Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Manutenção de Software no Tribunal de Justiça de Goiás: emprego da DSR

Michel Alves Ribeiro

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientadora
Prof.a Dr.a Rejane Maria da Costa Figueiredo

Brasília

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

ALVES RIBEIRO, MICHEL
AA474m Manutenção de Software no Tribunal de Justiça de Goiás: Emprego da DSR / MICHEL ALVES RIBEIRO; orientador REJANE MARIA DA COSTA FIGUEIREDO. -- Brasília, 2021.
286 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2021.

1. Engenharia de Software. 2. Manutenção de Software com Ágeis. 3. Processo de Manutenção para Serviço Público. 4. CI/CD para Métodos Ágeis. 5. Método Design Science Research. I. MARIA DA COSTA FIGUEIREDO, REJANE, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Manutenção de Software no Tribunal de Justiça de Goiás: emprego da DSR

Michel Alves Ribeiro

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

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

Prof. Dr.a Edna Dias Canedo (Interno)
CIC/UnB

Prof. Dr.a Rosana Teresinha Vaccare Braga (Externo)
ICMC - USP/São Carlos

Prof. Dr. Glauco Vitor Pedrosa (Suplente)
FGA/UnB

Prof. Dr. Marcelo Ladeira
Coordenador do Programa de Pós-graduação em Computação Aplicada

Brasília, outubro de 2021

Dedicatória

Dedico este trabalho à Deus, aos meus pais, esposa e mestres que formam os alicerces para que eu alcance meus sonhos.

Agradecimentos

Agradeço a Deus, por ter me dado plenas condições de traçar um caminho de sucesso na vida.

À minha família, meus pais Antônio e Onísia, minha esposa Graziela Colarullo pela compreensão e suporte nos momentos difíceis.

A todos do Programa de Computação Aplicada da UNB que me possibilitaram uma experiência incrível e que, mesmo em tempos de pandemia, se adaptaram e continuaram com as atividades de ensino e com as pesquisas.

À minha orientadora, Dra. Rejane Figueiredo. Sinto-me muito privilegiado e agradeço o carinho, a dedicação e a oportunidade de ter sido seu orientando.

Aos gestores e colegas do Tribunal de Justiça de Goiás que permitiram que eu conduzisse esta pesquisa em seu âmbito.

Resumo

A tecnologia avança rapidamente e muitas mudanças são decorrentes dela. Conciliar as mudanças das regras de negócio com as mudanças nos sistemas se torna um desafio, e a urgência se torna uma regra. Assim, tomadas pelo senso de urgência, as equipes de desenvolvimento de sistemas se veem num ciclo de manutenções urgentes e intermináveis. No setor público, as mudanças chegam mais lentamente, mas os desafios se esbarram numa gestão consolidada durante décadas de trabalhos não automatizados. As dificuldades em lidar com manutenção de software no Brasil se assemelham a outros países e a aplicação dos métodos ágeis pode diminuir ou solucionar problemas de manutenção em empresas públicas ou privadas. O *Tribunal de Justiça de Goiás (TJGO)* é um órgão do Poder Público Brasileiro que se destaca por sua prestação jurisdicional apoiada em tecnologia. Entretanto, a ausência de um processo bem definido de software colabora para desordem e retrabalhos que poderiam ser evitados. Neste contexto, o objetivo deste trabalho foi definir e avaliar a implantação de um processo de manutenção de software para o TJGO. A metodologia de pesquisa adotada é do tipo explicativa, com o emprego da técnica de pesquisa *Design Science Research (DSR)*, que permite conduzir a pesquisa pela produção de artefatos tecnológicos capazes de transformar situações, alterando suas condições para estados mais desejáveis. Os resultados apontaram que, devido as peculiaridades do TJGO, foi necessário tratar as vertentes da manutenção e definir uma arquitetura de *deployment* e um processo sistemático. De acordo com a percepção dos envolvidos, observou-se melhora na gestão, no trabalho em equipe, no clima organizacional, e na velocidade das entregas.

Palavras-chave: Manutenção de Software, Métodos Ágeis, Equipe de Servidores Públicos, DSR.

Abstract

Por outro lado, estudos recentes elaborados por Mantovani e Marczak [1] apontam que desde os anos 2000, o Brasil vem aderindo cada vez mais aos métodos ágeis, como indica a Figura 2.1, prática que se observa pelo mundo todo como uma das principais formas de inovação digital. Thus, taken by the urgency sense, the systems development teams enter in a loop of urgent and endless maintenance. In the public sector, the changes becomes more slowly, but the challenges add up to a consolidated management during decades of non-automated work. The difficulties to deal with software maintenance in Brazil are similar to other countries, but there are scientific consensus that the application of agile methods can reduce or solve maintenance problems in public or private companies. The *Goiás Court of Justice (TJGO)* is an organ of the Brazilian Public Power that stands out for its jurisdictional provision that is supported by several technology tools. However, the lack of a well-defined software process, contributes to rework that could be avoided. In this context, the objective of this work was to define and evaluate the implementation of a software maintenance process for TJGO. The research methodology adopted is explanatory type, with the use of the *Design Science Research (DSR)* technique, that allows to drive the research by technologics artifacts that enable to transform situations, to change their conditions to more desirable states. The the results indicate that, because of TJGO peculiarities, it was necessary to work the maintenance aspects and to define a deployment architecture, and define a systematic process. According to the perception of involved stakeholders, it was possible to observe to be observed management improvements, teamwork improvements, organizational climate improvements, and more speed of deliveries.

Keywords: Software Maintenance, Agile Methods, Civil Servant Teams, DSR

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

API Application Programming Interface.

CMMI Capability Maturity Model integration.

DDD Domain Driven Design.

DES Divisão de Engenharia de Software.

DIT Divisão de Infraestrutura Tecnológica.

DSR Design Science Research.

DTI Diretoria de Tecnologia da Informação.

ENTIC-JUD Estratégia Nacional de Tecnologia da Informação e Comunicação.

ER Engenharia de Requisitos.

FWAs Flexible Work Arrangements.

HTTP Hypertext Transfer Protocol.

IEC International Electrotechnical Commission.

IEEE Instituto de Engenheiros Eletrônicos e Eletricistas.

ISO International Organization of Standardization.

MPS-BR Melhoria de Processo de Software Brasileiro.

NTIID Núcleo Técnico de Internet/Intranet e Design.

NTSA Núcleo Técnico de Sistemas Administrativos.

NTSJ Núcleo Técnico de Sistemas Judiciais.

PBQP Programa Brasileiro da Qualidade e Produtividade de Software.

PMI Project Management Institute.

PO Product Owner.

QA Quality Assurance.

RAD Rapid Application Development.

RUP Rational Unified Process.

SMmm Software Maintenance Maturity Model.

TDD Test Driven Development.

TIC Tecnologia da Informação e Comunicação.

TJGO Tribunal de Justiça do Estado de Goiás.

UML Unified Modeling Language.

WIP Work In Progress.

Capítulo 1

Introdução

Neste capítulo apresenta-se o contexto deste trabalho, apontando a pergunta que motiva esta pesquisa. Apresenta-se também a revisão bibliográfica envolvendo os temas manutenção de software e o emprego da metodologia ágil na manutenção, seguidos do problema no contexto do Tribunal de Justiça do Estado de Goiás (TJGO), escopo deste estudo. Na sequência, são apresentados os objetivos desta pesquisa e a metodologia adotada para alcançá-los. Finalizando, apresenta-se a estrutura deste documento.

1.1 Contexto

A evolução tecnológica promove cenários que mudam rapidamente, sendo essencial que as organizações e os órgãos públicos acompanhem essas mudanças. Durante muitas décadas, a preocupação principal foi a inserção na era digital. Muitas pesquisas e produtos surgiram para facilitar e atender às necessidades primárias dos usuários. Algumas soluções ultrapassaram gerações e se tornaram difíceis de serem mantidas.

Os sistemas computacionais estão inseridos nesse contexto e muitos sistemas resistiram após as transformações e continuam funcionando. Alguns desses sistemas só existem devido às manutenções contínuas que, em alguns casos, até se confundem com o desenvolvimento.

Sommerville [15] ressalta que: "o divisor para saber onde termina o desenvolvimento e onde começa a manutenção está se tornando irrelevante, pois poucas aplicações de software são desenvolvidas a partir do zero".

Nesse cenário, as equipes dedicam boa parte do tempo em manutenções urgentes, podendo comprometer as demandas estratégicas e importantes da organização. A urgência resulta em falta de tempo que, muitas vezes, é usada para justificar o não cumprimento

de requisitos, menos cobrados mas não menos importantes, como: qualidade; segurança; testabilidade; e manutenibilidade.

Quando comparada a outras fases do ciclo de vida do software, a manutenção não recebe a devida atenção, seja por não ter o atrativo de "criar" o novo, pela falta de interesse do dono do produto ou pelo retorno financeiro[16]. Para Kajko-Mattsson [16], a manutenção tem sido uma das atividades mais complexas, caras, difíceis e menos compreendidas da Engenharia de Software.

A definição de um processo para manutenção de software se tornou fundamental para se alcançar o sucesso. O custo de manutenção de sistemas em muitas organizações tem sido observado como tendo uma variação de 40% a 70% dos recursos referentes ao ciclo de vida completo do software [17].

Após o Manifesto Ágil, em 2001 [18], observou-se o crescimento na adoção de metodologias ágeis no mundo, seja nas empresas privadas ou nos governos. Os resultados obtidos pelo emprego de métodos ágeis têm sido semelhantes entre o Brasil e o resto do mundo, com aplicações e adaptações conforme o contexto. Algumas vezes, esses resultados são influenciados pelo tamanho e maturidade das empresas [19, 20].

As práticas ágeis com foco na melhoria da qualidade do software, como o Desenvolvimento Orientado por Testes (TDD - *Test Driven Development*), refatoração, integração contínua, entre outras, estão sendo aplicadas em empresas mais experientes com métodos ágeis. Além disso, as práticas de gestão tradicionais estão passando por muitos ajustes e, em alguns casos, sendo abandonadas. Isso pode indicar um campo importante para pesquisas, com o intuito de se construir um bom conhecimento sobre adaptação das práticas ágeis aos diferentes cenários [19].

Outros temas de pesquisa relevantes são os relacionados à necessidade de melhoria dos aspectos de manutenibilidade de sistemas utilizando-se metodologias ágeis [21, 12, 22], visto que, no geral, os estudos são destinados à fase de desenvolvimento e poucos trabalhos são dedicados à fase de manutenção de software.

No Tribunal de Justiça de Goiás (TJGO), objeto de estudo deste trabalho, o cenário da manutenção de seus sistemas é complexo e carece da definição de processos modernos, que possibilitem uma melhor gestão e controle dos sistemas, com mais agilidade, eficiência, colaboração, satisfação e valorização da mão de obra dos envolvidos.

1.2 Revisão Bibliográfica

Existem diferentes definições de manutenção de software. Na literatura é possível identificar alguns autores que assumem uma abordagem mais específica, enquanto outros assumem uma abordagem mais geral. Entretanto, de acordo com Grubb e Takang [17], o mais comum é associar a manutenção à modificação do sistema quando os requisitos são alterados.

De acordro com Kong e Liu [23], diante de situações de manutenção de software, os métodos tradicionais como *Rational Unified Process* (RUP) são incapazes de lidar com a criticidade que algumas manutenções requerem.

Em 2001, o manifesto ágil trouxe mudanças nos processos de produção de software e devido às características próprias da manutenção e seu alto valor para as instituições, algumas pesquisas vêm sendo realizadas em busca de modelos mais apropriados.

Diante do exposto, considerando a pergunta de pesquisa, realizou-se uma revisão da literatura (Apêndice A) para investigação dos processos e métodos ágeis nos últimos 20 anos, dando maior ênfase aos trabalhos dos últimos 5 anos. Desta revisão extraiu-se o Referencial Teórico que pode ser visto no Capítulo 2.

Tarwani e Chug [24] avaliaram o desempenho e qualidade de software usando metodologias ágeis para manutenção entre os anos de 2001 a 2015 e garantem que a adoção das metodologias ágeis promove a melhoria contínua, maior produtividade e qualidade.

Mantovani e Marczak [1], apontam que algumas das principais razões para adoção dos métodos ágeis são: Acelerar as entregas de produtos; Aumentar a produtividade; Melhorar a capacidade de priorização; Reduzir riscos; e Melhorar a qualidade. Enquanto que os maiores desafios enfrentados por órgãos brasileiros, no setor de TI, são [1]: Mudanças culturais; Resistência às mudanças; Obter colaboração dos clientes; Customizar práticas ágeis; e Comprometimento da alta gestão.

Estudos apontam alguns fatores que otimizam o emprego das metodologias ágeis, são eles: trabalhos em equipe [25, 26, 27], utilizar *sprints* [25], buscar simplicidade [23], possibilitar solicitações urgentes [25], uso de histórias de usuários [26], equilibrar a quantidade de documentação [25], manter o cliente próximo da equipe [27], gerar transparência [26], trabalhar a sinergia na equipe [26, 25], testes durante todo ciclo independente do tipo de manutenção [25, 26, 27], separar cada manutenção em diferentes fluxos [3, 21, 22].

A pesquisa realizada por Svensson e Höst [27] apontou que os engenheiros de software se beneficiaram de uma estrutura organizada devido às boas práticas dos método ágeis.

Heeager e Rose [25] aplicaram as boas práticas dos métodos ágeis na *Aveva*[1], onde relataram melhorias na qualidade do código fonte, na autoestima da equipe, maior visibilidade do progresso da manutenção, melhor comunicação e compartilhamento de conhecimento entre os envolvidos.

Ribeiro e Domingues [28], analisaram as melhores condições para se empregar metodologias ágeis em uma empresa pública de Portugal. Os autores [28] aplicaram o *Scrum* com algumas adaptações necessárias ao setor público o qual destacaram que as organizações públicas possuem cultura, modo de operação e entrega diferentes do setor privado, podendo tornar a implementação dessas metodologias um desafio.

A burocracia e o poder hierárquico são desafios para a implantação dos métodos ágeis [29]. Nesse sentido, Rulinawaty e Lukman [29] realizaram um estudo que se concentrou em oferecer uma visão integrada de como as organizações podem se tornar ágeis, como combinar valores e missão com agilidade organizacional a partir de estruturas orientadas a objetivos.

Para Mergel et al. [30], no modelo de gestão pública tradicional, os especialistas em gerenciamento de informações se limitam a tarefas de gestão de contratos, planejamento e supervisão e, por isso, é bastante comum a resistência às mudanças. Esse modelo com raízes históricas reduz a capacidade de inovação em Tecnologia da Informação (TI) e uma das consequências percebidas são os incentivos crescentes para terceirização no setor, especialmente no desenvolvimento de TI e de serviços.

Para Kajko-Mattsson [31], nas manutenções de software, o objetivo é atender ao problema. A única maneira de diminuir a subjetividade é explorando completamente o domínio do software e identificando suas atividades inerentes. Para isso, há a necessidade de tratar a manutenção levando em consideração as subcategorias (corretiva, adaptativa, evolutiva) [31]. Assim, torna-se possível realizar uma análise minuciosa estabelecendo bem o escopo do desenvolvimento e da manutenção, permitindo então diminuir a subjetividade e construir processos com maior precisão [31].

Para construção de um processo de manutenção de software, Kajko-Mattsson [31], sugere que se responda às seguintes perguntas: “o que implementar ?”, “por que implementar ?” e “como implementar o processo de manutenção ?”. Em determinadas situações, as respostas a essas perguntas estão intimamente vinculadas aos problemas que se deseja solucionar [31].

Assim, pesquisas anteriores produziram evidências de que a manutenção ágil afeta os resultados e desempenhos de forma crítica. Entretanto, os fatores para se alcançar

[1]Empresa de software para engenharia naval

o sucesso na agilidade operacional vão além da aplicação direta dos modelos, sendo, muitas vezes, necessário conhecê-los e adaptá-los aos diversos cenários e perfis humanos.

Nesse sentido, Bouguerra et al. [32] explicam que ainda se sabe pouco sobre o vínculo entre a agilidade e processos colaborativos em prol da sustentabilidade do ambiente de trabalho. Eles verificaram que medidas ambientais que contribuam para a colaboração também podem ser aprimoradas por meio de Arranjos de Trabalhos Flexíveis (*Flexible Work Arrangements* (FWAs), por exemplo, horário flexível, redução do horário de trabalho e compartilhamento de trabalho [32]. Bouguerra et al. [32] afirmam ainda que, para que isso não se torne um problema, deve-se implementar ferramentas de controle eficazes.

Muitas dessas ferramentas podem ser obtidas por meio dos processos de melhoria contínua. Para Melo [33], avaliar continuamente os fatores de produtividade é importante, pois eles podem mudar sob as novas práticas de engenharia de software. No entanto, poucas pesquisas investigaram os principais fatores que influenciam a produtividade do time ágil.

Albuquerque [34], indicou que a continuidade de programas de melhoria está relacionada aos fatores humanos, de projeto, organizacionais e técnicos associados ao processo, e concluiu que alguns desses fatores estão presentes também na fase de implementação, mas que o nível de importância é diferente quando comparado ao de manutenção.

Como resultados das boas práticas e da melhoria contínua dos processos, Albuquerque [34] detectou uma maior satisfação dos clientes, redução de custos e prazos, além do aumento da produtividade e qualidade.

Os processos ágeis devem ser constantemente revistos e atualizados, de modo que se adéquem à realidade, para se sustentarem ao longo dos anos. Para aumentar as chances de sucesso, alguns modelos de maturidade foram criados, por exemplo: *Capability Maturity Model integration* (CMMI), Melhoria de Processo de Software Brasileiro (MPS-BR), e no âmbito da manutenção, o modelo SMmm (*Software Maintenance Maturity Model*) e modelo Mantus que podem ser vistos no APÊNDICE J.

Lucas [35], propôs a adoção das boas práticas do modelo de maturidade de manutenção de software SMmm , com foco em gestão, empregando-as em um processo adaptado e concluiu que a aplicação das práticas derivadas do SMmm contribuíram para uma melhora perceptível na redução de defeitos provocados pelas manutenções.

1.3 Problema

No contexto de desenvolvimento de sistemas do Tribunal de Justiça de Goiás (TJGO) é comum a sensação de agilidade, ao descumprir a hierarquia para atendimento de problemas isolados motivados por fatores políticos pouco ou nada alinhados aos objetivos estratégicos da organização, como consequência, tem-se uma TI frágil, reativa aos problemas gerados.

São premissas do Tribunal de Justiça do Estado de Goiás que a modernização tecnológica avance com qualidade, celeridade, acessibilidade, agilidade, eficácia e efetividade a custos reduzidos e com a satisfação dos seus usuários, sendo a satisfação da sociedade o maior objetivo a ser alcançado. Dentre seus objetivos estratégicos, estão o aumento da maturidade em governança de TI e a Comunicação, visando ao aprimoramento dos processos e à entrega de serviços com qualidade e eficiência [36].

Todavia, existem muitos desafios para tornar essa premissa uma verdade. Um deles está relacionado ao processo de desenvolvimento de software do TJGO, que apesar de dispor de infraestrutura física e de ferramentas modernas para o desenvolvimento, possui ainda deficiências primárias no desenvolvimento de software.

Os desenvolvedores possuem, à disposição, tecnologias de ponta e soluções complexas. No entanto, em caso de uma manutenção emergencial o apreço às tecnologias deve dar lugar às soluções fáceis e rápidas. Tecnologias complexas podem realmente diminuir o ritmo e prolongar o estado de emergência [23].

Além disso, os sistemas devem continuar servindo sem grandes impactos negativos causados pelas manutenções. Atualmente, o parque do TJGO conta com sistemas legados e sistemas em constante manutenção corretiva e evolutiva.

Desse modo, criou-se um ambiente em que a manutenção repetitiva predomina em relação ao desenvolvimento de novas soluções de software, resultando numa equipe individualista, altamente especializada, com resultados pequenos se comparados à capacidade coletiva, provocando baixa qualidade nas entregas e riscos inerentes.

A Estratégia Nacional de Tecnologia da Informação e Comunicação (ENTIC-JUD), instituída pelo Conselho Nacional de Justiça (CNJ), por meio da Resolução nº 211/2015, [37], tem o propósito de avaliar anualmente o nível de cumprimento das diretrizes estratégicas e evolução dos viabilizadores da Governança, Gestão e Infraestrutura de Tecnologia da Informação e Comunicação (TIC) do Poder Judiciário [38].

A Estratégia Nacional de Tecnologia da Informação e Comunicação (ENTIC-JUD), instituída pelo Conselho Nacional de Justiça (CNJ), por meio da Resolução nº 211/2015,

[37], tem o propósito de avaliar anualmente o nível de cumprimento das diretrizes estratégicas e evolução dos viabilizadores da Governança, Gestão e Infraestrutura de Tecnologia da Informação e Comunicação (TIC) do Poder Judiciário [38].

A Resolução 2011/2015, art. 12 [37], define estruturas organizacionais adequadas e compatíveis com a relevância e demanda de TIC, considerando, no mínimo, os seguintes macroprocessos:

- Macroprocesso de governança e de gestão: *de planejamento, orçamentária, de aquisições e contratações de soluções, de projetos, de capacitação;*
- Macroprocesso de segurança da informação: *de continuidade de serviços essenciais, de incidentes de segurança, de riscos;*
- Macroprocesso de software: *de escopo e requisitos, de arquitetura, de processos de desenvolvimento e sustentação;*
- Macroprocesso de serviços: *de catálogo, de requisições, de incidentes, de ativos de microinformática, de central de serviços;*
- Macroprocesso de infraestrutura: *de disponibilidade, de capacidade, de ativos de infraestrutura e de telecomunicações corporativas.*

A Resolução 2011/2015, art.2 , § 2, [37] estabelece ainda que: "Caberá a cada órgão definir os seus processos, observando as boas práticas pertinentes ao tema, criando um ambiente favorável à melhoria contínua".

De acordo com April [39], para uma organização ser considerada madura é necessário que ela tenha instituído o seu processo de manutenção de software. Nessa linha, a ausência de um processo bem definido de manutenção de software não só contraria uma recomendação do CNJ, como pode prejudicar, a longo prazo, as conquistas e objetivos que o TJGO vem alcançando no cenário nacional nos últimos anos.

É nesse contexto que surge a **pergunta de pesquisa**: *Como definir e implantar um processo de manutenção de software para o TJGO, empregando a metodologia ágil num cenário de órgão público brasileiro, com equipe interna de desenvolvimento ?*

1.4 Objetivos

Reconhecendo que o cenário majoritário de desenvolvimento de software vivido pelo TJGO mais se assemelha ao de manutenção, nesta pesquisa buscou-se realizar um estudo

da aplicação dos métodos ágeis, analisar as melhores práticas e recomendações para manutenção dos sistemas desse órgão e aplicar algumas das práticas com o propósito de transformar o processo de manutenção dos sistemas do TJGO.

Assim, o objetivo geral desta pesquisa é : *definir e implantar um processo de manutenção de software para o TJGO, empregando metodologias ágeis.*

Os objetivos específicos são:

- Analisar as recomendações de processos em órgãos públicos, tais como Conselho Nacional de Justiça e Tribunal de Contas da União, sobre métodos ágeis no governo brasileiro, em específico para o poder judiciário;
- Analisar e selecionar as metodologias ágeis para identificar os pontos que melhor atendam ao contexto de manutenção de software do TJGO;
- Selecionar projetos de software para execução e experimentação do processo de manutenção definido.

1.5 Metodologia

Dado o objetivo deste trabalho, nesta seção apresenta-se a classificação metodológica e uma breve descrição do plano metodológico definido.

A presente pesquisa objetivou gerar conhecimentos para solução de problemas específicos na prática. Envolve interesses locais, com abrangência social, fatores que a caracterizam como uma *pesquisa de natureza aplicada.*

Quanto à abordagem do conteúdo, foram coletados dados e analisados de forma *qualitativa*, utilizando-se da subjetividade.

Sobre a tipologia da pesquisa, dado seu objetivo, ela é classificada como *experimental*. De acordo com Gil [40], pesquisas explicativas (experimentais) "têm como preocupação central identificar os fatores que determinam ou que contribuem para a ocorrência dos fenômenos". Geralmente, estudos experimentais demonstram a viabilidade de determinada técnica ou programa como uma solução, potencial e viável. Os procedimentos de coleta de dados podem variar bastante e técnicas de observação podem ser desenvolvidas durante a pesquisa [41].

Considerando que o pesquisador é integrante da equipe de desenvolvimento e manutenção de sistemas do TJGO há mais de dez anos e atual coordenador da equipe responsável por sistemas administrativos (composta por servidores do quadro efetivo),

pode-se afirmar que o processo de produção de software do setor possui fatores ligados ao comportamento humano moldado pela cultura organizacional.

Desse modo, a técnica adotada é a *Design Science Research (DSR)*, que será vista em detalhes no Capítulo 4 (Materiais e Métodos). Esta escolha se deu porque a DSR permite conduzir a pesquisa pela produção de artefatos tecnológicos capazes de transformar situações, alterando suas condições para estados mais desejáveis, com viés epistemológico, tendo em conta os rigores inerentes da técnica. São exemplos de artefatos: componentes de software, hardware, organização, processo de negócio, serviço, método, técnica, etc.[2].

O plano metodológico adotado neste trabalho, conforme apresentado na Figura 1.1, e detalhado no Capítulo Materiais e Métodos, compreendeu quatro fases:

- Planejamento da Pesquisa;
- Coleta de Dados;
- Análise e Interpretação dos Dados; e
- Redação dos Resultados.

1.5.1 Planejamento da Pesquisa

Nesta fase foram definidos o tema da pesquisa, a questão da pesquisa, os objetivos geral e específicos, e o plano metodológico.

1.5.2 Coleta de Dados

Nessa fase foram adotadas as pesquisas bibliográfica e documental, e a técnica *Design Science Research* (DSR). A DSR propõe a criação de conhecimento usando *design*, análise, reflexão e abstração [42] e é uma técnica que exige rigor na coleta e na análise dos dados com foco no problema, a partir da produção de artefatos.

A DSR propõe a aplicação de artefatos no contexto social para solução de problemas observados e é adequada quando pesquisadores necessitam trabalhar de forma colaborativa com as organizações, para testar novas ideias em contextos reais [43].

A DSR é composta por um ciclo regulador chamado de ciclo de *Design Science* (DS), basicamente composto por artefatos e investigação. Wieringa [2] propõe cinco fases: *Investigação do problema; Design da Solução; Validação da Solução; Implementação da*

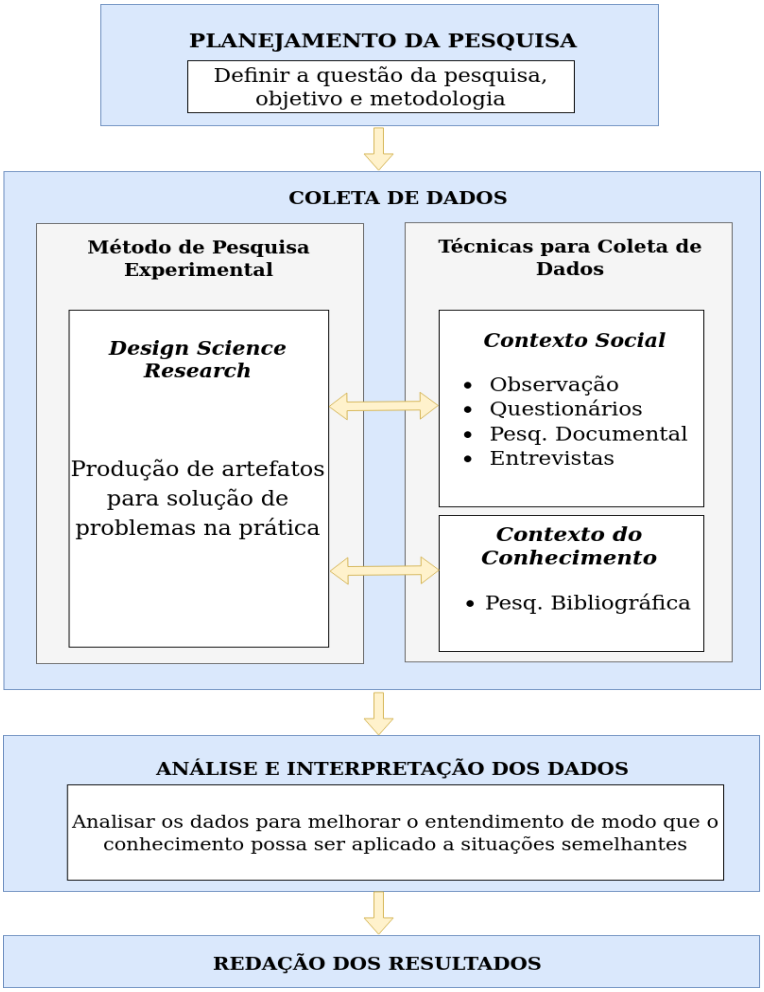


Figura 1.1: Plano Metodológico (Fonte: adaptado [2]).

Solução; e *Avaliação da Implementação*, que levam em consideração o *contexto social do problema* e o *contexto do conhecimento*.

Os instrumentos de coleta de dados foram definidos conforme os autores sugerem para uma pesquisa em Engenharia de Software [44], dado os objetivos da pesquisa e o universo investigado, conforme detalhamento:

- Observação: Conforme Wholin et al. [44], a técnica de observação pode ser utilizada para investigar como os engenheiros de software realizam uma determinada tarefa. As técnicas de observação podem utilizar recursos como gravações de áudio, reuniões, ou estímulos que levem aos pesquisados revelarem seus pensamentos [44].

- Questionários: O questionário consiste na aplicação de uma série de perguntas a um entrevistado. O pesquisador pode definir a forma das perguntas de acordo com seu propósito [43].
- Pesquisa Documental: A Pesquisa Documental permite coletar informações prévias sobre os tópicos que serão pesquisados, podem ser utilizados documentos verbais ou não verbais. Os documentos são classificados como fontes primárias ou secundária. Documentos primários são aqueles compilados ou criados pelo pesquisador, enquanto os secundários foram transcritos de fontes primárias ou, então tratam-se de fotografias, gravações, etc. [43].

1.5.3 Análise e Interpretação dos Dados

Essa fase se destina à análise dos resultados da pesquisa. O objetivo foi coletar as percepções dos envolvidos quanto ao processo de manutenção de software resultante, após o emprego da DSR. Para isso, foram empregadas a técnica *Entrevista* e o método *Análise de Conteúdo*:

- Entrevistas: As perguntas foram do tipo *semiestruturadas*. De acordo com Wohlin et al. [44], nas entrevistas semiestruturadas um conjunto de perguntas são usadas como base para se explorar os assuntos que se deseja obter informações. Elas possibilitam a improvisação e exploração dos objetos dando maior liberdade ao entrevistado para expor suas ideias.
- Método Análise de Conteúdo: Para análise dos dados resultantes das entrevistas utilizou-se o *método Análise de Conteúdo* [45], cujo objetivo é respaldar os estudos realizados com foco na questão de pesquisa. Como facilitador para aplicação do método, utilizou-se da ferramenta computacional Atlas.ti [46], conforme sugerido por Gibbis [47].

1.5.4 Redação dos Resultados

Esta monografia do trabalho compreende o relato desta pesquisa.

1.6 Estrutura do trabalho

Este trabalho está organizado em oito capítulos.

O Capítulo 1 - Introdução, compreende o contexto da pesquisa, a revisão bibliográfica, os objetivos e a metodologia adotada.

O Capítulo 2 - Manutenção de Software, apresenta uma breve contextualização sobre a manutenção de software, em seguida, relata trabalhos realizados sobre manutenção com metodologias pioneiras, como Mantema (1999) [21], e algumas mais recentes, como o guia Mantelasoft (2019) [48] e Impress (2019) [22].

No Capítulo 3 - Arquitetura de software e Testes, apresenta conceitos de Arquitetura de Software, em seguida um destaque maior é dado para o segmento de arquitetura de *deployment*. Adiante, são descritos conceitos de testes de software utilizando-se metodologias ágeis. Ambos assuntos apresentam significativa relevância para se estabelecer um processo de manutenção de software utilizando-se métodos ágeis.

O Capítulo 4 - Materiais e Métodos, diante do objetivo da pesquisa, apresenta-se o passo a passo do planejamento metodológico.

No Capítulo 5 - Contexto da TI no TJGO, apresenta o objeto de estudo desta pesquisa. Descreve o contexto da área de Engenharia de Software do TJGO, com ênfase no núcleo de sistemas administrativos e na organização das equipes, e no processo de trabalho.

O Capítulo 6 - Manutenção de Software do TJGO, apresenta a execução metodológica, expondo os ciclos da DSR executados e as mudanças no processo de manutenção de software, realizados no núcleo de sistemas administrativos do TJGO.

No Capítulo 7 - Análise de Conteúdo do Processo de Manutenção do TJGO - empregou-se a técnica Entrevistas e o método de Análise de Conteúdo para a validação dos achados.

No Capítulo 8 - Conclusão, o autor retoma o objetivo deste trabalho e descreve sobre suas considerações decorrentes do estudo realizado.

Por fim, encontram-se as *Referências Bibliográficas* deste trabalho, seguida dos apêndices e anexo:

- Apêndice A - Teoria do Enfoque Meta Analítico (TEMAC) - Para condução da pesquisa, foram realizadas revisões da literatura com base na Teoria do Enfoque Meta Analítico (TEMAC), proposto por Mariano e Rocha [49];
- Apêndice B - Relatório de Acompanhamento para Projetos de Manutenção Evolutiva - Segue um modelo de relatório utilizado para manutenções evolutivas. Trata-se de um artefato simples, mas extremamente importante para gestão das manutenções evolutivas propostas no processo Scrum de Rehman et al. [3];

- Apêndice C - Questionários com Time de Analista de Sistemas Administrativos e com os Diretores da Engenharia de Software e Diretor de TI - No decorrer da pesquisa foram feitos questionários com o Time de TI e Diretores com propósito de diminuir o viés inerente das observações do pesquisador. Foram analisadas expectativas, grau de aceitação e disposição perante as mudanças. Diante dessas análises, buscou-se melhorar as chances dos bons resultados;
- Apêndice D - Questionários com os Usuários Gestores - O questionário com os usuários gestores teve a intenção de obter suas percepções quanto às mudanças no processo de manutenção. Foram analisados os graus de satisfação e aprovação;
- Apêndice E - Categorias observadas nas entrevistas e revisão bibliográfica - A Análise de Conteúdo é um conjunto de ferramentas utilizadas para se obter informações que podem não estar explícitas nos dados. Uma das ferramentas utilizadas é a *Categorização* do conteúdo obtida através das entrevistas e da revisão bibliográfica;
- Apêndice F - Termo de Compromisso e Perguntas para os Entrevistados - Trata-se do Termo de Compromisso lido para cada entrevistado antes do início das entrevistas;
- Apêndice G - Transcrição das Entrevistas com os Diretores - São as transcrições das entrevistas realizadas com os Diretores após a conclusão dos experimentos;
- Apêndice H - Transcrição das Entrevistas com os Usuários gestores - São as transcrições das entrevistas realizadas com os Usuários gestores após a conclusão dos experimentos;
- Apêndice I - Transcrição das Entrevistas com o Time de TI - São as transcrições das entrevistas realizadas com o Time de TI após a conclusão dos experimentos;
- Anexo I - Elogio Formal do Excelentíssimo Senhor 3° Juiz Auxiliar da Corregedoria - Geral de Justiça de Goiás em Virtude da Agilidade na Manutenção de um dos Sistemas Administrativos - este anexo tem o propósito de reforçar os resultados obtidos através das técnicas utilizadas nessa pesquisa de natureza aplicada.

Capítulo 2

Manutenção de Software

2.1 Considerações Iniciais do Capítulo

Neste capítulo apresenta-se um histórico sobre a evolução dos métodos ágeis em empresas públicas brasileiras, seguida de uma breve contextualização sobre manutenção de software e depois alguns trabalhos que visam processos e métodos voltados para manutenção os quais poderão observar, entre outras coisas, a importância de se tratar as diferentes vertentes da manutenção, realizar entregas contínuas e testáveis.

2.2 Métodos Ágeis em Empresas Públicas Brasileiras

No Brasil, início dos anos 2000, os métodos ágeis eram vistos com ceticismo tanto por pesquisadores quanto pela indústria [19]. Melo et al.[19] apresentaram uma visão geral da evolução do movimento ágil no Brasil, que se iniciou pelo contato de pesquisadores brasileiros em conferências internacionais como, *ACM- OOPSLA* (1999) em Denver, no Colorado, e *Primeira Conferência Internacional de Extreme Programming e Processos Ágeis de Engenharia de Software* (XP'2000), na Sardenha, Itália.

Assim, no início de 2001, professores e profissionais brasileiros começaram a ministrar palestras sobre *Extreme Programming* em universidades, departamentos de TI do Governo e empresas relacionadas à indústria de software [19]. Vários foram os eventos em 2002 e 2003, destacando-se o *Extreme Programming* no Brasil, com a primeira e

única visita de Kent Beck [1] ao país [19].

As cidades de São Paulo, Rio de Janeiro, Recife, Florianópolis, São Carlos e Brasília foram fundamentais para a disseminação do emprego dos métodos ágeis em projetos de desenvolvimento de software começando, assim, a ganhar forças na indústria e no comércio brasileiro [19].

Estava claro que mudanças nos processos de software estavam acontecendo. Desde 1993, com a criação do Programa Brasileiro da Qualidade e Produtividade de Software (PBQP Software), o Brasil investe na melhoria contínua da qualidade de software [50].

> PBQP Software procura estimular a adoção de normas, métodos, técnicas e ferramentas da qualidade e da Engenharia de Software, promovendo a melhoria da qualidade dos processos, produtos e serviços de software brasileiros, de modo a tornar as empresas mais capacitadas a competir em um mercado globalizado [51].

Entre os programas do PBQP Software, destaca-se o MPS.BR - Melhoria de Processo do Software Brasileiro, lançado em 2003, coordenado pela SOFTEX - Associação para Promoção da Excelência do Software Brasileiro, com apoio do governo brasileiro, da indústria de software e de instituições de pesquisa [52, 53].

Lima [54] apresenta que grande parte das melhorias propostas no MPS.BR podem ser alcançadas utilizando-se Scrum: 67% dos resultados esperados pelo MPS.BR são atendidos pelo Scrum, 25% são parcialmente atendidos, contra apenas 8% dos resultados que não são atendidos.

Historicamente, o Brasil desenvolveu soluções digitais utilizando-se do processo de "big design up front", que requer especificação dos requisitos completa e validada antes do início do desenvolvimento real. De acordo com Mergel (2016, apud Mantovani e Marczak [1]), essa estratégia trouxe experiências negativas e falhas de gestão.

Ainda hoje os desafios são grandes quando se trata de agilidade em empresas públicas brasileiras, inclusive nas universidades públicas, como demonstra o estudo de Maurício e Neris [55].

No setor público, os desafios para se empregar metodologia ágil estendem para aspectos pautados no caráter formal das comunicações, divisão do trabalho, hierarquização da autoridade, rotinas e procedimentos padronizados, separação do público e do privado. Assim, o emprego de métodos ágeis deve levar em conta as peculiaridades de cada meio para que não sejam simplesmente aplicados como um manual, e sim para que ocorra a transformação cultural e organizacional de forma a se alcançar um nível alto de adaptabilidade.

[1] "Kent Beck é um engenheiro de software americano criador do Extreme Programming, Test Driven Development e um dos fundadores do Manifesto Ágil".

Estudos apontam que um dos principais desafios para implantação de Métodos Ágeis é o de promover a mudança cultural na organização [56, 57, 30, 58]. Uma análise mais profunda da cultura organizacional pode auxiliar na gestão da mudança e, consequentemente, contribuir para a implementação de ações táticas do processo de inovação com as metodologias ágeis [56].

Rulinawaty e Lukman [29] relataram que a incapacidade da burocracia de se tornar ágil é causada por fatores como: tempo reduzido, custos e sobreposições políticas. Entretanto, eles [29] afirmam que, embora a estrutura hierárquica não ajude com os processos ágeis, esse não é o maior problema, e sim o fato de que as organizações, ao buscarem agilidade, tendem a desestruturar a hierarquia, esquecendo-se dos objetivos organizacionais [29].

Por outro lado, estudos recentes elaborados por Mantovani e Marczak [1] apontam que desde os anos 2000, o Brasil vem aderindo cada vez mais aos métodos ágeis, como indica a Figura 2.1, prática que se observa pelo mundo todo como uma das principais formas de inovação digital.

Método Ágil	Porcentagem de respostas (%)
Scrum	73.1
Kanban	58.1
Hybrid customized	18.6
Iterative development	18.6
Scrum/XP hybrid	15.6
XP	15.0
Lean Development	10.8
Scrumban	10.2
Lean Startup	5.4
Others	12.6

Figura 2.1: Tendência dos métodos ágeis no Brasil em 2020. (Fonte:[1])

A aplicação de ágeis no governo brasileiro é, em sua maioria, bem-sucedida e, normalmente, conduzida em combinação com outras abordagens ágeis de desenvolvimento de software ou, em sua maioria, uma combinação de ágeis com métodos tradicionais [1]. Alguns exemplos, como:

- Santos et al. [59] relataram a aplicação da metodologia Ágil Kanban para Manutenção de Software em um órgão público brasileiro. A escolha do Kanban se deu pela capacidade de adaptação às frequentes mudanças nas requisições de ma-

nutenção de software pelo órgão junto ao fornecedor, prestador de serviços de manutenção.

- Noronha et al. [57] utilizaram o método ágil Kanban de forma adaptada junto em um Ministério brasileiro. O desafio foi implementar Kanban para apoiar a gerência de pedidos de manutenção à fábrica de software, que prestava serviços de manutenção de software. Foi empregada a pesquisa-ação. Os resultados permitiram constatar que o novo processo atenuou problemas, diminuiu a resistência às mudanças, e contribuiu para aumentar a satisfação dos envolvidos.
- Moraes [60] apresentou uma proposta de melhoria para o Processo de Desenvolvimento de Software do Exército Brasileiro, baseado em técnicas do modelo Ágil e Gestão de Riscos em conjunto com o guia PMBOK. O propósito foi atender de forma rápida e com qualidade as demandas da Administração Pública, que cada vez mais exigem agilidade na produção de software. Após a aplicação do modelo, constataram que os indicadores de desempenho melhoraram, assim como a satisfação dos clientes.

Vacari e Prikladnicki [58] buscaram evidências científicas sobre os benefícios da utilização de métodos ágeis na Administração Pública brasileira, em detrimento ao histórico de fracassos de projetos de TI no governo. Segundo os autores, as motivações para a adoção de ágeis incluem: (1) uma entrega rápida de valor ao cliente; (2) uma maior colaboração entre TI e negócios; (3) uma maior satisfação do cliente. Além disso, os autores [58] chamam a atenção para um fato novo: (4) aumento na moral da equipe de TI do governo, reduzindo a dependência de empresas contratadas.

De maneira secundária, outros benefícios têm sido encontrados, incluindo melhorias: (1) comunicação entre os membros da equipe de desenvolvimento, bem como, entre desenvolvedores e clientes; (2) aprendizado de novas tecnologias; (3) qualidade do produto; (4) visibilidade do projeto; (5) produtividade das equipes; (6) redução de custos; (7) capacidade de gerenciar as mudanças e as prioridades; (8) conformidade com exigências burocráticas do governo [58].

2.3 Manutenção de Software com Metodologias Ágeis

A corrida por informações e conhecimentos vem acompanhada por uma rotina de mudanças.

> Mudanças acontecem ao se adicionar, alterar ou encerrar recursos. Mudanças acontecem para corrigir defeitos, aumentar a segurança ou melhorar o desempenho. Mudanças acontecem para melhorar a experiência dos usuário, para adotar novas tecnologias, novas plataformas, novos protocolos, novos padrões. Acontecem para fazer com que os sistemas funcionem juntos, mesmo que eles não tenham sido projetados para isso [61].

A cada nova mudança pode ser necessária a elaboração de um novo projeto de software. Muitas vezes esses projetos são feitos intuitivamente, sem formalização adequada e principalmente sem controles apropriados. As boas práticas de gerenciamento de projetos vieram para ajudar a organizar as ideias e então otimizar o serviço, gerando resultados de alta qualidade em menor tempo, custo e risco.

De acordo com o *Project Management Institute* (PMI), projeto é definido como: "um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo". Essa definição leva em consideração as características de ser temporário, com início e fim bem definidos, exclusivo, como produto ou capacidade de produzir algo, e progressivo, em que se estabelecem etapas e progride por incrementos [62].

Isso envolve áreas de conhecimento como planejamento, gerência de custos e tempo, organização de recursos humanos (equipe) e garantia da qualidade do produto e do processo.

Essas áreas de conhecimentos estão alinhadas à Engenharia de Software que não se limita apenas aos processos técnicos de desenvolvimento, mas também proporciona modelos, formalismos, técnicas e ferramentas para o desenvolvimento de software com qualidade [15].

A abordagem sistemática e organizada promovida pela Engenharia de Software é comumente chamada de processo de software e conforme Sommerville [15], um processo de software é uma sequência de atividades que leva à produção de um produto de software.

Existem quatro atividades fundamentais comuns a todos os processos de software. São elas [15]:

- Especificação de software;
- Desenvolvimento de software;
- Validação de software;

- Evolução de software.

Os modelos de processos de software convencionais, como: cascata, espiral, prototipagem, incremental, Desenvolvimento de Aplicação Rápida (RAD), entre outros, especificam as atividades, saída e artefatos em diferentes fases do ciclo de vida do sistema e, assim, consideram que os sistemas possuem início, meio e fim, como ocorre com outros projetos.

Entretanto, o que se observa é que, embora o desenvolvimento do sistema termine, a manutenção continua ao longo de seu ciclo de vida e nem sempre ela pode esperar que os critérios de projeto sejam atendidos para que se inicie.

Os serviços de manutenção em software exigem respostas rápidas, sejam elas corretivas ou evolutivas, pois ao se juntar o código alterado ao código original o sistema pode se comportar de forma inesperada e novas manutenções podem surgir, inclusive, de forma urgente. Assim, uma dificuldade bastante comum, encontrada em projetos tradicionais de desenvolvimento de software ou manutenções evolutivas, é a capacidade de resposta à mudanças.

Desde o seu surgimento, a Engenharia de Software vem evoluindo, acompanhada de melhorias trazidas por ferramentas CASE, orientação a objetos, linguagens visuais, processos unificados, desenvolvimento web, entre outros. Cada vez mais os cidadãos pressionam os governos para que atendam mais rapidamente às suas necessidades que envolvem, entre outras coisas, as plataformas de redes sociais, comércio eletrônico, alta disponibilidade, acessos personalizados e disponíveis nos computadores portáteis dos mais diversos tipos.

Além disso, tendências do setor de tecnologia que envolvem análise de dados, aprendizado de máquina, redes direcionais de alta velocidade (5G), internet das coisas, entre outras, conduzem a formação de capacidades internas e conjuntos de habilidades para avaliar, responder e implementar novas tecnologias e processos [30].

Nesse sentido, uma das importantes formas de se promover a transformação digital nessas últimas décadas envolve a aplicação de métodos ágeis [1] como ferramenta capaz de promover qualidade, integração e respostas rápidas que o mercado requer.

Em 2001, o Manifesto Ágil [63] trouxe consigo os princípios para o desenvolvimento ágil de software que ganhou popularidade no mundo todo e vem ajudando os engenheiros de software a atingirem as expectativas dos clientes. Ele trouxe consigo, além de outros princípios, aqueles que favorecem a manutenção, tais como: processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente, mudanças são bem-vindas mesmo tardiamente no desenvolvimento, software funcionando é a medida

primária de progresso, contínua atenção à excelência técnica e bom design aumenta a agilidade, as melhores arquiteturas emergem de equipes auto-organizáveis, entre outros [63].

O foco do desenvolvimento ágil é a produção de software em alta qualidade, em um prazo que maximize seu valor para o negócio [64].

Desde o modelo em espiral, se assume a mudança como risco do projeto e atividades são propostas para a redução desse risco. A mudança é tratada como um problema. Nos processos ágeis houve uma atenção especial às mudanças e elas se tornaram parte da solução e não apenas do problema.

Para Bass et al. [61]: "Se a mudança é a única constante no universo, então a mudança de software não é apenas constante, mas onipresente".

De modo geral, os diversos modelos de desenvolvimento que surgiram após a chamada "crise do software" de 1970 lidam com a manutenção como uma fase dentro do processo de desenvolvimento. Entretanto, estudos apontam que a manutenção possui características próprias e muitas vezes a inobservância dessas diferenças pode gerar ainda mais manutenções [25, 32, 30, 3, 65].

Embora a manutenção possa ser considerada como uma continuação de novos desenvolvimentos, há uma diferença fundamental entre as duas atividades:

"No desenvolvimento de algo novo, mesmo com certas restrições, existe um campo aberto, livre, para as possibilidades. Já a Manutenção normalmente ocorre sob um ambiente de restrições e parâmetros pré-existentes" [17].

Essas mudanças podem ser definidas como frequentes manutenções evolutivas ou corretivas e, assim, algumas pesquisas são realizadas na busca de se minimizar os impactos negativos da manutenção e também com o objetivo de otimizar os resultados do produto de software. Nesse sentido, Pressman[66] salienta que: "não é raro uma organização de software despender de 60% a 70% de todos os recursos com manutenção de software".

Heeager e Rose [25], em 2015, afirmaram que:

"Os métodos ágeis foram bem aceitos e são muito utilizados no desenvolvimento de software e estão começando a ser utilizados nas equipes de manutenção. No entanto, há relativamente poucas pesquisas sobre ágeis na manutenção".

Atualmente, as pesquisas em manutenção ainda são poucas, conforme pode-se ver na revisão realizada (Apêndice A), principalmente quando comparadas ao desenvolvimento, entretanto, já existem fundamentações científicas de que os métodos ágeis podem ser utilizados com sucesso nas manutenções, com as devidas adaptações.

Assim, durante a revisão da literatura (Apêndice A), traçou-se um histórico dos últimos vinte anos (Figura 2.2) de alguns métodos ágeis, aplicados à manutenção de software, dando-se maior ênfase em temas recentes (últimos cinco anos).

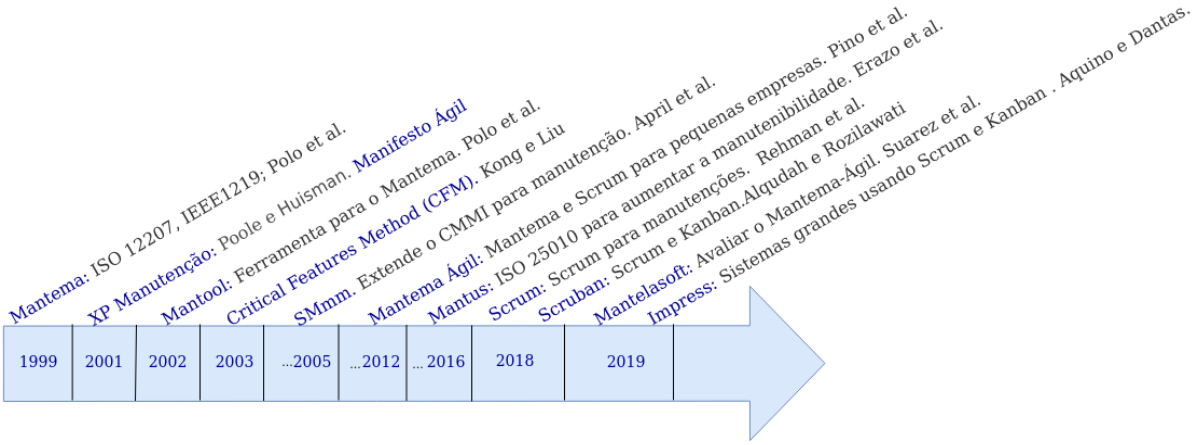


Figura 2.2: Artigos sobre manutenção de software nos últimos vinte anos (Fonte: autoria própria).

A seguir, apresenta-se os processos e métodos principais para se atingir os objetivos desta pesquisa.

2.3.1 Mantema

Ao final dos anos noventa, ainda na geração dos modelos de processos tradicionais de software, o Mantema [21] surge com propósitos específicos para manutenção de sistemas de médio e grande porte, especialmente os terceirizados como, por exemplo, os relacionados às instituições financeiras ou administrações públicas. Para atingir seus objetivos ele utiliza as normas de referência: ISO/IEC 12207 [67] e IEEE 1219 [68].

A ISO/IEC 12207 [67] possui grupos de atividades que podem ser usadas em todo ciclo de vida do software, para diferentes tipos de projetos, mas não detalha como implementar as atividades dependendo do tipo de manutenção que se irá fazer. Além disso, todas as atividades são agrupadas e distribuídas em sequência.

A falta de clareza de certas atividades da ISO/IEC 12207 referentes à manutenção, pode prejudicar os mantenedores no momento de escolherem quais atividades devem seguir. Essa dificuldade chamou a atenção de Polo et al. [21] que consideram este um fator crítico para o sucesso na manutenção de software.

A IEEE 1219 [68] atualizada pela ISO/IEC 14764 [69] serve como complemento no processo de manutenção da ISO/IEC 12207, descrevendo com mais detalhes as atividades

necessárias às manutenções de software, sejam elas corretivas, adaptativas, perfectivas ou preventivas.

Desse modo, para Polo et al. [21], as normas ISO/IEC 12207 [67] e IEEE 1219 [68] deveriam ser unidas e detalhadas com foco na manutenção. Essa união bem sucedida deu origem ao Mantema, que divide a manutenção em cinco tipos: Corretiva e urgente, Corretiva sem urgência, Evolutiva, Preventiva e Adaptativa. Além disso, descreve de forma prática e objetiva:

- O fluxo de atividades e tarefas a serem executadas;
- As entradas e saídas das atividades;
- As responsabilidades;
- A interface com outros projetos.

Mantema deu origem a diversos estudos posteriores como, por exemplo, as referências: [70, 71, 48, 72]. Este último, se refere ao Mantema Ágil [72] que une características do Mantema ao Scrum e será melhor detalhado na Seção 2.3.3.

2.3.2 Critical Feature Method (CFM)

Kong e Liu [23] chamam a atenção para manutenções críticas ao desenvolver o método *Critical Feature Method* (CFM). Trata-se de um método leve, adaptado para pequenas e médias empresas, que leva em consideração a criticidade da manutenção, principalmente em sistemas WEB, e que pode ser usado para manutenção durante situações de emergência, recuperação pós-emergência e ciclo normal de desenvolvimento do sistema. Os autores afirmam que embora existam muitas metodologias projetadas para lidar com incertezas, como as próprias metodologias ágeis, RUP e outras, elas não são adequadas para fazer alterações em situações de emergência que requerem manutenção em curto tempo, inclusive, aquelas que possam ser provenientes de ameaças de segurança ou restrições legais, como, por exemplo, os sistemas WEB que podem ultrapassar fronteiras internacionais [23].

Em tais condições, a prioridade é impedir a ameça e cumprir os requisitos legais primeiro, mesmo com o risco de perder algumas funcionalidades existentes. Segundo Kong e Liu [23], as funcionalidades perdidas podem ser recuperadas após o período de emergência, sendo que para isso basta utilizar-se de uma manutenção adicional capaz de recuperar as principais funcionalidades e, a partir daí, retornar ao ciclo normal de desenvolvimento e manutenção.

O *Critical Feature Method* (CFM) [23] baseia-se no fato de que o enfrentamento de uma situação de emergência não pode desperdiçar tempo com pesadas documentações de requisitos presentes nas metodologias tradicionais. Em vez disso, deve-se focar no desenvolvimento de soluções viáveis, com o mínimo de interrupções das funcionalidades existentes dentro de prazos curtos e com miníma participação dos clientes. Diante disso, o CFM elenca os principais valores:

- Comunicação entre executivos e desenvolvedores;
- Custo mínimo em comunicação e desenvolvimento;
- Desenvolvimento rápido;
- Documentação simples.

Para tais valores, destacam-se a aplicação das práticas:

- Particionar a manutenção em três estados:
  - Estado de emergência - dado que uma emergência aconteceu, interromper todo o processo e priorizar a emergência. Essa *situação reativa* substitui o planejamento e o *timeboxing* substitui o cronograma. Alguns recursos não críticos do sistema podem ser desativados temporariamente ou abandonados para que os recursos críticos sejam implementados [23];
  - Posterior a emergência - o principal objetivo durante esse estado é restaurar as funcionalidades desativadas ou perdidas durante o estado de emergência. A Documentação deve ser pouca nesse estado. O gerenciamento da configuração pode ser necessário, dependendo do tamanho do projeto. Quando as funcionalidades do sistema voltarem ao normal, o sistema retornará ao Estado de evolução [23];
  - Estado de evolução normal - a maior parte do tempo um sistema permanece em evolução. Envolve ações com gerenciamento de conteúdo, manutenção preventiva, manutenção adaptativa, refatoração de código e reengenharia [23].
- Estreitar a colaboração entre executivos de negócios e desenvolvedores;
- Mesclar modelagem de negócios, análise de requisitos e projeto em uma fase de par-prototipagem (entre o analista e o cliente);
- Fazer uso do processo de *design* ágil usando prototipagem em papel;

- Substituir a documentação formal por matrizes leves separadas em situações críticas e não críticas. O Manifesto Ágil [63] destaca que: *software funcionando é mais valorizado do que documentação muito abrangente* e, desse modo, algumas metodologias sugerem que a documentação seja, simples, podendo até mesmo ser substituída por artefatos descartáveis. Assim, o modelo CFM propõe uma "Matriz de Recursos Críticos" e uma "Matriz de Recursos Normais" para substituir o formalismo das documentações de requisitos, desenvolvimento e testes [23].

2.3.3 Mantema Ágil

Após o manifesto ágil [63], Pino et al. [72] observaram a necessidade de se adaptar o Mantema para utilização da metodologia ágil Scrum, resultando em uma proposta metodológica para manutenção de software focada em pequenas empresas chamada por ele de Mantema Ágil.

Essa metodologia define uma estratégia de manutenção ágil, estabelecendo em detalhes o que deve ser realizado, quando, como e por quem. Procura fornecer um guia detalhado para a implementação do processo de manutenção nesses tipos de empresas.

O Mantema Ágil também estabelece um conjunto de elementos, como tipos de manutenção, níveis de serviço e níveis de capacidade, que visam: (i) lidar com a complexidade que é inerente ao processo de manutenção e (ii) permitir que uma pequena empresa defina seu próprio processo de manutenção, levando em consideração suas características e necessidades particulares em um esforço para melhorar a manutenção [72].

Para Pino et al. [72], diferentemente de outros métodos ágeis, como Programação Extrema e Programação Pragmática, o Scrum fornece uma estrutura para gerenciar projetos sem incluir práticas técnicas, enquanto que os demais se concentram nas descrições práticas das atividades e técnicas relacionadas ao ciclo de desenvolvimento de software. Esses aspectos possibilitam combinar a estrutura de gerenciamento de projetos do Scrum com processos técnicos de uma área de conhecimento específica como o Mantema, que passa a usufruir dos benefícios de uma metodologia ágil.

2.3.4 Mantelasoft

Com base no Mantema-Ágil [72], na norma ISO/IEC 12207 [67] e com propósito de torná-lo mais prático para aplicação em pequenas e médias empresas, respeitando as características e necessidades peculiares desse setor, Suarez et al. [48], desenvolveram o Mantelasoft, que foi empregado com sucesso na cidade de Pereira (Colômbia).

Durante suas pesquisas e avaliações, Suarez et al. [48] constataram que o Mantema junto aos ágeis é o preferido para a realização das manutenções por causa de sua qualidade, competitividade e histórico, sendo um processo de muito sucesso nos últimos dez anos. Entretanto, trata-se de um método complexo que requer tempo e recursos humanos consideráveis para sua implementação. Esses foram os fatores que os motivaram a desenvolver o Mantelasoft, que funciona como um guia ao Mantema-Ágil, para que, de forma pontual as tarefas e atividades necessárias à manutenção de software em pequenas e médias empresas.

O Mantelasoft [48], assim como Mantema-Ágil [72], utiliza o Scrum para alcançar melhor desempenho e qualidade nas entregas parciais e periódicas. Além disso, por se tratar de um guia passo a passo de tarefas e atividades, ele consegue tornar a manutenção de software um processo controlável e mensurável.

O guia Mantelasoft [48] se divide em oito fases, sequenciais, que são: manutenção geral, planejamento da manutenção, processo de manutenção, métricas de manutenção, estimativa de custos de manutenção, pessoal de manutenção, evolução ou melhoria dos produtos e atividade finais de manutenção. Essas fases, por sua vez, possuem tarefas e atividades que instruem sobre a maneira de se realizar a manutenção utilizando-se de um *check-list* [48].

Como resultados, os autores [48] afirmam que esse guia vem contribuindo para a melhoria do processo de manutenção de software, área que se tornou vital e que tem possibilitado que aqueles que a realizam de forma sistemática alcancem muitos benefícios. Os autores citam como exemplo: aumento na vida útil dos sistemas, satisfação dos clientes e adequação dos sistemas às mudanças geradas pelo negócio [48].

2.3.5 Kanban

Kanban é uma ferramenta que utiliza da informação visual. Surgiu com foco na manutenção, sendo usada a princípio para manutenções dos veículos da Toyota e depois se estendeu para diversos tipos de manutenção, inclusive para as manutenções de software.

As empresas de software estão cada vez mais adotando Kanban para manutenção, pois trata-se de uma ferramenta que se afirmou depois de demonstrar bons resultados, por oferecer visibilidade aprimorada do projeto, qualidade no software, motivação na equipe, melhor comunicação e colaboração [73].

Assim, Ahmad et al. [73] pesquisaram os motivos de algumas empresas finlandesas estarem deixando o Scrum para utilizarem o Kanban na manutenção dos sistemas. Seus resultados mostraram que as equipes de manutenção, ao utilizar Scrum, enfrentaram

desafios como falta de visibilidade do trabalho, dificuldades em priorizar as tarefas, falhas na comunicação e colaboração, sobrecarga devido aos compromissos exigidos pelas sprints, dificuldades em sincronizar os trabalhos ou substituir pessoas.

As entrevistas realizadas por Ahmad et al. [73] demonstraram que as equipes de manutenção não conseguiam encaixar as atividades urgentes nas *sprints* pré-estabelecidas e as datas de lançamentos eram quase aleatórias. Além disso, as reuniões diárias, muitas vezes, duravam bem mais do que o estabelecido e muitas vezes os membros do time não estavam disponíveis para comparecerem nas reuniões. Tais fatores foram resolvidos ao se utilizar o Kanban.

Após a aplicação do Kanban em alguns estudos de caso, Ahmad et al. [73] concluíram que, em resumo, para lidar com o desafio da falta de visibilidade, o Kanban é mais adequado sendo indicado para onde exista um alto grau de variabilidade da prioridade, como na manutenção em ciclos curtos. Nesse sentido, Kanban contribuiu para benefícios como visibilidade e priorização das tarefas, melhoria na moral da equipe, melhoria na comunicação, proteção das equipes quanto às exigências excessivas e compartilhamento de conhecimento entre os envolvidos.

Entretanto, pesquisas apontam que o Kanban, assim como outros métodos, possui significantes desafios que impactam negativamente no processo de desenvolvimento de software, como falta de mecanismo de rastreamento do andamento do projeto, dificuldades em se estimar o WIP (*Work In Progress*), atrasos e, consequentemente, falhas nos projetos [74].

Nesse sentido, Alaidaros et al. [74] propuseram melhorias no processo utilizando Kanban que estendem o rastreamento do progresso, geram limites ótimos de WIP e maior visão do fluxo de trabalho. Para isso, utilizaram métodos tradicionais de gerência de projetos, e algoritmos otimizados para determinação do melhor WIP.

Eles [74] destacaram que muitos profissionais estão utilizando métodos ágeis misturados inclusive com outras técnicas, como as de gerência de projetos tradicionais, ou métodos ágeis com outros métodos ágeis, como o método que une Kanban e Scrum, formando o Scrumban [75], visto na Seção 2.3.7 e o Impress [22] mostrado na Seção 2.3.8.

2.3.6 Scrum

Heeager e Rose [25] revelaram os desafios de se adotar Scrum para processos de manutenção. Eles [25] relataram que muitos dos problemas de se aplicar Scrum para manutenção se dão pelo fato de que uma emergência não pode aguardar o cumprimento de uma *sprint*

com todos seus requisitos, e quando acontecem acabam atrasando o cumprimento das metas. Assim, afirmaram que a aplicação dos métodos ágeis na manutenção pode não ocorrer automaticamente e algumas otimizações podem ser necessárias para acomodar os diferentes tipos de manutenção, diferentes fontes de trabalho e práticas locais.

Para tratar a urgência das manutenções corretivas, Heeager e Rose [25] sugerem que os projetos evolutivos sejam planejados com um tempo extra para que, caso ocorra alguma emergência, seja possível atendê-la sem muitos impactos na manutenção evolutiva em curso.

Pino et al. [72] aconselham o uso de sprints separadas para a manutenção, sendo uma mais breve para correções de emergências imprevistas e outra maior com planejamento para manutenções evolutivas.

Verificando que a literatura sugere o tratamento diferenciado para as diferentes categorias de manutenção (adaptativa, corretiva, perfectiva e preventiva), Rehman et al. [3] constataram que quando isso é atendido os resultados da manutenção com métodos ágeis aumentam os casos de sucesso.

Diante do exposto, Rehman et al. [3] abordaram esse problema da concorrência das manutenções se baseando nos trabalhos de Heeager e Rose [25] e Pino et al. [72], utilizando-se do método ágil Scrum, e propuseram o modelo conforme a Figura 2.3.

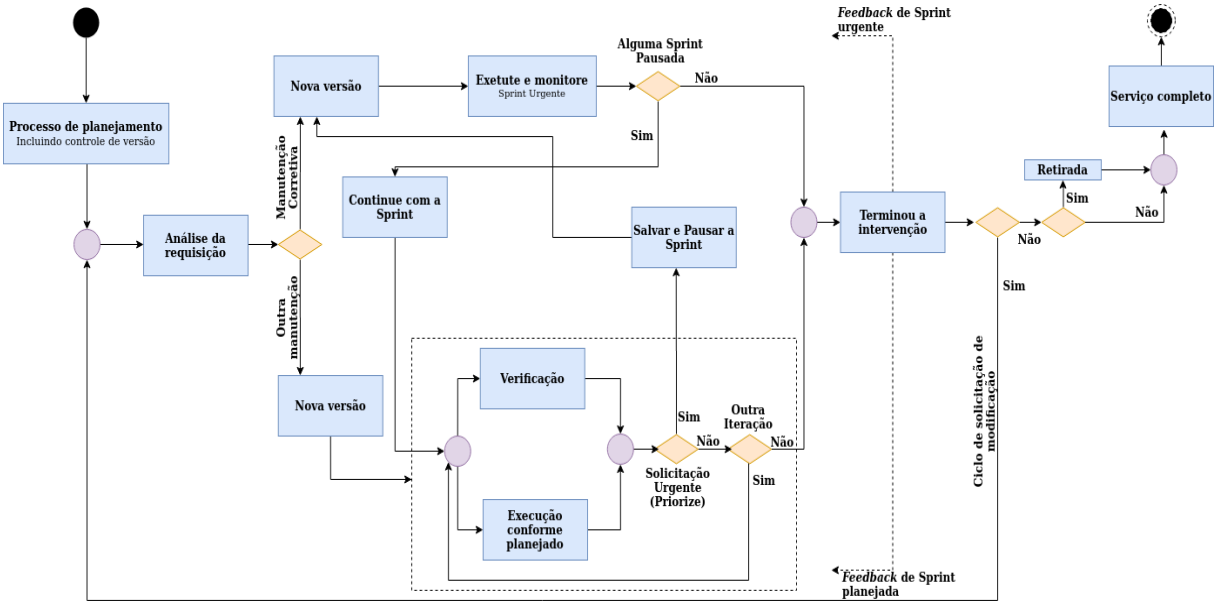


Figura 2.3: Modelo utilizando Scrum para mautenção de software (Fonte:[3]).

O modelo se inicia com o planejamento para rastreamento de alterações no sistema

e logo após verifica-se o tipo de manutenção, sendo que a manutenção corretiva é prioritária e o trabalho é iniciado por meio de uma nova versão de código [3].

Caso a manutenção seja evolutiva, surge a necessidade de um monitoramento para verificação de *sprints* urgentes e de emergência durante sua execução e, caso ocorra a emergência, a *sprint* atual é pausada, o trabalho realizado é armazenado no sistema de controle de versão e é iniciada a manutenção corretiva [3].

Após as manutenções corretivas, é verificado se há solicitações pausadas e então as retorna de onde foram interrompidas.

Rehman et al. [3] esclarecem ainda que a manutenção deve ser realizada da seguinte forma:

- Planeje a manutenção levando em consideração situações urgentes e não urgentes. Não é necessário fazer todas as iterações com a mesma duração (como estabelece o Scrum), dependendo da natureza do trabalho em manutenção, pequenas iterações devem ser permitidas. Controle cada iteração com uso de ferramentas de controle de versão;
- Mantenha o cliente sempre presente para orientar sobre emergência e prioridades.
- Execute cada tarefa de manutenção, corretiva e evolutiva, de forma separada e então quando estiverem estáveis junte-as ao sistema original.
- Documente a iteração para manter a integridade do sistema utilizando-se de casos de uso ou casos de teste.
- A manutenção corretiva e a urgente devem ser testáveis: testes de integração, regressão e outros.

O trabalho de Rehman et al. [3] foi conferido por meio de um estudo de caso e comprovou que o Modelo de Manutenção aumentou a satisfação do cliente, diminuiu o retrabalho, melhorou a rastreabilidade dos erros e melhorias nos casos de testes. Assim, consideraram que a meta de manutenção com o Scrum é possível e satisfatória.

2.3.7 Scrumban

Scrumban [75] é o resultado da união entre Scrum e Kanban com propósito de se aproveitar o melhor de cada um em prol de uma maior flexibilização do emprego das práticas ágeis em diferentes situações de projeto.

Segundo os estudos de Alqudah e Rozilawati [4], os profissionais de Kanban e Scrum estão convencidos de que para certas situações a combinação de ambos os métodos é melhor que o uso de um ou de outro isoladamente. Entretanto, não é fácil selecionar as características que serão utilizadas de cada um e para se escolher Scrum, Kanban ou ambos exige-se um entendimento dos fatores que influenciam a seleção ou hibridização.

Não existem práticas específicas para Scrumban, os membros da equipe ágil precisam entender quais práticas de Scrum e Kanban agregam mais valor e escolher as práticas apropriadas para a situação. Para Alqudah e Rozilawati [4] Scrumban é mais adaptável, especialmente quando há muitas alterações nos requisitos do usuário.

Alqudah e Rozilawati [4] avaliaram o Kanban, Scrum e Scrumban quanto à quantidade de prescrição de cada um, papéis e responsabilidades, tempo de adoção, tamanho da equipe, tamanho do projeto ou tarefa, priorização de requisitos,prazo de entrega, práticas técnicas, custos e qualidade. Eles [4] verificaram que o Scrumban é bastante adaptável, especialmente em situações nas quais ocorrem constantes alterações de requisitos.

Com propósito de se comparar características do Scrum, Kanban e Scrumban, Alqudah e Rozilawati [4] definiram resumidamente a Figura 2.4:

Os seguintes casos foram apontados por Algudah e Rozilawati [4] para indicar quando utilizar cada método:

- Kanban: quando os envolvidos não estão dispostos a atuarem com papéis pré-definidos; equipes que precisam executar tarefas prioritárias em menos de uma semana, constantemente ou mesmo em questões de horas com entregas pequenas; em equipes que precisam diminuir o *lead time* (tempo de entrega), aumentar a qualidade e reduzir custos.

- Scrum: quando os envolvidos demonstram preferência na adoção dos métodos recomendados; quando estão dispostos a exercerem os papéis prescritos; para entregas maiores que duram uma ou mais de uma semana; situações que o tempo de espera pode ser maior; disseminação dos conhecimentos na equipe.

- Scrumban: as escolhas para sua composição se dão para cada caso particular. Por exemplo, poderia iniciar o projeto adotando o Scrum evitando *sprints*, planejamento, revisão e *backlog* da *sprint* quando a tarefa é pequena para caber em mais de uma *sprint* ou quando é difícil estimar o tamanho das *sprints* e então poderiam controlar a entrega ajustando o WIP. Da mesma forma, podem existir muitos cenários e aplicações diferentes, ficando a cargo da equipe ajustar o método Scrumban

Critério	Scrum	Kanban	Scrumban
Prescrição	Prescritivo [21].	Não prescritivo e flexível [3],[5].	Não prescritivo e flexível [45],[47]
Regras e Responsabilidades	Predefinidas [7],[21],[41],[44],[54].	Não predefinido [4], [5], [44].	Baseado na decisão do time [48],[50],[51].
Tempo de adoção	A transição é um pouco desafiadora [6]. Todavia, as organizações Ágeis parecem adotar o Scrum antes do Kanban [15].	A transição para o Kanban é fácil [6], especialmente quando os times migram de métodos estruturados como o waterfall [15].	Experiência do time com Scrum e Kanban são necessária para combiná-las [45],[51].
Tamanho do time	Um time é composto de 5-11 membros com regras predefinidas [21],[22]. Assim, para times grandes, Kanban pode ser melhor que Scrum[33].	Mais flexível que Scrum quando considera o tamanho do time [33]; Times podem ser menores que 5 membros [55], ou pode ser maior que 11 (até 14 membros) [33].	Mais flexível quando comparado com Scrum e de certo modo parecido com Kanban [45],[51].
Tamanho da "carga" (tamanho WIP)	Possui maior capacidade de carga de trabalho se comparado ao Kanban [5],[15] e é requerido ao time entregas em sprints de tempo [5],[41], [54].	A capacidade de carga de trabalho é pequena [5], [6], [15], [54]. Em qualquer hora a entrega de itens urgentes podem ser feitas [5], [44].	Baseado na decisão do time [45],[51].
Priorização de requisitos	Priorização de requisitos são baseados no tamanho da sprint [5], [44]	Priorização de requisitos são feitos de forma contínua, em qualquer momento [5], [44].	Baseado na decisão do time [45], [51].
Tamanho das entregas	Entregas de tamanhos pequenos [6],[15].	O tamanho da entrega é pequeno quando comparado ao Scrum [6],[15].	Baseado na decisão do time [51].
Tempo de espera (*lead time*)	Scrum evita reduzir o tempo de espera ao contrário do Kanban [28].	Kanban reduz o tempo de espera evitando multitarefas e limitando o WIP [15],[16],[29],[41]	Pequeno e considerado melhor que Scrum e Kanban [46],[47],[51].
Práticas técnicas	Sem prática técnicas [44],[56].	Kanban também não tem práticas técnicas [39],[57]	Assim como Scrum e Kanban, Scrumban não tem práticas técnicas [45],[51]
Custo	Ao contrário do Kanban, o Scrum evita economia de custos, mas se concentra mais no conhecimento, experiência e tomada de decisão com base no que é conhecido [10]	Concentra-se na redução de custos, especialmente para operações [15],[58]	Concentra-se mais no corte de custos em comparação com Scrum e Kanban [49].
Qualidade	A reunião de revisão da sprint é a principal prática para melhorar a qualidade no método Scrum [59]	Kanban se concentra mais em melhorar a qualidade quando comparado ao Scrum [15], [28].	A qualidade será maior quando se usa Scrumban [51].

Figura 2.4: Comparação entre Kanban, Scrum e Scrumban (Fonte:[4]).

para cada situação. Para isso, deve-se entender as práticas de ambos os métodos para poder selecionar quais usar.

Algudah e Rozilawati [4] apresentaram diferentes estudos de caso sobre seleção de Scrum, Kanban e Scrumban e mostraram como o Scrumban pode ser formado para diferentes situações com sucesso baseando-se nas práticas do Scrum e Kanban.

2.3.8 Impress

Com o propósito de se manter um sistema de grande porte, com alto fluxo de mudanças, em funcionamento por mais tempo, justificar os altos custos desses sistemas em produção e adaptá-los às mudanças do negócio, Aquino e Dantas [22] propuseram o Impress.

O Impress é um Scrumban, que sustenta que demandas diferentes devem ser tratadas por meio de estratégias de manutenções diferentes, de modo a respeitar suas peculiaridades quanto às definições de escopo, tempo, partes envolvidas, complexidade, urgência e tamanho das solicitações. Assim, as diferentes demandas são divididas em duas categorias: Sustentação e Evolução [22].

De acordo com Aquino e Dantas [22], Impress se baseia nos princípios e conceitos amplamente aceitos pelas áreas de conhecimento em Manutenção e Evolução de software e princípios e práticas das metodologias ágeis, sendo as principais Scrum e Kanban.

Quanto à manutenção corretiva, o Impress se sustenta nos seguintes pilares [22]:

- *Deploy* contínuo e rápido por meio de técnicas de priorização de tarefas;
- Visibilidade e controle do fluxo de tarefas por meio do quadro Kanban e suas práticas;
- Engajamento ponta-a-ponta por meio da participação de todos os envolvidos no ciclo de vida das tarefas;
- Monitoramento contínuo com metas de curto prazo (semanais), reuniões diárias e manutenção em tempo real de métricas e Dashboards públicos.

Quando se trata de manutenção evolutiva, o Impress se baseia nas práticas do Scrum e enfatiza as dificuldades de se eleger o *Product Owner* (PO) em instituições em que os sistemas de grande porte são compartilhados por muitos POs e, desse modo, o Impress recomenda a criação de comitês envolvendo representantes das diversas áreas para a tomada de decisões. As demais práticas do Scrum se adéquam perfeitamente às demandas direcionadas ao fluxo de sustentação [22].

Ao final, Aquino e Dantas [22] enfatizam a importância de se administrar bem a mudança, visto que equipes distintas trabalham com prazos e processos diferentes para alterar evolutivamente e corretivamente o mesmo sistema.

2.4 Considerações Finais do Capítulo

Neste capítulo foram apresentados um histórico sobre a evolução dos métodos ágeis no Brasil, seguida de uma breve contextualização sobre a manutenção de software em que foram apresentados alguns métodos voltados para manutenção. Esses método proporcionam observar a evolução dos trabalhos buscando o aperfeiçoamento das técnicas de acordo com a dinâmica dos ambientes onde serão implantados. No Capítulo 3 apresentam-se conceitos importantes para se sustentar a qualidade e agilidade pressupostos dos métodos ágeis.

Capítulo 3

Arquitetura de Software e Testes

3.1 Considerações Iniciais do Capítulo

Neste capítulo, apresenta-se inicialmente conceitos de Arquitetura de Software, em seguida, um destaque maior é dado para o segmento de arquitetura de *deployment* e por fim, apresenta-se conceitos de testes utilizando-se métodos ágeis, diante da significante importância de ambos para os processos que se utilizam de métodos ágeis.

3.2 Arquitetura de Software

Para utilização dos métodos ágeis para manutenção, de acordo com a realidade de cada instituição, são necessárias ferramentas para suportar a agilidade necessária das manutenções emergenciais, dado que a excelência técnica e um bom design aumentam a agilidade [63].

> Criar uma arquitetura de software é uma tarefa crítica no desenvolvimento de sistemas de software porque as estruturas do projeto irão ditar se o sistema exibirá boa modificabilidade, por desempenho, interoperabilidade e outras qualidades [76].

Para muitos sistemas, atributos de qualidade, como: desempenho, confiabilidade, segurança e modificabilidade, são importantes para certificar que o software funciona corretamente. A capacidade de um sistema de software produzir resultados corretos não é útil se levar muito tempo para fazer isso ou se o sistema não permanecer ativo tempo suficiente para entregá-lo, ou mesmo se o sistema revelar os resultados para pessoas indesejadas. Essas preocupações são abordadas nos requisitos arquiteturais [5].

Existem muitas definições para arquitetura de software, seguem algumas:

- A arquitetura de software de um sistema é o conjunto de estruturas necessárias sobre o sistema, que compreende elementos de software, relações entre eles e propriedades de ambos [61].
- Arquitetura é um conjunto de significantes decisões sobre: a organização de um sistema de software; a seleção de elementos estruturais e suas interfaces pelas quais o sistema é composto; juntamente com seu comportamento, conforme especificado nas colaborações entre esses elementos; a composição desses elementos estruturais e comportamentais em subsistemas progressivamente maiores; e o estilo de arquitetura que orienta a organização: os elementos estáticos e dinâmicos, suas interfaces, suas colaborações e sua composição [77].
- Conforme aumenta o tamanho e complexidade dos sistemas, o projeto da solução vai além dos algoritmos e estruturas de dados. Projetar e especificar a estrutura geral do sistema surge como um novo tipo de problema que inclui: organização; estruturas de controles globais; protocolos de comunicação; sincronização e acesso aos dados; delegação de funcionalidades; composição de elementos; escala; performance e escolha entre as alternativas de projetos. Isso é o nível do projeto de arquitetura de um software [78].
- Sistemas de software são construídos para satisfazer os objetivos de negócios das organizações. A arquitetura é a ponte entre esses objetivos e os resultados finais do sistema [61].

Assim, observa-se que a definição de arquitetura é complexa e ultrapassa gerações. A arquitetura de software envolve estruturas que descrevem a organização, dependência entre os módulos, entidades e seus relacionamentos, tempo de execução de componentes, organização para *deployment*, componentes de hardware e outros.

Os requisitos de um sistema podem envolver diferentes requisitos funcionais e requisitos não funcionais.

Os requisitos funcionais podem ser expressos utilizando-se diagramas tradicionais UML [79], mais recentes como histórias de usuários ou uma variação de ambos os métodos [80]. Já os requisitos não funcionais podem também serem representados por UML [79], mas no geral não seguem padrão. Os requisitos não funcionais são também chamados de requisitos de atributo de qualidade e envolvem performance, manutenibilidade, disponibilidade, usabilidade, escalabilidade, segurança e outros. Eles podem definir a arquitetura, *designs patterns* a serem utilizados e decisões nos projetos [61].

Geralmente, eles são divididos em documentos chamados de visões que podem ser usadas para entendimento, comunicação entre as partes interessadas e base para construção e análises[5].

As visões podem ser categorizadas em visões de módulo (foco no código fonte), visões de *deployment* (foco no deploy da aplicação), visões de runtime (foco nos componentes e suas interações), visões de hardware (foco nos componentes de harware em geral) e outros [5].

Métodos ágeis como o Scrum funcionam de forma iterativa e requerem frequentes *feedbacks* dos clientes para assegurar respostas rápidas às mudanças [81]. Em adicional o manifesto ágil [63] sugere que as entregas devem ser frequentes. Desse modo, a busca por arquiteturas de *deployment* que suportem a agilidade requerida pelos métodos ágeis, têm se tornado cada vez mais frequentes [82].

Embora a arquitetura de software, como um todo, tenha bastante influência sobre os métodos ágeis, este capítulo objetiva gerar conhecimento para definição de uma arquitetura de *deployment* direcionada à realidade do ambiente em que será aplicada capaz de prover entregas do software em produção de forma controlada, rápida, frequente e que atenda aos requisitos não funcionais de manutenibilidade, segurança, qualidade, escalabilidade e alta disponibilidade.

3.2.1 Arquitetura de *Deployment*

Uma arquitetura de *deployment* possui foco no *deploy* [1] da aplicação, utiliza-se de componentes e conectores para alocação de software ao hardware (plataforma computacional) onde poderá ser executado. Uma alocação bem realizada possibilita que os requisitos expressos no software sejam satisfeitos pelas características de hardware [5].

Arquitetura de *deployment* se mostra importante, entre outras coisas, para propiciar: o *deploy* de manutenções urgentes sem interromper projetos de manutenção evolutiva em curso; o engajamento para trabalho em equipe; automatização de testes e a realização de entregas com segurança.

Testadores utilizam da visão de arquitetura de *deployment* para entender as dependências em tempo de execução, os integradores a usam para planejar a integração e integrar testes [5].

[1] Processo de entrega do software até o ambiente de produção.

Uma notação informal para arquitetura de *deployment* contém caixas, círculos, linhas, setas que representam o software e elementos comportamentais [5]. Na Figura 3.1 tem-se um exemplo.

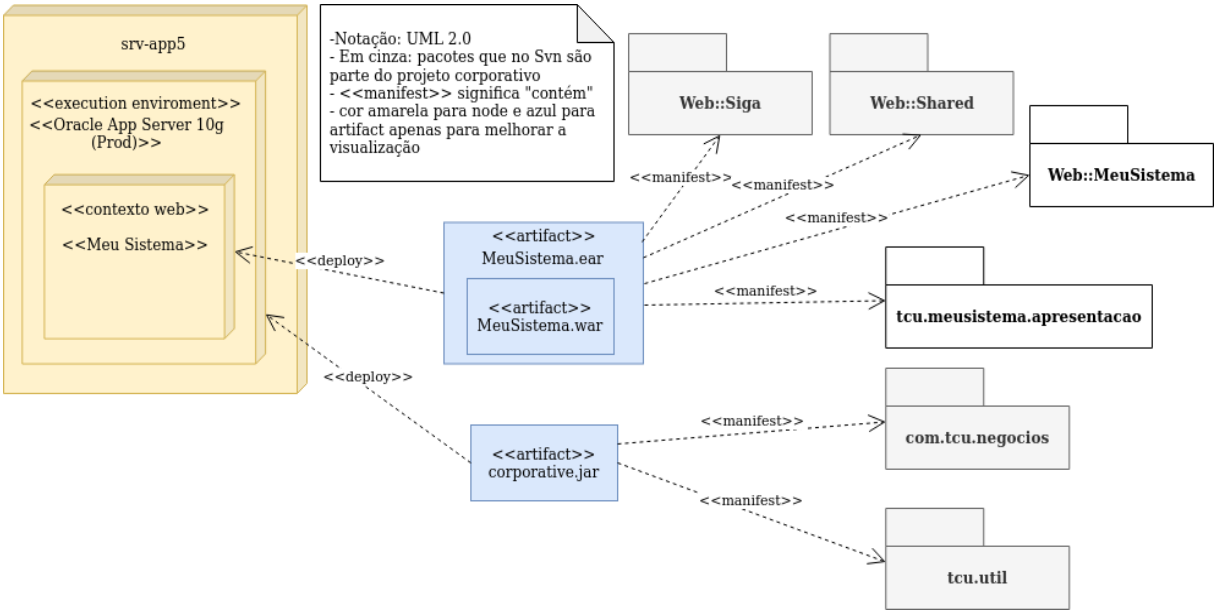


Figura 3.1: Exemplo de visão de *deployment* com notação UML 2.0 (Fonte: [5]).

Na Figura 3.1 é possível verificar onde cada *artifact* será entregue (deploy) ao hardware (servidor). Trata-se de uma aplicação java com controle de versão Svn, sobre um servidor Oracle 10g com seus respectivos contextos. Embora a Figura 3.1 esteja utilizando notação UML 2.0, é muito comum encontrar visões arquiteturais utilizando notação informal, com círculos representando armazenamento, retângulos representando módulos de hardware ou software e setas para fluxo ou conexões, como no caso da Figura 3.2.

A materialização da visão de arquitetura de *deployment* para métodos ágeis pode ser por meio de ferramentas de automatização capazes de criar múltiplos estágios para entrega de software. O nome comum dado a essas ferramentas é: "pipeline" [82]. Um pipeline utiliza de automação das fases de desenvolvimento do software para entregas rápidas, contínuas e seguras (recuperação de código). Além disso, possibilita integração de códigos promovendo o trabalho em equipe, monitoramento e testes.

Inicialmente o desenvolvedor envia seu código (git push) para o ambiente de controle de versões, o pipeline (*CloudBees* com *Jenkins*) da Figura 3.2 é o responsável por fazer o *deploy* da aplicação entre os ambientes de *stage build*, *stage teste* e *stage deploy*. Para o sucesso dessa automatização, a Figura 3.2 mostra outras ferramentas comumente

utilizadas e necessárias para integração contínua e entrega contínua (CI/CD), do inglês *continuous integration/continuous delivery*, são elas: GIT para integração contínua e controle de versões e o orquestrador Openshift (ferramenta proprietária originada do Kubernates), o orquestrador adiciona o código fonte em um container Docker [2], e assim assegura escalabilidade, alta disponibilidade e gerência dos módulos.

Um método estabelecido pelo movimento ágil para se implementar o pipeline é o CI/CD . O movimento *DevOps* [3] adaptou essas práticas e adicionou o pipeline de *deployment* como um dos principais requisitos para automatizar o processo de desenvolvimento de software melhorando a flexibilidade e facilidade de manutenção dos sistemas [82].

Embora os métodos ágeis sejam cada vez mais comuns, muitas organizações não conseguiram alcançar os benefícios de CI/CD devido ao distanciamento entre equipes de desenvolvimento e infraestrutura, fato motivador para que o DevOps se desenvolvesse [85]. A aproximação e estabelecimento de processos de comunicação entre as equipes de desenvolvimento e operação são fundamentais para prover agilidade nos projetos. Isso se dá porque a arquitetura de software requer elementos de hardware e software alinhados às necessidades do negócio.

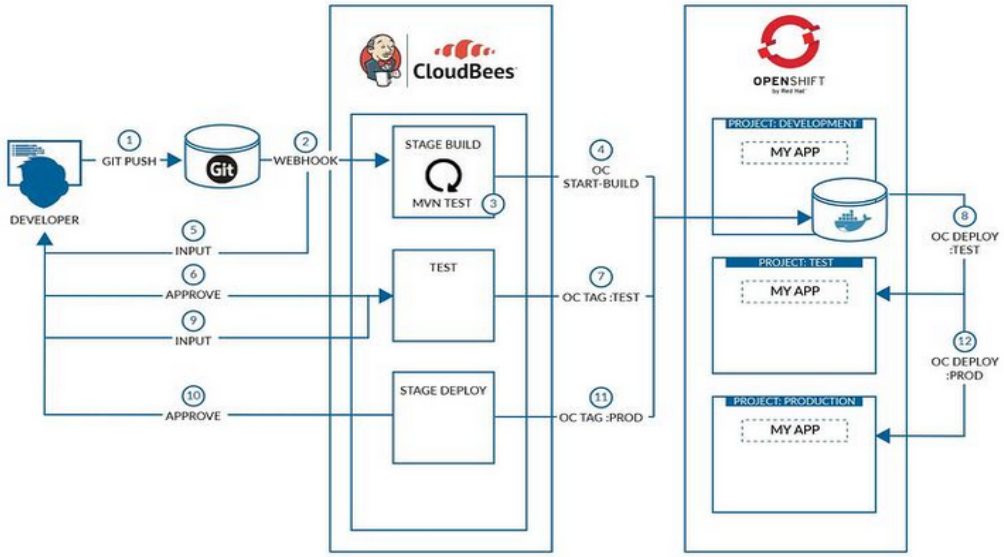


Figura 3.2: Exemplo de Aqruitetura de *deployment* ágil(Fonte: [6]).

[2]Docker é uma ferramenta para criar, executar e gerenciar máquinas virtuais leves [83].

[3]DevOps: uma prática de engenharia de software, que busca unificar o desenvolvimento de software (Dev) e as operações de software(Ops)[84].

A utilização da automatização por meio de CI/CD é considerada essencial para emprego dos métodos ágeis, como bem expõe Vassallo et al. [86]. Tal automatização traz vantagens como descoberta precoce de erros (de integração), controle das entregas, aumento na qualidade, entre outras.

Laukkanen et al. [87] realizaram uma revisão da literatura em que constataram sete principais problemas para emprego de CI/CD, cinco deles estão relacionados às diferentes atividades de desenvolvimento de software: projeto da arquitetura, projeto do sistema, integração, testes e entregas. Os outros dois não estão conectados a nenhuma parte individual e são: fatores humanos/organizacionais e de falta de recursos. A maioria dos problemas se concentram na integração e nos testes.

CI/CD estão relacionados e pode-se dizer que CD depende de CI [82]. Para se ter entregas contínuas (CD), a integração entre os códigos (CI) deve ser praticamente transparente, a arquitetura deve ser suficiente para suportar o desenvolvimento com qualidade, os testes devem ser rápidos, não ambíguos e estáveis, o gerente de projetos não deve demorar nas tomadas de decisões do que deve ou não ser enviado para produção. Além disso, controlar os fatores humanos, como falta de disciplina, falta de motivação e falta de experiência é crucial para manter entregas constantes.

3.2.2 *Continuous Integration* (CI)

No desenvolvimento ágil, existe a necessidade do trabalho colaborativo. Para que isso aconteça, os desenvolvedores precisam trabalhar ao mesmo tempo em recursos diferentes na mesma aplicação. No entanto, o momento de unir esses códigos (conhecido como *merge day*) pode ser trabalhoso e demorado. Isso acontece porque podem ocorrer conflitos entre a codificação de um desenvolvedor e a codificação de outro. Esse problema pode ser agravado se cada desenvolvedor tiver seu próprio ambiente de desenvolvimento local. O ideal seria que a equipe trabalhasse em um mesmo ambiente compartilhado baseado em nuvem [88].

Evitar situações que possam prejudicar o andamento da CI é fundamental. Quando a CI não funciona apropriadamente e falha, causa uma perda de tempo e desvio de foco dos desenvolvedores, diminuindo a produtividade [87].

Uma CI bem sucedida é quando novas mudanças no código de uma aplicação são desenvolvidas, testadas e entregues regularmente em um repositório compartilhado. Construir sistemas com capacidade para suportar integrações frequentes é um pré-requisito para *continuous integration* e suas práticas [89].

Trata-se de uma solução ideal para evitar conflitos quando muitas aplicações são desenvolvidas ao mesmo tempo. Vários testes automatizados, geralmente de unidade e integração, são feitos para garantir que as mudanças não corrompam a aplicação. A CI facilita a correção de qualquer erro encontrado[88].

3.2.3 *Continuous Delivery* (CD)

De acordo com Humble e Farley(2010, apud Steffens et al. [82]), o termo *Continuous Delivery* significa: "um conjunto de práticas que objetivam entregar valor para os clientes rapidamente, de forma confiável e constante com mínimo de sobrecarga".

Trata-se de uma estratégia de *deployment* que envolve um conjunto de ações de desenvolvimento de software que uma organização deve entregar[87]. Visa minimizar o esforço da entrega de novos códigos, novas correções em produção para uso do cliente evitando a perda de tempo e sobrecarga das equipes com processos de *deploy* [88].

O conjunto de ações a serem entregues depende da necessidade da organização e sua adoção pode ser diferente de caso para caso. Embora existam modelos prescritivos, na vida real a adoção do CD requer iterações e ações específicas para cada caso[87]. Normalmente, espera-se que uma arquitetura de CD assegure poder computacional elástico e alta capacidade de monitoramento para suportar as mudanças frequentes nos requisitos, fato que gera implicações financeiras devido a dificuldades em se mensurar a quantidade de recursos necessários [90].

Para se estabelecer uma arquitetura de CD com alto grau de flexibilidade e manutenção, é importante fornecer recursos que reforçam sua autonomia. Uma arquitetura monolítica só é capaz de oferecer autonomia lógica. Para se alcançar maior flexibilidade arquitetural, uma proposta é utilizar um padrão ou estilo de microsserviços [82].

3.2.4 Microsserviços

Os sistemas são compostos de funcionalidades, entretanto, essas funcionalidades independem das estruturas que são escritas, elas não decidem sobre a arquitetura, ou seja, pode-se dividir as funcionalidades em uma série de maneiras e então atribuir sobre diversas arquiteturas [61].

Diate disso, Bass et al. [61] diz: "[...] se a funcionalidade fosse a única coisa que importasse, você não teria que dividir o sistema em partes; uma única bolha monolítica sem estrutura serviria muito bem".

A frase de Bass et al. [61] resume a ideia do que é um monólito de software e chama a atenção para a ideia da quebra desse monólito em partes menores, resultando na definição de microsserviços.

Tradicionalmente, os sistemas são desenvolvidos de forma monolítica, como um único bloco composto de rotinas e funções ou métodos e classes, mas ao compilar ou executar, tudo se conecta formando um mesmo bloco monolítico de código.

As arquiteturas monolíticas geralmente usam uma abordagem de implantação "big bang" que atualiza toda a aplicação de uma vez e algumas vezes incluem atualizações de banco de dados. Essa abordagem pode ser lenta e sujeita a erros devido às suas ramificações pesadas e, desse modo, menos ágeis [84].

Por outro lado, com o advento das metodologias ágeis e da cultura do DevOps, uma nova forma de arquitetura vem sendo implementada, que é a arquitetura de microsserviços.

A ideia de se ter um baixo acoplamento, é de conseguir evoluir o sistema se apoiando na substituição de seus componentes sem impactar significativamente o sistema por completo [91]. Microsserviços vieram para estender esse conceito até o nível de *deploy* em que pequenos módulos coesos de software são entregues e se comunicam, normalmente, utilizando-se protocolo HTTP.

Não há um consenso quanto à definição de microsserviços, o que se percebe é que diversos autores tentam adicionar características que se espera que um microsserviço tenha, como por exemplo [92]:

- Possuir poucas linhas de código;
- Evitar chamadas circulares entre serviços;
- Ser desenvolvido por uma equipe pequena;
- Evitar chamadas síncronas (chamadas de solicitação-resposta);
- Oferecer operações de baixa granularidade;
- Ser executado pela equipe que os construiu;
- Não participar de composições de serviços complexas;
- Ter liberdade de usar novas linguagens, *frameworks* e plataformas;
- Gerenciar seus próprios bancos de dados;
- Ser monitorado por uma sofisticada infraestrutura de registro e monitoramento;

- Ser desenvolvido por equipes divididas em torno de capacidades de negócios motivadas pela lei de Conway (trata-se de um ditado que afirma que as organizações projetam sistemas que refletem sua própria estrutura de comunicação [93]).
- Estar alinhado ao contexto de Domain Driven Design DDD[94];
- Ser encontrado rapidamente;
- Ser pequeno o suficiente para ser re-escrito em duas semanas.

Portanto, os requisitos para um microsserviço estão em quase tudo e costumam mencionar boas práticas de engenharia de software conhecidas há décadas [92]. Conforme Merson [92], uma definição bastante difundida e aceita é a de Lewis e Fowler[95]:

> Resumindo, o estilo de arquitetura de microsserviço é uma abordagem para desenvolver um único aplicativo como um conjunto de pequenos serviços, cada um executando em seu próprio processo se comunicando com mecanismos leves, muitas vezes uma API de recurso HTTP. Esses serviços são desenvolvidos em torno de capacidades de negócios e podem ser implantados de forma independente por mecanismos de implantação completamente automatizados.

Uma forma que facilita a replicação e implantação dos microsserviços é por meio da utilização de *containers* de software. O padrão atual, amplamente difundido para *containers* é o Docker, um *container* leve que possui um kernel Linux e interage com ele praticamente sem *overhead* [83].

Utilizando-se Docker torna-se possível, ao programador, se concentrar na produção de código, pois, após configurado, o Docker pode prover toda infraestrutura necessária de um servidor de aplicação, podendo ser usado para manter o alinhamento tecnológico entre os desenvolvedores sem lhes tirar a autonomia de criação.

Após a codificação, o desenvolvedor pode embutir o código no Docker e distribuir em um *cluster* aproveitando de seu tamanho pequeno e leveza. Isso permite alcançar repetibilidade e uniformidade perfeitas nos ambientes [83].

Para que todos os *containers* comuniquem entre si e também com outros servidores é necessário uma estrutura capaz de realizar a orquestração entre eles. Essa estrutura é chamada de orquestrador, sendo o mais comum na comunidade de software livre o Kubernates [96].

Além de prover a orquestração dos componentes, o Kubernates possibilitam uma série de outras funcionalidades como, por exemplo: escalabilidade horizontal, alta disponibilidade, rápidas e transparentes atualizações do software, computação distribuída, entre outras [96].

Assim como o Docker, o Kuberntates é fudamental para construir sistemas com capacidade para suportar CI/CD. Entretanto, deve-se buscar equilíbrio na quantidade de microsserviços, pois a modularização em excesso pode ser um fator de aumento da complexidade para *Continuous Delivery* [87].

3.3 Testes de Software e as Metodologias Ágeis

Conforme apontam as pesquisas, os métodos ágeis requerem atributos de testabilidade para alcançar qualidade nas entregas e, assim, diminuir o retrabalho e manutenções corretivas.

Para Ambler (2005 apud Piovesan, 2018) [18] considera-se que a qualidade ágil desejada é resultado de várias práticas, sendo elas: o desenvolvimento iterativo e incremental, técnicas de teste ágeis, automação de testes unitários e de regressão, métricas de codificação e o trabalho colaborativo.

Os testes de software são considerados elementos críticos para garantia da qualidade, são práticas caras tanto para implantar e manter mas, geralmente, não tão caras quanto os custos gerados por sua não aplicação como: constantes correções de erros, entregas de softwares de má qualidade.

Em sistemas críticos, os testes são relevantes, dado que as perdas podem ser enormes, sejam elas materiais ou humanas. Diante disso, as tecnologias de testes vêm evoluindo nas últimas décadas para diminuir os riscos de perdas.

Os algoritmos das suítes de testes estão cada vez mais avançados e se utilizam de redes neurais, algoritmos genéticos, estratégias de mutantes, etc. Entretanto, as exigências por softwares cada vez mais complexos colaboram para que a falha sempre esteja presente. Um exemplo recente foi o acidente com um veículo autônomo resultante de uma falha do software em detectar um caminhão branco contra o céu brilhante [97].

Desse modo, considerar a criticidade do software ao realizar os testes pode fazer a diferença entre o sucesso e o fracasso. Mesmo em cenários que aparentemente não sofreriam tantos danos, ao se fazer uma análise mais minuciosa é possível encontrar riscos que seriam catastróficos. Por exemplo: no campo de estudo desta pesquisa (Tribunal de Justiça) os prejuízos por uma falha processual podem ter consequências humanas e materiais imensuráveis.

3.3.1 Documentação de Requisitos e as Metodologias Ágeis

No passado, os métodos ágeis e os requisitos tradicionais eram conflitantes, particularmente pelo hábito de se tratar a Engenharia de Requisitos (ER) de forma restrita, como um conjunto de declarações do que o sistema "deveria" fazer com base em documentações que os ágeis preferem evitar. No entanto, a ER cobre um amplo campo de requisitos, implícitos ou explícitos, que podem se adequar ao estilo ágil de comportamento e pesamentos [98]

A definição e gerenciamento de requisitos são considerados como fatores críticos na Engenharia de Software [99]. Uma especificação de requisitos inadequada funciona como catalisador para outros problemas, como baixa produtividade da equipe e dificuldade em manter o software [100].

Para se adequar à dinâmica dos métodos ágeis, os analistas de requisitos propuseram uma documentação mais simples que fosse clara, objetiva e adaptável, dando origem ao conceito de *living documentation*, ou documentação viva [14].

Diz-se que a documentação está viva porque ela deve ser dinâmica durante todo o ciclo do projeto, deve haver uma comunicação clara entre analistas e clientes (*linguagem ubíqua*) com propósito mais voltado ao comportamento e menos aos procedimentos.

A pesquisa de Schön (2017), citado por Fraga e Barbosa[101], identificou que os principais documentos de requisitos utilizados nos métodos ágeis são: histórias de usuário, protótipos, casos de uso e cartões de histórias. Alguns autores (Haugset e Stlhane, Mead, Schön), conforme Fraga e Barbosa [14], indicaram também que os requisitos podem estar documentados nos casos de testes.

A Figura 3.3 elenca as principais diferenças entre o levantamento de requisitos tradicional e ágil.

As pesquisas indicam uma documentação de ER que se destaca quando se utiliza métodos ágeis, são as *histórias de usuários* [102, 100, 99]. Elas são usadas para representar a necessidade do cliente, devem ser simples, objetivas e expor os benefícios a serem entregues. Os requisitos são discutidos de forma detalhada durante toda a implementação. Em alguns casos, as histórias de usuários não são necessárias, sendo substituídas pela comunicação face a face [101].

Embora o uso de requisitos ágeis gere ganho inicial de tempo, a ausência de detalhes na documentação pode resultar em não alinhamento com as expectativas do cliente [14]. Na pesquisa de Medeiros et al. [100] cerca de 50% das tarefas de manutenção analisadas tiveram problemas devido à insuficiência de documentação de requisitos. Já os estudos de Mendes et al. [103] mostraram que o débito de documentação gera um aumento no

Atividades ER	Abordagem Tradicional	Abordagem Ágil	Práticas Ágeis que apoiam a ER
Elicitação de Requisitos	Definir todos os requisitos no início do projeto	Definição dos requisitos distribuída por todo o processo de desenvolvimento	- ER iterativa - Comunicação cara-a-cara
Análise de requisitos e negociação	Foco na resolução de conflitos	Foco no refinamento, mudança e priorização de requisitos iterativamente	- ER iterativa - Comunicação cara-a-cara - Planejamento constante - Priorização extrema
Documentação dos Requisitos	Documentação formal, contendo requisitos detalhados	Sem documentação formal e detalhada. Baseado, basicamente, em histórias de usuário	- Comunicação cara-a-cara
Validação dos Requisitos	Análise da consistência e completude do requisito	Foco em analisar se o requisito contempla a necessidade do cliente	- Reuniões de Revisão - Comunicação cara-a-cara

Figura 3.3: Comparação entre abordagens tradicional e ágil em relação ao processo de Engenharia de Requisitos (Fonte: [7]).

esforço 47% maior que o estimado no projeto e um custo extra que chega aos 48%. Para Borrego et al. [104], um dos desafios dos métodos ágeis é manter o conhecimento ao longo do tempo devido à insuficiência de documentação.

Embora o Manifesto Ágil [63] preze mais pelo software em funcionamento do que uma documentação abrangente, ele não se opõe a qualquer prática adicional que vise melhorar a qualidade.

Com o crescimento dos métodos ágeis nos últimos anos, as pesquisas apontam diversas limitações em relação às atividades de Engenharia de Requisitos. Desse modo, buscar o equilíbrio entre documentação e agilidade pode ser a melhor estratégia.

3.3.2 Teste no Time Ágil

É comum empresas delegarem a responsabilidade dos testes para outros times especialistas. Ao fazerem isso o processo de desenvolvimento ágil é prejudicado, pois normalmente os times especialistas estão acostumados com o modelo tradicional de testes. Essa situação gera conflitos entre as equipes que acabam culpando umas às outras pela demora [105].

Os métodos ágeis possuem poucos papéis definidos e geralmente o time é responsável desde a concepção do produto até a entrega, não existindo o papel do testador[106]. Devido à natureza do desenvolvimento ágil, os testes devem ser executados de forma contínua e integrada durante todo o projeto.

Salerno et al. [106] discutiram o impacto de se ter um profissional do time como facilitador na identificação dos problemas e para auxiliar no melhor entendimento das necessidades e engajamento do usuário, a este profissional deu-se o nome de *Product Designer* que se mostrou crucial na identificação de melhores soluções aos problemas dos usuários.

Profissionais de qualidade, também chamados de QA (*Quality Assurance*) ou *testers*, passam a exercer papel dentro do time e geralmente não desperdiçam tempo escrevendo documentação para depois fazer testes manuais. Normalmente, eles levantam os requisitos junto ao profissional de negócio de modo que a especificação se torna o próprio teste na medida que os desenvolvedores vão entregando pequenos incrementos facilmente testáveis [105].

Muitos métodos de testes surgiram e alguns ganharam forças de modo a sustentar os processos baseados nos métodos ágeis. Entre eles, vale ressaltar os conceitos de *Test-Driven Development* (TDD), *Acceptance Test Driven Development* (ATDD), *Behavior-Driven Development* (BDD) [107]. Para mais informações sobre essas técnicas veja o APÊNDICE K.

3.4 Considerações Finais do Capítulo

Neste capítulo, foram apresentados conceitos como Arquitetura de *Deployment* e testes de software com métodos ágeis. São conceitos relevantes para este trabalho pois proporcionam o trabalho colaborativo, com envolvimento das partes interessadas, de forma segura, rápida e com qualidade. No Capítulo 4 apresentam-se os materiais e métodos adotados nesta pesquisa de acordo com a metodologia predefinida.

Capítulo 4

Materiais e Métodos

4.1 Considerações Iniciais do Capítulo

Em face do objetivo desta pesquisa, apresenta-se neste capítulo os materiais e métodos do plano metodológico adotado, com o detalhamento das 4 fases em cada seção (Planejamento da Pesquisa; Coleta de Dados; Análise e Interpretação dos Dados; e Redação dos Resultados).

4.2 Planejamento da Pesquisa

Nessa fase, *Planejamento da Pesquisa* foram definidos o tema da pesquisa, a questão da pesquisa, os objetivos e a classificação metodológica, conforme descritos no Capítulo 1.

Dado o contexto e o objetivo desta pesquisa, *Definir e avaliar a implantação de um processo de manutenção de software para o TJGO, empregando metodologias ágeis*, as demais fases da pesquisa foram iniciadas.

4.3 Coleta de Dados

Nessa fase Coleta de Dados foram adotadas as pesquisas bibliográfica e documental, e a técnica *Design Science Research* (DSR), conforme Figura 4.1,

A *Design Science Research* (DSR) possibilita responder a questão da pesquisa utilizando soluções práticas para resolver os problemas observados no contexto social. Trata-se de uma método reconhecido na comunidade científica, que possibilita evidenciar os resultados obtidos e reduzir a imparcialidade, favorecendo assim, a coleta e análise dos

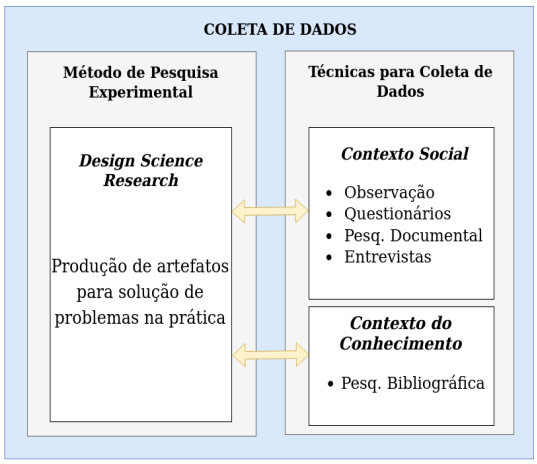


Figura 4.1: Fase de Coleta de Dados (Fonte: autoria própria).

resultados.

A DSR se utiliza de fontes que ajudam a compreender o problema no contexto social e de fontes no contexto do conhecimento que ajudam a solucionar o problema:

- O Contexto Social se refere ao contexto do Tribunal de Justiça de Goiás (TJGO), isso é, o ambiente de experimentação onde o problema foi observado;

- O Contexto do Conhecimento é construído com base na Pesquisa Bibliográfica.

A DSR proposta por Wieringa [2] foi a adotada neste trabalho. Entretanto, com uma adaptação no ciclo DS original. Um fluxo de retorno entre as fases de *Validação* e *Design da Solução* foi criado, dado que, antes da *Implementação* podem ser necessárias algumas validações para que se tenha um *Design da Solução* satisfatório. A DSR adaptada é apresentada na Figura 4.2.

4.3.1 Instrumentos de Coleta de Dados adotados

Para compreender o problema no contexto do TJGO, conforme planejamento metodológico, foram adotadas técnicas como: *Observação, Questionários, Pesquisa Documental* e ao final da pesquisa, utilizou-se da técnica de *Entrevista* para avaliação dos resultados no contexto social conforme descrito na Seção 4.4.1.

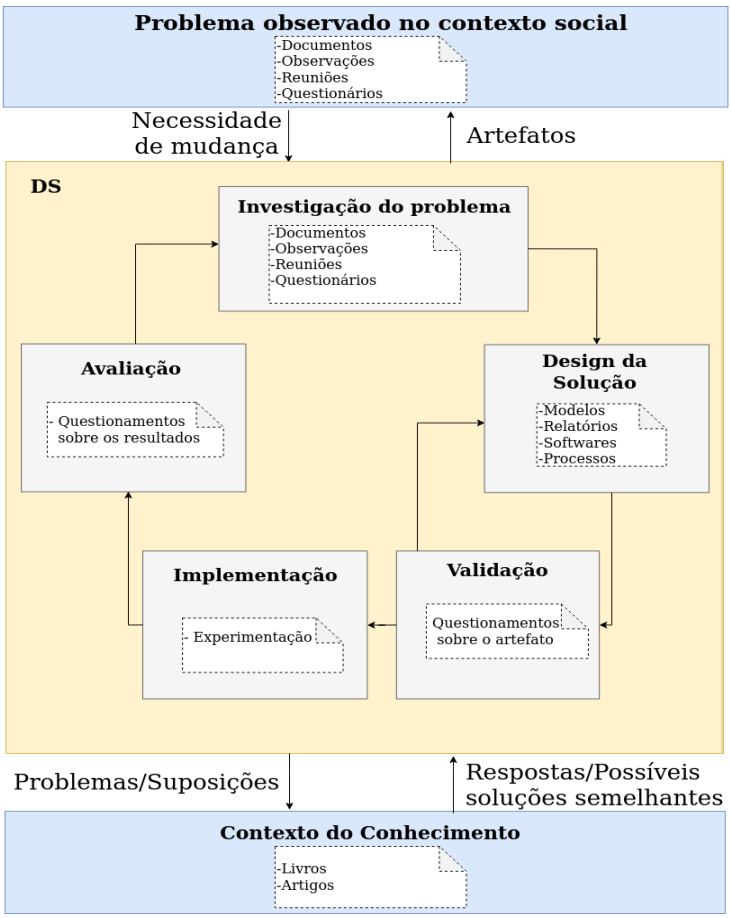


Figura 4.2: Framework DSR de Wieringa (Fonte: adaptado [2]).

As amostras e experimentações ocorreram no ambiente de trabalho do TJGO, respeitando os critérios éticos que permeiam a administração pública. Para entendimento do problema utilizou-se de:

- **Observação**: Neste trabalho, a técnica de observação participativa foi empregada no contexto do TJGO por meio dos sentidos e experimentação, gravações de áudio e reuniões. Por ser servidor de carreira há mais de 13 anos e Coordenador da Equipe de Softwares Administrativos há 3 anos, o pesquisador pôde indagar sobre as características do objeto, analisar pontos fortes e pontos fracos.
- **Questionários**: Neste trabalho, os questionários foram aplicados com perguntas abertas e de múltipla escolha, por amostra no universo do time de desenvolvedores, diretoria de informática e clientes conforme APÊNDICES C e D. A utilização de questionários complementou as demais técnicas possibilitando diminuir o viés inerente das limitações humanas, provenientes das observações feitas a partir dos

sentidos. O propósito foi obter dados para direcionamento da pesquisa, obter informações para avaliar se as alterações estavam gerando os efeitos esperados ou não.

- **Pesquisa Documental:** Neste trabalho, a pesquisa documental, permitiu revisar documentos oficiais, procedimentos internos e externos direcionados ao contexto social da pesquisa. Foram realizadas pesquisas referente ao TJGO, como: lei 20.756/2020 dos servidores estatutários do Estado de Goiás [108], análise de documentação de software, Plano Estratégico de TI [36] e instruções normativas do CNJ [37]. Tais documentações, ajudaram no entendimento do contexto social do problema.

Como fonte de soluções foram pesquisados livros e artigos dispostos na Pesquisa Bibliográfica, Capítulo 2, possibilitando a geração de conhecimento sobre as manutenções de software nos últimos vinte anos e o mapeamento de autores que contribuem para a manutenção de software.

Diante do material coletado traçou-se um histórico de alguns métodos ágeis aplicados à manutenção de software com ênfase maior em temas recentes (últimos cinco anos) e naqueles que realizaram aplicações práticas. Os procedimentos são detalhados no APÊNDICE A.

4.3.2 Ciclos de *Design Science*

A DSR permitiu explorar experimentalmente o contexto do conhecimento e o contexto social utilizando-se um ciclo regulador (ciclo DS) capaz de gerar artefatos artificiais para serem testados no mundo natural. A fim de se atingir os objetivos desta pesquisa, utilizou-se de equipe própria de desenvolvimento.

Neste trabalho, o ciclo de *Design Science* (DS) está dividido em cinco fases: *Investigação do problema; Design da Solução; Validação; Implementação;* e *Avaliação.*

Os ciclos ocorreram entre o contexto social do problema e contexto do conhecimento. Foram planejados e realizados os ciclos DS:

- Ciclo de Definição da Arquitetura;
- Ciclo de Experimentação do Kanban;
- Ciclo de Experimentação do Scrum de Rehman et al. [3]; e
- Ciclo de Experimentação de metodologia Híbrida sobre o processo de Rehman et al [3].

Iniciou-se pela *arquitetura*, seguida pela *experimentação do emprego do Kanban*. Com a experiência e limitações observadas, partiu-se para *experimentação do SCRUM* utilizando-se o processo de Rehman et al. [3] e por fim, a aplicação do processo de Rehman et al. [3] com *metodologia híbrida*. Apresentam-se os detalhes no Capítulo 6.

4.3.3 Critérios de Aceitação para os Ciclos de *Design Science*

Para o Ciclo 01 - *Definição da Arquitetura*, cuja a arquitetura foi a de *deployment*, os critérios de aceitação para a definição da arquitetura foram:

1. Uma arquitetura que possibilitasse o *deploy* contínuo;
2. Promovesse a integração e recuperação de versões;
3. Possibilitasse trabalhar com manutenções urgentes e outras manutenções; e
4. Suportasse a realização dos testes.

Esse ciclo foi essencial para a definição de um processo.

Já para os Ciclos de *definição de processo*, os critérios de aceitação foram:

1. Utilizasse os métodos ágeis, tratasse as vertentes da manutenção separadamente para manutenções urgentes e outras manutenções;
2. Possibilitasse o controle das versões, contribuísse com a organização estratégica sem confrontar com os princípios da administração pública e seus poderes (em especial o poder hierárquico);
3. Promovesse o trabalho em equipe e gerasse qualidade nas entregas.

Neste trabalho, os 04 ciclos estão representados na Figura 4.3. Cada fase da DS possui entradas relativas a um problema inicial ou resultante da interação anterior, formando assim, uma conexão, um alinhamento entre os ciclos. Os refinamentos e alinhamentos foram guiados pelos critérios de satisfação que conduziram para o término da pesquisa. Cada ciclo possibilitou a melhoria contínua por meio da construção dos artefatos e análise dos dados gerados por esses artefatos avaliados no contexto social. A análise dos dados resultantes de cada ciclo conduziu a novos ciclos, e esses, até o nível de satisfação desejado.

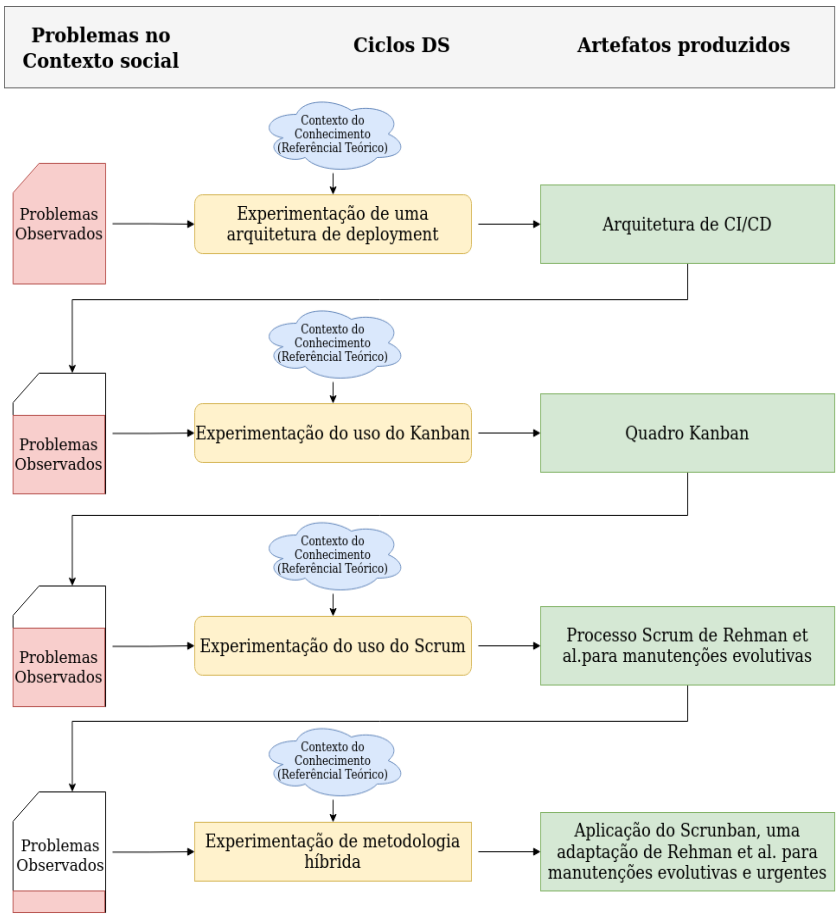


Figura 4.3: Ciclos DSR (Fonte: autoria própria).

4.4 Análise e Interpretação dos Dados

Ao longo da pesquisa foram empregadas as técnicas de *Observações*, *Questionários* e *Pesquisa Documental* que direcionaram os ciclos da DSR nas escolhas de métodos e ferramentas.

Nessa fase, Análise e Interpretação dos Dados, o objetivo foi possibilitar ao pesquisador encontrar novas dimensões e gerar maior entendimento sobre os impactos da implantação do processo de manutenção de software no TJGO.

Foram empregadas a técnica *Entrevistas* e o *Método de Análise de Conteúdo*.

4.4.1 Entrevistas

As entrevistas *semiestruturadas* foram realizadas por videoconferência utilizando-se ferramenta computacional *Zoom Video Communications* [109], possibilitando a realização

remota e a gravação para posterior transcrição, conforme APÊNDICES (G, H e I).

Antes das entrevistas, um termo de compromisso e consentimento foi lido conforme o APÊNDICE F. No termo são descritas cláusulas de ética e preservação do anonimato para resguardar as partes. O objetivo de se aplicar as entrevistas foi coletar as percepções dos envolvidos quanto ao processo de manutenção de software resultante. As entrevistas foram realizadas após a realização do 4º ciclo da DSR. No Capítulo 7 apresentam-se os 04 ciclos.

A escolha por *entrevistas* se deu pela oportunidade de entrevistar os membros da equipe, diretores e usuários gestores e possibilitar análises de modo a superar as incertezas, enriquecer a leitura, e obter comprovações sobre o contexto do Tribunal e sobre os artefatos gerados para diminuição dos problemas observados.

Os seguintes critérios foram utilizados para escolha dos participantes:

- Equipe de TI: os quatro servidores, da equipe Administrativa, que mais participaram, das manutenções no último ano;
- Os diretores de TI diretamente envolvidos: Diretor de Engenharia de Software e diretor de TI;
- Usuários gestores: Os participantes dos dois últimos projetos de manutenção evolutiva realizados.

4.4.2 Método de Análise de Conteúdo adotado

Para análise dos dados resultantes das entrevistas utilizou-se o método de *Análise de Conteúdo* [45], cujo objetivo é respaldar os estudos realizados com foco na questão de pesquisa. Para a visualização dos dados, foi adotada a ferramenta computacional Atlas.ti [46], sugerida por Gibbis [47].

A análise utilizou *abordagem qualitativa*, dada a natureza aplicada da pesquisa, considerando seu objetivo, com contexto de aplicabilidade local que reflete opiniões de uma pequena população amostral. Assim, adotou-se o método mais assertivo, apresentando resultados pautados sob uma fundamentação metodológica confiável.

Todavia, considerou-se analisar quantitativamente as demandas urgentes, mas pode-se verificar que variáveis aleatórias prejudicavam as análises mais conclusivas diante do pouco tempo de uso do processo. Por exemplo, ao se verificar a quantidade de demandas urgentes, percebeu-se um aumento, que pode ser resultado do estabelecimento de um canal único de solicitações ou da mudança de diretoria. Portanto, para uma análise quantitativa de valor científico, será necessário mais tempo.

4.5 Redação dos Resultados

A fase de Redação do Resultados compreende esta monografia do trabalho de pesquisa realizado.

4.6 Considerações Finais do Capítulo

Neste Capítulo foram apresentados o planejamento metodológico e as técnicas e instrumentos adotados. O emprego da DSR possibilitou testar um método e ao avaliar o contexto, nortear a busca de um novo método mais adequado a realidade do órgão. No Capítulo 5 apresenta-se um detalhamento do objeto de estudo, o TJGO, e em seguida, no Capítulo 6 descreve-se a execução da técnica DSR, seguido do Capítulo 7 em que se descreve a análise e interpretação dos dados.

Capítulo 5

Contexto da TI no TJGO

5.1 Considerações Iniciais do Capítulo

Neste capítulo, apresenta-se uma descrição da área de tecnologia de informação (TI), em especial da área de desenvolvimento de sistemas do TJGO. Apresenta-se o organograma da área de TI, em que se dá maior atenção às divisões que desenvolvem ou fornecem suporte para construção de sistemas. Apresenta-se um resumo da organização das equipes e como é percebido pelo autor o conceito de proatividade na Divisão de Engenharia de Software (DES). Em seguida, apresenta-se a divisão que está intimamente ligada a esta pesquisa, o *Núcleo Técnico de Sistemas Administrativos*. São apresentadas as relações interpessoais, a forma de trabalho e as tecnologias utilizadas.

5.2 Visão Geral da Estrutura Organizacional da TI no TJGO

O Tribunal de Justiça do Estado de Goiás (TJGO) é um órgão do poder judiciário da administração pública brasileira, com competência jurídica em todo o Estado de Goiás.

Presidido por um desembargador eleito a cada dois anos, o TJGO desenvolve funções típicas como atividades principais e atípicas como atividades acessórias, sendo as funções típica, aquelas referentes ao exercício jurisdicional do Estado e as atípicas, aquelas caracterizadas por atividades administrativas e legislativas.

É um órgão composto por magistrados e servidores, sendo que seus servidores são regidos pela Lei nº 20.756, de 28 de janeiro de 2020 [108].

Um resumo da estrutura organizacional do TJGO até 31 de dezembro de 2020 pode ser visto na Figura 5.1.

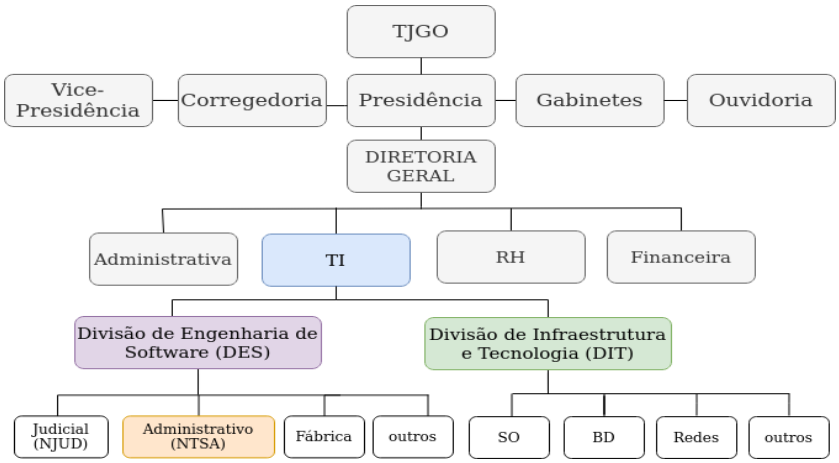


Figura 5.1: Organograma do TJGO até 31 de dezembro de 2020 (Fonte: adaptado [8]).

Após a posse do novo presidente, em fevereiro de 2021, o organograma sofreu alterações, sendo que a mais importante para este trabalho, foi a mudança da *Diretoria de TI* (DTI) para próximo da presidência Figura 5.2.

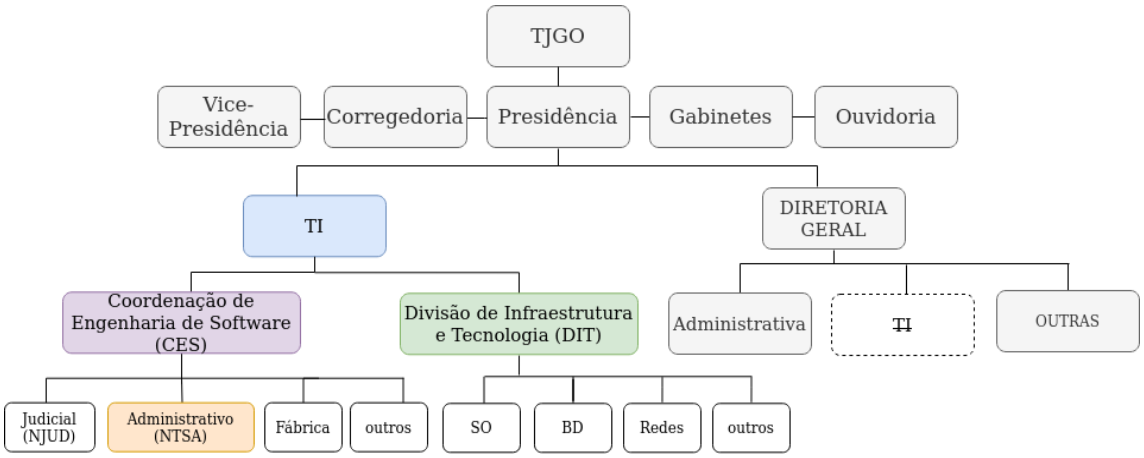


Figura 5.2: Organograma do TJGO após 31 de dezembro de 2020 (Fonte: adaptado [8]).

A mudança da Diretoria de TI para próximo da Presidência demonstra uma mudança de visão estratégica quanto aos anseios de utilização da TI para uma melhoria da prestação jurisdicional. O TJGO possui uma TI centralizada, pautada no Decreto Judiciário 2162/2018 [110], conforme apresentado na Figura 5.3.

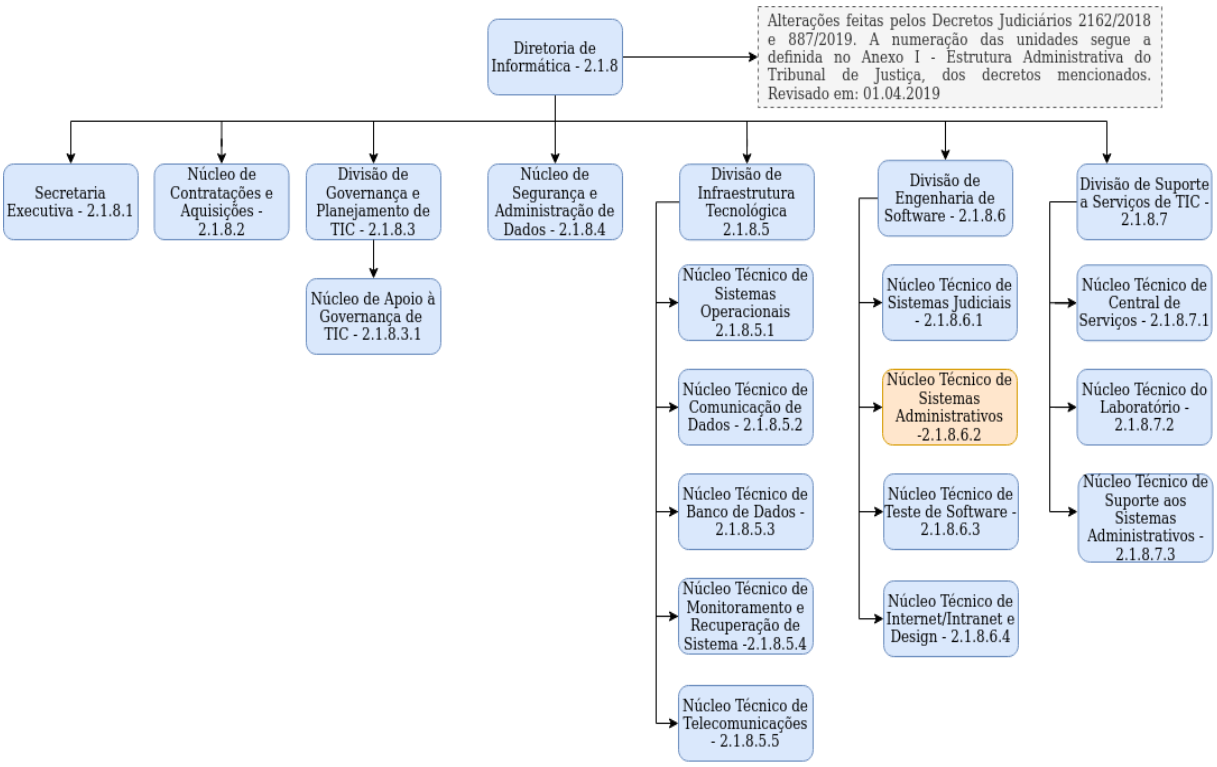


Figura 5.3: Organograma de TI do TJGO (Fonte:[8]).

As divisões que estão diretamente relacionadas a esta pesquisa são: Divisão de Engenharia de Software (DES) e Divisão de Infraestrutura Tecnológica (DIT) com subdivisões de Sistema Operacionais (SO), Banco de Dado (BD), Redes e outros.

Cada equipe da Divisão da DES possui integrantes, na maioria, servidores de carreira, sob uma Coordenação e todos estão subordinados à Diretoria de Engenharia de Software.

A DIT também é composta por servidores de carreira mas possui significativa presença de trabalhadores terceirizados, os quais colaboram com as atividades pertinentes da área. Cada Núcleo também possui um *Coordenador* e todos estão sob a gestão do *Diretor de Infraestrutura e Tecnologia.*

Ressalta-se que a comunicação entre a DES e a DIT é uma via frequente e necessária entre as duas divisões, todavia, trata-se de uma ferramenta que às vezes é prejudicada pela falta de processo ou mesmo alinhamento entre as equipes, pois geralmente essa comunicação ocorre de forma informal e direta entre os desenvolvedores e os técnicos de infraestrutura. Percebe-se esta falha de comunicação, também, entre os núcleos da DES gerando desalinhamento entre as equipes.

O TJGO é um órgão que teve poucos concursos para esfera administrativa, foram 4 concursos, sendo 3 nos últimos 15 anos. O autor dessa pesquisa tomou posse em 2008, no

segundo concurso, o de maior número de nomeados, cerca de 40 pessoas foram nomeadas para trabalhar na TI, divididas nos cargos de Técnicos Judiciários (Graduados na área de Informática), posteriormente alterado para Analista Judiciários pela lei 17.663/2012 [111] e Auxiliar Judiciário (Ensino Médio), atualmente extinto pela mesma lei. Era notável a motivação e empolgação no início e também perceptível a diminuição do ritmo ao longo dos anos, fenômeno observado também para os concursados subsequentes.

A demanda por produtos de software no TJGO acompanha o ritmo acelerado da sociedade que cada vez mais depende de soluções automatizadas para realizar suas atividades. Como resposta a essas necessidades, foi contratada uma *fábrica de software* que atualmente é gerenciada pela equipe de testes Núcleo para Testes de Software (NTTS)).

A seguir, detalhada-se as divisões da DES e NTSA, por serem os objetos principais desta pesquisa.

5.3 Divisão de Engenharia de Software (DES)

A quantidade de sistemas desenvolvidos e mantidos pela DES é alta, gerando uma demanda constante por manutenções evolutivas e corretivas que são, no geral, delegadas para um desenvolvedor, geralmente a mesma pessoa que desenvolve e mantém o sistema.

A DES se divide nas equipes Núcleo Técnico de Sistemas Judiciais (NTSJ), Núcleo Técnico de Testes de Software (NTTS), Núcleo Técnico de Internet/Intranet e Design (NTIID) e Núcleo Técnico de Sistemas Administrativos (NTSA).

As equipes de sistemas possuem a liberdade para se organizarem da forma que preferirem. No geral são profissionais multidisciplinares que realizam seus trabalhos individualmente com pouca colaboração ou participação de outros, realizando as fases de desenvolvimento da forma que acharem mais producente sem seguir metodologia ou controle dos possíveis riscos.

Quanto às lideranças ao longo dos últimos dez anos foram majoritariamente liberais. Este tipo de liderança pressupõe que os colaboradores possuem um nível de excelência a ponto de não precisarem da atuação tão intensa do líder que passa a focar em fatores de motivação, organização e delegação das tarefas. Possui como vantagens flexibilidade nas tomadas de decisões, divisão de responsabilidade entre líder e liderados e desvantagens como diminuição da produtividade por falta de orientação, falta de controle e qualidade das atividades, individualismo [112].

Segundo Sommerville [15], a necessidade de gerenciamento de sistemas é uma importante distinção entre o desenvolvimento profissional de software e a programação em nível amador.

A nova forma de Governo iniciada após a Constituição Federal de 1988 se baseia em princípios explícitos de Legalidade, Moralidade, Produtividade e Eficiência. Esses princípios têm proporcionado novas formas de gestão pública. Segundo Aquino [113], o princípio da eficiência deveria ser o norteador da administração pública, eliminando processos burocráticos e introduzindo modelos gerenciais com foco nos resultados.

Entretanto, eliminar processos burocráticos não significa eliminar processos. Diante da transformação tecnológica vivida pela sociedade, mais do que nunca, há uma crescente necessidade de investimentos em gerenciamento de projetos de software porque a engenharia de software profissional está sempre sujeita a restrições de orçamento e prazo.

De forma errônea é comum que os analistas da DES tomem decisões sem o conhecimento da diretoria, muitas vezes com justificativas de estarem diminuindo a burocracia em prol de um processo mais ágil. O fluxo de desenvolvimento se assemelha ao exposto na Figura 5.4.

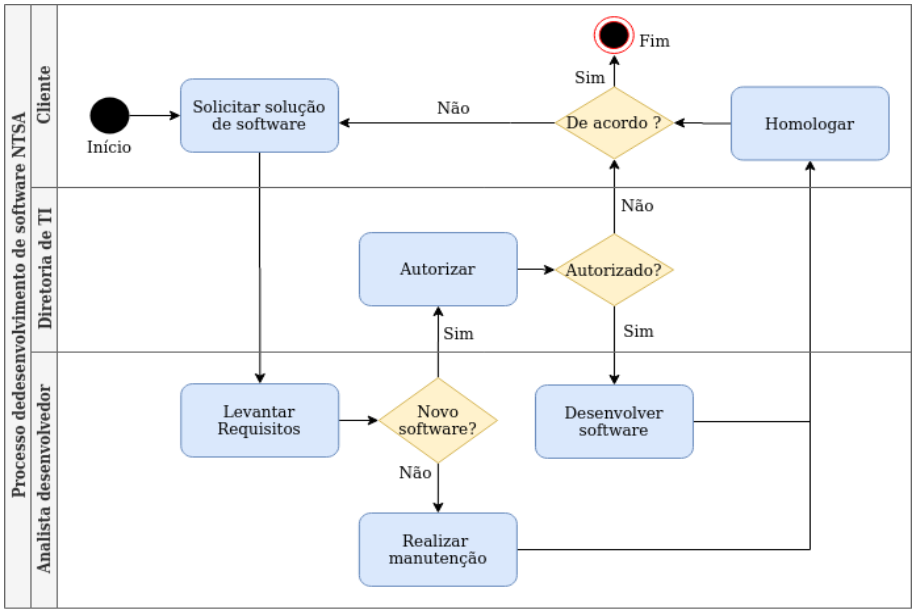


Figura 5.4: Fluxo de Desenvolvimento no NTSA. (Fonte: autoria própria)

O meio de comunicação mais comum com o cliente é o telefone. Geralmente o cliente entra em contato com o Gestor de TI ou diretamente com o desenvolvedor para informar sobre a necessidade de manutenção ou solução automatizada para um dado *problema*.

Após esse contato inicial, o desenvolvedor coleta os requisitos por telefone ou agenda uma reunião para analisar melhor o *problema*.

De posse dos requisitos, que não são documentados, o desenvolvedor realiza a atividade e disponibiliza em produção para que seja validado e testado e, a partir da fase de uso, são realizadas possíveis manutenções corretivas e evolutivas sem prazo estabelecido de término.

O modelo básico que move a DES, é um modelo baseado no conceito da proatividade e embora a equipe tenha um bom relacionamento interpessoal, seja bastante capacitada e habilitada para exercer as atividades, alguns já estão bem acostumados com suas rotinas e as propostas de mudança geralmente não são bem aceitas. Até mesmo para aqueles que de fato se mantêm proativos, a mudança não é bem vinda pois causa uma sensação de diminuição da importância individual.

Para Covey [114], *proatividade* significa muito mais do que apenas tomar a iniciativa, trata-se da capacidade humana natural de subordinar os sentimentos aos valores.

Pessoas proativas são responsáveis por fazer acontecer. Seu comportamento é produto de sua escolha consciente, baseada em valores, e não em resultados de um condicionamento. Caso esse condicionamento aconteça, essas pessoas deixam de ser proativas e passam a ser reativas [114].

Estudos apontam [113, 115, 116, 117, 55] que no setor público essa proatividade é mais evidenciada no início da carreira dos recém concursados que são estimulados pelos novos desafios, vontade de mudança, jovialidade. Contudo, após alguns anos essa motivação tende a diminuir.

Na DES pode-se observar que, ao longo dos anos, cada servidor foi se identificando com um ou outro sistema que melhor lhe agradava e se isolando dos demais. Muitas vezes são sistemas desenvolvido e mantidos pela mesma pessoa durante anos, havendo em certos casos, um acordo de não intervenção no trabalho um do outro. Isso acarreta uma alta dependência do profissional associado ao sistema que só ele mantém.

Como cada desenvolvedor realiza todo ciclo de desenvolvimento de software na maior parte do tempo sozinho, os testes também ficam a cargo de cada um e, no geral, se limitam a testes funcionais. Embora exista um núcleo de testes de software, ele é limitado aos testes dos produtos da fábrica. A cultura de testes de software na DES encontra-se em fase inicial.

Devido à deficiências no processo de desenvolvimento atual, as modificações são feitas às pressas, algumas entregas acabam não atendendo aos pré-requisitos, a documentação e testes são postergados, tendendo a transformar o software numa “Imensa bola de lama”

[118] e, em alguns casos, nem ocorrem. Isso tem se tornado um processo crítico no órgão. Todavia, mesmo com estas deficiência no processo, há de se considerar que, de modo geral, destacam-se os esforço individuais dos profissionais da DES.

5.3.1 Núcleo Técnico de Sistemas Administrativos (NTSA)

Dentre as subáreas da DES, tem-se o Núcleo Técnico de Sistemas Administrativos (NTSA). Esse núcleo é responsável por diversos sistemas, de porte pequeno e médio, tais como Sistema de Processos Administrativos, Sistema de Ponto Eletrônico, Sistema de Precatórios, Sistema de Orçamento, Sistema da Junta Médica, Sistema de Teleatendimento e outros.

Para desenvolver e manter os sistemas deste núcleo, estão disponíveis dez analistas, com faixa etária média de 40 anos, graduados em tecnologia, quase todos com pós-graduação lato-sensu e alguns stricto-sensu, coordenados pelo autor dessa pesquisa.

Os analista estão sujeitos a desenvolver todas as fases do ciclo de desenvolvimento do software, gerando situações de descontentamentos e estresse.

O NTSA possui sistemas deenvolvidos em *JAVA*, *PHP* e alguns em *Ruby on Rails*. A quantidade de tecnologias desalinhadas dentro da equipe também contribui para o clima de descontentamento e individualismo.

É comum que as aplicações sejam desenvolvidas em *localhost* e que os servidores de aplicação e outras ferramentas sejam instaladas gerando conflitos, erros e perda de tempo com atividades que não são de desenvolvimento.

O controle de versões não é um consenso e existe certa resistência à mudança para unificação. São feitos pelo *Apache Subversion* e também pelo Git à depender do sistema. Os sitemas *PHP* utilizam mais o Subversion e os sistemas escritos em *Java* e *Ruby* utilizam o Git.

5.4 Considerações Finais do Capítulo

Apresentou-se nesse capítulo, informações referentes ao objeto de estudo, enfatizando a área de Desenvolvimento de Sistemas do TJGO o qual esta pesquisa propõe melhorias no processo de manutenção de software. No Capítulo 6 apresenta-se a execução metodológica planejada, com a aplicação dos ciclos da DSR, seguido do Capítulo 7 em que se descreve a análise e interpretação dos dados.

Capítulo 6

Manutenção de Software no TJGO - unidade piloto NTSA

6.1 Considerações Iniciais do Capítulo

Neste capítulo, conforme a estratégia metodológica, apresenta-se a execução da DSR, realizada na área de desenvolvimento de sistemas administrativos do TJGO, expondo os ciclos da DSR, os artefatos gerados e as mudanças no processo de manutenção de software do TJGO.

6.2 Contexto

Alguns dos maiores desafios enfrentados por órgãos públicos, no setor de TI, por ordem de prioridade, são [1]:

- 1. Mudanças culturais;
- 2. Resistência à mudanças;
- 3. Obter colaboração dos clientes;
- 4. Customizar práticas ágeis;
- 5. Comprometimento da alta gestão.

Para se estabelecer uma mudança no processo de manutenção de software, como propõe este trabalho, tais desafios devem ser superados. Mas para vencer alguns desses

desafios, faz-se necessário a utilização de técnicas de gestão de pessoas, que não serão abordadas.

A utilização dos métodos ágeis ajudam a superar esses desafios, isso acontece porque são métodos flexíveis e capazes de trazer resultados rápidos e com qualidade.

Entretanto, conhecer as vertentes da manutenção, os requisitos para utilização dos métodos e as boas práticas é fundamental para, de fato, se definir um processo e assim, não cair no erro de achar que os métodos ágeis, por si só, cumprem essa tarefa.

Face ao exposto, identificou-se na revisão da literatura [21, 26, 70, 23, 72, 12, 32, 48, 22] quatro pontos principais que favorecem a definição de um processo de manutenção ágil em órgão públicos (Figura 6.1):

- **Arquitetura**: Definir uma arquitetura de *Deployment* que suporte a entrega contínua, rápida recuperação e trabalho integrado entre as equipes;
- **Processo**: conhecer as vertentes da manutenção e adaptar os métodos ágeis necessários para cada situação;
- **Testes**: realizar testes para melhorar a qualidade e suportar a agilidade;
- **Pessoas**: gestão de pessoas com foco constante na busca de apoio à mudança.

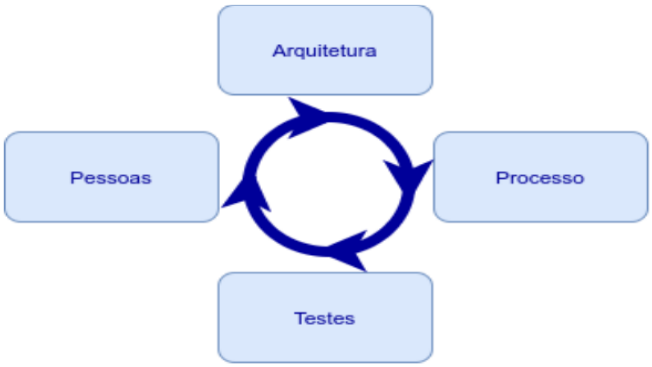


Figura 6.1: Fatores que favorecem a definição de um processo de manutenção com ágeis (Fonte: Autoria própria)

.

Os aspectos da *gestão de pessoas* não fazem parte do escopo deste trabalho. Logo, apresenta-se neste capítulo, o emprego da *Design Science Research* como método para se chegar no objetivo geral desta pesquisa explicativa: "definir e avaliar a implantação de um processo de manutenção de software para o TJGO, empregando metodologias ágeis", através dos pilares: Arquitetura, Processo e Testes.

Os ciclos da DSR foram realizados após a observação dos problemas no *contexto social* e do conhecimento adquirido com a revisão bibliográfica, que representa o *contexto do conhecimento.*

Na DSR adotada (Capítulo 4, Figura 4.2), cada ciclo é composto pelas fases: Investigação do problema, Design da Solução, Validação, Implementação e Avaliação. Foram realizados 04 ciclos de DSR, conforme:

- Ciclo de Experimentação da Arquitetura de *Deployment*;
- Ciclo de Experimentação do Kanban;
- Ciclo de Experimentação do Scrum, de Rehman et al. [3]; e
- Ciclo de Experimentação de Metodologia Híbrida sobre o processo de Rehman et al [3].

Nas seções seguintes são descritos: *Problemas Observados* no *Contexto social* do TJGO; a execução de cada um dos ciclos DSR planejados com base em respostas de soluções semelhantes obtidas no *Contexto do Conhecimento.*

6.3 Problemas Observados no Contexto Social

Retomando o Contexto Social do TJGO, descrito no Capítulo 5, O TJGO é um órgão do Poder Judiciário brasileiro, composto por magistrados e servidores. Possui, atualmente, uma diretoria de TI subordinada à Presidência dessa Corte, dividida entre várias áreas (Capítulo 5, Figura 5.2) sendo as principais para essa pesquisa: Diretoria de TI (DTI), Divisão de Infraestrutura e Tecnologia (DIT), Divisão de Engenharia de Software (DES ou CES[1]) e subordinado à DES, está o Núcleo Técnico de Sistemas Administrativos (NTSA). A seguir, apresenta-se uma breve descrição de cada divisão:

- **Diretoria de TI**: compete ao gestor dessa divisão, gerir toda área de Tecnologia da Informação do TJGO;
- **Divisão de Infraestrutura Tecnológica**: compete a essa divisão a gestão dos servidores de aplicação, banco de dados, redes e outros;

[1]Em 2021 a DES passou a se chamar CES (Coordenadoria de Engenharia de Software), mas devido ao andamento avançado dessa Dissertação continuar-se-á utilizando a nomenclatura de DES

- **Divisão de Engenharia de Software (DES)**: compete à DES desenvolver e manter sistemas computacionais. Ela se subdivide em: Núcleo de Sistemas Administrativos, Núcleo de Sistemas Judiciais, Núcleo de Internet e Intranet, e outros;
- **Núcleo Técnico de Sistemas Administrativos (NTSA)**: compete à NTSA desenvolver e manter sistemas administrativos no âmbito do Tribunal. São exemplos: Sistema de Processos Administrativos (Proad), Sistemas Financeiros, Sistema de Ponto e outros. Trata-se da unidade de pesquisa principal, dado que o pesquisador é servidor do TJGO há mais de treze anos e Coordenador do NTSA há mais de três anos. Para cumprir a missão de definir um processo de manutenção de software, o atual Coordenador do NTSA possui apoio dos respectivos diretores da DES e da DTI para realizar a transformação e possivelmente servir de piloto para demais áreas da DES. Como motivadores externos tem-se apoio do CNJ, por meio de suas resoluções [37] e Resoluções do Próprio TJGO [119]. O NTSA é composto de uma equipe própria de dez pessoas, com faixa etária média de 40 anos, todos servidores de carreira formados em Tecnologia, quase todos com pós-graduação lato sensu e alguns com stricto sensu, com média de 10 anos de emprego no TJ, divididos entre os cargos de analistas e de desenvolvedores de sistemas.

Na investigação dos problemas, na DES, observa-se nos últimos dez anos de serviço no TJGO algumas tentativas frustradas de emprego de processos de desenvolvimento de software:

- Em meados de 2013 iniciou-se um estudo, planejamento e documentação com base nos modelos *Workflows* de desenvolvimento. Foram definidos papéis, artefatos e fluxos necessários ao desenvolvimento. O objetivo principal foi organizar a produção de software, pois já se sentiam os efeitos causados por um ambiente sem orientação, por exemplo: sistemas repetidos entre equipes diferentes, auto índice de manutenções, retrabalhos, sobrecarga dos recursos humanos, baixa auto-estima, pouco trabalho colaborativo, insatisfação da alta direção e uma TI reativa e impotente;
- Na mesma época, observou-se uma iniciativa da Diretoria de Gestão Estratégica em manter os projetos de sistemas alinhados à gestão estratégica por meio de ferramenta própria baseada no PMBOK, mas houve muita resistência e a ferramenta não conseguiu acompanhar as exigências. A iniciativa se extinguiu com a troca de gestão da presidência do Tribunal (essa troca ocorre de dois em dois anos);

- Próximo de 2018, uma iniciativa interna utilizando o *Redmine*. Como as equipes estavam sobrecarregadas de demandas sem processo, acabaram resistindo ao uso. Passaram a ver a inciativa como apenas mais um trabalho desnecessário dada a cultura extremamente técnica da divisão.

Praticamente nenhum novo sistema foi desenvolvido nos últimos cinco anos na divisão de sistemas administrativos do TJGO. Ainda assim, o trabalho é intenso para se manter os sistemas legados, sendo que alguns, com tecnologias desatualizadas, requerem mão de obra exclusiva.

É comum que desenvolvedores reclamem da quantidade de linguagens de programação e peculiaridade de instalação de cada uma delas, o que os levam a se isolar naquelas tecnologias que mais dominam.

Ao alimentarem a cultura do individualismo, os desenvolvedores se tornaram altamente especializados, gerando uma extrema dependência de seus conhecimentos sobre as regras de negócios e domínios tecnológicos.

Diante dessa situação, criou-se o hábito dos demandantes dos sistemas interagirem diretamente com os desenvolvedores. A interação ocorre principalmente por telefone, e-mail e presencialmente a qualquer hora. Essa situação gera bastante pressão e todas as demandas se tornam urgentes e com prazos, algumas vezes absurdos. Isso resulta em um clima tenso de sobrecarga e estresse diante da quantidade crescente de demandas sem controle e principalmente sem alinhamento estratégico, o que causa prejuízos, não apenas aos desenvolvedores, mas também aos interesses institucionais.

Além dos prejuízos citados, as demandas sem controle resultam em entregas indiscriminadas em produção. Em seguida, o próprio cliente valida e testa se as funcionalidades estão satisfatórias, o que resulta em novas demandas corretivas, evolutivas e urgentes, num ciclo infinito de manutenções.

Em 2018, o TJGO investiu na contratação de uma Fábrica de Software para suprir as demandas de sistemas do Poder Judiciário goiano. De acordo com os diretores de TI, o objetivo da contratação de uma Fábrica de Software é adequar o TJGO às tendências Nacionais de desenvolvimento. Na contra-mão dos resultados esperados, a contratação da fábrica de software tem provocado ansiedade e desconforto dos integrantes das equipes.

Como integrante Coordenador da equipe NTSA e pesquisador, analisando os problemas no contexto social do TJGO, o autor desta dissertação, na expectativa de que os problemas observados diminuíssem, chegou na seguinte pergunta da pesquisa: “como de-

finir um processo de manutenção de software para o TJGO, empregando a metodologia ágil num cenário de órgão público brasileiro, com equipe interna de desenvolvimento?".

Após alguns meses de pesquisa esperava-se que o time estivesse de acordo com as mudanças que estavam sendo realizadas e então realizou-se um questionário para aferir esse sentimento. O questionário pode ser visto no APÊNDICE C) e demonstra unanimidade quanto ao desejo de mudanças conforme pode ser visto na Figura 6.2.

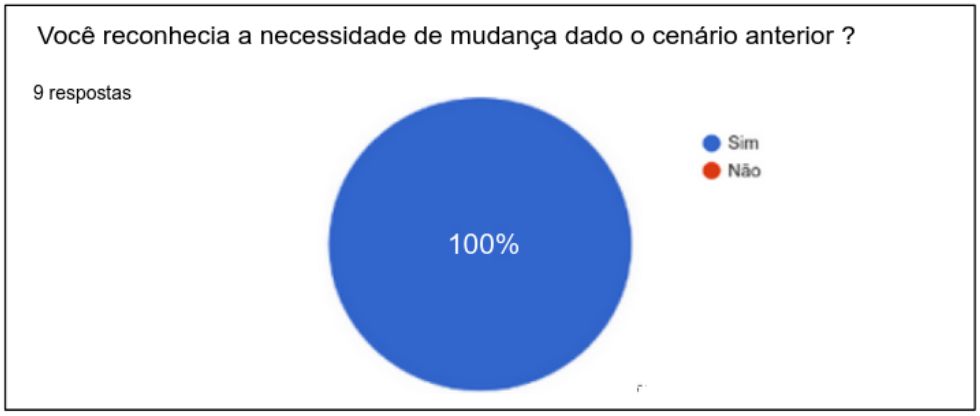


Figura 6.2: Time NTSA demonstra vontade de mudança (Fonte: Questionário APÊNDICE C).

Os clientes também foram observados e cientes das dificuldades do setor, esses se mostraram entusiasmados com as mudanças, conforme pode ser visto na Figura 6.3, em que 62%, dos que responderam ao questionário, se viram muito satisfeitos. Mais detalhes no APÊNDICE H.

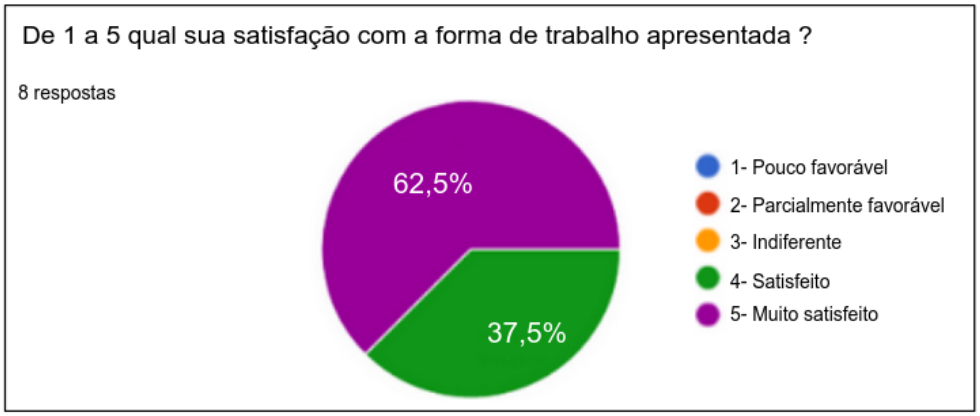


Figura 6.3: Aprovação das mudanças pelos Clientes (Fonte: Questionário APÊNDICE D).

O estímulo ao trabalho em equipe, frequentes reuniões de *brainstorming*, iniciativas sobre a implantação de uma cultura de testes, participação dos interessados durante todo ciclo do projeto, recorrentes *feedbacks* aos clientes, documentação e formalismos adequados, deram início à entrega de melhores produtos, conforme pode ser visto na Figura 6.4, em que 100% dos respondentes se mostraram, ao menos, satisfeitos com as entregas. Detalhes do questionário no APÊNDICE D.

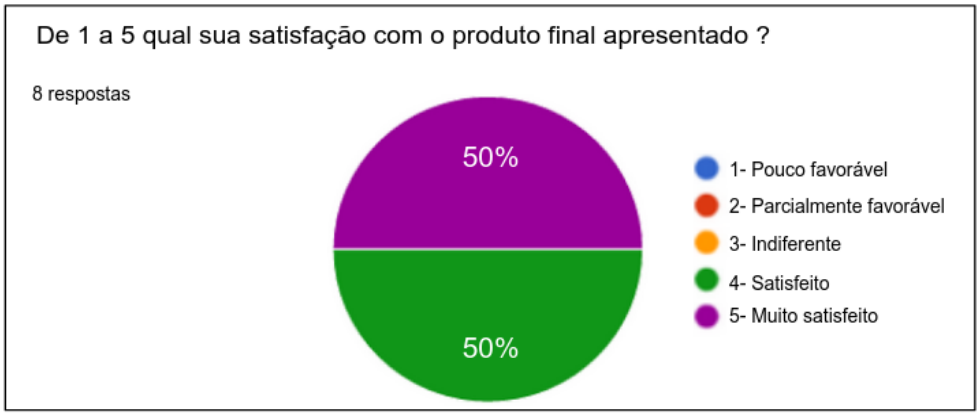


Figura 6.4: Satisfação com produto entregue (Fonte: Questionário APÊNDICE D).

Os questionários durante o andamento da pesquisa foram importantes para se compreender as percepções sobre os impactos das mudanças e possibilitar novo direcionamento.

A seguir, apresenta-se o ciclo de *Design Science* para cada classe de problema identificado: arquitetura de *deployment*, experimentação do Kanban, experimentação do Scrum, experimentação de metodologia híbrida.

6.4 Definir uma Arquitetura de *Deployment*

De acordo com as pesquisas bibliográficas [12, 26, 76, 84, 90, 82, 61, 120] e considerando o contexto do Tribunal de Justiça, para se sustentar manutenibilidade ágil e outros atributos de qualidade (requisitos não funcionais) são necessárias intervenções arquiteturais. Desse modo, buscou-se construir um artefato, utilizando-se DSR, capaz de atender aos requisitos de CI/CD, agilidade e qualidade necessários aos processo de manutenção utilizando-se métodos ágeis. Cada fase do ciclo de *Design Science* será descrito nas seções seguintes.

6.4.1 FASE Investigação do Problema

Diante do contexto social observado, uma arquitetura de *deployment* poderia diminuir os seguintes problemas:

- Reclamação dos desenvolvedores que possuem dificuldades em configurar e manter diferentes servidores de aplicação em *localhost* diante do número alto de sistemas e suas tecnologias;
- Dependência de *deploys* manuais em produção;
- Ausência de ambientes apropriados para desenvolvimentos e testes;
- Individualismo;
- ausência de controle de versões.

6.4.2 FASE Design da Solução

A capacidade de um sistema de software produzir resultados corretos não é útil se levar muito tempo para fazer isso ou se o sistema não permanecer ativo tempo suficiente para entregar o resultado, ou mesmo se o sistema revelar os resultados para pessoas indesejadas. Nos requisitos arquiteturais são onde essas preocupações são abordadas. [5]

> Criar uma arquitetura de software é uma tarefa crítica no desenvolvimento de sistemas de software porque as estruturas do projeto irão ditar se o sistema exibirá boa modificabilidade, desempenho, interoperabilidade e outras qualidades [76].

Uma das importantes condições para suportar as entregas ágeis é estabelecer uma arquitetura de entrega *Contínua e Integrada* (CI/CD) [88, 87, 18]. Assim, o planejamento compreendeu uma arquitetura capaz de suportar mudanças rápidas com segurança com baixo prejuízo dos sistemas em funcionamento, que possibilitasse a aproximação dos desenvolvedores ao trabalho colaborativo com foco no desenvolvimento e menos em configurações ou instalações. A visão de *deployment* dessa arquitetura pode ser vista na Figura: 6.5 e a dinâmica do fluxo, será explicada na Seção 6.8.2.

Conforme os problemas observados, a dificuldade em se instalar e manter diversos servidores de aplicação em *localhost* gera situação de estresse e perda de tempo. Desse modo, planejou-se uma arquitetura de CI/CD em que o *dev* pudesse enviar seus códigos em *localhost* para um ambiente compartilhado (Desenv1), facilmente escalonável (DesenvN). Conforme literatura especializada: no desenvolvimento ágil, existe a necessidade

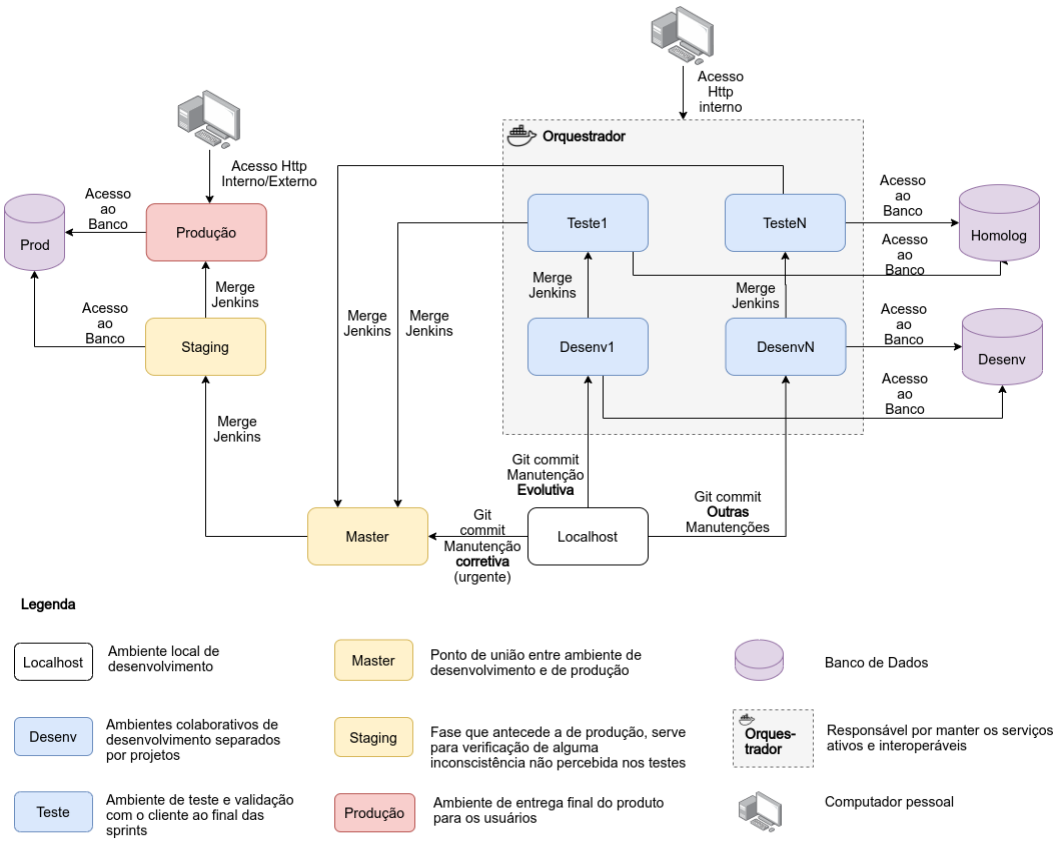


Figura 6.5: *Design* da Arquitetura de *Deployment* (Fonte: autoria própria).

do trabalho colaborativo, no entanto, unir o código pode ser lento e trabalhoso, ainda mais se o *dev* tiver seu próprio ambiente local, sendo que o ideal é que a equipe trabalhe em um mesmo ambiente compartilhado baseado em nuvem [88].

De acordo com os estudos, a manutenção deve ser testável, seja ela evolutiva ou urgente, compreendendo testes de integração e regressão [3].

Os testes durante todo ciclo de desenvolvimento são essenciais para metodologias ágeis [26, 27].

Diante do exposto, foram definidos os ambientes de testes: Teste1, TesteN, *Staging* em que o Teste1 e TesteN são ambientes utilizados durante as manutenções evolutivas, e assim, o testador e o cliente podem realizar testes em um ambiente controlado ao final de cada *sprint*. Já o *Staging* é o ambiente semelhante ao produção utilizado para testes antes de se enviar o código para produção, após o término de uma manutenção evolutiva ou corretiva. No *Staging* é onde se faz testes de sistema, de integração e regressão.

6.4.3 FASE Validação

Após a geração de um artefato, novas investigações e confrontações no contexto do conhecimento foram realizadas e em alguns casos geraram alterações no artefato antes de ser experimentado na prática. As ferramentas utilizadas foram discutidas com os especialistas da área de Infraestrutura (DIT), que possuem parceria com consultores da empresa Red Hat e se dispuseram a ajudar.

Para a *validação* foram feitos questionamentos sobre a aplicabilidade do artefato. Aqui há uma diferença em relação a fase de *avaliação*, pois neste momento, na *validação*, pode-se gerar pequenos ajustes mesmo que estes ajustes não estejam relacionados diretamente com o problema [2], podem ser ajustes técnicos necessários para instalação, ajustes que envolvam outras pessoas, equipes, ou mesmo adequações às normas internas ou externas.

6.4.4 FASE Implementação

Na implementação colocou-se o artefato em prática, isso é, no contexto social, consolidando a junção entre ciência e prática com propósito de solucionar os problemas causados pela ausência de processo de manutenção dos sistemas no TJGO. A equipe da Red Hat disponibilizou a ferramenta Openshift para se construir uma arquitetura de *deployment*. Detalhes sobre o fluxo de

Todas as ações a seguir foram tomadas em conjunto com as equipes de Infraestrutura de TI (DIT):

- Inicialmente, com intenção de aproximar a equipe e gerar colaboração, buscou-se uma ferramenta de controle de versões que possibilitasse a união dos códigos fontes de forma segura. Assim, escolheu-se uma das ferramenta mais utilizada atualmente, que é o Git e um *client* foi também instalado para controle e gestão: GitLab.
- Numa segunda ação, buscou-se suprimir as dificuldades de instalação e configuração observadas na etapa de investigação, adotando-se a utilização de *containers* com Docker. Esses proporcionaram maior portabilidade das aplicações e direcionamento dos desenvolvedores para o foco no desenvolvimento sem terem que se preocupar com questões de instalação, compatibilidade de versões e configurações que demoravam horas ou dias para serem feitas;

- A partir dos *containers*, novas oportunidades surgiram, como a possibilidade de se realizar *deploys* automatizados utilizando-se CI/CD seguindo o design da Figura 6.5, inserção dos sistemas do TJGO nos conceitos de microsserviços, integrações através de APIs *Legancy Wraper*, alta-disponibilidade, escalabilidade, deployabilidade, manutenibilidade, conforme poderá ser constatado na fase de *Avaliação* seguinte.

6.4.5 FASE Avaliação

A utilização da automatização por meio de CI/CD é considerada essencial para emprego dos métodos ágeis, como exposto por Vassallo et al. [86].

Nessa fase de avaliação procurou-se avaliar se o artefato foi capaz de diminuir o problema ao ser aplicado no contexto social do TJGO. De acordo com Wieringa [2], a avaliação da implementação é semelhante à fase de "investigação do problema", mas com objetivo diferente, isso é, seu propósito é avaliar impactos no *problema* após o artefato ter sido aplicado no contexto do problema original.

Nesse sentido, o artefatos da arquitetura de *deployment* possibilitaram, entre outras coisas:

- Avaliar pontos de manutenções recorrentes e seus motivos;
- Diminuiu a quantidade de *deploys* indiscriminadamente em produção, que era uma fonte constante de reclamação dos clientes pela quantidade de erros gerados e também pelas falhas de comunicação e ausências de treinamentos;
- Melhorou o engajamento da equipe;
- Diminuiu as dificuldades em se instalar e manter diversos servidores de aplicação, focando no desenvolvimento;
- Possibilitou organização das entregas,
- O processo de *deploy* automático evita erros e inconsistências de ambientes e sistemas;
- A ferramenta Git permitiu compartilhamento e versionamento do software, além da melhoria do trabalho em equipe;
- A arquitetura de *deployment* possibilitou a contrução de ambientes separados para se dar início aos testes. São eles: Teste e Staging.

- A arquitetura de *deployment* possibilitou o escalonamento horizontal dinâmico de recursos, gerando alta disponibilidade dos sistemas.

6.5 Experimentação do Kanban

Após a construção da arquitetura de *deployment*, procurou-se na literatura o próximo passo para responder a questão de pesquisa. De acordo com o Contexto do Conhecimento, as empresas estão cada vez mais adotando o Kanban para manutenção, pois trata-se de uma técnica que demonstra bons resultados, por oferecer visibilidade aprimorada ao projeto, aumento da qualidade do software, motivação da equipe, melhor comunicação e colaboração [73].

6.5.1 FASE Investigação do Problema

Após aplicação das ferramentas de CI/CD no contexto do TJGO, muitas melhorias foram percebidas conforme demonstrado na fase de Avaliação. Entretanto, muitos problemas continuaram, tais como:

- Individualismo;
- Dificuldade em lidar com as demandas de manutenção;
- Elevada quantidade de erros em produção;
- Dificuldade de sincronismo entre os trabalhos;
- Falhas na comunicação enquanto equipe.

6.5.2 FASE Design da Solução

Com base na revisão bibliográfica (Capítulo 1, Seção 1.2), três métodos ágeis se destacaram: XP, Scrum e Kanban, sendo os dois últimos os mais utilizados para manutenção de Software [121, 4, 22].

A estratégia adotada foi a de apresentar o quadro Kanban ao time e ir evoluindo de acordo com a aceitação e evolução da equipe. Assim, seguindo recomendações da literatura, esperava-se que o Kanban pudesse melhorar aspectos de dificuldade de sincronismo entre os trabalhos e falhas na comunicação, pois o Kanban possui como pontos fortes a capacidade de gerar maior visibilidade sobre o andamento dos projetos e liberdade para execução das atividades [112, 57].

6.5.3 FASE Validação

A validação foi feita levando em consideração a simplicidade da aplicação do método, entusiasmo da equipe em iniciar os trabalhos com metodologias ágeis e o custo benefício da ferramenta, visto que se aproveitou uma das ferramentas utilizadas para a implantação da arquitetura de *deployment*.

6.5.4 FASE Implementação

A ferramenta de apoio utilizada foi o GitLab [9] conforme apresentado na Figura 6.6, visto que após a instalação do controle de versões, ela foi instalada para gestão do CI (*continuous integration*).

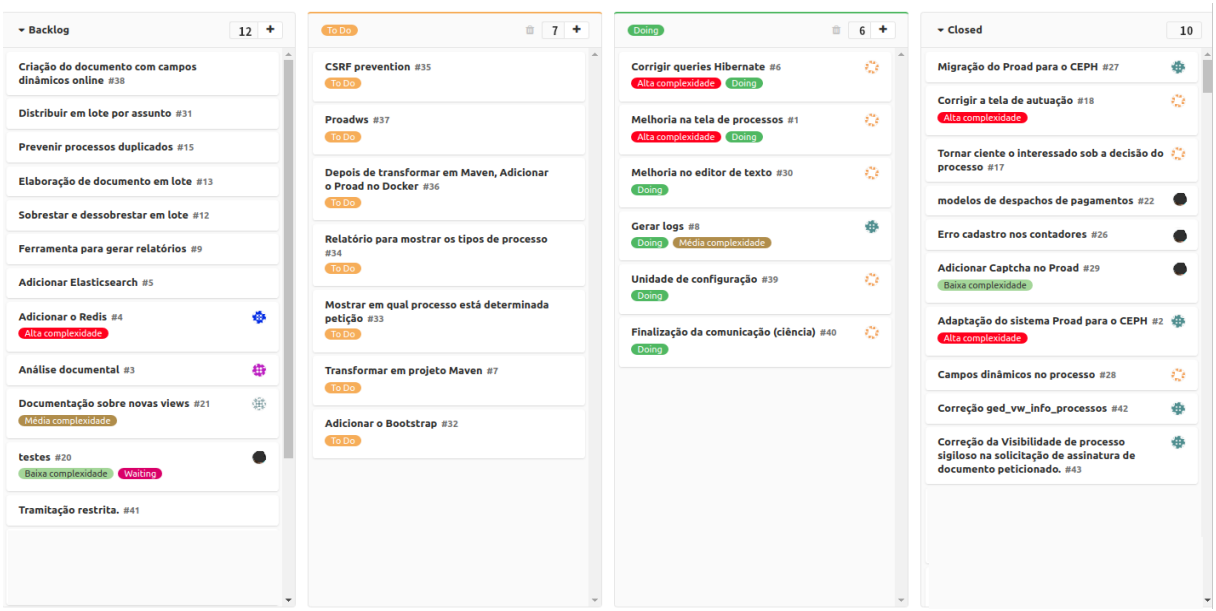


Figura 6.6: Quadro Kanban de manutenção no sistema de processos administrativos do TJGO. (Fonte: [9])

.

O quadro Kanban é um quadro dividido em raias sendo a raia mais à esquerda (*Backlog*) responsável pelo conjunto de demandas a serem feitas. De acordo com a prioridade, as demandas são arrastadas para a raia *ToDo* (fazer). Cada dev tem a liberdade de escolher aquela demanda que possui maior afinidade e então mover para próxima raia *Doing* (fazendo) para representar que está fazendo a demanda e quando concluída, a demanda é por fim movida pelo dev para a raia *Closed* (concluído).

6.5.5 FASE Avaliação

A avaliação inicial foi positiva e o Kanban teve boa aceitação pela equipe. Na Figura 6.6 observa-se que algumas tarefas foram concluídas (*Closed*).

Com o tempo, as tarefas começaram a não ser executadas. Diversas demandas de usuários gestores eram entregues diretamente aos desenvolvedores, muitas vezes por telefone, passando na frente de outras demandas como de costume. Os desenvolvedores passaram a ver o quadro Kanban como uma tarefa a mais e desnecessária, sendo priorizadas as demandas que os clientes levavam pessoalmente ou pediam por telefone, restando às tarefas do Kanban serem esquecidas no quadro.

6.6 Experimentação do Scrum

Conforme descrito na fase de Avaliação do Kanban, embora seja uma ferramenta amplamente utilizada para manutenções, ela não foi suficiente para se estabelecer um processo de manutenção no TJGO no seu contexto específico.

6.6.1 FASE Investigação do Problema

Ao experimentar o Kanban no contexto do TJGO, percebeu-se um entusiasmo por parte dos desenvolvedores, porém baixo interesse dos diretores em acompanhar a execução das demandas. Por conseguinte, as atividades atribuídas aos desenvolvedores e adicionadas ao quadro Kanban foram esquecidas em detrimento das demandas de outras diretorias passadas, na maioria das vezes, por telefone. Com isso, foram mantidos os mesmos problemas investigados nos ciclos DSR anteriores. São eles:

- Individualismo;
- Dificuldade em lidar com as demandas de manutenção;
- Elevada quantidade de erros em produção;
- Dificuldade de sincronismo entre os trabalhos;
- Falhas na comunicação enquanto equipe.

Diante da falha ao se tentar utilizar o Kanban no contexto do TJGO, supõe-se que fatores culturais como a liberdade em excesso e a forma como as demandas de manutenções eram feitas, pudessem estar prejudicando. Este F fato comprova os estudos

que relatam que um dos principais desafios para implantação dos métodos ágeis é o de promover a mudança cultural na organização [28, 56, 57, 30, 29, 55].

Desse modo, utilizar um método mais prescritivo, um método que também é bastante utilizado tanto para desenvolvimentos quanto para manutenções, apresentou ser uma alternativa de sucesso.

6.6.2 FASE Design da Solução

Conforme Ribeiro e Domingues [28], o Scrum foi capaz de mitigar os problemas de controle e gestão observados no setor público.

A ideia de utilizar um método mais prescritivo e que ofereça maior controle, remete a condução da pesquisa a experimentar o Scrum. Assim, esperava-se que o compromisso promovido pelas *sprints*, *timeboxing* de reuniões, transparência com cliente e estimativas de prazos favorecessem a mudança cultural.

A literatura aponta dificuldades de se mudar do Kanban para o Scrum, tais como: falta de visibilidade do trabalho, dificuldades em priorizar as tarefas, falhas na comunicação e colaboração, sobrecarga devido aos compromisso exigidos pelas *sprints*, dificuldades em sincronizar os trabalho ou substituir pessoas [73]

Heeager e Rose [25] revelaram os desafios de se adotar Scrum para processos de manutenção. Eles relataram que muitos dos problemas de se aplicar Scrum para manutenção se dá pelo fato de que uma emergência não pode aguardar o cumprimento de uma *sprint* com todos seus requisitos e quando acontece, acabam atrasando o cumprimento das metas. Assim, eles sugeriram que os projetos evolutivos sejam planejados com um tempo extra, porque, caso haja alguma emergência, torna-se possível atendê-la sem muitos impactos na manutenção evolutiva em curso [25].

Pino et al. [72] aconselham o uso de *sprints* separadas para a manutenção, sendo uma mais breve para correções de emergências imprevistas e outra maior com planejamento para manutenções evolutivas.

Diante do exposto, para lidar com os estados de manutenção, a simplicidade e agilidade devem ser maximizadas [23, 3, 25], dessa vez, adotou-se a estratégia de agir de forma simples e objetiva, sem depender de ferramenta de software para sua implantação e assim ter liberdade para se aplicar as boas práticas de diversos autores sem ter que seguir as regras funcionais de alguma ferramenta específica.

Para minimizar as falhas de comunicação, aumentar a visibilidade do trabalho, e suprimir a ausência de um sistema de gestão, foi gerando um artefato de controle chamado de *"relatório de acompanhamento"*, APÊNDICE B, que ao final de cada *sprint* deve

ser enviado, via e-mail corporativo aos envolvidos, gerando transparência e reforçando a característica de *push* do Scrum.

Além disso, faz-se necessário atentar-se ao tratamento diferenciado para as diferentes categorias de manutenção (adaptativa, corretiva, perfectiva e preventiva), Rehman et al. [3] constataram que quando isso é atendido, os resultados da manutenção com métodos ágeis aumentam os casos de sucesso. Então, utilizando-se os método ágil Scrum, os pesquisadores Rehman et al. [3] propuseram um modelo, conforme apresentado na Figura 6.7.

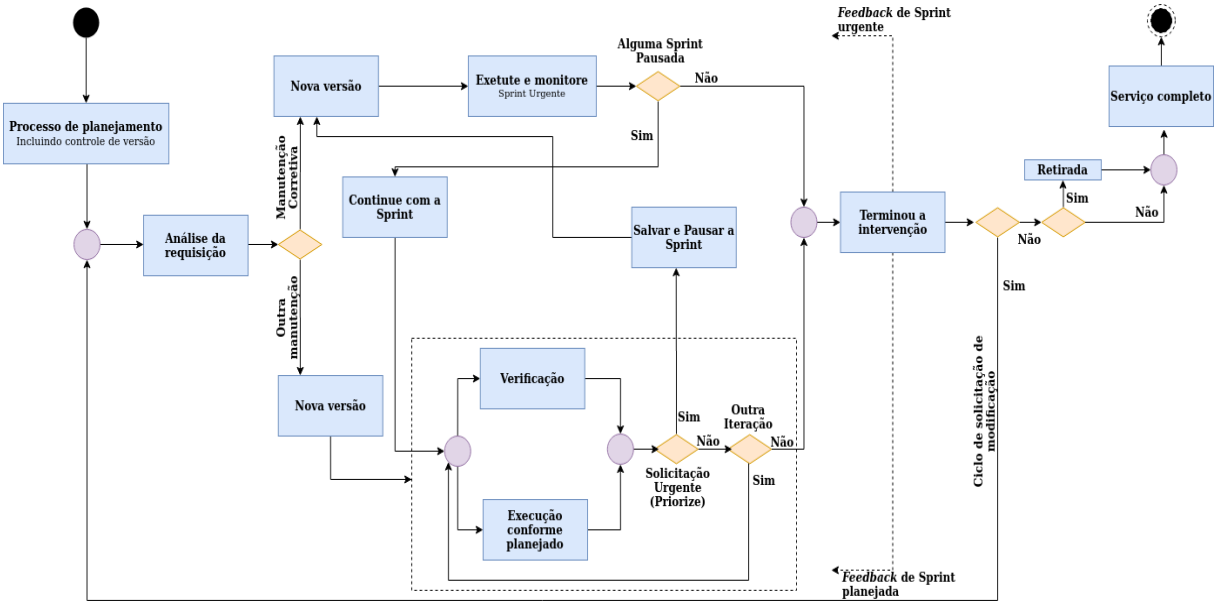


Figura 6.7: Modelo utilizando Scrum para mautenção de software (Fonte:[3]).

O processo de Rehman et al. [3] divide as manutenções em *corretivas* e *outras manutenções* e ambas requerem controle de versão. Seguindo o fluxo *outras manutenções*, percebe-se a utilização do Scrum nas fases *Execução conforme o planejado* (*sprint* após a reunião de planejamento) e *Verificação* (reuniões diárias, de revisão e retrospectiva). Se durante uma correção não urgente aparecer uma solicitação urgente, priorize, *Salve e pause a sprint*, atualize a versão de manutenção urgente e *Execute/monitore*.

Ao final das manutenções, envie uma resposta ao solicitante e se ele estiver satisfeito com a intervenção, encerra-se o processo ou, caso contrário, inicia-se um novo ciclo de manutenção.

6.6.3 FASE Validação

Perante o exposto, sobre as dificuldades encontradas na tentativa de se utilizar o Kanban, dados os aspectos culturais do TJGO, o processo de manutenção de Rehman et al. [3] apresentou ser adequado no TJGO, visto que trata-se de um método um pouco mais prescritivo e também devido sua simplicidade e bons resultados quando empregado em órgãos públicos [28, 54, 1].

6.6.4 FASE Implementação

Num primeiro momento o Scrum foi seguido à risca, com definição de papéis, reuniões diárias, planejamento da *sprint* e de revisão da *sprint*, utilização de *timeboxing*, tamanho das *sprints*, tamanho do time.

Observa-se que no processo proposto por Rehman et al. [3] utilizando o Scrum, na Figura 6.7, existem dois fluxos separados: um para *Manutenções Corretivas* e outro para *Outras Manutenções*, sendo que, em ambos os casos, há necessidade do controle de versões.

Alguns pontos principais do processo de Rehman et al. [3]: o processo se inicia com o "Processo de Planejamento" o qual se faz o rastreamento do controle de mudanças, ajusta-se o controle de versões. Na "Análise da Requisição" verifica-se o tipo da manutenção, sendo que *manutenções corretivas* possuem prioridade enquanto que *outras manutenções* são planejadas e executadas com prazos estimados.

Se alguma manutenção urgente surgir, a *sprint* é pausada e uma nova versão se inicia para tratar a solicitação urgente. Ao final da manutenção urgente, verifica se existe alguma *sprint* pausada e retorna sua execução.

Para cada projeto foi gerado um *relatório de acompanhamento do projeto* (APÊNDICE B), evoluído ao longo da pesquisa. Neste relatório tem-se registrado, entre outras informações, uma descrição sucinta sobre o projeto, seus objetivos, as partes envolvidas, *Backlog* do Produto, *Backlog* da Sprint, prazo estimado (sempre feita com prazo extra para se tratar as manutenções urgentes [3]), prazo realizado, andamento do projeto por meio de código de cores, registro das reuniões do Scrum, análise e mitigação de riscos.

6.6.5 FASE Avaliação

No início dessa fase houve bastante resistência do time e as prescrições do Scrum foram fundamentais para que ocorresse a mudança. Problemas como - "individualismo" -

deram lugar ao trabalho em equipe e, consequentemente, maior qualidade nas entregas e melhorias na comunicação.

O controle de versões e *pipeline* ajudaram no sincronismo dos trabalhos, organização e controle.

Um simples *"relatório de acompanhamento"*, APÊNDICE B, transformou-se numa poderosa ferramenta de gestão, gerando controle, planejamento e visibilidade. A transparência gerada pelo relatório de acompanhamento foi essencial para diminuir as demandas atravessadas.

Os testes em papel passaram a ser feitos na fase de Verificação e no final da intervenção, antes da entrega, reduzindo drasticamente a quantidade de erros em produção.

Entretanto, restava ainda um problema: "dificuldade em lidar com as vertentes de manutenção". O processo de Rehman et al. [3] não deixa claro como deve ser executada a fase: "Execute e monitore", embora se recomende que não se deve ficar preso ao tamanho das *sprints*, ainda assim o Scrum requer planejamentos e reuniões. Além disso, pausar uma *sprint* de manutenção não urgente e deslocar todo o time para uma nova manutenção urgente não pareceu ser uma boa estratégia.

6.7 Adoção de Metodologia Híbrida

Ao se pesquisar sobre métodos ágeis é comum encontrar referências sobre o Scrum, Kanban, Xp e outros, entretanto as pesquisas vêm avançando e muitos pesquisadores afirmam que, em muitos casos, o melhor a se fazer é não se ater a métodos específicos, mas sim buscar uma combinação que melhor atenda às necessidades de cada caso e isso tem levado a grande utilização de processos híbridos [4, 22]. Diante disso, novo ciclo DSR se estabeleceu conforme as seções seguintes.

6.7.1 FASE Investigação do Problema

As prescrições do Scrum foram fundamentais para romper problemas culturais percebidos há décadas no TJGO. Os clientes puderam participar da construção de softwares favorecendo sua aceitação. As estimativas de prazos aumentaram a confiança.

Entretanto, ainda havia o problema das manutenções urgentes. Embora Rhelman et al. [3] deixem claro que as *sprints* poderão ter tamanhos variados, o Scrum ainda prescreve: planejamento, papéis e reuniões que se contrapõem à agilidade que se espera na *sprint* urgente ("Execute e monitore"[3]).

De acordo com a literatura, para lidar com a urgência das manutenções, questões como documentação, planejamento, reuniões devem dar licença para o *foco na solução do problema* [23]. Diante disso, o Kanban se mostra mais indicado para executar tarefas prioritárias, com prazos curtos, como menos de uma semana ou mesmo em questões de horas, com entregas pequenas [4].

6.7.2 FASE Design da Solução

O contexto do conhecimento, representado pela pesquisa bibliográfica, aponta que a utilização do método Scrum e Kanban tem se tornado preferível para lidar com as vertentes da manutenção[121, 4, 22]. Nesse sentido, a fase "execute e monitore" de Rehman et al. [3] pode tirar proveito dos benefícios da utilização do Kanban. O processo resultante é apresentado na Figura: 6.8.

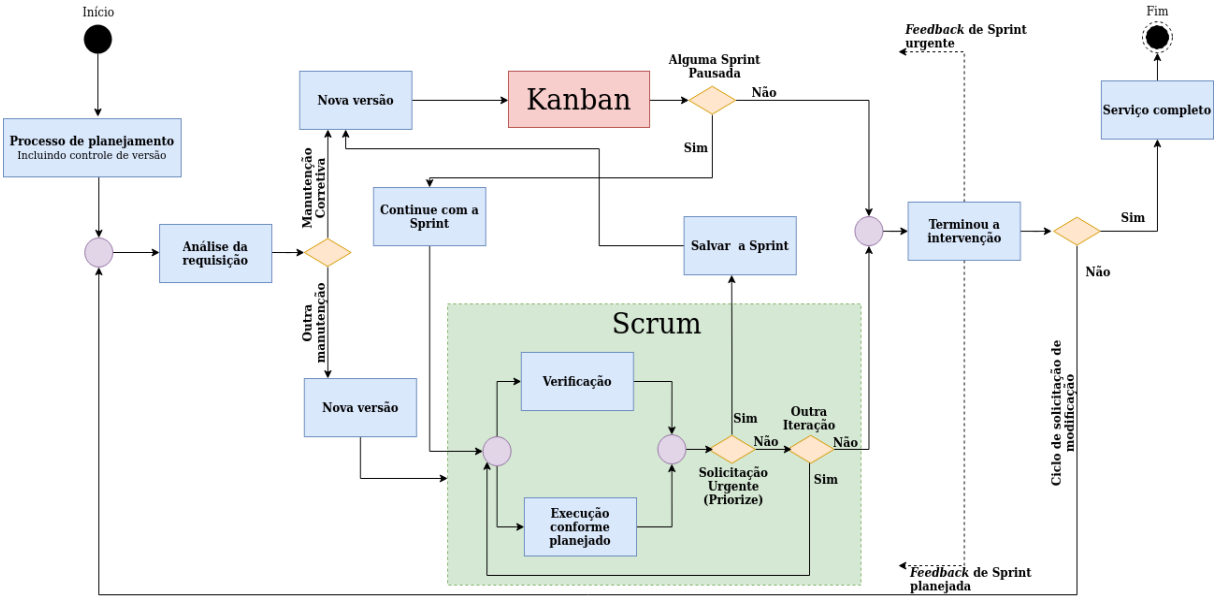


Figura 6.8: Modelo utilizando Scrum e Kanban para manutenção de software (Fonte: adaptado [3]).

6.7.3 FASE Validação

A utilização do Scrum e Kanban tem se tornado comum para manutenções [121, 22] de software, a essa união bem sucedida deu-se o nome de Scrumban[121], que nada mais é que um conceito diante da variedade de processos que podem se formar utilizando-se Scrum e Kanban, de acordo com cada necessidade [121, 4].

Neste trabalho, essa pequena mudança adicionando o Kanban ao processo de Rhelman et al. [3] mostrou ser adequada para a melhoria do processo de manutenção do TJGO.

6.7.4 FASE Implementação

Dado que a utilização do quadro Kanban no NTSA não surtiu os efeitos necessários devido aos fatores culturais, principalmente à inércia do time em adotá-lo, utilizou-se como estratégia o *e-mail institucional* para que todos os envolvidos soubessem o que deveria ser feito, conforme demonstrado na Figura 6.9.

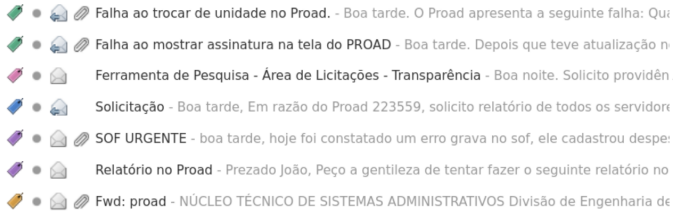


Figura 6.9: Caixa de entrada do e-mail institucional do TJGO (Fonte: [8])

O e-mail foi utilizado como uma adaptação ao Kanban, conforme Figura 6.10.

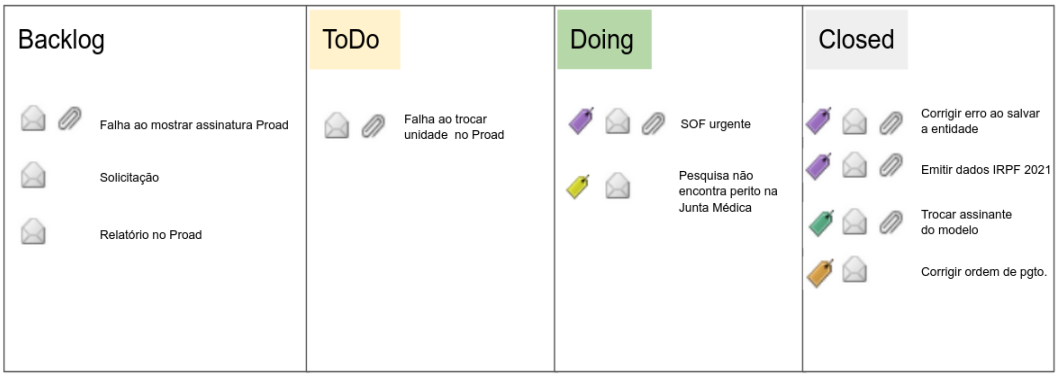


Figura 6.10: Ilustração do e-mail sendo utilizado como quadro Kanban (Fonte: autoria própria)

Considerando a *caixa de entrada do e-mail institucional* como *Backlog*, os e-mails chegam a esse *Backlog* e podem ser espontaneamente resolvidos por alguém do time ou de forma compulsória, em que Coordenador delega para que a manutenção seja iniciada (*ToDo*) prioritariamente.

Quando se inicia a execução da demanda, o Coordenador é avisado pelo executor, situação que representa o *status* (*Doing* ou fazendo) e ao término (*Closed*) um e-mail é enviado para o cliente (*feedback*) fortalecendo os laços de confiança.

Caso esteja ocorrendo uma manutenção evolutiva no Scrum e alguém do time precise executar uma manutenção corretiva via Kanban, isso é anotado no *relatório de acompanhamento*, podendo se transformar em um fator de risco para o projeto de manutenção evolutiva. O desenvolvedor com maior experiência para aquela manutenção urgente incia sua execução e a tarefa dele, *que estava sendo executada na manutenção evolutiva*, é redistribuída para o time buscando o menor impacto nos prazos estimados.

Quanto à estimativa dos prazos, é definido na reunião de planejamento do projeto e ocorre em consenso do time utilizando-se a técnica de *Poker Planning* do Scrum. A partir da estimativa de entrega do projeto, o time planeja a entrega da próxima *Sprint* dentro do *timeboxing*. Tanto para estimativa da entrega do projeto, quanto para as entregas dentro da *Sprint*, estima-se uma margem de segurança para caso seja necessário encaixar atividades urgentes, conforme sugerem Heeager e Rose [25]. Desse modo, não é necessário pausar nenhuma das manutenções em detrimento da outra.

As entrevistas realizadas por Ahmad et al. [69] demonstraram que as equipes de manutenção não conseguiam encaixar as atividades urgentes nas sprints pré-estabelecidas e as datas de lançamentos eram quase aleatórias. Além disso, as reuniões diárias, muitas vezes, duravam bem mais do que o estabelecido e geralmente os membros do time não estavam disponíveis para comparecerem nas reuniões. Neste trabalho, tais problemas foram resolvidos ao se utilizar o Kanban.

6.7.5 FASE Avaliação

Perante a ausência de uma ferramenta institucional de gestão de projetos e considerando os fracassos, o e-mail institucional em conjunto com o *"relatório de acompanhamento"* (APÊNDICE B), possibilitaram que as demandas fossem executadas satisfatoriamente.

Após o término das *sprints*, o *"relatório de acompanhamento"* é enviado, via e-mail (institucional), para os envolvidos gerando maior visibilidade.

O formalismo do envio por e-mail favorece que as demandas sejam cumpridas e o controle dos *deploys* (arquitetura de *deployment*) ajudam no monitoramento e controle, diminuindo assim, outros tipos de demanda como por exemplo: as demandas por telefone, agregando satisfação e qualidade nas entregas.

6.8 Síntese da DSR e Dinâmica do Processo Adotado

Compreender os ciclos da DSR, os dados de entrada e saída de cada fase foi essencial para a compreensão do processo metodológico adotado. Nessa seção apresenta-se uma síntese dos ciclos DSR e uma descrição sobre a dinâmica do processo de manutenção estabelecido no NTSA.

6.8.1 Resumo da DSR

A Tabela 6.1 refere-se à arquitetura de *deployment* que sustenta o processo de manutenção. Nela é possível ver um conjunto de problemas na fase de *Investigação do problema*, a construção do artefato nas fases de *Design da solução e Validação* e a experimentação nas fases de *Implementação e Avaliação*.

Tabela 6.1: Ciclo da Arquitetura de Deployment

Investigação do problema	Ausência de uma arquitetura de *deployment*; Dificuldades técnicas para preparação dos ambientes de dev; Pouco trabalho Colaborativo; Alta quantidade de erros em produção
Design da solução	Uma arquitetura de Deployment utilizando de ferramenta de controle de versões com *containers* e pipeline para deploy
Validação	Percepções dos especialistas da área de infraestrutura de TI
Implementação	Instalação de ferramentas de controle de versão, orquestração e pipeline
Avaliação	A arquitetura de Deployment, em sua versão final, sustentou a entrega ágil com CI/CD; Os *containers* favoreceram o DevOps e o controle de versões facilitou o trabalho em equipe

Após a fase de Avaliação, alguns problemas podem continuar e outros podem surgir, esse conjunto de problemas é enviado como entrada para o próximo ciclo, Tabela 6.2.

O Kanban mostrou-se bastante útil no cenário das manutenções de software, mas em princípio, fatores culturais do TJGO impediram que o método desse certo.

Historicamente, outras tentativas de usar métodos ágeis falharam no TJGO. Na maioria das vezes, as equipes esperavam que a ferramenta tecnológica direcionasse para

solução dos problemas e quando isso não acontecia, acabavam abandonando ou trocando de ferramenta.

Além disso, fatores como demandas por telefone e outros meios informais foram priorizados frente a outras demandas. Isso aconteceu, muitas vezes, sobre a justificativa de se estar desburocratizando e sendo ágil.

Tabela 6.2: Ciclo do Kanban

Investigação do problema	Dificuldade com sincronismo dos trabalhos; Falhas na comunicação da equipe
Design da solução	Do contexto do conhecimento se extrai que XP, Scrum e Kanban são os mais utilizados; Kanban possibilita boa visibilidade e liberdade
Validação	A equipe aceitou com entusiasmo; Custo benefício da ferramenta favorável
Implementação	Aproveitou-se a ferramenta de controle de versões com Kanban integrado
Avaliação	Teve boa aceitação pela equipe. Aumentou a visibilidade dos trabalhos. * A equipe descartou o Quadro Kanban com registros de demandas em detrimento a outros canais (telefone, clientes demandando presencialmente...)

Na tentativa de se diminuir os problemas observados, buscou-se um método um pouco mais prescritivo e sem uso de ferramenta computacional, prezando pela simplicidade, conforme sugerem Kong e Liu. [23]. Assim, como pode ser visto na Tabela 6.3, buscou-se o Scrum.

O Scrum foi adotado à risca inicialmente e assim gerou resistência.

O processo, utilizando Scrum, de Rehman et al. [3] adotado, possibilitou um avanço muito grande nas manutenções evolutivas. Com o tempo os resultados apareceram e o time diminui a resistência, passando a colaborar com a nova fase de mudança.

Entretanto, embora o processo de Rehman et al. [3] tenha contribuído sobremaneira com as manutenções evolutivas, não ficou claro como executar as manutenções urgentes. Após estudos sobre o Scrumban [4], adotou-se uma metodologia híbrida (Scrumban) conforme a Tabela 6.4.

Tabela 6.3: Ciclo do Scrum

Investigação do problema	Aos poucos o time foi deixando o Kanban de lado pois continuavam recebendo demandas paralelas e os problemas se mantiveram
Design da solução	Aumentar o controle: experimentar o Scrum e não usar ferramentas computacionais de modo que se tenha liberdade para experimentar e ajustar
Validação	A literatura recomenda o Scrum para se aumentar o controle, entretanto, recomenda cautela ao migrar do Kanban para o Scrum, e sugere que se preze pela simplicidade
Implementação	Utilização do Scrum à risca inicialmente. Adaptação de um relatório de acompanhamento para controle do *backlog*, das *sprints*, reuniões etc.
Avaliação	O rigor inicial foi importante mas aumentou a resistência. Com o tempo, os ajustes favoreceram a comunicação, colaboração e iniciaram-se os testes de sistema

Tabela 6.4: Ciclo Metodologia Híbrida

Investigação do problema	Embora o Scrum tenha consolidado o processo para manutenções evolutivas, ele não se mostrou apropriado para manutenções urgentes
Design da solução	Encontrar um processo na literatura que atenda às vertentes da manutenção evolutiva ou urgentes
Validação	A literatura sugere o tratamento das vertentes da manutenção de forma separada. Rehman et al. propõem um processo que se adequa às manutenções urgentes e às evolutivas
Implementação	Utilização do processo de Rehman et al. com adaptação
Avaliação	Avaliação interna e externa das partes envolvidas que demostraram satisfação com o novo processo e seus resultados

O uso de um processo com metodologia Híbrida permitiu tratar as vertentes da manutenção separadamente, Tabela 6.4. Assim as manutenções evolutivas são realizadas sem o risco causado pelas manutenções urgentes.

6.8.2 Dinâmica do processo adotado

A arquitetura de *depoyment* definida no TJGO e o processo de Rehman et al. [3] adaptado para manutenções de software pela equipe do Núcleo Técnico de Sistemas Administrativos (NTSA), demonstram sintonia conforme esboça a Figura 6.11.

Essa sintonia repercute nos resultados obtidos pela equipe e demonstrados no Capítulo 7.

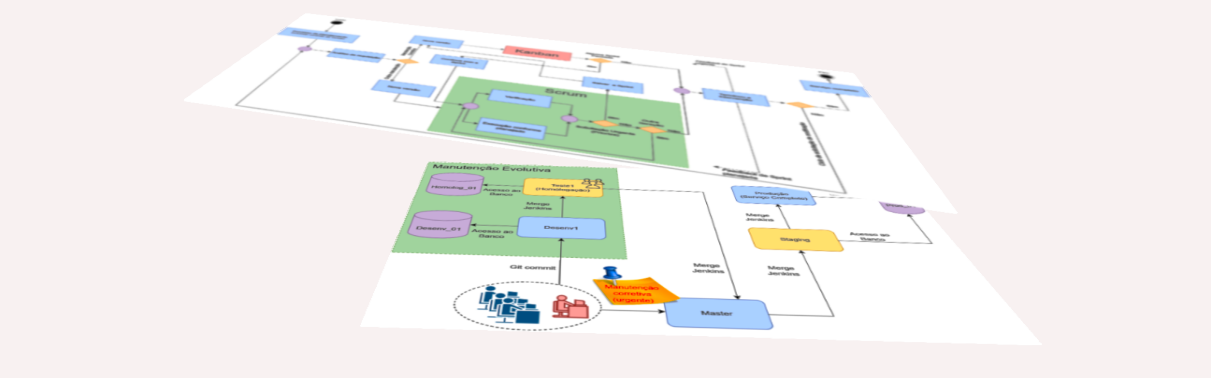


Figura 6.11: Consonância entre o Processo e a Arquitetura de Deployment. (Fonte: Autoria Própria)

.

O artefato gerado pelos ciclo DSR (Tabela 6.1), foi uma arquitetura de *deployment* representada pela Figura 6.12.

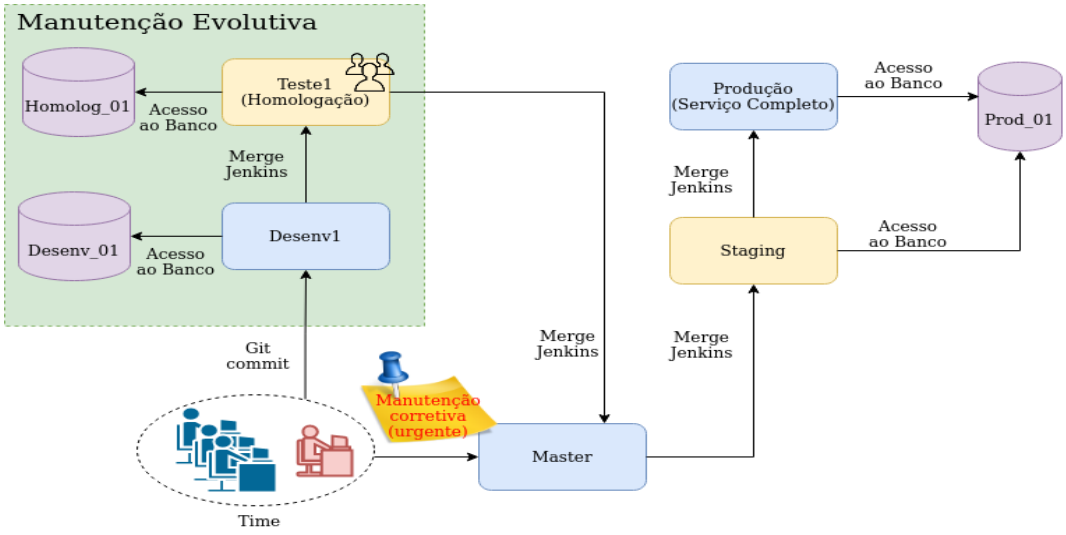


Figura 6.12: Artefato da Arquitetura de Deployment. (Fonte: Autoria Própria)

.

Na Figura 6.12 pode-se observar que existem duas possibilidades para o time enviar seus trabalhos: para o Desenv1 ou para o Master. Essas duas possibilidades correspondem a manutenção evolutiva e manutenção urgente, respectivamente.

Nas manutenções evolutivas, representadas no processo pela área verde Scrum (Figura 6.13), ao final das *sprints* o *entregável* é enviado para o Teste1 (Figura 6.12), onde faz-se a apresentação aos *stakeholders* e revisão do trabalho da *sprint*, documentada no relatório de acompanhamento do APÊNDICE B.

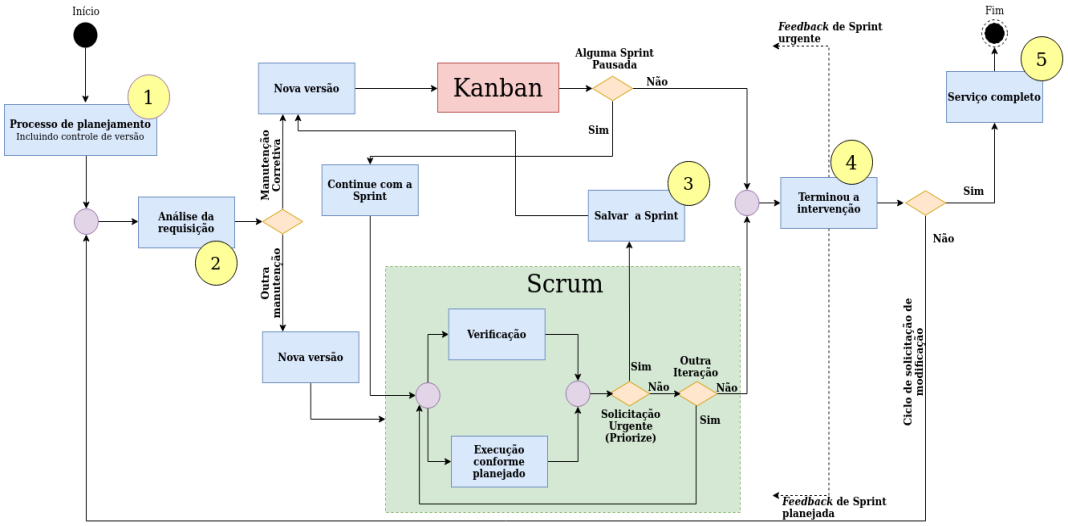


Figura 6.13: Processo híbrido adotado no TJGO. (Fonte: adaptado Rehman et al. [3])

.

Baseando-se nos conceitos de simplicidade de Kong e Liu[23] para tratar as manutenções evolutivas, desenvolveu-se um relatório de acompanhamento de controle e gestão (APÊNDICE B). Nele se faz uma descrição geral do projeto, os objetivos, formação do time, do cronograma, *backlog* do produto, resumo de cada reunião, *backlog* da *sprint*, atrasos, justificativa dos atrasos e riscos do projeto (foram acrescentados alguns conceitos do PMBOK [62]).

O *backlog* do produto é construído após reuniões de levantamento de requisitos utilizando-se histórias de usuários. Os testes iniciam-se nesse momento, e de acordo com cada história de usuário, elaboram-se os casos de teste em forma de *checklist*. Esses testes são executados e atualizados ao final de cada *sprint* no ambiente Teste1 (Figura 6.12) até o final do projeto. Nesse momento, os testes visam, além de identificar previamente os erros, homologar e verificar a usabilidade junto ao cliente.

O processo prevê, antes de tudo, uma fase de *Planejamento* (Figura 6.13, item 1) que nada mais é que verificar e garantir que o controle de versão e os requisitos não funcionais estejam preparados para manutenção ágil.

Logo após, análise da requisição (Figura 6.13, item 2) é realizada buscando classificar se a manutenção é corretiva (urgente) ou algum outro tipo de manutenção, e em seguida gera-se uma nova versão.

Normalmente, quando ocorre manutenção urgente durante uma manutenção evolutiva (Figura 6.13), um membro do time é deslocado para ela. Esse evento é documentado no relatório de acompanhamento APÊNDICE B.

Quando algum integrante é deslocado para resolver uma situação urgente, sua tarefa, no projeto de manutenção evolutiva é redistribuída para o time.

De acordo com Kong e Liu [23], quando uma situação de urgência acontece, ela deve ser priorizada, mesmo que depois tenha que recuperar alguma funcionalidade perdida. Diante disso, Rehman et al. [3] adicionaram a possibilidade de interromper a *sprint* a qualquer tempo, se necessário, conforme mostra a Figura 6.13, item 3.

Conforme Heeager e Rose [25] sugerem, os projetos evolutivos são planejados com um tempo extra visto que as manutenções urgentes podem ocorrer a qualquer tempo.

Assim, planejando e estimando o prazo com margem de segurança para caso surja alguma manutenção urgente [25], ainda não houve a necessidade de se pausar nenhuma *sprint* no contexto de manutenções do TJGO.

Na Figura 6.12 é possível ver o ambiente de *Staging* onde são feitos testes de regressão e integração independente do tipo de manutenção. Esses testes visam verificar as possíveis perdas de funcionalidades citadas por Kong e Liu [23] e possíveis erros em requisitos não funcionais.

Ao final da intervenção, Figura 6.13, item 4, sempre é fornecido um *feedback* ao cliente, conforme propõe os autores [122]. Isso aumenta a satisfação e confiança do cliente. Caso o cliente esteja de acordo e tenha terminado a intervenção, o "Serviço está completo" (Figura 6.13, item 5) e a manutenção é enviada para produção (Figura 6.12).

Os projetos são compostos por times pequenos e ciclos curtos, conforme sugerem Beck e Andres [122]. Caso o projeto demande muito tempo (mais que um trimestre), ele é dividido em mais de um projeto para assegurar as entregas contínuas e facilitar a gestão.

6.9 Considerações Finais do Capítulo

Neste capítulo, apresentou-se a realização de experimentos na área de sistemas administrativos do TJGO utilizando-se a DSR, com o propósito de se encontrar os melhores artefatos capazes de diminuir os problemas enfrentados pela ausência de um processo de manutenção de software.

No Capítulo 7 apresenta-se a Análise de Conteúdo[45] utilizando-se entrevistas e revisão da literatura. Uma análise das citações presentes na revisão bibliográfica são confrontadas com os resultados das entrevistas realizadas com o time de TI (APÊNDICE I), usuários gestores (APÊNDICE H) e diretores (APÊNDICE G).

Capítulo 7

Análise de Conteúdo do Processo de Manutenção do TJGO

7.1 Considerações Iniciais do Capítulo

Após a execução da técnica DSR, em que foram necessários 04 ciclos, apresenta-se neste capítulo uma análise dos dados resultantes do emprego da metodologia de pesquisa e do processos de manutenção definido para o TJGO.

7.2 Análise de Conteúdo

Uma vez concluída a pesquisa, é importante que o artefato desenvolvido, possa ser generalizado de modo que o conhecimento gerado em uma situação específica possa ser aplicado a outras situações similares que são enfrentadas por diversas organizações [43].

Dresch et al. [43] sugerem que se utilize o raciocínio indutivo, pelo qual o pesquisador procura generalizar a solução encontrada para uma classe de problemas.

> As iterações e os documentos são considerados como formas de constituir, de forma conjunta (ou conflituosa), processos e artefatos sociais. Todas essas abordagens representam formas de sentido, as quais podem ser reconstruídas e analisadas com diferentes métodos qualitativos que permitam ao pesquisador desenvolver modelos, tipologias, teorias (mais ou menos generalizáveis) como formas de descrever e explicar as questões sociais [47]

Um conjunto de técnicas utilizadas para fundamentar metodologicamente o raciocínio indutivo é a análise de conteúdo. A análise de conteúdo é um conjunto de técnicas que visam analisar as comunicações, sendo que qualquer meio de comunicação pode

ser analisado, sejam eles textos, livros, entrevistas, questionários, transcrições, áudios e outros. Utiliza procedimento sistemático e objetivos de descrição do conteúdo das mensagens [45].

A intenção da análise de conteúdo é a inferência (deduzir de maneira lógica) de conhecimentos relativos à condições de produção (ou, eventualmente, de recepção), inferência esta que recorre a indicadores (quantitativos ou não). A análise de conteúdo e a análise documental são bastante semelhantes. Se retirar a função de inferência e limitar as possibilidades técnicas apenas à análise categorial ou temática, pode-se identificá-la como análise documental [45].

A partir desses conceitos, utilizou-se entrevistas semiestruturadas e transcrição dessas entrevistas para se conduzir uma fundamentação metodológica baseada na Análise de Conteúdo, utilizando-se a técnica de Análise de Categorias com finalidade de se fazer inferências. Utilizou-se como base bibliográfica o método descrito por Bardin[45] que se divide em três etapas:

- Pré-analise;
- Exploração;e
- Interpretação.

Cada etapa e seus resultados são descritos nas seções seguintes.

7.3 Pré-analise dos Resultados

Na fase de Pré-análise Bardin[45] sugere-se que se faça uma leitura flutuante, que é uma leitura rápida para seleção e escolha dos documentos. Aqui se espera que haja seleção dos documentos que farão parte da análise.

A seleção dos documentos se deu pelas citações feitas nessa pesquisa, representadas pela Dissertação na Figura 7.1, e as entrevistas (demais documentos).

Para as entrevistas, algumas regras foram definidas:

- Entrevistar os dois diretores superiores imediatos. Eles podem ser vistos na Figura 7.1 representados por: *diretor da DES* (diretor imediato do pesquisador) e *diretor de TI* (diretor de toda área de Tecnologia da informação do TJGO);
- Entrevistar os usuários gestores participantes dos dois últimos projetos no NTSA. Eles estão representados na Figura 7.1 por: Gestor 1 e Gestor 2;

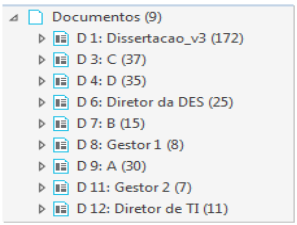


Figura 7.1: Documentos selecionados (Fonte: autoria própria)

- Entrevistar os quatro primeiros analistas que mais participaram de demandas evolutivas e corretivas no último ano. Na Figura 7.1 estão representados pelas letras *A,B,C e D*.

O propósito de se utilizar esses documentos é de se fazer um cruzamento entre a pesquisa bibliográfica, pesquisa documental e entrevistas de modo a buscar comprovações de como foi definido o processo de manutenções do TJGO, na equipe de sistemas administrativos.

As perguntas para diretores, gestores e time tiveram suas respectivas adequações e podem ser vistas no APÊNDICE F junto com o termo de compromisso e ética acordado entre entrevistados e entrevistador.

As transcrições completas das entrevistas podem ser vistas nos APÊNDICES I, H,G.

Além da leitura flutuante, espera-se que tenham os objetivos bem definidos para que a partir deles possam se gerar indicadores. Desse modo, mediante os experimentos e resultados obtidos com a aplicação da DSR, considerando a pergunta de pesquisa: *"Como definir e implantar um processo de manutenção de software para o TJGO, empregando a metodologia ágil num cenário de órgão público brasileiro, com equipe interna de desenvolvimento ?"* e o objetivo geral da pesquisa: *"definir e avaliar a implantação de um processo de manutenção de software para o TJGO, empregando metodologias ágeis"*, formulou-se as seguintes hipóteses:

- h1: devido às particularidades do setor público, o excesso de liberdade pode ser prejudicial ao processo.
- h2: para um processo utilizando metodologias ágeis dar certo é necessário uma arquitetura de *deployment* e testes para melhoria da qualidade.
- h3: para um processo de manutenção, é necessário tratar as vertentes de manutenção separadamente.

7.4 Exploração do Material

Essa etapa se refere a aplicação sistemática das decisões tomadas. Consiste essencialmente em operações de codificação, decomposição ou enumeração, em função de regras previamente formuladas [45].

A partir da leitura flutuante, a seguinte codificação foi definida a *priori* 7.2(a):

Nome	Magnitude	Densidade
Arquitetura de deployment...~	34	0
Benefícios do processo~	71	0
Fatores que favorecem o pr...~	39	0
Fatores que prejudicam o pr...~	54	0
Peculiaridades do setor púb...~	27	0
Práticas úteis para o processo~	68	0
Problemas anteriores perce...~	37	0
Tentativa do Kanban no TJGO~	7	0
Testes importam~	22	0
Tratar as vertentes de manu...~	26	0

(a) Codificação inicial

Nome	Magnitude	Densidade
Arquitetura de deployment importa~	34	1
Fatores que favorecem o processo~	18	2
Fatores que prejudicam o processo~	54	3
O processo no NTSA~	52	1
O que se espera do processo~	19	3
Peculiaridades do setor público~	26	1
Práticas úteis para o processo~	92	5
Problemas anteriores percebidos~	37	2
Tentativa de uso do Kanban no TJGO~	7	0
Testes importam~	22	1
Tratar as vertentes de manutenção~	26	1

(b) Codificação final

Figura 7.2: Codificação (Fonte: autoria própria)

A codificação foi pensada no sentido de se responder as hipóteses. Buscou-se identificar conceitos que sustentassem a teoria. Percebeu-se que novas divisões e agrupamentos ajudariam nas comprovações, resultando na Figura 7.2(b).

De acordo com a literatura [45] é importante que se descreva cada código e portanto, segue a descrição:

- Arquitetura de *deployment* importa: pretende-se verificar trechos que confirmem a importância da arquitetura de *deployment* para o processo ágil;
- Fatores que favorecem o processo: atitudes, comportamentos humanos ou ambientais, leis e normas que favoreceram a implantação do processo;
- Fatores que prejudicam o processo: atitudes, comportamentos humanos ou ambientais, leis e normas que dificultaram a implantação do processo;
- O processo no NTSA: Indicadores que o processo implantado no Núcleo Técnico de Sistemas Administrativos tenha dado resultados;
- O que se espera do processo: o que se espera de um processo de software;

- Peculiaridades do setor público: informações que demonstram que o setor público possui pontos diferentes que precisam ser entendidos e trabalhados;
- Práticas úteis para o processo: sugestões de técnicas ou métodos que favorecem a implantação e evolução do processo;
- Problemas anteriores percebidos: problemas percebidos no contexto social do TJGO ao longo dos anos;
- Tentativa de uso do Kanban no TJGO: informações que demonstrem a tentativa e os motivos do fracasso em se tentar utilizar o Kanban;
- Testes importam: indicadores da importância dos testes para os métodos ágeis;
- Tratar as vertentes da manutenção: indicadores de que tratar as vertentes da manutenção, separadamente, é necessário.

Ao se rearranjar e buscar novos grupos e relacionamentos, deu-se início ao processo de categorização.

A etapa de categorização, segundo Bardin [45] : "é uma operação de classificação de elementos constitutivos de um conjunto por diferenciação e, em seguida, por reagrupamento segundo o gênero (analogia), com critério previamente definidos", ou seja, um conjunto de agrupamento de códigos a partir da comparação inicial.

Após a codificação, foram definidas as seguintes categorias de documentos conforme a Figura 7.3.

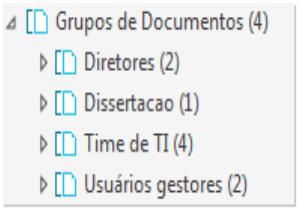


Figura 7.3: Grupos de documentos (Fonte: autoria própria)

Para melhor visualização dos códigos, buscou-se a categorização entre eles, conforme mostra a Figura 7.4, gerada a partir da ferramenta Atlas.ti (Fonte: [46]).

Além disso, verificou-se a cobertura das entrevistas em relação aos códigos. Seu resultado pode ser visto na Figura 7.4, que indica que houve uma boa cobertura das entrevistas sobre os resultados esperados por meio da categorização. Isso significa que as entrevistas foram bem conduzidas no sentido de se explorar o contexto social do TJGO após os impactos dos artefatos produzidos e aplicados nesse meio.

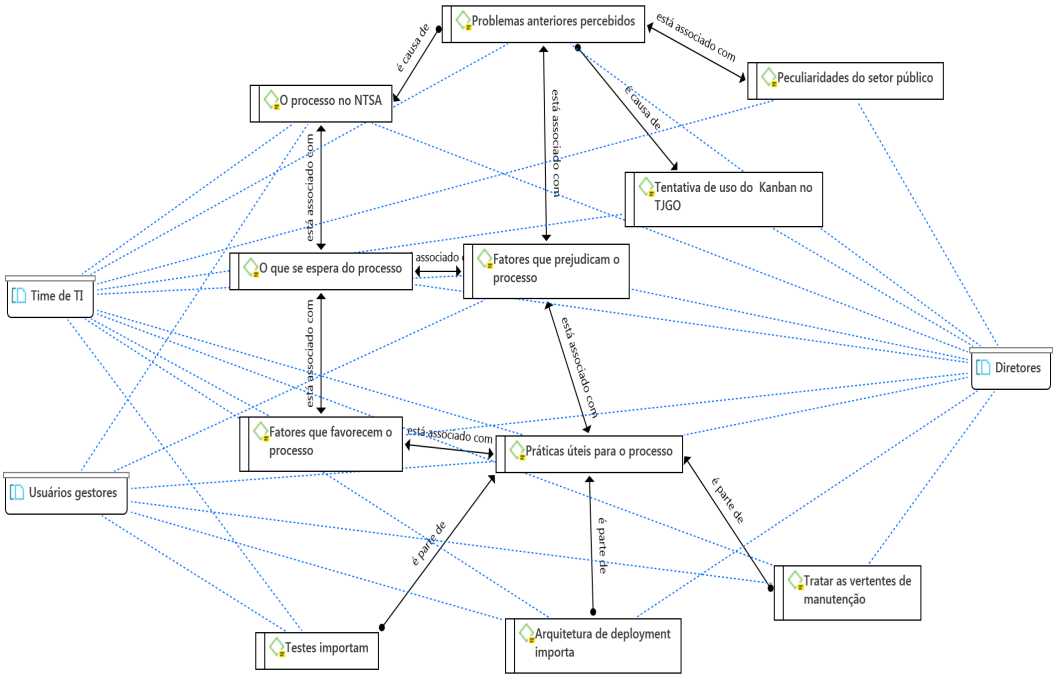


Figura 7.4: Cobertura das entrevistas em relação aos códigos (Fonte: autoria própria)

Após verificar a cobertura das entrevistas, procurou-se pontos de intersecções entre elas e a revisão da literatura. Iniciou-se pela análise dos indicadores de problemas, descritos na Seção 7.4.1. O mesmo foi feito para as *categorias de indicadores de soluções* na Seção 7.4.2 e *Indicadores do Processo Definido no NTSA* na Seção 7.4.3.

7.4.1 Categorias dos Indicadores de Problemas

Para indicadores de problemas agrupou-se as seguintes categorias: Problemas anteriores percebidos, Peculiaridades do setor público, Fatores que prejudicam o processo, Tentativa frustrada de uso do Kanban. A amostra do resultado pode ser visto na Figura 7.5 e o resultado completo no APÊNDICE E.

Na Figura 7.5 é possível ver amostras das falas dos entrevistados e também de citações presentes na dissertação.

Em *Problemas anteriores percebidos* (O conteúdo completo pode ser visto no APÊNDICE E) foi possível identificar:

- Individualismo, relatado por: A (p.8, 18:7, APÊNDICE I), B (p.1, 2:40, APÊNDICE I), C (p.3, 12:27, APÊNDICE I) e D (p.2, 2:45, APÊNDICE I);

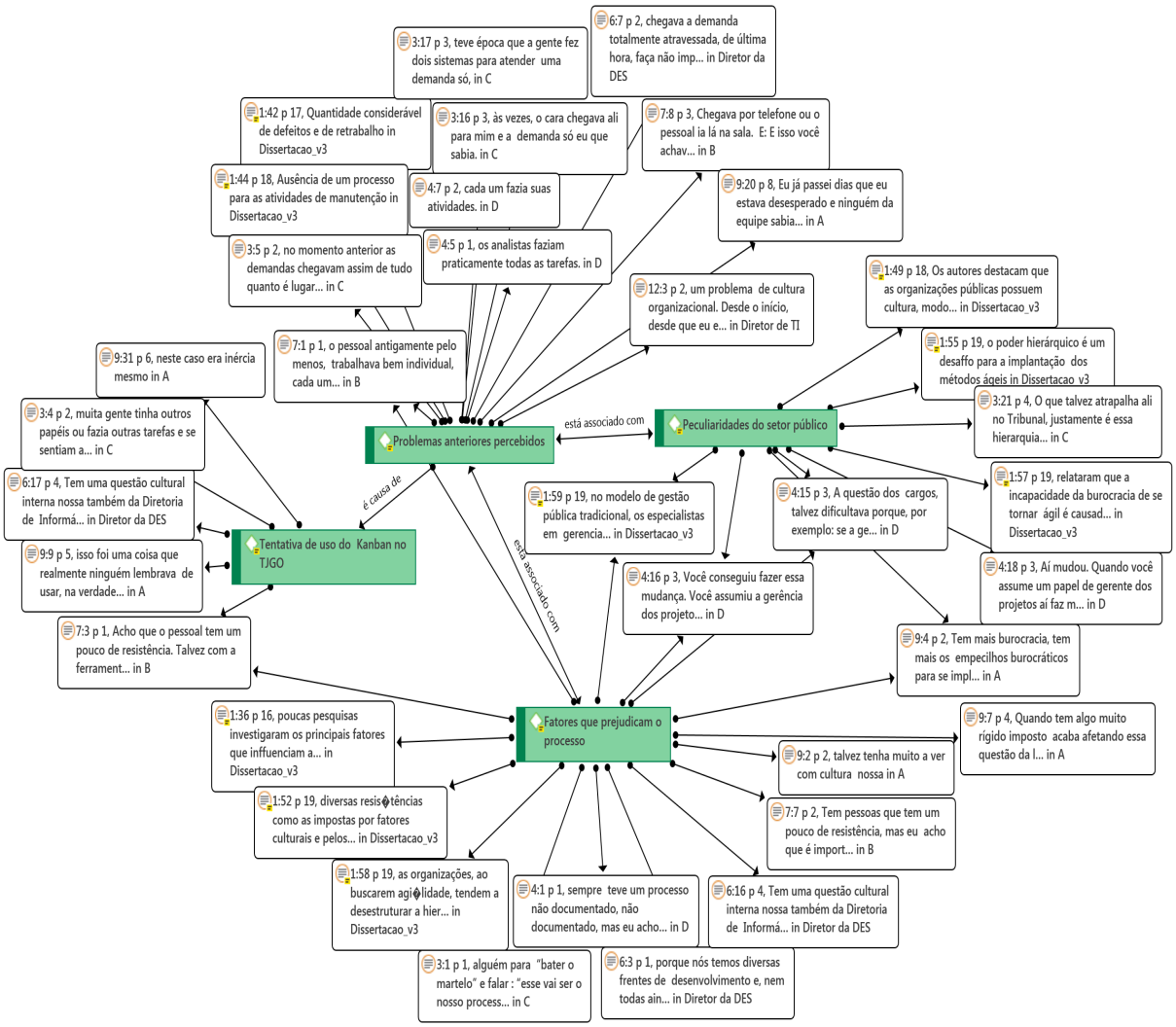


Figura 7.5: Amostra de indicadores de problemas no contexto social (Fonte: autoria própria)

- Demandas "atravessadas", relatado por: Diretor da DES (p.2, 6:37, APÊNDICE G), B (p.3, 5:29, APÊNDICE I) e C (p.2, 8:54, APÊNDICE I);
- Problema cultural, relatado por: Diretor de TI (p.2, 4:34, APÊNDICE G);
- Falta de organização, relatado por: C (p.3, 12:27, APÊNDICE I).

São problemas semelhantes aos observados pelo pesquisador no Contexto Social e também presentes no Contexto do Conhecimento conforme as citações: [35, 28, 1].

Na categoria referente às *Peculiaridades do setor público* (O conteúdo completo pode ser visto no APÊNDICE E):

- Dificuldades devido a hierarquia, relatado por: C (p.4, 13:57, APÊNDICE I);
- Dificuldade em se transformar estrutura funcional em projetizada, relatado por: D (p.3, 11:16, APÊNDICE I);
- Burocracia, relatado por: A (p.2,03:17, APÊNDICE I);
- Cargos e funções, relatado por: D (p.3,10:00, APÊNDICE I).

São problemas semelhantes aos conhecidos no *Contexto do Conhecimento* conforme as citações: [1, 28, 29, 30, 55, 60, 58].

Em *Fatores que prejudicam o processo* (O conteúdo completo pode ser visto no APÊNDICE):

- Cultura, relatado por: A (p.2, 03:09, APÊNDICE I), Diretor da DES (p.4, 12:36, APÊNDICE G) e Diretor de TI (p.2, 4:34, APÊNDICE G);
- Resistência, relatado por: A (p.4, 08:08, APÊNDICE I) e B (p.2, 05:29, APÊNDICE I);
- Burocracia, relatado por: C (p.4, 13:57, APÊNDICE I);
- Diferentes frentes de desenvolvimento (equipe de sistema legado de primeiro grau, equipe de sistema processual, equipe administrativa, fábrica de software e outros), relatado por: Diretor da DES (p.1, 2:50, APÊNDICE G);
- Sensação de estar bom como está, relatado por: D (p.1, 2:45, APÊNDICE I);
- Omissão da autoridade para decidir, relatado por: C (p.1, 3:20, APÊNDICE I);

São problemas semelhantes aos conhecidos no Contexto do Conhecimento conforme as citações: [28, 56, 57, 30].

Em *Tentativa de se usar o Kanban no TJGO* (O conteúdo completo pode ser visto no APÊNDICE E):

- Resistência, relatado por: A (p.6, 11:50, APÊNDICE I), B (p.2, 3:39, APÊNDICE I), C (p.2, 7:20, APÊNDICE I);
- Cultura, relatado por: Diretor da DES (p.4, 12:36, APÊNDICE G).

Diante da análise dessas amostras, buscou-se uma visão mais ampla sobre a densidade de cada categoria perante a dissertação e as entrevistas.

Na Figura 7.6, apresenta-se um diagrama Sankey [1], utilizado para mostrar as diferentes contribuições de cada categoria. Nele é possível observar os diretores opinando sobre as quatro categorias enquanto o time de TI se concentra mais nos problemas anteriores e fatores que prejudicam o processo. Já a dissertação cobre todos os pontos, exceto a Tentativa do Kanban no contexto do TJGO, dando maior ênfase nos problemas que prejudicam o processo visto que buscava-se entendimentos sobre eles.

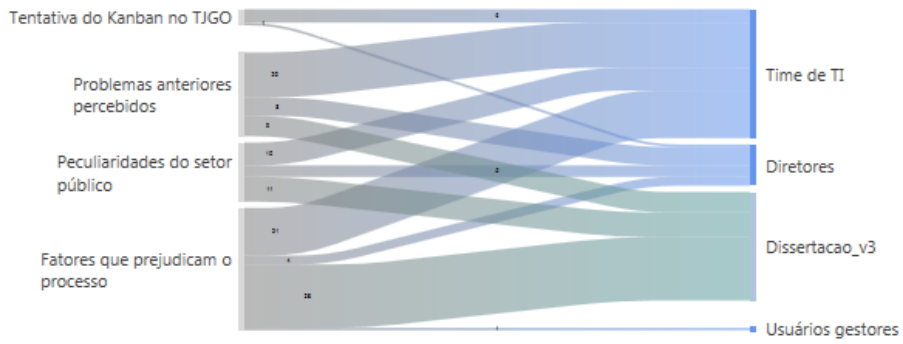


Figura 7.6: Problemas observados (Fonte: autoria própria)

7.4.2 Indicadores de solução

Diante dos problemas, buscou-se categorizar soluções e a Figura 7.7 apresenta uma amostra sobre citações bibliográficas e as entrevistas realizadas. O resultado completo pode ser visto no APÊNDICE E.

Das Práticas úteis para o processo tem-se:

- Importância da arquitetura de *deployment*, relatado por: A (p.7, 14:04, APÊNDICE I), C (p.7, 23:37, APÊNDICE I);
- Importância dos testes, relatado por: B (p.4, 9:15, APÊNDICE I) e Gestor2 (p.2, 3:19, APÊNDICE H);
- Tratar as vertentes da manutenção, relatado por: D (p.2, 7:47, APÊNDICE I), Gestor1 (p.3, 9:00, APÊNDICE H), Diretor de TI (p.3, 10:13, APÊNDICE G);
- Trabalho em equipe, relatado por: C (p.7, 23:37, APÊNDICE I);

[1] Desenvolvido em 1898 pelo Capitão Irlandês Matthew H. P. R. Sankey, visava mapear a eficiência de motor a vapor, atualmente é utilizado para apresentar fluxos de dados e suas associações com diversos elementos [123].

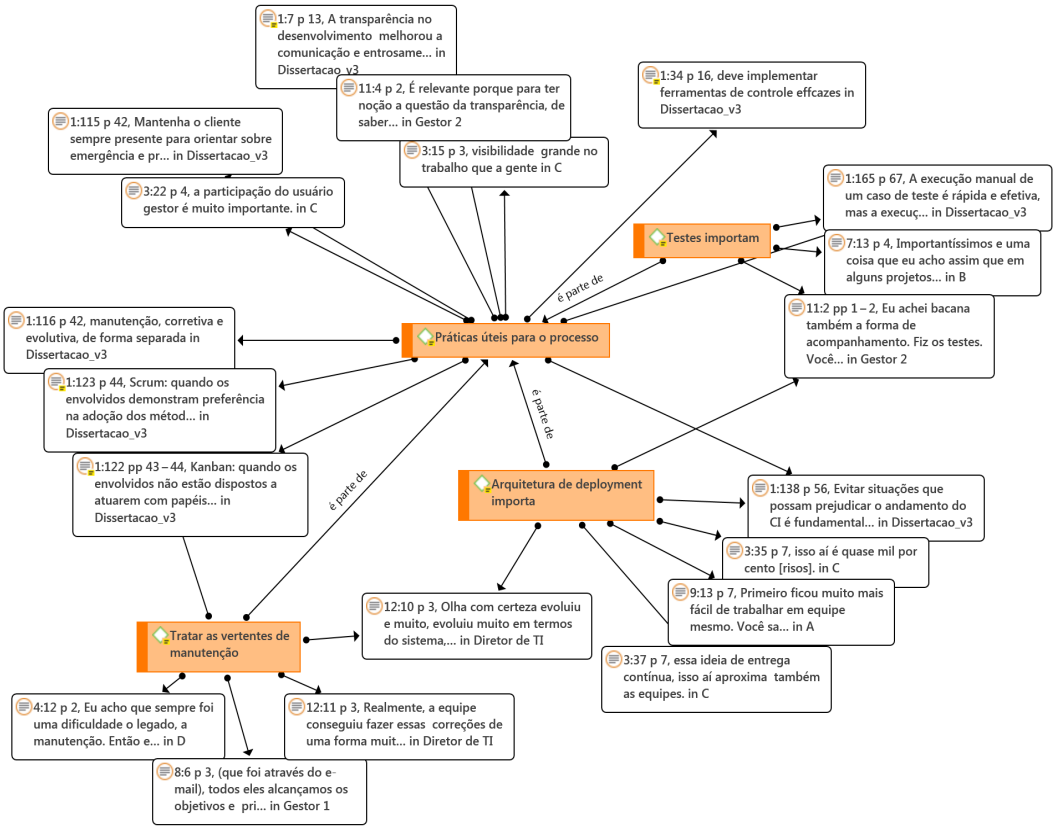


Figura 7.7: Amostra de soluções para o processo (Fonte: autoria própria)

- Visibilidade, relatado por: Gestor2 (p.2, 5:02, APÊNDICE H), C (p.3, 12:27, APÊNDICE I);
- Participação do usuário gestor, relatado por: C (p.4, 15:50, APÊNDICE I).

Essas práticas condizem com a revisão bibliográfica [3, 26, 32, 85, 121].

A Figura 7.8 mostra as categorias que favorecem o processo.

Na Figura 7.8 é possível verificar a equipe de TI opinado sobre cada categoria, enquanto os gestores abordam pontos do processo que mais lhe chamaram a atenção como: os testes, a manutenção evolutiva e a corretiva. Já os diretores viram maiores impactos devido a arquitetura e o processo de manutenção com suas vertentes. Também percebe-se uma grande densidade de prática úteis, fontes de diversos autores reunidas nesta dissertação.

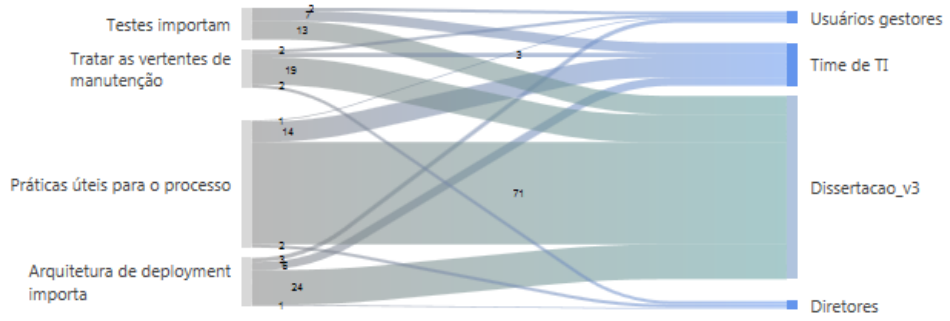


Figura 7.8: Problemas observados (Fonte: autoria própria)

7.4.3 Indicadores do Processo Definido no NTSA

Entendidos os problemas e conhecidas as boas práticas, buscou-se obter informações sobre o que se espera de um processo de software. Uma amostra dos resultados pode ser vista na Figura 7.9 e o resultado completo pode ser visto no APÊNDICE E.

Entre as respostas observadas tem-se: "Espera-se um planejamento" - (p.1, 2:50, Diretor da DES, APÊNDICE G) e "integração dos sistemas" - (p.1, 2:45, Diretor de TI, APÊNDICE G) e entre os fatores externos que favorecem a definição de um processo no TJGO estão: empenho do Poder Judiciário[37, 119] e do Governo Federal [50, 53].

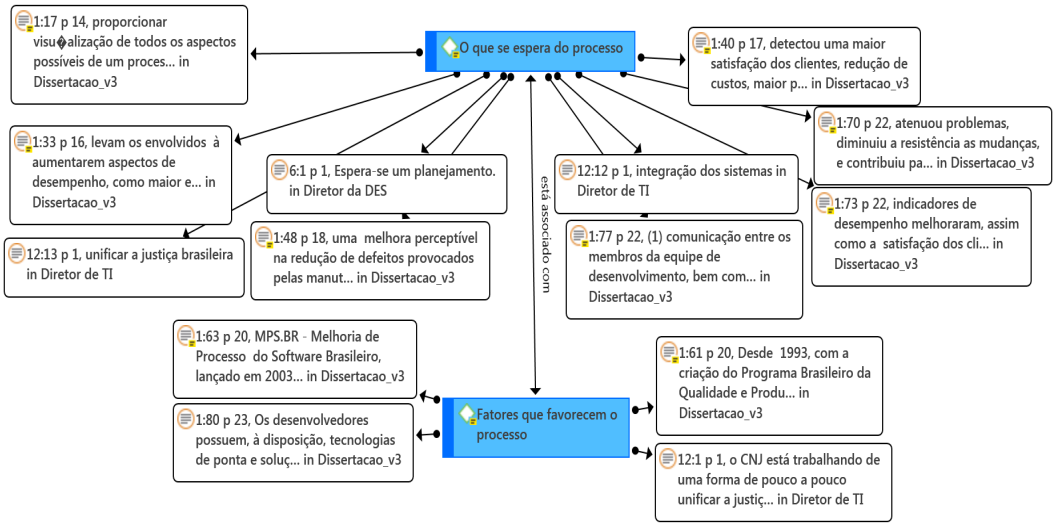


Figura 7.9: O que se espera de um processo de software e fatores que favorecem (Fonte: autoria própria)

Por fim, buscou-se evidências de que ocorreram mudanças no processo de manutenção de software no Núcleo de Sistemas Administrativos do TJGO nesses últimos dois anos

e a amostra de resultados pode ser vista na Figura 7.10.

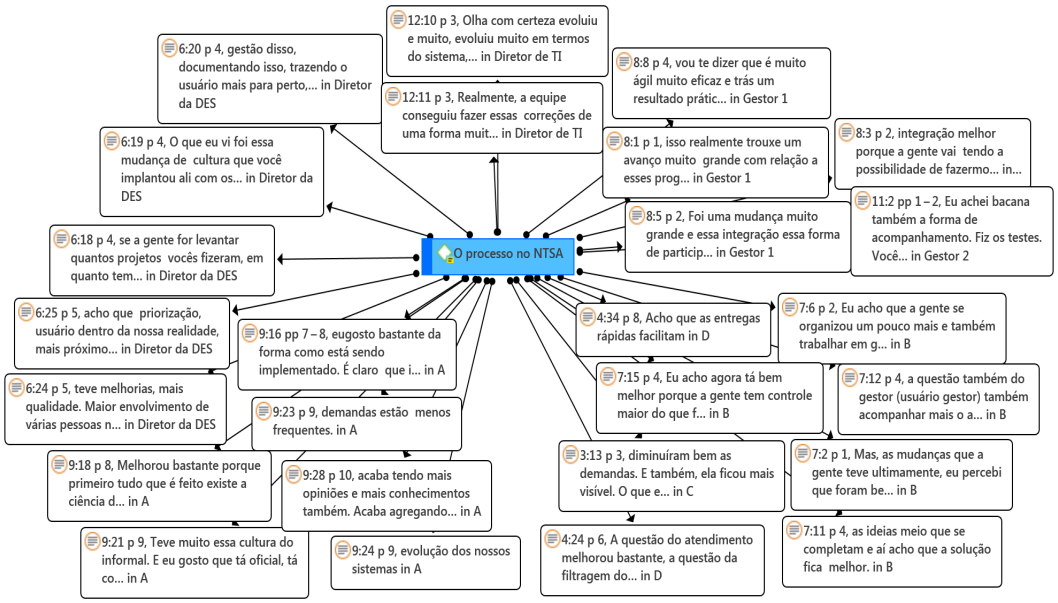


Figura 7.10: Amostra de resultados do processo (Fonte: autoria própria)

Cita-se como benefícios do processo conforme demonstra a Figura 7.10:

- Mudança na cultura: Diretor da DES (p.4, 15:33, APÊNDICE G) e A (p.9, 18:07, APÊNDICE I);
- Velocidade, conforme: Diretor de TI (p.3, 10:13, APÊNDICE G), Gestor1 (p.4,9:45, APÊNDICE H) e D (p.8,30:54, APÊNDICE I);
- Produtividade, conforme: Diretor da DES (p.4, 12:36, APÊNDICE G);
- Qualidade, conforme: Diretor da DES (p.5, 18:38, APÊNDICE G), Gestor2(p.1-2, 3:19, APÊNDICE H), B (p.4, 7:54, APÊNDICE I) e Gestor1(p.1,2:50, APÊNDICE H);
- Diminuição das demandas "atravessadas", conforme: A (p.9, 20:38, APÊNDICE I), C (p.3, 12:27, APÊNDICE I) e D (p.6, 23:22, APÊNDICE I);
- Organização, conforme: Diretor da DES (p.4, 15:33, APÊNDICE G), Diretor da DES (p.5, 18:38, APÊNDICE G) e B (p.4, 9:38, APÊNDICE I);
- Visibilidade, conforme: A (p.8, 18:07, APÊNDICE I), Gestor1 (p.2, 5:08, APÊNDICE H) e B (p.4,8:40, APÊNDICE H);

- Trabalho em equipe, conforme: A (p.10, 22:21, APÊNDICE I), B (p.2,5:29, APÊNDICE I) e Gestor1(p.2,5:08, APÊNDICE I);
- Melhora no clima organizacional: A (p.7-8, 14:04, APÊNDICE I) e B (p.1, 2:40, APÊNDICE I);
- Evolução, conforme: Diretor de TI (p.3,8:56, APÊNDICE G) e A (p.9,20:38, APÊNDICE I).

A Figura 7.11 retrata as perguntas direcionadas ao processo de manutenção e mostra que apenas os diretores opinaram sobre as expectativas do processo, o que era esperado pois essa pergunta foi direcionada apenas a eles, mas todos opinarão sobre as mudanças promovidas pelo processo de manutenção adotado.

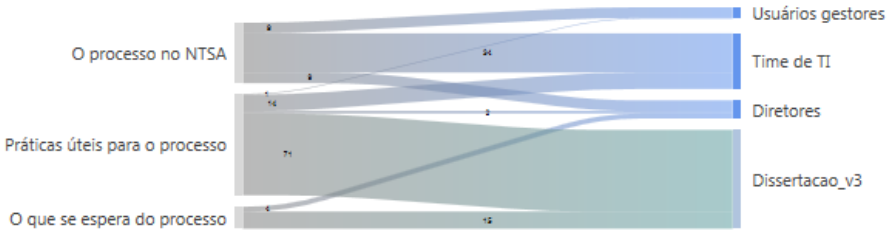


Figura 7.11: Processo de manutenção (Fonte: autoria própria)

7.5 Interpretação

Nesta seção serão feitas interpretações e inferências sobre os resultados para se concluir sobre as hipóteses.

Até o momento, as informações são suficientes para se fazer algumas inferências, mas os métodos quantitativos geram ainda mais confiança para confirmação das hipóteses. Diante disso, serão apresentados e discutidos alguns dados quantitativos.

Na Figura 7.12 é possível observar a quantidade de vezes que determinada categoria foi mencionada, as cores mais escuras indicam onde houveram maior concentração das respostas. Outra observação é que é possível ver pouca declaração dos gestores em relação aos problemas observados, enquanto os principais afetados são de fato o time de TI.

	1: Dissertacao_v3 168	Diretores 2 36	Time de TI 4 117	Usuários gestores 2 15	Totais
Fatores que prejudicam o processo 54	28	4	21	1	54
Peculiaridades do setor público 26	11	5	10		26
Problemas anteriores percebidos 37	9	8	20		37
Tentativa do Kanban no TJGO 7		1	6		7
Totais	48	18	57	1	124

Figura 7.12: Problemas observados (Fonte: autoria própria)

Dada a hipótese h1: *Devido às particularidades do setor público, o excesso de liberdade pode ser prejudicial ao processo.*

No contexto do TJGO, historicamente, houveram tentativas de utilização dos métodos ágeis, sendo o mais recente, a tentativa feita pelo pesquisador desta dissertação.

De acordo com alguns entrevistados, fatores como: cultura do TJGO mencionado pelo Diretor da DES (p.4, 15:33, APÊNDICE G), resistência a mudança relatado por B (p.2, 3:39, APÊNDICE I), hierarquia relatado por C (p.4, 13:57, APÊNDICE I), incompatibilidade do modelo funcional com o modelo projetizado exposto por D (p.3, 11:16, APÊNDICE I) contribuíram para o insucesso.

Organizações públicas possuem cultura [28], hierarquia [29], burocracia [29], sobreposições políticas [29], limitações à estrutura funcional [30] que podem tornar a implementação de métodos ágeis um desafio[55].

Ribeiro e Domingues [28] verificaram que Scrum e XP são os mais utilizados no setor público. Eles [28] escolheram utilizar o Scrum, com algumas adaptações, por sua capacidade de favorecer o controle e a gestão. Ainda assim, as adaptações foram necessárias, em grande parte devido a resistência à mudança que foi identificada como um dos principais problemas na implementação de metologias ágeis no setor público [1, 30, 57].

No NTSA, embora tenham havido resistências no início, o uso do Scrum com rigor foi importante para estabelecer os novos passos a serem seguidos:

- "As mudanças que a gente teve ultimamente, eu percebi que foram, bem [pensando], como vou dizer ? Bem... aceitas, **não no começo**, mas depois foi bem interessante". Transcrição de B (p.1, 2:40, APÊNDICE I);
- "[...]a resistência foi grande, acho que a resistência já diminuiu mas ela ainda existe, mas foi grande". Diretor da DES (p.4, 15:33, APÊNDICE G);

- “Na verdade eu acho que tem que ser assim, tem que chegar alguém e dizer que vai ser assim e pronto”. Transcrição de A (p.3,6:42, APÊNDICE I).

Pode-se ver na Figura: 7.12 uma proximidade na quantidade de “possíveis problemas” relatados na literatura com o relatos do time e diretores guardadas as devidas proporções em quantidade de diretores e time. No entanto, aparentemente, os usuários gestores não percebiam tantos problemas.

Desse modo, pode-se inferir, diante da revisão da literatura, das entrevistas e do contexto do TJGO que: *Devido às particularidades do setor público, o excesso de liberdade pode sim ser prejudicial a implantação de um processo de manutenção de software.*

Na literatura existe uma grande quantidade de práticas que favorecem o processo ao se trabalhar com métodos ágeis conforme mostra a Figura 7.13, coluna *Dissertação*. O time se mostra empenhado em contribuir com as práticas e descreve algumas que condizem com a literatura, mostrando que, no geral, após muita resistência, o time aceitou as ideias.

	1: Dissertacao_v3 168	Diretores 2 36	Time de TI 4 117	Usuários gestores 2 15	Totais
Arquitetura de deployment importa 34	24	1	6	3	34
Práticas úteis para o processo 88	71	2	14	1	88
Testes importam 22	13		7	2	22
Tratar as vertentes de manutenção 26	19	2	3	2	26
Totais	127	5	30	8	170

Figura 7.13: Praticas úteis observadas (Fonte: autoria própria)

Dada a hipótese h2: *para um processo utilizando metodologias ágeis dar certo é necessário uma arquitetura de deployment e testes para melhoria da qualidade.*

Percebe-se na Figura 7.13 que a arquitetura de *deployment* foi a mais comentada dada sua importância e benefícios percebidos.

Sem a definição de uma arquitetura, as entregas contínuas as quais a literatura sobre ágeis se referem [86, 124] seriam comprometidas visto que ela fornece o elo entre os objetivos e os resultados finais do sistema [61]. A (p.6,12:36, APENDICE I) relata a precariedade de como eram feitos os *deploys* na Divisão de Engenharia de Software do TJGO: “[...] era um processo inicialmente bem arcaico. Que em alguns pontos dava aquela falsa impressão de praticidade, mas era bem perigoso. Você desenvolvia sem passar por ninguém, já fazia seu *deploy* e estava em produção”.

Além disso as metodologias ágeis sugerem o trabalho em equipe, como por exemplo, os conceito de time do Scrum [81], ou de programação em pares do XP [124]. Nesse caso, a arquitetura de CI/CD [88] também contribui para esse objetivo, conforme observado por C (p.7, 23:37 ,APENDICE I): "Essa ideia de entrega continua, isso aí aproxima também as equipes".

Então infere-se que uma arquitetura de *deployment* apropriada é fundamental para trabalhar com um processo que se utiliza de métodos ágeis.

Quanto aos testes, pode-se dizer que ainda estão num processo bastante inicial no TJGO e assim como foi com o processo de manutenção com ágeis, exigirá mais tempo para que se torne uma realidade capaz de produzir melhores resultados. Entretanto, conforme sugerem outros autores [26], os testes foram introduzidos em todo ciclo de manutenção e de novo a arquitetura de *deployment* foi essencial para isso, visto que possibilitou a definição de ambientes propícios para execução deles. Como comprovou o Usuário Gestor2, ao final de cada *sprint*, ele foi convidado a testar as entregas em ambiente apropriado e relatou: "Eu achei bacana também a forma de acompanhamento. Fiz os testes [...]. Todos os testes que foram passados eu fiz." (p.2,3:19,APÊNDICE H).

O Gestor2 (p.2,4:37,APÊNDICE H), relata ainda que encontrou uma inconformidade com os requisitos e que foi possível identificá-la prematuramente por meio dos testes: "no início a gente tinha pensado de um jeito e acabou mudando a forma que a gente viu no teste e que precisava ser de outra forma".

Assim, fica constatada a importância dos testes para os métodos ágeis diante do requisitos ágeis de qualidade [63] e conforme o entrevistado B (p.4,9:15,APÊNDICE I): "Fica como lições aprendidas, é que às vezes a gente precisa caprichar mais um pouquinho no teste interno antes de passa para o usuário testar".

Considerando a hipótese h3: *para um processo de manutenção, é necessário tratar as vertentes de manutenção separadamente.*

Nos últimos dez anos, houveram algumas tentativas de se definir um processo de software no TJGO, uma delas foi baseada no RUP e fatores como: Documentação extensa, dificuldades em definir quais artefatos seriam necessários, falta de experiência, falta de treinamento conforme exposto por C (p.1, 3:20, APENDICE) e D (p.4, 15:20, APENDICE) contribuíram para o insucesso.

Outra tentativa foi o uso de ferramentas como Redmine e quadro Kanban e os fatores de ambos não terem dado certo são semelhantes e basicamente se resumem a: Fatores culturais causados por uma equipe técnica com baixa visão gerencial descrito pelo Diretor

da DES (p.4, 12:36, APÊNDICE G, resistência a mudança relatado por A (p.6, 11:50, APÊNDICE I), B (p.2, 3:39, APÊNDICE I), C (p.2, 7:20, APÊNDICE I).

Além dos problemas descritos, outra dificuldade ao se estabelecer um processo de manutenção é não tratar suas vertentes separadamente:

D (p.2-3,10:00, APÊNDICE I) exemplifica que o desenvolvimento do sistema judicial do TJGO é visto como sucessivas manutenções mas sem tratar suas vertentes: "[...] os de processo judicial - apesar de ter coisas novas - era meio que um incremento, uma nova funcionalidade, sempre uma nova funcionalidade [...] e não encarava assim também né ? Como: agora vou pegar uma alteração ou uma nova funcionalidade e tratá-la como um novo projeto".

Quando os modelos tratam a manutenção de forma geral sem levar em conta as subcategorias (corretiva, adaptativa, evolutiva) acabam não gerando conhecimento profundo das peculiaridades de cada subcategoria o que impacta diretamente na definição do escopo da manutenção. As categorias de manutenção diferem muito para serem agrupadas sob um mesmo modelo de processo [31].

Caso as características de cada manutenção não sejam tratadas separadamente, podem gerar ainda mais manutenções [25, 32, 3, 12].

Ao tratar as vertentes da manutenção utilizando-se canais apropriados institucionais (e-mail para manutenções corretivas e Sistema de Processo Administrativo para outras manutenções) em conjunto com métodos ágeis (Kanban para manutenções urgentes e Scrum para outras manutenções) os resultados foram satisfatórios conforme expõe o Diretor de TI (p.3,8:56, APÊNDICE G) exposto na Figura 7.14.

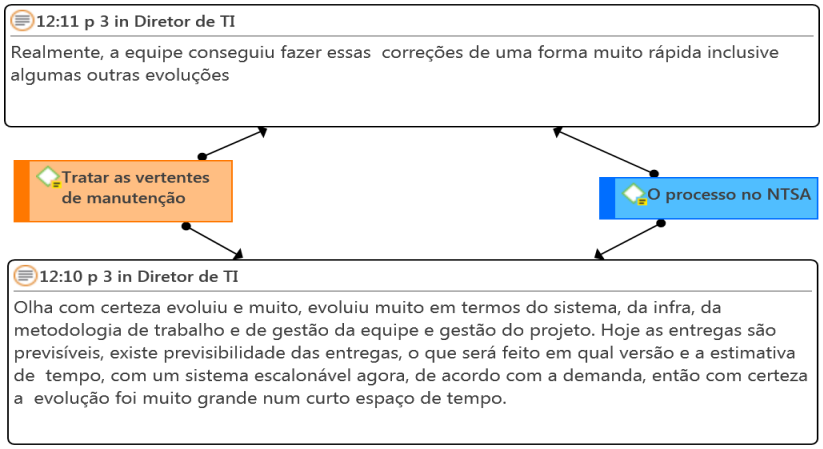


Figura 7.14: Relato do Diretor de TI obtido do cruzamento entre a importância de se *Tratar as vertentes da manutenção* e resultados do *Processo no NTSA* (Fonte: autoria própria)

Desse modo, fica comprovada a hipótese de que: *para um processo de manutenção, é necessário tratar as vertentes de manutenção separadamente.*

7.6 Considerações Finais do Capítulo

Neste capítulo, apresentou-se a Análise de Conteúdo [43], utilizando-se a técnica de categorização para realizar o cruzamento das informações obtidas nas entrevistas com os dados oriundos da revisão de literatura, como forma de reforçar metodologicamente os achados da pesquisa.

Capítulo 8

Conclusão

A adoção das práticas ágeis é crescente no mundo todo. Nos órgãos públicos brasileiros, a cobrança por soluções tecnológicas segue o ritmo acelerado, todavia fatores culturais e fatores humanos levam à necessidade de adaptação dos métodos ágeis à realidade de cada órgão.

As pesquisas apontam quatro pilares importantes para definir e implantar um processo de manutenção de sistemas baseado em metodologias ágeis: Pessoas, Processo, Arquitetura e Testes. Neste trabalho foram descritos os três últimos itens, empregados num órgão público do poder judiciário brasileiro (TJGO). A Arquitetura e os Testes foram empregados para a definição e avaliação da implantação de um processo de manutenção de software baseado em metodologias ágeis.

O TJGO não possuía um método de desenvolvimento bem definido, e contava com uma equipe que dedicava a maior parte dos esforços a ciclos infinitos de manutenções, respondendo às demandas por telefone.

Para melhorar as rotinas de trabalho, foram empregados métodos ágeis. Os resultados demonstraram que, na tentativa de se utilizar o Kanban, que é um método pouco prescritivo, não houve adesão, e os servidores mantiveram suas rotinas.

Em seguida, ao se empregar o Scrum, observou-se mais comprometimento dos participantes e, embora com resistência, a equipe aceitou o processo devido aos controles proporcionados pelo método e pelos bons resultados alcançados.

Entretanto, embora o emprego do Scrum tenha consolidado o processo para manutenções evolutivas, não se mostrou apropriado para manutenções urgentes. Desse modo, buscou-se um integração entre o Kanban para manutenções urgentes e o Scrum para manutenções evolutivas, resultando no processo Scrum de Rehman et al. [3] adaptado

para um Scrumban, confirmando assim a hipótese de que se faz necessário tratar as vertentes de manutenção separadamente em um processo de manutenção.

Observa-se que foi fundamental estabelecer os alicerces para se alcançar a agilidade e qualidade que os métodos ágeis requerem. Uma arquitetura de *deployment* possibilitou uma gestão segura do trabalho em equipe, entregas contínuas e íntegras, participação dos clientes utilizando-se ambientes apropriados, entre outros.

O emprego dos testes, mesmo em estágio inicial, mostrou-se relevante para a diminuição dos ciclos causados por manutenções mal feitas e para melhorias dos aspectos de qualidade. Além disso, uma nova cultura de testes se iniciou.

A experimentação de métodos ágeis utilizando a técnica DSR possibilitou que os ajustes fossem feitos para se adequar ao contexto social. Com isso, foi possível romper com a cultura antiga de desenvolvimento, resultando num método Scrumban, o qual tem gerado resultados satisfatórios, tanto para os diretores e clientes, quanto para equipe de TI, observadas na avaliação feita através da Análise de Conteúdo.

8.1 Trabalhos Futuros

Como trabalhos futuros, a oportunidade de se compreender a evolução do método Scrumbam adotado, avaliando os novos resultados estatisticamente para avaliar a diminuição de demandas corretivas urgentes. Avaliar, ainda, a necessidade de se dividir a manutenção evolutiva em mais vertentes. Estudar e implementar novo processo para a área de desenvolvimento de software do TJGO.

Referências

[1] Mantovani, F. R. e Marczak Sabrina: *Characteristics and challenges of agile software development adoption in brazilian government.* Journal of technology management & innovation, 15(2):3–10, 2020. vii, xiii, 3, 15, 16, 19, 61, 77, 95, 96, 102, 122

[2] Wieringa, Roel J.: *Design science methodology for information systems and software engineering.* Springer, 2014. xiii, 9, 10, 47, 48, 70, 71

[3] Rehman, Fateh, Maqbool Bilal, Riaz Muhammad, Qasim, Qamar Usman e Abbas Muhammad: *Scrum software maintenance model: Efficient software maintenance in agile methodology.* Em *2018 21st Saudi Computer Society National Computer Conference (NCC)*, páginas 1–5. IEEE, 2018. xiii, xiv, 3, 12, 20, 27, 28, 49, 50, 63, 69, 75, 76, 77, 78, 79, 80, 83, 85, 86, 87, 98, 105, 107, 125

[4] Alqudah, Mashal e Razali Rozilawati: *An empirical study of scrumban formation based on the selection of scrum and kanban practices.* International Journal on Advanced Science, Engineering and Information Technology, 8 (6), páginas 2315–2322, 2018. xiii, 29, 30, 72, 78, 79, 83

[5] Clements, Paul, Garlan David, Little Reed, Nord Robert e Stafford Judith: *Documenting software architectures: views and beyond.* 2003. xiii, 33, 35, 36, 68

[6] Pemberton, Andy: *Openshift ecosystem: Ultimate devops with cloudbees jenkins enterprise and openshift.* `https://www.openshift.com/blog/openshift-cloudbees-jenkins-enterprise-devops`, Accessed: 2021-05-05. xiii, 37

[7] Ramesh, Balasubramaniam, Cao Lan e Baskerville Richard: *Agile requirements engineering practices and challenges: an empirical study.* Information Systems Journal, 20(5):449–480, 2010. xiii, 44

[8] *Organograma de ti do tjgo.* `https://www.tjgo.jus.br/index.php/sge/institucional-sge/organograma-tjgo`, 2018. Accessed: 2020-06-01. xiii, xiv, 55, 56, 80

[9] *Ferramenta para controle e gerencia de repósitorio baseados no git.* `https://gitlab.com/`. xiv, 73

[10] Analytics, Clarivate: *Web of science*, 2020. `https://www-webofscience.ez54.periodicos.capes.gov.br/`. xiv, 123

[11] B.V., Elsevier: *Scopus*, 2020. `https://www-scopus.ez54.periodicos.capes.gov.br/`. xv, 124, 125

[12] Erazo, Martínez Jennifer, Gómez Andrés Florez e Pino Francisco J.: *Generando productos software mantenibles desde el proceso de desarrollo: El modelo de referencia mantus*. Ingeniare. Revista chilena de ingeniería, 24(3):420–434, 2016. xv, 2, 62, 67, 105, 125, 208, 209, 210

[13] April, Alain, Hayes J. H., Abran Alain e Dumke Reiner: *Software maintenance maturity model (smmm): the software maintenance process model*. Journal of Software Maintenance and Evolution: Research and Practice, 17(3):197–223, 2005. xv, 125, 211, 212, 213, 214

[14] Leal, Fábio Barros: *Uma abordagem usando features bdd e modelo de objetivos para o desenvolvimento ágil de software*. 2019. xv, 43, 217, 218, 219

[15] Sommerville, Ian: *Engenharia de software*. Pearson Brasil, 2011, ISBN 9788579361081. 1, 18, 58

[16] Kajko-Mattsson, Mira: *Corrective maintenance maturity model: Problem management*. Em *International Conference on Software Maintenance, 2002. Proceedings.*, páginas 486–490. IEEE, 2002. 2

[17] Grubb, Penny e Takang A.: *Software maintenance: concepts and practice*. World Scientific, 2003, ISBN 9812384263. 2, 3, 20

[18] Ambler, S. W.: *Modelagem Agil: Práticas eficazes para a programação eXtrema e o processo unificado*. Bookman, 2004, ISBN 9788536302980. 2, 42, 68

[19] Melo, Claudia de O, Santos Viviane, Katayama Eduardo, Corbucci Hugo, Prikladnicki Rafael, Goldman Alfredo e Kon Fabio: *The evolution of agile software development in brazil*. Journal of the Brazilian Computer Society, 19(4):523–552, 2013. 2, 14, 15, 125

[20] Gren, Lucas, Goldman Alfredo e Jacobsson Christian: *Agile ways of working: a team maturity perspective*. Journal of Software: Evolution and Process, 32(6):e2244, 2020. 2

[21] Polo, Macario, Piattini Mario, Ruiz Francisco e Calero Coral: *Mantema: A software maintenance methodology based on the iso/iec 12207 standard*. Em *Proceedings 4th IEEE International Software Engineering Standards Symposium and Forum (ISESS'99).'Best Software Practices for the Internet Age'*, páginas 76–81. IEEE, 1999. 2, 3, 12, 21, 22, 62, 122, 125

[22] Aquino Jr., G. S. de e Dantas A. M.: *An agile approach applied to intense maintenance projects uma abordagem ágil aplicada a projetos de manutenção intensa.* 2019. 2, 3, 12, 26, 31, 62, 72, 78, 79

[23] Kong, Xiaoying e Liu Li: *Critical feature method-a lightweight web maintenance methodology for smes.* J. Digit. Inf., 5, 2004. 3, 6, 22, 23, 24, 62, 75, 79, 83, 86, 87

[24] Tarwani, Sandhya e Chug Anuradha: *Agile methodologies in software maintenance: A systematic review.* Informatica, 40(4), 2016. 3

[25] Heeager, L. T. e Rose Jeremy: *Optimising agile development practices for the maintenance operation: nine heuristics.* Empirical Software Engineering, 20(6):1762–1784, 2015. 3, 4, 20, 26, 27, 75, 81, 87, 105, 125

[26] Poole, Charles e Huisman Jan Willem: *Using extreme programming in a maintenance environment.* IEEE Software, 18(6):42–50, 2001. 3, 62, 67, 69, 98, 104, 125

[27] Svensson, Harald e Höst Martin: *Introducing an agile process in a software maintenance and evolution organization.* páginas 256–264, 2005. 3, 69

[28] Ribeiro, Afonso e Domingues Luísa: *Acceptance of an agile methodology in the public sector.* Procedia computer science, 138:621–629, 2018. 4, 75, 77, 95, 96, 102

[29] Rulinawaty, Sofjan Aripin e Lukman Samboteng: *Leading agile organization can indonesian bureaucracy become agile?* Journal of Talent Development and Excellence, 12(3s):330–338, 2020. 4, 16, 75, 96, 102

[30] Mergel, Ines, Gong Yiwei e Bertot John: *Agile government: Systematic literature review and future research.* Government Information Quarterly, 35:291–298, 2018. 4, 16, 19, 20, 75, 96, 102

[31] Kajko-Mattsson, Mira: *Corrective maintenance maturity model: Problem management.* Univ., Department of Computer and Systems Sciences, 2001. 4, 105, 211

[32] Bouguerra, Abderaouf, Gölgeci İsmail, Gligor D. M e Tatoglu Ekrem: *How do agile organizations contribute to environmental collaboration? evidence from mnes in turkey.* Journal of International Management, página 100711, 2019. 5, 20, 62, 98, 105

[33] Melo, Claudia: *Productivity of agile teams: an empirical evaluation of factors and monitoring processes.* Tese de Doutoramento, Submetido ao Instituto de Matemática e Estatística da Universidade de São Paulo, 2015. 5

[34] Albuquerque, Regina: *Estudo sobre fatores que influenciam a manutenção de processos de software em empresas avaliadas por modelos de referência.* Tese de Mestrado, Pontifícia Universidade Católica do Paraná, PR, Brasil, 2014. 5

[35] Lucas, A. P. L.: *Gestão da manutenção de software: Um estudo de caso.* Tese de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, RJ, Brasil, 2016. 5, 95

[36] GOIÁS. Tribunal de Justiça do Estado de Goiás. *Plano Estratégico 2015/2020*. `https://www.tjgo.jus.br/index.php/sge/planejamento-estrategico/plano-estrategico`. 6, 49

[37] BRASIL. Conselho Nacional de Justiça. *Resolução 211 de 15 de dezembro de 2015*. `https://atos.cnj.jus.br/atos/detalhar/atos-normativos?documento=2227`. 6, 7, 49, 64, 99

[38] BRASIL. Conselho Nacional de Justiça. *Levantamento de Governança, Gestão e Infraestrutura de TIC do Poder Judiciário*, 2019. `https://www.cnj.jus.br/tecnologia-da-informacao-e-comunicacao/painel-do-igovtic-jud-do-poder-judiciario/`. 6, 7

[39] April, Alain: *Studying supply and demand of software maintenance and evolution services.* Em *2010 Seventh International Conference on the Quality of Information and Communications Technology*, páginas 352–357. IEEE, 2010. 7

[40] Gil, Antonio Carlos: *Como elaborar projetos de pesquisa*, volume 4. Atlas São Paulo, 2002. 8

[41] Marconi, M. A. Marina e Lakatos Eva M.: *Fundamentos de Metodologia Cientifica, 5 edição, Editora Atlas.* Editora Atlas, 2003. 8

[42] Vaishnavi, Vijay, Kuechler Bill e Petter Stacie: *Design science research in information systems.* January, 20:2004, 2004. 9

[43] Dresch, Aline, Lacerda Daniel Pacheco e Júnior J. A. V. Antunes: *Design science research: método de pesquisa para avanço da ciência e tecnologia.* Bookman Editora, 2015. 9, 11, 89, 106, 122, 125

[44] Wohlin, Claes, Runeson Per, Höst Martin, Ohlsson Magnus C, Regnell Björn e Wesslén Anders: *Experimentation in software engineering.* Springer Science & Business Media, 2012. 10, 11

[45] Bardin, Laurence: *Análise de Conteúdo.* Edições 70, 2016. 11, 52, 88, 90, 92, 93

[46] Charlottenburg, Amtsgericht: *Categorização das entrevistas realizadas no contexto do tjgo utilizando-se da ferramenta: Atlas.ti scientific software development gmbh.* `https://atlasti.com/`. 11, 52, 93

[47] Gibbs, Graham: *Análise de dados qualitativos: coleção pesquisa qualitativa.* Bookman Editora, 2009. 11, 52, 89

[48] Suárez, L. M., Montoya A. F. e Vélez J. I.: *Mantelasoft: una nueva guía para mipymes desarrolladoras de mantenimiento de software*. Entre Ciencia e Ingeniería, páginas 95–99, 2019. 12, 22, 24, 25, 62

[49] Mariano, Ari Melo e Rocha Maíra Santos: *Revisão da literatura: apresentação de uma abordagem integradora*. Em *AEDEM International Conference*, volume 18, 2017. 12, 121

[50] Cruz, Cláudio Silva da, Figuereido Rejane Maria da Costa e Andrade Edméia Leonor Pereira de: *Processo de contratação de serviços de tecnologia da informação para organizações públicas*. 2011. 15, 99

[51] *Ministério da ciência, tecnologia e inovaçãosecretaria de política de informáticaprograma brasileiro da qualidade e produtividade em software*. `https://www.softex.br/wp-content/uploads/2015/11/Programa-Brasileiro-de-Qualidade-e-Produtividade-em-Software.pdf`, 1993. Accessed: 2020-04-01. 15

[52] Santos, Gleison: *Influência e impacto do programa mps. br na pesquisa relacionada a qualidade de software no brasil*. SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 10:73–87, 2011. 15

[53] *Mpsbr*. `https://softex.br/mpsbr/`, 2003. Accessed: 2020-06-01. 15, 99

[54] Lima, Sheila Moutinho e Vendramel Wilson: *Mapeamento entre as práticas do scrum e os processos do nível g do mps. br*. FaSCi-Tech, 1(5), 2016. 15, 77

[55] Mauricio, Gustavo Carvalho e Neris Vânia Paula de Almeida: *Utilização de métodos ágeis por equipes de desenvolvimento de software em universidades públicas brasileiras*. Revista TIS, 4(3), 2016. 15, 59, 75, 96, 102

[56] Conceição, Fábio Augusto: *Implementação de metodologias ágeis no Brasil: a perspectiva cultural dos empregados de empresas de tecnologia*. Tese de Doutoramento, 2019. 16, 75, 96

[57] Noronha, A. P. V., Venson Elaine, Figueiredo R. M. C. e Modesto Augusto: *Applying kanban to manage outsourced maintenance services: An action research in a brazilian government agency*. Em *CIbSE*, páginas 665–678, 2017. 16, 17, 72, 75, 96, 102

[58] Vacari, Isaque e Rafael Prikladnicki: *Metodologias ágeis na administração pública: uma revisão sistemática da literatura*. Em *Embrapa Informática Agropecuária-Artigo em anais de congresso (ALICE)*. In: WORKSHOP BRASILEIRO DE MÉTODOS ÁGEIS, Florianópolis., 2014. 16, 17, 96

[59] Santos, J. V. P. dos, Noronha A. P. V. de, Figueiredo R. M. C. e Venson Elaine: *Using kanban in outsourced government projects of management maintenance demands: a descriptive research.* Em *13th International Conference on Information Systems and Technology Management*, 2016. 16

[60] Moraes, R. A.: *Proposta de melhoria para o processo de desenvolvimento de software do exército brasileiro com base no modelo de gestão de risco e na metodologia ágil.* Tese de Mestrado, Universidade de Brasília, Brasília, DF, Brasil, 2015. 17, 96

[61] Bass, Len, Clements Paul e Kazman Rick: *Software architecture in Pratice.* Addison-Wesley, 2012, ISBN 9780321815736. 18, 20, 34, 39, 40, 67, 103

[62] Nocêra, Rosaldo de Jesus: *Gerenciamento de Projetos - Teoria e Prática.* RJN, 2009, ISBN 9788590131892. 18, 86

[63] Beck, K., Beedle M., Bennekum A. van, Cockburn A., Cunningham W., Fowler M., Grenning J., Highsmith J., Hunt A., Jeffries R., Kern J., Marick B., Martin R. C., Mellor S., Schwaber K., Sutherland J. e Thomas D.: *Manifesto for agile software development*, 2001. `http://www.agilemanifesto.org/`, Accessed: 2020-04-01. 19, 20, 24, 33, 35, 44, 104

[64] Crispin, Lisa e Gregory Janet: *Agile testing: A practical guide for testers and agile teams.* Addison-Wesley Professional, 2009. 20, 216, 219

[65] Erazo, J. D., A. S. Florez e F. J. Pino: *Análisis y clasificación de atributos de mantenibilidad del software: una revisión comparativa desde el estado del arte.* Entre Ciencia e Ingeniería, 10(19):40–49, 2016. 20, 125

[66] Pressman, Roger S: *Engenharia de Software-7.* Amgh Editora, 2009. 20

[67] Singh, Raghu: *International standard iso/iec 12207 software life cycle processes.* Software Process Improvement and Practice, 2(1):35–50, 1996. 21, 22, 24

[68] *Ieee standard for software maintenance.* IEEE Std 1219-1998, páginas 1–56, 1998. 21, 22

[69] ISO/IEC: *International standard-iso/iec 14764 ieee std 14764-2006 software engineering; software life cycle processes &; maintenance.* 2006. 21

[70] Polo, Macario, Velthuis Mario Piattini e González Francisco Ruiz: *Mantool: a tool for supporting the software maintenance process.* Journal of Software Maintenance and Evolution: Research and Practice, 13(2):77–95, 2001. 22, 62

[71] Polo, Macario, Piattini Mario e Ruiz Francisco: *Using a qualitative research method for building a software maintenance methodology.* Software: Practice and Experience, 32(13):1239–1260, 2002. 22

[72] Pino, F. J., Ruiz Francisco, García Félix e Piattini Mario: *A software maintenance methodology for small organizations: Agile MANTEMA*. Journal Of Software Maintenance and Evolution: Research and Pratice, 2012. https://onlinelibrary-wiley.ez54.periodicos.capes.gov.br/doi/full/10.1002/smr.541, acesso em 2020-04-10. 22, 24, 25, 27, 62, 75, 125

[73] Ahmad, Muhammad Ovais, Kuvaja Pasi, Oivo Markku e Markkula Jouni: *Transition of software maintenance teams from scrum to kanban*. Em *2016 49th Hawaii International Conference on System Sciences (HICSS)*, páginas 5427–5436. IEEE, 2016. 25, 26, 72, 75

[74] Alaidaros, Hamzah, Omar Mazni e Romli Rohaida: *Towards an improved software project monitoring task model of agile kanban method*. Int. J Sup. Chain. Mgt Vol (IJSCM), 7(3):118–125, 2018. 26

[75] Ladas, Corey: *Scrumban-essays on kanban systems for lean software development*. Lulu. com, 2009. 26, 28

[76] Merson, Paulo: *Ultimate architecture enforcement: custom checks enforced at code-commit time*. páginas 153–160, 2013. 33, 67, 68

[77] Booch, Grandy, Rumbaugh James e Jacobson Ivar: *The unified modeling language user guide*. Reference manual, 1999. 34

[78] Garlan, David e Shaw Mary: *An introduction to software architecture*. 1994. 34

[79] Booch, Grady, Rumbaugh James e Jacobson Ivar: *UML: guia do usuário*. Elsevier Brasil, 2006. 34

[80] Silva, Dyego Alves da, de Oliveira Edgard Costa, Canedo Edna Dias e Martins Hugo Ferreira: *Application of a hybrid process software requirements management*. Em *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*, páginas 1–7. IEEE, 2016. 34

[81] Schwaber, Ken: *Scrum.org, the home of scrum*, 2021. https://www.scrum.org/, Accessed: 2021-05-20. 35, 104

[82] Steffens, Andreas, Lichter Horst e Döring Jan Simon: *Designing a next-generation continuous software delivery system: Concepts and architecture*. Em *2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE)*, páginas 1–7. IEEE, 2018. 35, 36, 37, 38, 39, 67

[83] Ruka, Adam: *My primer on docker*. https://www.endoflineblog.com/my-primer-on-docker, Accessed: 2021-05-06. 37, 41

[84] Yoder, Joseph W., Aguiar Ademar, Merson Paulo e Waseda Hironori Washizaki: *Deployment patterns for confidence*. Workshop at AsianPLoP 2019 Tokyo. Accessed: 2021-05-05. 37, 40, 67

[85] Hemon, Aymeric, Lyonnet Barbara, Rowe Frantz e Fitzgerald Brian: *From agile to devops: Smart skills and collaborations.* Information Systems Frontiers, 22(4):927–945, 2020. 37, 98

[86] Vassallo, Carmine, Zampetti Fiorella, Romano Daniele, Beller Moritz, Panichella Annibale, Di Penta Massimiliano e Zaidman Andy: *Continuous delivery practices in a large financial organization.* páginas 519–528, 2016. 38, 71, 103, 125

[87] Laukkanen, Eero, Itkonen Juha e Lassenius Casper: *Problems, causes and solutions when adopting continuous delivery—a systematic literature review.* Information and Software Technology, 82:55–79, 2017. 38, 39, 42, 68

[88] *Integração e entrega contínuas: pipeline ci/cd.* `https://www.redhat.com/pt-br/topics/devops/what-is-ci-cd`, 2021. Accessed: 2021-04-28. 38, 39, 68, 69, 104

[89] Mårtensson, Torvald, Ståhl Daniel e Bosch Jan: *Enable more frequent integration of software in industry projects.* Journal of Systems and Software, 142:223–236, 2018. 38

[90] Häkli, Aleksi, Taibi Davide e Systa Kari: *Towards cloud native continuous delivery: An industrial experience report.* páginas 314–320, 2018. 39, 67

[91] Mu, Lifeng, Sugumaran Vijayan e Wang Fangyuan: *A hybrid genetic algorithm for software architecture re-modularization.* Information Systems Frontiers, 22(5):1133–1161, 2020. 40

[92] Merson, Paulo: *Defining microservices.* `https://insights.sei.cmu.edu/blog/defining-microservices/`, Accessed: 2021-05-05. 40, 41

[93] Melvin, Conway: *Conway's law.* `https://www.melconway.com/Home/Conways_Law.html`, Accessed: 2021-05-05. 41

[94] Evans, Eric e Evans Eric J: *Domain-driven design: tackling complexity in the heart of software.* Addison-Wesley Professional, 2004. 41

[95] Lewis, James e Fowler Martin: *Microservices guide.* `https://www.martinfowler.com/microservices/`, Accessed: 2021-05-05. 41

[96] Foundation, Linux: *Kubernates.* `https://kubernetes.io/pt-br/`, Accessed: 2021-05-06. 41

[97] Tian, Yuchi, Pei Kexin, Jana Suman e Ray Baishakhi: *Deeptest: Automated testing of deep-neural-network-driven autonomous cars.* Em *Proceedings of the 40th international conference on software engineering*, páginas 303–314, 2018. 42

[98] Kasauli, Rashidah, Knauss Eric, Horkoff Jennifer, Liebel Grischa e de Oliveira Neto Francisco Gomes: *Requirements engineering challenges and practices in large-scale agile system development.* Journal of Systems and Software, 172:110851, 2021. 43

[99] Noel, Rene, Riquelme Fabián, Mac Lean Roberto, Merino Erick, Cechinel Cristian, Barcelos Thiago S, Villarroel Rodolfo e Munoz Roberto: *Exploring collaborative writing of user stories with multimodal learning analytics: A case study on a software engineering course*. IEEE Access, 6:67783–67798, 2018. 43

[100] Medeiros, Juliana, Vasconcelos Alexandre, Silva Carla e Goulão Miguel: *Requirements specification for developers in agile projects: Evaluation by two industrial case studies*. Information and Software Technology, 117:106194, 2020. 43

[101] Fraga, Bárbara e Barbosa Marcelo: *A engenharia de requisitos nos métodos ágeis: uma revisão sistemática da literatura*. Em *Anais do XIII Simpósio Brasileiro de Sistemas de Informação*, páginas 309–315. SBC, 2017. 43

[102] Hynninen, Timo, Kasurinen Jussi, Knutas Antti e Taipale Ossi: *Software testing: Survey of the industry practices*. páginas 1449–1454, 2018. 43, 217, 219

[103] Mendes, Thiago Souto, Farias Mário André de F., Mendonça Manoel, Soares Henrique Frota, Kalinowski Marcos e Spínola Rodrigo Oliveira: *Impacts of agile requirements documentation debt on software projects: a retrospective study*. páginas 1290–1295, 2016. 43

[104] Borrego, Gilberto, Morán Alberto L., Palacio Ramón R., Vizcaíno Aurora e García Félix O.: *Towards a reduction in architectural knowledge vaporization during agile global software development*. Information and Software Technology, 112:68–82, 2019. 44

[105] Duarte, Luiz Fernando: *Agile testing: dicas para testes de software em times ágeis*. 2018. `https://imasters.com.br/agile/agile-testing-dicas-para-testes-de-software-em-times-ageis`. 44, 45

[106] Salerno, Larissa, Signoretti Ingrid, Marczak Sabrina e Bastos Ricardo: *Repensando papéis em equipes ágeis: Um estudo de caso no uso de uma abordagem combinada de desenvolvimento ágil, user-centered design e lean startup*. páginas 72–77, 2019. 44, 45

[107] Moe, Myint Myint: *Comparative study of test-driven development (tdd), behavior-driven development (bdd) and acceptance test–driven development (atdd)*. International Journal of Trend in Scientific Research and Development, páginas 231–234, 2019. 45, 216, 217

[108] BRASIL. Governo do Estado de Goiás. *Lei nº 20.756, de 28 de janeiro de 2020*. `https://legisla.casacivil.go.gov.br/pesquisa_legislacao/100979/lei-20756`. 49, 54

[109] Zoom Video Communications, Inc.: *Ferramenta de videoconferência: Zoom*. `https://explore.zoom.us/pt/products/meetings/`. 51

[110] BRASIL. *Decreto Judiciário 2162/2018, Dispõe sobre a consolidação da estrutura administrativa e judicial das comarcas do Poder Judiciário do Estado de Goiás*, date = 2021, series = Tribunal de Justiça do Estado de Goiás, url = http://tjdocs.tjgo.jus.br/documentos/505411. 55

[111] Goiás, E: *Lei no 17.663, de 14 de junho de 2012*, 2012. Accessed: 2020-04-01. 57

[112] Santos, Leticia Cavassin Ferreira dos: *Estilos de liderança*. 2013. 57, 72

[113] Aquino, Jussara Maria Canuto de: *Identificação e Imagem do Servidro Público: Um estudo com os usuários do Tribunal de Justiça do Estado de Minas Gerais*. Tese de Doutoramento, Mestrado em Administração, 2010. 58, 59

[114] Covey, Stephen R: *Os sete hábitos das pessoas altamente eficazes*. Rio de Janeiro: Editora Best Seller, 2000. 59

[115] Branco, Rosangela Antunes e Ribeiro Paulo Eduardo: *A meritocracia e os cargos comissionados no serviço público*. Rev. Caribeña de Ciencias Sociales, 1(1):1–14, 2016. 59

[116] Stanislaw, Clóvis: *Trabalho no serviço público: padrão acomodação?* 2014. 59

[117] Carmo, Eduardo do, Medeiros Cintia Rodrigues de O e Loebel Eduardo: *Novatos motivados e veteranos acomodados: representações sociais de técnicos-administrativos sobre a atuação no trabalho*. Revista de Carreiras e Pessoas (ReCaPe)| ISSN-e: 2237-1427, 5(3), 2015. 59

[118] Foote, Brian e Yoder Joseph: *Big ball of mud*. 1997. `https://www.laputan.org/mud`, Accessed: 2020-09-05. 60

[119] BRASIL. Tribunal de Justiça do Estado de Goiás. *Resolução 119 de 27 de novembro de 2019*. `http://tjdocs.tjgo.jus.br/documentos/537215`. 64, 99

[120] ALVES, Gabriella M. T.: *Automatização de testes em equipes ágeis: um estudo qualitativo usando teoria fundamentada.* Tese de Mestrado, 2017. 67, 220, 221

[121] Rao, Nagesh: *Scrumban Software Maintenance: 5 Steps to Stop Starting and Start Finishing (English Edition)*. Createspace Independent Publishing Platform, 2017, ISBN 1546519319. 72, 79, 98

[122] Beck, Kent e Andres Cynthia: *Extreme programming explained: Embrace Change 2nd Edição*. addison-wesley professional, 2004. 87

[123] Matters, Why It: *Atlas. ti 9 user manual-windows*. `https://doc.atlasti.com/ManualWin.v9`, acesso em 2021-11-04. 97

[124] Beck, Kent: *Test-driven development: by example*. Addison-Wesley Professional, 2003. 103, 104, 216

[125] Iso, ISO: *Iec25010: 2011 systems and software engineering–systems and software quality requirements and evaluation (square)–system and software quality models*. International Organization for Standardization, 34:2910, 2011. 208

[126] Ellwanger, Polyana De Carli e Guarienti Priscila: *Estudo e aplicação de ferramenta para automatização de testes em software as a service*. Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação, 1(7), 2017. 216

[127] Beller, Moritz, Gousios Georgios, Panichella Annibale, Proksch Sebastian, Amann Sven e Zaidman Andy: *Developer testing in the ide: Patterns, beliefs, and behavior*. IEEE Transactions on Software Engineering, 45(3):261–284, 2017. 216, 220

[128] Borle, Neil C, Feghhi Meysam, Stroulia Eleni, Greiner Russell e Hindle Abram: *Analyzing the effects of test driven development in github*. Empirical Software Engineering, 23(4):1931–1958, 2018. 216

[129] Bernardo, Paulo Cheque e Kon Fabio: *A importância dos testes automatizados*. Engenharia de Software Magazine, 1(3):54–57, 2008. 219, 220

[130] Ali, Sadia, Hafeez Yaser, Hussain Shariq e Yang Shunkun: *Enhanced regression testing technique for agile software development and continuous integration strategies*. Software Quality Journal, páginas 1–27, 2019. 220

[131] Piovesan, Ana Claudia: *Framework para testes ágeis de software: uma proposta exploratória*. Tese de Mestrado, Universidade Tecnológica Federal do Paraná, 2018. 221

[132] Gambi, Alessio, Bell Jonathan e Zeller Andreas: *Practical test dependency detection*. Em *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, páginas 1–11. IEEE, 2018. 221

Apêndice A

Teoria do Enfoque Meta Analítico (TEMAC)

Para condução da pesquisa, foram realizadas revisões da literatura com base na Teoria do Enfoque Meta Analítico (TEMAC), proposto por Mariano e Rocha [49]. Esse método se divide em três etapas:

- Preparação da pesquisa;
- Apresentação e inter-relação dos dados;
- Detalhamento, modelo integrador e validação por evidências.

A etapa de **Preparação da pesquisa** foi estabelecida buscando-se respostas à pergunta principal da pesquisa: *como definir um processo de manutenção de software para o TJGO, empregando a metodologia ágil num cenário de órgão público brasileiro, com equipe interna de desenvolvimento ?* O TEMAC estabelece que na etapa de *Preparação da pesquisa* algumas perguntas sejam respondidas:

- Qual a palavra-chave da pesquisa ?
- Qual o campo espaço-tempo da pesquisa ?
- Quais bases de dados serão utilizadas ?
- Quais áreas do conhecimento serão utilizadas ?

Para responder as perguntas, estabeleceu-se alguns termos de busca, contexto e tempo, bases de dados, e as áreas de conhecimento:

- Definidas como palavras-chave: *Software maintenance*, *Agile*, empregando os operadores lógicos como “or” e “and” para refinamento da pesquisa;
- O campo espaço-tempo da pesquisa realizou-se principalmente nos últimos cinco anos mas, para contextualização e embasamento teórico, buscou-se também artigos dos últimos vinte anos.
- Foram utilizadas as bases de dados *Web of Sciense*, *Scopus* e *Google Scholar*;
- As áreas de conhecimento principais foram: Computação e Engenharia de Software.

Na etapa **Apresentação e inter-relação dos dados** foi realizado um levantamento das revistas (journals) mais relevantes; as revistas que mais publicam sobre o tema; evolução do tema ano a ano; documentos mais citados; autores que mais publicaram; países que

mais publicaram; universidades que mais publicaram; frequência de palavras-chave. O levantamento foi realizada nas plataformas *Web of Science* e *Scopus*.

Diferente de outros métodos de pesquisa, no método DSR adotado, o objetivo da revisão sistemática da literatura, é:

> [...] formar um arcabouço teórico-prático dos artefatos utilizados para solução de um determinado problema ou classe de problemas. Por prático, nesse caso, entende-se que os artefatos devem ter sidos testados em campo. Também é interesse do pesquisados identificar as heurísticas de construção ou contingenciais presentes em cada um dos estudos primários pesquisados [43].

Assim, buscou-se realizar uma análise dos artigos publicados nos últimos 20 anos, com maior foco nos últimos 5 anos e naqueles que se propuseram a solucionar problemas, na prática, semelhantes aos pesquisado.

O artigo mais antigo analisado foi o de Polo et al.[21] de 1999 e o mais recente foi o de Mantovani e Marczak [1] de 2020.

Além de Polo et al. [21], foi possível observar outras contribuições relevantes na Figura A.1, como os trabalhos precursores das metodologias ágeis de Kent Beck, nos anos 2000, dando início ao movimento ágil com *Extreme Programming*. Boehm e Turner (2002) com trabalhos que sugerem o equilíbrio nas escolhas das metodologias, entre outros. A Figura A.1 foi gerada pela pesquisa realizada na Web of Sience com os termos: *software maintenance and agile.*

Outros termos foram gerados e combinados com propósito de se entender os caminhos possíveis da pesquisa e podem ser vistos na Figura A.2, com suas respectivas quantidades de citações. As palavras-chave possibilitaram observar atributos que estão diretamente relacionados com a manutenção ágil.

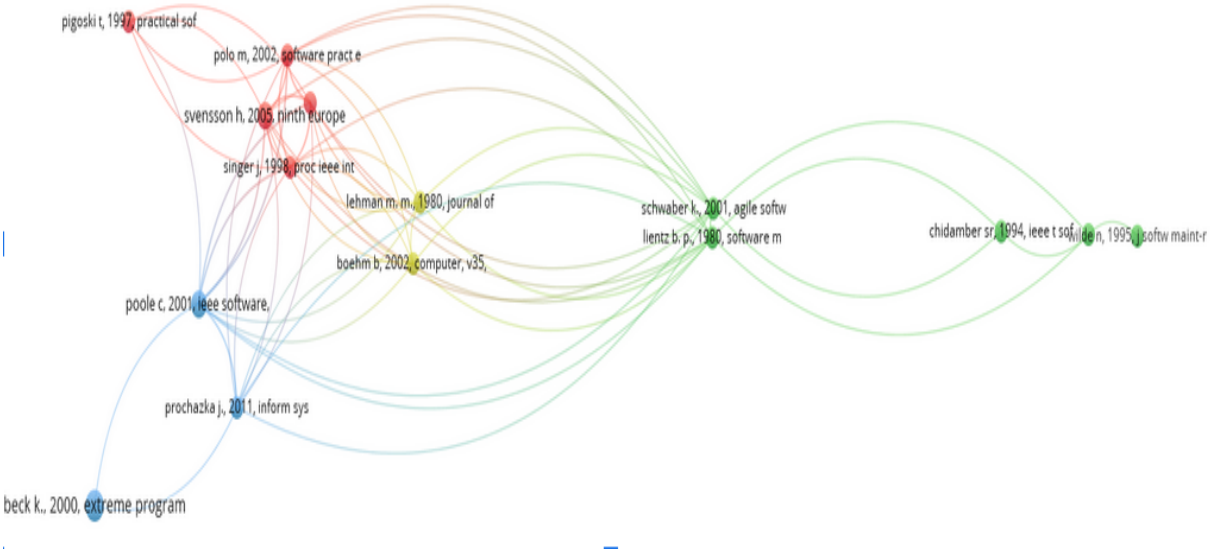


Figura A.1: Gráfico de co-citações de 2000 até 2020 (Fonte: [10])

.

Figura A.2: Conjunto de termos chave (Fonte: [10])

.

Foi observado que o Brasil é um país empenhado nas metodologias ágeis e vem participando com destaque de pesquisas neste setor como demonstra a Figura A.3.

A revisão da literatura foi feita de forma qualitativa com a intenção de sintetizar os achados de estudos e para isso utilizou-se de uma ficha com os artigos mais relevantes (citações, co-citações, autores, palavra-chave) para manutenção ágil iniciando de forma mais abrangente (últimos vinte anos, 19 artigos Web of Science e 139 no Scopus) até se chegar numa linha de refinamentos recentes (últimos cinco anos, 7 artigos no Web of Science e 49 no Scopus).

Além dessas bases, também se realizou pesquisas no *Google Scholar*, incluindo pesquisas em português, porém trata-se de uma base com muitas inconsistências para se

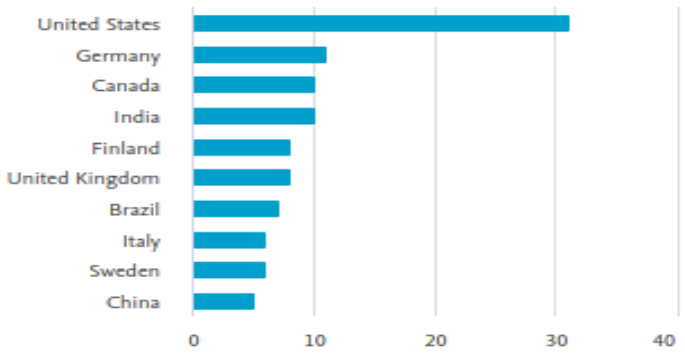


Figura A.3: Países que mais publicaram nos últimos 20 anos (Fonte: [11])

.

realizar uma análise mais profunda como as de *co-citações* e *coupling*.

Observa-se na Figura A.4 que somente a partir de 2009 houveram aumentos significativos na quantidade de publicações, o que demonstra oportunidades de novas pesquisas no meio.

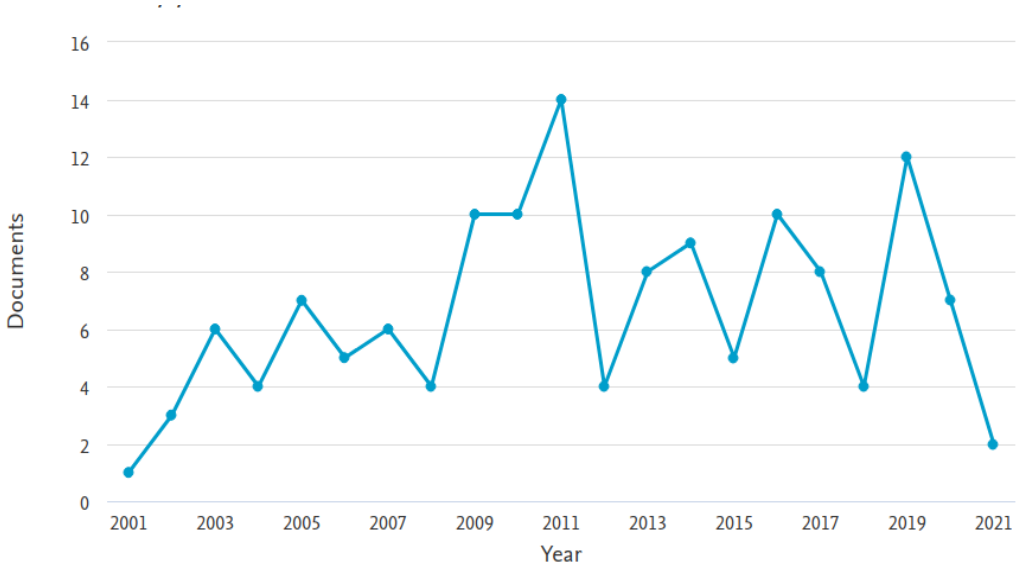


Figura A.4: Evolução da pesquisa ano a ano (Fonte: [11])

.

Na última etapa do TEMAC - **Detalhamento, modelo integrador e validação por evidências** - utilizou-se o software *Vosviewer* (disponível em `http://www.vosviewer.com`), com o propósito de se investigar relacionamentos entre os artigos pesquisados.

Foram realizadas análises de *co-citation*, Figura A.1, dos últimos vinte anos para se ter um embasamento teórico necessário para sustentar a pesquisa e *coupling*, Figura A.5 dos últimos cinco anos para se observar as tendências mais recentes das pesquisas e então identificar novas contribuições.

Figura A.5: Mapa de calor coupling de citações nos úlitmos cinco anos (Fonte: [11])

.

Após as análises foram reunidos documentos que possibilitaram traçar uma estratégia de pesquisa e de leitura de alguns dos principais artigos. Tornou-se possível mapear autores que contribuíram ou ainda contribuem para manutenção, nos últimos 20 anos, utilizando-se de métodos ágeis como por exemplo: Polo et al. [21, 72], Erazo et al. [12, 65], Poole e Huisman [26], Melo et al. [19], April [13], Heeager e Rose [25], Rehman et al.[3], Vassalo et al.[86] e outros.

A pesquisa foi ajustada em ciclos e maior ênfase foi dada aos artigos dos últimos 5 anos, a cada ciclo, novas descobertas, desafios, direcionamentos, ajustes e experimentações foram feitos utilizando-se a DSR. Por exemplo: inicialmente tinha-se a hipótese de se utilizar apenas um método ágil para manutenção, mas muitos estudos teóricos-práticos indicaram que o tratamento individual para, manutenção evolutiva e corretiva, pode levar a uma melhor solução.

De acordo com Dresh [43], a revisão sistemática da literatura utilizando-se da DSR visa formar um arcabouço teórico-prático de pesquisas que não só foram fundamentadas em teorias mas que também utilizaram artefatos para solução de problemas semelhantes no campo.

Desse modo, o critério de seleção adotado para escolha dos artigos, foi o de se escolher aqueles que resultaram em artefatos aplicados em casos reais.

Apêndice B

Relatório de Acompanhamento Projetos de Manutenção Evolutiva

Proad (280734) v 2.6.7

Situação : CONCLUÍDO

Data esperada de início	Data esperada de término	Tamanho da sprint em dias úteis
08/07/2021	23/08/2021	7

Data real de início	Data real de término	Tamanho da sprint em dias úteis
08/07/2021	30/08/2021	7

Relatório de manutenção no Proad: sigilo de processo

Michel Alves Ribeiro
Coordenador do Núcleo Técnico de Sistemas Administrativos (NTSA)

1. Descrição Geral

“Adicionar uma breve descrição sobre o escopo do projeto”

2. Objetivo

“Descrever os objetivos do projeto”

3. Prazo Estimado

Data esperada de início	Data esperada de término	Tamanho da sprint em dias úteis
08/07/2021	23/08/2021	7

4.Time

Nome	Papel
Beltrano	Usuário Gestor (PO)
João	Analista desenvolvedor (Design)
Ciclano	Analista desenvolvedor
Maria	Analista desenvolvedor
Fulano	Analista desenvolvedor
Pedro	Analista de Req. e Testes
Michel	Coordenador (Scrum Master)

5. Informações sobre este relatório de acompanhamento

- O projeto se divide em tarefas de menor granularidade que são discutidas e envolvem a **participação** e **consenso** de todo o time e do usuário gestor, inclusive, com relação a estimativa dos prazos utilizando-se da experiência e razoabilidade;

- Cada **asterisco** (*) indica uma Sprint de atraso normalmente é seguido de uma justificativa;

- Uma marca de **carinha feliz** =) , com tom de verde mais escuro, indica que o analista antecipou uma tarefa que estava aguardando início no *backlog*;

- Ao final de cada Sprint, espera-se a entrega de funcionalidades para serem **validadas pelo usuário gestor** e testadas pelo analista de testes;

- **Macro Entregas (Escopo):** São tarefas, no nível macro, que identificam o projeto e podem sofrer alteração, exclusão ou inclusão de acordo com o andamento do projeto;

- **Entregas de Análises e Testes:** São tarefas de controle e qualidade durante todo ciclo de desenvolvimento.

5.1 As seguintes cores são utilizadas para indicar o andamento do Projeto:

Azul	Tarefa identificada e aguardando início
Amarela	Tarefa em andamento
Verde claro	Tarefa concluída
Verde escuro	Tarefa concluída antecipadamente
Laranjada	Tarefa cancelada
Vermelha	Tarefa atrasada
Roxa	Tarefa atrasada por outra tarefa
Preta	Tarefa não realizada

6. Backlog do Produto

id		Versão	Data
		02	12/08/21
01	Implementar as novas regras para sigilo de documentos		
02	Melhorar a interface de processos sigilosos		
03	Adicionar avisos quando o processo se tornar sigiloso		
04	...		

7. Entregas de Análises e Testes

Nome	Papel	
Instalação, configuração e alinhamento da equipe	Coordenador	
Histórias de usuários	Anal. de req. e testes	
Especificação de Requisitos	Anal. de req. e testes	
Cronograma	Coordenador com aos *stackeholders*	
Testes funcional	Analistas dev	
Testes caixa preta	Anal. de req. e testes	
Homologação	Equipe de TI e Gestor	
Treinamento e suporte	Equipe de TI	
Rel. de lições aprendidas	Coordenador	

8. Histórico das Sprints (Sp)

Entrega					Início	Fim
					08/07/21	30/08/21
id	**Sp**	**Data de início**	**Data de fim**	**Ação**	**Observação**	
01	0	08/07/21	09/07/21	Requisitos Iniciais		
02	1	12/07/21	12/07/21	Planejamento		
03	1	13/07/21	21/07/21	Desenvolvimento		
04	1	22/07/21	22/07/21	Apresentação		
...	...	...	...	…		
...	...	...	...	….		
11	4	12/08/21	12/08/21	Planejamento		
12	4	13/08/21	23/08/21	Desenvolvimento	Final do prazo estimado	
13	4	24/08/21	24/08/21	Apresentação		
14	5	24/08/21	27/08/21	Correção dos erros		
15	5	30/08/21	30/08/21	Homologação e entrega		

9. Sprint em Andamento

Entrega 05				Início	Fim
				08/07/21	30/08/21
id	**Sp**	**Pacote**	**Tarefa**	**Responsável**	**Situação**
01	0	0	Levantamento de requisitos iniciais e testes dos requisitos	Pedro	Concluído
02	0	0	Mockups e demais especificações de requisitos	Pedro	Concluído
04	1	3	Adicionar marca de sigiloso nos docs e listagens	João	Concluído
05	1	1	Sigilo total no processo	Ciclano	Concluído
06	1	1	Sigiloso documento individual (sigilo ao documento)	Fulano	Concluído*
08	1	1	Esconder informações sobre os documentos sigilosos	Maria	Concluído
09	2	1	Tratar o push para processos sigilosos	Maria	Concluído =)
...	...	...	...		
11	2	0	Testes da Sp1 e atualização dos requisitos	Pedro	Concluído
29	4	4	Sigilo para as notificações e para acompanhamento	Ciclano	Concluído
31	4	4	Melhorar a pesquisa do processo sigiloso	João	Concluído
32	4	4	Teste da Sp3 e atualização dos requisitos	Pedro	Concluído

33	4	5	Erro no sigilo total do processo	Maria	Concluído
34	4	5	Erro ao listar processos sigilosos	Fulano	Concluído
35	4	5	Erro no assistente de libras	João	Concluído

“O talento vence jogos, mas só o trabalho em equipe ganha campeonatos” - Michael Jordan

10. Atrasos

Item	Motivo
06	Fulano precisou fazer manutenção urgente no sistema de Precatórios.:Atualização do Certificado Digital.

11. Reuniões

Entrega 01

REUNIÃO DE PLANEJAMENTO (maiores detalhes, ver doc. de requisitos) - *Planning Meeting*	Data	Hora ini	Hora fim
	12/07/21	14:00h	17:15h

“Informações importantes de planejamento do projeto e distribuição das tarefas para próxima *sprint*”

REUNIÃO DE ACOMPANHAMENTO (Reunião diária com duração de 20 min) - *Daily Meeting*	Data	Hora ini	Hora fim
	14/07/21	17:00h	17:20h
“São feitas três perguntas: O que foi feito até o momento ? O que falta fazer para concluir a tarefa ? - Se tem alguma dificuldade ou impedimento ?”			

REUNIÃO DE APRESENTAÇÃO - *Sprint Review e Sprint Restrospective*	Data	Hora ini	Hora fim
	22/07/21	14h	15:15
“Apresentação ao PO do que está pronto, do que falta ser feito ou modificado. Após a apresentação para o PO, a equipe discute sobre os problemas e sugestões de melhorias diante do que não saiu conforme planejado”			

12. Riscos

Item	Situação	Descrição	Solução
01	ATIVO MITIGADO INATIVO DESEJÁVEL	“Descrição sobre a situação incerta que se ocorrer poderá ter um efeito positivo ou negativo sobre o projeto”	“Provável solução para afastar a ameaça ou obter a oportunidade”

Apêndice C

Questionários com Time de Analista de Sistemas Administrativos e com os Diretores da Engenharia de Software e Diretor de TI

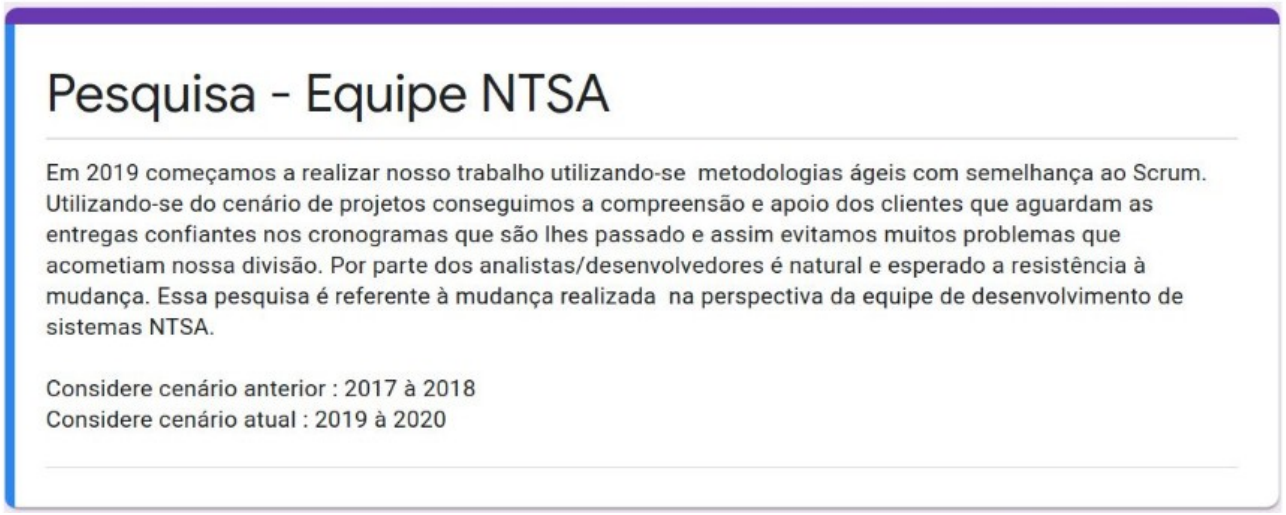
QUESTIONÁRIO 1

A seguinte pesquisa foi disponibilizada na divisão de desenvolvimento de sistemas administrativos do Tribunal de Justiça do Estado de Goiás, denominado Núcleo Técnico de Sistemas Administrativos (NTSA) no período de 01/06/2020 à 06/06/2020.

O questionário foi entregue para 11 pessoas e 9 responderam. São servidores do quadro efetivo lotados no NTSA e na diretoria de Engenharia de Software.

Para formulação e disponibilização do questionário, utilizou-se da ferramenta online Google Forms.

O objetivo foi identificar se as mudanças empregadas, através de métodos ágeis, estavam sendo bem recebidas pelo time e diretoria de informática.



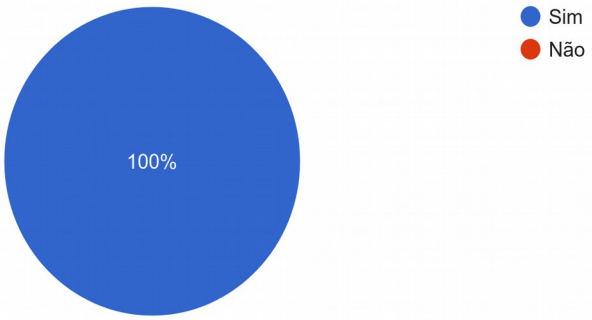
Pesquisa - Equipe NTSA

Em 2019 começamos a realizar nosso trabalho utilizando-se metodologias ágeis com semelhança ao Scrum. Utilizando-se do cenário de projetos conseguimos a compreensão e apoio dos clientes que aguardam as entregas confiantes nos cronogramas que são lhes passado e assim evitamos muitos problemas que acometiam nossa divisão. Por parte dos analistas/desenvolvedores é natural e esperado a resistência à mudança. Essa pesquisa é referente à mudança realizada na perspectiva da equipe de desenvolvimento de sistemas NTSA.

Considere cenário anterior : 2017 à 2018
Considere cenário atual : 2019 à 2020

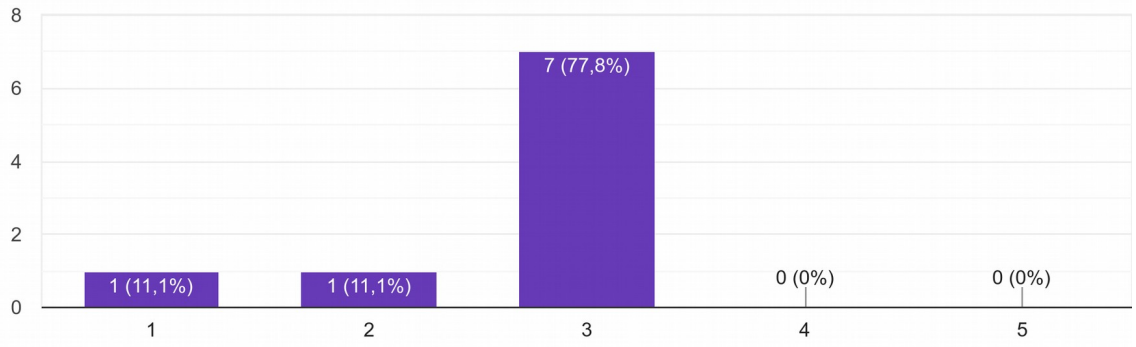
Você reconhecia a necessidade de mudança dado o cenário anterior ?

9 respostas



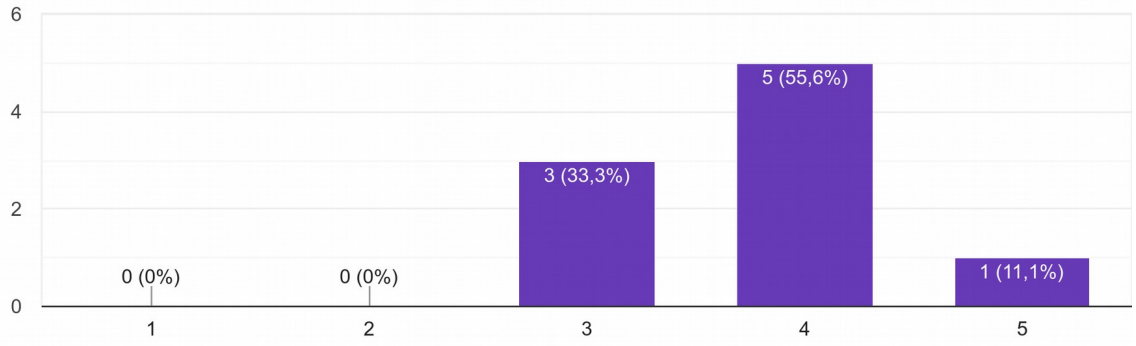
Quanto você se sentia motivado no cenário anterior ?

9 respostas



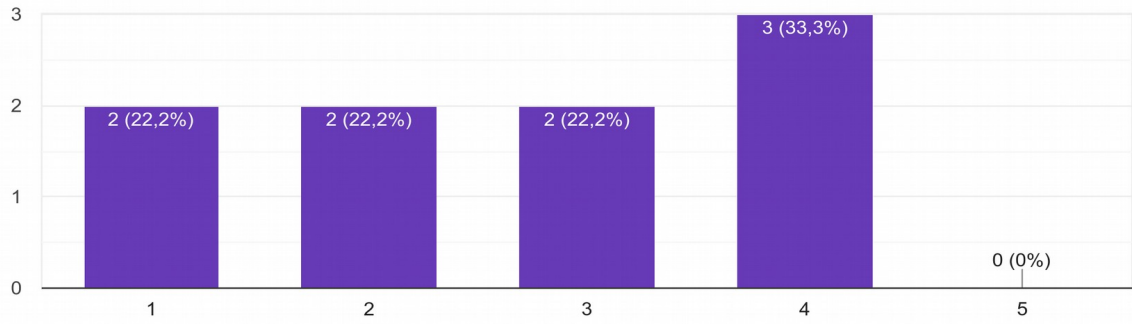
Quanto você se sente motivado no cenário atual ?

9 respostas



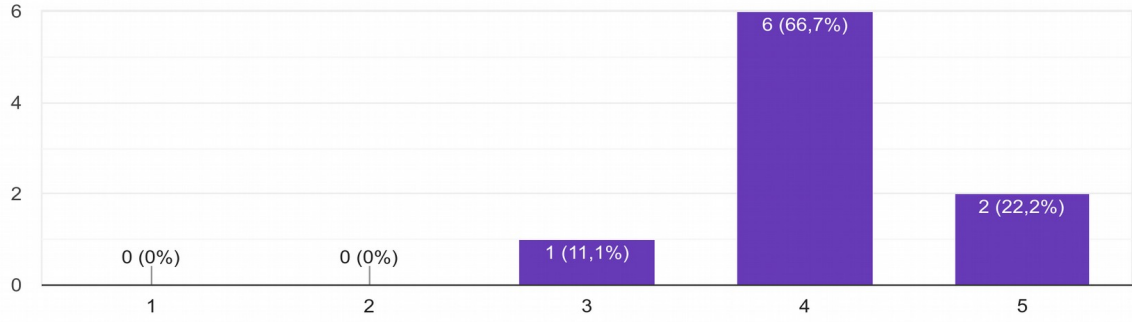
Como você via a produção (quantidade de trabalho entregue) do NTSA no cenário anterior considerando a equipe e não o esforço individual ...antidade de entregas por quantidade de pessoas ?

9 respostas



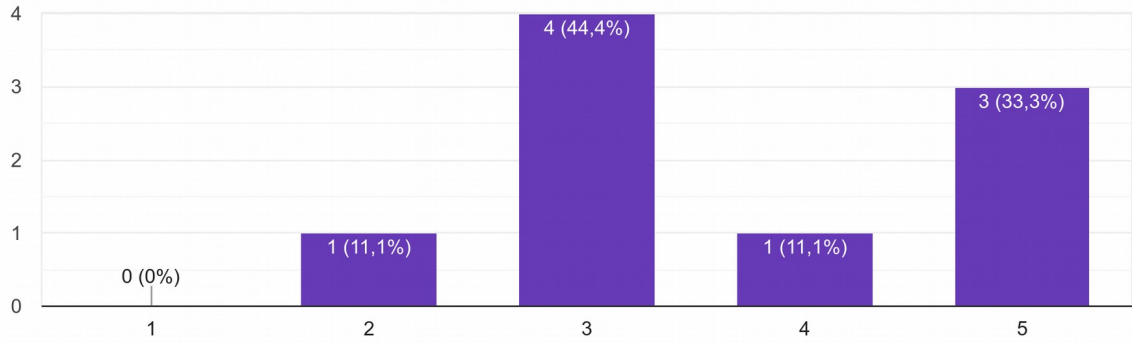
Como você vê a produção (quantidade de trabalho entregue) do NTSA no cenário atual como equipe ?

9 respostas



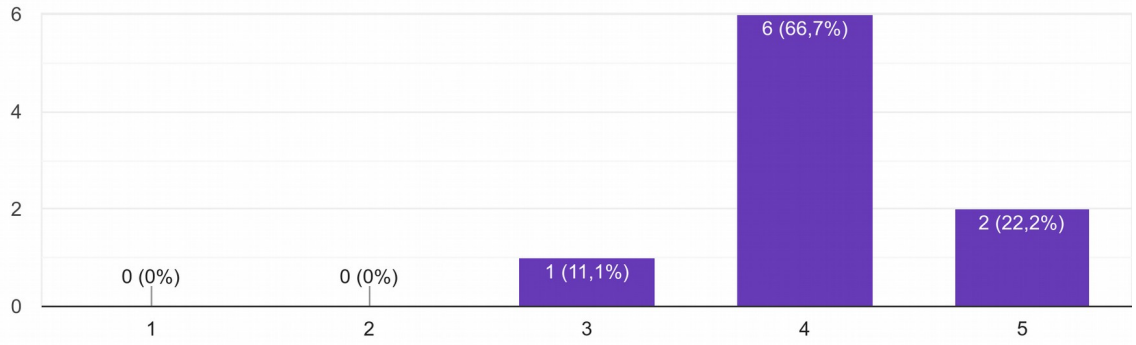
Como você se sentia no cenário anterior quanto às responsabilidades ?

9 respostas



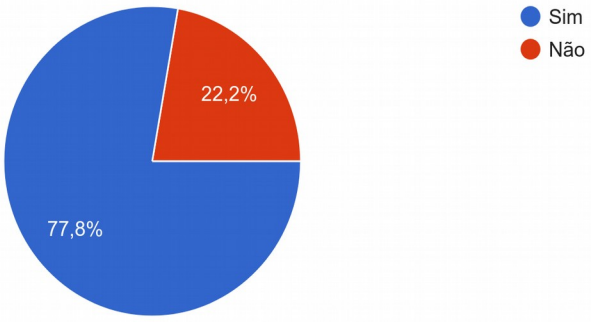
Qual nota você daria quanto a evolução na qualidade dos produtos entregues ?

9 respostas



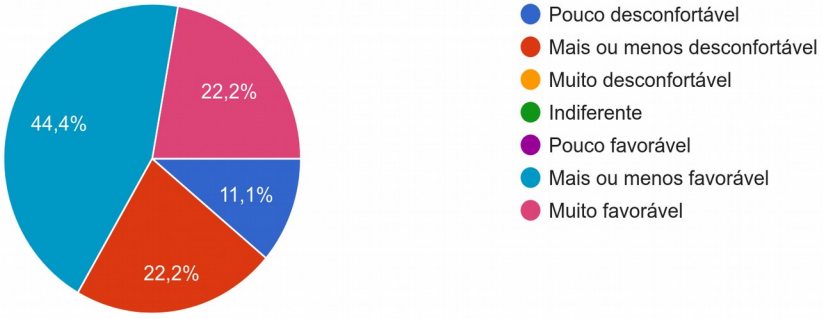
Você acha que a quantidade de manutenções diminuíram ?

9 respostas



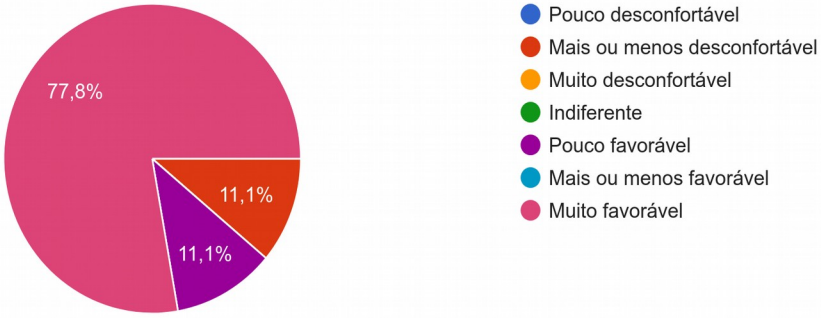
Como você se sentiu no início da mudança ?

9 respostas



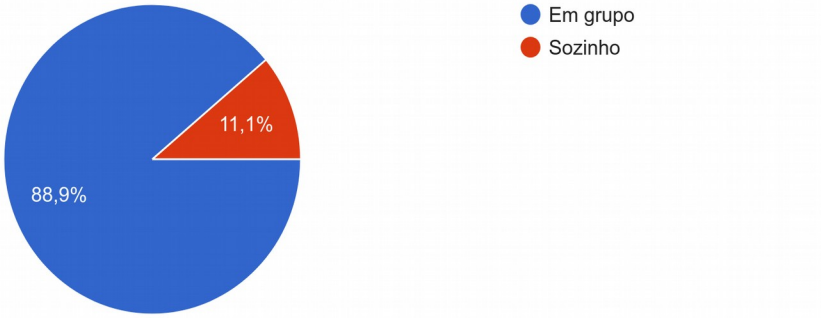
Como você se sente atualmente ?

9 respostas



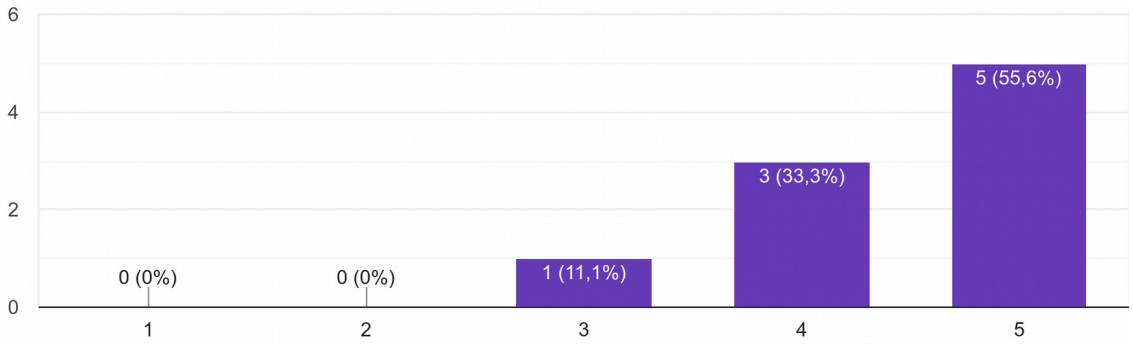
Como você gosta de solucionar os problemas ?

9 respostas



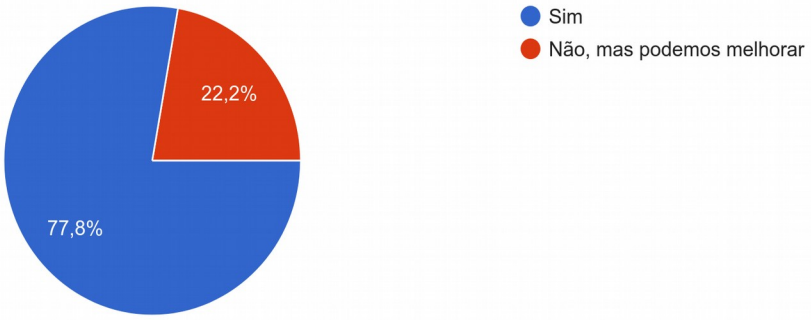
Você acha que as reuniões de planejamento são importantes (geram resultados positivos) ?

9 respostas



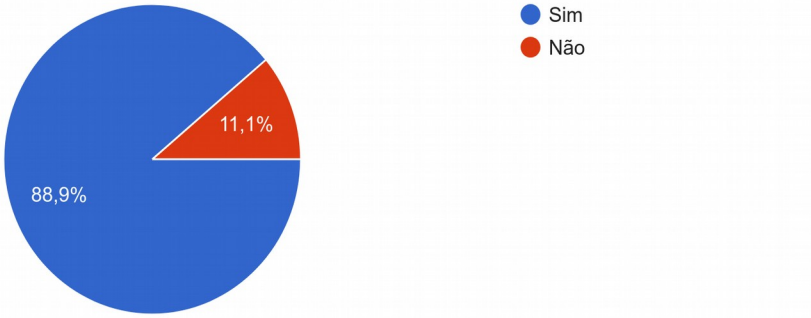
Você acha que nas reuniões de planejamento conseguimos os detalhes suficientes para a Sprint ?

9 respostas



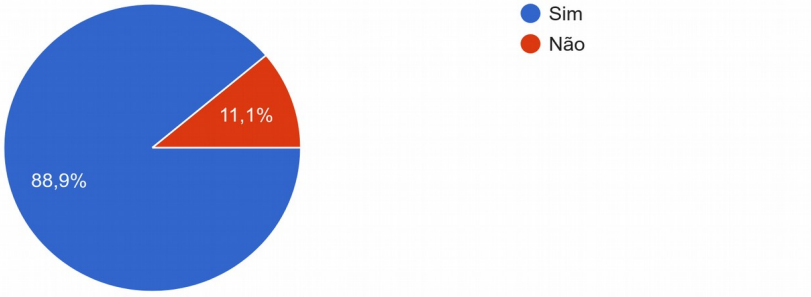
Você acredita que a participação do cliente durante todo projeto agrega qualidade no resultado ?

9 respostas



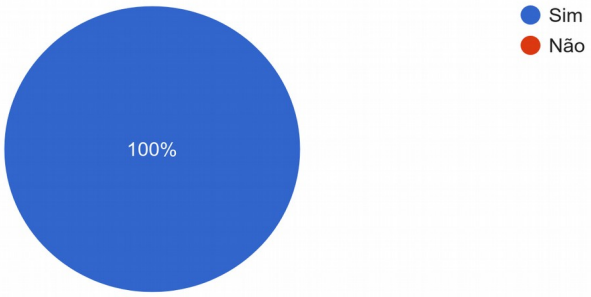
Você acredita que o desenvolvimento de novos sistemas ou manutenções evolutivas precisam ter: início, meio e fim ?

9 respostas



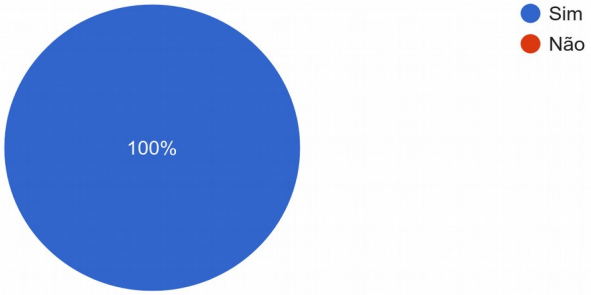
Você acredita que um processo de software bem definido reduz o retrabalho e a manutenção corretiva ?

9 respostas



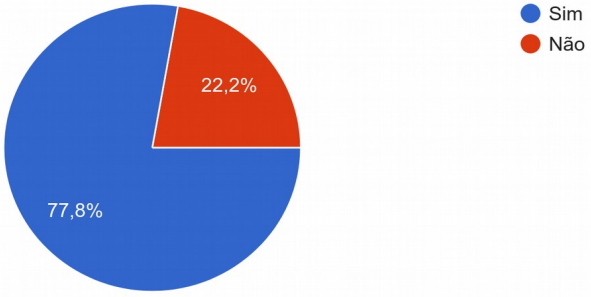
Você acredita que um processo de software bem definido aumenta a qualidade e consequentemente satisfação do cliente?

9 respostas



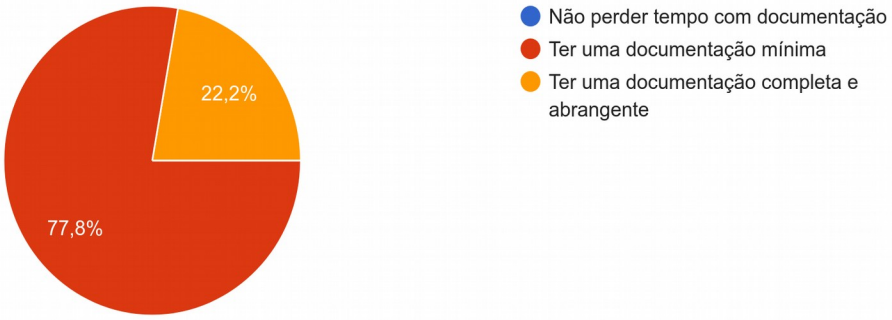
Você acredita que os produtos produzidos em equipe atingem melhores resultados que os produzidos individualmente ?

9 respostas



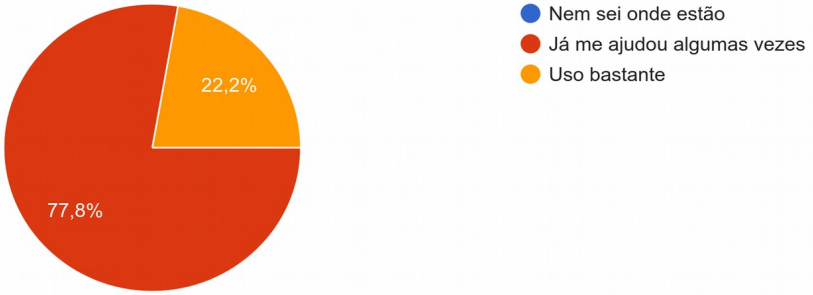
Sobre a documentação você prefere :

9 respostas



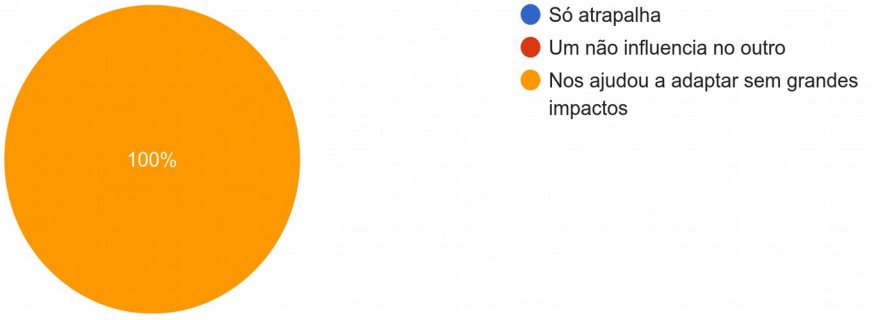
Você já utilizou a documentação (requisitos, testes, acompanhamento do projeto) para se orientar?

9 respostas



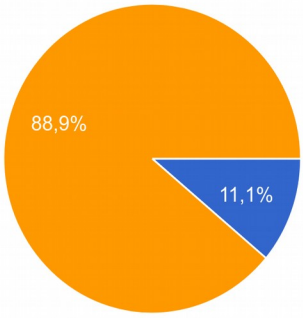
O que você acha do processo em realção ao home office (confinamento devido a Covid-19)?

9 respostas



Você acredita que mais pessoas produzem software mais rápido ?

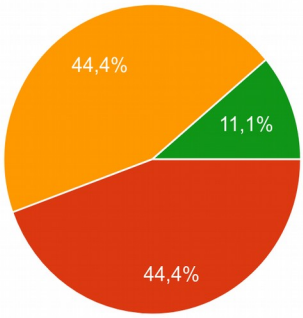
9 respostas



- Sim
- Não
- Talvez

O que você acha sobre prazos ?

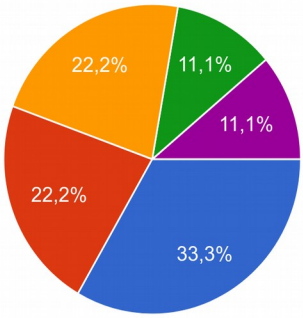
9 respostas



- Não gosto e acho desnecessário
- Não gosto mas acho necessário
- Gosto e não vejo problemas em cumprí-los
- Poderia ser mais tranquilo

Como você atua na execução das tarefas ?

9 respostas



- tento cumpri-las à medida do possível
- me esforço para fazer tudo rápido e ficar mais tempo tranquilo
- me esforço para fazer a tarefa para começar a próxima
- me esforço para fazer a tarefa rápido e deixar top, se der pra dar uma descansada entre uma e outra sucesso, se não der vamo que vamo
- me esforço pra fazer da forma correta

Por favor, fiquem a vontade para deixar, críticas, elogios e sugestões :

6 respostas

Acho que a equipe e a gerência evoluiu muito, mas ainda precisamos amadurecer o processo de trabalho.

Essa gestão trouxe grandes mudanças em relação a forma de trabalhar e eu aprovo mil porcento as mudanças que foram feitas. Na minha experiência no tribunal, passei por diversos projetos que não tinham fases bem definidas, e nem o acompanhamento mais próximo da equipe e interessados. Isto gerava sobrecarga de uns, tranquilidade excessiva pra outros, e produtos que nunca eram entregues, ou que chegavam ao cliente tanto tempo depois que não se sabia se eram ainda interessantes. As reuniões que estão acontecendo são sensacionais, pro acompanhamento da equipe e clientes, e divisão de tarefas entre a equipe. Temos muito pra evoluir ainda mas demos passos inegavelmente importantes.

Acho importante existir um processo bem definido de trabalho, até porque nunca tivemos isso antes, e por isso a evolução é claramente bem grande. Minhas respostas se baseiam muito nisso. Porém em algumas coisas no processo eu me sinto desconfortável, especialmente em planejamento global de projeto com muita gente em uma reunião. Acaba que sempre há umas duas pessoas mais empolgadas que acabam elaborando a maior parte do planejamento enquanto as demais acompanham e pontuam esporadicamente. Mas apesar de me sentir mais a vontade com desenvolvimento e manutenções corretivas, estou disposto a fazer o que precisar ser feito.

A forma atual de trabalho trouxe muitos benefícios e deve ser mantida.

Melhorou muito a comunicação, pelo menos entre aquelas pessoas que realmente se envolve.
Por consequência a produção de software vem melhorando.
Estamos tendo mais contato com coisas novas.
Estamos entregando mais coisas do que antes

Houve uma distribuição de responsabilidade. Não que ela diminuiu. Apenas nos deixa um pouco mais aliviados em saber que essa responsabilidade foi compartilhada, mas seu grau não diminuiu.

Creio que nossa imagem externa está melhorando devido ao processo. Estamos dando visibilidade para o pessoal que não é da área de o que é desenvolver um sistema. Aquelas colocações que sempre escutei, quando participava de reuniões, antes de existir o referido processo..."mas vem cá! O que vocês ficam fazendo mesmo lá na informática" tendem a diminuir.

Quanto a motivação. Eu sempre tive muita motivação, mas meio que ela veio diminuindo no decorrer da minha vida no TJ. Não pela quantidade de trabalho, mas mais por sentimentos das outras pessoas, meio que se sentia meio sozinho e pela questão recompensatória também acho que todos nós ali queremos sempre uma melhoria salarial né? A metodologia empregada está diminuindo um pouco essa solidão que eu estava sentindo relativo ao que eu faço ali no TJ e consequentemente aumentando a minha motivação um pouco.

Me deixa um pouco desconfortável a falta de alinhamento de uso tecnológico entre as pessoas da equipe. No inicio parecia que era meio uma competição de quem é mais que o outro, mas entendo que tem fatores que influenciam esse comportamento: como gratificação financeira e afinidade pessoal também...Nesse ponto estamos evoluindo e deveríamos participar de mais cursos juntos para tentar diminuir um pouco isso que sinto. É um sentimento meu.

Me deixou um pouco desconfortável meio que a mudança do nosso relacionamento extra trabalho. Uns achando que é pura falsidade esse negócio de relacionamento de amizade no trabalho, outros achando que éramos um monte de “piiii” que queria só detonar o outro...em fim isso me entristeceu um pouco...

Dando uma nota pelo trabalho que foi realizado até agora, relativo à metodologia de trabalho, do que era antes do depois, numa escala de 1 a 5 é 4.5.

Talvez pudéssemos ver outra forma de demonstrar o comprometimento da equipe ao diretor, uma vez que ele não é um cliente. Reuniões poderiam ser menos frequentes. A apresentação dos resultados poderiam ser feitas pelo responsável da área. Todos os acertos poderiam recair sobre a equipe e não individualmente, evitando exaltação ou constrangimentos individuais. Os erros ou falhas nem sempre deveriam ser repassandos aos superiores. Os prazos não precisariam ser tão rigorosos.

Apêndice D

Questionários com os Usuários Gestores

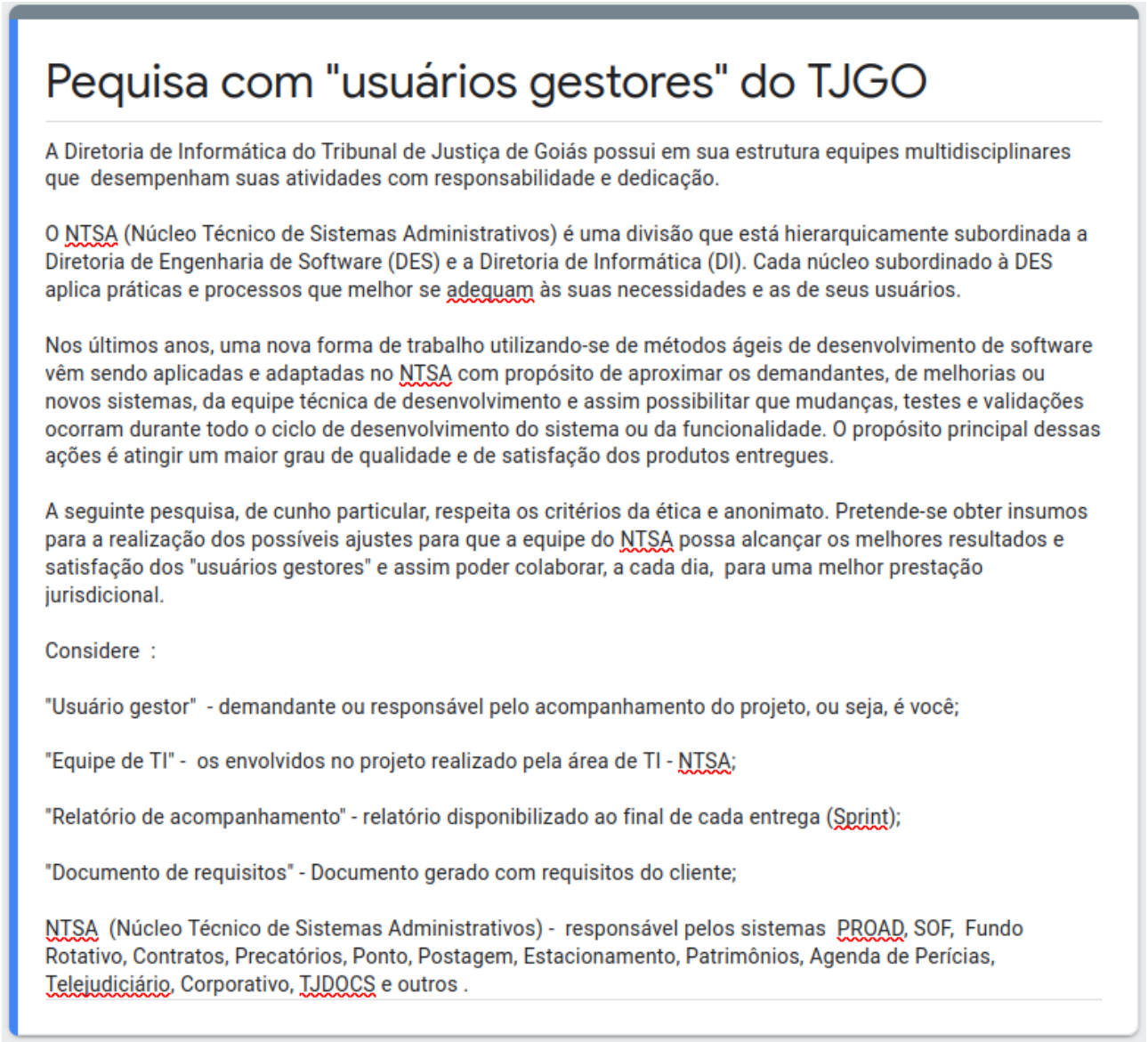
QUESTIONÁRIO 2

A seguinte pesquisa foi disponibilizada para os usuários gestores de acordo com que os projetos de manutenção evolutiva foram realizados, durante o período de 01/06/2020 à 01/02/2021.

O questionário foi entregue para 10 pessoas e 8 responderam. São todos servidores do TJGO designados para acompanhar o desenvolvimento da solução. Possuem bom conhecimento sobre os respectivos sistemas administrativos e sobre as demandas solicitadas.

Para formulação e disponibilização do questionário, utilizou-se da ferramenta online Google Forms.

O objetivos foram identificar se as mudanças empregadas, através de métodos ágeis, estavam gerando bons resultados e se estavam no sentido correto.



Pequisa com "usuários gestores" do TJGO

A Diretoria de Informática do Tribunal de Justiça de Goiás possui em sua estrutura equipes multidisciplinares que desempenham suas atividades com responsabilidade e dedicação.

O NTSA (Núcleo Técnico de Sistemas Administrativos) é uma divisão que está hierarquicamente subordinada a Diretoria de Engenharia de Software (DES) e a Diretoria de Informática (DI). Cada núcleo subordinado à DES aplica práticas e processos que melhor se adequam às suas necessidades e as de seus usuários.

Nos últimos anos, uma nova forma de trabalho utilizando-se de métodos ágeis de desenvolvimento de software vêm sendo aplicadas e adaptadas no NTSA com propósito de aproximar os demandantes, de melhorias ou novos sistemas, da equipe técnica de desenvolvimento e assim possibilitar que mudanças, testes e validações ocorram durante todo o ciclo de desenvolvimento do sistema ou da funcionalidade. O propósito principal dessas ações é atingir um maior grau de qualidade e de satisfação dos produtos entregues.

A seguinte pesquisa, de cunho particular, respeita os critérios da ética e anonimato. Pretende-se obter insumos para a realização dos possíveis ajustes para que a equipe do NTSA possa alcançar os melhores resultados e satisfação dos "usuários gestores" e assim poder colaborar, a cada dia, para uma melhor prestação jurisdicional.

Considere :

"Usuário gestor" - demandante ou responsável pelo acompanhamento do projeto, ou seja, é você;

"Equipe de TI" - os envolvidos no projeto realizado pela área de TI - NTSA;

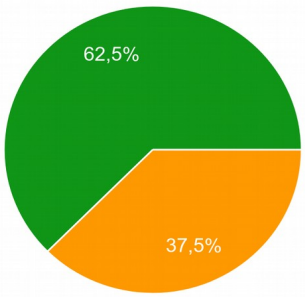
"Relatório de acompanhamento" - relatório disponibilizado ao final de cada entrega (Sprint);

"Documento de requisitos" - Documento gerado com requisitos do cliente;

NTSA (Núcleo Técnico de Sistemas Administrativos) - responsável pelos sistemas PROAD, SOF, Fundo Rotativo, Contratos, Precatórios, Ponto, Postagem, Estacionamento, Patrimônios, Agenda de Perícias, Telejudiciário, Corporativo, TJDOCS e outros .

Considerando o quadro excepcional da pandemia e o trabalho remoto. Você considera que o trabalho da TI, estruturado de forma projetizada c...ra entregar resultados da melhor forma possível ?

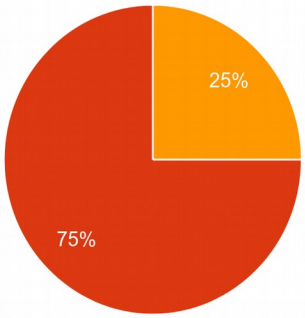
8 respostas



- Acho que não tem nada a ver
- Acho que tem pouco a ver
- Percebi uma melhora no seviço oferecido pela equipe de TI do NTSA
- Percebi grande melhora no seviço oferecido pela equipe de TI do NTSA

Quanto a duração do projeto:

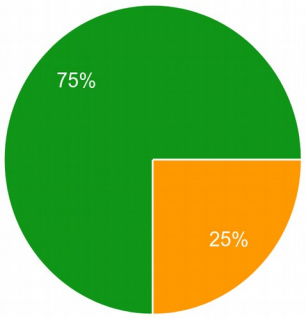
8 respostas



- Demorou mais do que eu imaginava
- Demorou mais ou menos o tempo que eu esperava
- Demorou menos que o esperado

O que foi produzido ficou de acordo com o que foi solicitado.

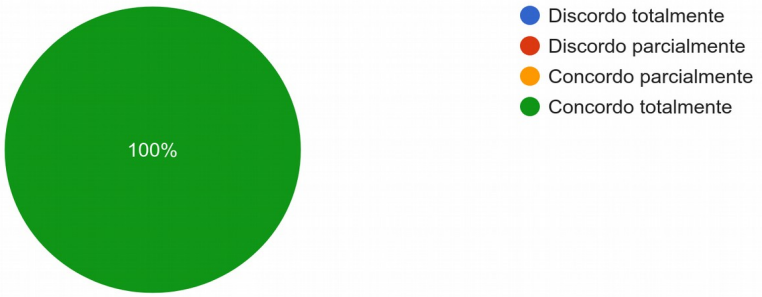
8 respostas



- Discordo totalmente
- Discordo parcialmente
- Concordo parcialmente
- Concordo totalmente

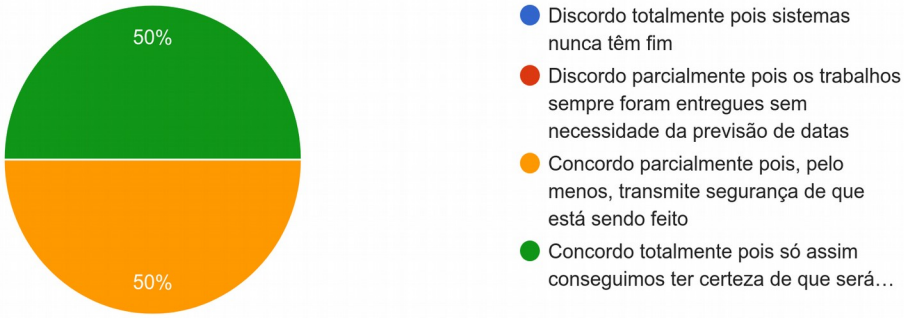
A participação do gestor, do início ao fim, do projeto proporcionou que as alterações fossem feitas antes da conclusão do projeto.

8 respostas



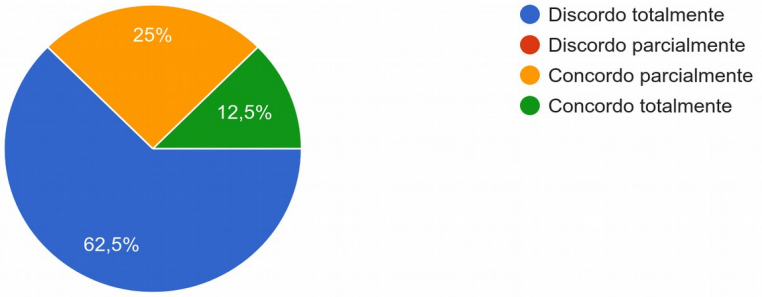
Estimar uma data de início e fim do projeto é importante

8 respostas



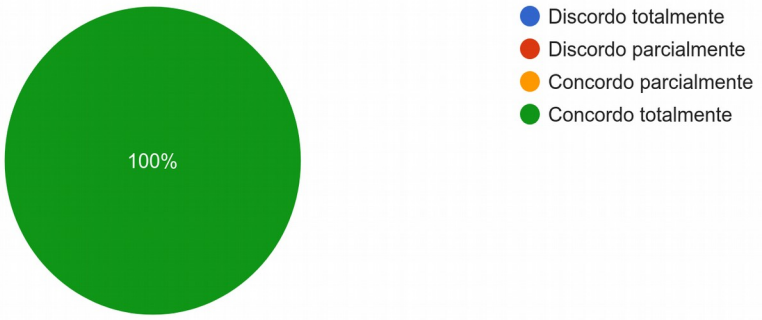
O contato direto com a equipe de TI, utilizando-se grupos do WhatsApp durante o projeto, é desnecessária porque a linguagem utilizada por eles é muito técnica.

8 respostas



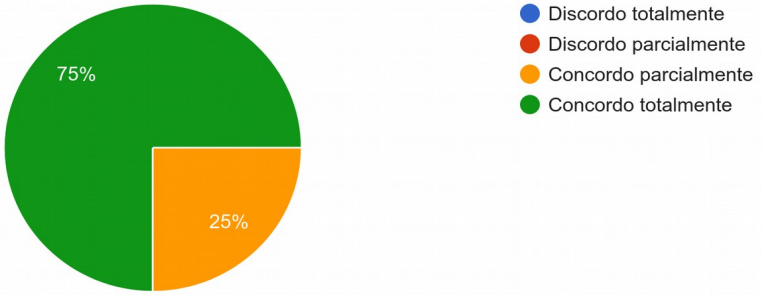
O contato direto com a equipe de TI, utilizando-se grupos do WhatsApp, facilitou a comunicação.

8 respostas



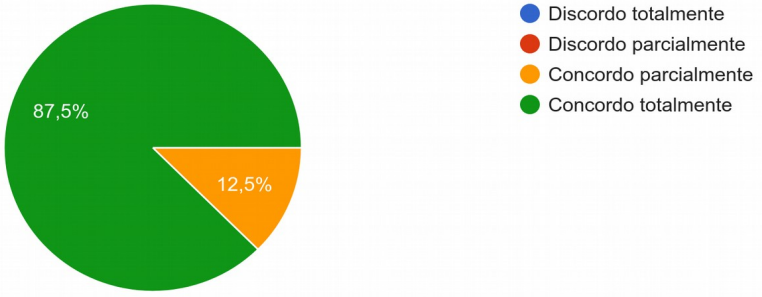
O contato direto com a equipe de TI, utilizando-se grupos do WhatsApp e reuniões de apresentação possibilitaram acompanhar bem o andamento do projeto.

8 respostas



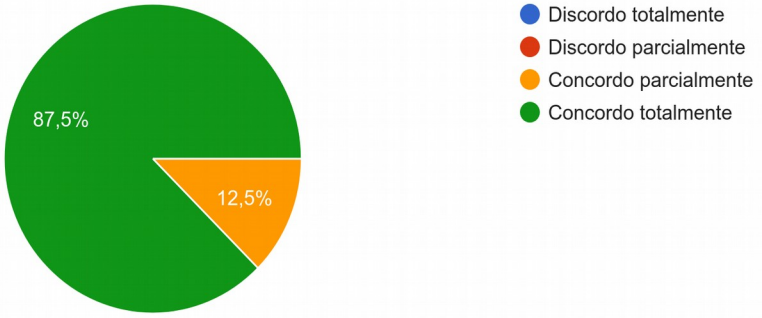
O método possibilitou que eu, como usuário gestor, antecipasse melhorias e mudanças nos requisitos durante o desenvolvimento do projeto.

8 respostas



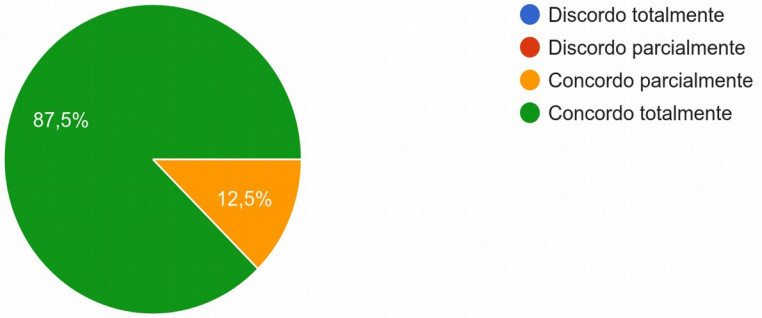
Os "Relatórios de Acompanhamento" ajudam a acompanhar o andamento do projeto.

8 respostas



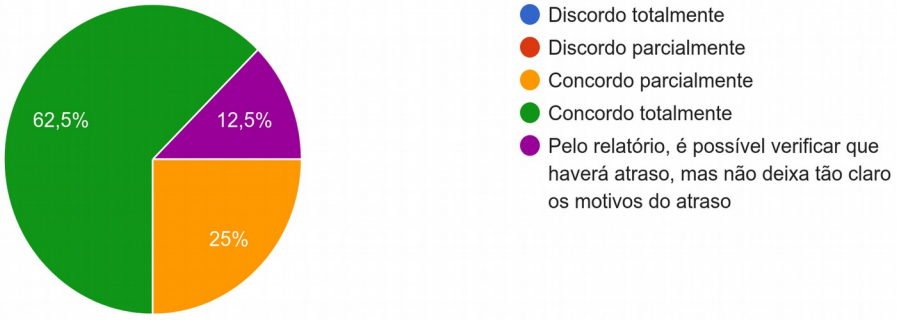
Os "Relatórios de Acompanhamento" deram maior transparência ao projeto.

8 respostas



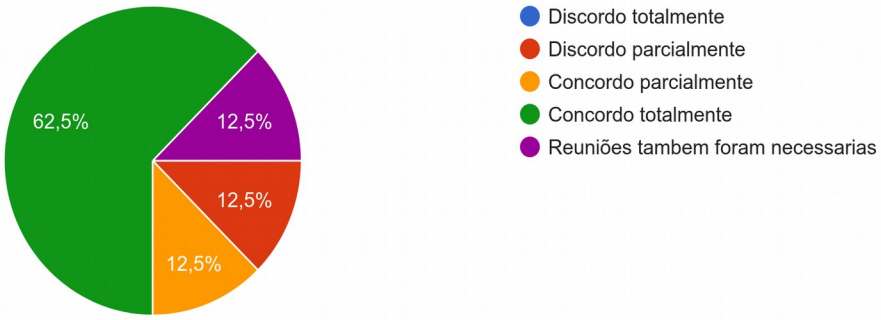
Os "Relatórios de Acompanhamento" ajudam a entender dificuldades e possíveis atrasos.

8 respostas



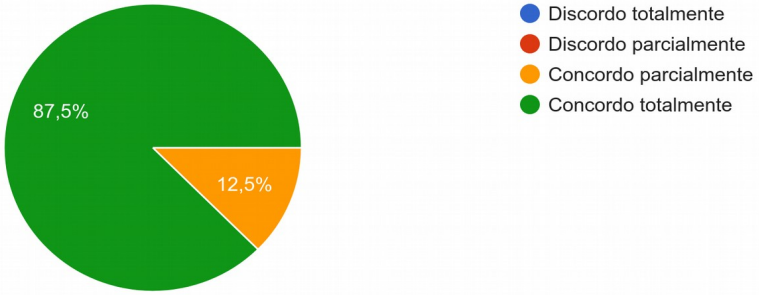
A "Documentação de Requisitos" foi suficiente para entendimento do que seria feito.

8 respostas



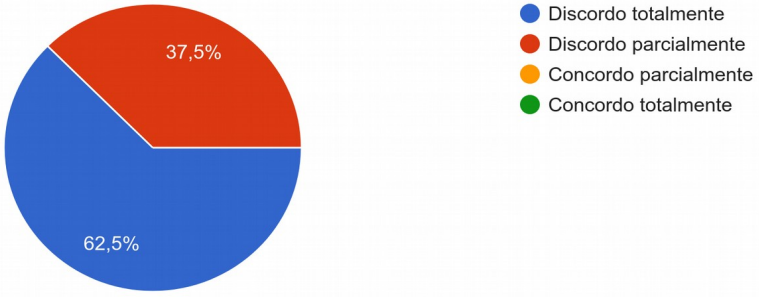
Os testes desde o começo foram suficientes para detectar a maioria dos erros que poderiam prejudicar o funcionamento.

8 respostas



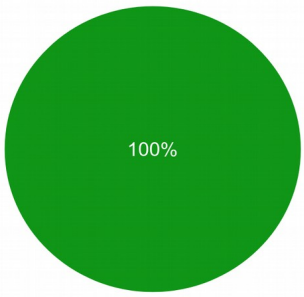
Não acho que documentar requisitos seja necessário já que o "usuário gestor" participou ativamente do projeto.

8 respostas



Quanto a reunião de apresentação trata-se de uma reunião bastante importante pois é nela que consigo acompanhar o andamento da funcionalidade que está sendo desenvolvida.

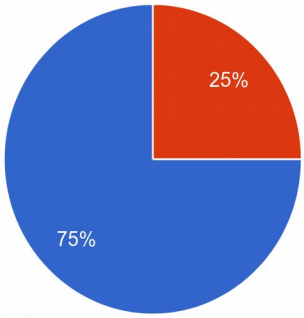
8 respostas



- Discordo totalmente
- Discordo parcialmente
- Concordo parcialmente
- Concordo totalmente

Quanto ao trabalho em equipe, de modo geral :

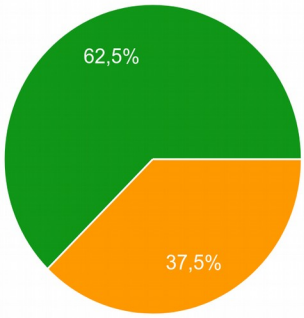
8 respostas



- Percebi um forte entusiasmo da equipe de TI em resover o problema
- Percebi pouco entusiasmo da equipe de TI em resolver o problema
- Percebi a equipe de TI um pouco desmotivada
- Percebi forte desmotivação da equipe de TI

Quanto as relações com a equipe do NTSA durante o projeto :

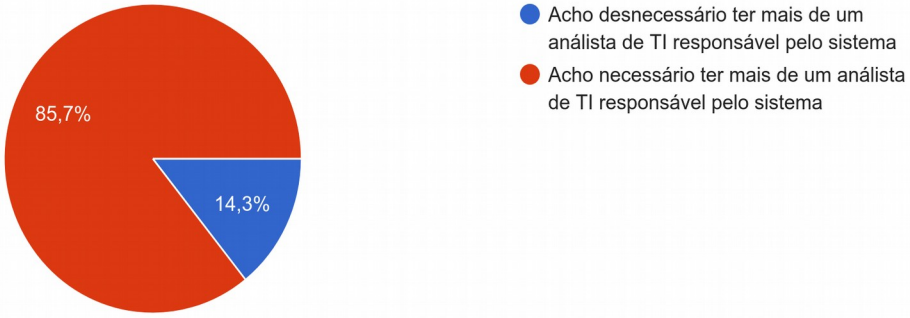
8 respostas



- São pouco receptivos
- São parcialmente receptivos
- São receptivos
- São muito receptivos

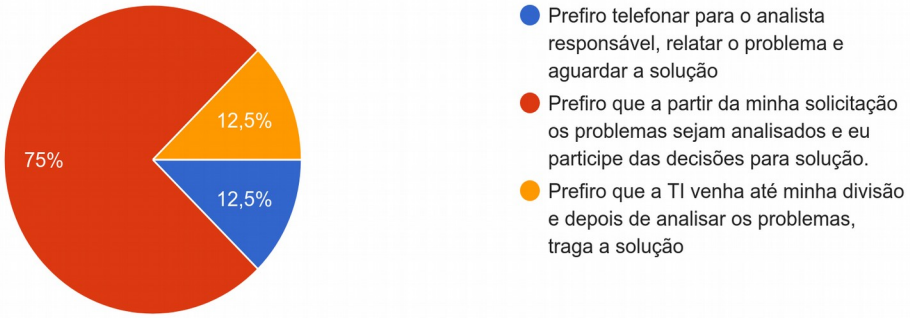
Quanto ao trabalho em equipe :

7 respostas



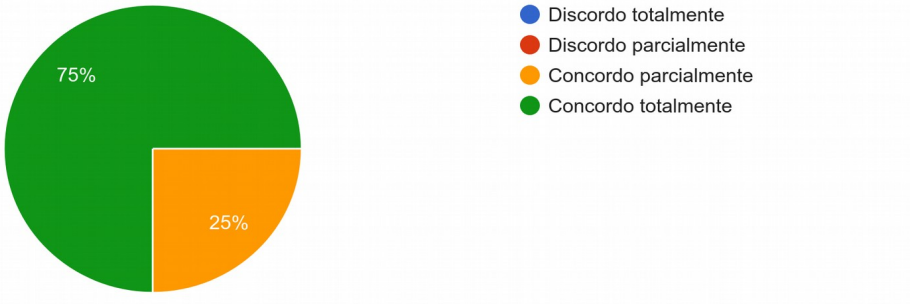
Com relação a forma de trabalho :

8 respostas



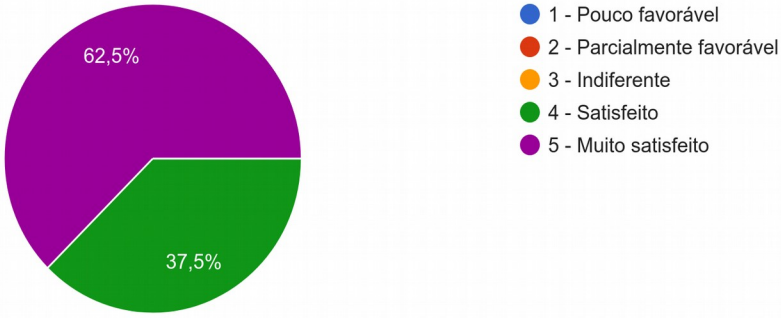
Considere a afirmação : "O sistema é como um carro e não desejo comprar um carro que dê defeito toda semana".

8 respostas



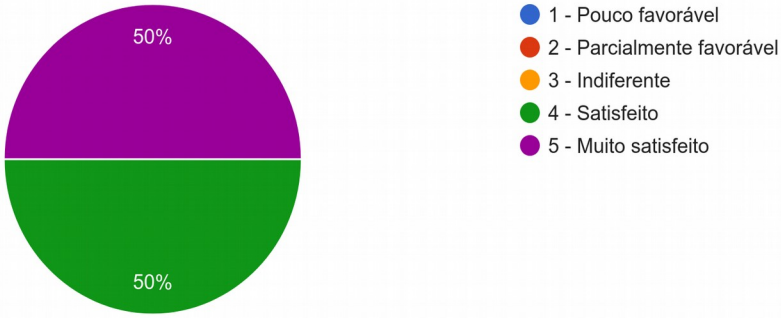
De 1 a 5 qual sua satisfação com a forma de trabalho apresentada.

8 respostas



De 1 a 5 qual sua satisfação com o produto final apresentado.

8 respostas



Apêndice E

Categorias observadas nas entrevistas e revisão bibliográfica

Categoria: Problemas anteriores percebidos

ID	Nome da Citação	Conteúdo da Citação
1:42	Quantidade considerável de defeitos e de retrabalho	Quantidade considerável de defeitos e de retrabalho. Fonte [36]
1:43	Inabilidade de mudar fácil e rapidamente o software, resultando em sol…	Inabilidade de mudar fácil e rapidamente o software, resultando em soluções incompatíveis. Fonte [36]
1:44	Ausência de um processo para as atividades de manutenção	Ausência de um processo para as atividades de manutenção. Fonte [36]
1:45	Insatisfação dos clientes quando solicitações de correção ou modiffcaç…	Insatisfação dos clientes quando solicitações de correção ou modificação resultavam em soluções incompatíveis gerando consequências negativas; Fonte [36]
1:46	Não existência de métricas para se avaliar a qualidade do software que…	Não existência de métricas para se avaliar a qualidade do software que sofreu alteração; Fonte [36]
1:47	Redução da qualidade global do software como resultado de mudanças que…	Redução da qualidade global do software como resultado de mudanças que introduzem erros latentes no software mantido. Fonte [36]
1:50	ausência de metodologia de software; comunicação ruim; ausência de pla…	ausência de metodologia de software; comunicação ruim; ausência de planejamento; atrasos recorrentes e baixa integração entre os desenvolvedores do projeto e os departamentos de infraestrutura. Fonte [29]
1:66	processo de “big design up front”, que requer especiffcação dos requi…	processo de “big design up front”, que requer especiffcação dos requisitos completa e validada antes do início do desenvolvimento real. De acordo com Mergel (2016, apud Mantovani e Marczak [1]) essa estratégia trouxe experiências negativas e falhas de gestão. Fonte [1]
3:5	no momento anterior as demandas chegavam assim de tudo quanto é lugar…	no momento anterior as demandas chegavam assim de tudo quanto é lugar e de todo quanto é nível hierárquico de pessoas do Tribunal. E aí a gente não conseguia falar se faz ou não faz e em quanto tempo faz, se é para semana que vem ou para três dias, e então é muito difícil trabalhar dessa forma. Então assim, mesmo que seja uma demanda pequena, mas são várias, aí vai acumulando
3:6	E: E quando você estava trabalhando em alguma coisa e chegava uma dema…	E: E quando você estava trabalhando em alguma coisa e chegava uma demanda dessa ? R: Pois é [risos] aí parava de fazer uma coisa e fazia essa demanda. Por exemplo era uma pessoa de uma hierarquia lá, fulano de tal que é muito importante e precisa disso (está precisando) então o negócio meio que se atropelava.
3:16	às vezes, o cara chegava ali para mim e a demanda só eu que sabia.	às vezes, o cara chegava ali para mim e a demanda só eu que sabia.
3:17	teve época que a gente fez dois sistemas para atender uma demanda só,	teve época que a gente fez dois sistemas para atender uma demanda só
3:36	Antigamente tínhamos servidores que nem o pessoal da infraestrutura s…	Antigamente tínhamos servidores que nem o pessoal da infraestrutura sabia que existia [risos]
4:5	os analistas faziam praticamente todas as tarefas.	os analistas faziam praticamente todas as tarefas.
4:7	cada um fazia suas atividades.	cada um fazia suas atividades.
4:10	se na nossa área lá, que tinha dez pessoas, dividia em grupos de cinc…	se na nossa área lá, que tinha dez pessoas, dividia em grupos de cinco pessoas, então cada um conseguia pegar aí dois a três projetos no máximo
4:12	Eu acho que sempre foi uma dificuldade o legado, a manutenção. Então e…	Eu acho que sempre foi uma dificuldade o legado, a manutenção. Então eu acho que uma das coisas que sempre dificultava, o início de um processo definido, com início, meio e fim [pensando]... foi essa manutenção, a necessidade de manutenção dos sistemas. Porque manutenção dos sistemas exigia dedicação de um outro, quase uma dedicação exclusiva.
4:17	Aí se você olha lá, era o modelo antigo, o cara era o chefe da área e…	Aí se você olha lá, era o modelo antigo, o cara era o chefe da área e tinham os projetos. Então, eu que ficava com os sistemas, tomava conta dos sistemas, assim eu era o gerente daqueles sistemas e fazia tudo, desde requisitos, implantação, treinamentos, divulgação etc.

4:21	a gente ia no cliente, passava dois, três dias e levantava um tanto de…	a gente ia no cliente, passava dois, três dias e levantava um tanto de documentação, fazia, não tinha mais tanto contato com o cliente e então entregava e daí passa três, quatro meses e o cliente cobrava e entregava
6:7	chegava a demanda totalmente atravessada, de última hora, faça não imp…	chegava a demanda totalmente atravessada, de última hora, faça não importa como se virem,
6:9	Normalmente ele demandam, na maioria das áreas ali, da forma que quer…	Normalmente ele demandam, na maioria das áreas ali, da forma que querem. Sem um vínculo estratégico de metas e objetivos da instituição. Projetos que muitas vezes não vão sequer agregar para o Tribunal.
6:12	a gente estava tentando implantar algo novo de baixo para cima e não a…	a gente estava tentando implantar algo novo de baixo para cima e não a instituição entendendo que precisava ter um processo. Então, a gente acabava lutando sozinhos, a instituição acabava não acompanhando essa ideia de planejamento, de Tecnologia da Informação.
6:13	as demandas chegavam atravessada em montantes gigantescos, uma quanti…	as demandas chegavam atravessada em montantes gigantescos e nós tínhamos que acabar.
6:14	Acabávamos parando para poder atender as demandas que chegavam. E deix…	Acabávamos parando para poder atender as demandas que chegavam. E deixávamos de planejar, de fazer todo um alinhamento porque a instituição não acompanhava ideia que nós tínhamos então a gente tava caminhando, nadando, sozinhos. Nós não tínhamos o apoio da alta gestão e sem esse apoio não chegamos a lugar nenhum.
6:15	a partir do momento que você está atolado de demandas você acaba sacr…	a partir do momento que você está atolado de demandas você acaba sacrificando o controle, o controle dessas demandas, o controle das atividades, a gente ficava muito preocupado é “ter que entregar”, “ter que entregar no prazo, prazo, prazo, prazo”, “não depois eu preencho”, “depois eu anoto no Redmine”, “depois...”, mas aí isso vira uma bola de neve e o “depois” nunca chegava.
7:1	o pessoal antigamente pelo menos, trabalhava bem individual, cada um…	o pessoal antigamente pelo menos, trabalhava bem individual, cada um da sua maneira, no seu próprio sistema e isso também dificultava um pouco a gente poder definir um método de trabalho,
7:4	Geralmente o pessoal trabalhava com poucos sistemas , um ou dois, só e…	Geralmente o pessoal trabalhava com poucos sistemas , um ou dois, só exclusivamente. Era até ruim porque poucas pessoas tinham o conhecimento, tanto de negócio quanto do funcionamento do sistema e aí geravam alguns problemas quando a pessoa não estava disponível, tava de férias… era bem complicado a gente ter que atender alguma demanda.
7:8	R: Chegava por telefone ou o pessoal ia lá na sala. E: E isso você ac…	R: Chegava por telefone ou o pessoal ia lá na sala. E: E isso você achava bom ou ruim? R: Não. Horrível ! O pessoal chegava lá, já ia fazendo a solicitação e as vezes a gente estava trabalhando em outra coisa.
7:9	E: E quando você estava trabalhando em outra coisa, o que que aconteci…	E: E quando você estava trabalhando em outra coisa, o que que acontecia? R: Era complicado né ? A gente tentava dialogar, assim...Ver se era urgente mesmo. Se não fosse, pelo menos terminar a demanda que estava sendo feita naquele momento, mas era complicado. Tendo um ponto que receba as demandas, priorize e tenha a parte de delegar mesmo, dividir ali entre os integrantes da equipe eu acho que é bem melhor.

9:8	É simplesmente parou. É daquelas coisas que não pegou. Tipo a própria…	É simplesmente parou. É daquelas coisas que não pegou. Tipo a própria posição da diretoria que não pegou. Não sei se não era prático ou não é. Eu acredito que era mais por conta da nossa cultura lá dentro. Aquela coisa de ter muito aquela cultura mais amadora e acabava que qualquer tentativa de implantação de algum padrão, de um processo mais qualificado acabava tendo que quebrar algumas barreiras. Isso começou a ser feito há uns dois anos atrás e que realmente conseguiu fazer com que algumas barreiras fossem quebradas. Foi um trabalho que teve que... foi executado por você, inclusive, que foi um trabalho de correr atrás, de dizer não, vai ser assim. A gente tem que melhorar, a gente tem que se profissionalizar mais e tal e isso demandou um certo trabalho de quem estava tentando implantar. Acho que isso que tenha faltado antes né
9:11	O processo era um processo e inicialmente era bem arcaico. Primeiro qu…	O processo era um processo e inicialmente era bem arcaico. Primeiro que você tinha um acesso direto. Que em alguns pontos dava aquela falsa impressão de praticidade, mas era bem perigoso. Você tinha acesso direto praticamente a produção direto, você desenvolvia sem passar por ninguém, já fazia seu deploy e estava em produção. Não passava por ninguém e era uma forma inicialmente, uma forma de quando entrei no tribunal, uma forma de copiar os arquivos pra área onde seria sincronizada e acabou, era simplesmente isso.
9:14	Era uma coisa bem… foi uma decadência em termos de padrão que eu tive…	Era uma coisa bem… foi uma decadência em termos de padrão que eu tive durante a carreira. Esse sistema era uma coisa bem simples, quase: “faz aí e manda !” o próprio trabalho em equipe era … [pensado] as vezes tinha que falar o que você está fazendo pro colega do lado e digitar o mesmo arquivo.
9:19	O que era no processo antigo, era simplesmente chegar a alguma coisa…	O que era no processo antigo, era simplesmente chegar a alguma coisa por telefone e às vezes da diretoria antiga era quase como se os clientes fossem chefes. E isso acontecia com o aval da própria diretoria nossa, que acontecia de um diretor financeiro te passar uma demanda direta sem passar por qualquer hierarquia dentro da TI e quando você pedia para entrar em contato, seguir uma hierarquia básica pelo menos, a própria figura de diretores antigos que existiram só repassava o telefone e pronto você trata direto com o diretor financeiro ou outra pessoa que virou na prática o seu chefe. E aí ninguém mais na equipe está sabendo o que você está fazendo
9:20	Eu já passei dias que eu estava desesperado e ninguém da equipe sabia…	Eu já passei dias que eu estava desesperado e ninguém da equipe sabia. As vezes tem dias que você não está… e também ninguém da equipe sabe porque não tem uma forma oficial de saber.
12:3	um problema de cultura organizacional. Desde o início, desde que eu e…	um problema de cultura organizacional. Desde o início, desde que eu estou na diretoria de TI eu vejo que o tribunal sempre tratou essa questão, de desenvolvimento de software, como algo que compete exclusivamente à antiga Diretoria de informática, atual diretora de TI, não fazendo... não querendo fazer parte do processo como demandantes, como levantadores de requisitos dos sistemas, de sistemas. Vejo que sempre foi muito isso. Jogava a responsabilidade para cima da diretoria, queria que a gente resolvesse de forma rápida. Não existia governança, nesse sentido das demandas, então era atendido quem pedia mais, então não existia uma análise se a demanda é realmente importante, é necessária, se ela deve ou não deve ser feita, primeiramente juridicamente, posteriormente analisando pelo critério técnico.
12:9	porque muitas pessoas abriam Proad (Processos Administrativos) e mand…	porque muitas pessoas abriam Proad (Processos Administrativos) e mandavam direto para a gente e esse processo por falta de prioridade ficavam sobrestados realmente na diretoria. E é aonde os usuários reclamavam que: atuou o Proad, a não sei quanto tempo, e tá até hoje parado

Categoria: Tentativas de uso do Kanban no TJGO

ID	Nome da Citação	Conteúdo da Citação
3:4	muita gente tinha outros papéis ou fazia outras tarefas e se sentiam a…	muita gente tinha outros papéis ou fazia outras tarefas e se sentiam assim: “poxa agora vou ter mais um trabalho, de ter que controlar alguma coisa ali, um painel, a pessoa me liga e eu vou ter que ir lá e anotar ?”
3:7	E aí, por exemplo isso impedia que eu outras pessoas iam lá e se orga…	E aí, por exemplo isso impedia que eu outras pessoas iam lá e se organizasse com as demandas que se está fazendo Obs: Aqui o entrevistado está se referindo a quantidade de demandas e como elas chegavam atrapalhando qualquer tentativa de se organizar
4:23	E: Na gestão passada tentaram utilizar o Redmine. Eu tentei colocar em…	E: Na gestão passada tentaram utilizar o Redmine. Eu tentei colocar em prática o Kanban no começo da gestão. Mas, nenhum evoluiu. Você tem uma opinião sobre isso ? Por que não foram para frente ? R: Assim, nesse caso, eu acho que faltou treinamento e acompanhamento. Toda ferramenta, toda ferramenta tem que ser … [pensando]. Tem duas formas, no órgão público ou no privado também, ou você institucionaliza e obriga: “a partir de hoje será assim !” ou você começa a ter maior acompanhamento disso.
6:17	Tem uma questão cultural interna nossa também da Diretoria de Informá…	Tem uma questão cultural interna nossa também da Diretoria de Informática que é muito técnica e acha, muitas vezes, que essa parte de gestão é muito burocrática e é desnecessária, que é uma coisa que a gente tem que…[pensando] nós estamos tentando quebrar isso, na cabeça do nosso pessoal internamente porque eles precisam começar a enxergar que nós somos área estratégica e não só executora. A gente precisa ter organização, assim, a maioria não tem uma ideia um pouco mais administrativa e de gestão mesmo de ter que lançar as atividades, ter que controlar o que está sendo feito, de ter evidências das atividades executadas, dos projetos entregues, se foi homologado, se não foi homologado.
7:3	Acho que o pessoal tem um pouco de resistência. Talvez com a ferrament…	Acho que o pessoal tem um pouco de resistência. Talvez com a ferramenta [pensativa] ou falta de hábito assim.
9:9	isso foi uma coisa que realmente ninguém lembrava de usar, na verdade…	isso foi uma coisa que realmente ninguém lembrava de usar, na verdade. Então parou porque ninguém usava. Você deixava as coisas lá mas a gente nem lembrava de usar na verdade, eu acho. Eu via lá,
9:10	Acabava que a gente via bastante porque tinha reunião de acompanhamen…	Acabava que a gente via bastante porque tinha reunião de acompanhamento direto, mas era só nisso... não é uma coisa natural, mais mas neste caso era inércia mesmo. A gente tinha um costume de mais de décadas de trabalho que para quebrar isso tem que demandar um esforço mesmo, e você sabe disso né ?

Categoria: Peculiaridades no setor público

ID	Nome da Citação	Conteúdo da Citação
1:2	Isso pode indicar um campo importante para pesquisas, com intuito de…	Isso pode indicar um campo importante para pesquisas, com intuito de se construir um bom conhecimento sobre adaptação das práticas ágeis aos diferentes cenários. Fonte: [20]
1:49	Os autores destacam que as organizações públicas possuem cultura, modo…	Os autores destacam que as organizações públicas possuem cultura, modo de operação e entrega diferentes do setor privado, o que podem tornar a implementação dessas metodologias um desafio. Fonte [29]
1:51	aplicaram o 10Scrum com algumas adaptações necessárias ao setor públi…	aplicaram o Scrum com algumas adaptações necessárias ao setor público. Fonte [29]
1:55	o poder hierárquico é um desafio para a implantação dos métodos ágeis	o poder hierárquico é um desafio para a implantação dos métodos ágeis. Fonte[30]
1:56	realizaram um estudo que se concentrou em oferecer uma visão integrad…	realizaram um estudo que se concentrou em oferecer uma visão integrada de como as organizações podem se tornar ágeis, como combinar valores e missão com agilidade organizacional a partir de estruturas orientadas a objetivos. Fonte[30]
1:57	relataram que a incapacidade da burocracia de se tornar ágil é causad…	relataram que a incapacidade da burocracia de se tornar ágil é causada por fatores como: tempo reduzido, custo e sobreposições políticas. Fonte[30]
1:59	no modelo de gestão pública tradicional, os especialistas em gerencia…	no modelo de gestão pública tradicional, os especialistas em gerenciamento de informações se limitam a tarefas de gestão de contratos, planejamento e supervisão e, por isso, é bastante comum a resistência às mudanças. Fonte [31]
1:65	Ainda hoje os desaffos são grandes quando se trata de agilidade em emp…	Ainda hoje os desafios são grandes quando se trata de agilidade em empresas públicas brasileiras, inclusive nas universidades públicas, como demonstra o estudo de Maurício e Neris. Fonte [55]
1:68	A aplicação de ágeis no Governo brasileiro é, em sua maioria, bem-suce…	A aplicação de ágeis no Governo brasileiro é, em sua maioria, bem-sucedida e, normalmente, conduzida em combinação com outras abordagens ágeis de desenvolvimento de software ou, em sua maioria, uma combinação de ágeis com métodos tradicionais. Fonte [1]
1:72	O propósito foi atender de forma rápida e com qualidade as demandas d…	O propósito foi atender de forma rápida e com qualidade as demandas da Administração Pública, que cada vez mais exigem agilidade na produção de software. Fonte [60]
1:74	buscaram evidências científfcas sobre os benefícios da utilização de…	buscaram evidências científicas sobre os benefícios da utilização de métodos ágeis na Administração Pública brasileira, em detrimento ao histórico de fracassos de projetos de TI no governo. Fonte [58]
3:21	O que talvez atrapalha ali no Tribunal, justamente é essa hierarquia…	O que talvez atrapalha ali no Tribunal, justamente é essa hierarquia [pensando]... horizontal. E aí isso atrapalha um pouco. A gente sente que, às vezes, depende do gosto lá do pessoal de cima e aí atrapalha um pouquinho sim, essa questão de ter muita hierarquia.
4:2	Existia um processo em que era delegado, por meio principalmente das…	Existia um processo em que era delegado, por meio principalmente das competências das áreas, entendeu ? Então, eu acho que tinha um, e em razão dos cargos.
4:3	tem as diretorias, os serviços, os cargos os papéis então estava meio…	tem as diretorias, os serviços, os cargos os papéis então estava meio que definido aí e tava no entendimento geral de que era assim, de que funcionava assim. Não era documentado.
4:15	A questão dos cargos, talvez dificultava porque, por exemplo: se a ge…	A questão dos cargos, talvez dificultava porque, por exemplo: se a gente fosse definir um processo de software, a gente sempre imaginava um processo de software mas não vendo o diretor da área como o gerente do projeto, mas sim como o gerente da área.
4:16	Você conseguiu fazer essa mudança. Você assumiu a gerência dos projeto…	Você conseguiu fazer essa mudança. Você assumiu a gerência dos projetos e isso não tinha né ? Isso até então não tinha. Normalmente o diretor da área não queria assumir, até porque era uma transição

4:18	Aí mudou. Quando você assume um papel de gerente dos projetos aí faz m…	Aí mudou. Quando você assume um papel de gerente dos projetos aí faz mais sentido, porque acaba que ocorre um choque. Por exemplo: a gente tá numa estrutura de projeto dentro da estrutura funcional.
4:19	E: Em que você acha que isso impacta ? A existência da burocracia, hie…	E: Em que você acha que isso impacta ? A existência da burocracia, hierarquia, legislação ? R: Eu acho que isso impacta, porque, por exemplo: se eu sou gerente de projeto dentro da área em que tem um chefe. Qual vai ser a minha autonomia em relação à equipe ? Qual vai ser a minha autonomia em relação ao tempo ? Ao cronograma ?
4:20	Poderia dar conflito [pensando] e sua função poderia ser subutilizada…	Poderia dar conflito [pensando] e sua função poderia ser subutilizada e ser simplesmente um distribuidor de RH: “eu quero fulano ali” então eu acho que o mais certo nessa estrutura funcional é que, talvez o diretor, seja encarado como gerente de projeto. Se tiver um gerente de projeto, se ele tiver autonomia, anda. Mas, se ele não tiver autonomia acaba que fica concorrente
4:29	sempre pensava no processo como se fosse uma fábrica é às vezes a fáb…	sempre pensava no processo como se fosse uma fábrica é às vezes a fábrica você pensa: “nossa tem gente nova todo dia chegando, tá crescendo, a equipe está ficando grande e a demanda está aumentando e a equipe está crescendo como se fosse na empresa privada”, se aumentou a demanda eu vou lá e contrato
6:6	E: Na sua opinião, por que a DES não possui um processo de software be…	E: Na sua opinião, por que a DES não possui um processo de software bem definido ? Você já falou um pouco na outra resposta, mas poderia falar um pouco mais ? R: Eu creio que é uma questão cultural do próprio Tribunal de Justiça. O Tribunal, acho que nesse momento está começando a enxergar a diretoria de informática e tecnologia da informação como uma área estratégica e nos últimos anos a gente não viu isso,
6:8	então eles estão percebendo, que nós temos que estar presentes nessas…	então eles estão percebendo, que nós temos que estar presentes nessas tomadas de decisão, nessa decisões estratégicas e que eles precisam ter consciência também do planejamento.
6:11	para a gente conseguir ter um dia um processo mesmo que funcione a in…	para a gente conseguir ter um dia um processo mesmo que funcione a instituição tem que mudar a forma de pensar
6:22	acaba que você tem que adequar uma ideia de gestão de projetos numa re…	acaba que você tem que adequar uma ideia de gestão de projetos numa realidade do Tribunal. Não tem algo que você consegue ir lá e só encaixar
6:23	A gente tem que fazer as adequações para realidade do Tribunal que é…	A gente tem que fazer as adequações para realidade do Tribunal que é bem específica com os nossos usuários bem “melindrosos” muitas vezes
9:4	Talvez no serviço público tenha o adicional de que é... [pensando] Te…	Talvez no serviço público tenha o adicional de que é... [pensando] Tem mais burocracia, tem mais os empecilhos burocráticos para se implantar alguma coisa. Você não basta chegar e decidir: “não vamos colocar isso aqui”, porque sempre vai esbarrar em alguém ou alguém que está acostumado com alguma coisa e não quer mudar

Categoria: Fatores que prejudicam o processo

ID	Nome da Citação	Conteúdo da Citação
1:19	embora existam muitas metodologias projetadas para lidar com incertez…	embora existam muitas metodologias projetadas para lidar com incertezas como as próprias metodologias ágeis, RUP e outras, elas não são adequadas para fazer alterações em situações de emergência que requerem manutenção em curto tempo. Fonte [24]
1:20	as funcionalidades perdidas podem ser recuperadas após o período de e…	as funcionalidades perdidas podem ser recuperadas após o período de emergência, sendo que para isso basta utilizar-se de uma manutenção adicional capaz de recuperar as principais funcionalidades e a partir daí, retornar ao ciclo normal de desenvolvimento e manutenção. Fonte [24]
1:21	situação de emergência não pode desperdiçar tempo com pesadas document…	situação de emergência não pode desperdiçar tempo com pesadas documentações de requisitos presentes nas metodologias tradicionais. Fonte [24]
1:36	poucas pesquisas investigaram os principais fatores que influenciam a…	poucas pesquisas investigaram os principais fatores que influenciam a produtividade do time ágil. Fonte [20]
1:41	Por outro lado, a implementação de processos de software é uma ativid…	Por outro lado, a implementação de processos de software é uma atividade que requer grande esforço e investimento por parte das empresas. Fonte [35]
1:52	diversas resistências como as impostas por fatores culturais e pelos…	diversas resistências como as impostas por fatores culturais e pelos integrantes do time Fonte [29]
1:53	Estudos apontam que um dos principais desafios para implantação de Mét…	Estudos apontam que um dos principais desafios para implantação de Métodos Ágeis é o de promover a mudança cultural na organização. Fonte [56, 57, 31, 58]
1:58	Entretanto, eles afirmam que, embora a estrutura hierárquica não ajud…	Entretanto, eles afirmam que, embora a estrutura hierárquica não ajude com os processos ágeis, esse não é o maior problema e sim o fato de que as organizações, ao buscarem agilidade, tendem a desestruturar a hierarquia, esquecendo-se dos objetivos organizacionais. Fonte [30]
1:59	no modelo de gestão pública tradicional, os especialistas em gerencia…	no modelo de gestão pública tradicional, os especialistas em gerenciamento de informações se limitam a tarefas de gestão de contratos, planejamento e supervisão e, por isso, é bastante comum a resistência às mudanças.Fonte [31]
1:66	processo de “big design up front”, que requer especificação dos requi…	processo de “big design up front”, que requer especificação dos requisitos completa e validada antes do início do desenvolvimento real. De acordo com Mergel (2016, apud Mantovani e Marczak [1]) essa estratégia trouxe experiências negativas e falhas de gestão. Fonte [1]
1:89	Atualmente as pesquisas em manutenção ainda são poucas, principalmente…	Atualmente as pesquisas em manutenção ainda são poucas, principalmente quando comparadas ao desenvolvimento [26]
1:90	A ISO/IEC 12207 [64] possui grupos de atividades que podem ser usadas…	A ISO/IEC 12207 [64] possui grupos de atividades que podem ser usadas em todo ciclo de vida do software, para diferentes tipos de projetos mas não detalha como implementar as atividades dependendo do tipo de manutenção que se irá fazer.
1:95	Suarez et al. [43] constataram que o Mantema junto aos ágeis é o pref…	Suarez et al. [48] constataram que o Mantema junto aos ágeis é o preferido para realização das manutenções dada sua qualidade, competitividade e histórico, sendo um processo de muito sucesso nos últimos dez anos. Entretanto trata-se de um método complexo que requer tempo e recursos humanos consideráveis para sua implementação.

1:102	Ahmad et al. [73] pesquisaram os motivos de algumas empresas Finlandes…	Ahmad et al. [73] pesquisaram os motivos de algumas empresas Finlandesas estarem deixando o Scrum para utilizarem o Kanban na manutenção dos sistemas. Seus resultados mostraram que as equipes de manutenção, ao utilizar Scrum, enfrentaram desafios como falta de visibilidade do trabalho, dificuldades em priorizar as tarefas, falhas na comunicação e colaboração, sobrecarga devido aos compromisso exigidos pelas sprints, dificuldades em sincronizar os trabalho ou substituir pessoas.
1:103	Ahmad et al. [73] demonstraram que as equipes de manutenção não conse…	Ahmad et al. [73] demonstraram que as equipes de manutenção não conseguiam encaixar as atividades urgentes nas sprints pré-estabelecidas e as datas de lançamentos eram quase aleatórias. Além disso as reuniões diárias, muitas vezes, duravam bem mais do que o estabelecido e muitas vezes os membros do time não estavam disponíveis para comparecerem nas reuniões. Fatores que foram resolvidos ao se utilizar o Kanban.
1:105	o Kanban, assim como outros métodos, possui significantes desafios qu…	o Kanban, assim como outros métodos, possui significantes desafios que impactam negativamente no processo de desenvolvimento de software como falta de mecanismo de rastreamento do andamento do projeto, dificuldades em se estimar o WIP, atrasos e consequentemente falhas nos projetos [74].
1:108	Heeager e Rose [26] revelou os desafios de se adotar Scrum para proces…	Heeager e Rose [26] revelou os desafios de se adotar Scrum para processos de manutenção. Eles relataram que muitos dos problemas de se aplicar Scrum para manutenção se dá ao fato de que uma emergência não pode aguardar o cumprimento de uma sprint com todos seus requisitos e quando acontecem acabam atrasando o cumprimento das metas da sprint. Assim afirmaram que a aplicação dos métodos ágeis na manutenção podem não ocorrer automaticamente e algumas otimizações podem ser necessárias para acomodar os diferentes tipos de manutenção, diferentes fontes de trabalho e práticas locais.
1:142	Normalmente espera-se que uma arquitetura de CD assegure poder comput…	Normalmente espera-se que uma arquitetura de CD assegure poder computacional elástico e alta capacidade de monitoramento para suportar as mudanças frequentes nos requisitos, fato que gera implicações financeiras devido a dificuldade em se mensurar a quantidade de recursos necessários [90].
1:145	As arquiteturas monolíticas geralmente usam uma abordagem de implantaç…	As arquiteturas monolíticas geralmente usam uma abordagem de implantação "big bang"que atualiza toda aplicação de uma vez e algumas vezes incluem atualizações de banco de dados. Essa abordagem pode ser lenta e sujeito a erros devido suas ramificações pesadas e, desse modo, menos ágeis [79]
1:149	Entretanto, deve-se buscar equilíbrio na quantidade de microsserviços…	Entretanto, deve-se buscar equilíbrio na quantidade de microsserviços pois a modularização em excesso pode ser um fator de aumento da complexidade para Continuous Delivery [85]
1:151	Uma especificação de requisitos inadequada funciona como catalisador…	Uma especificação de requisitos inadequada funciona como catalisador para outros problemas, como baixa produtividade da equipe e dificuldade em manter o software [98].
1:155	Embora o uso de requisitos ágeis gerem ganho inicial de tempo, a ausên…	Embora o uso de requisitos ágeis gerem ganho inicial de tempo, a ausência de detalhes na documentação podem resultar em não alinhamento com as expectativas do cliente [15].
1:156	Medeiros et al. [100] cerca de 50% das tarefas de manutenção analisada…	Medeiros et al. [100] cerca de 50% das tarefas de manutenção analisadas tiveram problemas devido a insuficiência de documentação de requisitos.
1:157	Mendes et al. [103] mostraram que o débito de documentação gera uma a…	Mendes et al. [103] mostraram que o débito de documentação gera uma aumento no esforço 47% maior que o estimado no projeto e um custo extra que chega aos 48%.

1:158	Borrego et al. [104] um dos desafios dos métodos ágeis é manter o con…	Borrego et al. [104] um dos desafios dos métodos ágeis é manter o conhecimento ao longo do tempo devido à insuficiência de documentação.
1:160	É comum empresas delegarem a responsabilidade dos testes para outros t…	É comum empresas delegarem a responsabilidade dos testes para outros times especialistas. Ao fazerem isso o processo de desenvolvimento ágil é prejudicado pois normalmente os times especialistas estão acostumados com o modelo tradicional de testes. Essa situação gera conflitos entre as equipes que acabam culpando umas às outras pela demora [103]
1:168	O momento para criação de testes automatizados varia de acordo com a e…	O momento para criação de testes automatizados varia de acordo com a experiência e conhecimento da equipe [110], muitas vezes a prioridade está na velocidade da entrega, mesmo que seja necessário abrir mão da qualidade e de outros atributos.
3:1	Na sua opinião, por que a DES ainda não possui um processo de softwar…	Na sua opinião, por que a DES ainda não possui um processo de software bem definido ? R: Bom eu acho que é por conta de não ter uma, [pensando] … uma união entre as pessoas que queiram implementar esse processo de software. Uma união e uma, digamos assim, alguém para "bater o martelo" e falar : "esse vai ser o nosso processo
3:2	Então, assim, mesmo dentro da nossa equipe ainda vejo "escape" de ide…	Então, assim, mesmo dentro da nossa equipe ainda vejo "escape" de ideias para definir o processo se está ok ou não está. É isso daí que eu acho que peca um pouco para expandir para toda DES.
3:3	eu acho que o RUP seria uma coisa muito robusta e talvez muito formal…	eu acho que o RUP seria uma coisa muito robusta e talvez muito formal para nosso ambiente aqui de início. Acho que isso pegou um pouco. Tem muitos artefatos e começou-se uma discussão de quais artefatos seriam usados ou não. Acho que peca nisso, a gente colocar um processo muito estruturado num momento que a gente não tinha muita experiência, ou capacidade de usar.
3:20	mesmo que tem um código ali meio bagunçado	mesmo que tem um código ali meio bagunçado
4:1	sempre teve um processo não documentado, não documentado, mas eu acho…	sempre teve um processo não documentado, não documentado, mas eu acho que desde o início tinha um processo de software. Talvez não no padrão da literatura, da arquitetura ou da engenharia mas existia um processo
4:6	já tinha uma forma de fazer	já tinha uma forma de fazer
4:8	Eu acho que a outra coisa que levou também a não ser tão definido, fo…	Eu acho que a outra coisa que levou também a não ser tão definido, foi que, também, no início era muito misturado… [pensando]... os servidores [pensando] – Como vou dizer ? - servidores de TI, principalmente os analistas, programadores, banco de dados atuavam muito diretamente com as outras áreas.
4:15	A questão dos cargos, talvez dificultava porque, por exemplo: se a ge…	A questão dos cargos, talvez dificultava porque, por exemplo: se a gente fosse definir um processo de software, a gente sempre imaginava um processo de software mas não vendo o diretor da área como o gerente do projeto, mas sim como o gerente da área.
4:16	Você conseguiu fazer essa mudança. Você assumiu a gerência dos projeto…	Você conseguiu fazer essa mudança. Você assumiu a gerência dos projetos e isso não tinha né ? Isso até então não tinha. Normalmente o diretor da área não queria assumir, até porque era uma transição
4:22	dificuldade do RUP era a documentação que era exigida também sabe, eu…	dificuldade do RUP era a documentação que era exigida também sabe, eu cheguei até a ver as coisas lá mas era mais pesado e talvez faltasse treinamento, e por conta de ter pouca gente, pouca gente assim, de já ter gente dedicada a manutenção de sistema, essa pessoa talvez não tenha tido tanto interesse em querer … se sou eu que faço a documentação, se sou eu que faço os testes, se sou eu que faço tudo, para que eu vou fazer documentação para mim mesmo ?
4:35	demonstração do sistema, ainda em desenvolvimento, eu acho arriscado.	demonstração do sistema, ainda em desenvolvimento, eu acho arriscado.

6:3	implantar em todo o contexto que a gente trabalha em todas as áreas,…	implantar em todo o contexto que a gente trabalha em todas as áreas, porque nós temos diversas frentes de desenvolvimento e, nem todas ainda, a gente conseguiu implantar um processo completo. A gente ainda está caminhando para isso.
6:10	gente deve ter umas 500 demandas ali em frente à nossa divisão de Eng…	gente deve ter umas 500 demandas ali em frente à nossa divisão de Engenharia de Software e muitas delas não agregam em nada de valor para o Tribunal é simplesmente um: “eu quero”.
6:16	Tem uma questão cultural interna nossa também da Diretoria de Informá…	Tem uma questão cultural interna nossa também da Diretoria de Informática que é muito técnica e acha, muitas vezes, que essa parte de gestão é muito burocrática e é desnecessária, que é uma coisa que a gente tem que…[pensando] nós estamos tentando quebrar isso, na cabeça do nosso pessoal internamente porque eles precisam começar a enxergar que nós somos área estratégica e não só executora.
6:21	a resistência foi grande, acho que a resistência já diminuiu mas elaai…	a resistência foi grande, acho que a resistência já diminuiu mas ela ainda existe, mas foi grande. Eu vi com bons olhos,
7:1	o pessoal antigamente pelo menos, trabalhava bem individual, cada um…	o pessoal antigamente pelo menos, trabalhava bem individual, cada um da sua maneira, no seu próprio sistema e isso também dificultava um pouco a gente poder definir um método de trabalho,
7:3	Acho que o pessoal tem um pouco de resistência. Talvez com a ferrament…	Acho que o pessoal tem um pouco de resistência. Talvez com a ferramenta [pensativa] ou falta de hábito assim.
7:7	Tem pessoas que tem um pouco de resistência, mas eu acho que é import…	Tem pessoas que tem um pouco de resistência, mas eu acho que é importante.
9:1	Eu acho que é um padrão, meio do mercado brasileiro, essa falta de pa…	Eu acho que é um padrão, meio do mercado brasileiro, essa falta de padronização
9:2	talvez tenha muito a ver com cultura nossa	talvez tenha muito a ver com cultura nossa
9:3	A pessoa vai lá e faz tudo e acaba sendo prático, barato. Assim a cur…	A pessoa vai lá e faz tudo e acaba sendo prático, barato. Assim a curto prazo era barato e o pessoal acomodava
9:4	Talvez no serviço público tenha o adicional de que é... [pensando] Te…	Talvez no serviço público tenha o adicional de que é... [pensando] Tem mais burocracia, tem mais os empecilhos burocráticos para se implantar alguma coisa. Você não basta chegar e decidir: “não vamos colocar isso aqui”, porque sempre vai esbarrar em alguém ou alguém que está acostumado com alguma coisa e não quer mudar
9:5	Uma pessoa que era de fora da equipe e chegou impondo e talvez tenha…	Uma pessoa que era de fora da equipe e chegou impondo e talvez tenha tido uma resistência interna
9:7	Quando tem algo muito rígido imposto acaba afetando essa questão da l…	Quando tem algo muito rígido imposto acaba afetando essa questão da liberdade mesmo, eu já vi em outros ambientes. Você realmente é limitado e isso acaba mais atrapalhando do que ajudando.
9:10	Acabava que a gente via bastante porque tinha reunião de acompanhamen…	Acabava que a gente via bastante porque tinha reunião de acompanhamento direto, mas era só nisso... não é uma coisa natural, mais mas neste caso era inércia mesmo. A gente tinha um costume de mais de décadas de trabalho que para quebrar isso tem que demandar um esforço mesmo, e você sabe disso né ?
11:7	Agora se considerar a data que a demanda foi encaminhada para o setor…	Agora se considerar a data que a demanda foi encaminhada para o setor né de engenharia de software até efetivamente começar o desenvolvimento aí eu acho que foi um tempo maior,

Categorias: Práticas úteis para o processo

ID	Nome da Citação	Conteúdo da Citação
1:4	conseguiram melhorar o processo por meio de alterações no ambiente fí…	conseguiram melhorar o processo por meio de alterações no ambiente físico, melhoria contínuas de manutenção, uso de histórias de usuário registradas em fichas que facilitaram a gerência dos projetos e ajustes do escopo (removendo ou adicionando histórias nas iterações) e através de apresentações semanais para dar maior visibilidade a todos os envolvidos;
1:6	utilizou-se de refatoração e correção dos módulos que geravam constan…	utilizou-se de refatoração e correção dos módulos que geravam constantes manutenções
1:7	A transparência no desenvolvimento melhorou a comunicação e entrosame…	A transparência no desenvolvimento melhorou a comunicação e entrosamento dos envolvidos e a rápida resposta aos problemas encorajaram a troca de experiências em prol de soluções e sucesso do time.
1:8	planejamento em detalhes a partir de histórias de usuários em período…	planejamento em detalhes a partir de histórias de usuários em períodos de duas semanas, desenvolvimento em pares, testes contínuos, e cliente próximo da equipe. Fonte [28]
1:9	os participantes foram solicitados a responder subjetivamente o quão…	os participantes foram solicitados a responder subjetivamente o quão fácil eles esperavam que seria a introdução do método ágil na empresa e suas respostas foram submetidas a uma análise qualitativa e classificadas utilizando-se o método Analytic Hierarchy Process (AHP). Fonte [28]
1:22	desenvolvimento de soluções viáveis, com o mínimo de interrupções das…	desenvolvimento de soluções viáveis, com o mínimo de interrupções das funcionalidades existentes dentro de prazos curtos e com miníma participação dos clientes. Fonte [24]
1:26	Estreitar a colaboração entre executivos de negócios e desenvolvedores	Estreitar a colaboração entre executivos de negócios e desenvolvedores Fonte [24]
1:27	Mesclar modelagem de negócios, análise de requisitos e projeto em uma…	Mesclar modelagem de negócios, análise de requisitos e projeto em uma fase de par-prototipagem (entre o analista e o cliente). Fonte [24]
1:28	Fazer uso do processo de design ágil usando prototipagem de papel e pr…	Fazer uso do processo de design ágil usando prototipagem de papel; Fonte [24]
1:29	Substituir a documentação formal por matrizes leves separadas em situa…	Substituir a documentação formal por matrizes leves separadas em situações crí- ticas e não críticas. Fonte [24]
1:30	software funcionando é mais valorizado do que documentação muito abra…	software funcionando é mais valorizado do que documentação muito abrangente e desse modo algumas metodologias sugerem que a documentação seja simples podendo até mesmo serem substituídas por artefatos descartáveis. Fonte [24]
1:31	medidas ambientais que contribuam para a colaboração também podem ser…	medidas ambientais que contribuam para a colaboração também podem ser aprimoradas por meio de Arranjos de Trabalhos Flexíveis. Fonte [33]
1:32	horário flexível, redução do horário de trabalho e compartilhamento d…	horário flexível, redução do horário de trabalho e compartilhamento de trabalho. Fonte [33]
1:34	deve implementar ferramentas de controle effcazes	deve implementar ferramentas de controle eficazes. Fonte [33]
1:35	avaliar continuamente os fatores de produtividade é importante, pois…	avaliar continuamente os fatores de produtividade é importante, pois eles podem mudar sob as novas práticas de engenharia de software. Fonte [34]
1:37	Veriffcação empírica da importância da produtividade	Verificação empírica da importância da produtividade. Fonte [34]
1:39	importância da melhoria contínua de processos e seus resultados indic…	importância da melhoria contínua de processos e seus resultados indicaram que a continuidade de programas de melhoria estão relacionados aos fatores humanos, de projeto, organizacionais e técnicos relacionados ao processo e também concluiu que alguns desses fatores estão presentes também na fase de implementação mas que o nível de importância é diferente quando comparado ao de manutenção. Fonte [35]

1:54	Uma análise mais profunda da cultura organizacional pode auxiliar na…	Uma análise mais profunda da cultura organizacional pode auxiliar na gestão da mudança e, consequentemente, contribuir para a implementação de ações táticas do processo de inovação com as metodologias ágeis. Fonte [56]
1:71	apresentou uma proposta de melhoria para o Processo de Desenvolviment…	apresentou uma proposta de melhoria para o Processo de Desenvolvimento de Software do Exército Brasileiro, baseado em técnicas do modelo Ágil e Gestão de Riscos em conjunto com o guia PMBOK. Fonte [60]
1:81	No entanto, em caso de uma manutenção emergencial o apreço as tecnolog…	No entanto, em caso de uma manutenção emergencial o apreço as tecnologias deve dar lugar a soluções fáceis e rápidas. Novas e complexas tecnologias podem realmente diminuir o ritmo e prolongar o estado de emergência [24].
1:85	projeto é deffnido como: “um esforço temporário empreendido para cria…	projeto é deffnido como: “um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo” Fonte [62]
1:91	para Polo et al. [20] as normas ISO/IEC 12207 [67] e IEEE 1219 [68] d…	para Polo et al. [20] as normas ISO/IEC 12207 [67] e IEEE 1219 [68] deveriam ser unidas e detalhadas com foco na manutenção
1:93	Essa metodologia deffne uma estratégia de manutenção ágil, estabelecen…	Essa metodologia deffne uma estratégia de manutenção ágil, estabelecendo em detalhes o que deve ser realizado, quando, como e por quem. Fonte [67]
1:94	Para Pino et al. [68] diferentemente de outros métodos ágeis como Prog…	Para Pino et al. [68] diferentemente de outros métodos ágeis como Programação Extrema e Programação Pragmática o Scrum fornece uma estrutura para gerenciar projetos sem incluir práticas técnicas enquanto que os demais se concentram nas descrições práticas das atividades e técnicas relacionadas ao ciclo de desenvolvimento de software. Esses aspectos possibilitam combinar a estrutura de gerenciamento de projetos do Scrum com processos técnicos de uma área de conhecimento específfca como o Mantema que passa a usufruir dos benefícios de uma metodologia ágil.
1:96	O guia Mantelasoft [48] Se divide em oito fases em sequência que são:…	O guia Mantelasoft [48] Se divide em oito fases em sequência que são: manutenção geral, planejamento da manutenção, processo de manutenção, métricas de manutenção, estimativa de custos de manutenção, pessoal de manutenção, evolução ou melhoria dos produtos e atividade ffnais de manutenção. Essas fases, por sua vez, possuem tarefas e atividades que instruem sobre a maneira de se realizar a manutenção utilizando-se de um check-list [48].
1:98	o Impress se sustenta nos seguintes pilares [23]: • Deploy contínuo e…	o Impress se sustenta nos seguintes pilares [23]: • Deploy contínuo e rápido através de técnicas de priorização de tarefas; • Visibilidade e controle do fluxo de tarefas através do quadro Kanban e suas práticas; • Engajamento ponta-a-ponta através da participação de todos os envolvidos no ciclo de vida das tarefas; • Monitoramento contínuo com metas de curto prazo (semanais), reuniões diárias e manutenção em tempo real de métricas e Dashboards públicos.
1:99	Quando se trata de manutenção evolutiva, o Impress se baseia nas práti…	Quando se trata de manutenção evolutiva, o Impress se baseia nas práticas do Scrum e enfatiza as diffculdade de se eleger o Product Owner (PO) em instituições em que os sistemas de grande porte são compartilhados por muitos POs e desse modo o Impress recomenda a criação de comitês envolvendo representantes das diversas áreas para a tomada de decisões. As demais práticas do Scrum se adequam perfeitamente às demandas direcionadas ao ffuxo de sustentação [23].

1:100	Aquino e Dantas [23] enfatizam a importância de se administrar bem a…	Aquino e Dantas [23] enfatizam a importância de se administrar bem a mudança visto que equipes distintas trabalham com prazos e processos diferentes para manutenir evolutivamente e corretivamente o mesmo sistema e para atender e essa alta necessidade de gerência de conffguração e mudança o Impress incorpora padrões amplamente reconhecidos como Release Line, Release-Prep Codeline, Active Development Line e Integration build, e se inspira fortemente no modelo de gerência de conffguração Gitffow.
1:101	As empresas de software estão cada vez mais adotando Kanban para manut…	As empresas de software estão cada vez mais adotando Kanban para manutenção pois trata-se de uma ferramenta que se affrmou depois de demonstrar bons resultados, por oferecer visibilidade aprimorada do projeto, qualidade no software, motivação na equipe, melhor comunicação e colaboração [73].
1:104	Ahmad et al. [73] concluíram que, em resumo, para lidar com desaffo…	Ahmad et al. [73] concluíram que, em resumo, para lidar com desaffo da falta de visibilidade, o Kanban é mais adequado sendo indicado para onde exista um alto grau de variabilidade da prioridade como na manutenção em ciclos curtos. Nesse sentido, Kanban contribuiu para benefícios como visibilidade e priorização das tarefas, melhoria na moral da equipe, melhoria na comunicação, proteção das equipes quanto à exigências excessivas e compartilhamento de conhecimento entre os envolvidos.
1:106	Alaidaros et al. [74] propuseram melhorias no processo utilizando Kan…	Alaidaros et al. [74] propuseram melhorias no processo utilizando Kanban que estendem o rastreamento do progresso, gera limites ótimos de WIP e maior visão do ffuxo de trabalho. Para isso utilizaram métodos tradicionais de gerência de projetos, e algoritmos otimizados para determinação do melhor WIP
1:107	Eles destacaram que muitos profissionais estão utilizando métodos ágei…	Eles destacaram que muitos proffssionais estão utilizando métodos ágeis misturados inclusive com outras técnicas como as de gerência de projetos tradicionais ou métodos ágeis com outros métodos ágeis, como o método que une Kanban e Scrum formando o Scrumban [75]
1:109	Para tratar a urgência das manutenções corretivas, Heeager e Rose [26]…	Para tratar a urgência das manutenções corretivas, Heeager e Rose [26] sugerem que os projetos evolutivos sejam planejados com um tempo extra para que, caso ocorra alguma emergência, seja possível atendê-la sem muitos impactos na manutenção evolutiva em curso.
1:112	O modelo se incia com o planejamento para rastreamento de alterações n…	O modelo se incia com o planejamento para rastreamento de alterações no sistema e logo após veriffca-se o tipo de manutenção sendo que a manutenção corretiva é prioritária e o trabalho é iniciado por meio de uma nova versão de código [3].
1:113	Caso a manutenção seja evolutiva surge a necessidade de um monitoramen…	Caso a manutenção seja evolutiva surge a necessidade de um monitoramento para veriffcação de sprints urgentes e de emergência durante sua execução e caso ocorra a emergência, a sprint atual é pausada, o trabalho realizado é armazenado no sistema de controle de versão e é iniciada a manutenção corretiva [3].
1:114	Planeje a manutenção levando em consideração situações urgentes e não…	Planeje a manutenção levando em consideração situações urgentes e não urgentes. Não é necessário fazer todas as iterações com a mesma duração (como estabelece o Scrum), dependendo da natureza do trabalho em manutenção, pequenas iterações devem ser permitidas. Controle cada iteração com uso de ferramentas de controle de versão;

1:115	Mantenha o cliente sempre presente para orientar sobre emergência e pr…	Mantenha o cliente sempre presente para orientar sobre emergência e prioridades.
1:116	Execute cada tarefa de manutenção, corretiva e evolutiva, de forma sep…	Execute cada tarefa de manutenção, corretiva e evolutiva, de forma separada e então quando estiverem estáveis junte-as ao sistema original.
1:117	Documente a iteração para manter a integridade do sistema utilizando-s…	Documente a iteração para manter a integridade do sistema utilizando-se de casos de uso ou casos de teste
1:121	Para Alqudah e Rozilawati [4] Scrumban é mais adaptável, especialmente…	Para Alqudah e Rozilawati [4] Scrumban é mais adaptável, especialmente quando há muitas alterações nos requisitos do usuário.
1:122	Kanban: quando os envolvidos não estão dispostos a atuarem com papéis…	Kanban: quando os envolvidos não estão dispostos a atuarem com papéis pré- definidos; equipes que precisam executar tarefas prioritárias em menos de uma semana, constantemente ou mesmo em questões de horas com entregas pequenas; em equipes que precisam diminuir o lead time (tempo de entrega), aumentar a qualidade e reduzir custos. Fonte [4]
1:123	Scrum: quando os envolvidos demonstram preferência na adoção dos métod…	Scrum: quando os envolvidos demonstram preferência na adoção dos métodos recomendados; quando estão dispostos a exercerem os papéis prescritos; para entregas maiores que duram uma ou mais de uma semana; situações que o tempo de espera pode ser maior; disseminação dos conhecimentos na equipe
1:124	Scrumban: as escolhas para sua composição se dão para cada caso partic…	Scrumban: as escolhas para sua composição se dão para cada caso particular; Por exemplo, poderia iniciar o projeto adotando o Scrum evitando sprints, planejamento, revisão e backlog da sprint quando a tarefa é pequena para caber em mais de uma sprint ou quando a estimativa do tamanho da sprints é difícil e então poderiam controlar a entrega ajustando o WIP. Da mesma forma, podem existir muitos cenários e aplicações diferentes, ficando a cargo da equipe ajustar método Scrumban para cada situação. Para isso deve-se entender as práticas de ambos os métodos para poder selecionar as práticas. Fonte [4]
1:125	São características do software que podem facilitar a manutenção quan…	São características do software que podem facilitar a manutenção quando forem necessárias e são divididas em 5 características [125]: • modularidade; • reutilização; • capacidade de ser analisado; • capacidade de ser modificado; • capacidade de ser testado.
1:132	Um pipeline utiliza de automação das fases de desenvolvimento do soft…	Um pipeline utiliza de automação das fases de desenvolvimento do software para entregas rápidas, contínuas e seguras (recuperação de código). Além disso, possibilita integração de códigos promovendo o trabalho em equipe, monitoramento e testes.
1:133	O movimento DevOps 1 adaptou essas práticas e adicionou o pipeline de…	O movimento DevOps adaptou essas práticas e adicionou o pipeline de deployment como um dos principais requisitos para automatizar o processo de desenvolvimento de software melhorando a flexibilidade e facilidade de manutenção dos sistemas [82].
1:135	A utilização da automatização por meio de CI/CD é considerada essencia…	A utilização da automatização por meio de CI/CD é considerada essencial para emprego dos métodos ágeis, como bem expõem Vassallo et al. [86].

1:136	Laukkanen et al. [87] realizaram uma revisão da literatura em que consta…	Laukkanen et al. [87] realizaram uma revisão da literatura em que constatou sete principais problemas para emprego de CI/CD, cinco deles estão relacionados às diferentes atividades de desenvolvimento de software: projeto da arquitetura, projeto do sistema, integração, testes e entregas. Os outros dois não estão conectados a nenhuma parte individual e são: fatores humanos/organizacionais e de falta de recursos. A maioria dos problemas se concentram na integração e nos testes.
1:137	No entanto o momento de unir esses códigos (conhecido como merge day)…	No entanto o momento de unir esses códigos (conhecido como,merge day) pode ser trabalhoso e demorado. Isso acontece porque podem ocorrer conflitos entre a codificação de um desenvolvedor e a codificação de outro. Esse problema pode ser agravado se cada desenvolvedor tiver seu próprio ambiente de desenvolvimento local. O ideal seria que a equipe trabalhasse em um mesmo ambiente compartilhado baseado em nuvem [88]
1:138	Evitar situações que possam prejudicar o andamento do CI é fundamental…	Evitar situações que possam prejudicar o andamento do CI é fundamental. Quando CI não funciona apropriadamente e falham, causa uma perda de tempo e desvio de foco dos desenvolvedores, diminuindo a produtividade [87]
1:139	Construir sistemas com capacidade para suportar integrações frequente…	Construir sistemas com capacidade para suportar integrações frequentes é um pré-requisito para *continuous integration* e suas práticas [89].
1:140	Vários testes automatizados, geralmente de unidade e integração, são…	Vários testes automatizados, geralmente de unidade e integração, são feitos para garantir que as mudanças não corrompam a aplicação. A CI facilita a correção de qualquer erro encontrado[88].
1:143	Uma arquitetura monolítica só é capaz de oferecer autonomia lógica. P…	Uma arquitetura monolítica só é capaz de oferecer autonomia lógica. Para se alcançar maior flexibilidade arquitetural, uma proposta é utilizar um padrão ou estilo arquitetônico de microsserviços [82].
1:144	Bass et al. [61] resumem a ideia do que é um monólito de software e cha…	Bass et al. [61] resumem a ideia do que é um monólito de software e chama atenção para ideia da quebra desse monólito em partes menores que resulta na deffnição de microsserviços.
1:146	Após a codiffcação o desenvolvedor pode embutir o código no Docker e d…	Após a codiffcação o desenvolvedor pode embutir o código no Docker e distribuir em um cluster aproveitando de seu tamanho pequeno e leveza. Isso permite alcançar repetibilidade e uniformidade perfeitas nos ambientes [83]
1:147	Para que todos *containers* comuniquem entre si e também com outros serv…	Para que todos *containers* comuniquem entre si e também com outros servidores é necessário uma estrutura capaz de realizar a orquestração entre eles. Esta estrutura é chamada de orquestrador sendo o mais comum na comunidade de software livre o Kubernates [96].
1:148	Além de prover a orquestração dos componentes, o Kubernates possibilit…	Além de prover a orquestração dos componentes, o Kubernates possibilita uma série de outras funcionalidades como por exemplo: escalabilidade horizontal, alta disponiblidade, rápidas e transparentes atualizações do software, computação distribuída, entre outros [96].
1:152	Para se adequar à dinâmica dos métodos ágeis, os analistas de requisit…	Para se adequar à dinâmica dos métodos ágeis, os analistas de requisitos propuseram uma documentação mais simples que fosse clara, objetiva e adaptável, dando origem ao conceito de living documentation, ou documentação viva [15].
1:153	Schön (2017), citado por Fraga e Barbosa[101], identiffcou que os prin…	Schön (2017), citado por Fraga e Barbosa[101], identiffcou que os principais documentos de requisitos utilizados nos métodos ágeis são: histórias de usuário, protótipos, casos de uso e cartões de histórias
1:154	alguns autores (Haugset e Stlhane, Mead, Schön), conforme Fraga e B…	alguns autores (Haugset e Stlhane, Mead, Schön), conforme Fraga e Barbosa [10], indicaram também que os requisitos podem estar documentados nos casos de testes.

1:155	Embora o uso de requisitos ágeis gerem ganho inicial de tempo, a ausên…	Embora o uso de requisitos ágeis gerem ganho inicial de tempo, a ausência de detalhes na documentação podem resultar em não alinhamento com as expectativas do cliente [15].
1:159	Embora o Manifesto Ágil [63] preze mais pelo software em funcionamento…	Embora o Manifesto Ágil [63] preze mais pelo software em funcionamento do que uma documentação abrangente, ele não se opõe a qualquer prática adicional que vise melhorar a qualidade.
1:161	Os métodos ágeis possuem poucos papéis definidos e geralmente o time é…	Os métodos ágeis possuem poucos papéis definidos e geralmente o time é responsável desde a concepção do produto até a entrega não existindo o papel do testador[106].
1:162	Salerno et al. [106] discutiram o impacto de se ter um proffssional do t…	Salerno et al. [106] discutiram o impacto de se ter um profissional do time como facilitador na identificação dos problemas e para auxiliar no melhor entendimento das necessidades e engajamento do usuário, a este profissional deu-se o nome de Product Designer que se mostrou crucial na identificação de melhores soluções aos problemas dos usuários.
1:163	Proffssionais de qualidade, também chamados de QA (Quality Assurance)…	Profissionais de qualidade, também chamados de QA (Quality Assurance) ou testers passam a exercer papel dentro do time e geralmente não desperdiçam tempo escrevendo documentação para depois fazer testes manuais. Normalmente eles levantam os requisitos junto ao profossional de negócio de modo que a especificação se torna o próprio teste a medida que os desenvolvedores vão entregando pequenos incrementos facilmente testáveis [105].
1:164	Muitos métodos de testes surgiram e alguns ganharam forças de modo a s…	Muitos métodos de testes surgiram e alguns ganharam forças de modo a sustentar os processo baseados nos ágeis. Entre elas, vale ressaltar os conceitos de (TDD), Acceptance Test Driven Development (ATDD), Behavior-Driven Development (BDD) que utilizam testes unitários para entregar funcionalidades em um processo ágil [107].
1:165	A execução manual de um caso de teste é rápida e efetiva, mas a execuç…	A execução manual de um caso de teste é rápida e efetiva, mas a execução, atualização e repetição de um número elevado de testes é uma tarefa cansativa que com o tempo se torna inviável. Desse modo, é comum que os testadores não executem todos os casos a cada mudança e daí surgem os erros, levando a prejuízo para todos os envolvidos e diminuindo a qualidade das entregas[129].
1:167	Cada modiffcação ou correção de um erro resulta em muito esforço para…	Cada modificação ou correção de um erro resulta em muito esforço para execução de todos testes manuais e caso não seja feito poderá acarretar erros de regressão, que são aqueles erros que afetam outros módulos do sistema que estavam funcionando antes da manutenção, e assim entrar num ciclo de manutenção tornando uma tarefa tão difícil que passa a valer a pena reconstruir o software novamente [129].
1:169	Uma prácia comum é iniciar a criação de testes quando os requisitos es…	Uma prácia comum é iniciar a criação de testes quando os requisitos estão estáveis sem solicitação de grandes mudanças e após a execução dos testes manuais. As equipes que possuem membros que adotam papel de testador e desenvolvedor em momentos diferentes não costuma apresentar testes automatizados visto que não sobra tempo para tantas tarefas [120].
1:170	Duas estratégias para automatização de testes foram identiffcada no tr…	Duas estratégias para automatização de testes foram identiffcada no trabalho de Alves [120]: começar com as partes mais simples ou, pensando no custo com a manutenção na automação, aqueles testes de maior frequência

1:171	Entretanto, construir uma suíte de testes automatizados, requer padron…	Entretanto, construir uma suíte de testes automatizados, requer padronização de atividades, arquitetura que suporte os testes e conhecimento em codificação e análise por parte do testador. Para fazer testes de unidade, é necessário entender sobre o código, se for teste de sistema deve-se conhecer as regras de negócio e os resultados esperados. Além disso, conhecer bem os cenários evita a construção de casos de testes repetidos e quais testes devem ser de fato automatizados pois o trabalho para mantê-los atualizados já é bastante grande [120].
1:172	Através de métricas é possível identiffcar se os testes estão sendo va…	Através de métricas é possível identificar se os testes estão sendo vantajosos. Uma das formas mais utilizadas é a contagem de defeitos obtidos em desenvolvimento, homologação e produção [131].
3:14	ficou mais visível.	ficou mais visível.
3:15	visibilidade grande no trabalho que a gente	visibilidade grande no trabalho que a gente
3:19	ter um tempo ali, organizar essas demandas	ter um tempo ali, organizar essas demandas
3:22	a participação do usuário gestor é muito importante.	a participação do usuário gestor é muito importante.
3:23	aquela reunião de apresentação, e ali ele acaba participando também,	aquela reunião de apresentação, e ali ele acaba participando também,
3:24	poder abrir uma vídeo chamada e o cara participar sem sair da mesa del…	poder abrir uma vídeo chamada e o cara participar sem sair da mesa dele.
3:25	Então isso aí ajudou demais, ajuda demais. O cara ver o que está produz…	Então isso aí ajudou demais, ajuda demais. O cara ver o que está produzindo, falar que tem erro, que não é assim tal eu acho muito importante.
3:26	É de suma importância esses prazos aí. Até ajudou a organizar o trabal…	É de suma importância esses prazos aí. Até ajudou a organizar o trabalho a gente se organiza melhor. Porque antes não tinha condição de dar prazos, quem definia o prazo era o cara que estava pedindo ele e todo mundo que pede quer a coisa daqui dois, três dias, coisa impossível assim.
3:27	Tem que ter prazos, e mesmo que ele seja assim, vamos definir o prazo…	Tem que ter prazos, e mesmo que ele seja assim, vamos definir o prazo ali do início sem saber na verdade quanto tempo vai levar exatamente. Tem que ter um prazo.
3:28	A gente começa pela reunião de planejamento, marca para próxima seman…	A gente começa pela reunião de planejamento, marca para próxima semana para falar o que a gente produziu, eu acho interessante. Tem que cumprir esses prazos.
3:29	seguir os prazos que foram definidos, prazos das reuniões, dos encontr…	seguir os prazos que foram definidos, prazos das reuniões, dos encontros eu achei bem interessante.
4:26	eu acho que é importante um cara ter um pouco de domínio de um determi…	eu acho que é importante um cara ter um pouco de domínio de um determinado módulo, ou seja, o sistema dependendo do tamanho, pode até ter do sistema todo, do que ele não ter o controle
4:30	manutenção evolutiva, talvez quem tem mais dormindo sistema faça mais…	manutenção evolutiva, talvez quem tem mais dormindo sistema faça mais rápido,
9:6	Na verdade eu acho que tem que ser assim, tem que chegar alguém que v…	Na verdade eu acho que tem que ser assim, tem que chegar alguém que vai dizer que vai ser assim e pronto. Talvez o que não tenha funcionado é a interação entre quem impôs e a equipe.
11:4	Sobre aquele relatório de acompanhamento que era disponibilizado ao fi…	Sobre aquele relatório de acompanhamento que era disponibilizado ao final de toda reunião de acompanhamento, né? Você viu importância nele, ou não? R: É relevante porque para ter noção a questão da transparência, de saber o que tá acontecendo. Eu acho extremamente válido.
12:4	Então, nesse sentido na gestão passada nós criamos o comitê de govern…	Então, nesse sentido na gestão passada nós criamos o comitê de governança de TI, que uma das competências é analisar todas as demandas de alteração de sistemas do tribunal, se são pertinentes ou não e se sim, qual a ordem de prioridade delas.

12:5	Conseguimos também aprovar lá no órgão especial uma resolução que tra…	Conseguimos também aprovar lá no órgão especial uma resolução que trata, que estabelece a política de desenvolvimento e sustentação de software no âmbito do tribunal de Justiça de Goiás , que dá essas diretrizes sobre o procedimento e cria formalmente em nome do tribunal essas figuras do: demandante, do levantador de requisitos, dos desenvolvedores, mostrando que algo multidisciplinar.

Categoria: Arquitetura

ID	Nome da Citação	Conteúdo da Citação
1:79	TJGO, que apesar de dispor de infraestrutura física e de ferramentas…	TJGO, que apesar de dispor de infraestrutura física e de ferramentas modernas para o desenvolvimento
1:114	Planeje a manutenção levando em consideração situações urgentes e não…	Planeje a manutenção levando em consideração situações urgentes e não urgentes. Não é necessário fazer todas as iterações com a mesma duração (como estabelece o Scrum), dependendo da natureza do trabalho em manutenção, pequenas iterações menores devem ser permitidas. Controle cada iteração com uso de ferramentas de controle de versão;
1:116	Execute cada tarefa de manutenção, corretiva e evolutiva, de forma sep…	Execute cada tarefa de manutenção, corretiva e evolutiva, de forma separada e então quando estiverem estáveis junte-as ao sistema original.
1:126	O modelo Mantus contém seis processos e uma série de práticas que são…	O modelo Mantus contém seis processos e uma série de práticas que são definidas para cada processo sendo eles: de requisitos de software, projeto, arquitetura, construção, avaliação e gestão da documentação cada um deles tenta maximizar as sub-características de manutenção descritas na norma ISO/IEC 25010 [5].
1:127	Para suportar os métodos ágeis para manutenção, de acordo com a realid…	Para suportar os métodos ágeis para manutenção, de acordo com a realidade de cada instituição, são necessários o uso de ferramentas para suportar com agilidade necessária as manutenções emergenciais, dado que a excelência técnica e bom design aumentam a agilidade [28]
1:128	A capacidade de um sistema de software produzir resultados corretos n…	A capacidade de um sistema de software produzir resultados corretos não é útil se levar muito tempo para fazer isso ou se o sistema não permanecer ativo tempo suficiente para entregá-lo, ou mesmo se o sistema revelar os resultados para pessoas indesejadas. Nos requisitos arquiteturais é onde estas preocupações são abordadas [7]
1:129	Sistemas de software são construídos para satisfazer os objetivos de n…	Sistemas de software são construídos para satisfazer os objetivos de negócios das organizações. A arquitetura é a ponte entre estes objetivos e os resultados finais do sistema [57].
1:130	Os requisitos não funcionais são também chamados de requisitos de atri…	Os requisitos não funcionais são também chamados de requisitos de atributo de qualidade e envolvem performance, manutenibilidade, disponibilidade, usabilidade, escalabilidade, segurança e outros. Eles podem definir a arquitetura, designs patterns a serem utilizados e decisões nos projetos [57]
1:131	Uma alocação bem realizada possibilita que os requisitos expressos no…	Uma alocação bem realizada possibilita que os requisitos expressos no software sejam satisfeitos pelas características de hardware [7].
1:132	Um pipeline utiliza de automação das fases de desenvolvimento do soft…	Um pipeline utiliza de automação das fases de desenvolvimento do software para entregas rápidas, contínuas e seguras (recuperação de código). Além disso, possibilita integração de códigos promovendo o trabalho em equipe, monitoramento e testes.
1:133	O movimento DevOps 1 adaptou essas práticas e adicionou o pipeline de…	O movimento DevOps adaptou essas práticas e adicionou o pipeline de deployment como um dos principais requisitos para automatizar o processo de desenvolvimento de software melhorando a fiexibilidade e facilidade de manutenção dos sistemas [78].

1:134	Embora os métodos ágeis sejam cada vez mais comuns, muitas Organizaçõe…	Embora os métodos ágeis sejam cada vez mais comuns, muitas Organizações não conseguiram alcançar os benefícios de CI/CD devido ao distanciamento entre equipes de desenvolvimento e infraestrutura, fato motivador para que o DevOps se desenvolvesse [80]
1:135	A utilização da automatização por meio de CI/CD é considerada essencia…	A utilização da automatização por meio de CI/CD é considerada essencial para emprego dos métodos ágeis, como bem expõem Vassallo et al. [80].
1:136	Laukkanen et al. [82] realizaram uma revisão da literatura em que consta…	Laukkanen et al. [82] realizaram uma revisão da literatura em que constatou sete principais problemas para emprego de CI/CD, cinco deles estão relacionados às diferentes atividades de desenvolvimento de software: projeto da arquitetura, projeto do sistema, integração, testes e entregas. Os outros dois não estão conectados a nenhuma parte individual e são: fatores humanos/organizacionais e de falta de recursos. A maioria dos problemas se concentram na integração e nos testes.
1:137	No entanto o momento de unir esses códigos (conhecido como merge day)…	No entanto o momento de unir esses códigos (conhecido como merge day) pode ser trabalhoso e demorado. Isso acontece porque podem ocorrer confiitos entre a codificação de um desenvolvedor e a codificação de outro. Esse problema pode ser agravado se cada desenvolvedor tiver seu próprio ambiente de desenvolvimento local. O ideal seria que a equipe trabalhasse em um mesmo ambiente compartilhado baseado em nuvem [83]
1:138	Evitar situações que possam prejudicar o andamento do CI é fundamental…	Evitar situações que possam prejudicar o andamento do CI é fundamental. Quando CI não funciona apropriadamente e falham, causa uma perda de tempo e desvio de foco dos desenvolvedores, diminuindo a produtividade [82]
1:139	Construir sistemas com capacidade para suportar integrações frequente…	Construir sistemas com capacidade para suportar integrações frequentes é um pré-requisito para continuous integration e suas práticas [84].
1:141	Embora existam modelos prescritivos, na vida real a adoção do CD requ…	Embora existam modelos prescritivos, na vida real a adoção do CD requer iterações e ações específicas para cada caso[82]
1:142	Normalmente espera-se que uma arquitetura de CD assegure poder comput…	Normalmente espera-se que uma arquitetura de CD assegure poder computacional elástico e alta capacidade de monitoramento para suportar as mudanças frequentes nos requisitos, fato que gera implicações financeiras devido a dificuldade em se mensurar a quantidade de recursos necessários [85].
1:143	Uma arquitetura monolítica só é capaz de oferecer autonomia lógica. P…	Uma arquitetura monolítica só é capaz de oferecer autonomia lógica. Para se alcançar maior fiexibilidade arquitetural, uma proposta é utilizar um padrão ou estilo arquitetônico de microsserviços [78].
1:144	Bass et al. [57] resumem a ideia do que é um monólito de software e cha…	Bass et al. [57] resumem a ideia do que é um monólito de software e chama atenção para ideia da quebra desse monólito em partes menores que resulta na definição de microsserviços.
1:147	Para que todos *containers* comuniquem entre si e também com outros serv…	Para que todos *containers* comuniquem entre si e também com outros servidores é necessário uma estrutura capaz de realizar a orquestração entre eles. Esta estrutura é chamada de orquestrador sendo o mais comum na comunidade de software livre o Kubernates [92].
1:148	Além de prover a orquestração dos componentes, o Kubernates possibilit…	Além de prover a orquestração dos componentes, o Kubernates possibilita uma série de outras funcionalidades como por exemplo: escalabilidade horizontal, alta disponibilidade, rápidas e transparentes atualizações do software, computação distribuída, entre outros [92].

1:171	Entretanto, construir uma suíte de testes automatizados, requer padron…	Entretanto, construir uma suíte de testes automatizados, requer padronização de atividades, arquitetura que suporte os testes e conhecimento em codificação e análise por parte do testador. Para fazer testes de unidade, é necessário entender sobre o código, se for teste de sistema deve-se conhecer as regras de negócio e os resultados esperados. Além disso, conhecer bem os cenários evita a construção de casos de testes repetidos e quais testes devem ser de fato automatizados pois o trabalho para mantê-los atualizados já é bastante grande [113].
3:35	E: A gente ter usado controle de versão, pipeline, escalabilidade…23:3…	E: A gente ter usado controle de versão, pipeline, escalabilidade… R: Ah não... isso aí é quase mil por cento [risos]. Isso aí ficou bom demais. Isso aí foi uma coisa que… no começo a gente vinha usando um controle de versão mais "malê-má" (mais ou menos) também. Mas essa, que eles chamam de DevOps aí, que envolve toda essa infraestrutura de entrega do software ficou excelente. Isso aí melhorou, para mim, melhorou muito. Isso aí teria que ter... foi a "chave de ouro" para mim [risos]. Ficou muito fácil para gente disponibilizar um software
3:37	Quando a gente começou a trabalhar com essa ideia de entrega contínua,…	Quando a gente começou a trabalhar com essa ideia de entrega contínua, isso aí aproxima também as equipes. Por exemplo, o pessoal da infra acaba que agora está sabendo o que estamos fazendo. Tipo ver um software ou olham uma telinha e falam que determinado software está muito lento. Antigamente não tinha isso, então melhorou muito
4:33	E: Quanto ao a forma de trabalho, com relação à arquitetura, como era…	E: Quanto ao a forma de trabalho, com relação à arquitetura, como era antes e como é hoje ? R: Teve uma mudança boa viu. Eu acho que isso foi uma evolução necessária boa mesmo.
8:7	é uma das formas que tem de dar um retorno para a sociedade de forma…	é uma das formas que tem de dar um retorno para a sociedade de forma na prestação jurisdicional de forma ágil, célere e que atenda a questão das publicidades dos atos é um investimento em tecnologia, em TI e em projetos que realmente possa ter essa função principalmente de trazermos ferramentas que vá realmente otimizar os serviços do Judiciário
9:11	O processo era um processo e inicialmente era bem arcaico. Primeiro qu…	O processo era um processo e inicialmente era bem arcaico. Primeiro que você tinha um acesso direto. Que em alguns pontos dava aquela falsa impressão de praticidade, mas era bem perigoso. Você tinha acesso direto praticamente a produção direto, você desenvolvia sem passar por ninguém, já fazia seu deploy e estava em produção. Não passava por ninguém e era uma forma inicialmente, uma forma de quando entrei no tribunal, uma forma de copiar os arquivos pra área onde seria sincronizada e acabou, era simplesmente isso.
9:12	Depois que foi implementado o controle de versão. Nem isso tinha ante…	Depois que foi implementado o controle de versão. Nem isso tinha antes e já foi um grande avanço o que melhorou muito muitas coisas e agora a gente está com um processo mais adequado.

9:13	Primeiro ficou muito mais fácil de trabalhar em equipe mesmo. Você sa…	Primeiro ficou muito mais fácil de trabalhar em equipe mesmo. Você sabe o que cada um fez e não conflita trabalho, você tem menos trabalho também, trabalho de mostrar o que você fez na forma do commit e subir. Segue o padrão, não é você que vai subir, tem um processo de quem vai fazer isso. Como vai chegar a atividade para você e de como você vai passar essa atividade para frente e isso foi uma evolução enorme que teve no TJ e até comentei isso no questionário do ano passado que a gente realmente nunca teve um padrão de verdade que não fosse aquele tentado em 2015, naquela época... mas, que abrangesse mais o processo todo. Desde como as coisas chegam para você. De como você divide o que vai ser feito. O processo ficou completo agora.
11:2	Eu achei bacana também a forma de acompanhamento. Fiz os testes. Você…	Eu achei bacana também a forma de acompanhamento. Fiz os testes. Você perguntou dos testes. Todos os testes que me foram passados eu fiz. Inclusive, depois que teve algum erro eu entrei em contato e tive o suporte necessário então foi tranquilo
11:3	isso foi extremamente importante porque no início a gente tinha pensa…	isso foi extremamente importante porque no início a gente tinha pensado de um jeito e acabou mudando para outra forma, que a gente viu no teste e que precisava ser de outra forma.
12:10	Olha com certeza evoluiu e muito, evoluiu muito em termos do sistema,…	Olha com certeza evoluiu e muito, evoluiu muito em termos do sistema, da infra, da metodologia de trabalho e de gestão da equipe e gestão do projeto. Hoje as entregas são previsíveis, existe previsibilidade das entregas, o que será feito em qual versão e a estimativa de tempo, com um sistema escalonável agora, de acordo com a demanda, então com certeza a evolução foi muito grande num curto espaço de tempo.

Categoria: Vertentes da manutenção

ID	Nome da Citação	Conteúdo de Citação
1:12	Estudos enfatizam a importância de se considerar as subcategorias da m…	Estudos enfatizam a importância de se considerar as subcategorias da manutenção. Fonte [3,22,23]
1:13	Quando os modelos tratam a manutenção de forma geral sem levar em cont…	Quando os modelos tratam a manutenção de forma geral sem levar em conta as subcategorias (corretiva, adaptativa, evolutiva) acabam não gerando conhecimento profundo das peculiaridades de cada subcategoria. Fonte [32]
1:14	O gerenciamento abrangente e complexo utilizando o mesmo modelo de pro…	O gerenciamento abrangente e complexo utilizando o mesmo modelo de processo leva a incertezas e muitas perguntas sem respostas. As categorias de manutenção diferem muito para serem agrupados sob um mesmo modelo de processo. Uma forma de contornar essa situação é construindo modelos de processos dedicados para cada categoria de manutenção (modelos especializados). Fonte [32]
1:15	Ao concentrar esforços em apenas um domínio da manutenção torna-se pos…	Ao concentrar esforços em apenas um domínio da manutenção torna-se possível realizar uma análise minuciosa estabelecendo bem o escopo do desenvolvimento e da manutenção. Fonte [32]
1:16	o processo de gerenciamento de problemas de software é dominante na m…	o processo de gerenciamento de problemas de software é dominante na manutenção corretiva. Fonte [32]
1:18	pode ser usado para manutenção durante situações de emergência, recup…	pode ser usado para manutenção durante situações de emergência, recuperação pós-emergência e ciclo normal de desenvolvimento do sistema. Fonte [24]
1:23	Estado de emergência	Estado de emergência. Fonte [24]
1:24	Posterior a emergência	Posterior a emergência. Fonte [24]
1:25	Estado de evolução normal	Estado de evolução normal. Fonte [24]
1:68	A aplicação de ágeis no Governo brasileiro é, em sua maioria, bem-suce…	A aplicação de ágeis no Governo brasileiro é, em sua maioria, bem-sucedida e, normalmente, conduzida em combinação com outras abordagens ágeis de desenvolvimento de software ou, em sua maioria, uma combinação de ágeis com métodos tradicionais. Fonte [1]
1:87	Entretanto, estudos apontam que a manutenção possui características p…	Entretanto, estudos apontam que a manutenção possui características próprias e muitas vezes a inobservância dessas diferenças podem gerar ainda mais manutenções [26, 33, 31, 3, 64]
1:92	Essa união bem sucedida deu origem ao Mantema que divide a manutenção…	Essa união bem sucedida deu origem ao Mantema que divide a manutenção em cinco tipos: Corretiva e urgente, Corretiva sem urgência, Evolutiva, Preventiva e Adaptativa. Fonte [22]
1:97	O Impress sustenta que demandas diferentes devem ser tratadas através…	O Impress sustenta que demandas diferentes devem ser tratadas através de estratégias de manutenções diferentes, de modo a respeitar suas peculiaridades quanto as definições de escopo, tempo, partes envolvidas, complexidade, urgência e tamanho das solicitações. Assim, as diferentes demandas são divididas em duas categoria: Sustentação e Evolução [23]
1:108	Heeager e Rose [26] revelou os desafios de se adotar Scrum para proces…	Heeager e Rose [26] revelou os desafios de se adotar Scrum para processos de manutenção. Eles relataram que muitos dos problemas de se aplicar Scrum para manutenção se dá ao fato de que uma emergência não pode aguardar o cumprimento de uma sprint com todos seus requisitos e quando acontecem acabam atrasando o cumprimento das metas da sprint. Assim afirmaram que a aplicação dos métodos ágeis na manutenção podem não ocorrer automaticamente e algumas otimizações podem ser necessárias para acomodar os diferentes tipos de manutenção, diferentes fontes de trabalho e práticas locais.

1:110	Pino et al. [72] aconselham o uso de sprints separadas para a manutenç…	Pino et al. [72] aconselham o uso de sprints separadas para a manutenção, sendo uma mais breve para correções de emergências imprevistas e outra maior com planejamento para manutenções evolutivas.
1:111	Verificando que a literatura sugere o tratamento diferenciado para as…	Verificando que a literatura sugere o tratamento diferenciado para as diferentes categorias de manutenção (adaptativa, corretiva, perfectiva e preventiva), Rehman et al. [3] constataram que quando isso é atendido os resultados da manutenção com métodos ágeis aumentam os casos de sucesso.
1:116	Execute cada tarefa de manutenção, corretiva e evolutiva, de forma sep…	Execute cada tarefa de manutenção, corretiva e evolutiva, de forma separada e então quando estiverem estáveis junte-as ao sistema original.
1:119	Scrumban [75] é o resultado da união entre Scrum e Kanban com propósit…	Scrumban [75] é o resultado da união entre Scrum e Kanban com propósito de se aproveitar o melhor de cada um em prol de uma maior fiexibilização do emprego das práticas ágeis em diferentes situações de projeto.
1:120	Alqudah e Rozilawati [4] os profissionais de Kanban e Scrum estão con…	Alqudah e Rozilawati [4] os profissionais de Kanban e Scrum estão convencidos de que para certas situações a combinação de ambos os métodos é melhor que o uso de um ou de outro isoladamente entretanto não é fácil selecionar as características que serão utilizadas de cada um e para se escolher Scrum, Kanban ou ambos exige-se um entendimento dos fatores que infiuenciam a seleção ou hibridização.
4:12	Eu acho que sempre foi uma dificuldade o legado, a manutenção. Então e…	Eu acho que sempre foi uma dificuldade o legado, a manutenção. Então eu acho que uma das coisas que sempre dificultava, o início de um processo definido, com início, meio e fim [pensando]... foi essa manutenção, a necessidade de manutenção dos sistemas. Porque manutenção dos sistemas exigia dedicação de um outro, quase uma dedicação exclusiva.
4:13	Os de processo judicial sempre foi - apesar de sempre ter coisas nova…	Os de processo judicial sempre foi - apesar de sempre ter coisas novas - era meio que um sistema único então parecia como se fosse um incremento, uma nova funcionalidade, sempre uma nova funcionalidade,
4:14	E: Sempre uma manutenção evolutiva né ? R: É… e não encarava as…	E: Sempre uma manutenção evolutiva né ? R: É… e não encarava assim também né ? Como : agora vou pegar uma alteração e encarar uma alteração ou uma nova funcionalidade como um novo projeto. Encarava como manutenção evolutiva mesmo, como se: “eu vou corrigir isso aqui, isso aqui e fazer isso aqui a mais
8:6	Essa aí foi outra forma de que nós pudéssemos falar e realmente essa d…	Essa aí foi outra forma de que nós pudéssemos falar e realmente essa demanda que nós temos, em termos de necessidade, pudesse chegar até a parte operacional, na parte de execução da área de TI. Realmente trouxe um resultado muito bom. Inclusive, já encaminho, normalmente já trata, e já dá o retorno. Todas as demandas que eu solicitei por esse mecanismo, por essa ferramenta (que foi através do e-mail), todos eles alcançamos os objetivos e principalmente aquilo que agente buscava, que era alguma implementação, alguma melhoria alguma situação de relevância e de extrema urgência, que realmente foram todos realizados dentro do tempo satisfatório.
11:5	E depois que a gente entregou o sistema aconteceram alguns erros, né?…	E depois que a gente entregou o sistema aconteceram alguns erros, né? Mesmo depois de entregues. Como foi resolvido esses erros para você. Você acha que... na sua experiência como foi? R: Foi rápido, foi prontamente.

12:10	Olha com certeza evoluiu e muito, evoluiu muito em termos do sistema,…	Olha com certeza evoluiu e muito, evoluiu muito em termos do sistema, da infra, da metodologia de trabalho e de gestão da equipe e gestão do projeto. Hoje as entregas são previsíveis, existe previsibilidade das entregas, o que será feito em qual versão e a estimativa de tempo, com um sistema escalonável agora, de acordo com a demanda, então com certeza a evolução foi muito grande num curto espaço de tempo.
12:11	As demandas, pelo menos as que passaram por mim, ou as quais eu tive c…	As demandas, pelo menos as que passaram por mim, ou as quais eu tive ciência, eu vejo que todas foram entregues num curto espaço de tempo. Realmente, a equipe conseguiu fazer essas correções de uma forma muito rápida inclusive algumas outras evoluções, então com certeza também definitivamente elas impactam no cronograma das ações planejadas, mas infelizmente todos ou quase todos os projetos a gente acaba tendo alguns imprevistos. Porque uma coisa é um erro de um sistema onde não inviabiliza trabalho outra é realmente é um erro que esteja inviabilizando o projeto, o software, né a solução e requer uma atualização emergencial

Categoria: Testes importam

ID	Nome da Citação	Conteúdo de Citação
1:5	os testes foram adicionados a todo ciclo de desenvolvimento do sistema…	os testes foram adicionados a todo ciclo de desenvolvimento do sistema e foram executados todas as vezes que se faziam alguma mudança. Para se automatizar o processo de testes foi estabelecido um projeto também sob o aspecto da melhoria contínua;
1:8	planejamento em detalhes a partir de histórias de usuários em período…	planejamento em detalhes a partir de histórias de usuários em períodos de duas semanas, desenvolvimento em pares, testes contínuos, e cliente próximo da equipe.
1:47	Redução da qualidade global do software como resultado de mudanças que…	Redução da qualidade global do software como resultado de mudanças que introduzem erros latentes no software mantido
1:118	A manutenção deve ser testável: testes de integração, regressão e tamb…	A manutenção deve ser testável: testes de integração, regressão e também vale para a manutenção corretiva urgente.
1:136	Laukkanen et al. [87] realizaram uma revisão da literatura em que consta…	Laukkanen et al. [87] realizaram uma revisão da literatura em que constatou sete principais problemas para emprego de CI/CD, cinco deles estão relacionados às diferentes atividades de desenvolvimento de software: projeto da arquitetura, projeto do sistema, DevOps uma prática de engenharia de software que busca unificar o desenvolvimento de software (Dev) e as operações de software(Ops)[84] integração, testes e entregas. Os outros dois não estão conectados a nenhuma parte individual e são: fatores humanos/organizacionais e de falta de recursos. A maioria dos problemas se concentram na integração e nos testes.
1:140	Vários testes automatizados, geralmente de unidade e integração, são…	Vários testes automatizados, geralmente de unidade e integração, são feitos para garantir que as mudanças não corrompam a aplicação. A CI facilita a correção de qualquer erro encontrado[88].
1:150	Para Ambler (2005) conforme Piovesan [131] considera que a qualidade ág…	Para Ambler (2005) conforme Piovesan [131] considera que a qualidade ágil desejada é resultado de várias práticas sendo elas: o desenvolvimento iterativo e incremental, técnicas de teste ágeis, automação de testes unitários e de regressão, métricas de codiffcação e o trabalho colaborativo.
1:163	Profissionais de qualidade, também chamados de QA (*Quality Assurance*)…	Profissionais de qualidade, também chamados de QA (*Quality Assurance*) ou *testers* passam a exercer papel dentro do time e geralmente não desperdiçam tempo escrevendo documentação para depois fazer testes manuais. Normalmente eles levantam os requisitos junto ao profissional de negócio de modo que a especificação se torna o próprio teste a medida que os desenvolvedores vão entregando pequenos incrementos facilmente testáveis [105].
1:164	Muitos métodos de testes surgiram e alguns ganharam forças de modo a s…	Muitos métodos de testes surgiram e alguns ganharam forças de modo a sustentar os processo baseados nos ágeis. Entre elas, vale ressaltar os conceitos de (TDD), *Acceptance Test Driven Development* (ATDD), *Behavior-Driven Development* (BDD) que utilizam testes unitários para entregar funcionalidades em um processo ágil [105].

1:165	A execução manual de um caso de teste é rápida e efetiva, mas a execuç…	A execução manual de um caso de teste é rápida e efetiva, mas a execução, atualização e repetição de um número elevado de testes é uma tarefa cansativa que com o tempo se torna inviável. Desse modo, é comum que os testadores não executem todos os casos a cada mudança e daí surgem os erros, levando a prejuízo para todos os envolvidos e diminuindo a qualidade das entregas[129].
1:166	Crispin e Gregory [64] existem alguns motivos para se querer automati…	Crispin e Gregory [64] existem alguns motivos para se querer automatizar testes, são eles: • Testes manuais são demorados; • Testes manuais são propensos a erros; • Liberar tempo para trabalhar melhor; • Gerar uma rede segura de testes; • Fornecer frequente e rápido feedback; • Codiffcação guiada por testes e exemplos podem ir mais longe; • Testes fornecem documentação; • Retorno sobre o investimento.
1:168	O momento para criação de testes automatizados varia de acordo com a e…	O momento para criação de testes automatizados varia de acordo com a experiência e conhecimento da equipe [120], muitas vezes a prioridade está na velocidade da entrega, mesmo que seja necessário abrir mão da qualidade e de outros atributos.
1:171	Entretanto, construir uma suíte de testes automatizados, requer padron…	Entretanto, construir uma suíte de testes automatizados, requer padronização de 60 atividades, arquitetura que suporte os testes e conhecimento em codificação e análise por parte do testador. Para fazer testes de unidade, é necessário entender sobre o código, se for teste de sistema deve-se conhecer as regras de negócio e os resultados esperados. Além disso, conhecer bem os cenários evita a construção de casos de testes repetidos e quais testes devem ser de fato automatizados pois o trabalho para mantê-los atualizados já é bastante grande [120].
3:30	do jeito que tá assim, por exemplo, iniciar o processo de teste de apl…	do jeito que tá assim, por exemplo, iniciar o processo de teste de aplicação aí que tem 3, 4 anos que ela tá aí eu acho que, teria que talvez a gente teria que, ter começado o teste ao contrário. Sei lá, assim, a gente queria começar a fazer os testes unitários ali para saber se o código está ok, aí eu não sei porque para funcionar o processo eu acho que tem que começar primeiro do teste.
3:31	A gente se desenvolve baseado em testes, primeiro você vai ali planeja…	A gente se desenvolve baseado em testes, primeiro você vai ali planejando seu teste tal e depois você cai na implementação. Agora testar uma coisa que já tá disponível, talvez, a gente teria que ter outra coisa
3:32	Eu acho que é importante fazer esses testes é mesmo que o desenvolvedo…	Eu acho que é importante fazer esses testes é mesmo que o desenvolvedor que está programando e vai testando ao mesmo tempo, vai testando a interface,
3:33	mas o papel do teste é importante assim teria que ter mesmo porque é…	mas o papel do teste é importante assim teria que ter mesmo porque é uma coisa muito cansativa também como a gente não tem o processo desenvolvimento iniciado voltado ao teste no caso dessas aplicações rodando, assim para quem tá desenvolvendo é uma coisa tediosa
3:34	do jeito que está hoje está quase compensando pedir para o usuário ges…	do jeito que está hoje está quase compensando pedir para o usuário gestor testar, falar para ele assim: “entra aí no ambiente e faz uns testes aí”
7:13	E: E sua opinião sobre os testes? Tem alguma? 9:15 R: Importantíssimo…	E: E sua opinião sobre os testes? Tem alguma? 9:15 R: Importantíssimos e uma coisa que eu acho assim que em alguns projetos lá a gente, meio que fica como lições aprendidas, é que às vezes a gente precisa caprichar mais um pouquinho mais no teste interno mesmo antes de passar para o usuário testar às vezes.
7:14	Eu acho que se a gente conseguir melhorar Esse aspecto vai trazer mais…	Eu acho que se a gente conseguir melhorar Esse aspecto vai trazer mais qualidade ainda para nossos entregas.

11:2	Eu achei bacana também a forma de acompanhamento. Fiz os testes. Você…	Eu achei bacana também a forma de acompanhamento. Fiz os testes. Você perguntou dos testes. Todos os testes que me foram passados eu fiz. Inclusive, depois que teve algum erro eu entrei em contato e tive o suporte necessário então foi tranquilo
11:3	isso foi extremamente importante porque no início a gente tinha pensa…	isso foi extremamente importante porque no início a gente tinha pensado de um jeito e acabou mudando para outra forma, que a gente viu no teste e que precisava ser de outra forma.

Categoria: O que se espera do processo

ID	Nome da Citação	Conteúdo da Citação
1:10	possibilitaram à Aveva melhorias na qualidade do código fonte, melhor…	possibilitaram à Aveva melhorias na qualidade do código fonte, melhoria na autoestima da equipe, maior visibilidade do progresso de manutenção, melhor comunicação e compartilhamento de conhecimento entre os envolvidos. Fonte [26]
1:11	os engenheiros de software se beneficiaram de um estrutura ordenada e…	os engenheiros de software se beneficiaram de um estrutura ordenada em seu trabalho oferecida pelas boas práticas. Fonte [28]
1:17	proporcionar visualização de todos os aspectos possíveis de um proces…	proporcionar visualização de todos os aspectos possíveis de um processo de manutenção. Fonte [32]
1:33	levam os envolvidos à aumentarem aspectos de desempenho, como maior e…	levam os envolvidos à aumentarem aspectos de desempenho, como maior engajamento, produtividade, criatividade e moral. Fonte [33]
1:40	detectou uma maior satisfação dos clientes, redução de custos, maior p…	detectou uma maior satisfação dos clientes, redução de custos, maior previsibilidade de custos e prazos, além do aumento da produtividade e qualidade. Fonte [35]
1:48	uma melhora perceptível na redução de defeitos provocados pelas manut…	uma melhora perceptível na redução de defeitos provocados pelas manutenções. Fonte [36]
1:64	apresenta que grande parte das melhorias propostas no MPS.BR podem se…	apresenta que grande parte das melhorias propostas no MPS.BR podem ser alcançadas utilizando-se Scrum: 67% dos resultados esperados pelo MPS.BR são atendidos pelo Scrum, 25% são parcialmente atendidos, contra apenas 8% dos resultados que não são atendidos. Fonte [54]
1:68	A aplicação de ágeis no Governo brasileiro é, em sua maioria, bem suce…	A aplicação de ágeis no Governo brasileiro é, em sua maioria, bem-sucedida e, normalmente, conduzida em combinação com outras abordagens ágeis de desenvolvimento de software ou, em sua maioria, uma combinação de ágeis com métodos tradicionais Fonte [1]
1:69	A escolha do Kanban se 13deu pela capacidade de adaptação às frequent…	A escolha do Kanban se deu pela capacidade de adaptação às frequentes mudanças nas requisições de manutenção de software pelo órgão junto ao fornecedor, prestador de serviços de manutenção. Fonte [59]
1:70	Os resultados permitiram constatar que o novo processo atenuou proble…	Os resultados permitiram constatar que o novo processo atenuou problemas, diminuiu a resistência as mudanças, e contribuiu para aumentar a satisfação dos envolvidos. Fonte [57]
1:73	Após a aplicação do modelo, constataram que os indicadores de desempe…	Após a aplicação do modelo, constataram que os indicadores de desempenho melhoraram, assim como a satisfação dos clientes. Fonte [58]
1:75	os autores, as motivações para a adoção de ágeis incluem: (1) uma ent…	os autores, as motivações para a adoção de ágeis incluem: (1) uma entrega rápida de valor ao cliente; (2) uma maior colaboração entre TI e negócios; (3) uma maior satisfação do cliente. Além disso, os autores [57] chamam a atenção para um fato novo: (4) aumento na moral da equipe de TI do governo, reduzindo a dependência de empresas contratadas. Fonte [58]

1:77	(1) comunicação entre os membros da equipe de desenvolvimento, bem com…	(1) comunicação entre os membros da equipe de desenvolvimento, bem como, entre desenvolvedores e clientes; (2) aprendizado de novas tecnologias; (3) qualidade do produto; (4) visibilidade do projeto; (5) produtividade das equipes; (6) redução de custos; (7) capacidade de gerenciar as mudanças e as prioridades; e (8) conformidade com exigências burocráticas do Governo. Fonte [58]
1:84	De acordo com April [38], para uma organização ser considerada madura…	De acordo com April [38], para uma organização ser considerada madura é necessário que ela tenha instituído o seu processo de manutenção de software.
1:134	Embora os métodos ágeis sejam cada vez mais comuns, muitas Organizaçõe…	Embora os métodos ágeis sejam cada vez mais comuns, muitas Organizações não conseguiram alcançar os benefícios de CI/CD devido ao distanciamento entre equipes de desenvolvimento e infraestrutura, fato motivador para que o DevOps se desenvolvesse [85]
6:1	E: Na sua opinião, dado o contexto TJGO e recomendações do CNJ, o que…	E: Na sua opinião, dado o contexto TJGO e recomendações do CNJ, o que se espera de um processo de software ? 2:50 R: Espera-se um planejamento.
6:2	planejamento da demanda, um alinhamento estratégico da instituição, u…	planejamento da demanda, um alinhamento estratégico da instituição, um alinhamento estratégico c
6:4	ter um planejamento melhor na parte de desenvolvimento para alcançar u…	ter um planejamento melhor na parte de desenvolvimento para alcançar uma prestação jurisdicional mais eficaz.
6:5	o sistema tem que garantir também que essas regras estão bem tratadas…	o sistema tem que garantir também que essas regras estão bem tratadas e que não vá ter nada prejudicial ao usuário em si.

Categoria: Fatores que favorecem

ID	Nome da Citação	Conteúdo da Citação
1:60	Reino Unido e Estados Unidos têm adotado novas estruturas organizacio…	Reino Unido e Estados Unidos têm adotado novas estruturas organizacionais de forma que as equipes de serviços digitais, no âmbito governamental, sejam capazes de responder mais rapidamente a solicitações específicas de seus clientes utilizando-se metodologia ágil. Fonte [31]
1:61	Desde 1993, com a criação do Programa Brasileiro da Qualidade e Produ…	Desde 1993, com a criação do Programa Brasileiro da Qualidade e Produtividade de Software (PBQP Software), o Brasil investe na melhoria contínua da qualidade de software. Fonte [50]
1:62	no início de 2001, professores e profissionais brasileiros começaram a…	no início de 2001, professores e profissionais brasileiros começaram a ministrar palestras sobre Extreme Programming em universidades, departamentos de TI do Governo e empresas relacionadas à indústria de software. Fonte [20]
1:63	MPS.BR - Melhoria de Processo do Software Brasileiro, lançado em 2003…	MPS.BR - Melhoria de Processo do Software Brasileiro, lançado em 2003, coordenado pela SOFTEX - Associação para Promoção da Excelência do Software Brasileiro, com apoio do Governo brasileiro, da indústria de software e de instituições de pesquisa. Fonte [52,53]
1:67	desde os anos 2000, o Brasil vem aderindo cada vez mais aos métodos á…	desde os anos 2000, o Brasil vem aderindo cada vez mais aos métodos ágeis. Fonte [1]
1:79	TJGO, que apesar de dispor de infraestrutura física e de ferramentas…	TJGO, que apesar de dispor de infraestrutura física e de ferramentas modernas para o desenvolvimento
1:80	Os desenvolvedores possuem, à disposição, tecnologias de ponta e soluç…	Os desenvolvedores possuem, à disposição, tecnologias de ponta e soluções complexas.
1:83	por meio da Resolução nº 211/2015, [60], tem o propósito de avaliar a…	por meio da Resolução nº 211/2015, [38], tem o propósito de avaliar anualmente o nível de cumprimento das diretrizes estratégicas e evolução dos viabilizadores da Governança, Gestão e Infraestrutura de Tecnologia da Informação e Comunicação (TIC) do Poder Judiciário [39]. Obs: Em 2019 o CNJ lançou uma Resolução específica para o TJGO, Resolução 119/2019, com propósito de se institucionalizar um processo de software no TJGO
1:86	Em 2001 o Manifesto Ágil [63] trouxe consigo os princípios para desenv…	Em 2001 o Manifesto Ágil [63] trouxe consigo os princípios para desenvolvimento ágil de software que ganhou popularidade no mundo todo e vem ajudando os engenheiros de software a atingirem as expectativas dos clientes. Ele trouxe consigo, além de outros princípios, aqueles que favorecem a manutenção
1:88	Pressman[66] salienta que: “não é raro uma organização de software de…	Pressman[66] salienta que: “não é raro uma organização de software despender de 60% a 70% de todos os recursos com manutenção de software”.
1:136	Laukkanen et al. [87] realizaram uma revisão da literatura em que consta…	Laukkanen et al. [87] realizaram uma revisão da literatura em que constatou sete principais problemas para emprego de CI/CD, cinco deles estão relacionados às diferentes atividades de desenvolvimento de software: projeto da arquitetura, projeto do sistema, integração, testes e entregas. Os outros dois não estão conectados a nenhuma parte individual e são: fatores humanos/organizacionais e de falta de recursos. A maioria dos problemas se concentram na integração e nos testes.

4:4	Eu penso que nesses dez anos, acabou que a gente cresceu junto com a…	Eu penso que nesses dez anos, acabou que a gente cresceu junto com a informática. A informática foi crescendo e em 2007 é que entraram mais pessoas.
4:9	? A medida que a demanda vai aumentando e se não tem pessoas suficien…	A medida que a demanda vai aumentando e se não tem pessoas suficientes, aí surge a necessidade de ter um processo.
4:11	eu acho que a demanda que faz surgir a necessidade da definição do pr…	eu acho que a demanda que faz surgir a necessidade da definição do processo
4:28	Como não tem uma rotatividade tão grande de gente,	Como não tem uma rotatividade tão grande de gente,
12:1	o CNJ está trabalhando de uma forma de pouco a pouco unificar a justiç…	o CNJ está trabalhando de uma forma de pouco a pouco unificar a justiça brasileira.
12:2	E a outra coisa que eles também deixou muito claro, o desejo deles, é…	E a outra coisa que eles também deixou muito claro, o desejo deles, é que não faça uso de tecnologias privadas, condicionando o tribunal a uma tecnologia existente (detentora somente por parte de uma empresa). Obs: Diretor de TI está se referindo aos desejos do CNJ
12:6	Durante a gestão passada então, a gente trabalhou muito com esse foco…	Durante a gestão passada então, a gente trabalhou muito com esse foco normativo e início de uma implantação nova de cultura.

Categoria: O que se espera do processo

ID	Nome da Citação	Conteúdo da Citação
1:10	possibilitaram à Aveva melhorias na qualidade do código fonte, melhor…	possibilitaram à Aveva melhorias na qualidade do código fonte, melhoria na autoestima da equipe, maior visibilidade do progresso de manutenção, melhor comunicação e compartilhamento de conhecimento entre os envolvidos. Fonte [26]
1:11	os engenheiros de software se beneficiaram de um estrutura ordenada e…	os engenheiros de software se beneficiaram de um estrutura ordenada em seu trabalho oferecida pelas boas práticas. Fonte [28]
1:17	proporcionar visualização de todos os aspectos possíveis de um proces…	proporcionar visualização de todos os aspectos possíveis de um processo de manutenção. Fonte [32]
1:33	levam os envolvidos à aumentarem aspectos de desempenho, como maior e…	levam os envolvidos à aumentarem aspectos de desempenho, como maior engajamento, produtividade, criatividade e moral. Fonte [33]
1:40	detectou uma maior satisfação dos clientes, redução de custos, maior p…	detectou uma maior satisfação dos clientes, redução de custos, maior previsibilidade de custos e prazos, além do aumento da produtividade e qualidade. Fonte [35]
1:48	uma melhora perceptível na redução de defeitos provocados pelas manut…	uma melhora perceptível na redução de defeitos provocados pelas manutenções. Fonte [36]
1:64	apresenta que grande parte das melhorias propostas no MPS.BR podem se…	apresenta que grande parte das melhorias propostas no MPS.BR podem ser alcançadas utilizando-se Scrum: 67% dos resultados esperados pelo MPS.BR são atendidos pelo Scrum, 25% são parcialmente atendidos, contra apenas 8% dos resultados que não são atendidos. Fonte [54]
1:68	A aplicação de ágeis no Governo brasileiro é, em sua maioria, bem suce…	A aplicação de ágeis no Governo brasileiro é, em sua maioria, bem-sucedida e, normalmente, conduzida em combinação com outras abordagens ágeis de desenvolvimento de software ou, em sua maioria, uma combinação de ágeis com métodos tradicionais Fonte [1]
1:69	A escolha do Kanban se 13deu pela capacidade de adaptação às frequent…	A escolha do Kanban se deu pela capacidade de adaptação às frequentes mudanças nas requisições de manutenção de software pelo órgão junto ao fornecedor, prestador de serviços de manutenção. Fonte [59]
1:70	Os resultados permitiram constatar que o novo processo atenuou proble…	Os resultados permitiram constatar que o novo processo atenuou problemas, diminuiu a resistência as mudanças, e contribuiu para aumentar a satisfação dos envolvidos. Fonte [57]
1:73	Após a aplicação do modelo, constataram que os indicadores de desempe…	Após a aplicação do modelo, constataram que os indicadores de desempenho melhoraram, assim como a satisfação dos clientes. Fonte [58]
1:75	os autores, as motivações para a adoção de ágeis incluem: (1) uma ent…	os autores, as motivações para a adoção de ágeis incluem: (1) uma entrega rápida de valor ao cliente; (2) uma maior colaboração entre TI e negócios; (3) uma maior satisfação do cliente. Além disso, os autores [57] chamam a atenção para um fato novo: (4) aumento na moral da equipe de TI do governo, reduzindo a dependência de empresas contratadas. Fonte [58]

1:77	(1) comunicação entre os membros da equipe de desenvolvimento, bem com…	(1) comunicação entre os membros da equipe de desenvolvimento, bem como, entre desenvolvedores e clientes; (2) aprendizado de novas tecnologias; (3) qualidade do produto; (4) visibilidade do projeto; (5) produtividade das equipes; (6) redução de custos; (7) capacidade de gerenciar as mudanças e as prioridades; e (8) conformidade com exigências burocráticas do Governo. Fonte [58]
1:84	De acordo com April [38], para uma organização ser considerada madura…	De acordo com April [38], para uma organização ser considerada madura é necessário que ela tenha instituído o seu processo de manutenção de software.
1:134	Embora os métodos ágeis sejam cada vez mais comuns, muitas Organizaçõe…	Embora os métodos ágeis sejam cada vez mais comuns, muitas Organizações não conseguiram alcançar os benefícios de CI/CD devido ao distanciamento entre equipes de desenvolvimento e infraestrutura, fato motivador para que o DevOps se desenvolvesse [82]
6:1	E: Na sua opinião, dado o contexto TJGO e recomendações do CNJ, o que…	E: Na sua opinião, dado o contexto TJGO e recomendações do CNJ, o que se espera de um processo de software ? 2:50 R: Espera-se um planejamento.
6:2	planejamento da demanda, um alinhamento estratégico da instituição, u…	planejamento da demanda, um alinhamento estratégico da instituição, um alinhamento estratégico c
6:4	ter um planejamento melhor na parte de desenvolvimento para alcançar u…	ter um planejamento melhor na parte de desenvolvimento para alcançar uma prestação jurisdicional mais eficaz.
6:5	o sistema tem que garantir também que essas regras estão bem tratadas…	o sistema tem que garantir também que essas regras estão bem tratadas e que não vá ter nada prejudicial ao usuário em si.

Apêndice F

Termo de Compromisso e Perguntas para os Entrevistados

TERMO DE COMPROMISSO

Esta pesquisa integra parte do trabalho de coleta de dados para conclusão do curso de mestrado no Programa de Pós-graduação em Computação Aplicada da Universidade de Brasília (UNB). Trata-se de um programa que, entre outros objetivos, tenta levar possíveis soluções para dentro das empresas, principalmente empresas públicas, através do rigor acadêmico do mestrado.

Pretende-se com essa entrevista obter dados que evidenciem as atividades de processo de software desempenhadas dentro do TJGO. Serão inspecionados aspectos referentes às dificuldades encontradas, tecnologias utilizadas, benefícios conseguidos e perfil das partes envolvidas para lidar com manutenções de software.

Trata-se de uma pesquisa na qual as preocupações quanto ao anonimato, confidencialidade e ética estão em primeiro lugar. Você está sendo convidado a participar por livre vontade e poderá deixar de participar a qualquer momento, sem prejuízo. As respostas serão tratadas com finalidade única e exclusiva de se entender os fenômenos que envolvem a Engenharia de Software.

Entretanto, é importante esclarecer que trata-se de uma pesquisa que se tornará pública, faz uso da ferramenta de videoconferência Zoom, e possui perguntas que serão gravadas e transcritas, que se forem bem analisadas podem dar indícios de autoria. Desse modo, compete ao entrevistado a responsabilidade sobre as informações prestadas e ao entrevistador a cautela quanto às perguntas.

A transcrição será anexada ao trabalho final e o(a) senhor(a) receberá uma cópia por e-mail.

O tempo de resposta aproximado será de 10 min.

PERGUNTAS PARA EQUIPE DE TI

Solicitar apresentação:
Qual a formação ? Qual o cargo ? Quanto tempo de serviço no TJ ?

1 - Na sua opinião por que a DES ainda não possui um processo de software bem definido ?
- estimular a fala sobre: os porquês do RUP não ter dado certo
- estimular a fala sobre: o Redmine e sobre o Kanban

2- Descreva, brevemente, sobre o processo de manutenção de software antigo, como era sua rotina ?
- estimular a fala sobre: o modelo de demandas

3 - Descreva, brevemente, sobre o processo de manutenção de software atual (como é sua rotina ?)
- estimular a fala sobre: deploy em produção, testes, qualidade, prazos

PERGUNTAS PARA USUÁRIOS GESTORES

Solicitar apresentação:
Qual a função ? Qual a lotação atual ? Quanto tempo de serviço no TJ ?

1- Gostaria que você falasse um pouco sobre o projeto que fizemos juntos, se gostou de participar, se viu diferença com outros projetos de software que já tenha participado no TJGO?
- estimular a fala sobre: os ambiente de testes, relatório de acompanhamento, reuniões de apresentação desde o começo.

2- Descreva sobre os resultados entregues viu melhoria na qualidade em relação às anteriores?

PERGUNTAS PARA OS DIRETORES

Solicitar apresentação:
Qual a função ? Qual a lotação atual ? Quanto tempo de serviço no TJ ?

1- Na sua opinião, dado o contexto do TJGO e recomendações do CNJ, o que se espera de um processo de software ?

2 - Na sua opinião, por que a DES ainda não possui um processo de software bem definido (Resolução 119/2019)?
- estimular a fala sobre: as principais dificuldades, outras tentativas não deram certo.

3 - Nesses últimos anos tentamos utilizar um processo de manutenção. Poderia falar um pouco sobre o processo anterior e o atual do NTSA ?
- estimular a fala sobre: Qualidade, Velocidade, Arquitetura Deploy, erros e reclamações

Apêndice G

Transcrição das Entrevistas com os Diretores

TRANSCRIÇÃO DA ENTREVISTA REALIZADA COM O DIRETOR DE TI

14/06 13h
Vide Termo de Compromisso no APÊNDICE F.

TRANSCRIÇÃO DA ENTREVISTA REALIZADA COM O DIRETOR DE TI

14/06 13h
Vide Termo de Compromisso no APÊNDICE F.

Transcrição das entrevistas em documento externo.
Interessados, entrar em contato com o autor.
E-mail: eng.maribeiro@gmail.com

Apêndice H

Transcrição das Entrevistas com os Usuários gestores

TRANSCRIÇÃO DA ENTREVISTA REALIZADA COM O USUÁRIO GESTOR 1

11/06/2021 - 15h
Vide Termo de Compromisso no APÊNDICE F.

Transcrição das entrevistas em documento externo.
Interessados, entrar em contato com o autor.
E-mail: eng.maribeiro@gmail.com

TRANSCRIÇÃO DA ENTREVISTA REALIZADA COM O USUÁRIO GESTOR 2

11/06 - 17 h
Vide Termo de Compromisso no APÊNDICE F.

Transcrição das entrevistas em documento externo.
Interessados, entrar em contato com o autor.
E-mail: eng.maribeiro@gmail.com

Apêndice I

Transcrição das Entrevistas com o Time de TI

TRANSCRIÇÃO DA ENTREVISTA REALIZADA COM O ANALISTA "A"

10/06/2021 12:00h
Vide Termo de Compromisso no APÊNDICE F.

Transcrição das entrevistas em documento externo.
Interessados, entrar em contato com o autor.
E-mail: eng.maribeiro@gmail.com

TRANSCRIÇÃO DA ENTREVISTA REALIZADA COM O ANALISTA "B"

10/06/2021 13 h
Vide Termo de Compromisso no APÊNDICE F.

Transcrição das entrevistas em documento externo.
Interessados, entrar em contato com o autor.
E-mail: eng.maribeiro@gmail.com

TRANSCRIÇÃO DA ENTREVISTA REALIZADA COM O ANALISTA "C"

10/06/2021 14h
Vide Termo de Compromisso no APÊNDICE F.

Transcrição das entrevistas em documento externo.
Interessados, entrar em contato com o autor.
E-mail: eng.maribeiro@gmail.com

TRANSCRIÇÃO DA ENTREVISTA REALIZADA COM O ANALISTA "D"

10/06/2021 -16h
Vide Termo de Compromisso no APÊNDICE F.

Transcrição das entrevistas em documento externo.
Interessados, entrar em contato com o autor.
E-mail: eng.maribeiro@gmail.com

Apêndice J

Modelos para melhoria contínua em Manutenção de Software

A seguir, serão apresentados dois modelos que podem apoiar a evolução de um processo de manutenção de sistemas. Primeiro, apresenta-se o modelo Mantus que, com base na ISO 25010, identifica atributos que influenciam na manutenção desde o momento do desenvolvimento do sistema e adiante, apresenta-se o modelo de maturidade SMmm de melhoria contínua em manutenção de software.

J.0.1 Mantus um Modelo apoiado na ISO 25010

O Modelo de referência Mantus [12] propõe a identificação dos atributos que influenciam na manutenção desde o desenvolvimento e então estabelecem boas práticas a fim de se alcançar um produto final altamente manutenível. Para isso, Erazo et al. [12] buscaram na norma de qualidade de software ISO/IEC 25010 [125] essas boas práticas para aumentar a manutenibilidade.

Conforme a ISO/IEC 25010 [125], manutenibilidade é: "o grau de eficácia e eficiência que um produto ou sistema pode ser modificado pelos mantenedores". São características do software que podem facilitar a manutenção quando forem necessárias e são divididas em 5 [125]:

- modularidade;
- reutilização;
- capacidade de ser analisado;
- capacidade de ser modificado;
- capacidade de ser testado.

De acordo com as pesquisas de Erazo et al. [12], incluir essas características no momento do desenvolvimento do software pode ajudar a aumentar o grau de manutenibilidade, sendo este o propósito do modelo Mantus.

O modelo Mantus contém seis processos e uma série de práticas que são definidas para cada processo, sendo eles: de requisitos de software, projeto, arquitetura, construção, avaliação e gestão da documentação cada um deles tenta maximizar as sub-características de manutenção descritas na norma ISO/IEC 25010 [12].

Esse modelo se destina a empresas desenvolvedoras de software que desejam garantir a manutenção de seus produtos durante o processo de desenvolvimento e também para aquelas que desejam obter o certificado ISO 25000 de gestão da qualidade do produto de software [12].

Para definir o modelo Mantus [12], considerou-se atributos desde o início do ciclo de desenvolvimento em que foram estabelecidos seis processos: Análise e Requisitos de Software na Perspectiva da Manutenibilidade (DEV-MANT.1); Desenho Arquitetural de Software na Perspectiva da Manutenibilidade (DEV-MANT.2), Desenho Detalhado de Software na Perspectiva da Manutenibilidade (DEV-MANT.3); Construção de Software na Perspectiva da Manutenibilidade (DEV-MANT.4); Provas de Avaliação de Software na Perspectiva da Manutenibilidade (DEV-MANT.5); Gestão da Documentação na Perspectiva da Manutenibilidade (SUP-MANT.1). A partir disso, foram investigados os atributos que influenciam na manutenção do produto e então se analisou como é possível potencializar cada sub característica para aumentar a manutenibilidade.

Erazo et al. [12] obtiveram a classificação de 18 atributos de software levando em conta: as sub características de manutenção da ISO/IEC 25010, do fluxo de desenvolvimento de software de acordo com o RUP por meio de uma análise semântica de suas definições e dos processos de melhoria de software definidos na ISO/IEC 15504. Os resultados das relações entre esses elementos são mostrados na Figura J.1:

Subcaracterísticas / Atributos	CA	M	CM	CP	R	Processos
Acoplamento	X	X	X	X	X	Desenho arquitetural, Desenho detalhado
Aninhamento	X		X	X		Construção de software
Capacidade de expansão			X			Operação de software
Coesão	X	X	X	X	X	Desenho detalhado
Comentários	X		X		X	Construção de software
Complexidade	X	X	X	X	X	Desenho arquitetural, Desenho detalhado, Construção de software
Consistência	X		X			Análise de requisitos, Desenho arquitetural, Desenho detalhado, Construção de software, Provas de avaliação de software, Operação de software
Documentação	X		X	X	X	Análise de requisitos, Desenho arquitetural, Desenho detalhado, Construção de software, Provas de avaliação de software, Operação de software
Duplicação	X		X	X		Construção de software
Encapsulamento	X	X	X	X		Desenho detalhado
Padronização	X		X			Construção de software
Facilidade de leitura	X		X	X	X	Construção de software
Facilidade de entendimento	X		X	X	X	Desenho arquitetural, Desenho detalhado, Construção de software
Herança	X		X	X	X	Desenho detalhado
Polimorfismo	X		X		X	Desenho detalhado
Simplicidade	X	X	X	X	X	Análise de requisitos, Desenho arquitetural, Desenho detalhado, Construção de software
Tamanho	X		X	X	X	Construção
Rastreabilidade	X		X			Análise de requisitos, Desenho arquitetural, Desenho detalhado, Construção de software, Provas de avaliação de software,

CA = Capacidade para ser analizado, M = Modularidade, CM = Capacidade para ser modificado, CP = Capacidade para ser testado e R = Reusabilidade

Figura J.1: Classificação dos atributos por sub características de manutenção e processos (Fonte:[12]).

Mantus não descreve como fazer e sim o quê fazer e, desse modo, pode ser implementado de diferentes formas. Esse modelo estabelece a relação entre seus processos em duas perspectivas: de forma geral e independente para características de manutenção e espe-

cífica para cada uma das cinco sub características de manutenção: modularidade (M), reusabilidade (R), capacidade para ser analisado (CA), capacidade para ser modificado (CM) e capacidade para ser provado (CP). Para cada sub característica, se estabelecem práticas bases, atividades, necessárias para atingir os resultados esperados.

J.0.2 *Software Maintenance Maturity Model* (SMmm)

Formalmente, ao definir, medir e melhorar processos é recomendável definir um mapeamento do mundo empírico para o relacional formal [31].

April et al.[13] revisaram 40 modelos de maturidade e identificaram alguns como principais para se medir e melhorar processos, que são:

- ISO/IEC14764;
- IEEE1219;
- ISO/IEC12207;
- CMMI;
- SWEBOK.

A partir dessa revisão sistemática, April et al.[13] apresentaram o modelo SMmm que aborda atividades de manutenção de software que ajudam a identificar pontos de melhorias no processo de manutenção. O SMmm funciona como um complemento para o processo de melhoria contínua proposto no *Capability Maturity Model Integration* (CMMI). Esse modelo agrupa os processos de manutenção em três grupos: Processos Primários, processos de suporte para apoiar os processos primários e processos organizacionais.

O SMmm também recebe insumos e faz referência a outros seis modelos de maturidade e melhores práticas de publicações que consideram uma variedade de tópicos relacionados à manutenção de software [13]:

- Modelo de maturidade Camélia;
- Modelo para melhorar a manutenção de software;
- Governança, controle e auditoria de informação e tecnologia relacionada (COBIT);
- Modelo de maturidade de manutenção corretiva (CM3): Gerenciamento de problemas;
- Serviço de TI CMM.

De acordo com April et al. [13] sempre que apresentavam o modelo SMmm em conferências eles eram questionados se tal pesquisa era realmente necessária, visto que existem outros modelos, como o próprio CMMI, que já abordam a manutenção de software. Para responder a essa pergunta, eles explicavam que as atividades são exclusivas dos mantenedores e não fazem parte dos processos de desenvolvimento de software. Quando essas atividades de manutenção exclusivas são comparadas com o conteúdo do modelo CMMI, pode-se observar que o modelo CMMI, com seu foco principal no gerenciamento de projetos, não aborda explicitamente os problemas específicos da manutenção de software.

Desse modo, o SMmm foi projetado como um modelo de referência complementar ao CMMI. Isso é útil porque, embora existam atividades diferentes, algumas são comuns para desenvolvedores e mantenedores como, por exemplo:

- Definição de processo;
- Codificação;
- Teste;
- Gerenciamento de configurações; e
- Garantia da Qualidade.

Manter um alinhamento também é importante para que aqueles que já usam o CMMI possam reconhecer e usar a estrutura do modelo de maturidade proposto. A arquitetura do SMmm difere um pouco da versão CMMI, mas as diferenças mais significativas são:

- mudança da área de processo de gerenciamento de projetos para uma área de processo de gerenciamento de solicitações;
- adaptou os CMMI KPAs para focar em atividades exclusivas de manutenção de software; e
- referências detalhadas a documentos e exemplos sobre como implementar as muitas tarefas de manutenções práticas.

O SMmm incorpora práticas adicionais aos seguintes tópicos [13]:

- gerenciamento de solicitações de eventos e serviços;
- atividades de planejamento de manutenção específicas para mantenedores (versão, SLA, análise de impacto);

- SLA;
- Transição de software;
- suporte operacional;
- processo de resolução de problemas; e
- rejuvenescimento, conversão e aposentadoria de software.

Segundo seus autores, April at al. [13], o SMmm é um modelo focado no cliente. O objetivo final dos programas de melhoria com resultado da aplicação do SMmm é uma maior satisfação do cliente em vez de rígida conformidade com padrões referenciados por esse modelo representando para as organizações/clientes:

- atingir os níveis de serviço pretendidos e cumprir as prioridades do cliente;
- implementar as melhores práticas disponíveis para os mantenedores de software;
- obter transparência sobre os serviços de manutenção de software e custos competitivos;
- experimentar o menor tempo possível de manutenção de software.

Além disso, contribuir para:

- auditoria da capacidade de manutenção de software de um fornecedor de serviços de manutenção; ou
- melhorar as organizações internas de manutenção de software.

O SMmm estabelece os seguintes níveis de maturidade [13]:

Nível	Nome do nível	Risco	Interpretação
1	Realizado	Muito alto	Processo de manutenção *Ad hoc*
2	Gerenciado	Alto	Processo básico
3	Estabelecido	Médio	Processo de manutenção no estado da arte
4	Previsível	Baixo	Geralmente difícil de atingir
5	Otimizado	Muito baixo	Tecnologicamente desafiador para se atingir

Figura J.2: Níveis de maturidade SMmm (Fonte: [13]).

De acordo com April et al. [13], atingir uma maior maturidade em manutenção pode resultar em:

- menores custos de manutenção e suporte;
- menor tempo de ciclo e intervalos;
- maior capacidade de atingir níveis de serviço; e
- maior capacidade de atender aos objetivos quantificáveis da qualidade em todas as etapas do processo de manutenção e serviço.

O SMmm usa um modelo contínuo de representação da maturidade com o propósito de obter maior granularidade nos domínios, identificar práticas específicas nos níveis de maturidade, identificar a trajetória do nível mais baixo (1 - realizado) para um nível mais alto de maturidade. Essa representação contínua está alinhada às recomendações da ISO/IEC 15504 e possui quatro domínios de processo e dezoito KPAs com suas respectivas práticas. Embora alguns KPAs sejam exclusivos para manutenção, outros foram derivados do CMMI e de outros modelos e foram modificados para mapear as características de manutenções diárias [13].

Apêndice K

Conceitos de Técnicas de Testes Utilizadas com Métodos Ágeis

A seguir, serão apresentadas conceitos sobre as técnicas de testes que a comunidade ágil tem adotado. Primeiro apresenta-se o TDD, uma técnica que tem ganhado popularidade com os métodos ágeis, em especial aos adeptos do *Extreme Programming* (XP), em seguida ATDD, BDD e motivos para se automatizar os testes.

K.0.1 TDD *Test-Driven Development*

TDD é a consciência da lacuna entre a decisão e a consequência durante a programação [124]. Mesmo que se projete em papel por uma semana e em seguida, ao codificar, fazê-lo dirigido por testes, estará praticando TDD pois controlou-se a lacuna entre a decisão e a consequência deliberadamente [124].

Basicamente o TDD faz com que os desenvolvedores se concentrem primeiro nos requisitos e a partir deles escrevam testes unitários antes do desenvolvimento do sistema[107].

De acordo com Crispin e Gregory [64], nos testes ágeis os programadores nunca seguem a frente dos testadores, porque uma história nunca estará pronta até que tenha sido testada.

O processo TDD consiste dos seguintes passos [107]:

1. Selecionar uma história de usuário;
2. Escrever um teste que cumpra uma pequena tarefa da história do usuário e executar esse teste. Em seguida produzir um teste que falhe;
3. Escrever o código que implemente a funcionalidade;
4. Executar os testes pré-existentes novamente. Quando o código estiver correto, vá para o estágio de refatoração;
5. Quando o estágio de refatoração for concluído, o código correto estará pronto podendo passar para a próxima história de usuário.

Ao escrever os testes antes do desenvolvimento faz com que se tenha maior foco nos requisitos e seus resultados, evitando assim que os testes sejam enviesados. Este método traz benefícios como: aumento da qualidade, produtividade além da redução de envolvimento humano em atividades manuais e repetitivas [107, 126].

Entretanto, embora o TDD seja uma prática sugeria para se alcançar qualidade de software utilizando métodos ágeis, ela ainda está longe de ser uma prática amplamente utilizada [127, 128].

K.0.2 ATDD (*Acceptance Test Driven Development*)

ATDD foca na qualidade externa do software, requer a participação do cliente para fornecer histórias de usuários finais à equipe de desenvolvimento/teste. Essas histórias são refinadas em testes de aceitação que conduzem o processo de desenvolvimento. Consiste dos seguintes passos [107]:

1. Selecionar uma história de usuário;
2. Escrever teste de aceitação;
3. Implementar história de usuário;
4. Rodar o teste de aceitação;
5. Fazer pequenas modificações/refatoração.

Os testes de aceitação consistem na avaliação dos clientes e usuários se o software entregue encontra-se de acordo com o que foi solicitado e se está ou não suficiente para ser entregue. São exemplos de teste de aceitação: o teste Alfa e o teste Beta.

A implementação de testes de aceitação ganharam forças nos métodos ágeis, devido a escassez de tempo para implementação de outros testes [102]. Eles podem ser simples ao se utilizar histórias de usuário e linguagem natural, e assim, guiar todo desenvolvimento por teste de aceitação [14].

A abordagem ATDD ajuda a equipe de desenvolvimento a transformar requisitos do software em casos de teste permitindo a validação do sistema com mais facilidade e eficiência [14].

K.0.3 BDD (*Behavior-Driven Development*)

No BDD as histórias de usuário são transformadas em cenários. Estes cenários visam representar o comportamento do software e são submetidos a critérios de aceitação que descrevem se o funcionamento está de acordo com o esperado. Trata-se de uma técnica para se melhorar a comunicação entre os envolvidos.

BDD se concentra na perspectiva dos resultados de negócios, foca no aspecto comportamental do sistema, ao contrário do TDD que foca no aspecto de implementação. Consiste dos seguintes passos [107]:

1. Escrever um cenário;

2. Rodar o cenário que falha;
3. Escrever os testes que correspondam a especificação do cenário;
4. Escrever o código mais simples para passar no teste e depois no cenário;
5. Refatorar para eliminar duplicação.

A linguagem utilizada no TDD pode afastar os clientes da equipe de desenvolvimento pois trata-se de uma linguagem técnica utilizada pelos programadores. Assim, a utilização de uma comunicação clara e próxima da humana se mostra conveniente para aproximar os clientes do desenvolvimento da solução.

O uso de uma linguagem ubíqua entre interessados e o time de desenvolvimento, aumenta a transparência e possibilita que os requisitos sejam testados ainda na fase de elicitação [14].

A linguagem utilizada no BDD utiliza sintaxe Gherkin, sendo o Cucumber uma das ferramentas mais populares para essa escrita. Além de ser uma linguagem de fácil entendimento por qualquer pessoa, ela possibilita a automação dos casos.

As palavras-chave utilizadas na abordagem BDD são [14]:

- Feature: Define uma funcionalidade do sistema, pode conter um ou mais cenários.
- Background: Descreve um cenário inicial que será executado antes de cada cenário;
- Scenario: Define um cenário de uso de sua respectiva *feature* e é composto de um ou mais *steps*.
- Steps: É a unidade básica do BDD, define um passo de execução do cenário. Possibilita a verificação de determinado estado e *outputs*. Os steps são baseados nas palvras-chave *Given*, *When*, *Then*, *And* e *But*.
- Tags: Usadas quando se deseja acrescentar informações.

A seguir segue um exemplo:

K.0.4 Testes automatizados

A execução manual de um caso de teste é rápida e efetiva, mas a execução, atualização e repetição de um número elevado de testes é uma tarefa cansativa que com o tempo se torna inviável. Desse modo, é comum que os testadores não executem todos os casos

```
Feature: Multiple site support
  Only blog owners can post to a blog, except administrators,
  who can post to all blogs.

  Background:
    Given a global administrator named "Greg"
    And a blog named "Greg's anti-tax rants"
    And a customer named "Dr. Bill"
    And a blog named "Expensive Therapy" owned by "Dr. Bill"

  Scenario: Dr. Bill posts to his own blog
    Given I am logged in as Dr. Bill
    When I try to post to "Expensive Therapy"
    Then I should see "Your article was published."

  Scenario: Dr. Bill tries to post to somebody else's blog, and fails
    Given I am logged in as Dr. Bill
    When I try to post to "Greg's anti-tax rants"
    Then I should see "Hey! That's not your blog!"

  Scenario: Greg posts to a client's blog
    Given I am logged in as Greg
    When I try to post to "Expensive Therapy"
    Then I should see "Your article was published."
```

Figura K.1: Exemplo de BDD com sintaxe Gherkin (Fonte: [14])

a cada mudança e daí surgem os erros, levando a prejuízo para todos os envolvidos e diminuindo a qualidade das entregas[129].

Nos últimos anos a disponibilidade, o nível das ferramentas de testes e a preocupação com a qualidade aumentaram em toda indústria [102].

De acordo com Crispin e Gregory [64] existem alguns motivos para se querer automatizar testes, são eles:

- Testes manuais são demorados;
- Testes manuais são propensos a erros;
- Liberar tempo para trabalhar melhor;
- Gerar uma rede segura de testes;
- Fornecer frequente e rápido *feedback*;
- Codificação guiada por testes e exemplos podem ir mais longe;
- Testes fornecem documentação;
- Retorno sobre o investimento.

Cada modificação ou correção de um erro resulta em muito esforço para execução de todos testes manuais e caso não seja feito poderá acarretar erros de regressão, que são

aqueles erros que afetam outros módulos do sistema que estavam funcionando antes da manutenção, e assim entrar num ciclo de manutenção tornando uma tarefa tão difícil que passa a valer a pena reconstruir o software novamente [129].

O teste de regressão tem ligação direta com a manutenção de sistemas, a cada manutenção corre-se o risco de que novos erros apareçam. Assim, sempre que ocorre uma manutenção é necessário que novos testes sejam realizados com propósito de se encontrar novas falhas provocadas pela manutenção. Nesse contexto os testes automatizados podem facilitar a integração contínua (CI) requerida pelos métodos ágeis.

Entretanto, devido a quantidade de teste que surgem com as manutenções, apenas automatizá-los não é suficiente. Saber o quê automatizar também é muito importante. Ali et al. [130] propôs uma abordagem para melhorar a qualidade dos lançamentos que consistem em: calcular a frequência de execução dos casos de testes, agrupar os casos de testes semelhantes em *clusters* e utilizar parâmetros para priorização.

Segundo Vassallo et al. a integração contínua é uma prática ágil de desenvolvimento de software. Esta propõe a integração de recursos com a frequência ágil em vez de integrá-los apenas momentos antes da entrega do produto. Isso encurta o tempo de lançamento e melhora a qualidade. Um dos problemas que devem ser controlados é que caso ocorra falha no *commit* o processo será interrompido. Como resultado, os desenvolvedores gastam uma quantidade significativa de seu tempo de trabalho compreendendo e resolvendo quebras de construção.

Assim como nos testes manuais, nos testes automatizados, os *scripts* são criados através da especificação de requisitos, são executados e avaliados os resultados sem acesso direto ao código, chamados de testes de caixa preta, possuem como principal vantagem a possibilidade de serem executados a qualquer tempo e com pouco esforço conforme Kon (2008, apud Alves) [120].

Além disso, os testes automatizados tiram proveito da eficiência computacional, realizando tarefas que seriam impossíveis pelos testes manuais, como por exemplo os testes de carga.

O momento para criação de testes automatizados varia de acordo com a experiência e conhecimento da equipe [120], muitas vezes a prioridade está na velocidade da entrega, mesmo que seja necessário abrir mão da qualidade e de outros atributos. Mas, muitos estudos foram desenvolvidos durante os últimos anos e o momento para criação dos testes permanece em discussão, aparecendo como uma dos grandes desafios de pesquisa em engenharia de software empírica [127].

Uma prácia comum é iniciar a criação de testes quando os requisitos estão estáveis

sem solicitação de grandes mudanças e após a execução dos testes manuais. As equipes que possuem membros que adotam papel de testador e desenvolvedor em momentos diferentes não costuma apresentar testes automatizados visto que não sobra tempo para tantas tarefas [120].

Duas estratégias para automatização de testes foram identificada no trabalho de Alves [120]: começar com as partes mais simples ou, pensando no custo com a manutenção na automação, aqueles testes de maior frequência. Em ambos os caso é importante que se faça a modularização do código para que assim possa ser facilmente rastreável através de uma matriz de rastreabilidade. A não adoção de boas práticas pode elevar os custos da automatização dos testes.

Entretanto, construir uma suíte de testes automatizados, requer padronização de atividades, arquitetura que suporte os testes e conhecimento em codificação e análise por parte do testador. Para fazer testes de unidade, é necessário entender sobre o código, se for teste de sistema deve-se conhecer as regras de negócio e os resultados esperados. Além disso, conhecer bem os cenários evita a construção de casos de testes repetidos e quais testes devem ser de fato automatizados pois o trabalho para mantê-los atualizados já é bastante grande [120].

Através de métricas é possível identificar se os testes estão sendo vantajosos. Uma das formas mais utilizadas é a contagem de defeitos obtidos em desenvolvimento, homologação e produção [131].

Estar atento a ordem de execução dos testes é fundamental, pois muitos testes possuem dependências e em muitos casos, simplesmente alterando a ordem de execução, é suficiente para produzir resultados de teste diferentes e uma propriedade fundamental é dos testes automatizados de regressão é a capacidade de serem rapidamente reproduzidos sem erros [132].

Anexo I

Elogio Formal do Excelentíssimo Senhor 3° Juiz Auxiliar da Corregedoria-Geral de Justiça de Goiás em Virtude da Agilidade na Manutenção de um dos Sistemas Administrativos

PODER JUDICIÁRIO
Tribunal de Justiça do Estado de Goiás
CORREGEDORIA-GERAL DA JUSTIÇA
Gabinete do 3º Juiz Auxiliar

Nº Processo PROAD acima

PARECER Nº 000629/2020

Trata-se de PROAD instaurado por solicitação do Gabinete do 3º Juiz Auxiliar da CGJ com a sugestão de criação de 3 prioridades ("propriedades" de prioridade de tramitação) para utilização no cadastramento de procedimentos administrativos, através do Sistema PROAD: Urgente CGJ, Urgente CNJ e COVID-19

A Assessoria Correicional apresentou informação a favor da criação das prioridades.

A Diretoria de Tecnologia de Informação se manifestou indicando que o sistema PROAD é desenvolvido e mantido pelo Núcleo de Sistemas Administrativos da Divisão de Engenharia de Software da Diretoria de Informática do TJGO.

Os autos foram encaminhados à Diretoria de Informática para consulta sobre (a) a viabilidade técnica da ferramenta sugerida e (b) o cronograma necessário para implementação.

Por fim, a Divisão de Engenharia de Software informou que as prioridades foram criadas e sugeriu o encaminhamento dos autos ao Comitê Gestor do PROAD.

A seguir, o presente PROAD foi à conclusão do 3º Juiz Auxiliar da Corregedoria-Geral da Justiça.

Manifesto.

Excelentíssimo Senhor Corregedor-Geral da Justiça.

A experiência que tenho com processo judicial eletrônico (há mais de 10 anos), fez com que desde o início da Gestão 2019/2021, empreendesse um trabalho cuidadoso e dedicado a identificar possíveis pontos de melhoramento das ferramentas disponíveis no Sistema PROAD, utilizado para a movimentação dos processos administrativos.

Fato é que sentia a necessidade de uma classificação mais precisa das prioridades indicadas para os procedimentos em análise, algo próximo do que é proposto pela matriz de priorização de processos já bem consolidada no Sistema de Processo Judicial Digital (PROJUDI).

E com a pandemia COVID-19, temos uma situação extremamente excepcional, sem precedentes na história, e que nos impele a utilizar todos os instrumentos e ferramentas tecnológicas que temos para otimizar os trabalhos desempenhados.

Assim, sugeri a criação das prioridades Urgente CGJ, Urgente CNJ e COVID-19, para incrementar a lista de prioridades existente: (a) idade igual ou superior a 60 (sessenta) anos, (b) pessoa com deficiência, (c) portador de doença grave e (d) reforma predial (Portaria 05/2020).

Ocorre que em diligência junto a Diretoria de Informática para verificar a viabilidade técnica, bem como eventual cronograma necessário para implementação, as prioridades foram imediatamente criadas.

A gestão eficiente daquela Diretoria nos impressiona e é digna de elogio, pois além de extremamente veloz (as prioridades foram criadas menos de 7 dias úteis após a instauração do PROAD), se mostrou sensível às necessidades dos usuários ao atender a um pleito que ainda sequer tinha sido objeto de deliberação de Vossa Excelência.

Além disso, a matriz de priorização de processos utilizada pela Divisão de Engenharia de Software repercute com exatidão as necessidades de gestão processual no PROAD. É que os

procedimentos identificados com a prioridade **COVID-19** aparecem em destaque vermelho e como os primeiros na caixa, tanto de entrada, quanto da área de trabalho. Na sequência, a prioridade **Urgente CNJ** aparece em destaque azul, e depois a prioridade **Urgente CGJ** em destaque verde.

Destarte, sugerirei o encaminhamento do PROAD ao exame do Comitê Gestor do PROAD, conforme sugerido pela Divisão de Engenharia de Software, e a formalização de elogio com os respectivos agradecimentos ao Diretor de Informática, Sr. Anderson Yagi Costa.

Posto isso, OPINO (a) pelo encaminhamento do PROAD ao exame do Comitê Gestor do PROAD, (b) pela formalização de elogio junto a Diretoria de Recursos Humanos no dossiê do Diretor de Informática, Sr. Anderson Yagi Costa, bem como (c) pela cientificação do referido diretor com os respectivos agradecimentos.

Submeto esse parecer ao crivo do Corregedor-Geral da Justiça.

Goiânia-GO, datado e assinado digitalmente.

ALDO GUILHERME SAAD SABINO DE FREITAS
3º Juiz Auxiliar da CGJ

ASSINATURA(S) ELETRÔNICA(S)

Tribunal de Justiça do Estado de Goiás
Para validar este documento informe o código 309774875878 no endereço https://proad-v2.tjgo.jus.br/proad/publico/validacaoDocumento

Nº Processo PROAD: 202004000223041

ALDO GUILHERME SAAD SABINO DE FREITAS
JUIZ DE DIREITO
GABINETE DO JUIZ AUXILIAR DA CORREGEDORIA 3
Assinatura CONFIRMADA em 12/05/2020 às 12:23