Framework de Avaliação de Maturidade para Práticas de Gerenciamento e Desenvolvimento de Software baseado no ITIL4

Guilherme Spati[1], **Rodolfo Miranda de Barros**[1]

[1]Departamento de Computação – Universidade Estadual de Londrina (UEL)
Caixa Postal 10.011 – CEP 86057-970 – Londrina – PR – Brasil

guilherme.spati723@gmail.com, rodolfo@uel.br

Abstract. *Currently, the majority of organizations rely on services provided by the Information Tecnology department, making it an extremely important element for ensuring the organization's success. Having a software development process is essential to prioritize software quality, generating more value for the customer, especially as the industry in this field is experiencing continuous growth, leading to an increasingly competitive market. Therefore, this present work proposes the development of a framework to diagnose the maturity level of software management and development practices according to ITIL4.*

Resumo. *Atualmente, a maior parte das organizações depende dos serviços prestados pela área de Tecnologia da Informação, tornando esta um objeto de extrema importância para a garantia de sucesso da organização. Possuir um processo de desenvolvimento de software é imprescindível para que a qualidade do software seja o objetivo, gerando mais valor ao cliente, visto que a indústria nesta área esta em crescente expansão, tornando o mercado cada vez mais competitivo. Sendo assim, o presente trabalho propõe o desenvolvimento de um framework para diagnosticar o grau de maturidade da prática de gerenciamento e desenvolvimento de software de acordo com o ITIL4.*

1. Introdução

Com o constante avanço das tecnologias de informação e a crescente dependência das organizações em sistemas de *software* eficientes, a adoção de boas práticas de gerenciamento e desenvolvimento de *software* tornou-se essencial para garantir a qualidade, confiabilidade e alinhamento dos processos de TI com os objetivos estratégicos. Nesse contexto, o ITIL (*Information Technology Infrastructure Library*) em sua 4ª edição emergiu como um conjunto abrangente de diretrizes para aprimorar a entrega de serviços de TI, incluindo o gerenciamento de *software*.

Este trabalho propõe uma investigação aprofundada sobre o estado atual das práticas de gerenciamento e desenvolvimento de *software* nas organizações, utilizando como referência o *framework* ITIL4. O foco central recai sobre o desenvolvimento de um *framework* que permita avaliar o nível de maturidade dessas práticas, possibilitando uma análise objetiva e direcionada para a melhoria contínua.

O resultado esperado deste projeto é um instrumento prático e orientado para a ação, que permitirá às organizações identificar áreas de melhoria, lacunas e oportunidades dentro de seus processos internos. A relevância desse trabalho reside na contribuição

para a otimização das práticas de TI, alinhando-as com as diretrizes do ITIL4 e, consequentemente, melhorando a qualidade dos produtos de *software* entregues. Em um ambiente tecnológico em constante evolução, esse projeto busca capacitar as organizações a alcançar níveis mais elevados de excelência em suas operações de TI, resultando em benefícios tanto para as empresas quanto para seus usuários finais.

Em seguida, a seção 2 apresenta os conceitos fundamentais para a assimilação do tema, tal como trabalhos e pesquisas correlatas; o capítulo 3 descreve os objetivos que tentam ser alcançados com o entendimento dos conceitos apresentados na seção 2; a seção 4, por sua vez, define os métodos e técnicas utilizados para que o objetivo proposto seja alcançado; na seção 5 será apresentado o cronograma que ditará o rumo das atividades feitas durante o período de composição do trabalho; por fim, no capítulo 6 é descrito as contribuições e resultados que são esperados com a conclusão do projeto.

2. Fundamentação Teórico-Metodológica e Estado da Arte

Nesta seção serão apresentados conceitos introdutórios e necessários para fomentar a base do desenvolvimento deste trabalho.

2.1. *Information Technology Infrastructure Library* (ITIL) 4

A ITIL (*Information Technology Infrastructure Library*) foi criada pelo governo britânico no *Office of Government Commerce* (OGC), na década de 1980 com o objetivo de estabelecer um padrão para o gerenciamento dos processos da área de Tecnologia da Informação (TI) de seus departamentos. A princípio esse método seria utilizado pelas organizações do setor público a fim de garantir bons resultados tanto na qualidade quanto no custo. Segundo [6], a ITIL preocupa-se, basicamente, com a entrega e o suporte aos serviços de forma apropriada e aderente aos requisitos do negócio, é o modelo de referência para gerenciamento dos serviços de TI mais aceito mundialmente.

[4] diz que a ITIL não define os processos para serem implementados na área de TI na empresa, mas oferece uma base para colocar os processos já existentes em um contexto estruturado, validando tarefas, atividades, procedimentos e regras da organização. Tais práticas podem ser adotadas da forma que melhor atender às necessidades de cada organização.

A quarta edição do ITIL apresenta um modelo operacional que facilita a entrega de produtos e serviços tecnológicos. Esta edição incorpora abordagens contemporâneas e flexíveis em comparação com sua predecessora, a ITILv3 (2011), alinhando-se com as tendências atuais do mercado. Ela também integra práticas modernas de gestão de serviços, como *Agile*, *DevOps* e *Lean*, para uma abordagem mais dinâmica e eficaz. [2]

Os elementos centrais da ITIL4 compreendem o Sistema de Valor de Serviço (SVS) e o modelo de Quatro Dimensões. O SVS detalha como os diversos componentes e atividades de uma organização operam de forma sinérgica para promover a geração de valor em relação aos serviços que envolvem Tecnologia da Informação. Os componentes principais do SVS são: [2]

- a cadeia de valor de serviços;
- as práticas ITIL;
- os princípios orientadores ITIL;

- governança;
- melhoria contínua.

A figura 1 retirada de [7] a seguir, ilustra o SVS

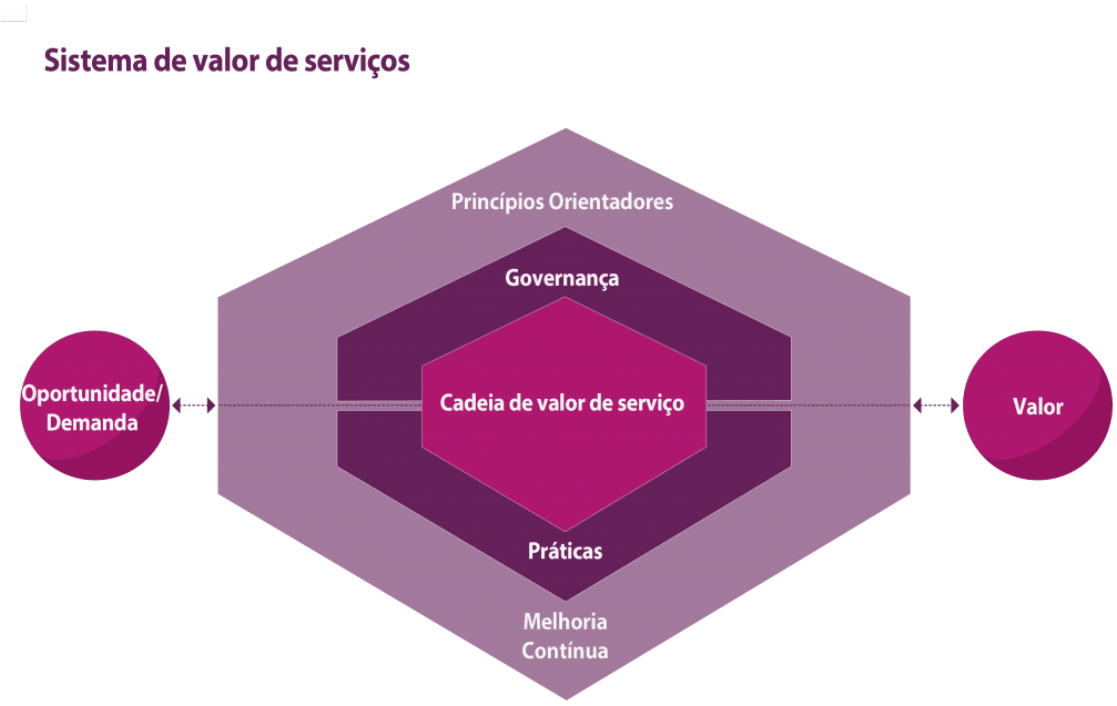

Figura 1. SVS ITIL4

O ITIL4 ainda define um sistema de gerenciamento de serviços de quatro dimensões, sendo elas críticas para a geração de valor para os clientes, enumeradas a seguir: [2]

- organizações e pessoas;
- informação e tecnologia;
- parceiros e fornecedores;
- fluxos de valores e processos;

2.2. Gerenciamento e Desenvolvimento de *Software*

O ITIL4, adota algumas práticas para gerenciar serviços de TI. Uma prática de gestão é um conjunto de recursos organizacionais projetados para realizar um trabalho ou alcançar um objetivo. O ITIL4 inclui 34 práticas de gerenciamento. Para cada prática, há vários tipos de orientação, como termos e conceitos-chave, fatores de sucesso, atividades-chave ou objetos de informação. As práticas são divididas em três categorias, sendo elas [2]

1. Práticas gerais de gerenciamento
2. Práticas de gerenciamento de serviço
3. Práticas de gerenciamento técnico

No presente trabalho, focaremos na prática de Gerenciamento e Desenvolvimento de *Software*, cujo propósito é assegurar que os aplicativos atendam às necessidades das partes interessadas internas e externas. Os aplicativos de *software*, desenvolvidos internamente ou em parceria com um fornecedor, desempenham um papel crucial na entrega de valor aos clientes em negócios orientados por serviços tecnológicos. Por consequência, o desenvolvimento e gerenciamento de *software* é uma prática chave em qualquer organização de TI moderna, garantindo adequação dos aplicativos para suas finalidades e usos. A prática de desenvolvimento e gerenciamento de *software* envolve atividades como: [2]

- Arquitetura da solução.
- Design da solução (incluindo interface do usuário, experiência do cliente, design de serviço, etc.).
- Desenvolvimento de *software*.
- Teste de *software* (abrangendo testes de unidade, integração, regressão, segurança da informação e aceitação).
- Gerenciamento de repositórios de código ou bibliotecas para manter a integridade dos artefatos.
- Criação de pacotes para eficiente implantação dos aplicativos.
- Controle de versão, compartilhamento e contínuo gerenciamento de blocos menores de código.

Ainda segundo [2], os dois modelos de desenvolvimento aceitos para o desenvolvimento de *software* são referidas como modelo prescrito e ágil. O gerenciamento de *software* é uma prática mais ampla, abrangendo as atividades contínuas de projetar, testar, operar e melhorar aplicativos de *software* para que continuem para facilitar a criação de valor. Os componentes de *software* podem ser continuamente avaliados usando um ciclo de vida que rastreia o componente desde a concepção até o contínuo melhora e, eventualmente, aposentadoria.

A qualidade de *software* é utilizada para descrever o *software* como produto e em seu uso, comumente em termos como: [1]

- qualidade do produto: adequação funcional, eficiência de desempenho, compatibilidade, usabilidade, confiabilidade, segurança, manutenibilidade e portabilidade;
- qualidade em uso: eficácia, eficiência, satisfação, ausência de riscos e cobertura de contexto.

Muitos dos requisitos para essas características de qualidade de *software* são insumos para o desenvolvimento e a gestão de software. Eles são determinados pelo proprietário do *software*. O componentes mais importantes para o fator de sucesso desta prática são: [1]

- compreender o código-fonte, como os vários módulos estão inter-relacionados e a arquitetura da aplicação;
- compreender os requisitos e o contexto em que a aplicação é utilizada;
- criar testes antes da codificação;
- controle efetivo de versão de todos os artefatos da aplicação;
- abordar a tarefa de codificação com plena compreensão de sua tremenda complexidade e respeitar as limitações intrínsecas da mente humana;
- adotar convenções de codificação;

- revisão por pares;
- *feedback* rápido proveniente dos testes, por exemplo, através do uso de testes automatizados, e tomar medidas corretivas rapidamente.

2.3. Modelos de desenvolvimento prescritivo

Conforme [10], os modelos de processos prescritivos têm como finalidade descrever as atividades a serem realizadas no processo de desenvolvimento de *software*. Essa categoria de modelos visa transformar a atividade de criação de sistemas, que costumava ser um processo artesanal, em um procedimento bem estruturado, documentado e de alta qualidade, apto a ser incorporado às operações de produção empresarial. Esses processos são estruturados seguindo determinados padrões e abordagens, conhecidos como ”ciclos de vida”. Entre os principais paradigmas prescritivos tradicionais estão o Modelo Cascata, também conhecido como *Waterfall*, o Modelo Espiral, o Modelo Incremental e o Modelo de Prototipagem.

O Modelo Cascata sugere uma abordagem sequencial e sistemática para o desenvolvimento de *software*. [10] destaca que este aa origem deste modelo remete à década de 1970 e sua filosofia é fundamentada no conceito de BDUF (*Big Design Up Front*, “design completo antes de tudo”, em português). Este princípio consiste em elaborar uma análise e design detalhados antes de iniciar a codificação. Desta forma, quando o código for efetivamente produzido, ele estará mais o próximo possível dos requisitos alinhados com o cliente. As etapas deste modelo estão ilustrados na figura 2, retirada de [3].

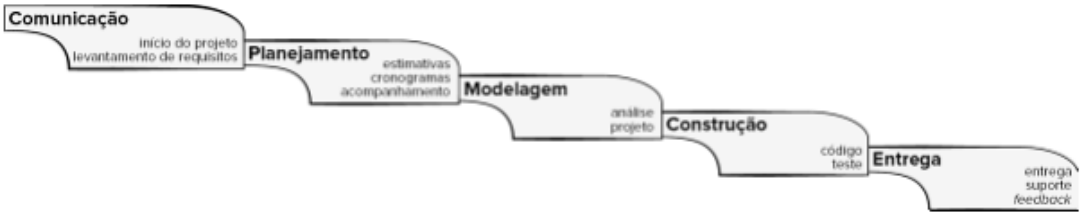

Figura 2. Modelo Cascata

Ao final de cada fase, o *Waterfall* incorpora uma atividade de revisão, garantindo que o projeto progrida de maneira adequada para a próxima fase. Caso o projeto não esteja pronto para avançar, deverá permanecer na mesma fase. Embora o modelo tenha sido amplamente utilizado no passado, ele também apresenta suas desvantagens. Os problemas mais frequentes encontrados são [3].

- em projetos reais dificilmente ocorre um fluxo sequencial como propõe o modelo;
- o processo de comunicação com o usuário frequentemente não aponta de forma explicita suas necessidades devido principalmente a uma incerteza natural no início do projeto, onde muitos detalhes são deixados de lado;
- o cliente não receberá uma versão operacional do sistema antes do final do projeto. Assim, um erro grave que não seja identificado até a revisão do sistema operacional pode gerar consequências desastrosas.

O modelo incremental surgiu como uma alternativa para suprir alguns dos problemas do Modelo Cascata, combinando elementos de seu predecessor com etapas interativas, cujo objetivo é apresentar um produto operacional a cada incremento realizado. Cada incremento passa por todas as etapas do *Waterfall*: comunicação, planejamento, modelagem, construção e disponibilização. Seu uso viabiliza que organizações sem acesso imediato a recursos humanos para o desenvolvimento integral de um sistema possam realizar mudanças futuras em momentos oportunos.

Em relação ao modelo de prototipagem, ele emprega o conceito de prototipação, com o proposito de desenvolver, de forma ágil e econômica, versões de interface do projeto para que sejam avaliadas e testadas diversas alternativas pelo cliente antes da entrega do produto. Segundo [3], o paradigma da prototipação é particularmente vantajosa em cenários nos quais o cliente não consegue definir detalhadamente os requisitos das funcionalidades e recursos desejados, ou ainda, quando o programador está inseguro quanto à eficácia do algoritmo implementado ou quando há dúvidas quanto à adaptabilidade do sistema operacional e a interação homem-máquina.

O Modelo Espiral concentra as qualidade mais vantajosas do ciclo de vida clássico e de prototipação, agregando um componente adicional: a análise riscos. Ele usa uma abordagem "evolucionária" que capacita tanto os desenvolvedores quanto os clientes a compreenderem e reagirem aos riscos em cada fase evolutiva, Aproveitando-se da prototipagem como uma ferramenta para mitigar riscos, ele também oferece a flexibilidade de empregar essa abordagem em qualquer estágio do desenvolvimento do projeto. Conforme mencionado por [3], o Modelo Espiral é considerado um modelo de processo de *software* evolucionário que une a natureza iterativa da prototipação aos aspectos sistêmicos e controlados do Modelo Cascata.

2.4. Modelos de desenvolvimento ágil

Em 2001, um grupo formado por dezessete pesquisadores e profissionais de TI se reuniram para discutir as técnicas existentes para projetos de desenvolvimento de *software*. O objetivo deles era descobrir novos valores e princípios para melhorar o desenvolvimento de projetos. O acontecimento mais importante em relação à este movimento foi a assinatura do Manifesto Ágil. [5]

Os valores e princípios ágeis promovidos por essa aliança e a sua relação com os valores tradicionais foram apresentados com o proposito de auxiliar as equipes e aprimorar entrega de produtos de *software*. Com a popularização desses métodos, passaram a ser utilizados em diversas outras áreas de negócio, como criação de empresas e desenvolvimento de produtos. Os princípios descritos no Manifesto e considerados os pilares para os métodos ágeis são: [5]

1. **Indivíduos e interações** mais que processos e ferramentas;
2. ***Software* em funcionamento** mais que documentações abrangentes;
3. **Colaboração com o cliente** mais que negociação de contratos;
4. **Responder a mudanças** mais que seguir um plano.

2.4.1. *Scrum*

O Scrum é um *framework* que se baseia nos valores e princípios do Manifesto Ágil para o desenvolvimento de *software*. A sua concepção parte do reconhecimento de que o desenvolvimento de *software* é intrinsecamente imprevisível, dado o grande número de variáveis técnicas e ambientais que podem sofrer alterações ao longo do ciclo de desenvolvimento. Isso resulta em uma elevada incerteza associada a esses projetos. O Scrum foi idealizado com proposito de gerenciar e solucionar problemas associados a projetos complexos e adaptativos especialmente aqueles de natureza inovadora. Assim, o foco deste *framework* está em encontrar uma abordagem flexível para produzir um produto de *software*, tendo em mente que as mudanças e as dificuldades inerentes ao produto final são uma constante no processo. [8]

O Scrum possui três pilares fundamentais: [9]

1. **Transparência:** aspectos significativos do processo devem estar visíveis aos responsáveis pelos resultados;
2. **Inspeção:** os usuários Scrum devem, frequentemente, inspecionar os artefatos Scrum e o progresso em direção a detectar variações;
3. **Adaptação:** devido a mudanças constantes no projeto, o projeto pode sofrer alterações, de acordo com as necessidades da empresa.

Conforme mencionado em [9], o Scrum envolve três papeis fundamentais: O *Product Owner*, o Scrum *Master*, o e os *Developers*. O *Product Owner* desempenha o papel de liderança, sendo responsável por decisões relacionadas ao produto. Ele define as funcionalidades e recursos que devem ser desenvolvidos juntamente com sua ordem de implementação. Além disso, o *Product Owner* também é responsável pela comunicação dos objetivos do projeto à equipe. Por sua vez, o Scrum *Master* é a pessoa responsável por apoiar todos membros do time, garantindo que todos compreendam plenamente os valores, princípios e práticas Scrum. Ele lidera os processos e auxilia a equipe no desenvolvimento de suas próprias abordagens. Os *Developers* são os indivíduos encarregados da construção efetiva do projeto. No contexto do Scrum, eles têm autonomia para decidir como as tarefas devem ser executadas, em vez de dependerem de liderança direta. A equipe se auto-organiza para alcançar as metas estabelecidas pelo *Product Owner*.

Ainda seguindo o Guia do Srum [9], o Scrum é composto por uma série de eventos e artefatos que, em conjunto, definem o ciclo de vida de desenvolvimento de *software*. Esses elementos incluem: *Product Backlog, Sprint Backlog, Sprint* e Retrospectiva da *Sprint*. O ciclo de vida tem inicio com a definição das funcionalidades principais para o *Product Backlog*, seguido pela *Sprint Planning*, onde são estabelecidos os conjuntos de tarefas a serem realizados nas *Sprints* que têm uma duração média de duas a quatro semanas. Durante o desenvolvimento da *Sprint*, o time Scrum realiza reuniões diárias para monitorar o progresso do projeto. Conclui-se o ciclo de vida com a retrospectiva e revisão da *Sprint*.

3. Objetivos

O objetivo principal do presente trabalho é desenvolver um *framework* abrangente e eficaz para avaliar o nível de maturidade das práticas de gerenciamento e desenvolvimento de *software*, conforme preconizado pelo ITIL 4. Este *framework* visa fornecer às

organizações uma ferramenta sólida e estruturada para avaliar sua capacidade de gerenciamento de *software* e identificar áreas que requerem aprimoramento. Ao criar um método sistemático de diagnóstico utilizando a ferramenta GAIA, espera-se melhorar a qualidade do *software* produzido, aumentar a eficiência operacional e, por fim, agregar mais valor aos clientes.

4. Procedimentos metodológicos/Métodos e técnicas

Inicialmente será feita uma revisão bibliográfica de trabalhos, livros e artigos já existentes sobre o Gerenciamento e Desenvolvimento de *Software* com base no ITIL 4, além de metodologias prescritivas e ágeis para desenvolvimento de *software*. Em seguida será iniciado o desenvolvimento do *framework*, tendo como primeira etapa o diagnóstico do ambiente organizacional, conforme figura 3. Este ambiente subsidia a aplicação do mecanismo de análise, que, por sua vez, necessita que as informações sobre a organização sejam extraídas por meio de um questionário. Após a obtenção das informações organizacionais, o mecanismo de análise compila o resultado, baseando-se em eixos de eficiência que serão elencados em uma outra etapa do projeto.

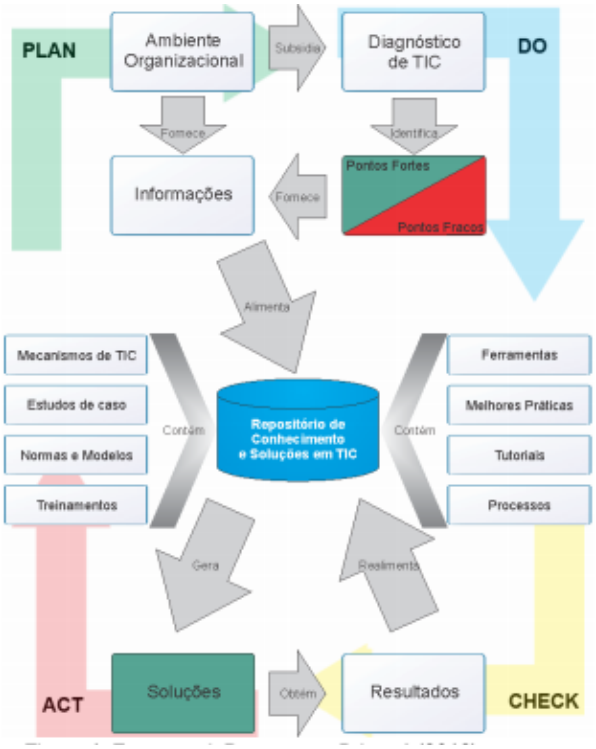

Figura 3. *Framework* Proposto pelo Grupo de Pesquisa em Engenharia de *Software* e Banco de Dados do Departamento de Computação da UEL

Como resultado da análise, são produzidos indicadores que apontam quais os eixos que precisam ser melhorados. Na Figura 4, é apresentado um exemplo de resultado de diagnóstico utilizando seis eixos. Esta informação é utilizada posteriormente para selecionar os itens que deverão ser atacados visando a melhoria do contexto avaliado.

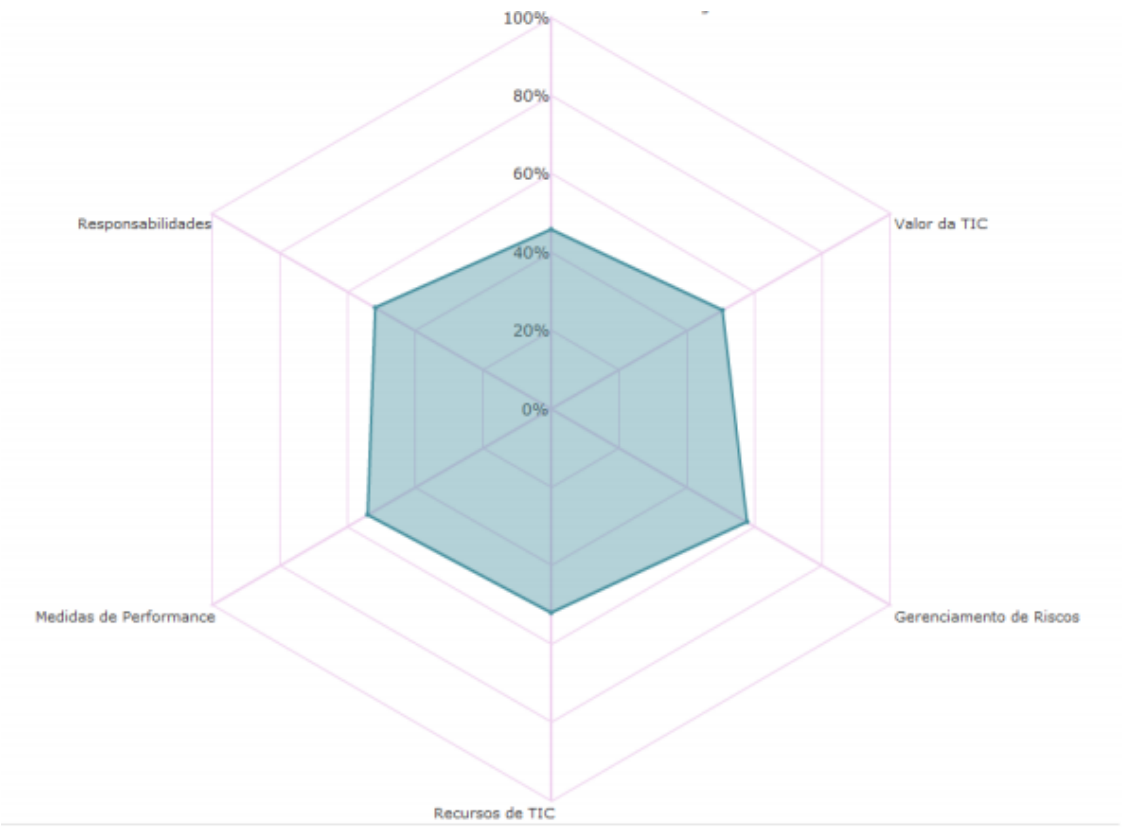

Figura 4. Resultado do Diagnóstico da Governança de TI a partir da análise das respostas do questionário

Por meio da ferramenta produzida, podemos avaliar não somente a questão da governança, mas sermos mais específicos em diagnosticar como a área de TI gerencia riscos, trata do catálogo de serviços, gerencia fornecedores de TI, enfim, podemos fazer vários diagnósticos para subsidiar o caminho da implantação da Governança de TI mais adequado no processo de Gerenciamento e Desenvolvimento de *Software* em uma empresa.

5. Cronograma de Execução

A Tabela 1 exibe o cronograma de atividades a serem desenvolvidas.

Atividades:

1. Levantamento bibliográfico;
2. Elencar eixos de importância;
3. Fazer perguntas e respostas atreladas aos eixos;
4. Usar a ferramenta Gaia;
5. Avaliação da ferramenta implementada;
6. Correções e ajustes do modelo;
7. Gerar o grau de maturidade;
8. Escrita do relatório.

Tabela 1. Cronograma de Execução

	ago	set	out	nov	dez	jan	fev	mar	abr
Atividade 1	X	X	X						
Atividade 2			X	X					
Atividade 3				X	X				
Atividade 4			X	X	X	X	X	X	
Atividade 5						X			
Atividade 6						X	X		
Atividade 7								X	
Atividade 8		X	X	X	X	X	X	X	X

6. Contribuições e/ou Resultados esperados

Com o presente trabalho, espera-se que o desenvolvimento do *Framework* de Gerenciamento e Desenvolvimento de *Software* utilizando a ferramenta Gaia forneça uma abordagem estruturada e objetiva para avaliar a maturidade das organizações em relação às práticas do ITIL4 permitindo a identificação de áreas de melhoria e o estabelecimento de estratégias de aprimoramento. Além disso, este projeto visa contribuir para a disseminação e adoção do ITIL4 no contexto de desenvolvimento de *software*, preenchendo uma lacuna de conhecimento nesse campo. Os resultados esperados incluem a validação e aplicação prática do modelo em organizações reais, evidenciando sua eficácia e utilidade na melhoria dos processos de gerenciamento e desenvolvimento de *software*. Espera-se que essa pesquisa beneficie tanto profissionais da área quanto as organizações, promovendo uma abordagem mais madura e eficaz na entrega de serviços de TI e no desenvolvimento de *software* de alta qualidade

7. Espaço para assinaturas

Londrina, *18 de Setembro de 2023*.

_____ _____

Aluno Orientador

Referências

[1] Axelos. Software development and management: Itil 4 practice guide. `https://www.axelos.com/resource-hub/practice/software-development-and-management-itil-4-practice`. Acessado em Setembro de 2023.

[2] Axelos. *ITIL Foundation - ITIL 4 Edition*. The Stationery Office, 4th edition, 2019.

[3] Roger S. Pressman e Bruce R. Maxim. *Engenharia de Software: uma abordagem profissional*. AMGH, 9th edition, 2021.

[4] Ivan Luizio Magalhães e Walfrido Brito Pinheiro. *Gerenciamento de serviços de TI na prática: uma abordagem com base na ITIL : inclui ISO/IEC 20.000 e IT Flex*. Novatec Editora, 2007.

[5] Beck et al. Manifesto for agile software development. `https://agilemanifesto.org/`, 2021. Acessado em Setembro de 2023.

[6] Ricardo Mansur. *Governança de TI: Metodologias, Frameworks e Melhores Práticas*. Brasport, 2007.

[7] ITSM NA PRÁTICA. Itil®: o que é, para que serve e como tirar a certificação. `https://www.itsmnapratica.com.br/tudo-sobre-itil/`. Acessado em Agosto de 2023.

[8] Ken Schwaber. *Agile Project Management with Scrum*. Microsoft Press, 2004.

[9] Ken Schwaber and Jeff Sutherland. The 2020 scrum guide ™. `https://scrumguides.org/scrum-guide.html`. Acessado em Setembro de 2023.

[10] Raul Wazlawick. *Engenharia de software: conceitos e práticas*. GEN LTC, 2nd edition, 2019.