UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CÂMPUS CORNÉLIO PROCÓPIO

DIRETORIA DE PESQUISA E PÓS - GRADUAÇÃO

PROGRAMA DE PÓS - GRADUAÇÃO EM INFORMÁTICA

HELLEN CHRISTINE SERÓDIO THOMAZINHO

UMA ANÁLISE DO PROCESSO DE MANUTENÇÃO DE SOFTWARE EM ORGANIZAÇÕES DE TECNOLOGIA DA INFORMAÇÃO

DISSERTAÇÃO DE MESTRADO

CORNÉLIO PROCÓPIO

2018

HELLEN CHRISTINE SERÓDIO THOMAZINHO

UMA ANÁLISE DO PROCESSO DE MANUTENÇÃO DE SOFTWARE EM ORGANIZAÇÕES DE TECNOLOGIA DA INFORMAÇÃO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Tecnológica Federal do Paraná – UTFPR como requisito parcial para a obtenção do título de “Mestre em Informática”.

Orientador: Prof. Dr. Alexandre L’Erario

CORNÉLIO PROCÓPIO

2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Programa de Pós-Graduação em Informática

Título da Dissertação Nº XX:

"UMA ANÁLISE DO PROCESSO DE MANUTENÇÃO DE SOFTWARE EM ORGANIZAÇÕES DE TECNOLOGIA DA INFORMAÇÃO".

por

Hellen Christine Seródio Thomazinho

Orientador: **Prof. Dr. Alexandre L'Erario**

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM INFORMÁTICA – Área de Concentração: Computação Aplicada, pelo Programa de Pós-Graduação em Informática – PPGI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Cornélio Procópio, às 08h00 do dia 19 de Fevereiro de 2018. O trabalho foi aprovado pela Banca Examinadora, composta pelos professores:

Prof. Dr. Alexandre L'Erario
(Presidente)

Prof. Dr. Wagner Fontes Godoy
(UTPFR-Cornélio Procópio)

Prof. Dr. Carlos J. Costa
(Univeridade de Lisboa - ISEG)

Visto da coordenação:

Danilo Sipoli Sanches
Coordenador do Programa de Pós-Graduação em Informática
UTFPR Câmpus Cornélio Procópio

Dedico este trabalho a Deus e minha família.

AGRADECIMENTOS

Sendo este trabalho o culminar de um ciclo, provavelmente o mais importante e desafiador de minha vida até o presente momento, gostaria de expressar meus sinceros reconhecimentos e agradecimentos a todos que, direta e indiretamente, apoiaram-me na jornada e nas experiências alcançadas por esta dissertação.

A Deus, por me dar oportunidade, capacidade e força de vontade para realizar este trabalho.

À minha família, em especial, minha mãe Julia e meu padrasto Oliveira. Pelo incentivo, ensinamentos, os gestos de carinho e os momentos dedicados a mim nessa jornada jamais serão esquecidos.

A minha amiga Vanessa, por todos os momentos de incentivo e apoio nessa jornada, por acreditar que eu seria capaz de realizar esse trabalho.

A minha amiga Diomara Barros, pelos momentos que estudamos juntas e finais de semanas que compartilhamos conhecimentos.

Aos amigos William, Eduardo e Maria Eduarda, por todos os momentos de apoio na realização desse trabalho e momentos que compartilhamos conhecimentos.

Ao meu orientador, Prof. Dr. Alexandre L'Erario, que com sua imensa paciência e sabedoria me conduziu no desenvolvimento desta pesquisa, um exemplo de profissional, de pesquisador e de pessoa que admiro. Agradeço pela motivação, por acreditar na capacidade do ser humano em aprender e praticar.

Aos professores do Programa de Mestrado em Informática da UTFPR – Câmpus Cornélio Procópio, PPGI, por me proporcionarem adquirir conhecimento e abrirem as portas da pesquisa.

“Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível.”

Charles Chaplin

RESUMO

THOMAZINHO, Hellen Christine Seródio. **Uma análise do processo de manutenção de software em organizações de tecnologia da informação.** 2018. Dissertação de Mestrado – Programa de Pós-Graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

A manutenção de software é a atividade em que ocorrem modificações nos artefatos de um software após sua entrega, com o propósito de mantê-lo disponível, corrigir suas falhas, melhorar seu desempenho e adequá-lo aos requisitos novos ou modificados, conforme as necessidades de seus usuários. É uma atividade imprescindível, sem a qual os sistemas existentes rapidamente se tornariam defasados e ineficientes para as organizações. Dentro desse contexto, o objetivo desse trabalho é, realizar uma análise do processo de manutenção de software em organizações do setor produtivo e apresentar um modelo de processo de manutenção. Nesse sentido, o propósito é amparar organizações, que desenvolvem software na identificação dos elementos que compõem um processo de manutenção e promover, melhorias no planejamento e na execução das atividades. Para atingir esse objetivo foi aplicado um método híbrido, composto por experimentos e múltiplos estudos de casos, ambos com o propósito de validar o modelo apresentado neste trabalho. Os experimentos demonstraram que foi possível validar o modelo preliminar e medir a capacidade de aprendizagem de equipes na aplicação de processos e atividades relacionadas à manutenção de software. Os resultados obtidos a partir dos estudos de caso, mostraram que as organizações não seguem um modelo único de manutenção e que a maior demanda é por correções e adaptações. Além disso, estratégias associadas ao gerenciamento do conhecimento do usuário e da equipe de desenvolvimento/manutenção são relevantes para aumentar a eficácia do processo de manutenção. Desta maneira, espera-se que tal modelo seja aderente em qualquer organização do setor produtivo, seja esta certificada ou não por algum modelo de qualidade.

Palavras-chave: Manutenção de Software, Estudo de Caso, Experimentação

ABSTRACT

THOMAZINHO, Hellen Christine Seródio. ***An analysis of the software maintenance process in information technology organizations*.** 2018. Dissertação de Mestrado – Programa de Pós-Graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

Software maintenance is the activity in which changes occur in the artifacts of a software after its delivery, in order to keep it available, correct its faults improve its performance and adapt it to new or modified requirements, according to the needs of your users. It is an essential activity, without which existing systems would quickly become lagging and inefficient for organizations. In this context, the objective of this work is to perform an analysis of the software maintenance process in organizations of the productive sector and present a model of maintenance process. In this sense, the purpose is to support organizations, which develop software in the identification of the elements that make up a maintenance process and promote, improvements in planning and execution of activities. To achieve this objective, a hybrid method was applied, consisting of experiments and multiple case studies, both with the purpose of validating the model presented in this work. The experiments demonstrated that it was possible to validate the preliminary model and measure the learning capacity of teams in the application of processes and activities related to software maintenance. The results obtained from the case studies showed that organizations do not follow a single maintenance model and that the greatest demand is for corrections and adaptations. In addition, strategies associated with knowledge management of the user and the development / maintenance team are relevant to increase the effectiveness of the maintenance process. In this way, it is expected that such model will be adherent in any organization of the productive sector, whether or not it is certified by some quality model.

Keywords: *Software Maintenance, Case Study, Experimentation*

LISTA DE GRÁFICOS

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

O ciclo de vida do desenvolvimento de software é composto por uma série de fases: planejamento, análise e especificação de requisitos, projeto, implementação, testes, entrega e implantação, operação e manutenção. Após a conclusão do seu desenvolvimento o software é entregue ao cliente e independente do domínio de aplicação, tamanho e complexidade e continuará a evoluir com o tempo. No âmbito de software as alterações ocorrem quando são corrigidos erros, quando há adaptação a um novo ambiente, quando o cliente solicita novas melhorias ou novas funções e quando a aplicação passa por um processo de reengenharia para proporcionar benefícios em contexto moderno (PRESSMAN, 2007).

Alterações ocorrem porque erros foram encontrados, porque o software precisa ser adaptado para acomodar mudanças em seu ambiente externo, ou porque o cliente necessita de funcionalidade adicional ou aumento de desempenho. Muitas vezes, dependendo do tipo e porte da manutenção necessária, essa fase pode requerer a definição de um novo processo, em que cada uma das fases precedentes é reaplicada no contexto de um software existente ao invés de um novo (PFLEEGER, 2004). A pesquisa de Nguyen; Boehm; Danphitsanuphan (2011), constatou que em um projeto de software, os principais esforços no ciclo de desenvolvimento deste foram feitos em fase de manutenção.

O desafio é que a manutenção de software está se tornando um foco central nos negócios de hoje no contexto da Tecnologia da Informação, mais empresas estão se concentrando em atualizar softwares existentes do que na implementação de novos projetos (CHUA, 2010). Apesar de ser um desafio, a manutenção de software pode sair cara para as empresas de softwares. Algumas empresas gastaram cerca de 80% ou mais de 90% do seu orçamento na manutenção de software. Portanto, é importante estimar o esforço de manutenção possível, para facilitar o planejamento e gestão de manutenção de software (LI et al., 2010). As estimativas para o custo relativo de manutenção de software e

gerenciamento de sua evolução têm variado de mais de 50% para mais de 90% do seu custo total nos orçamentos da empresas (ALARANTA; BETZ, 2011). Também é importante ressaltar que, na medida em que o software se torna mais crítico para as organizações, os procedimentos de governança de tecnologia da informação tendem a elevar o custo total de execução das atividades de manutenção, como observa Chapin (2009).

Observando as considerações supramencionadas torna-se evidente que, as empresas de software entendam a necessidade de pesquisar e relacionar mais informações sobre a atividade de manutenção de software, de maneira a evidenciar quais são as principais dificuldades e desafios dessa atividade, quais as melhores práticas e processos a serem adotados na manutenção de software e pesquisa de ferramentas existentes que apoiem essa atividade de manutenção. Sommerville (2007) ressalta que o sucesso das operações de diversas organizações está diretamente vinculado com o funcionamento adequado dos sistemas. A falta de conhecimento em modelos e abordagens que auxiliem no processo de manutenção de software nas organizações, faz com que muitos sistemas tenham características de sistemas legados (MELLEGARD et al., 2016) e (AHMAD et al., 2016). Portanto, faz-se necessário garantir a disponibilidade dos sistemas e a eficiência do atendimento das solicitações de manutenção de software.

Na manutenção de software a participação dos usuários é apresentada como elemento essencial para o sucesso de um software, pois esse sucesso está ligado a sua aceitação, satisfação e utilização (BANO; ZOWGHI, 2013). Todas as mudanças no software para corrigir defeitos e deficiências ou até mesmo novas melhorias, são encontradas durante a sua utilização pelo usuário, por isso o amparo de usuários durante essa atividade é muito importante, pois ajuda que o produto final atenda a todas as necessidades. Um produto de software pode ter um grande número de usuários e estes poderão fornecer informações pertinentes com relação à sua manutenção, pois são fontes de informações importantes para o produto.

1.1 OBJETIVOS

O objetivo deste trabalho é realizar uma análise do processo de manutenção de software em organizações do setor produtivo e apresentar um modelo de processo de manutenção. O propósito do modelo aqui apresentado é amparar organizações com relação aos elementos que compõe um processo de manutenção de software, desta maneira, espera-se que tal modelo ajude a promover nestas organizações melhorias relacionadas ao processo (redução de retrabalho, produtividade, rotatividade de equipes) e ao produto (aumento da satisfação do usuário).

Para alcançar os objetivos supramencionados os seguintes objetivos específicos foram executados neste trabalho:

- Realizar um mapeamento sistemático sobre a manutenção de software, com o objetivo de encontrar e analisar trabalhos primários relevantes nessa área, podendo assim identificar evidências que apontem desafios, melhores práticas, modelos e ferramentas de apoio a manutenção de software nas organizações de tecnologia da informação;
- Compor um modelo preliminar;
- Realizar experimentos com alunos da área de engenharia de software, com o objetivo de medir a capacidade de aprendizado das equipes em aplicar processos e atividades relacionados à manutenção de software e assim validar um primeiro modelo preliminar para manutenção de software.
- Realizar múltiplos estudos de caso, em diversas organizações de tecnologia da informação, com o objetivo de identificar elementos do processo de manutenção de software, e validar um modelo efetivo de manutenção de software.

1.2 ESTRUTURA DO TRABALHO

Esse trabalho está divido em 5 capítulos. No presente capítulo foi apresentado o contexto que insere o trabalho, as motivações para seu desenvolvimento e os objetivos. No capítulo 2 é apresentado o mapeamento sistemático para identificar, interpretar e avaliar as pesquisas existentes na área de manutenção de software, assim como a revisão bibliográfica relativa à manutenção de software, expondo os conceitos importantes que serão utilizados ao longo do texto. No capítulo 3 é apresentada a metodologia hibrida escolhida para condução desse trabalho, assim como hipóteses e operacionalização de cada metodologia. No capítulo 4 são apresentadas as análises e resultados de ambas as metodologias aplicadas nesse estudo. No capítulo 5 são apresentados os resultados dessa pesquisa, e para finalizar, as considerações finais e trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

A base teórica necessária para a realização da pesquisa e o entendimento do estudo é apresentada nesse capítulo. O capítulo está organizado da seguinte forma:

2.1 Manutenção de Software – nesta seção são apresentados os conceitos sobre Manutenção de Software, como: objetivo e motivação para sua realização, suas características e tipos de manutenções.

2.2 Mapeamento Sistemático – nesta seção são apresentados os procedimentos seguidos pelo mapeamento sistemático, com apresentação do protocolo definido, a forma como os dados foram extraídos, analisados e sintetizados.

O mapeamento sistemático, neste trabalho, tem com o objetivo de encontrar e analisar trabalhos primários relevantes e reconhecidos na área de manutenção de software, que pudesse responder as questões de pesquisa apresentadas na seção 2.2.1.

2.1 MANUTENÇÃO DE SOFTWARE

A atividade de manutenção de software é caracterizada pela modificação de um produto de software já entregue ao cliente, para a correção de eventuais erros, melhorias de desempenho, ou qualquer atributo, ou ainda para adaptação desse produto a um ambiente modificado. A manutenção de software pode ser definida como uma parte do ciclo de desenvolvimento do software (STANDARD, 2006). Para Sommerville (2011), a manutenção de software é o processo geral de modificação de um sistema depois que ele é liberado para uso. A manutenção de software corrige defeitos, adapta o software para atender a um ambiente em

mutação e melhora a funcionalidade para atender as necessidades dos clientes (PRESSMAN, 2007). De fato, não é raro em uma organização de software depender de 60% a 70% de todos seus recursos com manutenção de software (PRESSMAN, 2007).

Manutenção é direcionada para a preservação do software existente, corrigindo os erros e fazendo as mudanças necessárias para manter o software em reciprocidade com o ambiente operacional. Evolução, por outro lado, melhora a funcionalidade e a qualidade do software. Manutenção é impulsionada por relatórios de erros e pedidos de mudança. Evolução é impulsionado por novas exigências, funcionais ou não funcionais. Os requisitos funcionais solicitam funcionalidades adicionais, requisitos não-funcionais solicitam a melhoria da qualidade (ASSESSMENT; PRENTNER; SNEED, 2016). Ambos vêm na forma de um pedido de mudança.

2.1.1 Tipos de Manutenção de Software

Segundo Rezende (2005) “de maneira geral, todo software sofre manutenções, sejam elas para simples ajustes pós-implantação, ou por melhorias substanciais, por força da legislação e, finalmente, por estar gerando erros.” Neste sentido Sommerville (2011) lembra que existem três diferentes tipos de manutenção de software: manutenção para reparar os defeitos no software, manutenção para adaptar o software a um ambiente operacional diferente e manutenção para acrescentar funcionalidade. Apesar das diferenças em terminologia, pesquisas em geral indicam que o investimento financeiro é maior na fase de manutenção do software do que na fase de projeto (desenvolvimento); e dentro da fase de manutenção, existe um esforço maior na implementação de novos requisitos do que na correção de defeitos (SOMMERVILLE, 2011a).

A manutenção corretiva refere-se a modificações necessárias por erros reais em um produto de software. Se o produto de software não atender aos requisitos, a manutenção corretiva será executada. A manutenção de emergência é uma modificação não programada realizada para manter temporariamente um sistema operacional pendente de manutenção corretiva (STANDARD, 2006). Um exemplo desse tipo de manutenção consiste em um usuário apresentando um problema ao executar uma consulta de clientes. O número de informações é muito grande de acordo com o período selecionado, e aparece uma mensagem de erro no sistema. O problema foi identificado como uma falha ao executar a consulta, no banco de dados, sendo necessário alterar a scripts que busca essas informações.

Manutenções do tipo adaptativas referem-se a adequar o software ao seu ambiente externo, incluem modificações para implementar novos requisitos de interface do sistema, novos requisitos de sistema, ou novos requisitos de *hardware* (STANDARD, 2006). O exemplo apontado por Pfleeger (2004) ilustra bem essa categoria. Suponha um gerenciador de banco de dados, que faz parte um sistema maior de *hardware* e software. Em uma atualização do gerenciador, os programadores perceberam que as já existentes rotinas de acesso a disco precisavam agora de mais um parâmetro adicional. Essa manutenção corresponde a uma manutenção adaptativa, uma vez que teve por finalidade adequação do software ao seu ambiente e não a correção de um defeito.

Manutenções do tipo perfectivas têm por objetivo acrescentar novos recursos de funcionalidade ao software, normalmente em razão de solicitações dos usuários, uma modificação perfectiva pode implicar proporcionando novas melhorias de funcionalidade para os usuários ou engenharia reversa para criar documentação de manutenção que não existia anteriormente ou para modificar a documentação existente (STANDARD, 2006). Significam ainda redesenhar partes de um software, de forma a tornar mais simples a compreensão e utilização do mesmo. Como exemplo, pode-se citar o pedido do usuário por um novo relatório com informações que até então não podiam ser obtidas do banco de dados.

Uma quarta categoria de manutenção é apresentada por alguns autores, como Pressman (2007). Essa categoria se refere a manutenções do tipo preventivas

que buscam identificar previamente possíveis fontes de problemas no software e corrigi-las antecipadamente. De acordo com a norma ISO/IEC 14764:2006 a manutenção preventiva refere-se às modificações exigidas pela detecção de possíveis erros em um produto de software (STANDARD, 2006).

Estes quatro tipos de manutenção de software podem ser estruturados como apresentado pela Figura 1, que exemplifica esta estrutura a partir de uma do pedido de modificação de software, seguido por uma avaliação genérica dos tipos de manutenção que seriam os processos de correção ou melhoria no software.

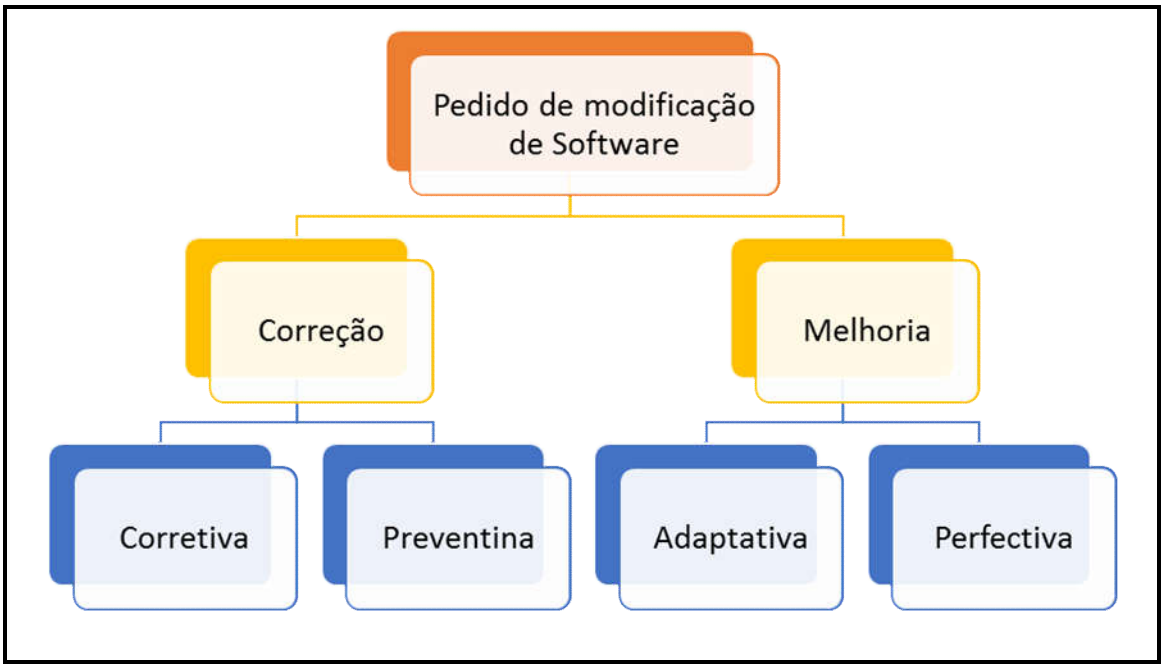

Figura 1 - Tipos de Manutenção de Software
Fonte: Adaptado de ISO/IEC 14764:2006

2.1.2 Normas de Manutenção de Software

O documento padrão de manutenção de software da ISO é a norma ISO/IEC 14764 (ISO, 2006). A norma ISO/IEC 14764 descreve um *framework*, cujo objetivo é apoiar o planejamento, o gerenciamento e a execução das atividades de manutenção, independentemente do tamanho e da complexidade do software, da criticidade das modificações e do domínio do negócio. As diretrizes apresentadas por essa norma englobam desde o planejamento da fase de manutenção e a definição da estratégia a ser adotada até a retirada do software do ambiente de produção ao final da sua vida útil.

O processo de manutenção proposto pela norma ISO/IEC 14764 possui duas fases, a primeira é responsável pelas atividades executadas durante o desenvolvimento inicial e visam, por exemplo, auxiliar a transferência do conhecimento entre a equipe de desenvolvimento e a equipe de manutenção. Essas atividades também viabilizam o planejamento da execução do processo de manutenção após a entrega do software, mitigando antecipadamente os riscos de insucesso. A segunda fase está relacionada ao período após o desenvolvimento inicial e almeja garantir a operacionalidade e a evolução do software para que esse acompanhe as necessidades dos usuários.

A equipe de manutenção deve garantir que o processo de manutenção existe e é funcional antes do desenvolvimento de qualquer produto de software. O processo de manutenção deverá ser acionado após solicitação de mudanças no produto de software. Assim que o processo for acionado, os planos e procedimentos de manutenção devem ser desenvolvidos e os recursos devem ser alocados especificamente para o tipo de manutenção. Depois que o produto de software é entregue, a equipe de manutenção deve modificar o código e documentação associada, em resposta a um pedido de modificação ou relatório de problema. O objetivo global da manutenção de software é modificar o produto existente, preservando a sua integridade. Este processo suporta o produto de software desde

o seu início através da migração para novos ambientes, ao seu encerramento. O processo termina quando o produto software está finalmente encerrado (ISO, 2006).

As atividades que compõem o processo de manutenção são:

a) Implementação do processo;

b) Análise do problema e da modificação;

c) Implementação da modificação;

d) Aceitação e revisão da modificação;

e) Migração;

f) Retirada/Encerramento.

Cada uma das atividades relacionadas supramencionadas possui entradas, tarefas, controles, suporte e saídas. As entradas são transformadas ou consumidas pelas atividades de manutenção para produzir resultados. Controles fornecem orientações para garantir que a atividade de manutenção produza saídas corretas. O suporte contempla os recursos ou objetos utilizados para consulta. As saídas são os dados ou os objetos produzidos pela atividade da manutenção. Na Figura 2 é possível analisar todas as atividades que compõe o processo de manutenção de software da norma ISO/IEC 14764.

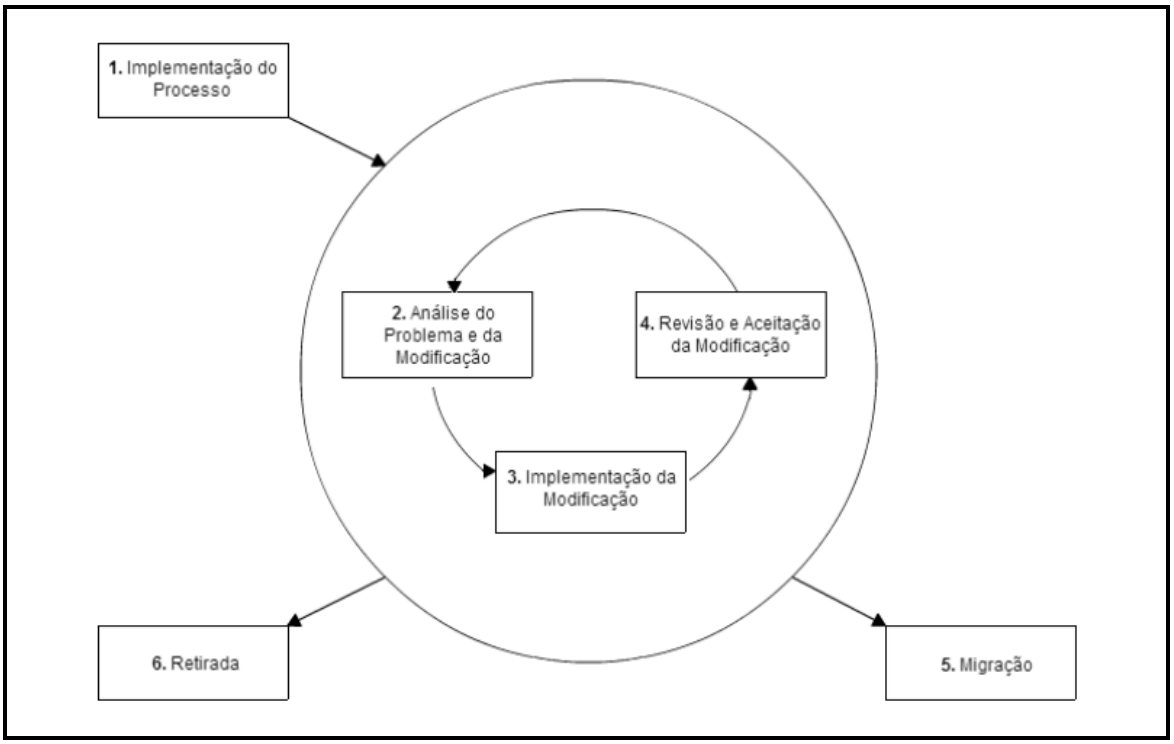

Figura 2 - Processo de manutenção de software - ISO/IEC 14764
Adaptado da Norma ISO/IEC 14764

A ISO/IEC 12207 estabelece uma estrutura comum para os processos de ciclo de vida de *software*, com terminologia bem definida, que pode ser referenciada pela indústria de *software*. A estrutura contém processos, atividades e tarefas que servem para ser aplicadas durante a aquisição de um sistema que contém *software*, de um produto de *software* independente ou de um serviço de *software*, e durante o fornecimento, desenvolvimento, operação e manutenção de produtos de *software*. A norma também provê um processo que pode ser utilizado para definir, controlar e melhorar os processos de ciclo de vida de *software*. Essa norma foi elaborada para adquirentes de sistemas e produtos e serviços de *software*, e para fornecedores, desenvolvedores, operadores, mantenedores, gerentes, gerentes de garantia da qualidade e usuários dos produtos de *software* (ISO, 2008)(KOSCIANSKI; SOARES, 2007).

As atividades dessa norma são agrupadas e podem ser executadas durante o ciclo de vida de *software* em cinco processos fundamentais, oito

processos de apoio e quatro processos organizacionais. Cada processo de ciclo de vida é dividido em um conjunto de atividades; cada atividade é então dividida em um conjunto de tarefas. Esses processos são classificados em três tipos: fundamentais, de apoio e organizacionais, conforme ilustra a Figura 3. Os processos de apoio e organizacionais devem existir independentemente da organização e do projeto que está sendo executado. Os processos fundamentais são instanciados de acordo com a situação (ISO, 2008).

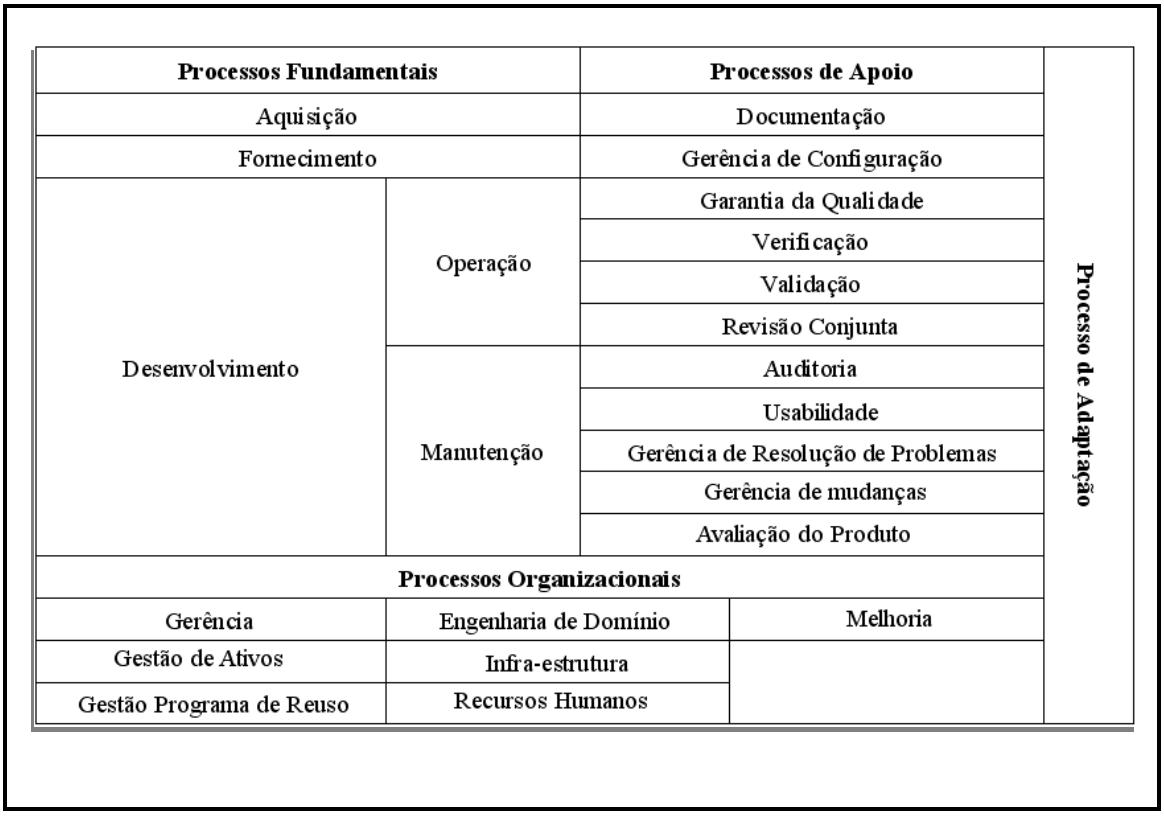

Figura 3 - Processos ISO 12207
Adaptado da Norma ISO/IEC 12207

O CMMI (*Capability Maturity Model Integration*) é um metamodelo de processo abrangente qualificado em uma série de capacidades de sistema de engenharia de *software* que devem estar presentes à medida que as organizações alcançam diferentes níveis de capacidade e maturidade de processo (PRESSMAN, 2007). As práticas que são abordadas neste modelo são: gerenciamento de requisitos, manipulação de riscos, medição de desempenho, planejamento de trabalho, tomada de decisão, entre outros

O objetivo da CMMI é servir de guia para melhoria de processos na organização e também da habilidade dos profissionais em gerenciar o desenvolvimento, aquisição e manutenção de produtos ou serviços. Espera-se que com o uso do CMMI a organização seja mais eficiente, respeitando seus próprios prazos e construindo *software* com menos erros (KOSCIANSKI; SOARES, 2007).

Ahern Dennis et al (2008) acrescentam que o CMMI é utilizado como guia e suporte para as atividades gerenciais e técnicas através dos objetivos: criar produtos e serviços de qualidade; enriquecer o dia a dia dos funcionários, aumentar a satisfação do cliente; implantar economia de custo e melhores práticas, além de encorajar a medição como ferramenta de gerencia.

As normas associados a software apresentam diferentes focos, por exemplo, o CMMI não apresenta explicitamente nenhuma abordagem relacionada à manutenção, enquanto que a norma da ISO/IEC 14764 faz uma abordagem somente nos procedimentos técnicos de manutenção de software e a ISO/IEC 12207 descreve a arquitetura dos processos, embora não detalhe como implementá-los, em relação a todas as tarefas necessárias para o desenvolvimento e a manutenção do software.

2.2 MAPEAMENTO SISTEMÁTICO

De acordo com Petersen et al. (2008) o mapeamento sistemático fornece uma visão geral da área de pesquisa, possibilitando que se conheça a frequência, quantidade e tipos de estudos referentes às publicações ao longo do tempo, permitindo identificar tendências de pesquisas. Nesta seção são apresentados tópicos do protocolo de pesquisa que guiou o mapeamento. Este estudo baseou-se nas diretrizes propostas por Petersen et al. (2008) e foi conduzido em cinco passos a seguir: (i) definição de questões de pesquisa, (ii) condução da pesquisa, (iii) *String* de busca (iv) seleção dos estudos primários, e (v) análise dos resultados.

2.2.1 Definição das questões de pesquisas

Procurou-se responder as questões de pesquisa que serviram de guia no mapeamento desta pesquisa, de modo que foi possível concluir e explanar acerca da manutenção de software em empresas. As questões previamente definidas foram:

(Q1) Quais os principais desafios na manutenção de software?

(Q2) Quais as melhores práticas adotadas na manutenção de software?

(Q3) Que ferramentas existem para apoiar a manutenção de software?

2.2.2 Condução da pesquisa

Com as questões de pesquisa definidas, foi realizada uma busca manual de possíveis trabalhos. Para este processo, restringiu-se a busca nas seguintes bibliotecas digitais: IEEEXplore[1], ACM *Digital Library*[2], *Springer* [3], com o objetivo de encontrar estudos de manutenção de software em organizações.

2.2.3 *String* de busca

Para realização das buscas nas bibliotecas digitais, foi elaborada uma *string* de busca em inglês, formatada de acordo com as regras de cada biblioteca digital. De acordo com Petersen et al. (2008), os estudos primários são identificados por utilizar *strings* de busca em base de dados ou manualmente em publicações periódicas. A Tabela 1 apresenta a *strings* elaborada para utilização nas bases de dados.

Tabela 1 - *String* de busca realizadas

(("Software maintenance " OR "Software maintenance in companies" OR "model for maintenance software") AND (techniques for maintenance software OR process maintenance software OR models for maintenance software))

[1] http://ieeexplore.ieee.org/Xplore/home.jsp

[2] http://dl.acm.org/

[3] http://www.springer.com

2.2.4 Seleção dos estudos primários

Os estudos selecionados foram analisados a fim de que fossem identificados aqueles relevantes ao assunto da pesquisa. Segundo Kitchenham; Charters (2007) as buscas iniciais retornam uma grande quantidade de estudos que não são relevantes, não respondem às questões ou mesmo não tem relação com o tópico em questão. Sendo assim a etapa de seleção dos estudos primários foi dividida em: Definição de Critérios de Inclusão e Exclusão de estudos primários e definição do processo de seleção dos mesmos.

Nesse estudo foi definido como critérios de inclusão:

1. Estudos primários publicados em conferências e periódicos que relatam Manutenção de Software;
2. Artigos que mostram estudos similares, somente o mais recente (a partir de 2009) deverá ser incluído;
3. Artigos publicados entre 2009 e 2017;
4. Estudos que apresentem relatos de experiência na fábrica de software, ou pesquisas de caráter experimental ou teórico, contanto que apresente exemplos de aplicação, descrição de experimentos ou casos reais de uso de abordagens para apoio às atividades de Manutenção de Software.

Os critérios de exclusão desse estudo foram:

1. Artigos que não apresentem relação com Manutenção de software em empresas serão excluídos;
2. Estudos que não estiverem inseridos no contexto de Projetos de Software, Indústria de Software ou Engenharia de Software;
3. Estudos que claramente não atendam às questões de pesquisa;
4. Estudos que não atendam o período selecionado.

Inicialmente as buscas retornaram 43 estudos primários (O Anexo 1 apresenta a lista de todos eles) que relatavam a manutenção de software em empresas, assim foi realizada a leitura dos títulos, resumos e palavras-chave de todos os artigos e cada um gerou uma lista de estudos selecionados. Após a análise da lista, foi realizada novamente a leitura dos resumos, introdução e conclusão e a aplicação dos critérios de inclusão e exclusão, o que resultou em um subconjunto de 20 estudos considerando as 3 bibliotecas digitais.

2.2.5 Análise dos resultados

Após identificação foram analisados 20 estudos, respondendo as questões de pesquisa definidas na Seção 2.1.1. Após serem apresentados dados gerais da análise, serão apresentadas nos tópicos abaixo as respostas das questões de pesquisa baseada na análise que foi realizada no mapeamento sistemático.

(Q1) Quais os principais desafios na manutenção de software nas organizações?

De acordo com o levantamento concluído, os autores Li et al. (2010), citam os altos custos como maior desafio na manutenção de software. Já os autores Stojanov et al. (2013) afirmam que a estimativa de esforço de manutenção de software em empresas é um dos principais desafios, pois a manutenção de software consome a maior parte das horas diárias de trabalho para todos os programadores. Além disso, a maior parte das atividades de manutenção de software estão relacionados com a resolução de pedidos dos usuários, por outro lado os autores Tsunoda; Ono (2014) e Tsunoda et al. (2015), afirmam que a eficiência de trabalho para manutenção e suporte de software é algo a ser estudados para as empresas. Ainda nesse contexto os autores Van Der Schuur; Jansen; Brinkkemper (2011), afirmam que a priorização das tarefas de manutenção de software assim como a

implementação de processos nessa atividade, é totalmente essencial para as tarefas de manutenção de software.

A falta de conhecimento na atividade de manutenção corretiva de software nas empresas é citada pelos autores Alaranta; Betz (2011), e os autores Mellegard et al. (2016) e Ahmad et al. (2016) citam a falta de modelos e abordagens que auxiliem o desenvolvimento desses softwares nas empresas, pois muitos deles têm as características de sistemas legados.

A avaliação da qualidade de produtos de software, medição de defeitos e abordagens de identificação de versões individuais em manutenção de software é relatada por Singh; Sangwan (2014) como um dos desafios, pois os três permitem que os profissionais de software avaliem o que funciona e o que não funciona em um software identificando melhorias e facilidade de manutenção.

Para os autores Poklemba; Sivy; Havlice (2009), o principal desafio em manutenção de software é o alto custo que a empresa tem no momento de um novo desenvolvimento para realizar a manutenção em seus produtos.

Mudar a estruturação dos analistas que participam da manutenção de software e a prática de novas melhorias nessa equipe de manutenção é fundamental na visão dos autores Ohba et al. (2007), pois a partir dessa nova estruturação será possível criar um indicador para cronometrar a transferência de responsabilidades de manutenção de software dentro da empresa e ponderar sobre o tempo de entrega das responsabilidades de manutenção do sistema.

Os autores Stojanov; Brtka; Dobrilovic (2014) e Marques-Neto; Aparecido; Valente (2013) consideram que a avaliação dos processos de manutenção é um grande desafio das empresas, pois além de melhorar as atividades de planejamento é essencial para aumentar a eficiência dos serviços prestados aos clientes e a qualidade dos produtos de software.

A análise de ticket (chamados) para os autores Christa et al (2016), se torna um grande desafio, pois muita das vezes não fornece informações relacionadas ao tipo de manutenção que deve ser realizada. Classificar e analisar

esses tickets torna-se uma tarefa crítica para a equipe que está executando a manutenção.

Para os autores Budiardjo et al. (2016), um dos principais obstáculos encontrados na manutenção de software é a falta de conhecimento da equipe de desenvolvimento nos produtos de software fornecidos pelas organizações.

(Q2) Quais as melhores práticas a serem adotadas na manutenção de *software*?

Para os autores Stojanov et al. (2013), a melhor prática a ser adotada é um modelo para estimar o número médio de horas de trabalho para a frequência conhecida dos pedidos de manutenção de software. Os autores Poklemba; Sivy; Havlice (2009) e Dantas et al. (2013), acreditam que a melhor prática a ser adotada é a implantação de um processo que explore o que deveria ser a maneira correta de *design* de software, quais são os erros mais comuns, e como ganhar a manutenção de baixo custo nesse software. E propor uma avaliação e seleção de processos de manutenção de software que auxiliem no planejamento de atividades de manutenção é abordagem que as empresas de software devem adotar segundo os autores Stojanov; Brtka; Dobrilovic (2014) e Marques-Neto; Aparecido; Valente (2013).

A aplicação de metodologia ágil é uma prática para facilitar a manutenção de software, pois esses métodos ágeis podem ajudar a acelerar o processo e melhorar a qualidade do código, trazendo a satisfação do usuário e a motivação assim como a comunicação entre os membros da equipe de manutenção segundo os autores Ahmad et al. (2016). Em relação ao trabalho da equipe de manutenção de software, os autores Van Der Schuur; Jansen; Brinkkemper (2011) defendem que aplicar um histórico de operações de software, irá promover o consenso entre os funcionários sobre a prioridade de tarefas de manutenção a serem desenvolvidas e quantos *bugs* devem ser corrigidos, assim facilitando a comunicação entre as

equipes e priorizando datas de entregas e atividades e os autores Ohba et al.(2007), afirmam que a criação de um modelo para estruturação para nova equipe de manutenção de software, analisando assim o tempo para entrega de melhorias na manutenção e determinando o momento ideal para fazê-lo é essencial para a adaptação e produtividade dessa equipe de manutenção. Nesse mesmo segmento em melhoria de equipe de manutenção de software, os autores Alaranta; Betz (2011) sugerem que a implantação de uma teoria da coordenação de conhecimento em grupos de desenvolvimento, poderá sanar os problemas no processo de manutenção de software das empresas.

Implantar uma estrutura de métricas de processo de software orientada a aspectos para prever a bugs e identificar o índice de manutenção de software é uma proposta de melhoria de acordo com os autores Singh; Sangwan (2014).

A aplicação e o desenvolvimento de diagramas UML no desenvolvimento de um *software* (*Unified Modeling Language*) segundo os autores Fernandez-Saez et al (2015) e Dantas et al. (2013), facilitaria as tarefas de manutenções de software futuras, encorajando mantenedores a manter os diagramas atualizados e assim ajudar as empresas a analisar e verificar como os sistemas estão sendo mantidos, ficando mais fácil de atingir as perspectivas de seus usuários finais.

(Q3) Que ferramentas existem para apoiar a manutenção de *software*?

Com o objetivo de responder a questão 3 as ferramentas encontradas nos artigos mapeados, que apóiem processos ou melhorias na manutenção de software nas empresas, foram identificadas.

A norma atual (em revisão) que apoia o planejamento, o gerenciamento e a execução das atividades de manutenção de software é a ISO/IEC 14764:2006, esta norma define que um meio potencial de conter os custos de manutenção de software é a utilização de ferramentas CASE. As atividades de manutenção de software necessitam de ajuda de ferramentas. A visão para CASE é um conjunto inter-relacionado de ferramentas de apoio a todos os aspectos do desenvolvimento e

manutenção de software. Esta coleção inter-relacionada de ferramentas CASE devem ser reunidas sob a forma de um Ambiente de Engenharia de Software (SEE) para apoiar os métodos, políticas, diretrizes e normas que suportam as atividades de manutenção de software.

De acordo com Ahmad et al. (2016), o *Kanban* tem sido uma ferramenta que apoie a manutenção de software, pois, oferece vários benefícios para os trabalhos de manutenção, tais como trazer visibilidade às tarefas de manutenção, protegendo as equipes do excesso de comprometimento, ajudando na priorização de tarefas, sincronização de trabalho com outras equipes, ainda afirmam que ao usar o *Kanban*, a ação da equipe para o trabalho urgente é mais espontâneo, e o trabalho pode ser puxado de acordo com altas prioridades.

Os autores Poklemba; Sivy; Havlice (2009), relatam em seu artigo que o modelo *V-Model* é um modelo de processo no desenvolvimento de software, desenvolvido pelo Ministério Federal Alemão de Defesa dos Estados Unidos. Esse modelo pode ser visto como um modelo Cascata, permitindo um *feedback* ao correspondente fases durante o processo.

A implantação da ferramenta para manutenção e suporte à sistemas ISBSG (*International Software Benchmarking Standards Group*) é citada pelos autores Tsunoda; Ono (2014), pois ela constrói um conjunto de dados de manutenção e suporte de software, assim podendo analisar relatórios do planejamento, da fase de manutenção e suporte do projeto das empresas de softwares.

2.3 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os constructos teóricos desta pesquisa. Foram apresentados nesse capítulo o conceito de manutenção de software, tipos, características e padrões, o que pode se concluir que os modelos e normas aqui apresentados que estabelecem o processo de manutenção de software possuem diferentes focos, sendo que alguns se atentam mais a questões econômicas, outros ao produto e outros ao próprio processo. Todos possuem vantagens e desvantagens, não existindo um modelo único aplicável às diversas situações e que atenda perfeitamente a manutenção de software nas organizações. Foi apresentado também o mapeamento sistemático, como o mesmo foi estruturado, conduzido e as razões de uso dos procedimentos e métodos.

3. METÓDOS E PROCEDIMENTOS DE PESQUISA

De acordo com Fonseca et al.(2002) a pesquisa possibilita uma aproximação e um entendimento da realidade a investigar, como um processo permanentemente inacabado. Ela se processa através de aproximações sucessivas da realidade, fornecendo subsídios para uma intervenção no real. Fabri et al. (2005) afirmam que toda e qualquer pesquisa científica deve trazer consigo um valor agregado, para que a humanidade possa se beneficiar e conquistar uma melhoria constante em sua qualidade de vida.

Para o desenvolvimento desse trabalho, foi utilizado o método híbrido composto por experimentos e um estudo de múltiplos casos, o objetivo dos experimentos foi verificar um primeiro modelo de processo preliminar e medir a capacidade de aprendizagem de equipes na aplicação de processos e atividades relacionadas à manutenção de software, e o objetivo dos estudos de caso foi consolidar um modelo para apoiar a atividade de manutenção do software nas organizações.

Os resultados parciais do estudo experimental descrito neste capítulo também é relatado no artigo "*Teaching software maintenance with ludic techniques supported by robotics",* Thomazinho, H.C.S, L´Erario, Alexandre, Fabri, José Augusto, publicado e apresentado na *Frontiers in Education Conference (FIE),* Indianópolis, Estados Unidos (DOI: 10.1109/FIE.2017.8190720).

3.1 EXPERIMENTO

De acordo com Wohlin et al. (2012), a pesquisa experimental consiste em determinar um objeto de estudo, selecionando as variáveis que poderiam influenciá-

lo, definindo as formas de controle e as observações dos efeitos que a variável produz no objeto.

A pesquisa experimental caracteriza-se pela manipulação de um aspecto da realidade pelo pesquisador. O pesquisador introduz, por exemplo, uma nova técnica em uma empresa de software e observa se a produtividade aumenta (WAZLAWICK, 2014).

Os experimentos apresentados nesse trabalho, foram realizados com base em um plano de execução dividido nas seguintes etapas: Definição da Hipótese, Definição do Ambiente, Definição do Assunto, Definição da Amostra, Execução do Experimento.

3.1.1 Definição da hipótese

Um aspecto que diferencia o trabalho científico do trabalho técnico é a existência de uma hipótese de pesquisa. Para Wazlawick (2014), “a hipótese é uma afirmação da qual não se sabe a princípio se é verdadeira ou falsa”.

A questão da pesquisa, delineada neste trabalho, que forneceu o mapeamento do método experimental foi:

- **O modelo preliminar de processo de manutenção de software é válido?**
- **É possível o ensino de Manutenção de Software, por meio de técnicas lúdicas em robótica?**

Para a questão da pesquisa, existem duas hipóteses possíveis:

- H1) Sim, o modelo preliminar de processo de manutenção de software é válido;

- H2) Não, o modelo preliminar de processo de manutenção de software não é válido.
- H3) Sim, é possível o ensino de manutenção de software, através de tecnicas lúdicas em robótica, desde que os alunos tenham conhecimento no produto e em suas regras de negócio;
- H4) Não é possível o ensino de manutenção de software através de tecnicas lúdicas em robótica.

3.1.2 Definição do ambiente

Os experimentos foram realizados em um ambiente acadêmico, o primeiro experimento foi realizado na disciplina de Gerenciamento de Projetos de Software existente no programa de mestrado da Universidade Tecnológica Federal do Paraná (UTFPR) - Campus Cornélio Procópio - PR, cuja linha de pesquisa é Engenharia de Software. E o segundo experimento foi realizado na disciplina de Engenharia de Software do curso de graduação em Tecnologia em Análise e Desenvolvimento de Sistemas, no mesmo campus.

Os alunos foram selecionados aleatoriamente do programa de graduação e mestrado para realizar os experimentos. Cada experimento foi seguido sem interferência durante a execução. Os grupos foram divididos da seguinte forma:

Graduação:

1 - Experimento executado com grupo de 2 alunos (GRUPO 1);

2 – Experimento executado com grupo de 4 alunos (GRUPO 2);

Mestrado:

1 - Experimento executado com grupo de 4 alunos (GRUPO 3);

2 -Experimento executado com 2 grupos no total de 10 alunos (GRUPO 4);

Cada participante teve acesso a um formulário de caracterização. Este formulário visa identificar minimamente o profissional que participará do experimento. O formulário usado neste experimento pode ser obtido no endereço https://goo.gl/VlZtSP

3.1.3 Definição da amostra

Os experimentos foram divididos por grupos, nos experimentos realizados no mestrado profissional, o número de participantes foi de 14 alunos, nos experimentos realizados na graduação o número de participantes foi de 6 alunos.

3.1.4 Execução do Experimento

A execução do experimento visa caracterizar os passos que o pesquisador seguirá para realização do mesmo, importante ressaltar que os alunos aplicaram um processo de manutenção de software. Eles seguiram um processo como desenvolvedores, conhecendo a tecnologia, mas não o processo como um todo. As etapas do experimento abaixo são do ponto de vista do ensino:

1. Reúna todos os alunos em uma sala de aula;

2. Realize a distribuição de alunos por grupos;

2. Solicite que cada profissional preencha o formulário de Caracterização, contendo Nome, Grupo Participante, Idade, Curso pertencente, Tempo de

experiência em Engenharia de Software, Experiência em desenvolvimento em Java e Experiência em *Lejos*.

3. Entregue o termo de consentimento para assinatura, disponível em https://goo.gl/usWevJ

4. Realize uma apresentação aos participantes sobre os conceitos de manutenção de software e funcionalidades do robô Lego *Mindstorms*. Apresentações disponíveis em https://goo.gl/fDGKnG

5. Peça que a os grupos executem a configuração do ambiente, disponível em https://goo.gl/BmK7XB

6. Depois do ambiente configurado, os grupos devem baixar a primeira versão do projeto, disponível em: (https://github.com/alerario/LejosCircuit).

7. Cada grupo recebeu uma ordem de serviço de manutenção, e essa ordem de serviço continha o detalhe do problema apresentado no robô (produto), visto que o mesmo realizava um trajeto com problemas.

8. Disponibilize para cada grupo de participantes o robô *Mindstorms* para a execução da manutenção de software com o propósito de fazer o robô realizar o trajeto corretamente.

9. Defina um tempo estimado para realização da atividade proposta.

10. Cada grupo deve entregar o código fonte para correção, e disponibilizá-lo no repositório.

11. Após realizar a manutenção e execução das atividades propostas, será necessário preencher outro questionário para avaliar o uso de *Mindstorms* na atividade realizada de manutenção de software. Este formulário estará disponível em https://goo.gl/RBZQKD.

A proposta para os experimentos realizados foi resolver um erro de percurso realizado pelo robô LEGO *Mindstorms*. Foi realizada uma manutenção corretiva para que o robô pudesse executar um percurso identificando cores e seguindo seu trajeto corretamente até o final da pista disponibilizada.

Além de realizar esta manutenção, as equipes deviam considerar as atividades de análise e manutenção de software. Uma dessas atividades era identificar através de uma ordem de serviço o problema a ser resolvido, com base na documentação disponível. A outra atividade era a execução de testes após a alteração no código-fonte. Importante salientar que os alunos não tinham informações sobre a tecnologia LEGO *Mindstorms*. Dado esse fato, uma das atividades deste experimento foi precisamente mapear esta informação para a análise de desempenho no problema indicado.

A Figura 4 irá ilustrar o caminho a ser realizado pelo robô nos experimentos realizados.

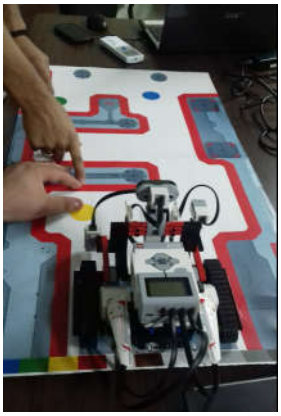

Figura 4 - Trajeto a ser executado pelo Robô

Um questionário composto por 21 questões foi aplicado após a manutenção corretiva do software, o objetivo desse questionário foi levantar informações sobre a tarefa realizada pelos participantes.

Tabela 2 - Questões Aplicadas após Experimento

Q1	A realização da manutenção foi uma tarefa simples?
Q2	Você considera que sua equipe teve habilidade para executar a manutenção?
Q3	Você teve capacidade de relacionar/atender o usuário?
Q4	O tempo alocado foi suficiente para executar a manutenção solicitada?
Q5	Você precisou realizar um treinamento com um colega?
Q6	No seu grupo foi definida uma pessoa para realizar a aceitação/revisão da manutenção realizada?

Q7	Você acha importante alguém com o papel para realizar/aceitar a manutenção realizada?
Q8	Houve a identificação do problema a ser solucionado?
Q9	Você considerou importante conhecer o produto, antes de executar a manutenção?
Q10	O conhecimento do usuário foi necessário para a manutenção do produto?
Q11	Você considerou importante conhecer a regra de negócio do produto, antes de executar a manutenção?
Q12	Você teve dificuldade na configuração do ambiente para desenvolvimento da manutenção solicitada?
Q13	O tempo para iniciar o desenvolvimento da manutenção foi adequado?
Q14	No prazo estipulado, você concorda que não gastou muito tempo para conhecer o produto?
Q15	Você executou os procedimentos (processos) de manutenção conforme estipulados?
Q16	Você considerou importante conhecer a arquitetura do software, antes de executar a manutenção?
Q17	Você considerou importante conhecer as bibliotecas utilizadas pelo produto, antes de executar a manutenção?
Q18	Você considerou importante conhecer as regras de negócio do produto, antes de executar a manutenção?
Q19	Você considerou importante conhecer a IDE, antes de executar a manutenção?
Q20	Foi possível identificar o problema através da Solicitação da manutenção (Ordem de serviço)?
Q21	Em algum momento você precisou entrar em contato com o usuário que abriu a Ordem de Serviço para esclarecimento?

3.2 ESTUDO DE CASO

De acordo com Yin (2013) "*o estudo de caso é uma investigação empírica de um fenômeno contemporâneo dentro de um contexto da vida real, sendo que os limites entre o fenômeno e o contexto não estão claramente definido".*

Os estudos de caso não buscam a generalização de seus resultados, mas sim a compreensão e interpretação mais profunda dos fatos e fenômenos específicos. Embora não possam ser generalizados, os resultados obtidos devem possibilitar a disseminação do conhecimento, por meio de possíveis generalizações ou proposições teóricas que podem surgir do estudo (YIN, 2013).

Yin (2013) apresenta quatro tipos básicos de estudo de caso:

1. Projetos de caso único holístico – unidade única de análise e único caso;

2. Projetos de caso único incorporado – unidades múltiplas de análise e único caso;

3. Projetos de casos múltiplos holísticos – unidade única de análise e múltiplos casos;

4. Projetos de casos múltiplos incorporados – unidades múltiplas de análise e múltiplos casos.

Nesse trabalho foi utilizado o estudo de casos múltiplos holísticos, pois o mesmo protocolo de pesquisa foi aplicado em várias organizações do setor produtivo de software do Brasil.

3.2.1 Protocolo da Pesquisa

O estudo de caso utiliza para coleta de dados, principalmente, seis fontes distintas de informação: documentos, registros em arquivos, entrevistas, observação direta, observação participante e artefatos físicos (YIN, 2013).

A coleta de dados desta pesquisa utilizou como fonte, principalmente, entrevistas nas organizações e artefatos disponíveis nos processos de manutenção. Estas entrevistas buscaram identificar modelos específicos de manutenção, utilizado nas organizações e como a gestão de conhecimento de seus colaboradores e também dos seus clientes ajudam no processo de manutenção de software. Todos os elementos investigados estão dispostos na tabela 3.

A questão da pesquisa, delineada neste estudo de caso foi:

- **Como as organizações de Tecnologia de Informação executam seus processos de manutenção?**

A partir desta questão um conjunto de 3 hipóteses foram elaboradas. As seguintes hipóteses foram conduzidas durante o processo investigativo:

- H1) As organizações executam um processo de manutenção de software, seja ele ad-hoc ou não;
- H2) Os processos de manutenção seguem as normas vigentes ou as normas vigentes atendem à demanda da organização;
- H3) Os elementos identificados na Figura 7 e na Figura 8 estão presentes nos processos de manutenção;
- H4) Os elementos identificados nas Figuras: 6 (Processo de Manutenção), 7 (Gerenciamento do conhecimento do cliente), 8

(Gerenciamento do conhecimento interno) são essenciais para a execução da manutenção;

Após a definição da questão e hipóteses aplicadas no estudo de caso, os seguintes elementos foram investigados. A tabela 3 exibe os elementos investigados nas organizações.

Tabela 3 - Elementos Investigados

ID	Descrição
E1	Gestão de conhecimento interno
E2	Gestão de conhecimento do usuário
E3	Comunicação com usuário
E4	Processo de manutenção de software
E5	Ferramentas de *tickets*
E6	*Turnover*
E7	Gestão de configurações

Além do mapeamento do processo de software de cada caso, o conjunto de elementos presentes na tabela 3 foi investigado com o propósito de atender as hipóteses supramencionadas. Como estes elementos foram investigados, está descrito a seguir.

- **(E1) Gestão de conhecimento interno:** Objetivo é levantar informações referentes ao conhecimento da equipe de manutenção da organização de TI. São investigados o conhecimento desta equipe relacionada às regras de negócios (E1.1), aos aspectos arquitetônicos do software (E1.2) e aos

aspectos técnicos (E1.3) como linguagem de programação, banco de dados, frameworks, ambiente de execução;

- **(E2) Gestão de conhecimento do usuário:** Objetivo é levantar informações do usuário do software. Tais informações são pertinentes, uma vez que o processo de manutenção pode ser iniciado pelo usuário. Neste sentido, é investigado o conhecimento do usuário com relação ao pacote contratado (E2.1), as suas regras de negócio (E2.2) e operações do software (E2.3);

- **(E3) Comunicação com o usuário:** Objetivo é levantar as informações referentes a comunicação do analista/desenvolvedor com o cliente, quando o mesmo está atendendo uma ordem de serviço. São identificados quais papeis tem contato direto com o usuário final (E3.1) e qual a influência que este pode desencadear no processo de manutenção (E3.2);

- **(E4) Processo de manutenção de software:** Objetivo é mapear o processo de manutenção de software adotado nas organizações. Este mapeamento inclui desde o planejamento até o acompanhamento pós-entrega. Neste caso, são identificados atividades, tarefas, papéis e artefatos que compõem o processo utilizado (E4.1). Também são analisadas tarefas que incluem a priorização (E4.2), análise de impacto (E4.3), testes (E4.4) e homologação (E4.5);

- **(E5) Ferramentas de tickets:** Objetivo é levantar as informações das organizações em relação a ferramentas que auxiliem as demandas de solicitações de manutenção (ordem de serviço). Estas ferramentas comumente permitem acesso ao usuário final, que inicia o processo de

manutenção e também com a própria organização que pode utilizá-la para gerir o processo de manutenção;

- **(E6) *Turnover*:** Objetivo é levantar informações referentes ao *turnover* entre os desenvolvedores de novos produtos e desenvolvedores que efetuam operações de manutenção nas organizações e se há impacto nas atividades de manutenção.

- **(E7) Gestão de configurações:** Objetivo é levantar informações referentes ao conjunto de atividades que permitem a absorção ordenada das mudanças inerentes ao desenvolvimento de software durante a aplicação da manutenção. Este elemento investiga o tempo/complexidade despendido pelo desenvolvedor em configurar seu ambiente de trabalho para iniciar a manutenção (desenvolvimento) (E7.1), o tempo/complexidade para implantar as modificações no software em execução (disponibilizar para o usuário) (E7.2).

- **(E8) Base histórica:** Objetivo é identificar se são armazenados os registros desde o primeiro contato com o cliente (E8.1), o registro de execução da manutenção (E8.2), o registro de entrega (E8.3) até a pós-entrega (E8.4).

3.2.2 Operacionalização

Todas as informações coletadas nesse estudo de caso são confidenciais e por esse motivo não serão divulgados os nomes das organizações, as mesmas serão caracterizadas nesse estudo como organização A, B, C, D e E.

A organização A, é uma fábrica de software focada em desenvolvimento de softwares para empresas comerciais, localizada no Sul do Brasil. Possui certificação CMMI nível 3, na produção de software, buscando sempre atingir a padronização e qualidade dos seus produtos de softwares. É composta por uma equipe com quarenta e cinco profissionais das áreas de engenharia de software.

Esta organização tem seu próprio processo para realização dessa atividade. Atualmente a equipe de desenvolvimento é a mesma que realiza manutenção nos produtos de softwares desenvolvidos e qualquer integrante da equipe está apto para realizar a manutenção em um produto de software, uma vez que todos os projetos são documentados. O profissional que realiza a manutenção tem contato direto com o usuário para esclarecimento de dúvidas que venha surgir nesse processo.

Há um processo gestão de conhecimentos de todos os integrantes da equipe desenvolvimento/manutenção, o que torna esse processo muito ágil, a organização se preocupa para que todos os profissionais tenham o mesmo nível de conhecimento em seus produtos, regras de negócio, linguagens de programação, banco de dados, e em todo planejamento de projeto há um tempo estimado para que todos os profissionais se aperfeiçoarem.

O processo de manutenção é realizado pelo *Help Desk* da empresa, onde são definidos os ciclos de atendimento aos chamados abertos pelos clientes, por criticidade e projetos e o prazo para atendimentos destes. As manutenções mais realizadas são a corretiva e a adaptativa. Após essa triagem, a manutenção entra no ciclo normal de produção.

A organização B é uma secretaria de tecnologia de informação localizada na região centro oeste do Brasil. É uma organização governamental, composta por uma equipe de dezessete profissionais das áreas de Engenharia de software, que desenvolvem software para rede educacional do estado. A organização tem seu próprio processo para realização da atividade de manutenção de software. Nesse processo existente, dependendo do projeto há um profissional específico para a realização da manutenção de software na equipe. Não existe uma rotatividade significativa de profissionais na área, qualquer profissional consegue realizar manutenção de software nos produtos existentes. O profissional que realiza a manutenção tem contato direto com o usuário para esclarecimento de dúvidas que venha surgir nesse processo e se tratar de uma manutenção perfectiva, podem ser agendadas reuniões entre as áreas de tecnologia e educação. A documentação dos processos de software desenvolvidos são armazenados em um sistema de gestão de projetos que mantém um histórico de tudo que foi alterado. Na organização não há um processo para desenvolvimento de software formalizado, e nem todos os integrantes da equipe tem conhecimento nas linguagens de programação utilizadas, banco de dados e etc.

Todo processo de manutenção de software é gerenciado e documentado pelo *Help Desk* que define os ciclos de atendimento aos chamados abertos pelos clientes, por criticidade e projetos e o prazo para atendimentos destes. As manutenções mais realizadas são a corretiva e a adaptativa.

A organização C é uma fábrica de software focada em desenvolvimento de softwares comerciais, localizada no interior do estado de São Paulo. É composta por uma equipe de quarenta colaboradores na área de Engenheira de software. A organização tem seu próprio processo para realização da atividade de manutenção de software.

A equipe que realiza o desenvolvimento dos produtos de software é a mesma que realiza a manutenção. O profissional que realiza a manutenção tem relacionamento direto com o cliente que solicitou a manutenção. Na organização não há um processo para desenvolvimento de software, e nem todos os integrantes da equipe tem conhecimento nas linguagens de programação utilizadas, banco de

dados e não existe documentação de apoio para auxiliar no suporte de uma manutenção de software.

As atividades do processo de manutenção de software são registradas pelo *Help Desk* da empresa, esse por sua vez auxilia no processo de abertura de solicitações dos clientes. As manutenções mais realizadas são a corretiva e a adaptativa.

A organização D é um setor de tecnologia da informação de uma universidade no interior do estado de São Paulo. Atualmente esse setor desenvolve e realiza suporte a todos os sistemas disponíveis na universidade: Área acadêmica, biblioteca, financeiro, jurídico e recursos humanos. É composta por uma equipe de cinco colaboradores na área de desenvolvimento de software. A organização tem seu próprio processo para realização da atividade de manutenção de software. Nesse processo existente, a equipe de desenvolvimento é a mesma que realiza a manutenção de software. E nem todos os integrantes conseguem realizar manutenção em um produto específico.

Na organização não há um processo para desenvolvimento de software, e nem todos os integrantes da equipe tem conhecimento nas linguagens de programação utilizadas, banco de dados e não existe documentação de apoio para auxiliar no suporte de uma manutenção de software.

As solicitações de manutenção de software chegam à equipe por meio de ligações ao próprio setor de tecnologia da informação ou via e-mail, e não há registros de demandas atendidas, concluídas ou em andamento. As maiorias das solicitações são de manutenção corretiva.

A organização E é uma organização brasileira de software para agroindústria, localizada no estado de São Paulo. Atualmente desenvolve sistemas de gestão para área de agronegócio. A unidade investigada é composta por uma equipe de quarenta colaboradores na área de Desenvolvimento de software.

A organização tem seu próprio processo para realização da atividade de manutenção de software. Nesse processo existente, a equipe de desenvolvimento é a mesma que realiza a manutenção de software. E nem todos os integrantes

conseguem realizar manutenção em um produto específico, isso porque os times são divididos por módulos de produtos e cada time atende os seus módulos. E mesmo dentro dos módulos, os mais inexperientes precisam de apoio para realizar a manutenção. Dentro de cada módulo todos desenvolvem e realizam a manutenção. Esporadicamente um integrante de um time pode ser emprestado para outro, onde existe uma rotatividade.

Na organização não há um padrão para desenvolvimento de software, e nem todos os integrantes da equipe tem conhecimento nas linguagens de programação utilizadas, banco de dados. As solicitações de manutenção de software chegam à equipe por meio de abertura de chamados, pelo sistema de *Help Desk* da empresa. Na maioria das vezes o analista que atende a solicitação de manutenção, não possui contato com o cliente, já que o contato é realizado pela equipe de atendimento. Em algumas situações o desenvolvedor entra em contato com o usuário para maiores esclarecimentos. As maiorias das solicitações são de manutenção corretiva e adaptativa.

4. ANÁLISE E RESULTADOS

Este capítulo apresenta a análise e resultados obtidos nos experimentos e no estudo de caso, as ameaças de validade desses estudos, e o modelo de manutenção de software com o objetivo de validar a proposta desse trabalho.

4.1 ANÁLISE DOS RESULTADOS - EXPERIMENTOS

Após a execução dos experimentos, foi realizada uma análise nos 20 questionários aplicados aos alunos participantes, os mesmos foram tabulados em duas planilhas para que seus resultados pudessem ser analisados. A primeira planilha caracteriza as respostas obtidas em porcentagem de todos os questionários respondidos pelo curso de Graduação. A segunda planilha caracteriza as respostas obtidas em porcentagem de todos os questionários respondidos pelo programa de Mestrado.

De posse das planilhas, os autores prepararam quatro gráficos. Planilhas e gráficos podem ser obtidos através do link https://goo.gl/9r4soK.

As opções para respostas às perguntas que do questionário são caracterizados pela Escala *Likert* (LIKERT, 1932)(1 – Discordo Plenamente, 2 – Discordo Parcialmente, 3 - Indiferente, 4 – Concordo Parcialmente e 5 - Concordo Plenamente).

4.1.1 Resultados dos dois experimentos aplicados na graduação

No nível de graduação, dois experimentos foram realizados no último módulo do curso, com um total de 6 alunos. O primeiro experimento continham 2 alunos, apresentados no trabalho como Grupo 1 e o segundo experimento continham 4 alunos, apresentados no trabalho como Grupo 2. Ambos tem a mesma experiência em Engenharia de Software (Menos de 1 ano) conforme demonstrado no gráfico 1.

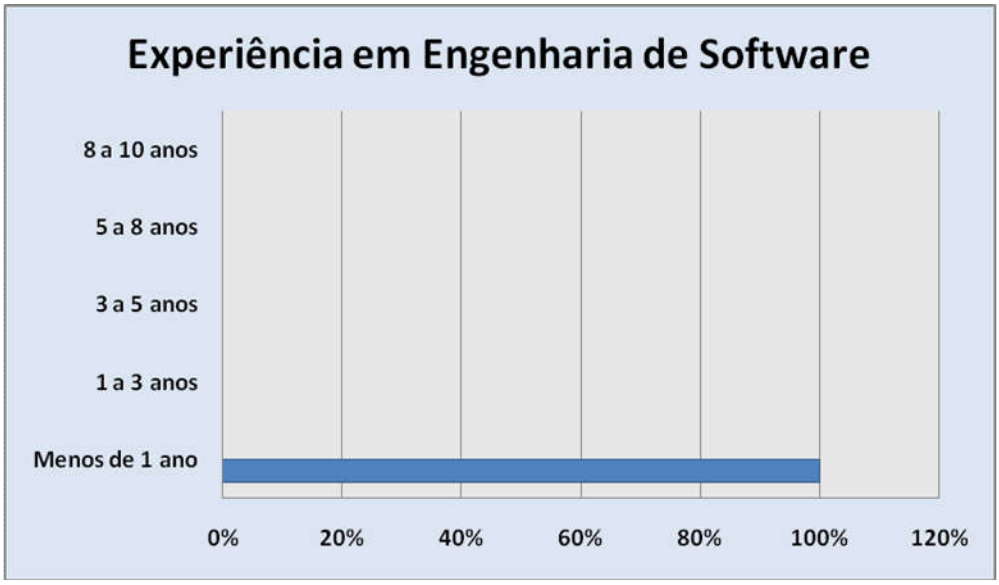

Gráfico 1 - Experiência em Engenharia de Software - Graduação

Analisando o gráfico 2, é possível notar que as legendas Q1 a Q21 do eixo X representam as perguntas respondidas pelos alunos da graduação no questionário de manutenção do software. No eixo Y são apresentadas as porcentagens totais de cada pergunta respondida por todos os alunos participantes, ambas baseadas na escala *Likert*.

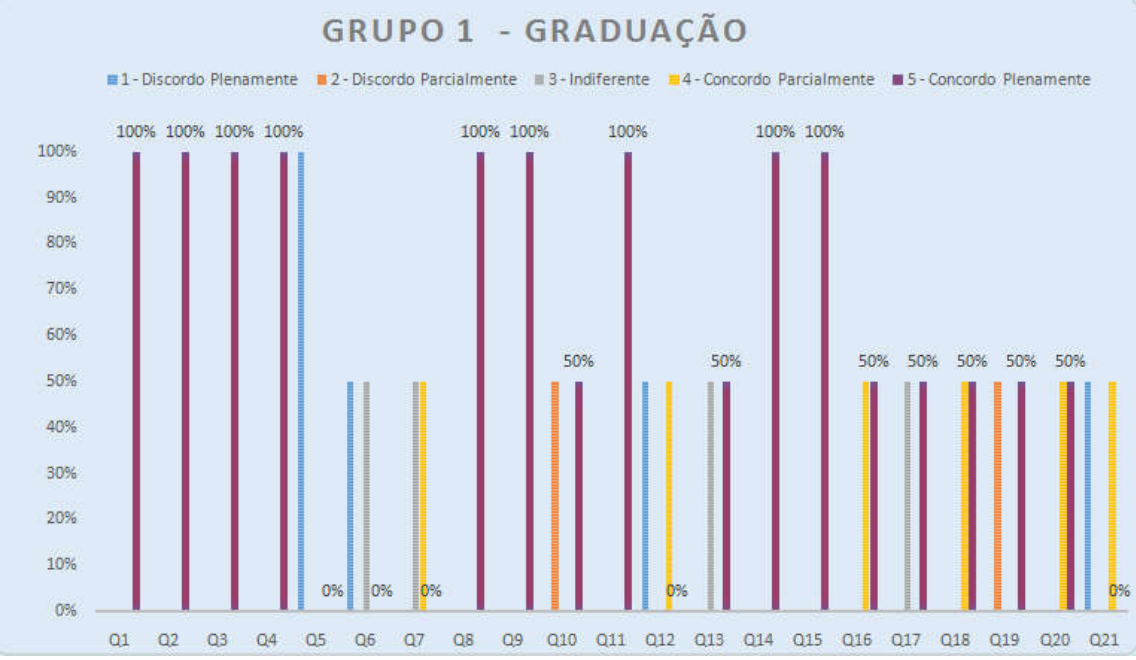

Gráfico 2 – Grupo 1 - Análise de respostas da graduação

Na análise do grupo 1, conclui-se que na questão 1 (Q1), que todos os alunos participantes concordaram plenamente que a realização da manutenção foi uma tarefa simples.

Na (Q2) todos os alunos participantes concordaram plenamente que a equipe teve a capacidade de realizar manutenção e capacidade de atender o usuário (Q3).

Em relação ao tempo ser suficiente para realizar as atividades de manutenção (Q4), todos concordaram plenamente que foi suficiente.

Em relação à questão (Q5), todos discordaram plenamente de que não havia necessidade de realizar treinamento com o colega para ajudar na manutenção informada.

Em relação à (Q6), 50% discordaram plenamente de que nenhuma pessoa foi definida para realizar aceitação / revisão de manutenção e 50% acharam indiferente essa definição.

Em relação à importância desse papel (Q7), 50% concordaram parcialmente e 50% acharam indiferente ter alguém para desempenhar esse papel.

Todos os alunos concordaram plenamente que o problema de manutenção foi identificado (Q8).

Todos os alunos concordaram plenamente que é importante conhecer o produto antes de realizar a manutenção (Q9).

Quanto ao conhecimento do usuário ser necessário para manutenção do produto (Q10), 50% concordaram plenamente e 50% discordaram parcialmente.

No que se refere a considerar importante conhecer a regra de negócio do produto, antes de realizar a manutenção (Q11), todos concordaram plenamente que é necessário.

Quanto à dificuldade na configuração do ambiente de desenvolvimento para realizar a manutenção solicitada (Q12), 50% concordaram plenamente e 50% acharam indiferente.

Sobre o tempo para iniciar o desenvolvimento de a manutenção ter sido adequado (Q13), 50% concordaram plenamente e 50% acharam indiferente.

Todos concordaram plenamente que não levaram muito tempo a conhecer o produto (Q14).

Todos concordaram plenamente que realizaram os processos de manutenção conforme estipulado (Q15).

Em relação à importância de conhecer a arquitetura do software antes de realizar a manutenção (Q16), 50% concordaram plenamente que é importante e 50% concordaram parcialmente.

Sobre a importância de conhecer as bibliotecas utilizadas pelo produto (Q17), 50% concordaram plenamente que é interessante e 50% acharam indiferente esse conhecimento.

Sobre o conhecimento das regras do negócio do produto, antes de realizar a manutenção (Q18), 50% concordaram parcialmente e 50% concordam plenamente que é importante esse conhecimento.

Quanto à importância de conhecer o IDE antes de realizar a manutenção (Q19), 50% concordaram plenamente e 50% não concordaram parcialmente.

Em relação à identificação de problemas pela ordem de serviço (Q20), 50% concordaram parcialmente e 50% concordaram plenamente que foi possível essa identificação.

Sobre a necessidade de entrar em contato com o usuário que abriu a ordem de serviço para esclarecimentos (Q21), 50% concordaram parcialmente que tiveram que entrar em contato com o usuário e 50% discordaram parcialmente que não foi necessário esse contato.

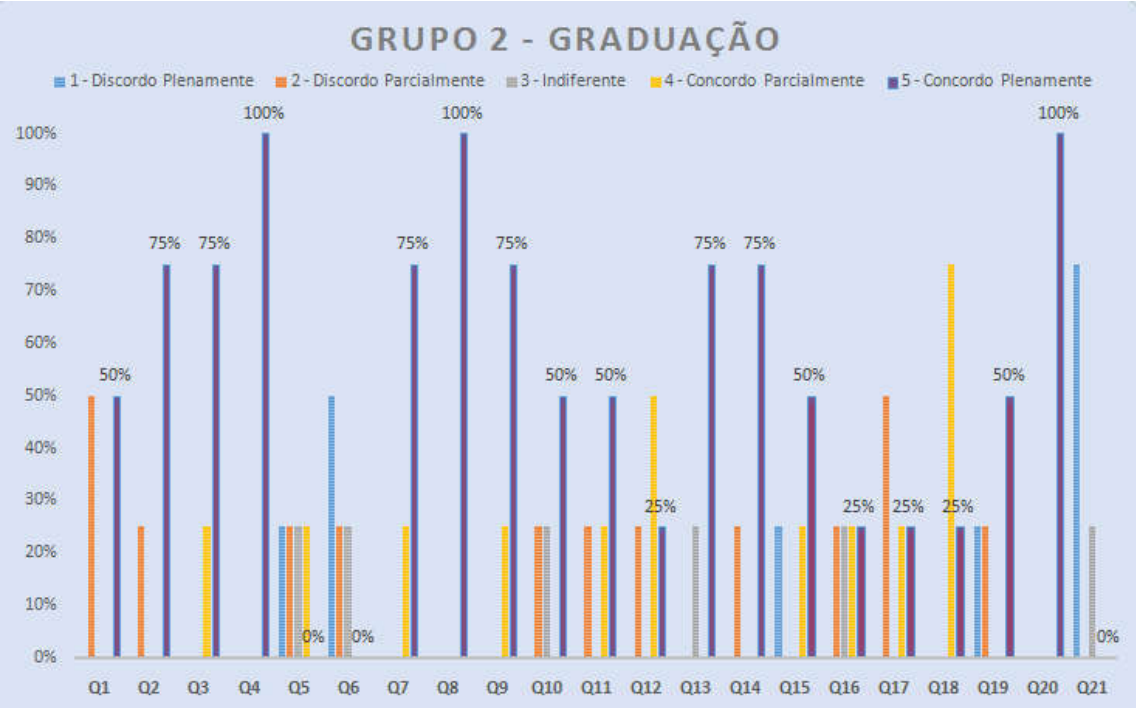

Gráfico 3 - Grupo 2 - Análise de respostas da graduação

Na análise do Grupo 2, demonstrado no Gráfico 3 conclui-se que na questão 1 (Q1), 50% dos alunos participantes concordaram plenamente que a manutenção foi uma tarefa simples e 50% discordaram parcialmente.

Analisando a questão (Q2) 75% dos alunos concordaram plenamente que os participantes tiveram capacidade de realizar manutenção e 25% discordaram parcialmente.

Quanto à capacidade de atender o usuário (Q3) 75% concordaram plenamente e 25% concordaram parcialmente.

Sobre o tempo ser suficiente para realizar as atividades de manutenção (Q4), todos concordaram planamente que foi suficiente.

Quanto à necessidade de ter um treinamento com um colega (Q5), 25% discordaram plenamente, 25% discordaram parcialmente, 25% acharam indiferente e 25% concordaram parcialmente sobre essa necessidade.

Em relação à (Q6), 50% discordaram plenamente de que não foi definida uma pessoa para realizar aceitação / revisão de manutenção, 25% acharam indiferente essa definição, 25% discordaram parcialmente sobre essa definição.

Em relação à importância desse papel (Q7), 75% concordaram plenamente e 25% concordaram parcialmente em ter alguém para desempenhar esse papel.

Todos os alunos concordaram plenamente que o problema de manutenção foi identificado (Q8).

Em relação ao conhecimento do produto 75% concordaram plenamente que é importante conhecer o produto para realizar a manutenção e 25% concordaram parcialmente (Q9).

Quanto ao conhecimento do usuário ser necessário para manutenção do produto (Q10), 50% concordaram plenamente, 25% concordaram parcialmente e 25% acharam indiferente.

No que se refere a considerar importante conhecer a regra do negócio do produto, antes de realizar a manutenção (Q11), 50% concordaram plenamente, 25% concordaram parcialmente e 25% discordaram parcialmente.

Quanto à dificuldade em configurar o ambiente para realizar a manutenção solicitada (Q12), 50% concordaram parcialmente, 25% concordaram plenamente e 25% discordaram parcialmente.

Sobre o tempo para iniciar o desenvolvimento de manutenção ter sido adequado (Q13), 75% concordaram plenamente e 25% o acharam indiferente.

Quanto ao tempo de conhecimento do produto antes de realizar a manutenção (Q14), 75% concordaram plenamente e 25% discordaram parcialmente, que não gastaram muito tempo.

Em relação à execução dos procedimentos de manutenção 75% concordaram plenamente que realizaram procedimentos de manutenção conforme estipulado (Q15) e 25% concordaram parcialmente.

Em relação à importância de conhecer a arquitetura de software antes de realizar a manutenção (Q16), 25% concordaram plenamente, 25% concordaram parcialmente, 25% acharam indiferente e 25% discordaram parcialmente.

Em relação ao conhecimento das bibliotecas utilizadas pelo produto antes de realizar a manutenção, 50% discordaram parcialmente, 25% concordaram plenamente e 25% concordaram parcialmente (Q17).

Sobre o conhecimento das regras do negócio do produto, antes de realizar a manutenção (Q18), 75% concordaram parcialmente e 25% concordaram totalmente e 25% discordaram parcialmente.

Quanto à importância de conhecer o IDE antes de realizar a manutenção (Q19), 50% concordaram plenamente, 25% discordaram parcialmente sobre essa importância.

Na identificação do problema através da ordem de serviço (Q20), todos concordaram plenamente que foi possível essa identificação.

Os alunos discordaram plenamente em 75% que foi preciso entrar em contato com o usuário que abriu a ordem de serviço de manutenção (Q21) e 25% acharam indiferente esse contato.

4.1.2 Resultados dos dois experimentos aplicados no mestrado

No programa de mestrado foram realizados dois experimentos, com um total de 14 alunos. O primeiro experimento continham 4 alunos, definidos no trabalho como Grupo 3 e o segundo experimento 10 alunos, definidos no trabalho como Grupo 4. Ambos tiveram o tempo de experiência em engenharia de software distinta, conforme mostrada abaixo no gráfico 4:

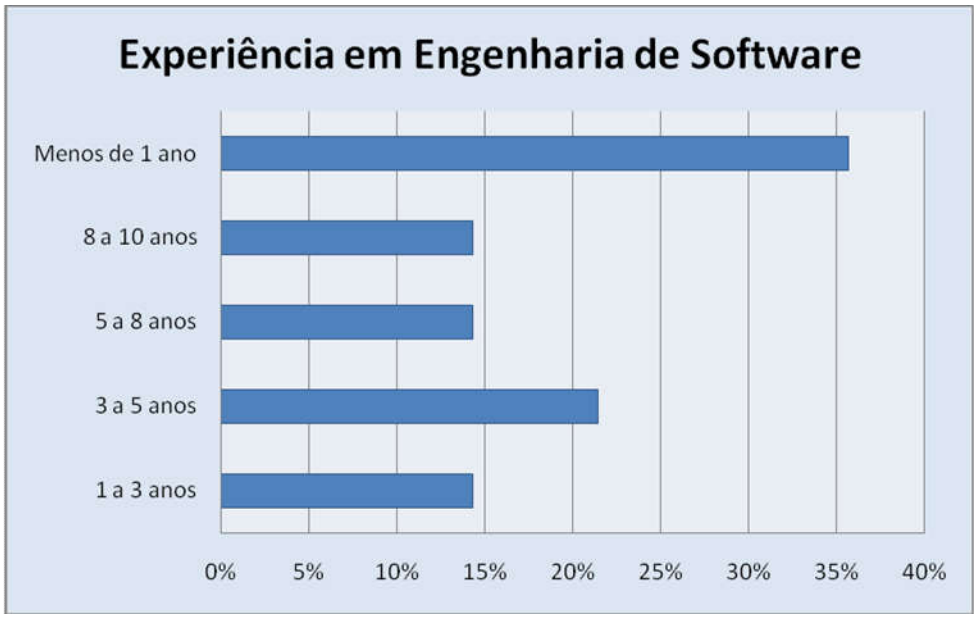

Gráfico 4 - Experiência em Engenharia de Software – Mestrado

Na análise do Grupo 3 (Gráfico 5), concluiu-se que a questão 1 (Q1), 50% dos alunos participantes discordaram parcialmente de que a manutenção era uma tarefa simples, 25% concordaram plenamente e 25% concordaram parcialmente.

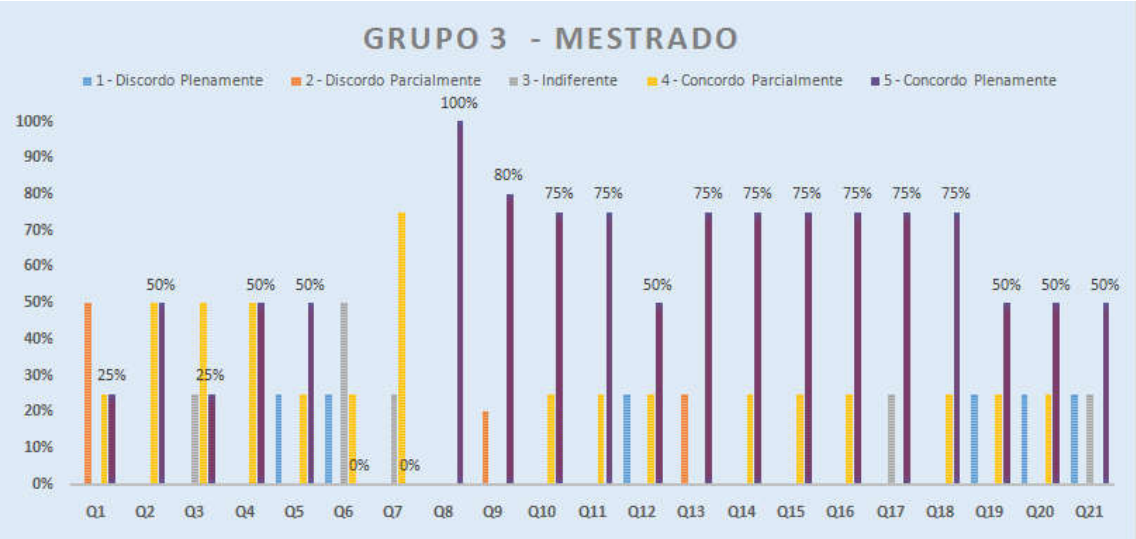

Gráfico 5 - Grupo 3 – Análise das respostas do mestrado

Em relação à habilidade da equipe em realizar a manutenção (Q2), 50% dos alunos participantes concordaram que a equipe teve a capacidade de realizar manutenção e 50% concordaram parcialmente.

Quanto à capacidade de atender o usuário (Q3), 50% concordaram parcialmente, 25% concordaram plenamente e 25% o acharam indiferente que a equipe teve essa capacidade em relação ao atendimento ao usuário.

Quanto ao tempo ser suficiente para realizar a atividade de manutenção (Q4), 50% concordaram plenamente, 50% concordaram parcialmente que foi suficiente esse tempo.

Sobre a realização de um treinamento com um colega (Q5), 50% concordaram plenamente, 25% concordaram parcialmente e 25% discordaram plenamente de que não havia necessidade de treinar com o colega por manutenção informada.

Quanto ao tempo ter sido suficiente para realizar as atividades de manutenção (Q4), 60% concordaram plenamente que foi suficiente, 20% concordaram parcialmente, 10% discordaram parcialmente e 10% discordaram plenamente.

Em relação à definição de uma pessoa para realizar aceitação/ revisão da manutenção (Q6), 50% acharam indiferente essa definição, 25% concordaram parcialmente e 25% discordaram plenamente que é interessante essa definição.

Em relação a este papel (Q7), 75% concordaram parcialmente e 25% acharam indiferente ter alguém para desempenhar esse papel.

Todos os alunos concordaram plenamente que o problema de manutenção foi identificado (Q8).

Sobre o conhecimento ao produto antes de realizar a manutenção 80% concordaram plenamente que é importante conhecer o produto para realizar a manutenção (Q9) e 20% discordaram parcialmente.

Quanto ao conhecimento do usuário ser necessário para a manutenção do produto (Q10), 75% concordaram plenamente e 25% concordaram parcialmente.

No que se refere a considerar importante conhecer a regra do negócio do produto, antes de realizar a manutenção (Q11), 75% concordaram plenamente e 25% concordaram parcialmente.

Quanto à dificuldade em configurar o ambiente para realizar a manutenção solicitada (Q12), 50% concordaram plenamente e 50% concordaram parcialmente, que tiveram essa dificuldade.

Em relação ao tempo para iniciar o desenvolvimento da manutenção ter sido adequado (Q13), 75% concordaram plenamente e 25% concordaram parcialmente, que foi adequado.

Quanto ao tempo para conhecimento do produto (Q14), 75% concordaram plenamente que não gastaram muito tempo e 25% concordaram parcialmente.

Sobre execução dos procedimentos de manutenção, 75% concordaram plenamente que realizaram procedimentos de manutenção conforme estipulados (Q15) e 25% concordaram parcialmente.

Em relação à importância de conhecer a arquitetura do software antes de realizar a manutenção (Q16), 75% concordaram plenamente e 25% concordaram parcialmente.

Sobre a importância de conhecer as bibliotecas utilizadas no produto 75% concordaram plenamente e 25% acharam indiferente o conhecimento das bibliotecas usadas pelo produto antes de realizar a manutenção (Q17).

Sobre o conhecimento das regras do negócio do produto, antes de realizar a manutenção (Q18), 75% concordaram plenamente e 25% concordaram parcialmente.

Quanto à importância de conhecer o IDE antes de realizar a manutenção (Q19), 50% concordaram plenamente, 25% concordaram parcialmente e 25% discordaram plenamente.

Em relação à identificação de problemas pela ordem de serviço (Q20), 50% concordaram plenamente, 25% concordaram parcialmente e 25% discordaram plenamente, que foi possível identificar.

E os alunos concordaram parcialmente em 50%, que eles tiveram que entrar em contato com o usuário que abriu a ordem de serviço (Q21) e 25% discordaram plenamente de que não precisou entrar em contato e 25% acharam indiferente esse contato com o usuário.

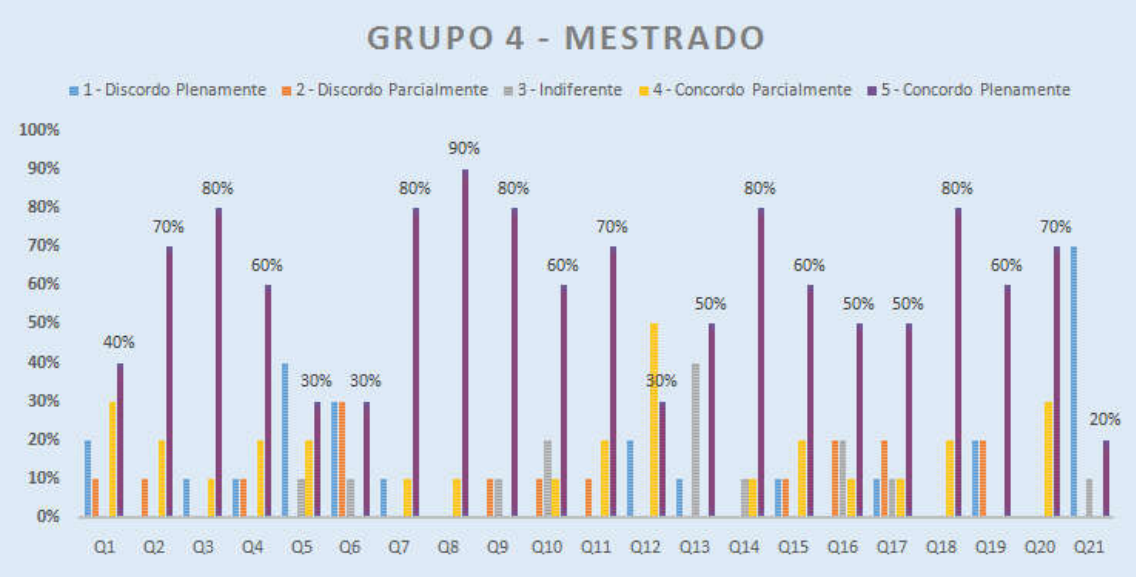

Gráfico 6 - Grupo 4 – Análise das respostas do mestrado

Na análise do Grupo 4 (Gráfico 6), concluiu-se que na questão 1 (Q1), que 40% dos alunos participantes concordaram plenamente que a manutenção realizada foi uma tarefa simples, 30% concordaram parcialmente, 20% discordaram plenamente e 10% acharam indiferente.

Em relação à habilidade da equipe em realizar a manutenção (Q2), 70% dos alunos participantes concordaram plenamente que a equipe teve a capacidade de realizar manutenção, e 20% concordaram parcialmente, 10% discordaram parcialmente.

Quanto à capacidade de atender o usuário (Q3), 80% concordaram plenamente, 10% concordaram parcialmente e 10% discordaram plenamente, que a equipe teve essa capacidade em relação ao atendimento ao usuário.

Quanto ao tempo ter sido suficiente para realizar as atividades de manutenção (Q4), 60% concordaram plenamente que foi suficiente, 20% concordaram parcialmente, 10% discordaram parcialmente e 10% discordaram plenamente.

Sobre a realização de um treinamento com um colega (Q5), 40% discordaram plenamente, 30% concordaram plenamente, 20% concordaram

parcialmente e 10% acharam indiferente de que não havia necessidade de treinar com o colega por manutenção informada.

Em relação à definição de uma pessoa para realizar aceitação/ revisão da manutenção (Q6), 30% concordaram plenamente com essa definição, 30 % discordaram parcialmente, 30% discordaram plenamente e 10% acharam indiferente que é interessante essa definição.

Em relação à importância desse papel (Q7), 80% concordaram plenamente, 10% concordaram parcialmente e 10% acharam indiferente em ter alguém para desempenhar esse papel.

Os alunos concordaram plenamente em 90%, que o problema de manutenção foi identificado (Q8), e 10% concordaram parcialmente.

Quanto à importância de conhecer o produto antes de realizar a manutenção (Q9), 80% concordaram plenamente, 10% acharam indiferente esse conhecimento e 10% discordaram parcialmente.

Em relação ao conhecimento do usuário ser necessário para a manutenção do produto (Q10), 60% concordaram plenamente, 20% acharam indiferente, 10% concordaram parcialmente e 10% discordaram parcialmente.

No que diz respeito a considerar importante conhecer a regra do negócio do produto, antes de realizar a manutenção (Q11), 70% concordaram plenamente, 20% concordaram parcialmente e 10% discordaram parcialmente.

Quanto à dificuldade em configurar o ambiente para realizar a manutenção solicitada (Q12), 50% concordaram parcialmente que tiveram essa dificuldade, 30% concordaram plenamente e 20% discordaram plenamente.

Sobre o tempo para iniciar o desenvolvimento da manutenção ter sido suficiente (Q13), 50% concordaram plenamente, 40% acharam indiferente e 10% discordaram plenamente que foi suficiente.

Quanto ao tempo para conhecimento do produto (Q14), 80% concordaram plenamente que não gastaram muito tempo 10% concordaram parcialmente e 10% acharam indiferente.

No que diz respeito aos processos de manutenção ter sido realizado conforme estipulado (Q15), 60% concordaram plenamente, 20% concordaram parcialmente, 10% discordaram parcialmente e 10% discordaram plenamente.

Quanto à importância de conhecer a arquitetura de software antes de realizar a manutenção (Q16), 50% concordaram plenamente, 20% acharam indiferente, 20% discordaram parcialmente e 10% concordaram parcialmente.

No que diz respeito ao conhecimento das bibliotecas utilizadas pelo produto, antes de realizar a manutenção (Q17), 50% concordaram plenamente que é importante esse conhecimento, 20% discordaram parcialmente, 10% concordaram parcialmente, 10% acharam indiferente e 10% discordam plenamente.

Sobre o conhecimento das regras do negócio do produto, antes de realizar a manutenção (Q18), 80% concordaram plenamente e 20% concordaram parcialmente.

Quanto à importância de conhecer o IDE antes de realizar a manutenção (Q19), 60% concordaram plenamente, 20% discordaram parcialmente e 20% discordaram plenamente.

No que diz respeito à identificação de problemas através da ordem de serviço (Q20), 70% concordaram plenamente, 30% concordaram parcialmente.

E os alunos discordaram plenamente em 70%, que eles tiveram que entrar em contato com o usuário que abriu a ordem de serviço (Q21), 20% concordaram plenamente de que não precisou entrar em contato e 10% acharam indiferente esse contato com o usuário.

Após a análise da forma individual por grupo / experimento, a tabela 4 abaixo mostra a média de respostas com base na escala *Likert*, onde a maioria das respostas foi positiva após a execução do experimento.

Tabela 4 - Análise geral dos experimentos

	Graduação	**Mestrado**
Q1	5	2
Q2	5	5
Q3	5	5
Q4	5	5
Q5	1	5
Q6	1	3
Q7	4	5
Q8	5	5
Q9	5	5
Q10	5	5
Q11	5	5
Q12	4	4
Q13	5	5
Q14	5	5
Q15	5	5
Q16	4	5
Q17	5	5
Q18	4	5
Q19	5	5
Q20	5	5
Q21	1	1

Os resultados do ponto de vista do ensino foram relatados no FIE2016.

Após a aplicação do experimento o processo de manutenção de software preliminar foi avaliado e foram constatados as seguintes melhorias relacionadas a gestão de conhecimento dos desenvolvedores, tempo de setup, gestão de conhecimento do usuário.

4.1.3 Ameaças a validade – Experimentos

Segundo Wholim et al. (2012), um estudo experimental está sujeito a situações que podem ameaçar a validade dos resultados obtidos a partir deste estudo. As principais ameaças tratadas neste estudo são as ameaças à validade Interna e externa.

Esse tipo de ameaça refere-se às questões que afetam a habilidade de assegurar que os resultados não foram obtidos em decorrência de uma coincidência e que os mesmos podem ser generalizados para um contexto mais amplo daquele selecionado para o estudo.

Sendo assim, os fatores importantes que podem ter influenciado nos resultados deste experimento são: (i) o curto espaço de tempo em que a tarefa de manutenção de software ficou sob avaliação; (ii) a experiência dos alunos em relação ao robô *Lejos*. Contudo, não foram demonstradas expectativas a favor ou contra a tarefa sob análise, para que os participantes não fossem influenciados; e (iii) a falta de clareza das afirmações dos questionários apresentados aos avaliadores, que pode ter gerado interpretações ambíguas. Com o intuito de mitigar essas possíveis ameaças, pretende-se replicar tal experimento com um grupo de participantes maior e também em um espaço de tempo maior.

4.2 ANÁLISE DOS RESULTADOS – ESTUDO DE CASO

Na execução dos estudos de caso os autores coletaram as informações das entrevistas e também dos artefatos que tiveram acesso.

Após a execução dos casos, os dados foram tabulados, a tabela 5 descreve os casos analisados.

Tabela 5 - Descrição dos Casos

Caso	Colaboradores	Certificação	Modelo de Processo
A	40	CMMI – Level 3	Interativo/Incremental
B	17	Inexistente	Interativo/Incremental
C	40	Inexistente	Inexistente
D	5	Inexistente	Inexistente
E	40	Inexistente	Interativo/Incremental

A tabela 6 apresenta a tabulação dos dados obtidos a partir dos estudos de casos, comparados aos elementos identificados na tabela 3.

Tabela 6 - Elementos identificados nos estudos de casos

ID	A	B	C	D	E
E1	Existente	Existente	Existente	Inexistente	Existente
E1.1	Sim	Sim	Sim	Não	Sim
E1.2	Sim	Sim	Sim	Não	Sim
E1.3	Sim	Sim	Sim	Não	Sim
E2	Existente	Existente	Inexistente	Inexistente	Existente
E2.1	Sim	Sim	Não	Não	Sim
E2.2	Sim	Não	Não	Não	Sim
E2.3	Sim	Não	Sim	Não	Sim
E2.4	Sim	Sim	Sim	Sim	Sim
E3	Existente	Existente	Existente	Existente	Existente
E3.1	Sim	Sim	Sim	Sim	Sim
E3.2	Sim	Sim	Sim	Sim	Sim

E4	Existente	Existente	Inexistente	Inexistente	Existente
E4.1	Sim	Sim	Parcialmente	Não	Sim
E4.2	Sim	Sim	Sim	Sim	Sim
E4.3	Sim	Sim	Não	Não	Não
E4.4	Sim	Sim	Não	Não	Não
E4.5	Sim	Sim	Não	Não	Sim
E4.6	Sim	Sim	Não	Não	Sim
E5	Existente	Existente	Existente	Inexistente	Existente
E5.1	Sim	Sim	Sim	Não	Sim
E6	Inexistente	Inexistente	Existente	Inexistente	Inexistente
E6.1	Sim	Sim	Sim	Sim	Sim
E6.2	Não	Sim	Sim	Sim	Sim
E6.3	Não	Não	Não	Não	Não
E7	Existente	Existente	Inexistente	Inexistente	Existente
E7.1	Baixo	Baixo	Baixo	Baixo	Baixo
E7.2	Médio	Médio	Médio	Médio	Médio

Analisando o elemento 1 (E1), cujo o objetivo era levantar informações sobre os aspectos de conhecimentos técnicos das equipes que realizam a atividade de manutenção, somente na organização D esse elemento é inexistente. Foi observado que o elemento (E1.1), nas organizações A, B, C e E é existente, os integrantes das equipes que realizam a manutenção tem conhecimento as regras de negócios do software adquirido, o que torna a resolução da manutenção muito mais fácil, ou seja, evitando uma perda de tempo para entendimento do software, na organização D esse elemento é inexistente, pois esse conhecimento fica muito centrado a apenas um integrante da equipe. Em relação aos aspectos arquitetônicos (E1.2), somente nas organizações A, B, C e E esse conhecimento é existente em

relação a arquitetura do software, na organização D, somente um integrante possui habilidade em relação a arquitetura os demais integrantes da equipe não possuem. Em relação a (E1.3), somente na organização D é inexistente, pois nem todos os integrantes tem conhecimento técnico nas linguagens em que os produtos de software foram desenvolvidos para realizar uma devida manutenção, nas demais organizações esse elemento é existente.

Em relação ao elemento 2 (E2.1), onde o objetivo era levantar informações sobre o conhecimento do usuário ao pacote de serviços contratado, somente nas organizações A, B e E há um entendimento melhor do usuário em relação ao contrato do pacote de serviços, na organização B por ser um setor publico, não existe um contrato de prestação de serviços da mesma forma ocorre na organização D. Foi observado que o elemento (E2.2), é existente somente nas organizações A e E, os usuários (cliente) tem conhecimento as regras de negócios do software adquirido, o que torna a resolução da manutenção muito mais fácil, ou seja, há um entendimento dessa regra por parte do cliente, evitando aberturas de ordens de serviços desnecessárias, nas organizações B,C e D esse elemento é inexistente, pois não há um controle de quem realiza a abertura da ordem de serviço, muitas vezes, a pessoa que realiza a abertura dessa ordem não tem conhecimento das regras de negócios do software e solicita mudanças que não são condizentes com o produto. Em relação às operações do software (E2.3), somente nas organizações A, C e E esse conhecimento é existente, nas organizações B e D, foi identificado uma certa dificuldade dos usuários.

Em nível de comunicação com o usuário (E3), todas as organizações afirmam que esse contato com usuário é existente e que são de ambos os papéis (E3.1), nas organizações A, B é realizado pelo mantenedor e nas demais organizações pelo próprio desenvolvedor. Foi observado nos estudos que o contato facilita a resolução da manutenção (E3.2), pois muita das vezes a ordem de serviço não é clara, e gera dúvidas em relação à operacionalidade do sistema em chegar ao descrito na ordem de serviço (manutenção corretiva), esse contato é muito presente nas organizações, seja ele via *Help Desk*, telefone ou até mesmo e-mail.

Sobre o processo de manutenção de *software* (E4), nenhuma das organizações seguem atividades propostas pela norma ISO/IEC 14764, todas possuem um processo para executar as manutenções, na organização A o processo de manutenção (E4. 1), é realizado pela equipe de desenvolvimento e qualquer integrante (Desenvolvedor/Mantenedor) consegue realizar a manutenção. Na organização B e D a manutenção também é realizada pelo time de desenvolvimento, dependendo do projeto de software, há um profissional especifico somente para a realização da manutenção. Nas organizações C e E esse processo é realizado apenas pelo time de desenvolvimento, não havendo um profissional especifico para realizar a manutenção. Em relação à priorização das tarefas de manutenção (E4.2), somente nas organizações A, B, e E são existentes esse controle, ou seja, as manutenções corretivas/adaptativas são atendidas com uma certa priorização as demais organizações o processo de priorização é inexistente. A análise de impacto (E4.3) dessas manutenções foi observada somente nas organizações A e B, assim como os testes (E4.4), nas demais organizações não foram identificadas essas atividades antes da entrega da manutenção, foi observado que as organizações C, D e E não possuem um profissional para essa atividade, e o testes são realizados somente pelo desenvolvedor. A homologação da versão após a manutenção (E4.5), foi identificada somente nas organizações A, B e E nas demais organizações a nova versão é colocada diretamente em produção.

Em relação ao elemento de ferramenta de *tickets* (E5), somente a organização D não possui um controle de ordem de serviço. As demais organizações possuem alguma ferramenta que gerencia essa abertura de chamados, e muita das vezes a documentação e históricos de manutenções ficam armazenados lá.

O *Turnover* (E6), foi identificado somente na organização C, e de acordo com a organização, ele afeta muito a manutenção de *software*, pois em alguns casos, o conhecimento de regras de negócios de um software, fica centralizado somente a um usuário, o que pode trazer problemas as organizações, nas demais organizações esse problema não foi identificado.

A Gestão de configurações (E7), investiga o tempo/complexidade despendido pelo desenvolvedor em configurar seu ambiente de trabalho para iniciar a manutenção (E7. 1), na organização A o tempo é de 40 minutos aproximadamente para configuração, na organização B o tempo para configuração do ambiente de trabalho é de 1 hora, as organizações C, D e E não souberam mensurar esse tempo. Em relação ao tempo implantar as modificações do software em execução (E7.2), a organização A informou que são 2 dias o prazo máximo, quando se trata de manutenção corretiva, manutenções adaptativas depende do contrato com o cliente, a organização B informou que o prazo para manutenção corretiva são de até 4 dias, dependendo da demanda, manutenção adaptativa não tem uma estimativa de prazo. A organização C, as manutenções corretivas são realizadas em até 5 dias, dependendo da sua criticidade. Na organização D, o prazo depende das atividades do desenvolvedor, se for algo muito urgente pode ser realizada em até 5 dias, mais não tem uma estimativa real, já na organização E, o prazo de manutenção corretiva é de até 2 dias.

A base histórica (E8), o objetivo foi identificar onde são armazenados os registros de cada cliente (E8.1), nas organizações A,B,C e E os registros do primeiro contato com o cliente (E8.1), o registro da execução da manutenção (E8.2), o registro da entrega (E8.3) e pós entrega (E8.4) são armazenados no próprio sistema de *Help Desk*, na organização D esse processo de registro é inexistente.

4.2.1 Ameaças a validade – Estudo de Caso

A seleção das organizações para realização desse estudo foi baseada em: Fábricas de *Software* onde a manutenção é recorrente, Organizações privadas certificadas, pois contribuiria na análise da manutenção de *software* quando o conjunto de conhecimentos é institucionalizado, organizações privadas onde não há certificação, pois poderíamos comparar os dados com as que possuem;

organizações públicas que atuam com desenvolvimento de software e consequentemente há manutenção;

Uma das ameaças desse estudo de caso é a provável impossibilidade de generalização dos resultados observados, pelo fato de ser um estudo de caso que representa apenas uma experiência, nesse caso com cinco organizações e responsáveis diferentes. Sendo assim, não há como afirmar que os resultados obtidos são válidos para todos os casos em que o modelo for aplicado. Outro ponto de atenção é com relação à coleta e análise dos dados, como parte dos itens de medição trata-se de respostas abertas através de questionários, pode se tornar uma ameaça depender de percepções subjetivas, por parte dos entrevistados.

A última ameaça observada está relacionada ao envolvimento direto do pesquisador com o estudo, pois participou ativamente na aplicação dos questionários e coleta das informações.

4.3 RESULTADOS – MODELO PARA MANUTENÇÃO DE SOFTWARE

Após a análise de resultados de ambos os estudos, um conjunto de elementos pertinentes a manutenção de software foi enumerado e organizado e assim elaborado um modelo. A organização hierárquica destes elementos é ilustrada na Figura 5. Os elementos presentes nos trabalhos citados na Tabela 3 estão presentes neste modelo e cada elemento recebeu uma identificação (E1, E1.1, E2, E,3) com o propósito de facilitar a condução deste trabalho.

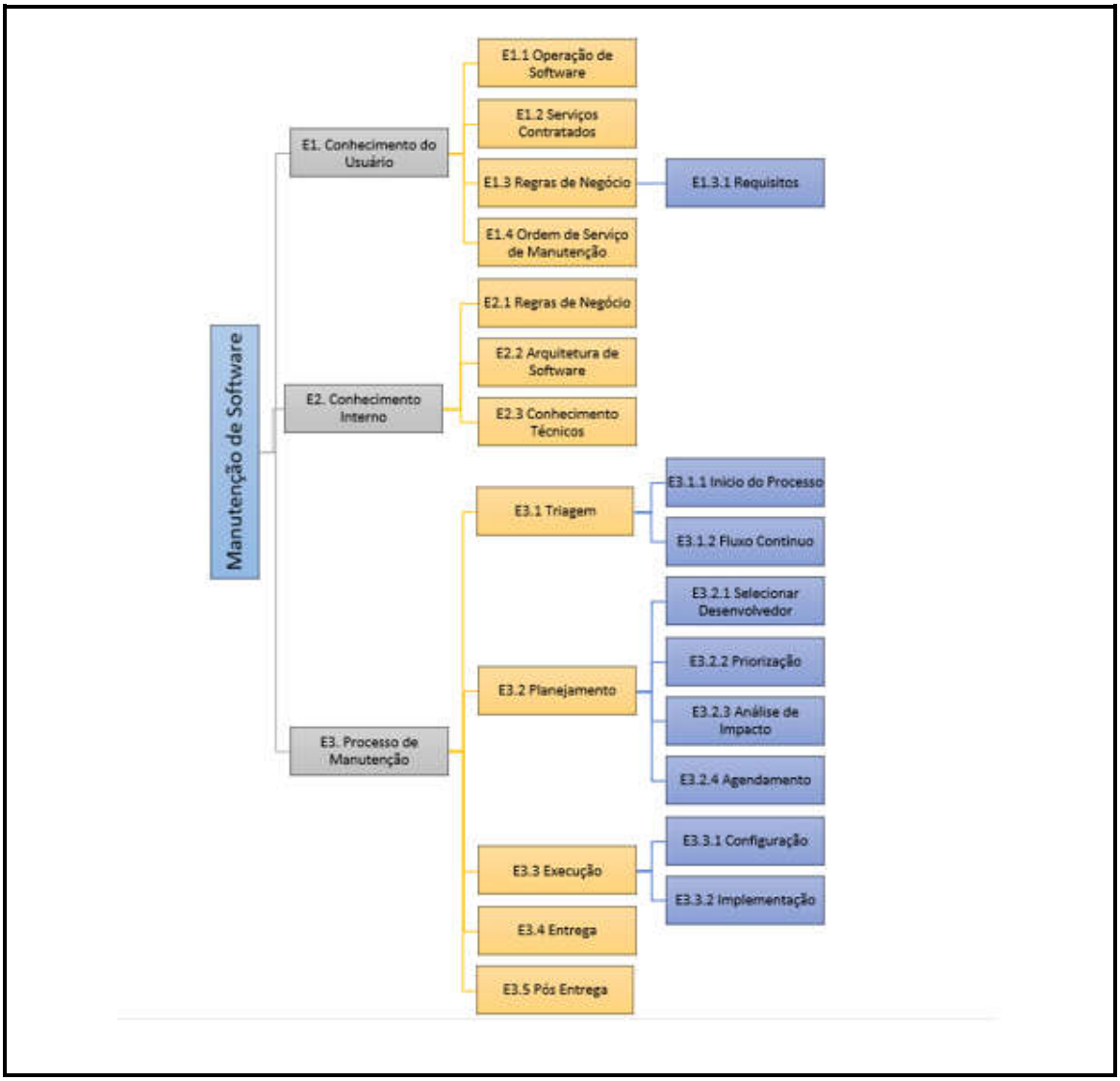

Figura 5 - Modelo para Manutenção de Software

Os elementos identificados neste trabalho, ilustrados na Figura 5 abrange três dimensões da manutenção de software. São eles: o Conhecimento dos Usuários, o Conhecimento Interno da organização de TI e o Processo de Manutenção de Software. Estas dimensões são descritas a seguir.

Considerando o elemento E1 (Figura 5), cujo o objetivo foi levantar informações sobre o gerenciamento do conhecimento do usuário, foi identificado que em relação às operações do software (E1.1), as organizações A, C e E detém este

subprocesso. Há uma preocupação inerente nestes casos em fornecer aos clientes conhecimento (por meio de manuais, textos, treinamentos) sobre a operação do produto. Os casos B e D não fornecem tais subsídios para os clientes e nestes casos, foi identificado que, os clientes possuem certa dificuldade em operar o software. O fato de fornecer conhecimento de como operar o software minimiza a quantidade de falsos chamados que o usuário efetua ao *call center*. Além disso, promove uma melhor compreensão dos aspectos técnicos de operação do produto o que torna seu uso muito mais efetivo e simples do ponto de vista do usuário final.

O conhecimento sobre os serviços/produtos contratados (E1.2) estão presentes nos casos A, B e E e nestes casos há um entendimento melhor do usuário em relação ao contrato do pacote de serviços. Na organização B, por ser um setor público, não existe um contrato de prestação de serviços e o mesmo ocorre na organização D. Este elemento é o indicativo de escopo do que foi contratado. O elemento E1.2 apresenta para o cliente os limites que este pode solicitar modificações/atualizações e ajuda sobre o produto/serviço contratado.

Foi observado que o elemento E1.3 (Regras de Negócio), e consequentemente os requisitos (E1.3.1) que tais regras demandam, é existente somente nas organizações A e E. Tal conhecimento direciona o usuário para a compreensão dos requisitos funcionais que o software atende. Quando o usuário tem o domínio de suas regras de negócio, os requisitos repassados para a organização são mais eficazes do ponto de vista do cliente, que recebe uma implementação do produto adequada e do ponto de vista da organização de TI que minimiza implementações errôneas advindas de requisitos do cliente.

Todas as organizações possuem algum método de abertura de chamado (E1.4) que foi informado para o usuário. As organizações A, B,C, e E possuem uma ferramenta de tickets enquanto que a organização D não possui nenhum mecanismo de controle de ordem de serviço. Foi detectado que nos casos A, B, C e E a documentação e históricos de manutenções ficam armazenadas (ou associadas) na própria ferramenta de ticket.

Em todos os casos, o elemento E1 e seus subprocessos foram detectados como altamente importantes, pois:

- Promove a eficácia e eficiência do serviço/produto contratado;
- Torna a execução da manutenção mais fácil e rápida disseminando o conhecimento sobre as regras de negócio para todos (cliente e organização de TI);
- Evita a abertura de falsos chamados de manutenção por parte do usuário;
- Resolve rapidamente problemas de abertura de chamado quando este é ou não uma manutenção;
- Cria registros das interações entre organização de TI e seu cliente;
- Possibilita renegociar o escopo do contrato entre o cliente e organização de TI;

O elemento E2 teve o propósito de identificar aspectos sobre o gerenciamento do conhecimento da equipe de TI que realiza a manutenção. Não foi detectado o subprocesso E2.1, referente ao conhecimento das regras de negócio somente no caso D. Nos demais casos foi detectada uma preocupação intrínseca sobre disseminar o funcionamento das regras de negócio do cliente para os colaboradores de TI, neste caso, todos os desenvolvedores, envolvidos no produto em questão detinham o conhecimento sobre tais regras. Estas eram disseminadas frequentemente por meio de treinamentos e workshops. O conhecimento das regras de negócio, por parte dos desenvolvedores, amplia as opções de pessoal envolvidas em um possível processo de manutenção. Neste sentido, gerenciar o conhecimento das regras de negócio do cliente dentro da organização de TI minimiza o *lock-in* a um desenvolvedor específico.

Em relação aos aspectos arquitetônicos (E2.2), somente as organizações A,B,C e E gerenciam o conhecimento em relação a arquitetura do software, na organização D, somente um integrante possui habilidade em relação a arquitetura os demais integrantes da equipe não possuem. O conhecimento da arquitetura por parte do desenvolvedor evita que o mesmo execute procedimentos de manutenção e degrade outro ponto do software.

Analisando o E2.3, cujo o objetivo foi levantar informações sobre os aspectos de gerenciamento do conhecimentos técnicos das equipes que realizam a

atividade de manutenção, somente na organização D esse elemento é inexistente, pois nem todos os integrantes tem conhecimento técnico nas linguagens em que os produtos de software foram desenvolvidos para realizar uma devida manutenção.

Em todos os casos o elemento E2 e seus subprocessos foram identificados como relevante, pois:

- Facilitam a compreensão dos chamados executados pelos clientes por parte dos desenvolvedores;
- Minimizam a dependência da organização de TI a um desenvolvedor específico;
- Possibilitam que vários desenvolvedores possam executar procedimentos de manutenção;
- Minimizam impactos negativos (degradação de outra funcionalidade, por exemplo) quando a manutenção é executada;
- Fornecem subsídios para um melhor planejamento da manutenção (Elemento E3.2);
- Minimizam o tempo de setup (E3.3.1) por parte dos desenvolvedores;

Embora todas as organizações possuíssem um processo de manutenção, indicado na Figura 5 como E3 nenhuma delas segue a norma ISO/IEC 14764. Todas as organizações indicaram que após ou durante o contato do cliente (que pode ser realizada por ferramenta *online* de *tickets*, *help desk*, telefone, por exemplo) um primeiro processo de triagem é executado. Tal processo é indicado como E3.1 na Figura 5, o processo de triagem tem duas abordagens: A primeira delas ocorre quando a solicitação de manutenção do usuário é efetiva, ou seja, realmente o software precisa de modificações. Neste caso, inicia-se o processo de manutenção (E1.1.1) enviando o pedido para a próxima etapa/papel responsável. Porém, tal pedido do cliente pode ser identificado como uma dúvida e neste caso ocorre o fluxo denominado como continuo (E3.1.2). O usuário irá receber a resposta, resolvendo seu problema, mas sem executar procedimento de manutenção (modificar/atualizar o software). O subprocesso E3.1 (triagem) foi detectado em todos os casos. O responsável pela execução da triagem nas organizações A, B, C, e E é o analista de suporte, ou *ticket manager* responsável pelo *HelpDesk*, é onde ele verificará se a

solicitação do cliente é uma dúvida no sistema, ou uma correção/manutenção. No caso da organização D, esse papel não possui responsável, visto que a maioria das solicitações são iniciadas via telefone ao setor.

O subprocesso planejamento (E3.2) contém os seguintes subprocessos: escolha dos desenvolvedores (E2.3.1), priorização (E2.3.2), análise de impacto (E2.3.3) e elaboração de cronograma (E2.3.4). Na organização A qualquer desenvolvedor consegue realizar a manutenção e por este motivo, qualquer colaborador pode ser escolhido para executar a manutenção. Na organização B e D a manutenção também é realizada pelo time de desenvolvimento, dependendo do projeto de software, nestes casos há um profissional especifico somente para a realização da manutenção. Nas organizações C e E esse processo é realizado apenas pelo time de desenvolvimento, não havendo um profissional especifico para realizar a manutenção. O efeito de *turnover* de profissionais foi identificado somente na organização C. De acordo com a organização, este afeta muito a manutenção de software, pois em alguns casos, o conhecimento de regras de negócios do software centram no usuário, o que pode trazer problemas para a organização de TI, nos demais casos esse problema não foi identificado. Em relação à priorização das tarefas de manutenção (E3.2.2), as organizações A, B, e E executam este subprocesso enquanto que nas demais organizações tal processo é inexistente. A ausência deste subprocesso nos casos C e D ocorrem pois a priorização é deliberada pelo cliente. Neste sentido, o gerenciamento interno de alocação de colaboradores torna-se sob demanda.

A análise de impacto (E3.2.3) dessas manutenções foi observada nas organizações A e B. Esta análise é essencial para que os desenvolvedores possam efetuar a manutenção sem afetar outros módulos/funcionalidades do software. Nos demais casos, este subprocesso não foi relatado como importante e/ou foi executado durante a própria execução da manutenção por parte do desenvolvedor. Tal fato ocorreu nos casos C e D e por este motivo houve relato de desenvolvedores que acidentalmente afetaram outro módulo do produto. O caso E aloca um *expertise* (colaborador que conhece a regra de negócio e a tecnologia) para executar a manutenção e por isso, tal problema é minimizado.

A organização A informou que são 2 dias o prazo máximo, quando se trata de manutenção corretiva (E3.2.4), manutenções adaptativas depende do contrato com o cliente, a organização B informou que o prazo para manutenção corretiva são de até 4 dias, dependendo da demanda, manutenção adaptativa não tem uma estimativa de prazo. A organização C, as manutenções corretivas são realizadas em até 5 dias, dependendo da sua criticidade. Na organização D, o prazo depende das atividades do desenvolvedor, se for algo muito urgente pode ser realizada em até 5 dias, mais não tem uma estimativa real, já na organização E, o prazo de manutenção corretiva é de até 2 dias.

Na organização A, quem executa a manutenção (E.3.3) é um analista responsável por manutenção do time de desenvolvimento. Na organização B, é um analista específico que só executa manutenção. Nas organizações C, D e E é o próprio desenvolvedor, ou desenvolvedor que tem um grande nível de conhecimento no software.

A Gestão de configurações (E3.3.1), investiga o tempo/complexidade despendido pelo desenvolvedor em configurar seu ambiente de trabalho para iniciar a manutenção, na organização A o tempo é de 40 minutos aproximadamente para configuração, na organização B o tempo para configuração do ambiente de trabalho é de 1 hora, as organizações C, D e E não souberam mensurar esse tempo. Somente o Caso A continha uma documentação específica de gerenciamento de configurações que permitia o desenvolvedor configurar seu ambiente. Nos demais casos todos os desenvolvedores continham em seu ambiente de trabalho o produto já configurado. Após a configuração do ambiente (E3.3.1) as organizações realizavam efetivamente a manutenção. Este processo é indicado como Elemento implementação (E3.3.2) na Figura 5. O subprocesso de implementação pode conter várias atividades: codificação, ajustes de *layout*, testes e homologação. Durante a codificação, todas as organizações afirmam que o usuário pode ser contatado. Nas organizações A, B a comunicação é realizada pelo mantenedor e nas demais organizações pelo próprio desenvolvedor. Foi observado nos estudos que o contato com o cliente nesta etapa facilita a resolução da manutenção, pois muita das vezes a ordem de serviço não é clara e gera dúvidas em relação à operacionalidade do

sistema. Essa interação é muito presente nas organizações, seja ele via *Help Desk*, telefone ou até mesmo e-mail.

Na organização A, tem um responsável para executar os testes funcionais no software. Na organização B o teste realizado pelo próprio mantenedor. Na organização C os testes são realizados pelo desenvolvedor, assim como na organização D, E. A homologação da nova versão, após a manutenção, foi identificada nas organizações A, B e E nas demais organizações a nova versão é colocada diretamente em produção.

O elemento entrega (E3.4), e pós entrega (E3.5) na organização A, o cliente é notificado que há uma versão já disponível para homologação, quem realiza essa atividade de entrega é o analista que realiza a manutenção. A versão antiga permanece no ambiente produtivo, até que seja realizada a aprovação da versão que está no ambiente de homologação do cliente. Essa aprovação/aceitação tem um período de resposta do cliente em 24 horas. Na organização B, o responsável que realizou a abertura do chamado recebe uma notificação via *HelpDesk* que há uma versão disponível no ambiente de homologação. O analista que executou a manutenção é o mesmo que atualiza ambos ambientes (homologação e produção) e realiza o acompanhamento no cliente no prazo de 48 horas. Na organização E, o cliente é notificado também via *HelpDesk* sobre a disponibilização de uma nova versão. O desenvolvedor que realiza a manutenção é quem atualiza o ambiente do cliente, o acompanhamento junto ao cliente é no período de 24horas.

A Figura 6 indica um modelo de processo de manutenção de software, identificado após a execução do método de pesquisa nas organizações A, B, C, D e E.

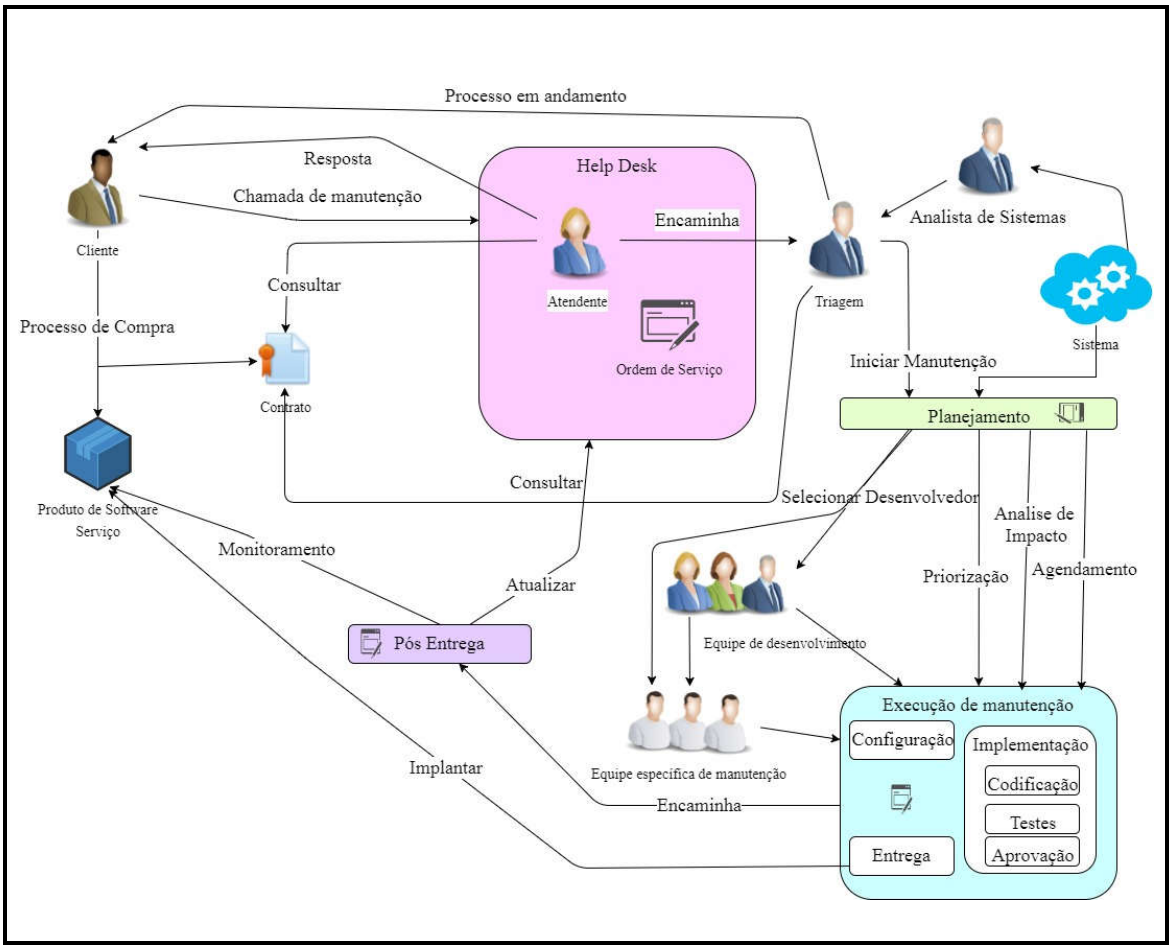

Figura 6 - Processo de Manutenção das organizações

O processo de manutenção pode ser iniciado pelo cliente ou por um analista relacionado ao sistema. Se de um lado, o cliente pode solicitar manutenções, o analista de sistemas pode detectar alguma necessidade e solicitar a manutenção, sem interferência do cliente. Durante o processo de aquisição, o cliente compra um pacote de software/serviços de manutenção e como resultado desta transação há um contrato. O cliente pode solicitar uma manutenção em alguma funcionalidade desse software (Manutenção Perfectiva), ou solicita uma modificação devido a algum erro no produto (Manutenção Corretiva). Para ambas as solicitações são geradas uma ordem de serviço no *Help Desk*, que contém a descrição da solicitação/problema do usuário, o prazo para atendimento destas é

definido via contrato com o cliente. Há dois possíveis encaminhamentos pelo *help desk*. Caso o próprio *help desk*, detecte que a ordem de manutenção, na verdade é uma dúvida do cliente, poderá encaminhar o pedido para um fluxo de trabalho continuo. O mesmo pode ocorrer, caso a triagem detecte posteriormente. O fluxo continuo é um fluxo cujo propósito é resolver o problema do cliente sem executar modificações no software. Tal fluxo pode envolver mais papéis além dos indicados na Figura 6.

Após a triagem, inicia-se a manutenção cujo primeiro processo é o planejamento. No planejamento são definidos os desenvolvedores com habilidade de executar a manutenção ou o atendimento para levantar os novos requisitos do software. É importante salientar que tal procedimento pode ser executado por um time específico ou não. Independente do papel (mantenedor/desenvolvedor), a ordem e serviço entrarão no processo de manutenção de cada organização, garantindo a entrega da correção ou melhoria ao cliente solicitante. Todos os casos realizam contato direto com o usuário solicitante da ordem de serviço, durante a execução da manutenção. Esse contato existe, pois ajuda no processo de dúvidas, no levantamento dos objetivos de novos requisitos, ou até mesmo operacionalidade que ocasionou determinado erro no software.

Os resultados obtidos nos estudos de casos mostraram que as organizações investigadas executam processos similares. A Figura 6 é uma generalização desta. Importante ressaltar que os quatros tipos de manutenções dispostos na literatura, ilustrados na Figura 1 não são equiparados e que grande parte das organizações concentram-se em manutenção corretiva e adaptativa.

Após analisar todos os aspectos presentes na Figura 6 que indicam o modelo de como o processo de manutenção é executado pelas organizações, há também outros dois processos que influenciam fortemente na execução e adequação no processo de manutenção. O processo de gerenciamento de cliente, indicado na Figura 7 é extremamente importante pois ampara o usuário, tornando o software mais adequado ao seu ambiente e evita que este realize falsos chamados de manutenção. Neste processo os conhecimentos sobre o produto, o contrato, as

regras de negócio deverão ser disseminados e institucionalizados entre os *stakeholders*.

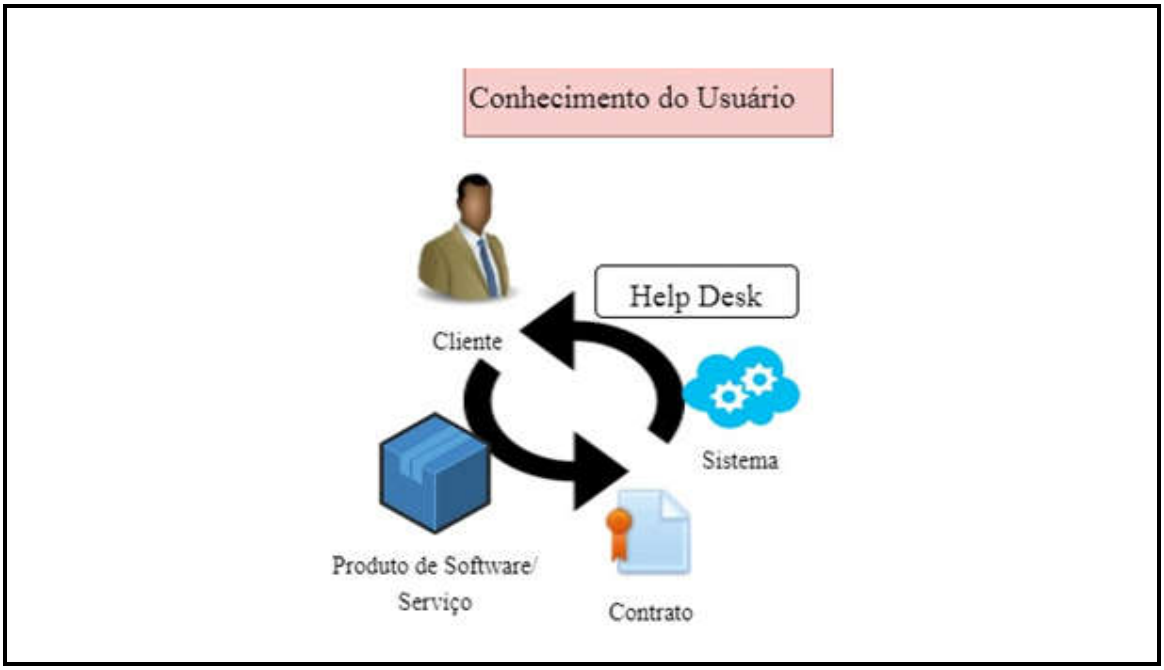

Figura 7 - Conhecimento do Usuário

O processo de gerenciamento interno, indicado pela Figura 8 ampara o processo de manutenção, tornando-o mais ágil e reduzindo erros. Neste caso, temos, os aspectos relacionados às regras de negócio, aspectos relacionados a arquitetura do software e aspectos técnicos. Tais conceitos precisam ser disseminados e institucionalizados entre os participantes do processo de manutenção.

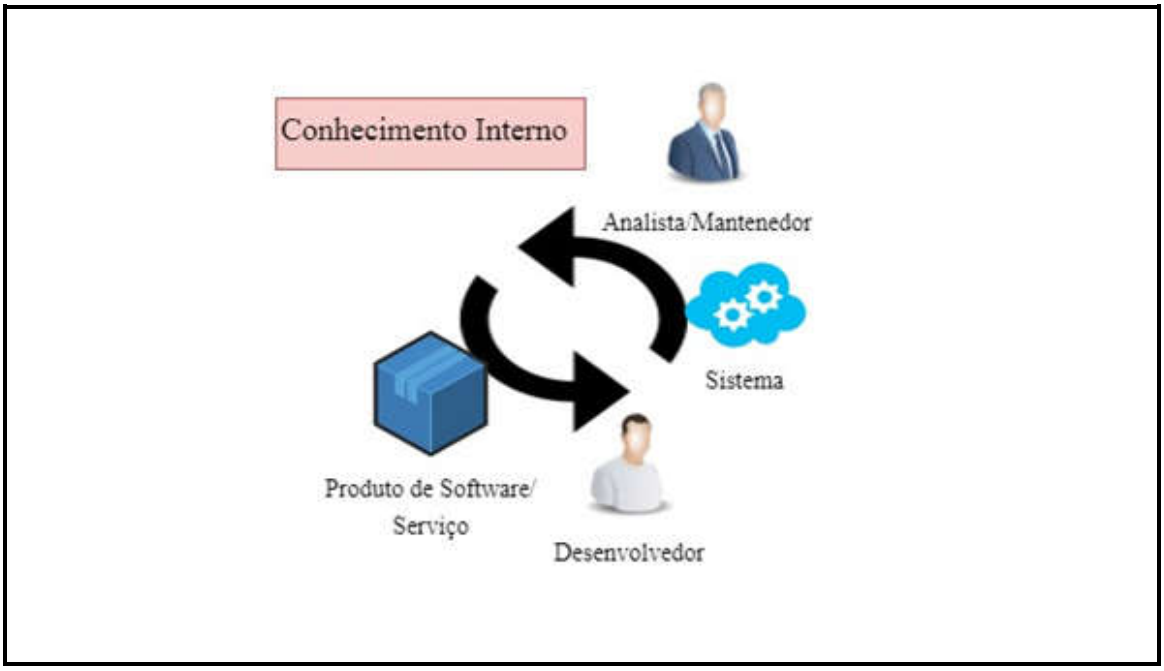

Figura 8 - Conhecimento Interno

5. CONCLUSÕES

O presente trabalho apresentou um estudo híbrido, o qual duas metodologias foram aplicadas: Experimentos com o objetivo de medir a capacidade de aprendizagem de equipes inseridas no contexto acadêmico, na aplicação de processos e atividades relacionadas à manutenção de software, e um estudo de múltiplos casos em organizações, cujos problemas enfrentados na manutenção de software, são agravados pela falta de um modelo para as atividades e tarefas dentro do próprio contexto empresarial. Foi evidenciado em ambos os estudos que os problemas internos interferem diretamente na atividade de manutenção de software, gerando falta de planejamento, alto custo, sobrecarga de tarefas, e falha na comunicação com os usuários desses sistemas.

Os experimentos apresentaram os resultados descritos no subcapitulo 4.1. Ao analisar os resultados dos quatros experimentos apresentados nos Gráficos 2, 3, 5 e 6 e de acordo com as questões da Tabela 2, é possível concluir que em ambos os experimentos:

- A realização da manutenção foi uma tarefa simples, e as equipes tiveram habilidades em executar as exigências e programas suficientes para atender à solicitação dos usuários.

- O tempo atribuído para manutenção foi suficiente, por mais que os alunos tivessem que configurar o ambiente para executar a tarefa proposta.

- Na graduação, os alunos não precisaram realizar um treinamento com o colega para entender a tarefa, ao contrário do mestrado, onde foi possível observar que muitos tiveram experiências em desenvolvimento de software, mas tiveram que treinar alguém para realizar o processo de manutenção.

- Foi a primeira vez que ambas equipes tiveram contato com o produto, embora fosse uma questão simples, eles tiveram receio (relatados por eles mesmos) de atualizar o produto principal no repositório.

- Todos concordaram que era possível identificar o problema na ordem do serviço, ou seja, o nível de conhecimento do usuário que abriu o pedido de serviço era suficiente para a clareza do problema proposto.
- Todos concordaram que o conhecimento da regra de negócio é necessário para realizar a manutenção no software.
- Ambas as equipes tiveram dificuldades em configurar o ambiente, ou seja, o usuário precisava necessariamente de algum conhecimento na linguagem de programação para realizar qualquer manutenção.
- Os alunos concordaram com um todo que conheceram o produto, realizaram a manutenção e não se preocuparam com os outros recursos do sistema, concentrando-se apenas na resolução do que foi proposto.

O objetivo dos experimentos foi verificar se é possível o ensino de manutenção de software, através de técnicas lúdicas em robótica, e os resultados mostraram que as hipóteses H1 e H3 foram respondidas positivamente:

H1) Sim, o modelo preliminar de processo de manutenção de software é válido;

H3) Sim, é possível o ensino de manutenção de software, através de técnicas lúdicas em robótica, desde que os alunos tenham conhecimento no produto e em suas regras de negócio;

Observando os estudos realizados nas organizações pode-se perceber que muitas das vezes a manutenção fica centralizada em apenas em usuário (mais experiente), por falta de conhecimento dos demais integrantes do time, criando uma relação de manutenção a apenas um profissional e dificultando a agilidade nesse processo por parte da equipe.

Pode-se concluir que todas as organizações investigadas executam um processo de manutenção de software pré-estabelecido, mesmo que sem formalização e/ou documentação. Tal informação colabora positivamente para responder a primeira hipótese (H1) deste trabalho.

Nenhuma organização atentou-se as normas relacionadas a manutenção, no que tange a modelos específicos de manutenção, os todos os casos, as organizações não mostraram-se estimuladas a implantação dos modelos citados na seção 2.2, porém, percebe-se que há certa aderência a eles. Foi constatado que os casos com processos mais formalizados (Caso A), a própria certificação CMMI, embora não implantada com o propósito específico de amparar a manutenção de software tornou o processo de manutenção mais formalizado e consequentemente o melhorou. Tais informações, respondem a segunda hipótese deste trabalho (H2).

As hipóteses H3 e H4 foram respondidas positivamente pelos casos. Embora os casos com processos poucos formais (Caso C e D) não tenham implantado tais processos, a ausência destes, principalmente no que tange a gestão de conhecimento interno e cliente causa atualmente um alto custo de manutenção. Estes casos estudam a implantação de tais processos justamente com o propósito de minimizar custos e melhorar o atendimento ao cliente.

Portanto, com a análise apresentada nesse estudo de caso, foi possível responder a seguinte pergunta: **Como as organizações executam o processo de manutenção de software?**

Foi identificado que as organizações possuem processo para a atividade de manutenção de software e que esses processos são informais e documentados no caso das organizações A, B, E, nas organizações C e D o processo de manutenção é informal e não documentado. Os processos de engenharia de software dão amparo na organização A, onde o conjunto de conhecimento é institucionalizado, nas demais organizações o processo de engenharia de software não dá amparo na manutenção.

Em todas as organizações, a equipe de desenvolvimento é a mesma que realiza a manutenção de software, em nenhuma organização existe um time especifico somente para realizar as atividades de manutenção, porém elas não descartaram a possibilidade da existência de tal time e que há tal possibilidade de acordo com o contrato.

Pelo todo exposto, espera-se que essa pesquisa contribua para o conhecimento dentro do campo da Manutenção de Software, apresentando uma experiência positiva no emprego desse modelo de manutenção, assim podendo fornecer essa experiência para outras organizações. As dificuldades existentes para realização dessa pesquisa é o fato de cada organização ter seu processo de manutenção e o mesmo estarem em execução há muito tempo, podendo assim gerar certa resistência a mudanças, outro fator importante é que a metodologia proposta não poderá interromper o processo atual que está em execução nas organizações.

5.1 TRABALHOS FUTUROS

Sugere-se como trabalho futuro, a implantação desse modelo nas organizações e a elaboração de um levantamento, junto aos colaboradores e gestores, para validação da eficiência da proposta.

O modelo aqui apresentado contempla todos os tipos de manutenções ilustrados na Figura 1. Porém, nos casos investigados não houveram relatos de manutenção corretiva e perfectiva. Neste sentido, como trabalho futuro, um próximo estudo de caso, será aplicado em uma organização que contempla estes dois tipos de manutenção com o propósito de validar o modelo como um todo.

5.2 DIVULGAÇÃO DOS RESULTADOS

Durante o processo dessa dissertação de mestrado, obtiveram-se as publicações apresentadas a seguir:

- THOMAZINHO, H.C.S.; LERÁRIO, A.L.; FABRI, J.A. **A software maintenance strategy with a large number of users: A case study.** In: 12th Iberian Conference on Information Systems and Technologies, Lisbon, Portugal, 2017. (DOI: 10.23919/CISTI.2017.7975860).

- THOMAZINHO, H.C.S.; LERÁRIO, A.L.; FABRI, J.A. **Teaching software maintenance with ludic techniques supported by robotics.** In: Frontiers in Education Conference (FIE), Indianapolis, Estados Unidos, 2017. (DOI: 10.1109/FIE.2017.8190720).

- THOMAZINHO, H.C.S.; LERÁRIO, A.L.; FABRI, J.A. **A case study on the strategy of maintaining commercial software with a large number of users.** In: JISEM: Journal of Information Systems Engineering. **Published:** August 30, 2017. (DOI: https://doi.org/10.20897/jisem.201723).

A seguir, o artigo que se encontra em fase de elaboração:

- THOMAZINHO, H.C.S.; LERÁRIO, A.L.; FABRI, J.A. **An approach of Software Maintenance Process: a case study.** A ser submetido em revista na área de Engenharia de Software.

REFERÊNCIAS

AHERN DENNIS; AARON, C.; RICHARD, T. **CMMI® Distilled: A Practical Introduction to Integrated Process Improvement, Third Edition**. 3. ed. Nova Jersey: PEARSON EDUCATION, 2008. v. 39

AHMAD, M. O. et al. Transition of software maintenance teams from scrum to Kanban. **Proceedings of the Annual Hawaii International Conference on System Sciences**, v. 2016-March, n. i, p. 5427–5436, 2016.

ALARANTA, M.; BETZ, S. Knowledge problems in corrective software maintenance - A case study. **Proceedings of the Annual Hawaii International Conference on System Sciences**, p. 3746–3755, 2011.

ASSESSMENT, M. P.; PRENTNER, W.; SNEED, H. M. Analyzing Data on Software Evolution Processes. p. 1–10, 2016.

BANO, M.; ZOWGHI, D. User Involvement in Software Development and System Success: A Systematic Literature Review. **EASE '13: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering**, p. 125–130, 2013.

BUDIARDJO, E. K. et al. Ontology-Based Knowledge Management System Model for R3ST Software Maintenance Environment. **Proceedings of the Fifth International Conference on Network, Communication and Computing**, p. 198–202, 2016.

CHAPIN, N. **Software maintenance in complying with IT governance: A report from the field**IEEE International Conference on Software Maintenance, ICSM. **Anais**...2009

CHRISTA, S.; SUMA, V. Significance of Ticket Analytics in Effective Software Maintenance. **Proceedings of the ACM Symposium on Women in Research 2016 - WIR '16**, p. 126–130, 2016.

CHUA, B. B. **Rework requirement changes in software maintenance**Proceedings - 5th International Conference on Software Engineering Advances, ICSEA 2010. **Anais**...2010

DANTAS, F. et al. Enhancing design models with composition properties: A software maintenance study. **AOSD 2013 - Proceedings of the 2013 ACM on Aspect-Oriented Software Development**, p. 49–60, 2013.

FABRI, J. A. et al. A importância da Abordagem dos Conceitos de Metodologia de Pesquisa para os Cursos de Ciências da Computação. **XIII Congreso Iberoamericano de Educación Superior em Computación. Santiago de Cali Colômbia. 10 a 14 de outubro de 2005**, 2005.

FERNANDEZ-SAEZ, A. M. et al. On the use of UML documentation in software maintenance: Results from a survey in industry. **2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems, MODELS 2015 - Proceedings**, p. 292–301, 2015.

FONSECA, J. J. S. et al. Metodologia do Trabalho Científico. **São Carlos: Serviço de Biblioteca e Informação …**, p. 1–48, 2002.

ISO. International Standard - ISO/IEC 14764 IEEE Std 14764-2006 Software Engineering - Software Life Cycle Processes - Maintenance. **ISO/IEC 14764:2006 (E) IEEE Std 14764-2006 Revision of IEEE Std 1219-1998)**, p. 0_1–46, 2006.

ISO. **International Standard ISO/IEC 12207: Systems and software engineering — Software life cycle processes**. [s.l: s.n.]. v. 2

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing Systematic Literature Reviews in Software Engineering. **Engineering**, v. 2, p. 1051, 2007.

KOSCIANSKI, A.; SOARES, M. DOS S. **Qualidade de Software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. 2. ed. São Paulo: NOVATEC, 2007.

LI, J. et al. **Cost drivers of software corrective maitenance: An empirical study in two companies**IEEE International Conference on Software Maintenance, ICSM. **Anais**...2010

LIKERT, R. A technique for the measurement of attitudes. **Archives of Psychology**, v. 22 140, p. 55, 1932.

MARQUES-NETO, H.; APARECIDO, G. J.; VALENTE, M. T. A quantitative approach for evaluating software maintenance services. **Proceedings of the ACM Symposium on Applied Computing**, p. 1068–1073, 2013.

MELLEGARD, N. et al. Impact of Introducing Domain-Specific Modelling in Software Maintenance: An Industrial Case Study. **IEEE Transactions on Software Engineering**, v. 42, n. 3, p. 248–263, 2016.

NGUYEN, V.; BOEHM, B.; DANPHITSANUPHAN, P. A controlled experiment in assessing and estimating software maintenance tasks. **Information and Software**

Technology, v. 53, n. 6, p. 682–691, 2011.

OHBA, M. et al. Development of the Simulation for Taking Over Process in the Software Maintenance. **40th International Conference on Computers and Industrial Engineering: Soft Computing Techniques for Advanced Manufacturing and Service Systems, CIE40 2010**, p. 0–5, 2007.

PETERSEN, K. et al. Systematic mapping studies in software engineering. **EASE'08 Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering**, p. 68–77, 2008.

PFLEEGER, S. L. **Engenharia de Software: Teoria e Prática**. 2. ed. São Paulo: Pearson Education do Brasil, 2004.

POKLEMBA, T.; SIVY, I.; HAVLICE, Z. Maintenance software processes for web 2.0 based learning management systems. **Emerging eLearning Technologies and Applications (ICETA), 2011 9th International Conference on**, p. 167–170, 2009.

PRESSMAN, R. S. **Engenharia de Software - Uma Abordagem ProfissionalDevelopment**, 2007.

REZENDE, D. A. **Engenharia de software e sistemas de informação**. São Paulo: Brasport, 2005.

SINGH, P. K.; SANGWAN, O. P. A process metrics based framework for Aspect Oriented Software to predict software bugs and maintenance. **Proceedings of the 5th International Conference on Confluence 2014: The Next Generation Information Technology Summit**, p. 831–836, 2014.

SOMMERVILLE, I. **Engenharia de Software**. 8. ed. Nova Jersey: PEARSON EDUCATION, 2007.

SOMMERVILLE, I. **Engenharia de Software**. São Paulo: Pearson Brasil, 2011a.

SOMMERVILLE, I. **Engenharia de Software**. [s.l: s.n.].

STANDARD, I. **INTERNATIONAL STANDARD ISO / IEC Software Engineering — Software Life**. 2. ed. [s.l: s.n.]. v. 2006

STOJANOV, Z. et al. Context dependent maintenance effort estimation: Case study in a small software company. **2013 IEEE 8th International Symposium on Applied Computational Intelligence and Informatics (SACI)**, p. 461–466, 2013.

STOJANOV, Z.; BRTKA, V.; DOBRILOVIC, D. Evaluating Software Maintenance Processes in Small Software Company based on Fuzzy Screening. p. 67–72, 2014.

TSUNODA, M. et al. Benchmarking Software Maintenance Based on Working Time.

2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence, p. 20–27, 2015.

TSUNODA, M.; ONO, K. Pitfalls of analyzing a cross-company dataset of software maintenance and support. **2014 IEEE/ACIS 15th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2014 - Proceedings**, 2014.

VAN DER SCHUUR, H.; JANSEN, S.; BRINKKEMPER, S. Sending out a software operation summary: Leveraging software operation knowledge for prioritization of maintenance tasks. **Proceedings - Joint Conference of the 21st International Workshop on Software Measurement, IWSM 2011 and the 6th International Conference on Software Process and Product Measurement, MENSURA 2011**, n. section VII, p. 160–169, 2011.

WAZLAWICK, R. S. **Metodologia de Pesquisa para Ciência da Computação**. 2. ed. Rio de Janeiro: [s.n.].

WOHLIN, C. et al. **Experimentation in software engineering**. 1. ed. New York: Springer, 2012. v. 9783642290

YIN, R. K. Case study research: Design and methods. **Case Study Research: Design and Methods**, n. September 2011, 2013.

ANEXO A - Artigos encontrados no mapeamento sistemático realizado nesta pesquisa:

1. M. O. Ahmad, P. Kuvaja, M. Oivo and J. Markkula, "Transition of Software Maintenance Teams from Scrum to Kanban," *2016 49th Hawaii International Conference on System Sciences (HICSS)*, Koloa, HI, 2016, pp. 5427-5436. doi: 10.1109/HICSS.2016.670

2. J. Sillito and E. Wynn, "The Social Context of Software Maintenance," *2007 IEEE International Conference on Software Maintenance*, Paris, 2007, pp. 325-334. doi: 10.1109/ICSM.2007.4362645

3. J. W. Eckstein, C. A. Newman and R. L. Shaw, "Visualization of software maintenance attrition," *2012 IEEE Systems and Information Engineering Design Symposium*, Charlottesville, VA, 2012, pp. 30-33. doi: 10.1109/SIEDS.2012.6215152

4. T. Hall, A. Rainer, N. Baddoo and S. Beecham, "An empirical study of maintenance issues within process improvement programmes in the software industry," *Proceedings IEEE International Conference on Software Maintenance. ICSM 2001*, Florence, 2001, pp. 422-430. doi: 10.1109/ICSM.2001.972755

5. Z. Car and B. Mikac, "A method for modeling and evaluating software maintenance process performances," *Proceedings of the Sixth European Conference on Software Maintenance and Reengineering*, Budapest, 2002, pp. 15-23. doi: 10.1109/CSMR.2002.995786

6. Jingyue Li, T. Stålhane, J. M. W. Kristiansen and R. Conradi, "Cost drivers of software corrective maintenance: An empirical study in two companies," *2010*

IEEE International Conference on Software Maintenance, Timisoara, 2010, pp. 1-8.
doi: 10.1109/ICSM.2010.5609538

7. J. Kunstar and Z. Havlice, "Architecture for ease of software systems maintenance," *2008 6th International Symposium on Applied Machine Intelligence and Informatics*, Herlany, 2008, pp. 195-200. doi: 10.1109/SAMI.2008.4469163

8. M. Bombardieri and F. A. Fontana, "A specialisation of the SQuaRE quality model for the evaluation of the software evolution and maintenance activity," *2008 23rd IEEE/ACM International Conference on Automated Software Engineering - Workshops*, L'Aquila, 2008, pp. 110-113. doi: 10.1109/ASEW.2008.4686328

9. L. Agostini, L. Cerchio, G. Ciccarella and R. Saracco, "Software maintenance in the telecommunication area-status and perspectives," *Real Time, 1990. Proceedings., Euromicro '90 Workshop on*, Horsholm, 1990, pp. 47-58.
doi: 10.1109/EMWRT.1990.128226

10. M. Tsunoda, A. Monden, K. Matsumoto, S. Ohiwa and T. Oshino, "Benchmarking Software Maintenance Based on Working Time," *2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence*, Okayama, 2015, pp. 20-27.
doi: 10.1109/ACIT-CSI.2015.13

11. W. K. Wiener-Ehrlich, J. R. Hamrick and V. F. Rupolo, "Modeling Software Behavior in Terms of a Formal Life Cycle Curve: Implications for Software Maintenance," in *IEEE Transactions on Software Engineering*, vol. SE-10, no. 4, pp. 376-383, July 1984.
doi: 10.1109/TSE.1984.5010250

12. Z. Stojanov, D. Dobrilovic, J. Stojanov and V. Jevtic, "Context dependent maintenance effort estimation: Case study in a small software company," *2013 IEEE 8th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, Timisoara, 2013, pp. 461-466. doi: 10.1109/SACI.2013.6609019

13. Bagnato *et al.*, "Testing and remote maintenance of real future internet scenarios: Towards FITTEST and FastFix advanced software engineering," *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Szczecin, 2011, pp. 925-932.

14. Alaranta and S. Betz, "Knowledge Problems in Corrective Software Maintenance--A Case Study," *2012 45th Hawaii International Conference on System Sciences*, Maui, HI, 2012, pp. 3746-3755. doi: 10.1109/HICSS.2012.406

15. P. Rossi, P. Antonini and T. Lanza, "Experience on the control of software maintenance in SIP," *Proceedings. Conference on Software Maintenance 1991*, Sorrento, 1991, pp. 256-260.
doi: 10.1109/ICSM.1991.160340

16. S. Selvi, G. D. Maheshwari, S. Khan, A. K. Gupta, B. N. Anand and A. K. Biswal, "Design, development and implementation of maintenance management software for RDCIS, SAIL," *2013 4th International Conference on Computer and Communication Technology (ICCCT)*, Allahabad, 2013, pp. 252-257.
doi: 10.1109/ICCCT.2013.6749636

17. J. Leonard, J. Pardoe and S. Wade, "Software maintenance-Cinderella is still not getting to the ball," *Second IEE/BCS Conference: Software Engineering, 1988 Software Engineering 88.*, Liverpool, 1988, pp. 104-106.

18. N. Mellegård, A. Ferwerda, K. Lind, R. Heldal and M. R. V. Chaudron, "Impact of Introducing Domain-Specific Modelling in Software Maintenance: An Industrial Case Study," in *IEEE Transactions on Software Engineering*, vol. 42, no. 3, pp. 245-260, March 1 2016.
doi: 10.1109/TSE.2015.2479221

19. D. Mancl and W. Havanas, "A study of the impact of C++ on software maintenance," *Proceedings. Conference on Software Maintenance 1990*, San Diego, CA, 1990, pp. 63-69.
doi: 10.1109/ICSM.1990.131325

20. D. A. Penny, "An estimation-based management framework for enhancive maintenance in commercial software products," *International Conference on Software Maintenance, 2002. Proceedings.*, 2002, pp. 122-130.
doi: 10.1109/ICSM.2002.1167759

21. Sellink and C. Verhoef, "An architecture for automated software maintenance," *Proceedings Seventh International Workshop on Program Comprehension*, Pittsburgh, PA, 1999, pp. 38-48.
doi: 10.1109/WPC.1999.777742

22. Z. Stojanov, V. Brtka and D. Dobrilovic, "Evaluating software maintenance processes in small software company based on fuzzy screening," *2014 IEEE 9th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, Timisoara, 2014, pp. 67-72.
doi: 10.1109/SACI.2014.6840037

23. Y. Nakamura, N. Oomiya, M. Ohba, H. Yamamoto and Y. Maruyama, "Development of the simulation for taking over process in the software maintenance," *The 40th International Conference on Computers & Indutrial Engineering*, Awaji, 2010, pp. 1-6.
doi: 10.1109/ICCIE.2010.5668383

24. M. Tsunoda and K. Ono, "Pitfalls of analyzing a cross-company dataset of software maintenance and support," *15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, Las Vegas, NV, 2014, pp. 1-6. doi: 10.1109/SNPD.2014.6888729

25. K. Kurtel, "Measuring and Monitoring Software Maintenance Services: An Industrial Experience," *2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement*, Ankara, 2013, pp. 247-252. doi: 10.1109/IWSM-Mensura.2013.43

26. G. Antoniol, G. Casazza, G. A. Di Lucca, M. Di Penta and F. Rago, "A queue theory-based approach to staff software maintenance centers," *Proceedings IEEE International Conference on Software Maintenance. ICSM 2001*, Florence, 2001, pp. 510-519.
doi: 10.1109/ICSM.2001.972764

27. P. J. Layzell and L. Macaulay, "An investigation into software maintenance-perception and practices," *Proceedings. Conference on Software Maintenance 1990*, San Diego, CA, 1990, pp. 130-140.
doi: 10.1109/ICSM.1990.131342

28. S. Smit, P. H. N. de With and G. J. van Dijk, "Evolution of a software maintenance organization from cost center to service center," *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.*, 2003, pp. 209-212.
doi: 10.1109/ICSM.2003.1235423

29. S. Harlev, Y. Gevirzman and J. Solomon, "IAI software maintenance procedure," *[1989] Proceedings. The Fourth Israel Conference on Computer Systems and Software Engineering*, Herzlia, 1989, pp. 84-89. doi: 10.1109/ICCSSE.1989.72720

30. L. B. Arfa, A. Mili and L. Sekhri, "An empirical study of software maintenance," *Proceedings. Conference on Software Maintenance 1991*, Sorrento, 1991, pp. 52-58.
doi: 10.1109/ICSM.1991.160306

31. P. Siqueira, "Software Requirements for Reliability-Centered Maintenance Application," *2006 International Conference on Probabilistic Methods Applied to Power Systems*, Stockholm, 2006, pp. 1-7.
doi: 10.1109/PMAPS.2006.360282

32. H. Donner, "Ground rules for software maintenance," in *Computer*, vol. 28, no. 10, pp. 84-85, Oct 1995.
doi: 10.1109/2.467605

33. J. Krogstie and A. Solvberg, "Software maintenance in Norway: a survey investigation," *Proceedings 1994 International Conference on Software Maintenance*, Victoria, BC, 1994, pp. 304-313.
doi: 10.1109/ICSM.1994.336764

34. H. M. Sneed, "Offering software maintenance as an offshore service," *2008 IEEE International Conference on Software Maintenance*, Beijing, 2008, pp. 1-5.
doi: 10.1109/ICSM.2008.4658047

35. M. Fernández-Sáez, D. Caivano, M. Genero and M. R. V. Chaudron, "On the use of UML documentation in software maintenance: Results from a survey in industry," *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Ottawa, ON, 2015, pp. 292-301.
doi: 10.1109/MODELS.2015.7338260

36. Z. Stojanov, V. Brtka and D. Dobrilovic, "Evaluation of the software maintenance tasks based on fuzzy screening," *2013 IEEE 11th International*

Symposium on Intelligent Systems and Informatics (SISY), Subotica, 2013, pp. 375-379.
doi: 10.1109/SISY.2013.6662605

37. Vogel-Heuser, J. Fischer, S. Rösch, S. Feldmann and S. Ulewicz, "Challenges for maintenance of PLC-software and its related hardware for automated production systems: Selected industrial Case Studies," *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Bremen, 2015, pp. 362-371.
doi: 10.1109/ICSM.2015.7332487

38. M. Ino, "Current state of software maintenance in Japan: in depth view," *Proceedings Conference on Software Maintenance 1992*, Orlando, FL, 1992, pp. 27-29.
doi: 10.1109/ICSM.1992.242562

39. M. Reformat and V. Wu, "Analysis of software maintenance data using multi-technique approach," *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*, 2003, pp. 53-59.
doi: 10.1109/TAI.2003.1250170

40. W. C. C. Chu, "Retrospect of Taiwan's software industry and issues of software maintenance and evolution," *2008 IEEE International Conference on Software Maintenance*, Beijing, 2008, pp. 480-481.
doi: 10.1109/ICSM.2008.4658114

41. Andrea De Lucia, Eugenio Pompella, and Silvio Stefanucci. 2002. Effort estimation for corrective software maintenance. In Proceedings of the 14th international conference on Software engineering and knowledge engineering (SEKE '02). ACM, New York, NY, USA, 409-416.
doi: 10.1145/568760.568831

42. Sharon Christa, Suma V, 2016. Significance of Ticket Analytics in Effective Software Maintenance: Awareness. In Proceedings of the ACM Symposium on Women in Research: ACM, Indore, India 126-130.
Doi: 10.1145/2909067.2909091

43. Eko K. Budiardjo, Elviawaty M. Zamzami, Ganjar Ramadhan, 2016. Ontology-Based Knowledge Management System Model for R3ST Software Maintenance Environment. Published in: ICNCC '16 Proceedings of the Fifth International Conference on Network, Communication and Computing Pages 198-202
Doi: 0.1145/3033288.3033340

ANEXO B – Formulário utilizado para entrevistas do estudo de caso nas organizações:

Nome do Entrevistado:	Data da Entrevista:
Cargo:	Departamento:

DADOS GERAIS DA EMPRESA

- Característica da empresa:
- Porte da empresa (Pequena/Média/Grande):
- Número de colaboradores:
- Número de clientes:
- Número de Projetos:
- Certificações (qual nível? Que unidades de negócio se aplica?):

RECURSOS HUMANOS

- A equipe de desenvolvimento é a mesma que realiza a manutenção de um mesmo software?
- Qualquer integrante da equipe tem conhecimento para prestar manutenção em todos os produtos de software da empresa?
- Existe alguma rotatividade das equipes (desenvolvimento/manutenção de software)? Se sim, com que frequência ocorre essa rotatividade?
- A equipe que executa a manutenção tem relacionamento direto com o usuário? (Caso positivo, qual o perfil do profissional que tem o contato direto com o usuário?)
- Na equipe (projeto, empresa, time) existem tarefas de *mentoring*?
- As certificações ajudaram a atividade de manutenção de software?
- Poderia citar quais as mudanças a empresa teve após adquirir a certificação?

PRODUTO/PROJETO INVESTIGADO

- Qual a linguagem de programação?
- Qual banco de dados utilizado?
- Qual ambiente de execução (Web, Mobile)?
- Quais modelagens/ ferramentas utilizadas?

- Quais ferramentas/processos para controle de versão?
- Quais as tecnologias e interfaces utilizadas (Ws, rest, etc)?
- Quais os frameworks utilizados?
- Quantidade de pessoas envolvidas e/ou papéis envolvidos?
- Existe documentação?
- Existe modelo de processo de software?
- Existe algum processo automatizado?
- A organização capta novos projetos com restrição de tecnologia?

REGRAS DE NEGÓCIO – CONHECIMENTO EQUIPE

- Todos os desenvolvedores da equipe relacionados a manutenção, tem conhecimento de todas as linguagens de programação dos produtos de software desenvolvidos?
- A equipe tem conhecimento em todas as bibliotecas utilizadas para o desenvolvimento dos produtos de software?
- A equipe tem conhecimento em arquitetura de software?
- A equipe tem conhecimento das regras de negócio dos softwares desenvolvidos?
- A equipe tem conhecimento em todas IDE utilizadas para o desenvolvimento dos softwares?
- A equipe tem conhecimento em todos os bancos de dados utilizados pelos produtos de software/ recursos do banco de dados*: psql, store procedures, sequencials*?
- A equipe tem conhecimento em todas as interfaces utilizadas?

PROCESSOS DE GESTÃO DE CONFIGURAÇÕES

- Há algum processo de gestão de configuração do ambiente para o desenvolvimento do produto?
    - ✓ Tempo para configuração do ambiente.
    - ✓ Tempo para aprender a manusear a IDE e ferramentas.
    - ✓ Tempo para aprender sobre o produto.
    - ✓ Tempo para aprender regras de negócio.
- Custos de gestão de configurações (setup):
    - ✓ A empresa aloca um colaborador experiente para realizar a manutenção?

      ou
    - ✓ A empresa aloca um novo colaborador para realizar a manutenção?

TURNOVER

- O *turnover* tem trazido impactos nos custos e prazos dos projetos de software?
- Há (re)alocação ou substituição de colaboradores para determinada atividade?
- A substituição dos colaboradores é reativa ou proativa?
- Qual o prazo estimado para o colaborador adquirir conhecimento nas regras de negócio?
- Qual o prazo estimado para o colaborador adquirir conhecimento nas configurações do ambiente de desenvolvimento?

PROCESSO DE MANUTENÇÃO DE SOFTWARE

- Existe planejamento das tarefas ou atividades de manutenção de software?
- Existe algum processo de manutenção de software na empresa?
- A empresa segue alguma atividade da norma ISO/IEC 14764?
- Existe alguma documentação para registro das atividades de manutenção de software?
- Existe priorização das demandas de manutenção de software?
- Existe análise de impacto baseada no planejamento da manutenção de software?
- Existe análise de impacto baseada na entrega da manutenção de software?
- Existe um *feedback* para o cliente do prazo da resolução da manutenção?
- Existe acompanhamento após build/release?
- Existe testes funcionais?
- Existe testes de regressão?