes devem ser atendidos em no máximo cinco dias;
- Após o desenvolvimento, deve-se realizar o teste funcional (teste caixa-preta);
- A reintegração em pré-produção ocorre na segunda terça da quinzena. Sendo necessário realizar os testes de integração;
- As publicações das demandas, devidamente testadas, ocorrem na segunda sexta da quinzena.

Não é costume da equipe do Sesol-3 realizar programação orientada a teste, logo políticas relacionadas a ao desenvolvimento, propriamente dito, não foram identificadas.

Considerando algumas políticas de qualidade relativas ao tempo que as solicitações devem ser atendidas e entregues, infere-se a existência a utilização de duas das quatro classes e acordo de nível de serviço. As Demandas são claramente pertencentes a Classe de Prazo Fixo, onde as solicitações tornam-se prioritárias ao aproximar-se da data de entrega. Já os Incidentes pertencem a Classe Urgente, uma vez que quase todos, senão todos, possuem um caráter impeditivo.

A próxima etapa foi elaborar um padrão para os cartões a serem fixados no quadro *Kanban*. A equipe do Sesol-3 identificou algumas informações essenciais, sendo elas:

- Número da Solicitação no SD;
- Descrição;
- Data de Entrada;
- Ambiente de publicação;
- Nome da *Branch*;
- Sistemas alterados;
- Atual responsável pela Solicitação;
- Identificação visual para os estados de urgência e bloqueio;
- Identificação visual para sinalizar um Incidente que retornou para o desenvolvimento após ter sido encontrado erro durante o teste;

Definiu-se então que as informações relativas ao *Branch*, Ambiente de publicação e sistemas envolvidos seria inseridos em um cartão menor, a ser anexado ao cartão da demanda. A identificação de bloqueio também deveria ser realizada anexando um cartão na cor rosa/vermelho. Na Figura 19 é possível visualizar o *template* do cartão contendo as informações da Solicitação, e na Figura 20 é apresentado o modelo do cartão com as informações relativas ao *Branch*.

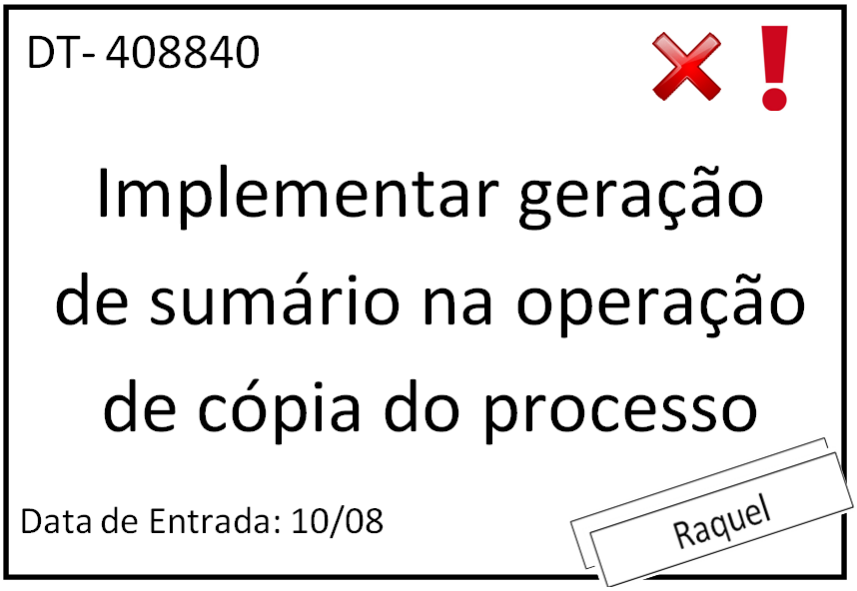

Figura 19: Modelo do cartão contendo as informações da Solicitação. Fonte (BRASIL, 2015f).

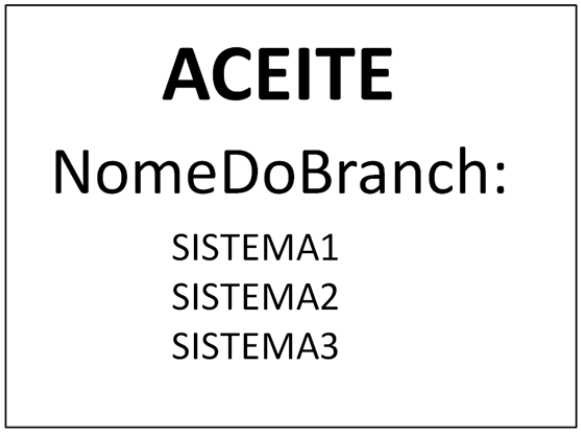

Figura 20: Modelo do cartão com as informações relativas ao Branch. Fonte (BRASIL, 2015g).

Ao fim destas etapas, obteve-se um processo baseado nos princípios do *framework Kanban* e um quadro *kanban* com o processo mapeado. A primeira implementação desse quadro foi realizada de maneira física e pode ser visualizado na Figura 21. A utilização do quando físico restringe a utilização das métricas sugeridas, logo elas não foram utilizadas inicialmente.

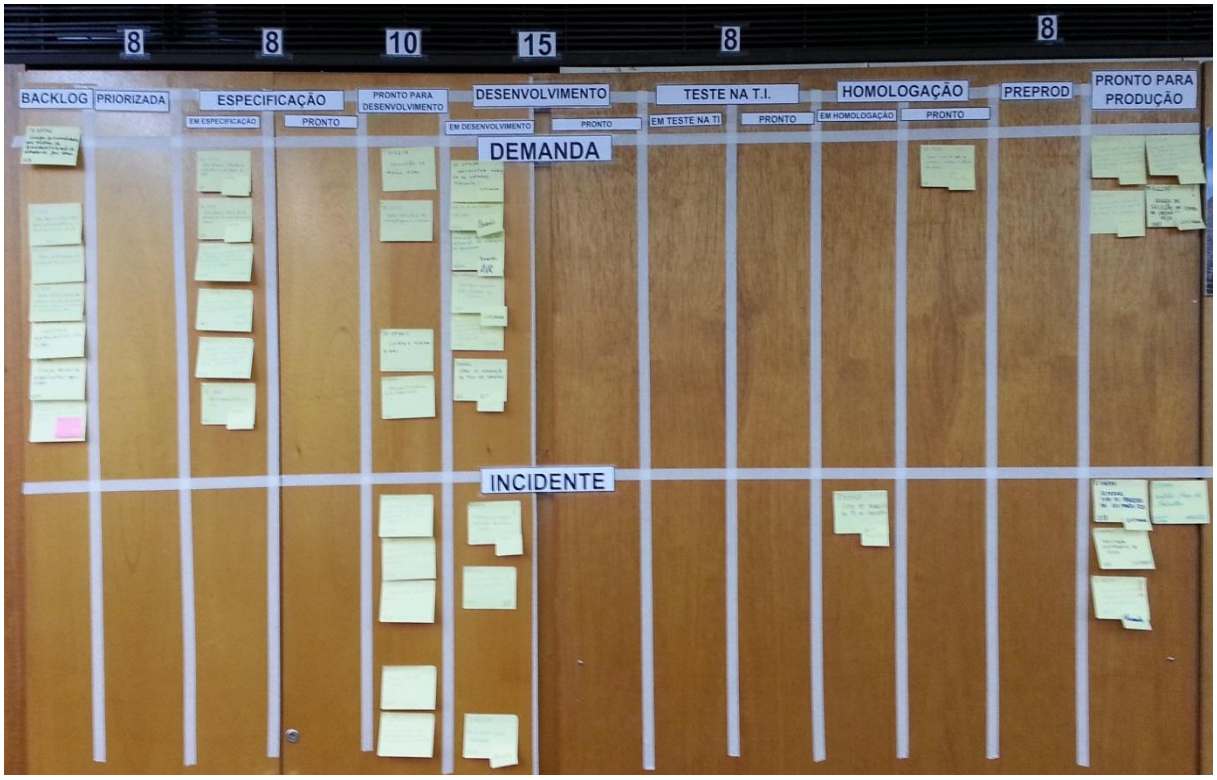

Figura 21: Primeira implementação do quadro Kanban. Fonte: (BRASIL, 2015h).

Após algumas semanas utilizando o novo processo, identificou-se a necessidade de ajustar o WIP em algumas fases do processo, assim como inserir um *buffer*. Devido a mudanças na equipe do Sesol-3, a quantidade de programadores aumentou, e o WIP da fase de "Desenvolvimento" foi alterada para 18. Também se notou que a fase de "Especificação" gerava uma quantidade maior de *outputs* do que a quantidade de *inputs* da fase de "Desenvolvimento", logo criou-se o buffer "Pronto para Desenvolvimento". Juntamente a essa segunda etapa, iniciou-se a migração para uma ferramenta *kanban* online.

Antes de realizar a migração, realizou-se uma pesquisa das principais ferramentas de *Kanban* disponíveis e suas vantagens e desvantagens. Após uma pesquisa detalhada, optou-se pela utilização da ferramenta Kanbanize. Inicialmente, manteve-se ambos os quadros, pois era necessário verificar o comportamento da equipe em relação a ferramenta e se ela realmente estava de acordo com as necessidades. Durante esse período inicial, notou-se que seriam necessários alguns ajustes, principalmente em relação as simbologias adotadas. As principais mudanças foram:

- As demandas urgentes, antes representadas com um ponto de exclamação, passaram a ser identificadas pela cor vermelha;

- A identificação visual do bloqueio utilizando um cartão rosa/vermelho, foi substituída por um símbolo de bloqueio específico da ferramenta;
- A identificação visual de demandas que retornaram para Desenvolvimento devido a erros encontrado na fase de teste, passou a ser representada pela cor laranja;
- O cartão com as informações relativas ao *Branch*, ambiente e sistemas envolvidos, foi substituído pelo uso das *tags*;
- Também se implementou o conceito de projetos, onde as cores marrom e azul são relativas a projetos específicos.
- Na Figura 22 é possível visualizar o novo modelo do cartão, e na Figura 23 é possível visualizar o quadro *kanban* elaborado na ferramenta Kanbanize.

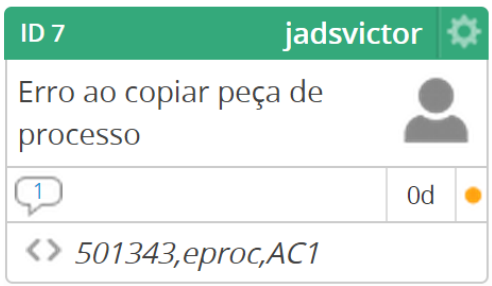

Figura 22: Modelo do cartão utilizado no Kanbanize. Fonte (BRASIL, 2015i).

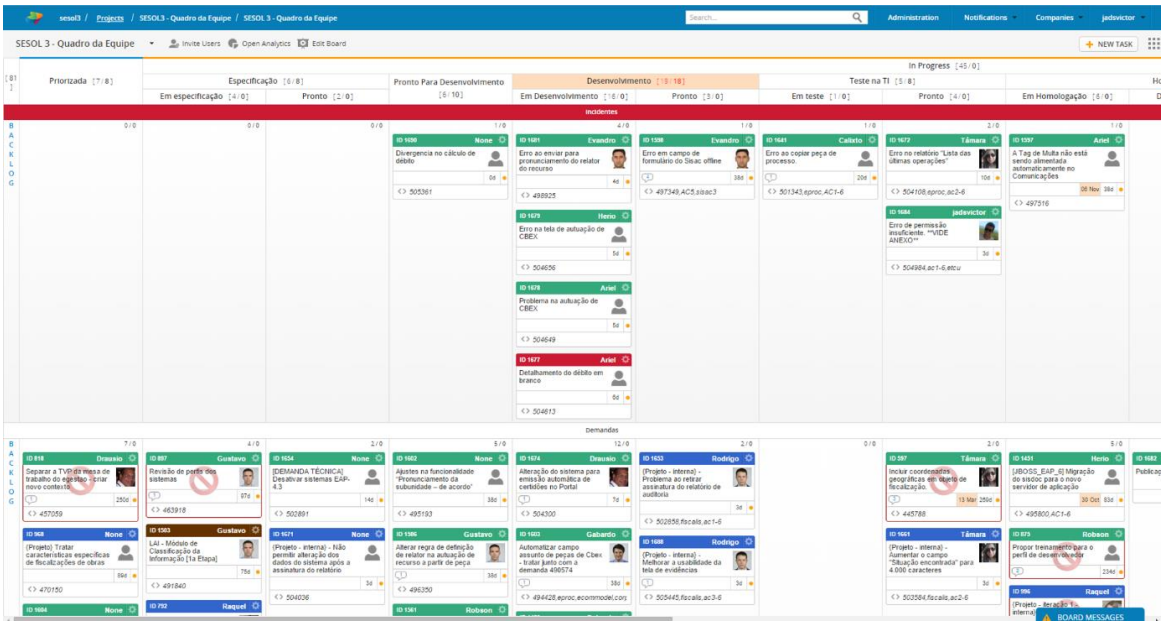

Figura 23: Parte do Quadro *Kanban* utilizando a ferramenta Kanbanize. Fonte (BRASIL, 2015j).

Com a migração para o *e-kanban*, foi possível fazer uso das métricas sugeridas. A própria ferramenta já disponibiliza gráficos de fluxo cumulativo, tempo de ciclo, distribuição das solicitações por Demanda e Incidente, tempo médio de bloqueio, entre outros. Apesar do Kanbanize ter atendido todas as necessidades do Sesol-3, ela tornou-se paga e optou-se por migrar para outra ferramenta.

Novamente realizou-se um estudo das ferramentas disponíveis e identificou-se uma capaz de substituir o Kanbanize, chamada TargetProcess (TP). O TP permitiu recriar uma estrutura de quadro *kanban* e cartões semelhantes as utilizadas, diminuindo a curva de aprendizado. A principal desvantagem identificada foi a incapacidade de agrupar colunas, forçando a ajustar o WIP. Logo as colunas possuíam subcolunas e compartilhavam do mesmo WIP foram adaptadas, um exemplo dessa adaptação pode ser visto na coluna "Desenvolvimento". Antes ela compartilhava o WIP no valor de 18, com a mudança, a sub coluna "Em Desenvolvimento" e "Pronto" tornaram-se colunas e os WIP foram alterados para 14 e 7, respectivamente. Na Figura 24 é possível visualizar o modelo do cartão utilizado no TP e na

Figura **25** é possível visualizar o quadro *kanban* reproduzido no TargetProcess

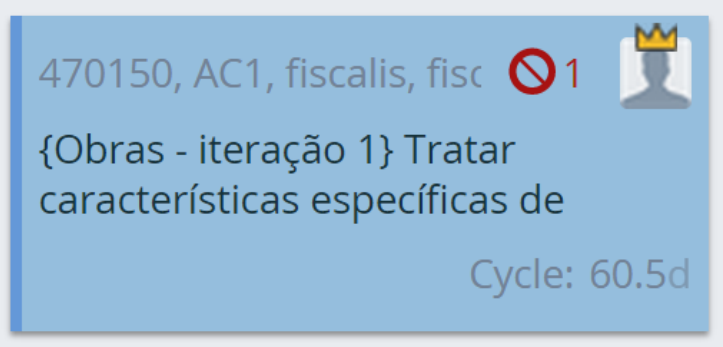

Figura 24: Modelo do cartão utilizado no TargetProcess. Fonte (BRASIL, 2015k).

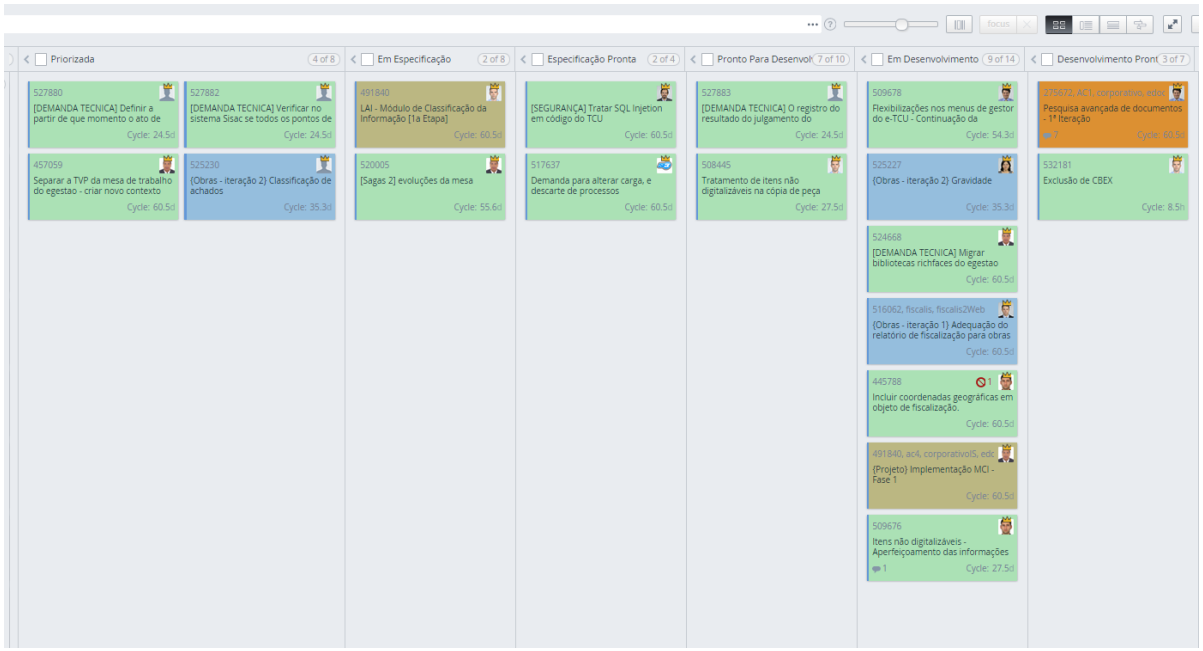

Figura 25: Parte do Quadro *kanban* utilizando a ferramenta TargetProcess. Fonte (BRASIL, 2015l).

A melhoria do processo de gestão de demandas de manutenção utilizando o *framework Kanban* permitiu visualizar que a cadência de duas semanas para a publicação das solicitações, salvo exceções, implicava negativamente no processo como um todo. Todavia, a área de TI do TCU alterou algumas normas, permitindo a publicação semanal. Essa alteração beneficiou, principalmente, o atendimento dos incidentes, uma vez que esses possuem um prazo de cinco dias para serem atendidos. Por fim, a modelagem do atual processo da Sesol-3 pode ser visualizada na Figura 26.

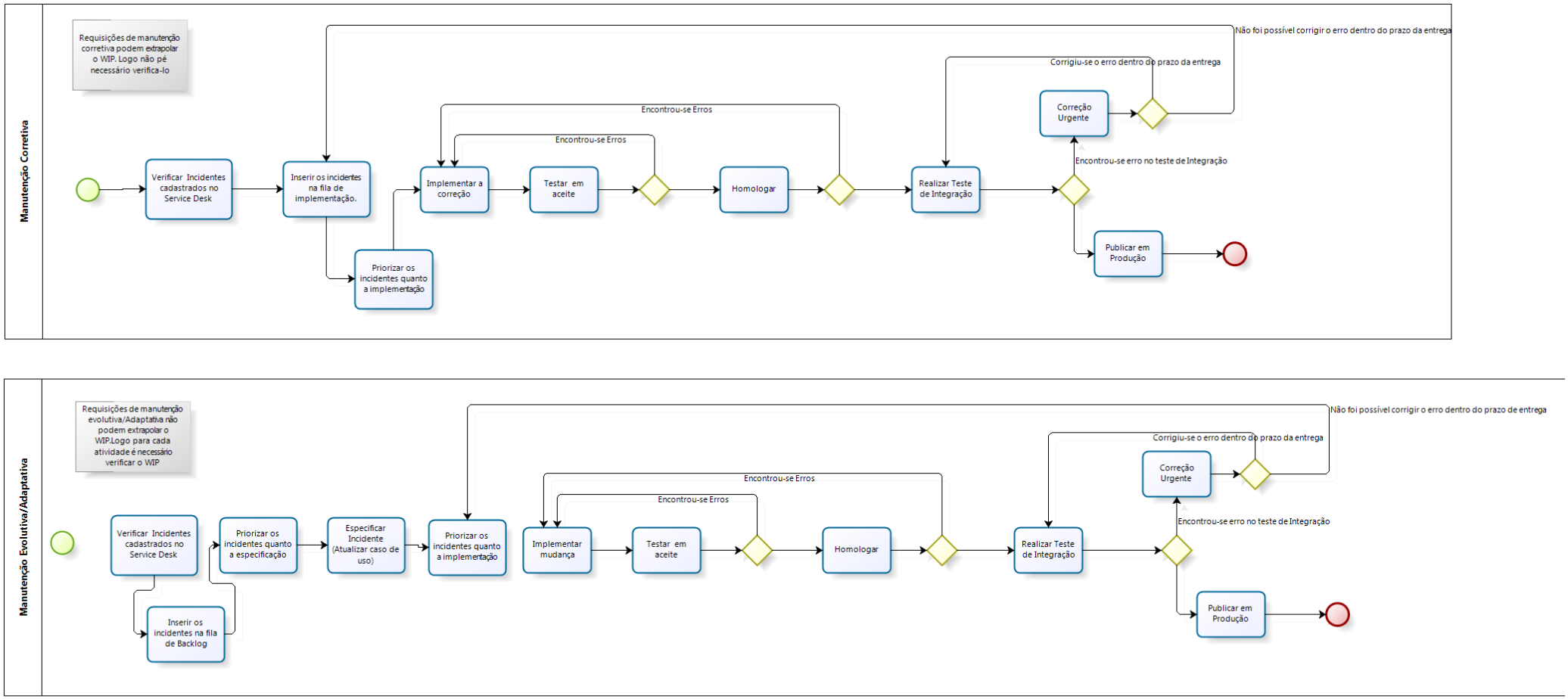

Figura 26: Modelagem do Processo de manutenção da Sesol-3. Fonte (BRASIL, 2015m)

5.3 INSTITUTO NACIONAL DE ESTUDOS E PESQUISAS EDUCACIONAIS ANÍSIO TEIXEIRA (INEP)

O Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP) foi fundado, por lei, no dia 13 de janeiro de 1937, e foi criado inicialmente para organizar a documentação relativas as doutrinas e técnicas pedagógicas, manter intercâmbio com instituições estrangeiras, realizar pesquisas, prestar assistência técnica aos serviços educacionais dos estados e municípios, assim como instituições particulares. Ao incorporar a Secretária de Avaliação e Informação Educacional, em 1997, o INEP assumiu a responsabilidade de realizar estudos, avaliações e pesquisas sobre o Sistema Educacional Brasileiro, no intuito de levantamentos estatísticos que subsidiem a formulação e implementação de políticas educacionais. Atualmente, entre os exames aplicados pelo INEP, estão o Exame Nacional de Ensino Médio (ENEM) e o Sistema Nacional de Avaliação da Educação Superior (BRASIL, 2015d).

No INEP, a diretoria responsável por fornecer, manter e organizar os serviços e recursos de TI é chamada de Diretoria de Tecnologia e Disseminação de Informações Educacionais (DTDIE). Conforme é possível visualizar na Figura 27, a DTDIE é composta por três coordenações gerais, sendo elas: Coordenação-geral de Sistemas de Informação (CGSI), Coordenação-geral de Infraestrutura e Serviços (CGIS) e a Coordenação-geral de Informação e Indicadores Educacionais (CGIIE) (BRASIL, 2013a).

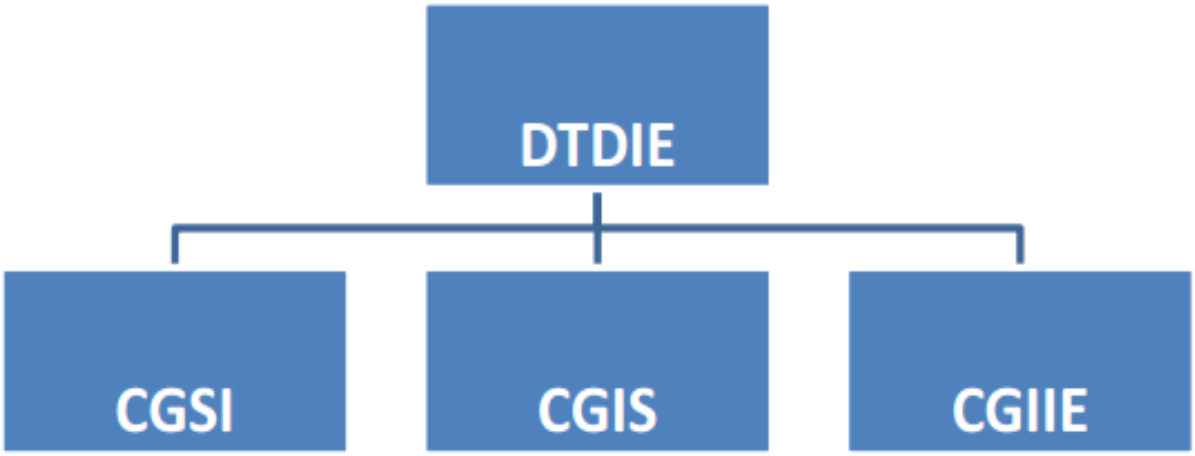

Figura 27: Estrutura Organizacional da DTDIE. Fonte: (BRASIL, 2013a)

A CGSI é a responsável por desenvolver e manter os sistemas utilizados pelo INEP, sendo este voltados para atender as necessidades dos processos de negócio da instituição, estima-se que mais de 90% dos sistemas foram desenvolvidos e são mantidos pela CGSI.

Ao se tratar do processo de manutenção, o INEP define que a empresa contratada deve utilizar o *framework Kanban*, dando continuidade aos valores ágeis já implementados pela instituição. Sendo assim, o Edital do Pregão Eletrônico nº 14/2012 elaborado pelo INEP traz uma modelagem do processo de manutenção executado, exibida na Figura 28, e faz exigências quanto as práticas do *Kanban* que devem ser adotadas (BRASIL, 2012b).

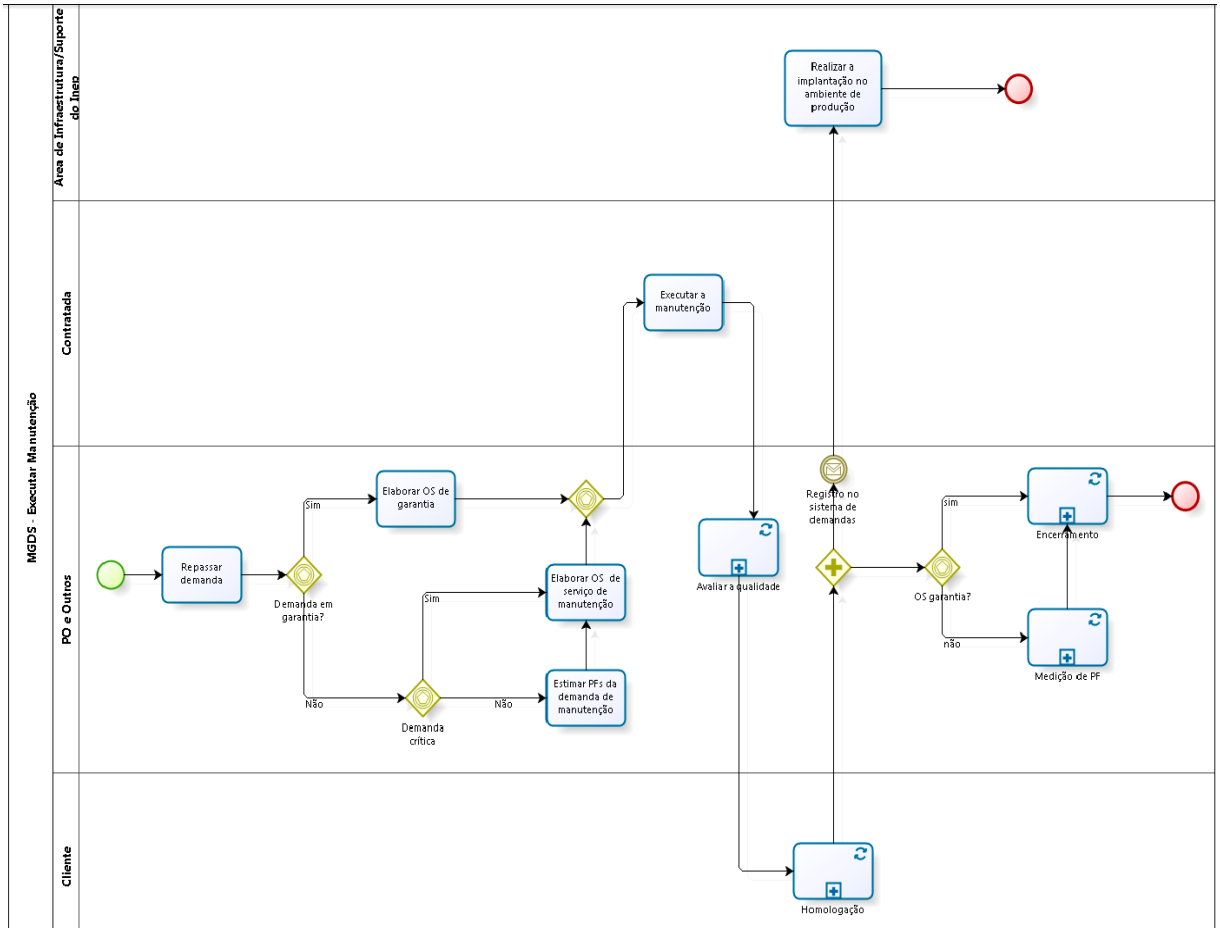

Figura 28: Processo de Manutenção INEP. Fonte (BRASIL, 2012b)

O processo de manutenção inicia-se no *Product Owner* (PO) com a atividade de repassar a demanda para analisar se esta encontra-se dentro da garantia ou não. Se estiver na garantia, elabora-se uma Ordem de Serviço (OS) de garantia e então a demanda é repassada para a contratada. Caso a demanda não tenha garantia, analisa-se a criticidade dela afim de verificar se é necessário estimar o PF e então elaborar a OS, ou se esta será elaborada sem a necessidade de verificar o PF's. Finalizado esta etapa, a demanda, também, é repassada para a contratada.

A contratada é a responsável por realizar a manutenção e, ao finalizar, retorna a demanda para o (PO) no intuito de averiguar a qualidade. Após esta etapa, o cliente homologa a demanda e informa ao PO. Este por sua vez solicita a Área de Infraestrutura/Suporte do Inep para realizar a publicação em

ambiente de produção, e paralelamente, o PO realiza o encerramento da demanda, e se necessário, realiza a medição do PF.

Com o processo devidamente mapeado, o Edital exige que as seguintes práticas sejam implementadas:

- **Visualização do Fluxo de trabalho**: Necessário dividir as demandas em tarefas, escrevendo cada uma em um cartão e fixando-o no quadro *kanban*.
- **Ilustrar o fluxo de trabalho**: Necessário utilizar colunas nomeadas que representem cada item no fluxo de trabalho
- **Limite WIP**: Necessário atribuir limites para a quantidade de tarefas que podem estar sendo desenvolvidas em cada estado do fluxo de trabalho, fornecendo previsibilidade no tempo de entrega.
- **Medir o Tempo de Ciclo**: Necessário medir o tempo médio para concluir uma atividade e otimizar o processo.
- **Otimização Contínua:** Reuniões semanais deverão ocorrer para realizar a inspeção e adaptação do processo de manutenção. As pautas destas reuniões devem incluir análise do quadro *kanban* e dos gráficos, os eventos ocorridos durante a semana e suas causas, análise do WIP estimativas de novas demandas.

Com base nas exigências acima, é sugerido um modelo de *kanban* com a estrutura mínima necessária para atender as necessidades identificadas. Como é possível visualizar na Figura 30, o exemplo apresentado separa o processo de manutenção em seis fases, sendo elas: *Backlog*, Selecionado, Desenvolver (Iniciado, Teste, Pronto), Avaliação de Qualidade, Homologação e Produção. Também se utiliza o conceito de raias para separar projetos e limites de WIP na fase Selecionado e na Desenvolver (BRASIL, 2012b). Por fim, as prioridades das tarefas são representadas pela cor do cartão. É possível visualizar a descrição de cada cor na

Figura **29**.

Figura 29: Criticidade relativa a cada cor de cartão. Fonte: (BRASIL, 2012b)

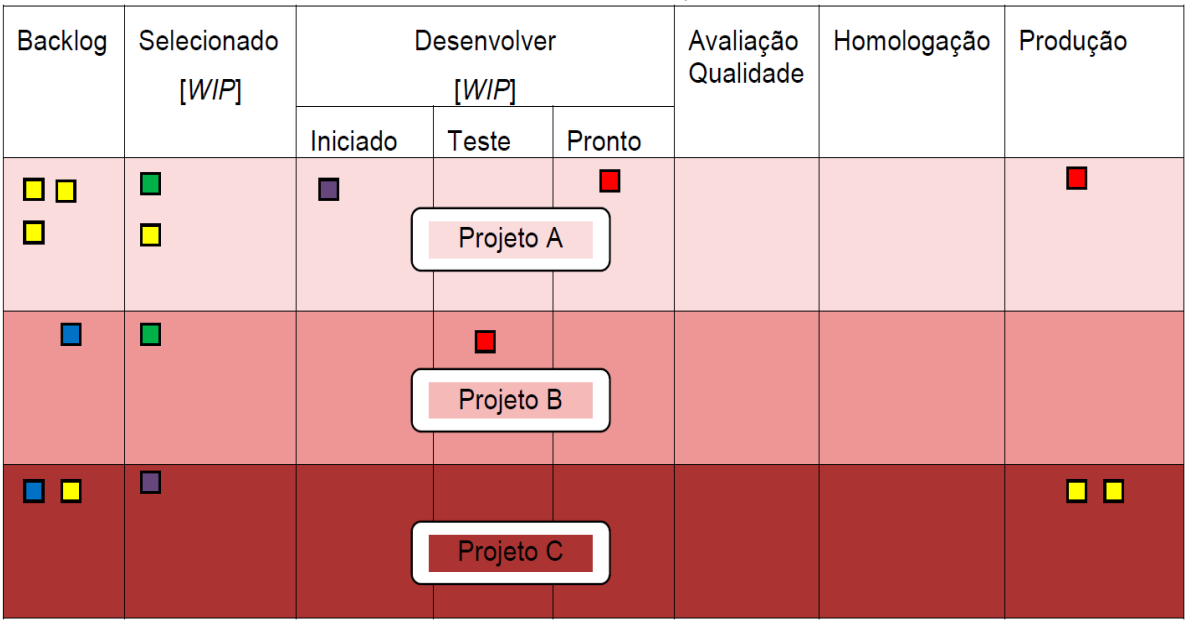

Figura 30: Modelo de quadro *kanban* proposto pelo edital. Fonte: (BRASIL, 2012b)

5.4 CONSIDERAÇÕES FINAIS

Neste capítulo, apresentou-se o estudo de caso de dois órgãos públicos que já fazem uso do *Framework Kanban* para gerenciar as demandas de manutenção de software. A análise foi essencial a este trabalho para uma maior compreensão das adaptações realizadas, além de representarem exemplos consolidados e fonte de experiências para embasar a implementação do *Kanban* no *Ministério X*

CAPÍTULO 6 – MATERIAIS E MÉTODOS

6.1 CONSIDERAÇÕES INICIAIS

Neste Capítulo apresentam-se os materiais e métodos adotados para o desenvolvimento deste trabalho. Inicialmente é realizada a caracterização do *Ministério X*, descrevendo a atual competência do órgão, assim como sua estrutura organizacional e fornecedores. Também é realizado uma análise comparativa do atual processo de gestão de demandas de manutenção com as práticas adotadas pelo *Kanban*.

6.2 CARACTERIZAÇÃO DO MINISTÉRIO

O órgão selecionado como objeto de estudo é um Ministério do Governo Federal Brasileiro responsável pelos serviços de telecomunicações, radiofusão e postais. Também lhe compete a elaboração de políticas nacionais para essas áreas. Devido a sua abrangência, o Ministério descentraliza suas funções e em sua estrutura organizacional, a Coordenação-Geral de Tecnologia e Informação (CGTI) é a responsável pela TI (BRASIL, 2014c).

A CGTI é uma unidade que responde à Subsecretária de Planejamento, Orçamento e Administração (SPOA), sendo de sua responsabilidade o desenvolvimento de planos, processos e programas referentes à TI. Logo é de sua responsabilidade o desenvolvimento de softwares de para o Ministérios, assim como fornecer manutenção adequada para os mesmos (BRASIL, 2014c). Por não se tratar de uma atividade fim do Ministério, o CGTI delega essas atividades a empresas terceirizadas, ficando responsável somente pela gestão dos contratos envolvidos. A terceirização realizada segue o processo de contratação apresentado pela IN 04/2014, apresentado na Figura 3.

De acordo com Souza (2014) atualmente os três contratos de maior influência no CGTI são referentes a Fábrica de Software, a Área de Qualidade e a Infraestrutura de TI.

- **Fábrica de Software:** Até o ano de 2014 era responsável por desenvolver e manter os sistemas desenvolvidos para o Ministério, mas atualmente é responsável apenas pela manutenção.
- **Área de Qualidade:** Responsabiliza-se por verificar e validar as entregas da Fábrica, assim como realiza a verificação da contagem dos pontos de função.
- **Infraestrutura de TI**: Responsabiliza-se por fornecer manutenção à infraestrutura de TI do Ministério.

Em um total de 61 pessoas que compõe a equipe da CGTI, 11 são servidores, 2 são administrativos e 48 são terceirizados. Com 78,69% de sua força de trabalho terceirizada, a dependência da TI em relação as empresas contratadas é uma das fraquezas levantadas no documento conjunto do Plano Estratégico de Tecnologia da Informação (PETI) e Plano Diretor de Tecnologia da Informação (PDTI) (BRASIL, 2014c).

Ao caracterizar o *Ministério X* é necessário entender que este encontra-se em um processo de evolução, e para nortear esta nova etapa elaborou-se o PETI e o PDTI com vigência de 2013 a 2015. O PETI visa contribuir com uma gestão efetiva, ética e eficiente, traçando um caminho baseado nos valores, missão e objetivos estratégicos. Já o PDTI contempla as carências de informação e serviços de TI do *Ministério X*, assim como suas metas e ações. Ao analisar este documento conjunto percebe-se facilmente menções diretas as práticas ágeis, sendo provavelmente a tendência a ser seguida pelo Ministério (BRASIL, 2014c).

Ao apresentar a missão, visão e valores do *Ministério X*, o documento aborda os conceitos de transparência, colaboração, qualidade e melhoria incremental contínua (BRASIL, 2014c). Sendo estes pilares do *framework Kanban.* O quadro *kanban* representa a transparência almejada, possibilitando visualizar o processo como um todo. Também promove a interação da equipe, onde todos colaboram para manter o quadro atualizado. A equipe também é incentivada a participar da melhoria contínua por meio de *feedbacks* e sugestões de mudanças. Por fim, a preocupação com a qualidade é sempre constante, inclusive sugere-se que as políticas de qualidade sejam explicitadas no quadro.

O documento conjunto PETI e PDTI apresenta também uma análise das Forças, Fraquezas, Oportunidades e Ameaças referentes ao CGTI. As forças definidas e que se relacionam com o *framework Kanban* são: *Equipe comprometida; Equipe aberta a mudanças de processos e práticas; Equipe conhecedora do ambiente do MC, das práticas boas e ruins implementadas no passado; Ambiente saudável e Colaborativo.* Já as fraquezas que devem ser acompanhadas cautelosamente, pois apresentam alto risco a implementação do *Kanban* são*: Quantitativo inadequado de servidores; Metodologias e processos de trabalho não totalmente definidos e/ou formalizados; Conhecimento e processos críticos concentrados na equipe dos fornecedores; Rotatividade dos envolvidos* (BRASIL, 2014c).

Com base nas Forças e fraquezas identificadas, será proposto um *framework Kanban* adaptado a realidade e necessidade do *Ministério X*. Todavia, antes de iniciar esta etapa, é necessário identificar o que já está sendo empregado no processo de gestão de manutenção de software e que contribuirá com a nova metodologia a ser proposta.

6.3 GESTÃO DE DEMANDAS DE MANUTENÇÃO DO *MINISTÉRIO X*

Ao iniciar a atividade de verificar a atual situação da gestão de demandas de manutenção, fica evidente a falta de transparência e visibilidade em todo o processo. Não há um processo de manutenção definido e o e-mail é uma das principais ferramentas entre o gestor e a fábrica, dificultando a visualização de um cenário mais abrangente. Esta carência fere um dos pilares do *Kanban*, que tanto preza pela visibilidade e mapeamento do processo, e também dificulta realizar um estudo aprofundado devido à falta de documentação. Logo, no intuito de vislumbrar o atual cenário, necessitou-se de

reuniões com os servidores do Ministério, assim como os terceirizados. Estas, principalmente com os terceirizados, foram feitas sem grande formalismo e não foram devidamente documentadas em atas.

A falta de um processo de manutenção de software devidamente mapeado, impede que o gestor tenha controle sobre o que está sendo executado e a atual situação da demanda. Consequentemente inviabiliza a utilização adequada do WIP e a limitação do trabalho em progresso. Isto afeta diretamente a essência de criar um sistema puxado, onde o gestor não necessita atribuir responsabilidades, sendo essa responsabilidade "puxada" pelos responsáveis por executar a próxima atividade da demanda.

Após algumas reuniões fica claro a dificuldade do Ministério com o processo de gestão, pois até mesmo as práticas que não são próprias do *Kanban*, tal como a priorização das demandas pelo gestor e a utilização de métricas no processo de gestão, são ignoradas. Apesar de não implementar quase todas práticas do *Kanban*, as SLA's são claras e bem definidas, pois por intermédio delas é definido o prazo em que a demanda deve ser implementada, assim como o valor que será repassado para a empresa contratada (BRASIL, 2011).

De acordo com o Edital de Pregão Eletrônico Nº 038/2011 – MC, as demandas são medidas em Ponto por Função (PF), sendo cada ponto o equivalente a 12h. Este também determina que demandas com mais de 100 PF's devem ser considerados projetos (BRASIL, 2011). As SLA's para as demandas de desenvolvimento de software, quanto para a manutenção evolutiva, são apresentadas na Tabela 3. Já as de manutenção corretiva são apresentadas na Tabela 4.

Tabela 3: SLA's para manutenção evolutiva por ponto de função. Fonte (BRASIL, 2011)

Prazos de atendimento para desenvolvimento/manutenção Evolutiva de Sistemas		
Tamanho em PF	**Prazo para iniciar atendimento**	**Prazo para Conclusão**
1 - 150	A partir da aprovação da Ordem de Serviço	45 dias
151 – 300	7 dias	90 dias
301 - 450	7 dias	135 dias
451 - 600	7 dias	180 dias
601 - 750	15 dias	220 dias
751 - 1000	15 dias	300 dias
Acima de 1000	30 dias	360 dias

Tabela 4: SLA's para manutenção corretiva de sistemas. Fonte (BRASIL, 2011)

Prazos de atendimento para manutenção corretiva de sistemas			
Criticidade	**Característica**	**Início de atendimento**	**Prazo para Conclusão**
Alta	Incidente com paralisação do sistema ou comprometimento de dados, processos ou ambientes.	Imediatamente	Em até 2 horas corridas a partir do início do atendimento
Média	Incidente sem paralisação do sistema, mas com comprometimento mediano de dados, processos ou ambientes.	Em até 2 horas corridas depois de informado o incidente a contratada	Em até 4 horas corridas a partir do início do atendimento
Baixa	Incidente sem paralisação do sistema, com pequeno ou nenhum comprometimento de dados, processos ou ambientes.	Em até 4 horas corridas depois de informado o incidente a contratada	Em até 6 horas corridas a partir do início do atendimento

6.4 MODELAGEM DO ATUAL PROCESSO DE MANUTENÇÃO

Por meio de análise documental e entrevistas informais, pode-se concluir que, atualmente, o *Ministério X* possui duas frentes de trabalho relativa a manutenção de seus sistemas, realizando manutenções corretivas ou evolutivas.

De acordo com a modelagem apresentada na Figura 31, a manutenção evolutiva pode ser dividida em quatro etapas, sendo elas: Avaliação, Desenvolvimento/Teste, Homologação e Publicação.

O processo de manutenção evolutiva, assim como a etapa de avaliação inicia-se na "Atribuição do Ticket no OTRS", com o ticket da demanda em mãos, o Analista avalia o escopo, gera um DME ou RM e atribui a responsabilidade para o gerente. Por sua vez, o gerente é o responsável por apresentar a proposta ao CAB e, se aprovada, atribui um desenvolvedor para implementa-la. Caso não seja aprovada, a demanda é encerrada.

A etapa de Desenvolvimento inicia-se com o Desenvolvedor implementando o que fora requisitado. Em seguida, a Equipe de Teste realiza os testes necessários, gerando um Registro de Teste,

e caso algum erro seja identificado, a demanda retorna ao Desenvolvedor. Se nenhum erro for encontrado, esta é encaminhada para homologação.

A primeira atividade da Homologação é publicar a demanda em um ambiente em que o usuário tenha acesso. Após a publicação, é feita uma verificação para certificar-se que o processo foi realizado com êxito. Com a publicação pronta, o Usuário é responsável por testar e homologar. Caso este encontre algum erro, é feita uma verificação para averiguar se é relacionado ao código ou não. Se for erro na codificação, a demanda retorna ao Desenvolvedor, e se não for, será feita uma análise para identificar e tomar as providências cabíveis. Com a demanda devidamente homologada, inicia-se a etapa de Publicação.

Na etapa de publicação, o Analista prepara a *build* para ser publicada em produção e faz a requisição de *deploy* para a Infra. Com a demanda publicada em ambiente de produção, encerra-se o chamado e o Processo de manutenção evolutiva

O processo de manutenção corretiva modelado, Figura 32, assemelha-se ao de manutenção evolutiva, sendo que ambas se diferem apenas no início. Ao contrário das evolutivas, as corretivas não necessitam da aprovação do CAB. Logo, após a análise, o Analista atribui a demanda diretamente para o Desenvolvedor e a continuidade do processo é idêntica ao descrito no de evolução.

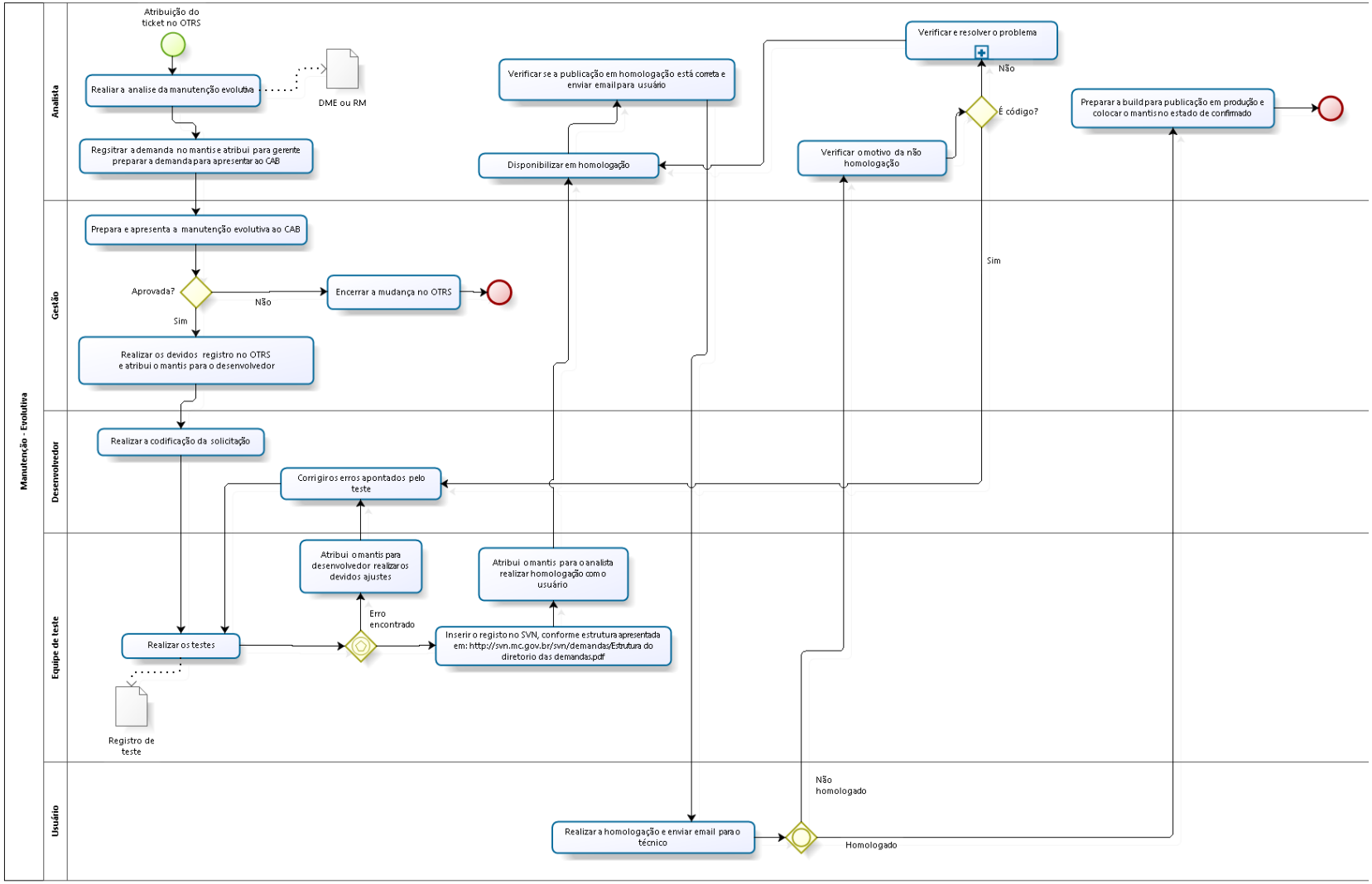

Figura 31: Modelagem do Atual Processo de Manutenção Evolutiva do *Ministério X*. Fonte: (*Ministério X*)

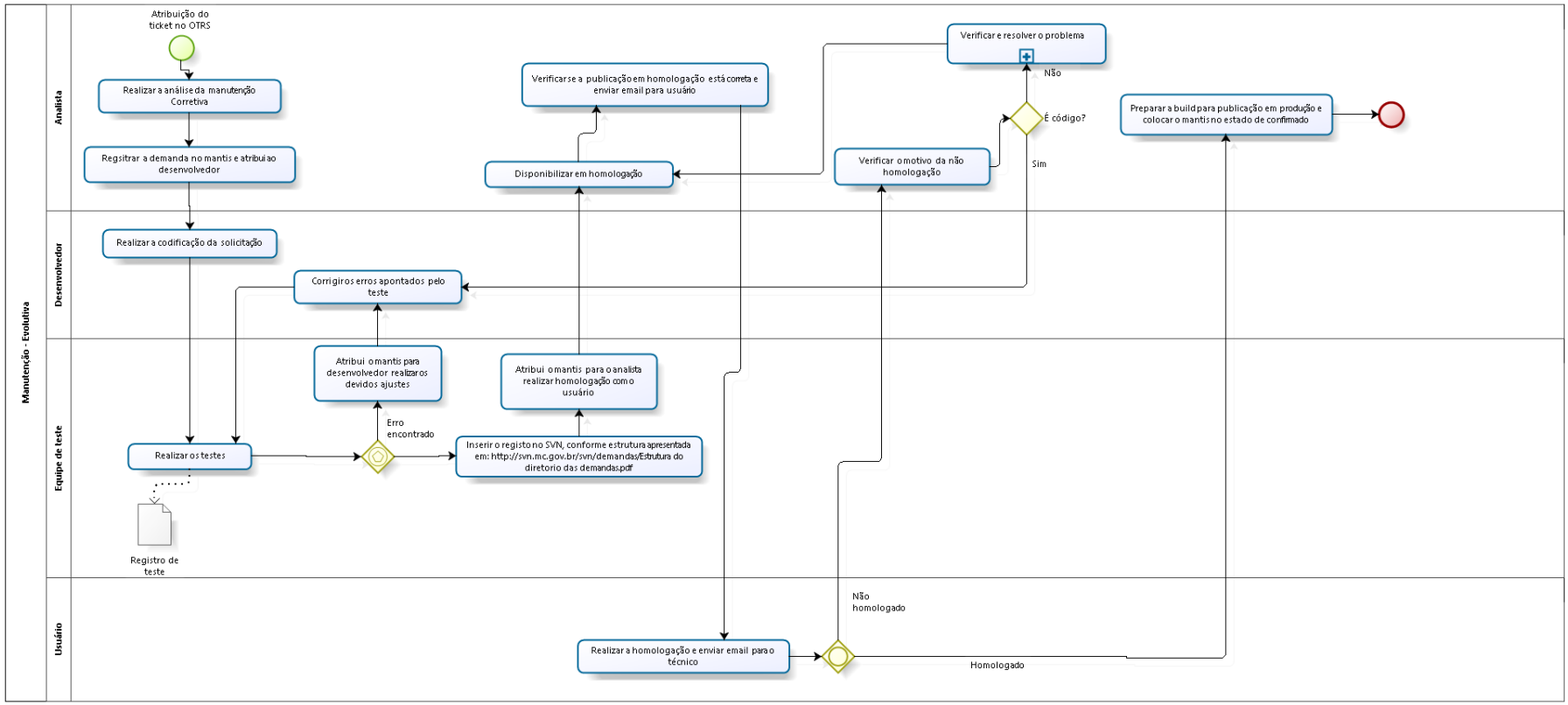

Figura 32: Modelagem do Atual Processo de Manutenção Corretiva do *Ministério X*. Fonte: (*Ministério X*)

Uma vez identificado a modelagem do atual processo de manutenção, parte da equipe que trabalha juntamente com o *Ministério X*, realizou entrevistas informais com envolvidos, afim de identificar os principais problemas encontrados durante a execução do processo. Logo no início das entrevistas, percebeu-se que ambas modelagens omitiam atividades e consequentemente não representavam a realidade do fluxo de trabalho do *Ministério X*. Sendo assim, seria necessário modelar o processo completo, afim de contemplar as atividades faltosas e identificar de maneira adequada os problemas do processo de manutenção. Erros menores também foram encontrados, tal como a nomenclatura dos processos que está repetida para ambas modelagens.

No intuito de mapear todas as atividades do processo de manutenção, parte da equipe envolvida modelou todas as atividades do processo de manutenção exercidas no *Ministério X*. Esta foi feita para agregar tanto a manutenção evolutivas, quanto a corretiva, logo omitiu-se etapa de aprovação do CAB. É possível visualizar o resultado na Figura 33.

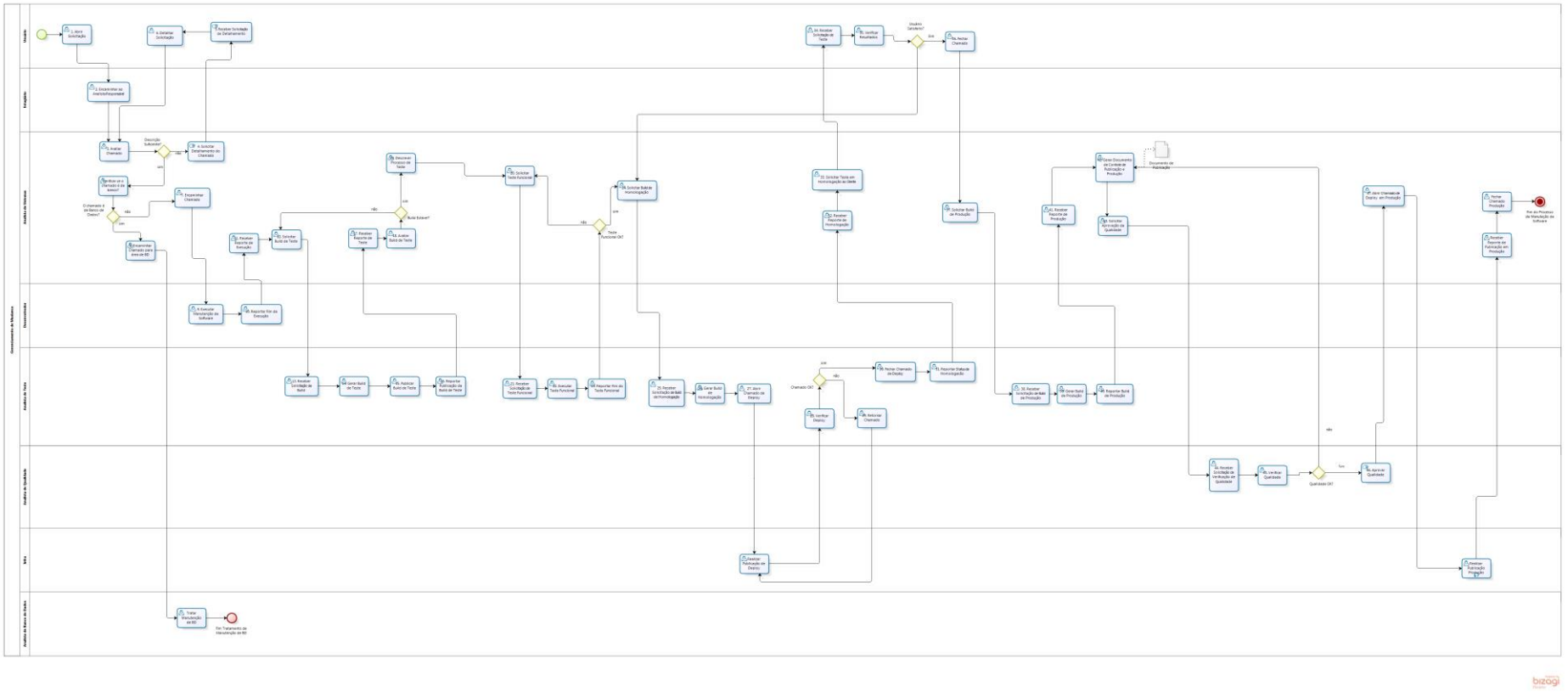

Figura 33: Mapeamento do processo de manutenção do *Ministério X* elaborado pela equipe da UnB.

As principais diferenças entre a modelagem disponibilizada pelo *Ministério X* e a realizada pela equipe da UnB são referentes à publicação das *builds* e ao processo de verificar a qualidade das demandas. Logo com essas atividades devidamente mapeadas, é possível notar o **segundo problema relatado**, que é justamente a burocratização das atividades. Logo no início do processo o Analista do Sistema necessita encaminhas o chamado para o Desenvolvedor, após a implementação, ele necessita informar ao Analista do Sistema, que por sua vez irá requisitar a *build* de teste para o Analista de Teste e então irá retornar ao Analista de Sistema que avaliará a *build* e descreverá o processo de teste. Por fim o Analista de Teste realizará o que foi especificado para testar e reportará o resultado ao Analista de sistema. Toda esta etapa poderia ser simplificada caso as atividades fossem distribuídas de maneira adequada.

O **terceiro problema relatado** é o gargalo gerado pelo teste de homologação. Por várias vezes a demanda atrasa devido à demora que o usuário, no caso o gestor do sistema apresenta para realizar o teste. Isto ocorre principalmente por dois motivos, sendo o primeiro, a falta de interesse do gestor do sistema, que nem sempre é afetado diretamente pelo problema ou pela necessidade de evolução de alguma funcionalidade. E o segundo motivo é que por diversas vezes o usuário não tem conhecimento que a demanda está pronta para ser testada

O **quarto problema identificado** foi inicialmente a não modelagem das atividades de qualidade que são exercidas pela Empresa de Qualidade. Após o devido mapeamento, percebeu-se que a contagem dos Ponto de Funções era realizada no final, e não após o término da especificação. E que a qualidade era aferida por intermédio de um documento gerado pelo Analista de Sistema.

O **quinto, e último, problema identificado** é a falta de transparência e visibilidade das demandas em execução. Isso dificulta a gestão, pois não se sabe ao certo em qual etapa do processo determinada demanda encontra-se.

Estes cinco problemas apresentados são os principais focos de melhoria para a modelagem de um processo de manutenção otimizado, e adaptado para o *Framework Kanban*.

6.5 CONSIDERAÇÕES FINAIS

O processo de gestão de demandas de manutenção do *Ministério X* possui vários aspectos a serem trabalhados e melhorados. Devido a essa ausência de atividades de gestão e da modelagem do processo, a comparação entre o que já está sendo empregado com o *Kanban*, não apresentou resultados gratificantes.

Durante a modelagem do processo por parte da equipe da UnB que compõe o projeto com o *Ministério X*, foi possível analisar cada etapa e identificar problemas e gargalos enfrentados durante a

execução do processo. Também se identificou possíveis melhorias a serem implementadas durante a modelagem do *framework Kanban.*

CAPÍTULO 7 - ADAPTAÇÕES DO FRAMEWORK KANBAN PARA UM ÓRGÃO PÚBLICO BRASILEIRO

7.1 CONSIDERAÇÕES INICIAIS

Neste capítulo apresenta-se uma proposta inicial do *Framework Kanban* voltado para o Processo de Gestão de Demandas de Manutenção de Software por Terceiros para um Órgão Público Federal Brasileiro. Esta tem como base o atual contexto do *Ministério X*, e a utilização do *Kanban* nos órgãos públicos abordados nos estudos de caso.

7.2 FRAMEWORK KANBAN ADAPTADO AO CONTEXTO DO *MINISTÉRIO X*

7.2.1 PROPOSTA DA MODELAGEM DO PROCESSO DE MANUTENÇÃO

Assim como sugerido por Boeg (2011), o primeiro passo para implementar o *Framework Kanban* é visualizar o fluxo e mapeá-lo. Nesta etapa verificou-se que as atuais modelagens do processo de manutenção não estavam de acordo com o praticado, sendo necessário modelar o processo integralmente. Após modelado, verificou-se focos de melhoria que seriam agregados ao novo modelo proposto pela equipe da UnB.

A primeira versão da modelagem do processo proposto pode ser visualizada na Figura 34 e conta com um total de cinco papéis, sendo eles:

- **Usuário**: Responsável por abrir um chamado e homologar a demanda;
- **Chefe de Sistema**: Responsável por aprovar as demandas de manutenção evolutiva e solicitar o *deploy* em Produção;
- **Desenvolvedores**: Responsável por analisar a demanda e, se necessário, especifica-la. Assim como implementa-la e testa-la. Também compete ao Desenvolvedor solicitar a homologação da demanda junto ao Usuário e gerar as *builds*;
- **Analista de Qualidade**: Responsável por verificar a qualidade da demanda implementada, assim como certificar-se da atualização dos documentos relacionados. A atividade de contagem dos pontos de função também é atribuída a este papel;
- **Infra**: A responsabilidade da Infra é publicar a *build* de Produção.

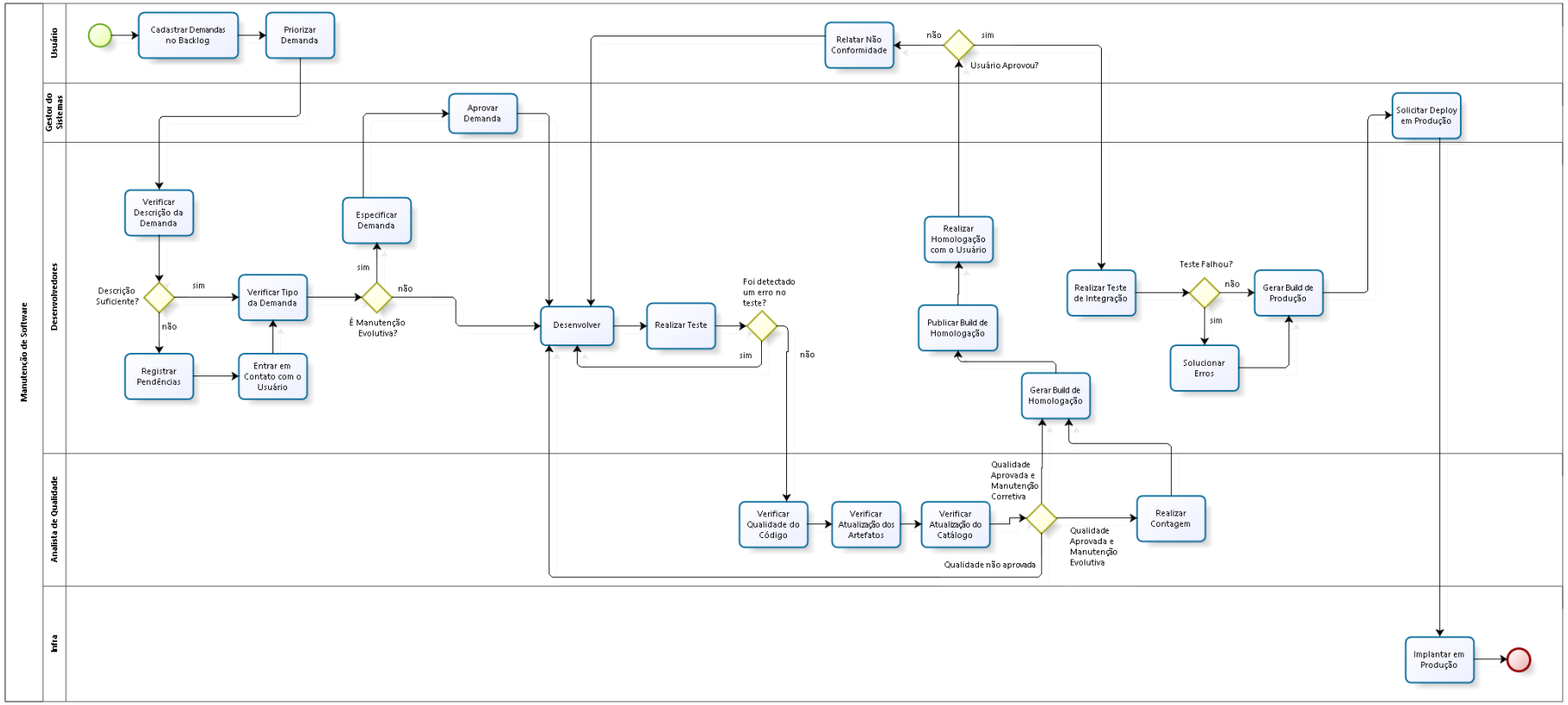

Figura 34: 1ª Versão da Modelagem do Processo de Manutenção. Fonte: Equipe do Projeto UnB e *Ministério X*

O processo de manutenção modelado é composto por um total de vinte e três atividades, sendo apresentadas na Tabela 5, conforme a ordem de execução.

Tabela 5: Relação das atividades da modelagem da primeira versão. Fonte: Autor

Atividade	**Responsável**
Cadastrar Demandas no Backlog	Usuário
Priorizar Demandas	Usuário
Verificar Descrição da Demanda	Desenvolvedor
Registrar Pendências	Desenvolvedor
Entrar em Contato com o Usuário	Desenvolvedor
Verificar Tipo da Demanda	Desenvolvedor
Especificar Demanda	Desenvolvedor
Aprovar Demanda	Chefe de Sistemas
Desenvolver	Desenvolvedor
Realizar Teste	Desenvolvedor
Verificar Qualidade do Código	Analista de Qualidade
Verificar Atualização dos Artefatos	Analista de Qualidade
Verificar Atualização do Catálogo	Analista de Qualidade
Realizar Contagem	Analista de Qualidade
Gerar *Build* de Homologação	Desenvolvedor
Publicar *Build* de Homologação	Desenvolvedor
Realizar Homologação com o Usuário	Desenvolvedor
Relatar Não Conformidade	Usuário
Realizar Teste de Integração	Desenvolvedor
Solucionar Erros	Desenvolvedor
Gerar *Build* de Produção	Desenvolvedor
Solicitar *Deploy* em Produção	Chefe de Sistemas
Implantar em Produção	Infra

Após a modelagem da primeira versão de um novo processo de manutenção, realizou-se uma reunião com os servidores do *Ministério X*, no intuito de apresentar e validar a proposta. Durante a reunião, evidenciou-se novos problemas, assim como a necessidade de mudar algumas atividades no fluxo.

O **primeiro problema** levantado na reunião e que teve impacto direto no processo é a necessidade da contagem dos pontos de função no início do fluxo. Mesmo definindo que as demandas tenham um tamanho padronizado e, preferivelmente, sejam atômicas, não há como garantir que o Usuário terá experiência para cumprir este requisito. Logo é necessário garantir que as demandas tenham um tamanho padrão e que não ultrapassem os 100 Pontos de função.

O **segundo problema** é referente à homologação do Usuário, que atualmente é considerado um gargalo e, comumente, atrasava o fluxo e a entrega da demanda para produção. A primeira solução proposta atribuía ao Desenvolvedor a responsabilidade de homologar a demanda junto com o Usuário, e mesmo sendo bem aceita pela equipe do *Ministério X*, esta sugeriu algumas mudanças que colaborarão com a agilidade dessa atividade. Ao incorporar as mudanças propostas ao processo, o Usuário deixará de homologar demandas individuais e irá homologar um conjunto de demandas já integradas, e caso algum erro seja encontrado, a demanda em específica retorna para correção dos erros, e as demais continuam o fluxo.

A **terceira modificação** realizada é antecipar no fluxo a realização dos testes de integração. Antes este teste era realizado no fim do processo de manutenção, mas devido a solução adotada para o segundo problema levantado durante a reunião, o teste de integração será realizado antes da homologação do Usuário. Ao mesmo tempo se levantou a necessidade de estipular uma data limite dentro da janela de entrega para aguardar as demandas, integra-las e dar prosseguimento com o teste. Essa data limite foi idealizada para garantir que as demais atividades do processo não sejam prejudicadas com a falta de tempo.

A **realização do teste de integração** antes da homologação também reduz os riscos da carência de um processo de integração contínua, pois como foi dito durante a reunião, já aconteceu de um desenvolvedor alterar uma funcionalidade e, no mesmo período, um outro desenvolvedor atendeu uma demanda que requeria a remoção da funcionalidade alterada. Devido ao fato da remoção ter sido solicitada depois, o correto seria entregar o sistema sem a funcionalidade, mas devido à falta de integração contínua, foram produzidas duas versões diferentes do código e a entregou-se aquela com a funcionalidade alterada. Logo, apesar de não

ser o objetivo do teste de integração, este erro seria facilmente detectado e corrigido antes de requisitar a homologação do usuário.

No intuito de solucionar os problemas levantados e contemplar as requisições feitas, criou-se uma segunda versão do processo de manutenção, que pode ser visualizado na Figura 35. Em relação à versão anterior nenhuma atividade foi retirada e os papéis permaneceram iguais. Somente a ordem de execução das atividades “Realizar Contagem”, “Realizar Teste de Integração” e “Solucionar erros” foram alteradas. Na Tabela 6 são apresentadas as atividades da segunda versão do processo, conforme sua ordem de execução. E em seguida apresenta-se o detalhamento das vinte e três atividades que compõem a nova versão do processo de manutenção.

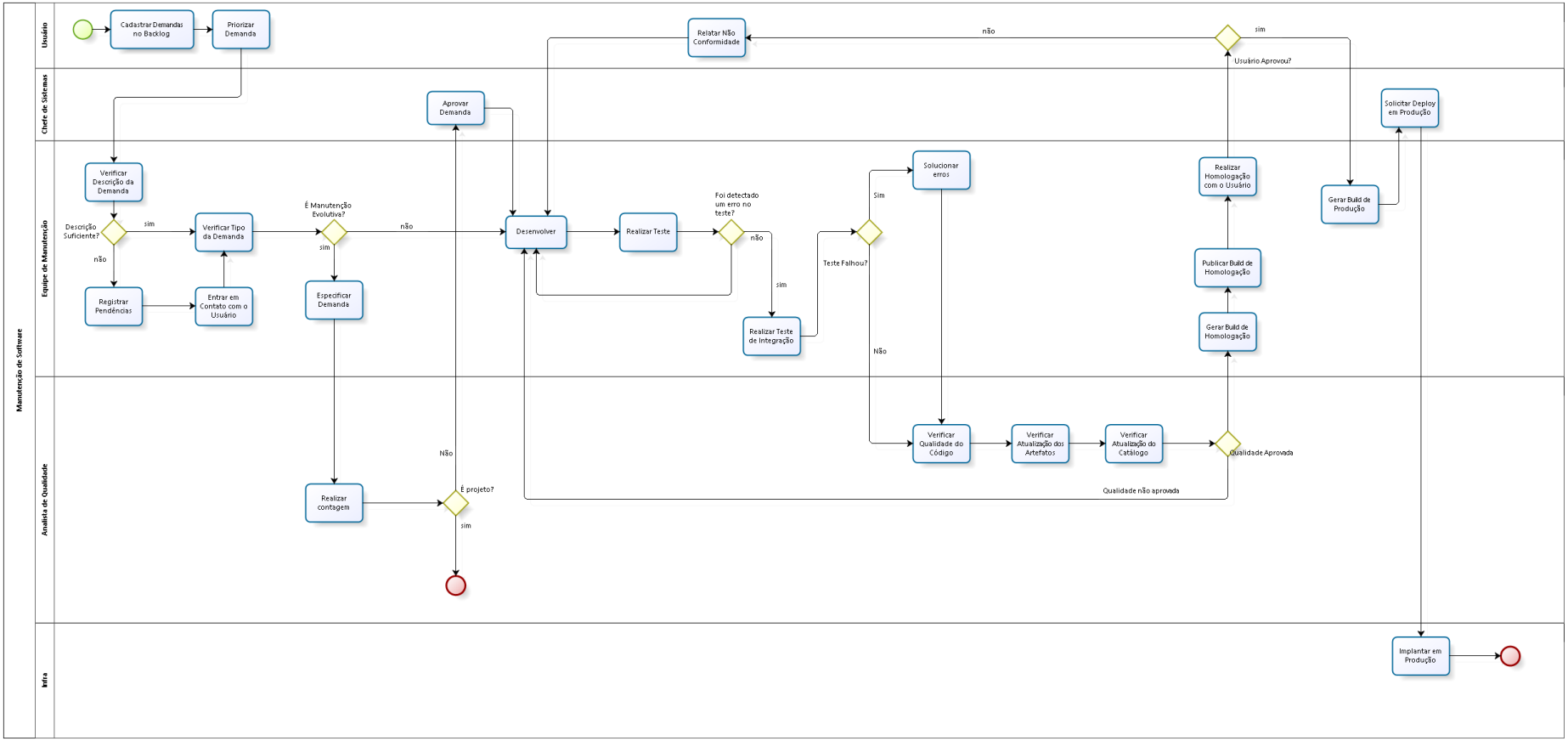

Figura 35: 2ª Versão da Modelagem do Processo de Manutenção. Fonte: Equipe do Projeto UnB e *Ministério X*

Tabela 6: Relação das atividades da modelagem da segunda versão. Fonte: Autor

Atividade	**Responsável**
Cadastrar Demandas no *Backlog*	Usuário
Priorizar Demandas	Usuário
Verificar Descrição da Demanda	Desenvolvedor
Registrar Pendências	Desenvolvedor
Entrar em Contato com o Usuário	Desenvolvedor
Verificar Tipo da Demanda	Desenvolvedor
Especificar Demanda	Desenvolvedor
Realizar Contagem	Analista de Qualidade
Aprovar Demanda	Chefe de Sistemas
Desenvolver	Desenvolvedor
Realizar Teste	Desenvolvedor
Realizar Teste de Integração	Desenvolvedor
Solucionar Erros	Desenvolvedor
Verificar Qualidade do Código	Analista de Qualidade
Verificar Atualização dos Artefatos	Analista de Qualidade
Verificar Atualização do Catálogo	Analista de Qualidade
Gerar *Build* de Homologação	Desenvolvedor
Publicar *Build* de Homologação	Desenvolvedor
Realizar Homologação com o Usuário	Desenvolvedor
Relatar Não Conformidade	Usuário
Gerar *Build* de Produção	Desenvolvedor
Solicitar *Deploy* em Produção	Chefe de Sistemas
Implantar em Produção	Infra

- **Cadastrar Demandas no *Backlog*:** O Usuário cadastra as demandas no *backlog* do quadro *kanban*, sendo que estas devem, preferivelmente, ser atômicas. O fato do usuário ser o responsável, visa aproxima-lo do processo como um todo;
- **Priorizar Demandas:** Após cadastrar as demandas, o Usuário deverá prioriza-las conforme as suas necessidades. Esta atividade também tem o intuito de sanar o problema da atual falta de priorização e é apoiada por Pigoski (1997);
- **Verificar Descrição da Demanda:** Após a priorização, o Desenvolvedor deve verificar se as informações relatadas pelo Usuário são suficientes para realizar a implementação. Se a descrição estiver adequada, o processo segue para a atividade "Verificar Tipo de Demanda". Mas se a descrição não for suficiente, será necessário complementa-la, e o processo segue para a atividade "Registrar Pendências";
- **Registrar Pendências:** Após verificar que a as informações fornecidas pelo Usuário não são suficientes, o Desenvolvedor registra as informações pendentes. A importância desse registro deve-se ao fato de, se necessário, permitir que outro Desenvolvedor possa dar continuidade ao atendimento da demanda;
- **Entrar em Contato com o Usuário:** Após registrar as pendências, o Desenvolvedor deve entrar em contato com o Usuário e obter as informações necessárias para dar continuidade com a demanda. É importante que este contato seja feito presencialmente ou por telefone, pois agiliza o processo e é possível auxiliar o Usuário a conceder as informações com maior exatidão;
- **Verificar Tipo da Demanda:** Com a descrição da demanda pronta, o Desenvolvedor deve verificar se a demanda de manutenção é evolutiva ou corretiva. Se for evolutiva o processo segue para a atividade "Especificar Demanda". Caso seja corretiva, a demanda possui um caráter de urgência e o processo segue para o desenvolvimento, descartando a necessidade de especificar a demanda e aguardar a aprovação pelo Chefe de Sistemas;
- **Especificar Demanda:** No caso das demandas evolutivas, após verificar o tipo da demanda é necessário especifica-las adequadamente, gerando as documentações necessárias e fornecendo insumo para os Desenvolvedores implementar a demanda adequadamente;

- **Realizar Contagem:** No caso das demandas evolutivas, o Analista de Qualidade realiza a contagem dos pontos de função da demanda no intuito de, ao fim do processo, realizar o pagamento para a Fábrica de Software, uma vez que esta é a responsável pelos desenvolvedores. O edital do *Ministério X* estabelece que demandas com mais de 100 PF's devam ser tratadas como projetos.

- **Aprovar Demanda:** Após a demanda de manutenção do tipo evolutiva ter sido especificada e com a contagem dos pontos de função finalizada, é necessário a aprovação do Chefe de Sistemas para dar continuidade com o seu desenvolvimento. Se a demanda for aprovada, ela é encaminhada para o desenvolvimento. Caso seja recusada, o fechasse o chamado e o processo é encerrado;

- **Desenvolver**: A demanda é implementada pelo Desenvolvedor e então deverá ser testada;

- **Realizar Teste:** Após o desenvolvimento da demanda, é necessário testa-la para certificar-se que a implementação está correta. Caso algum erro seja encontrado, a demanda retorna para o desenvolvimento e deverá ser testada novamente. Metodologias como testes unitários e testes funcionais devem ser adotadas de acordo com a necessidade;

- **Realizar Teste de Integração:** Com a demanda devidamente homologada, o processo segue para os testes de integração, onde o Desenvolvedor verifica-se a integração da demanda com os outros sistemas e modificações realizadas nesse período. Por mais que a demanda tenha sido devidamente testada, este teste visa encontrar possíveis erros oriundos da integração, sendo que estes não poderiam ser identificados nos testes realizados anteriormente.

- **Solucionar Erros:** Caso o teste de integração encontre algum erro, o Desenvolvedor deve soluciona-lo antes de gerar a *build* de homologação. É necessário que as correções sejam realizadas rapidamente, logo a atual atividade foi criada para evitar que a demanda retorne para a atividade "Desenvolver".

- **Verificar Qualidade do Código:** A verificação da qualidade é realizada pelo Analista de Qualidade e é composta por três atividades. Caso algum erro seja encontrado, é necessário finaliza-las antes de reportar os erros. A primeira atividade de qualidade é avaliar o código gerado. De acordo com April e Alan (2008) um dos problemas da

manutenção de software é a má elaboração do software, logo é necessário certificar-se que o código entregue atende aos parâmetros de qualidade. A verificação pode ser realizada com análise estática de código e outras metodologias;

- **Verificar Atualização dos Artefatos:** Após a verificação da qualidade do código, é necessário averiguar se os artefatos relativos à demanda foram devidamente atualizados. Pois de acordo com Briand (2003) e Huang e Tilley (2003), normalmente os sistemas não possuem sua documentação atualizada.
- **Verificar Atualização do Catálogo:** A última atividade, em relação à demanda, é verificar se o catalogo foi devidamente atualizado. Este é parte essencial da documentação relativa à demanda. Após finalizada as três atividades, caso algum erro tenha sido encontrado, a demanda retorna para a atividade "Desenvolver". Com a qualidade for aprovada, a próxima atividade depende do tipo da demanda. Se for manutenção corretiva, o fluxo segue para a atividade "Gerar Build de Homologação". Mas se for evolutiva, segue para "Realizar Contagem".
- **Gerar *Build* de Homologação:** Após a demanda ter sido devidamente testada e com o aval do Analista de Qualidade, o Desenvolvedor gera a *build* de homologação. Esta será disponibilizada para a homologação do Usuário**;**
- **Publicar *Build* de Homologação:** Com a *build* de homologação devidamente gerada, é necessário publica-la em um ambiente que permita o Usuário realizar a homologação.
- **Realizar Homologação com o Usuário:** A homologação pelo usuário foi um dos principais problemas levantados. Normalmente o Usuário demorava para homologar a demanda, gerando um gargalo e atrasando o fluxo. A solução adotada é tornar o Desenvolvedor como responsável por essa atividade, logo ele terá que entrar em contato com o cliente e solicitar uma reunião para que ambos realizem essa atividade. Espera-se que com a participação ativa do Desenvolvedor, o tempo gasto nessa atividade seja reduzido significativamente.
- **Relatar Não Conformidade:** Se durante a homologação, o Usuário encontrar algum erro, ele necessita relatar a não conformidade e o fluxo retorna para o desenvolvimento. Não há um fluxo alternativo retornando para análise da demanda,

pois espera-se que as atividades de especificação garantam que erros relativos ao escopo da demanda sejam mínimos.

- **Gerar *Build* de Produção:** Após realizar o teste de integração, e corrigir possíveis erros, o Desenvolvedor deve gerar a *build* que de produção. Esta será utilizada para a publicação das demandas em ambiente de produção**.**
- **Solicitar *Deploy* em Produção:** Com a *build* de produção pronta, o Chefe de Sistemas solicita o *deploy* em produção para a Infra.
- **Implantar em Produção:** Após a solicitação do *deploy*, a Infra publica a *build* em produção.

Com a modelagem do processo definida, de acordo com o Boeg (2011), o próximo passo é criar o quadro *kanban*

7.2.2 PROPOSTA DO QUADRO KANBAN

O Quadro *kanban* é o elemento responsável por proporcionar parte da visibilidade almejada pelo *Framework*. Nele são expostas as fases do processo, assim como os limites WIP, as políticas de qualidade, responsáveis pelas demandas e as próprias demandas.

Ao representar o processo no quadro *kanban*, Wingfield (2012) faz dois alertas quanto a representação do processo mapeado. O primeiro é para não representar o processo detalhadamente, pois pode tornar o Quadro complicado e confundir a equipe, principalmente se a rotatividade for alta. O segundo alerta é analisar a cadeia de valor do processo, evitando que atividades, que não agregam valor, sejam representadas. Quando este evento ocorre, as demandas costumam não parar nestas atividades. Ambos os alertas visam proteger o Quadro *kanban* de uma complexidade desnecessária, acarretando em uma possível rejeição por parte da equipe. Com base nessas informações, percebeu-se a necessidade de adaptar o fluxo mapeado, agrupando as atividades em fases maiores.

Considerando o alerta feito, foi necessário agrupar as atividades em fases maiores e somente então colocá-las no Quadro. Como é possível visualizar na Figura 36, as atividades foram divididas em nove fases, sendo elas:

- ***Backlog*:** Esta coluna representa a atividade " Cadastrar Demandas no *Backlog*" e é primeira coluna do Quadro *kanban*. Sua função é alocar as demandas abertas pelos Usuários que ainda não foram priorizadas adequadamente.
- **Priorizada:** Esta coluna representa a atividade "Priorizar Demandas". O Usuário é responsável por priorizar as demandas contidas no *Backlog*. Logo é necessário que ele mova

as demandas para a coluna “Priorizada” e ordene as demandas de acordo com a prioridade definida.

- **Especificação:** Esta fase foi idealizada no intuito de englobar as atividades “Verificar Descrição da Demanda”, “Registrar Pendências”, “Entrar em Contato com o Usuário”, “Verificar Tipo da Demanda” e “Especificar Demanda”. Estas atividades foram agrupadas devido ao fato de possivelmente serem executadas pela mesma pessoa, não sendo necessário sinalizar o término da atividade. Dividida em duas subcolunas, a demanda só deve ser movida de “Em Espec. ” para “Pronto” ao finalizar as atividades citadas acima.
- **Contagem PF:** Esta coluna representa a atividade “Realizar Contagem” e é dividida em duas subcolunas, sendo que a demanda só poderá ir para “Pronto” quando a contagem estiver finalizada.
- **Aprovação:** Esta coluna representa a atividade “Aprovar Demanda”, também dividida em duas subcolunas, a demanda apenas poderá ir para “Pronto” quando a demanda for considerada viável ou inviável, independente do resultado.
- **Desenvolvimento:** Esta coluna representa as atividades “Desenvolvimento”, “Realizar Teste”, “Realizar Teste de Integração” e “Solucionar Erros”. Também é dividida em duas subcolunas, e a demanda só deve ser movida de “Em Desenv.” para “Pronto” quando o Desenvolvedor finalizar a implementação e concluir os testes..
- **Qualidade:** Esta fase foi idealizada no intuito de englobar as atividades “Verificar Qualidade do Código”, “Verificar Atualização dos Artefatos” e “Verificar Atualização do Catálogo”. Assim como as colunas “Especificação” e “Teste”, agrupou-se as atividades para minimizar o esforço de atualizar o Quadro *kanban*. Também é dividida em duas subcolunas, e a demanda só deve ser movida de “Em Aval.” para “Pronto” quando o Analista de Qualidade não encontrar erros nas três atividades de qualidade.
- **Homologação:** Esta fase foi idealizada no intuito de englobar as atividades “Gerar *Build* de Homologação”, “Publicar *Build* de Homologação”, “Realizar Homologação com o Usuário” e “Relatar Não Conformidade”. Assim como as demais colunas que representam atividades agrupadas, o motivo para a junção permanece inalterado. Também é dividida em duas subcolunas, e a demanda só deve ser movida de “Em Homol.” para “Pronto” quando o Usuário aprovar.
- **Aguardando Publicação:** Após o Usuário aprovar, a demanda permanece nesta coluna, enquanto não é publicada em ambiente de produção.
- **Done:** Após a demanda ter sido concluída e publicada em ambiente de produção, ela deve ser movida para esta coluna.

BACKLOG	PRIORIZADA	ESPECIFICAÇÃO		CONTAGEM PF		APROVAÇÃO		DESENVOLVIMENTO		QUALIDADE		HOMOLOGAÇÃO		AGUARDANDO PUBLICAÇÃO	ENTREGUE
		EM ESPEC.	PRONTO	EM CONT	PRONTO	EM APROV	PRONTO	EM DESENV.	PRONTO	EM AVAL.	PRONTO	EM HOMOL.	PRONTO		

Figura 36: Proposta do quadro *kanban:* Fonte: Autor

Assim como a modelagem do processo de manutenção, o Quadro *kanban* é uma proposta que necessita ser homologada, e se necessário alterada. O fato de serem propostas não devia afetar diretamente na definição dos demais parâmetros, todavia o projeto de implantação do *Framework* ainda se encontra imaturo e novas reuniões são necessárias para dar continuidade de forma adequada. Em conjunto a imaturidade, há a necessidade de oficializar o novo edital do *Ministério X*, que ainda se encontra no jurídico, não estando disponível ao público. Devido este cenário, o limite WIP, as políticas de qualidade, os tipos de classes e suas políticas de serviços fazem parte de um trabalho futuro que dará seguimento a este.

7.3 BENEFÍCIOS DO EMPREGO DO KANBAN NO *MINISTÉRIO* X

Ao aplicar as práticas do *Framework Kanban* agrega-se diversos benefícios ao processo de gestão de manutenção de software. A primeira prática a ser analisado é a **Visualização do Fluxo,** esta permite que a equipe do *Ministério* X conheça o processo de desenvolvimento das demandas de manutenção e identifiquem possíveis focos de melhorias no processo. Conhecendo o processo e visualizando constantemente é possível lidar com as mudanças, tanto no processo, quanto nas demandas, da melhor forma possível. A segunda prática a ser analisada quanto aos benefícios atrelados a sua implementação é a **Reprodução do Fluxo no Quadro *kanban***.

Após identificar o processo do *Ministério* X, e ter analisado a cadeia de valor, então é possível criar um quadro *kanban* que permita visualizar as principais etapas do processo apeado. Explicitar o processo, permite que a equipe do *Ministério* X possa rapidamente visualizar e entender o que está acontecendo dentro do órgão. Ao implanta-lo é possível visualizar as demandas que estão dentro do fluxo, assim como os responsáveis atuais e os próximos responsáveis. Também incentiva a auto-organização da equipe, fazendo com que os integrantes assumam uma postura de maior responsabilidade. O quadro *kanban* também é composto pelo WIP e as Políticas de Qualidade.

Para o *Ministério* X, a importância do **WIP** ajustado e condizente com a real capacidade se faz importante, pois ele determina a capacidade do *Ministério* em atender as demandas solicitada. Com base nessa capacidade, será realizada a **Priorização das Demandas** evolutivas, assim como servir de base para a aprovação das demandas que necessitem passar por este filtro. O WIP também colabora com o a previsibilidade do processo, pois ele é um dos principais responsáveis por manter o fluxo contínuo.

As **Políticas de Qualidade** permitem que o *Ministério* X especifique as condições necessárias para que as empresas terceirizadas tenham conhecimento do que é preciso para a demanda continuar seu fluxo. Pois a demanda apenas poderá sair da coluna quando todas as políticas estabelecidas forem cumpridas. Esta prática também agrega qualidade ao produto que está sendo desenvolvido.

As **SLAs** permitem que o *Ministério* X estabeleça a melhor maneira de lidar com demandas de determinados tipo. Devido ao fato do *Ministério* trabalhar com empresas terceirizadas, este já faz uso desta prática, utilizando-a no acompanhamento das obrigações contratuais em relação a entrega das demandas é realizado por intermédio destas.

Estas práticas são acompanhadas por um conjunto de **Métricas**, que permitem visualizar diversos aspectos do processo. Inicialmente tem-se o **Fluxo Cumulativo,** nele é possível visualizar a quantidade de trabalho realizado em cada etapa do processo. Permitindo analisar se o sistema está funcionando corretamente. A segunda métrica é o **Tempo de Ciclo,** esta permite acompanhar o tempo de cada demanda dentro do processo, permitindo verificar se as SLAs estão sendo seguidas. Também é possível obter o tempo médio das demandas, permitindo uma previsibilidade de entrega das demandas futuras. O **Índice de Defeitos** permite verificar a relação entre o total de defeitos encontrados nas demandas entregues e a relação entre novos defeitos e defeitos já identificados. Com estas informações é possível verificar se a qualidade das demandas entregue estão consistente com o esperado, e se necessário também é possível aplicar multas, caso seja comprovado que a falta de qualidade é devido a inflação de algum item contratual. A **Quantidade de Itens Bloqueados** permite visualizar a média de itens nessa situação e fornece as principais informações para identificar a causa dos bloqueios. Por fim, outras métricas podem ser utilizadas, mas estas são as principais adotadas (BOEG, 2011).

Finalizando as práticas adotadas pelo *Framework Kanban*, a **Otimização Contínua** permite que o *Ministério* X análise constantemente as informações obtidas por intermédio de feedbacks e das métricas aplicadas, e realize ajustes sempre que necessário. Esta prática deve ter um caráter ativo, evitando que mudanças necessárias sejam adiadas.

CAPÍTULO 8 – CONCLUSÕES E TRABALHOS FUTUROS

O *framework Kanban* vem sendo empregado para auxiliar o processo de gestão, possibilitando otimizar o fluxo de trabalho e permitindo determinar iterações, sem o limite de tempo, o que possibilita alterações nas demandas de acordo com a aceitação da instituição. Logo neste trabalho apresentou-se a proposta de apoiar a *definição de um processo de gestão de demandas de manutenção de software por terceiros para um órgão do governo federal brasileiro, utilizando o Framework Kanban*. Ao fim, foi possível iniciar a construção do framework Kanban, modelando o atual processo do *Ministério*, levantando os principais problemas e propondo um novo processo. Também foi possível modelar um quadro *kanban* compatível com a real necessidade do órgão.

Para alcançar estes resultados se fez necessário um levantamento das empresas e, principalmente Instituições Públicas, que utilizam ou já utilizaram o *Kanban*. Os exemplos encontrados na literatura apresentaram resultados otimistas quanto ao uso deste *framework*, assim como os estudos realizados no TCU e INEP. Durante os estudos realizados, ficou claro que uma das principais necessidades é conhecer bem o contexto no qual se deseja implementar o *Kanban*. Logo se fez necessário caracterizar o *Ministério* X antes de começar, de fato, com o novo *framework*.

Durante a caracterização percebeu-se a falta de familiaridade da equipe com a metodologia, o que acabou dificultando o avanço do projeto. Em conjunto com esta dificuldade, aspectos inerentes ao atual processo de trabalho do órgão também não colaboraram com o desenvolvimento, tais como a falta de visibilidade, a burocratização das atividades, a dificuldade em lidar com empresas terceirizadas e até mesmo a falta de integração contínua. Todavia, como é possível ver no desenvolvimento deste trabalho, todos os problemas levantados foram tratados, no intuito de sana-los ou minimizar as consequências.

Apesar dos problemas acima, tanto a equipe da UnB, quanto a do *Ministério*, mostrou-se sempre disposta a ajudar e fornecer o máximo de informações possíveis, auxiliando na superação destes obstáculos. Mesmo assim, estes fatores atrasaram o desenvolvimento do *framework* e impediram que as demais características do *Kanban* fossem adaptadas ao contexto.

Entende-se, que este trabalho faz parte de um trabalho maior, no qual encontra-se uma equipe de professores e estudantes da Frente de Melhoria de Processos de Software (Framework de Soluções de TI, oriundo de um Termo de Cooperação entre a Universidade de Brasília e um *Ministério* do Governo Federal, denominado *Ministério X*). Nesta Frente já foi definido, para o *Ministério X*, um processo de *Gestão de Demandas de Desenvolvimento Ágil de Software* (GeDDAS), baseado em princípios ágeis, empregando o *framework Scrum*. Atualmente, o projeto de *Melhoria de Processos de Software* está no segundo ano de execução, realizando uma pesquisa explicativa, em que estão em execução 02 projetos piloto empregando o GeDDAS.

Assim como o GeDDAS, espera-se que com o êxito deste trabalho, o *Ministério X* dê continuidade na implementação do *Kanban* e que futuramente seja possível homologar as propostas de

processo de manutenção e quadro *kanban* propostos, assim como definir os demais elementos inerentes ao Framework, tais como o WIP e as Políticas de Qualidade. Também se espera que seja realizado uma pesquisa explicativa, com a execução de projetos piloto para avaliação e refinamento do *processo definido* junto ao *Ministério X*.

REFERÊNCIAS BIBLIOGRÁFICAS

ÅGERFALK, P.J. *ET AL*. **A framework for considering opportunities and threats in distributed software development**, Proceedings of International Workshop on Distributed Software Development, Austrian Computer Society, 2005, 47–61.

AMBLER, S.W. **Agile Modeling: Effective Practices for XP and RUP.** New York: John Wiley & Sons.2002

AMMANN, M. H.; CAMERON, R. D**. Measuring Program Structure With Inter-Module Metrics**, Computer Software and Applications Conference, 1994. Compsac 94. Proceedings., Eighteenth Annual International, pp. 139-144, 1994.

AMORIM, H. S. **Terceirização no serviço público: uma análise à luz da nova hermenêutica constitucional**. São Paulo: LTr, 2009.

ANDERSON, D. **Kanban - Successful Evolutionary Change for your Technology Business.** Blue Hole Press. ISBN 0-9845214-0-2, 2010

ANDRÉ, M. E. D. A. **Estudo de Caso em Pesquisa e Avaliação Educacional**. Brasília: Liber Livro Editora, 2005.

APRIL, A.; ABRAN, A. **Software Maintenance Management: Evaluation and Continuous Improvement.** Wiley, 2012.

AZIZ, J.; AHMED, F.; LAGHARI, M. S. **Empirical analysis of team and application size on software maintenance and support activities.** 1st IEEE International Conference on Information Management and Engineering (ICIME), Washington, EUA: IEEE Computer Society, p 47-51, 2009

BASILI, V. *ET AL***. Understanding and Predicting the Process of Software Maintenance Release**. Proceedings of the 18th International Conference on Software Engineering. IEEE Computer Society, p. 464–474, 1996.

Beck, K. **Extreme Programming Explained**: Embrace Change. Addison-Wesley, 2000

BOEG, J. ***Kanban Em 10 Passos***. Leonardo Galvão, n.d.

BOEHM, B, et al. **Quantitative Evaluation of Software Quality**, P. 605, 1976.

BRASIL, **Decreto-Lei Nº200**, 1967. Disponível em: <http://www.planalto.gov.br/ccivil_03/decreto-lei/del0200.htm >

_____, **Edital de Pregão Eletrônico Nº 038/2011-MC**, 2011. Disponível em: <http://www.mc.gov.br/o-ministerio/172-editais-e-avisos/23912-edital-de-pregao-eletronico-no-0382011-mc>

_____, **Guia de Boas Práticas em Contratação de Soluções de Tecnologia da Informação**, 2012a. Disponível em: <http://www.governoeletronico.gov.br/biblioteca/arquivos/guia-de-boas-praticas-em-contratacao-de-solucoes-de-tecnologia-da-informacao-tcu>

_____, **Edital de Pregão Eletrônico Nº 014/2012**, 2012b. Disponível em: < http://download.inep.gov.br/gestao_inep/aquisicoes/licitacoes/2012/pregao_elet_n_14_2012_fabrica_software_selo_cac.pdf>

_____, **Plano Diretor de Tecnologia da Informação-PDTI**, 2013a. Disponível em: <http://download.inep.gov.br/gestao_inep/gestao_ti/PDTI/2013/plano_diretor_tecnologia_informacao_2013_2015.pdf>

_____, Secretaria de Logística e Tecnologia da Informação. **Guia Prático para Contratação de Soluções de TI**, 2014b. Disponível em: <http://www.governoeletronico.gov.br/sisp-conteudo/nucleo-de-contratacoes-de-ti/modelo-de-contratacoes-normativos-e-documentos-de-referencia/guia-de-boas-praticas-em-contratacao-de-solucoes-de-ti>

_____, Secretaria de Logística e Tecnologia da Informação. **Instrução Normativa N° 4, de 11 de setembro de 2014. Dispõe sobre o processo de contratação de Soluções de Tecnologia da Informação pelos órgãos integrantes do Sistema de Administração de Recursos de Tecnologia da Informação e Informática (SISP) do Poder Executivo Federal**, 2014a. Disponível em: <http://www.governoeletronico.gov.br/biblioteca/arquivos/instrucao-normativa-nb0-4-de-11-de-setembro-de-2014-compilada/download>

_____, **Plano Estratégico de Tecnologia da Informação (PETI) e Plano Diretor de Tecnologia da Informação (PDTI) 2013-2015**, 2014c. Disponível em: http://www.mc.gov.br/index.php

_____, Tribunal de Contas da União. **Breve Histórico**, 2015a. Disponível em:

<http://portal2.tcu.gov.br/portal/page/portal/TCU/institucional/conheca_tcu/historia>

_____, _____. Sefti. **História**, 2015b. Disponível em: <http://portal2.tcu.gov.br/portal/page/portal/TCU/comunidades/tecnologia_informacao/sefti/sobre_unidade/historia>

_____, _____. **Indicador Eletrônico**, 2015c. Disponível em:

<http://contas.tcu.gov.br/pls/apex/f?p=9682:11:1328095315269101::NO:11:P11_UNIDADE:190432>

BRASIL, Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira. **História**, 2015d. Disponível em: < http://portal.inep.gov.br/institucional-historia >

BRASIL, Tribunal de Contas da União. Sesol-3, 2015e. Disponível em <https://contas.tcu.gov.br/wikiti/images/5/58/Quadro_modelo.png>

_____, _____. _____, 2015f. Disponível em <https://contas.tcu.gov.br/wikiti/images/f/f8/Cartao_modelo1.png>

_____, _____. _____, 2015g. Disponível em <https://contas.tcu.gov.br/wikiti/images/0/06/Cartao_modelo2.png>

_____, _____. _____, 2015h. Disponível em <https://contas.tcu.gov.br/wikiti/images/c/cf/Qadrokanban_fixo.jpg>

_____, _____. _____, 2015i. Disponível em <https://contas.tcu.gov.br/wikiti/images/d/d9/Cartao_kanbanize.png>

_____, _____. _____, 2015j. Disponível em <https://contas.tcu.gov.br/wikiti/images/5/59/Quadro_kanbanize.png>

_____, _____. _____, 2015k. Disponível em <https://contas.tcu.gov.br/wikiti/images/3/3d/CartaoTP.png>

_____, _____. _____, 2015l. Disponível em <https://contas.tcu.gov.br/wikiti/images/6/60/Quadro_tp.png>

_____, _____. _____, 2015m. Disponível em <https://contas.tcu.gov.br/wikiti/images/5/58/Modelagem_manutencao.png>

BRIAND, L.C. Software documentation: How much is enough. In Proceedings of the Seventh European Conference on Software Maintenance and engineering (CSMR'03), pages 13–17. IEEE, IEEE Comp. Soc. Press, March 26 - 28 2003.

CAMPOS, L. A. **Terceirização de serviços públicos**. Disponível em: <http://www.boletimjuridico.com.br/doutrina/texto.asp?id=1470>

CANFORA, G.; CIMITILE, A. **Software maintenance,** Handbook of Software Engineering and Knowledge Engineering, vol. 1. Fundamentals. Chang SK (ed.). World Scientific Publishing Co. Pte. Ltd.: Singapore, 2001; 91–120.

CAO, L.; RAMESH, B. **Agile requirements engineering practices: An empirical study**. *IEEE Software* 2008; **25**(1):60–67.

CHAN, T.; HO, T. **An Economic Model to Estimate Software Rewriting and Replacement Times**. IEEE Transaction on Software Engineering, p. 580-598, 1996.

CHENG T.;JANSEN S, REMMERS M. **Controlling and monitoring agile software development in three Dutch product software companies.** *Proceedings of the 2009 ICSE Workshop on Software Development Governance* (17–17 May 2009). *International Conference on Software Engineering*. IEEE Computer Society: Silver Spring MD, 2009; 29–35.

COHN, M. **Succeeding with agile: software development using Scrum**. Upper Saddle River, NJ: Addison-Wesley, 2010.

COHN, M; FORD, D. **Introducing an Agile process to an organization**. *Computer* 2003; **36**(6):74–78.

CONCAS, G.; LUNESU, M. I.; MARCHES M.;ZHANG, H. **Simulation of software maintenance process, with and without a work-in-process limit**, J. Softw. Evol. and Proc., vol. 25, no. 12, p. 1225–1248, Dec. 2013.

CUTLER, T. R. **Dis"card"ing the paper kanban: electronic web-based kanbans support lean manufacturing.(Special Report).** InMFG. Advantage Business Media. 2005.

DEKLEVA, S. **Delphi study of software maintenance problems.** In proceedings of the IEEE Conference on Software Maintenance. IEEE Computer Society Orlando, p.10-17, 1992

DINGSØYR, T.; DYBÅ, T.; ABRAHAMSSON, P. **A Preliminary Roadmap for Empirical Research on Agile Software Development.** IEEE, 2008

DRICKHAMER, D. **The Kanban e-volution.** Material Handling Management 60. P. 24-26. 2005

E. HILL, *et al.* **Exploring the Neighborhood with Dora to Expedite Software Maintenance**. Twentysecond Ieee/Acm International Conference on Automated Software Engineering, Atlanta, Georgia, Usa, 2007.

EBNER, G.; KAINDL, H. **Tracing All Around in Reengineering**. IEEE Software, p. 70-76, 2012

FEATHERS, M**. BEFORE CLARITY [SOFTWARE DESIGN]**, SOFTWARE, IEEE, VOL. 21, PP. 86-88, 2004.

FORWARD A.; LETHBRIDGE, T.C. **The relevance of software documentation, tools and technologies: a survey**. DocEng ’02: Proceedings of the 2002 ACM symposium on Document engineering, pages 26–33, New York, NY, USA, 2002. ACM Press.

FORWARD, A. **Software Documentation – Building and Maintaning Artefacts of Comunication**. Tese de Mestrado, Universidade de Ottawa, Otawwa, Toronto, Canadá, 2002

GIL, A. C. **Como elaborar projetos de pesquisa**. São Paulo: Atlas, 2008.

GLASS, R. L. **Building Quality Software**. Prentice-Hall, Englewood Cliffs, NJ, 19921.

GRAVES, K. 2011. **Taming the Customer Support Queue**. Agile 2011 Conference. IEEE Comput. Soc., Salt Lake City, UT, p. 154-160.

GROSS, J. M.; MCINNIS, K. R. **Kanban Made Simple : Demystifying and Applying Toyota’s Legendary Manufacturing Process**. New York, NY, USA: Amacom, 2003.

GRUBB, P.; TAKANG, A. A. **Software Maintenance: Concepts and Practice**. World Scientific, 2003.

H. MOHAMED. **Facilitating Tacit-Knowledge Acquisition Within Requirements Engineering**. 10th Wseas International Conference on Applied Computer Science (Acs '10*)*, Iwate Prefectural University, Japan, 2010.

HATTON, L. **How Accurately Do Engineers Predict Software Maintenance Tasks?** Computer , vol.40, no.2, pp.64,69, 2007.

HIGHSMITH, J. **Agile Software Development Ecosystems**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002

HIGHSMITH, J. **Adaptive software development: a collaborative approach to managing complex systems.** Addison-Wesley, 2013.

HUANG S.; TILLEY, S. **Towards a documentation maturity model**. SIGDOC '03: Proceedings of the 21st annual international conference on Documentation, pages 93–99, New York, NY, USA, 2003. ACM Press.

HUANG, S.; TILLEY, S. **Towards a Documentation Maturity Model**. Proceedings of the 21st Annual International Conference on Design of Communication, p. 93-99, 2003.

IEEE Software Engineering Standard, IEEE Std 729-1993, 1993.

IEEE Standard for Software Maintenance, IEEE Std 1219-1998, 1998

ISO/IEC 12207. **Information Technology. Software life cycle processes,** 1995.

ISO/IEC. **International Standard ISO/IEC/IEEE 14764 Software Engineering - Software Life Cycle Processes - Maintenance.** Piscataway, EUA, 2006.
ISO/IEC**. International Standard ISO/IEC 12207 Software Life Cycle Processes.** Genebra, Suíca, 2008

K. K. AGGARWAL, *et al.* **An Integrated Measure of Software Maintainability**. Reliability And Maintainability Symposium, 2002.

Kremer, W. **Birth–death and bug counting**, IEEE Transactions on Reliability 32(1), 37–47, 1983.

LIENTZ, B. P. AND SWANSON, E. B., **Software Maintenance Management**, Addison Wesley, Reading MA, 1980

LIENTZ, B.P.; SWANSON, E.B. **Problems in Application Software Maintenance.** Communications of ACM, c.24, n.11, p.31-7, 1981.

LIU, K.; ALDERSON, A.T; QURESHI, Z .**Requirements Recovery from Legacy Systems by**

Analysing and Modelling Behavior. Proceedings of the International Conference on Software Maintenance, IEEE Computer Society. Los Alamitos, p.3-12,1999

LIVERMORE J.A. **Factors that impact implementing an agile software development methodology.** Southeast Conference, 2007. *Proceedings*. IEEE Computer Society: Silver Spring MD, 2007; 82–86.

M. KERNAHAN, *et al*. **Extracting Traceability Information From C# Projects**, Wseas International Conference on Engineering Education, Athens, Greece, 2005.

MAASSEN, O.; SONNEVELT J. **Kanban at an Insurance Company (Are You Sure?)**. In Agile Processes in Software Engineering and Extreme Programming, 48:297–306. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2010.

MAMONE, S. **The IEEE Standard for Software Maintenance**. Software Engineering Notes, v.19, n.1, 1994.

MENS, T. et al. **Challenges in software evolution, Principles of Software Evolution**, Eighth International Workshop, pp.13,22, 2005

MIDHA, V., BHATTACHERJEE, **A Governance practices and software maintenance: A study of open source projects.** Decision Support Systems, vol. 54, no. 1, p. 23–32, 2012.

OHBA, M.; CHOU, X.M. **Does imperfect debugging affect software reliability growth?** Proceedings of the 11th International Conference on Software Engineering, pp. 237–244, 1989.

OHNO, T. **O Sistema Toyota de Produção – Além da Produção em Larga Escala**. Bookman, 1997

OSBORNE, T. **Internet Kanban Delivers Just in Time.** Illinois Manufacturer Magazine. 2002.

OZA, N.; FAGERHOLM, F.; MUNCH, J. **How does Kanban impact communication and collaboration in software engineering teams?** Cooperative and Human Aspects of Software Engineering. 6th International, p.125-128, 2013.

P. BHATT, *et al*. **Dynamics of Software Maintenance**, Sigsoft Softw. Eng. Notes, Vol. 29, pp. 1-5, 2004.

PADUELLI, M. M. **Manutenção de Software: problemas típicos e diretrizes para uma disciplina específica.** Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2007.

PETER, J. F.; PEDRYCZ, W. **Engenharia de Software: Teoria e Prática.** Rio de Janeiro: Campus, 2001.

PIETRO, M. S. Z. **Parcerias na Administração Pública: concessão, permissão, franquia, terceirização, parceria público-privada e outras formas**. 5. Ed. São Paulo: Atlas, 2005.

PIGOSKI, T. M. **Practical Software Maintenance: Best Practice for Managing you Software Investment.** Wiley, New York, 1997.

PIGOSKI, T.M. **Practical Software Maintenance: Best Practices for Software Investment**, John Wiley & Sons, Inc., 1996.

POLO, M.; PIATTINI, M.; RUIZ, F. **Advances in Software Maintenance Management: Technologies and Solutions**. Idea Group Pub. 2003

POLO, M.; PIATTINI, M.; RUIZ, F.; CALERO, C. **MANTEMA: a software maintenance methodology based on the ISO/IEC 12207 standard.** 4th IEEE International Symposium and Forum on Software Engineering Standard. Otago, Nova Zelândia: IEEE Computer Societty. p 76-81. 1999b

POLO, M.; PIATTINI, M.; RUIZ, F.; CALERO, C., **MANTEMA: A complete rigorous methodology for supporting maintenance based on the ISO/IEC 12207 standard.** 3rd IEEE European Conference on Software Maintenance and Reengineering Standards. Amsterdã, Holanda: IEEE Computer Society. p. 178-181. 1999a.

POOLE, C. J. *et al*. **Extreme maintenance**. International Conference on Software Maintenance, ICSM'01, pages 301–10. IEEE, IEEE Comp. Soc. Press, 2001.

PRESSMAN, R. S**. Software Engineering: A Practitioner's Approach**. Boston, 2011

PRESSMANN, R. **Software engineering: A practitioner's approach**, USA: McGraw-Hill. 2014.

QUMER, A.; HENDERSON-SELLERS B. **A framework to support the evaluation, adoption and improvement of agile methods in practice**. *Journal of System and Software* 2008; **81**(11):1899–1919.

RASHID, A.; WANG, W. Y.C.; DORNER, D. **Gauging the Differences between Expectation and Systems Support: The Managerial Approach of Adaptive and Perfective Software Maintenance**. Cooperation and Promotion of Information Resources in Science and Technology, Fourth International Conference on Cooperation and Promotion of Information Resources in Science and Technology, p. 45-50, 2009

RASHID, W.Y.C.; WANG, D. D. **Gauging the differences between expectation and systems support: the managerial approach of adaptive and perfective software maintenance**, 4th International Conference on Cooperation and Promotion of Information Resources in Science and Technology, 2009.

S. DAS, *et al.* **Understanding Documentation Value in Software Maintenance**, Symposium on Computer Human Interaction for the Management Of Information Technology, Cambridge, Massachusetts, 2007.

S.L. PFLEEGER, **What software engineering can learn from soccer**, IEEE Softw. 19, 64–65, 2002

SCHACH, S.R. **The Economic Impact of Reuse on Maintenance: Research and Practice**. v.6, n. 4, p. 185-96, 1994

SCHWABER, K.; BEEDLE, M. **Agile Software Development with Scrum**. Prentice Hall, Upper Saddle River, 2001

SERNA M. E., SERNA A. A. **Ontology for knowledge management in software maintenance.** International Journal of Information Management. vol. 34, p. 704-710, 2014

SILVA, P.P. **Terceirização nos Serviços Públicos**,2011. Disponível em: <tst.jus.br/documents/1295387/1313002/5.+Terceirização+nos+serviços+públicos?version=1.0>

SOMMERVILLE, I. **Software Engineering.** Pearson Education, 2011

SOUSA, T. L. DE. **Uso do Scrum na Contratação de Fábrica de Software: Uma Pesquisa-Ação em um Órgão Público Federal Brasileiro**. 2014

SOUZA, S. C. B. D. *et al.* **A Study of the Documentation Essential To Software Maintenance**, 23rd annual international conference on design of communication: documenting & designing for pervasive information, Coventry, United Kingdom, 2005.

Stapleton, J. **Business Focused Development**, second ed.Pearson Education, 2003

SWANSON, E. B. **The dimensions of maintenance, Proceedings of the 2nd international conference on Software en-gineering**, San Francisco, California, United States, 1976, pp492-497.

TAN, Y; MOOKERJEE, V. **Comparing uniform and flexible policies for software maintenance and replacemen**t. IEEE Transactions on Software Engineering, 2005.

THOMAS, B.; TILLEY, S. **Documentation for software engineers: what is needed to aid system understanding?** SIGDOC '01: Proceedings of the 19th annual international conference on Computer documentation, pages 235–236, New York, NY, USA, 2001. ACM Press.

VORA, U; SARDA, N. **Framework For Evolving Systems**, 5th Wseas International Conference On Signal Processing, Robotics And Automation (Ispra '06), Madrid, Spain, 2006.

WAN, H. **Measuring Leanness of Manufacturing Systems and Identifying Leanness Target by Considering Agility.** Virginia Polytechnic Institute and State University, Blacksburg. 2006

WAN, H.; CHEN, F.F. **A Web-based Kanban system for job dispatching, tracking, and performance monitoring**. Int J Adv Manuf Technol 38, p. 995–1005, 2008.

WANG, X.; CONBOY, K.; CAWLEY, O. **'Leagile' software development: An experience report analysis of the application of lean approaches in agile software development**. *Journal of Systems and Software*, vol. 85, p. 1287–1299, 2012.

WEBSTER, K. P. B.; OLIVEIRA, K. M.; ANQUETIL, N. **A Risk Taxonomy Proposal for Software Maintenance.** Proceedings of the 21st IEEE International Conference on Software Maintenance. IEEE Computer Society, 2005.

WINGFIELD, T. **Armadilha do Kanban: A Culpa Não Está na Técnica,** 2012. Disponível em**:** < http://www.infoq.com/br/articles/doing-kanban-wrong >

WONG, K.; TILLEY, S. R.; MULLER, H. A.; STOREY M. D. **Structural Redocumentation: A Case Study**, IEEE Software, Janeiro, 1995, p.46-54