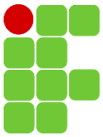


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS

***CAMPUS* PORTO NACIONAL**

LICENCIATURA EM COMPUTAÇÃO

IGOR BLENO GOMES DOS SANTOS

TENDÊNCIAS EMERGENTES DE SUCESSO PARA A ENGENHARIA DE SOFTWARE: Uma revisão sistemática

PORTO NACIONAL

2019

IGOR BLENO GOMES DOS SANTOS

TENDÊNCIAS EMERGENTES DE SUCESSO PARA A ENGENHARIA DE SOFTWARE: Uma revisão sistemática

Monografia apresentada a Coordenação do Curso de Licenciatura em Computação do Campus Porto Nacional, do Instituto Federal do Tocantins, como exigência à obtenção do grau Licenciado em Computação.

Orientador: Prof. Me. Rafael Miranda Correia

PORTO NACIONAL

2019

TENDÊNCIAS EMERGENTES DE SUCESSO PARA A ENGENHARIA DE SOFTWARE: Uma revisão sistemática

Monografia apresentada a Coordenação do Curso de Licenciatura em Computação do Campus Porto Nacional, do Instituto Federal do Tocantins, como exigência à obtenção do grau Licenciado em Computação.

Orientador: Prof. Me. Rafael Miranda Correia

Aprovado em ____/____/_____

BANCA EXAMINADORA

_____

Professor Orientador Me. Rafael Miranda Correia
Instituto Federal de Educação, Ciência e Tecnologia do Tocantins

_____

Professor Me. Deuzelina Tavares Chagas
Instituto Federal de Educação, Ciência e Tecnologia do Tocantins

_____

Professor Me. Elvis Nascimento Silva
Instituto Federal de Educação, Ciência e Tecnologia do Tocantins

A Deus, que se mostrou criador, que foi criativo. Seu fôlego de vida em mim me foi sustento e me deu coragem para questionar realidades e propor sempre um novo mundo de possibilidades.

AGRADECIMENTOS

A Deus por ter me dado saúde e força para superar as dificuldades.

A este Instituto Federal, seu corpo docente, direção e administração que oportunizaram a janela do vislumbro, um horizonte superior eivado pela acendrada confiança no mérito e ética presente.

Ao meu orientador, Prof. Me. Rafael Miranda Correia, pelo suporte no tempo que lhe coube, pelas suas correções e incentivos que culminaram na conclusão deste projeto.

Aos meus familiares, pelo amor, incentivo e apoio incondicional. E a todos que, direta ou indiretamente, fizeram parte da minha formação, o meu muito obrigado.

Estamos em uma era de que quem não tem conhecimento o suficiente da informática é considerado analfabeto.

Gilberto Martini Refatti

RESUMO

O presente trabalho aborda sobre Tendências emergentes de sucesso para a engenharia de software: Uma revisão sistemática, Identificar, verificar e propor o estudo das melhores tendências de sucesso como parte da ementa da disciplina de engenharia de software do curso de Licenciatura em Computação do *Campus* Porto Nacional do IFTO. assim, a problemática evidenciada foi, como atualizar a ementa da disciplina de engenharia de software com as atuais tendências de sucesso no que diz respeito ao uso das engenharias de desenvolvimento de software? Nesse sentido, levantou se a hipótese de que realização de um estudo bibliográfico e de campo, este último dentro do IFTO *campus* Porto Nacional, identificará as atuais tendências de uso das Engenharias de software, e com isso realizar a atualização do plano de ensino dentro da ementa da disciplina. Todas as etapas aqui descritas são frutos de um trabalho minucioso apresentado em formato de monografia. Para a realização deste estudo foram utilizadas pesquisas bibliográficas através de livros, artigos científicos e sites para obtenção de informações e a pesquisa de campo considerada quantitativa, utilizada para coleta de dados. O universo da pesquisa em que o estudo foi desenvolvido está definido com 96 pessoas, destas 57 responderam ao questionário, sendo: 82% discentes, 14% docentes, 2% convidados e 2% visitantes. Os resultados obtidos mostram que 35% disseram que a engenharia mais importante é a engenharia de proteção, 21% disseram que é a engenharia de desenvolvimento ágeis, 19% disseram que é a engenharia baseada em componentes, 13% que é a engenharia baseada em aspecto, enquanto que 12% disseram que é a engenharia de confiança, mostrando com isso a relevância desse trabalho.

Palavras Chaves: Engenharia. Software. Disciplina. Ementa.

ABSTRACT

This paper discusses Emerging Success Trends for Software Engineering: A Systematic Review, Identify, Verify, and Propose the Study of the Best Success Trends as part of the Software Engineering Course Menu of the Bachelor of Computing Course at Porto Campus IFTO National. Thus, the problem highlighted was how to update the software engineering discipline's menu with current successful trends regarding the use of software development engineering? In this sense, it was hypothesized that conducting a bibliographic and field study, the latter within the IFTO campus Porto Nacional, will identify the current trends in the use of software engineering, and thus carry out the updating of the teaching plan within the subject's menu. All the steps described here are the result of a detailed work presented in monograph format. To carry out this study we used bibliographic research through books, scientific articles and websites to obtain information and the field research considered quantitative, used for data collection. The research universe in which the study was developed is defined by 96 people, of which 57 answered the questionnaire: 82% students, 14% teachers, 2% guests and 2% visitors. The results show that 35% said the most important engineering is protection engineering, 21% said it was agile development engineering, 19% said it was component based engineering, 13% said it was aspect based engineering. , while 12% said it is reliable engineering, thereby showing the relevance of this work.

Keywords: Engineering. Software. Subject. Menu.

LISTA DE FIGURAS

LISTAS DE TABELAS

LISTAS DE ABREVIATURAS E SIGLAS

ASD - Adaptive Software Development

BD - Base de Dados

BDTD - Biblioteca Digital Brasileira de Teses e Dissertações

CAPES - Portal de Periódicos, da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

CBSE – Component Based Software Engineering

CS - Critério de Seleção

DFD - Fluxo de Dados

DSDM - Crystal, Dynamic Systems Development Method

EP - Extremming Programming

ES – Engenharia de Software

FDD - Feature Driven Development

IFTO – Instituto De Federal De Ciências e Tecnologia do Tocantins

MER - Modelo Entidade Relacionamento

MVC - Model View Controller

SCIELO - Periódico Científico Eletrônico

TB - Termos de Buscas

TO – Tocantins

SUMÁRIO

1. INTRODUÇÃO

O mundo atual não poderia existir sem o uso dos softwares. Infraestruturas e serviços nacionais são controlados por sistemas computacionais, e a maioria dos produtos elétricos inclui um computador e um software que o controla. A manufatura e a distribuição industriais são totalmente informatizadas, assim como o sistema financeiro. A área de entretenimento, incluindo a indústria da música, jogos de computador, cinema e televisão, faz uso intensivo de software. Portanto, a engenharia de software é essencial para o funcionamento de sociedades nacionais e internacionais (SORMMEVILLE, 2011).

Ao passo em que Pressman (2011) afirma que um software existe por uma única razão: gerar valor aos seus usuários, assim este trabalho visa atualizar o mundo acadêmico quais as atuais tendências de uso das engenharias de Software que se encontra em constantes atualizações e mudanças.

Nos últimos 20 anos, segundo Degoulet (1997) o hardware deixou de ser o item mais caro na implementação de um sistema, enquanto que o custo relacionado ao software cresceu e se tornou o principal item no orçamento da computação. Isso se deve principalmente pela crescente complexidade dos problemas a serem resolvidos pelos softwares. Sistemas como os de gestão hospitalar e sistemas de PEP chegam a possuir milhões de linhas de código e envolvem vários especialistas para o seu desenvolvimento.

Aliado a isso, alguns problemas inerentes ao processo de desenvolvimento de um software começaram a surgir Pressman (2011) : as estimativas de prazo e de custo frequentemente são imprecisas, a produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços e, a qualidade de software às vezes é menos que adequada, ocorrendo muito frequentemente a insatisfação do usuário.

A chave para se vencer esses problemas e dificuldades acima relatados é a larga utilização de uma abordagem de engenharia ao desenvolvimento de software, aliada a uma contínua melhoria das técnicas e ferramentas Degoulet (1997), Pressman (2011), no intuito também de melhorar a produtividade da equipe.

Dessa forma, podemos destacar duas tendências para justificar o uso da ES : primeiro, o software é um item de alto custo e em progressivo aumento; e segundo, que os softwares têm um importante papel no bem-estar da sociedade; dessa forma, a ES assume papel crítico para garantir que tarefas, dados, pessoas e tecnologia estejam apropriadamente alinhadas para produzir um sistema efetivo e eficiente.

O gerenciamento de projetos tem sido muito aplicado no mercado pelo fato de auxiliar na excelência, qualidade e confiabilidade no processo de desenvolvimento de projetos. Ele tem como objetivo buscar projeções de tempo, custo, recursos, qualidade e aquisições necessárias.

Dentre as áreas de conhecimento que compõem o gerenciamento de projetos, De Pádua (2003) informa que o gerenciamento de tempo atualmente é um dos pontos fundamentais, pois projeta para o cliente, a partir de levantamentos das atividades e recursos, qual será o tempo gasto para desenvolver o projeto através de um cronograma. Atrasos na entrega podem trazer consequências negativas para a profissional desde quebra de contrato, multa por atraso e prejuízos na imagem deste.

Conforme Miranda (2019), Como uma resposta às crescentes pressões por inovação em prazos cada vez mais reduzidos, às necessidades de constantes mudanças de requisitos e ao mau desempenho de grande parte dos projetos de desenvolvimento de software, esta pesquisa busca, atualizar os profissionais, professores e estudantes nesse seguimento.

Sendo assim, neste trabalho são apresentados os históricos, os conceitos, os valores e os princípios que norteiam as engenharias de desenvolvimento, assim como são descritos os métodos e camadas das etapas de desenvolvimento de software.

1.1. Problemática da Pesquisa

Como atualizar a ementa da disciplina de engenharia de software com as atuais tendências de sucesso?

1.2. Objetivos

Nesta seção estão descritos os objetivos geral e específicos a serem alcançados através desta pesquisa.

1.2.1. Objetivo Geral

Identificar, verificar e propor o estudo das melhores tendências de sucesso como parte da ementa da disciplina de engenharia de software do curso de Licenciatura em Computação do Campus Porto Nacional do IFTO.

1.2.2. Objetivos Específicos

a) Identificar as tendências de sucesso dentro da engenharia de software.

b) Verificar junto aos docentes e discentes do curso de Licenciatura em Computação do Campus Porto Nacional do IFTO, quais tendências seriam mais importantes para o estudo junto a disciplina de engenharia de software.

c) Propor o estudo das melhores tendências de sucesso como parte da ementa da disciplina de engenharia de software do curso de Licenciatura em Computação do Campus Porto Nacional do IFTO.

1.3. Hipótese

A hipótese levantada que solucionará a problemática constitui-se na realização de um estudo bibliográfico e de campo, este último dentro do IFTO *campus* Porto Nacional, a fim de identificar as atuais tendências de uso das Engenharias de software e com isso propor a atualização do plano de ensino dentro da ementa da disciplina.

1.4. Justificativa

A relevância deste trabalho está em apresentar uma abordagem multidisciplinar visando Identificar, verificar e propor o estudo das melhores tendências de sucesso como parte da ementa da disciplina de engenharia de software do curso de Licenciatura em Computação do Campus Porto Nacional do IFTO, uma vez que, se trata de um tema pouco explorado pela parte acadêmica. Nesse sentido, tende a apresentar uma melhoria junto ao plano de ensino da disciplina de Engenharia de Software do Instituto Federal de Educação, Ciência e Tecnologia *Campus* de Porto Nacional. Com isso o futuro Licenciado em computação sairá melhor qualificado para o mercado de trabalho.

Portanto, o presente trabalho ganha ênfase, pois dentro da literatura pesquisada não há outro igual ou que se assemelhe com esse, quanto a sua proposta, tema, problema e objetivos.

1.5. Organização do Trabalho

Para melhor entendimento, o presente trabalho está organizado em cinco capítulos:

No Capítulo 1, é apresentada a contextualização, o tema, a definição do problema, os objetivos a serem alcançados, a hipótese levantada, a justificativa para realização do presente estudo e, por fim, a presente organização do trabalho. No Capítulo 2, é apresentado o referencial teórico, em que são abordados elementos que proporcionam o suporte conceitual necessário para a elaboração da atual pesquisa. Os elementos abordados são: 2.1.Engenharia de Software, 2.1.1.Processo, 2.1.2.Método, 2.1.3.Ferramentas, 2.1.4.Tendências emergentes de sucesso para a engenharia de software, 2.1.4.1.Engenharia de Desenvolvimento ágeis, 2.1.4.2.Engenharia de confiança, 2.1.4.3.Engenharia de proteção, 2.1.4.4.Engenharia baseada em componentes, 2.1.4.5.Engenharia orientada a aspecto, 2.2.Ementa da disciplina de Engenharia de Software do IFTO campus Porto nacional – TO., 2.2.1.Plano de ensino da disciplina de Engenharia de Software, 2.3.Trabalhos correlacionados e 2.4. Comentários Parciais. No Capítulo 3, está descrito de forma sistematizada os procedimentos metodológicos, onde estão apresentadas de forma minuciosa, todas as etapas do desenvolvimento deste estudo: são apresentados os materiais e métodos. No Capítulo 4, são abordadas as análises e resultados da pesquisa, discussões. No Capítulo 5, é apresentado as conclusões as quais se chegaram após o presente estudo, bem como as perspectivas e propostas para trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresentará os aspectos teóricos sobre as áreas envolvidas no presente trabalho. São apresentados conceitos sobre: Engenharia de Software, Tipos de Engenharias de Software, Tendências emergentes de sucesso para a engenharia de software e Ementa da disciplina de Engenharia de Software do IFTO *campus* Porto Nacional – TO.

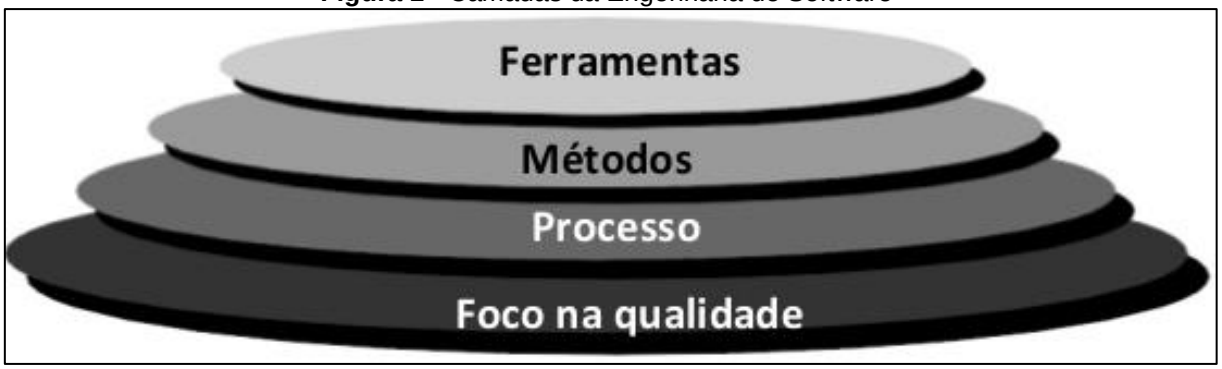
2.1. Engenharia de Software

A engenharia de software (ES) segundo Sommerville (2011), tem por objetivo apoiar o desenvolvimento profissional de software, mais do que a programação individual. Ela inclui técnicas que apoiam especificação, projeto e evolução de programas.

Ainda em Sommerville (2011), ES é uma disciplina de engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado.

Do mesmo modo, Pressman (2010) diz que a ES é uma abordagem sistemática e disciplinada para o desenvolvimento de software. E que a ES é uma tecnologia em camadas: processos, métodos e ferramentas. E a base de todas essas camadas é o foco na qualidade do software desenvolvido conforme a figura a baixo.

Figura 1 - Camadas da Engenharia de Software



Fonte: PRESSMAN (2010).

Conforme a figura 1, Pressman (2010) observa que todo o foco desta disciplina é na qualidade, que é a base de todas as camadas. O alicerce da ES, para tal, fica sendo no Processo, onde a partir daí tem-se os Métodos a serem aplicados, e as Ferramentas como apoio a todo esse esquema. deste conjunto é conhecido paradigma de ES.

2.1.1. Processo

O alicerce da ES é a camada de processo. O processo de engenharia de software é o adesivo que mantém unidas as camadas de tecnologia e permite o desenvolvimento racional e oportuno de softwares de computador (REZENDE, 2005).

O Processo de Software é um conjunto de atividades, métodos, práticas e transformações ordenadas com a intenção de atingir a qualidade do software. Sua meta fundamental é entregar, de maneira eficiente e previsível um produto de software capaz de atender as necessidades de negócio, definidas pela análise de requisitos, dos usuários.

Existem muitos processos de *software* diferentes, mas todos devem incluir quatro atividades fundamentais para a engenharia de *software* (SOMMEVILLE 2011). Conforme descrito na Tabela 1.

Tabela 1 – Atividades para processo de software

Atividades	Descrição
1. Especificação de software.	A funcionalidade do software e as restrições a seu funcionamento devem ser definidas.
2. Projeto e implementação de software.	O software deve ser produzido para atender às especificações.
3. Validação de software.	O software deve ser validado para garantir que atenda às demandas do cliente.
4. Evolução de software.	O software deve evoluir para atender às necessidades de mudança dos clientes.

Fonte: Sommerville, 2011, adaptada pelo autor.

2.1.2. Método

Método é uma palavra que vem do grego *métodos* Ferreira (1999), que significa caminho para se chegar a um fim. O termo metodologia é bastante controverso nas ciências em geral e na Engenharia de Software em particular. Muitos autores parecem tratar metodologia e método como sinônimos Pressman (2006), porém seria mais adequado dizer que uma metodologia envolve princípios filosóficos que guiam uma gama de métodos que utilizam ferramentas e práticas diferenciadas para realizar algo.

Já as metodologias de Engenharia de Software objetivam ensinar como fazer para construir softwares. Esses métodos incluem atividades de modelagem, construção de programas, testes e manutenção (SOMMERVILLE, 2005). Na Engenharia de Software as principais abordagens de Metodologias são: Metodologia

Estruturada (Utiliza como ferramental Dicionário de Dados, Diagrama de Fluxo de Dados (DFD), e o Modelo Entidade Relacionamento (MER); Metodologia Orientada a Objetos e Metodologias de Desenvolvimento Ágil.

2.1.3. Ferramentas

Ferramenta é uma palavra que vem do latim *ferramentum* significando qualquer utensílio empregado nas artes e ofícios (FERREIRA, 1999). As ferramentas de Engenharia de Software segundo Pressman (2010) fornecem apoio automatizado, ou semi-automatizado, para o processo e para os métodos. Quando ferramentas são integradas de modo que a informação criada possa ser usada por outra, é estabelecida a chamada engenharia de software apoiada por computador.

Uma das grandes dificuldades da engenharia do software é resolver o problema e deixar o cliente satisfeito com o software. Nesse sentido, Jalote (2005) argumenta que o desenvolvimento de um projeto de software quanto mais complexo é o software, maior é o empenho que o engenheiro de software deve fazer para desenvolver e tem que ter maior gerenciamento.

2.1.4. Tendências emergentes de sucesso para a engenharia de software

Nos dias de hoje, as empresas operam em um ambiente global, com mudanças rápidas. Assim, Sommerville (2011) argumenta que essas empresas precisam responder a novas oportunidades e novos mercados, a mudanças nas condições econômicas e ao surgimento de produtos e serviços concorrentes. Softwares fazem parte de quase todas as operações de negócios, assim, novos softwares são desenvolvidos rapidamente para obterem proveito de novas oportunidades e responder às pressões competitivas. Pressman (2010) afirma que o desenvolvimento e entrega rápidos são, portanto, o requisito mais crítico para o desenvolvimento de sistemas de software. Na verdade, muitas empresas estão dispostas a trocar a qualidade e o compromisso com requisitos do software por uma implantação mais rápida do software de que necessitam.

2.1.4.1. Engenharia de Desenvolvimento ágeis

Os Métodos Ágeis de Desenvolvimento de Software surgiram como uma reação aos métodos clássicos de desenvolvimento e do reconhecimento da necessidade premente de se criar uma alternativa a estes “processos pesados”, caracterizados pelo foco excessivo na criação de uma documentação completa (BECK, et al, 2005). Em meados dos anos 90, integrantes da comunidade de

desenvolvimento de software começaram a questionar estes processos, julgando-os pouco efetivos e, muitas vezes, impossíveis de serem colocados em prática (HIGHSMITH, 2001).

Sintetizando o pensamento deste grupo, Highsmith (2001) menciona que a indústria e a tecnologia sofrem modificações tão aceleradas que acabam por "atropelar" os métodos clássicos. Highsmith *et al* (2002) ainda acrescentam que os clientes, na maioria das vezes, são incapazes de definir de forma clara e precisa, os requisitos do software, logo no início de um projeto de desenvolvimento, o que inviabiliza a adoção dos métodos clássicos em muitos projetos.

Como resposta a esta situação, muitos especialistas criaram métodos próprios para se adaptar às constantes mudanças exigidas pelo mercado e às indefinições iniciais dos projetos. O agrupamento desses métodos deu origem à família dos Métodos Ágeis de Desenvolvimento de Software. Sendo assim,

> [...] os Métodos Ágeis podem ser considerados uma coletânea de diferentes técnicas e métodos, que compartilham os mesmos valores e princípios básicos, alguns dos quais remontam de técnicas introduzidas em meados dos anos 70, como os desenvolvimentos e melhorias iterativos (COHEN et al, 2003, p.2).

Existem diversos modelos ágeis, como: *Adaptive Software Development (ASD), Crystal, Dynamic Systems Development Method (DSDM), Extremming Programming, Feature Driven Development (FDD) e Scrum* (ABRAHAMSSON et al., 2003). Contudo, eles possuem características especificas, conforme apresentadas na Tabela 2.

Tabela 2 - Principais Métodos Ágeis de Desenvolvimento de Software

Nº	Processos de Software	Descrição
01	***Adaptive Software Development* - Desenvolvimento de Software Adaptativo**	Foca na busca de objetivos, na participação e na colaboração da equipe, com comunicação constante, garantindo aprendizado durante o projeto (HIGHSMITH, 1996).
02	***Dynamic Systems Development Method* – Método Dinâmico de Desenvolvimento de Sistemas**	Defende o processo de desenvolvimento com interações e ciclos incrementais (STAPLETON, 1997).
03	***Extreme Programming* - Programação Extrema - PX**	Centra o desenvolvimento de projetos em: planejamento, entregas frequentes, descrições metafóricas, projetos simples, testes, refatoração (simplificação de funcionalidade), desenvolvimento em duplas, acessível a todos os membros da equipe, com interações

		continuas, presença de cliente no processo e padronização de código (BECK; ANDRES, 2004).
04	**Feature-driven Development – Desenvolvimento Dirigido a Funções**	Construção de escopo de funcionalidades necessárias para satisfazer o cliente, ordenadas por prioridades. Um plano de desenvolvimento é realizado para cada funcionalidade, gerando uma estrutura de classes associada aos desenvolvedores responsáveis (SILVA et al., 2009).
05	**Scrum**	Divide o desenvolvimento em iterações, chamadas sprints. Como a equipe é pequena e claramente definida, formada por facilitador e equipe, é também a responsável pelos requisitos (SCHWABER, BEEDLE, 2002).
06	**Transformação**	A transformação de modelos em cada fase do desenvolvimento permite automatizar o processo de desenvolvimento, evitando erros e bugs. É pensar em 'o que deve ser feito' e não em 'como deve ser feito'. Além de permitir maior qualidade do software, permite que o desenvolvimento seja mais rápido (LAPENDA, LONIEWSKI, MADRUGA, 2008).
07	**Desenvolvimento Rápido de Aplicação - RAD**	permite uma prototipagem mais rápida e entrega iterativa do produto final. Trata-se de um modelo alternativo à tradicional modelo cascata que, em geral, foca em um processo de desenvolvimento sequencial e pouco flexível (MAGRI, 2008).

Fonte: Adaptado de Silva et al. (2013).

2.1.4.2. Engenharia de confiança

A engenharia de confiança segundo Sommerville (2011), está preocupada com as técnicas para aumentar a confiança de ambos os sistemas, críticos e não críticos. Essas técnicas suportam três abordagens complementares que são usadas no desenvolvimento de softwares confiáveis.

a). Prevenção de defeitos. O processo de projeto e implementação de software Bartié (2002), sugere que deve usar abordagens de desenvolvimento de software que ajudem a evitar erros de projeto e programação e, assim, Koscianski e

dos Santos (2007) diz que precisa minimizar o número de defeitos que possam surgir quando o sistema estiver em execução. Poucos defeitos significam menores chances de falhas durante a execução.

a). Detecção e correção de defeitos. Para Barros et al. (2019), os processos de verificação e validação são projetados para descobrir e remover defeitos em um programa, antes que este seja implantado para uso operacional. Sistemas críticos segundo Ramos e Amora (2019), exigem extensa verificação e validação para se descobrir o maior número possível de defeitos antes da implantação e para convencer os *stakeholders* de que o sistema é confiável.

c). Tolerância a defeitos. Para Scarpi (2015), o sistema é projetado de modo que os defeitos ou o comportamento inesperado sejam detectados durante a execução e é gerenciado de forma que não ocorra a falha de sistema. As abordagens simples de tolerância a defeitos, com base em verificação interna durante a execução, podem ser incluídas em todos os sistemas. Entretanto, as técnicas mais especializadas de tolerância a defeitos (como o uso de arquiteturas de sistema tolerantes a defeitos) são geralmente usadas quando é necessário alto nível de disponibilidade e confiabilidade de sistema.

Em Somerville (2011) um programa, a confiança pode ser alcançada evitando-se a introdução de defeitos, detectando e removendo defeitos antes da implantação do sistema e incluindo recursos de tolerância a defeitos, que permitem que o sistema se mantenha em funcionamento depois que um defeito tenha ocasionado uma falha de sistema.

2.1.4.3. Engenharia de proteção

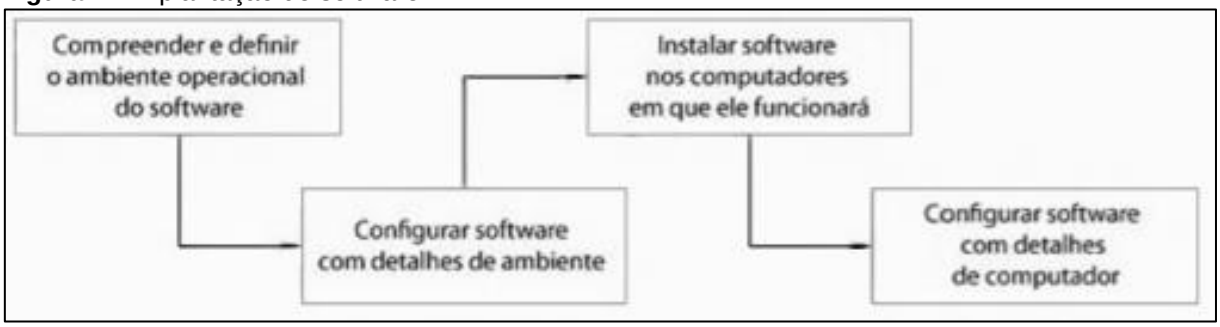
A engenharia de proteção de sistema é, de acordo com Pressman (2010), um aspecto cada vez mais importante do processo de engenharia de sistemas. A engenharia de proteção está preocupada com o desenvolvimento e a evolução de sistemas que possam resistir a ataques mal-intencionados para danificar o sistema ou seus dados.

No entanto, Sommerville (2011), informa que a engenharia de software de proteção é parte do campo mais geral da proteção de computadores. Isso se tornou prioridade para as empresas e indivíduos, uma vez que é crescente o número de criminosos que tentam explorar sistemas em rede para fins ilegais. Os engenheiros

de software devem estar cientes das ameaças à proteção enfrentadas pelos sistemas e das maneiras que essas ameaças podem ser neutralizadas.

A implantação de um sistema envolve configurar o software para operar em um ambiente operacional, instalar o sistema nos computadores desse ambiente e, em seguida, configurar o sistema instalado para esses computadores (Figura 2). Pfleeger e Pfleeger (2007) afirma que a configuração pode ser um processo simples que envolve o estabelecimento de alguns parâmetros internos do software para que reflitam as preferências do usuário. No entanto, às vezes, ela é complexa e exige a definição específica de modelos e regras de negócios que afetam a execução do software.

Figura 2 - Implantação de software



Fonte: Sommerville (2011).

Independentemente de quanto esforço seja dispendido na manutenção de sistemas de proteção, Pressman (2010) diz que deve sempre projetar seu sistema com base no pressuposto de que podem ocorrer falhas de proteção. Portanto, deve-se pensar em como recuperar de possíveis falhas e restaurar o sistema para um estado operacional protegido. Por exemplo, pode-se incluir um sistema de backup de autenticação para o caso de a autenticação de senhas ser comprometida.

2.1.4.4. Engenharia baseada em componentes

Atualmente muitos novos sistemas de negócios são desenvolvidos pela configuração de sistemas disponíveis no mercado. No entanto, Pressman (2011) afirma que quando uma empresa não pode usar um sistema de prateleira, porque eles não atendem a seus requisitos, o software de que necessitam precisa ser especialmente desenvolvido. Brown (2000) diz que para o desenvolvimento de softwares customizados, a engenharia de software baseada em componentes é uma forma eficaz, orientada ao reuso.

De acordo com Sommeville (2011), engenharia de software baseada em componentes (CBSE, do inglês *component-based software engineering*) surgiu na década de 1990 como uma abordagem para softwares de desenvolvimento de sistemas com base no reuso de componentes de softwares.

Inúmeras são as definições de componentes reusáveis são encontradas na literatura. Em Sametinger (1997) é apresentado que os componentes podem ser vistos como alguma parte do sistema de software que é identificável e reusável, ou como o estado seguinte de abstração depois de funções, módulos e classes.

Pressman (2011) e Sommeville (2011) concordam entre si, que os componentes são abstrações de nível mais alto do que objetos e são definidos por suas interfaces. Geralmente, eles são maiores que objetos individuais e todos os detalhes de implementação são escondidos de outros componentes. CBSE é o processo de definir, implementar, integrar ou compor componentes independentes, pouco acoplados em sistemas.

Na comunidade CBSE, não existe consenso sobre um componente ser uma unidade independente de software que pode ser composta com outros componentes para criar um sistema de software. Além disso, os autores têm definições diferentes a respeito de um componente de software. Councill e Heineman (2001) definem um componente como “Um elemento de software que está de acordo com um modelo de componentes padrão e pode ser independentemente implantado e composto, sem modificações, de acordo com um padrão de composição”.

Essa definição é essencialmente baseada em padrões, assim, uma unidade de software que esteja em conformidade com esses padrões são um componente. No entanto, Szyperski (2002) não menciona padrões em sua definição de um componente, mas, em vez disso, concentra-se nas principais características dos componentes:

> Um componente de software é uma unidade de composição com interfaces contratualmente especificadas e apenas dependências de contexto explícitas. Um componente de software pode ser implantado de forma independente e está sujeito a ser composto por parte de terceiros (SZYPERSKI, 2002, p. 43).

O que as definições anteriormente mencionadas têm em comum é o fato de concordarem que os componentes são independentes e que, em um sistema, eles são a unidade fundamental de composição. Na opinião de Sommeville (2011), uma melhor definição de um componente pode ser obtida combinando-se essas propostas.

A Tabela 3 mostra o que Sommeville (2011), considera como características essenciais de um componente como na CBSE (*Component Based Software Engineering*).

Tabela 3 - Característica de componentes

Característica do componente	Descrição
Padronizado	A padronização de componentes significa que um componente usado em um processo CBSE precisa obedecer a um modelo de componentes padrão. Esse modelo pode definir as interfaces de componentes, metadados de componente, documentação, composição e implantação.
Passível de composição	Um componente deve ser independente. Deve ser possível compor e implantá-lo sem precisar usar outros componentes específicos. Nessas situações em que os componentes precisam dos serviços prestados externamente, estes devem ser explicitamente definidos em uma especificação de interface 'requires'.
Implantável	Para ser implantável, um componente dever ser autocontido. Deve ser capaz de operar como uma entidade autônoma em uma plataforma de componentes que forneça uma implementação do modelo de componentes, o que geralmente significa que o componente é binário e não tem como ser compilado antes de ser implantado. Se um componente é implantado como um serviço, ele não precisa ser implantado por um usuário de um componente. Pelo contrário, é implantado pelo prestador do serviço.
Documentado	Os componentes devem ser completamente documentados para que os potenciais usuários possam decidir se satisfazem a suas necessidades. A sintaxe e, idealmente, a semântica de todas as interfaces de componentes devem ser especificadas.

FONTE: Sommerville (2011).

Atualmente, as empresas operam em um ambiente global, com mudanças rápidas. Assim, precisam responder a novas oportunidades e novos mercados, a mudanças nas condições econômicas e ao surgimento de produtos e serviços concorrentes.

Ainda em Sommeville (2011) encontra se os seguintes esclarecimentos, Softwares fazem parte de várias as operações de negócios, assim, novos softwares são confeccionados em passo acelerado para obterem proveito de novas oportunidades e responder às pressões concorrentes. O desenvolvimento e entrega rápidos são, portanto, o requisito mais crucial para o desenvolvimento de sistemas de software.

Dessa forma, o desenvolvimento Baseado em Componentes surge como uma nova perspectiva de desenvolvimento de software caracterizada pela composição de partes já existentes. Conforme Szyperski (2002), construir novas

soluções pela combinação de componentes desenvolvidos aumenta a qualidade e dá suporte ao rápido desenvolvimento, levando à diminuição do tempo de entrega do produto ao mercado.

> Ainda em Szyperski (1999) vamos encontrar o seguinte esclarecimento, sistemas definidos através da composição de componentes permitem que sejam adicionadas, removidas e substituídas partes do sistema sem a necessidade de sua completa substituição. Com isso, CBSE auxilia na manutenção dos sistemas de software, por permitir que os sistemas sejam atualizados através da integração de novos componentes e/ou atualização dos componentes já existentes (SZYPERSKI 1999, p. 56).

Entretanto, para que essa integração aconteça, surge à necessidade de uma estrutura que venha integrar/unir esses componentes, ou seja, uma arquitetura padrão que dão suporte ao reuso dos componentes. Para o desenvolvimento do sistema SGE, utilizou-se para esse fim o padrão MVC (modelo-visão-controlador, do inglês *Model-View-Controller*), (PRESSMAN, 2011). Esse padrão é a base para gerenciamento da interação em muitos sistemas baseados em Web.

2.1.4.5. Engenharia orientada a aspecto

O principal benefício de uma abordagem orientada a aspectos é que ela suporta a separação de interesses, Segundo Pinto (2013) separar interesses em elementos diferentes em vez de incluir interesses diferentes na mesma abstração lógica é uma boa prática de engenharia de software. Ao apresentar interesses transversais como aspectos, esses interesses podem ser entendidos, reusados e modificados de forma independente, sem a preocupação de onde o código é usado. Por exemplo, a autenticação de usuário pode ser representada como um aspecto que requisita um nome de usuário e uma senha. De acordo com Di Francescomarino e Tonella (2009), isso pode ser automaticamente embutido no programa sempre que uma autenticação for requerida.

Pesquisa e desenvolvimento sobre orientação a aspectos têm como foco principal a programação orientada a aspectos. As linguagens de programação orientadas a aspectos como AspectJ COLYER e CLEMENT, 2005; COLYER et al., 2005; KICZALES et al., 2001; LADDAD 2003a; LADDAD, 2003b foram desenvolvidas de forma a ampliar a programação orientada a objetos para incluir aspectos.

As principais empresas usaram programação orientada a aspectos em seus processos de produção de software COLYER e CLEMENT, 2005. No entanto, os interesses transversais são igualmente problemáticos em outros estágios do processo de desenvolvimento de software.

Os pesquisadores estão investigando, atualmente, como utilizar orientação a aspectos na engenharia de requisitos de sistema e na modelagem de sistema e como testar e verificar programas orientados a aspectos. Na tabela 4 a seguir descreve-se as terminologias usadas na engenharia de software orientada a aspectos.

Tabela 4 - Terminologia usada na engenharia de software orientada a aspectos

Termo	Definições
Adendo	Código que implementa um interesse.
Aspecto	Uma abstração de programa que define o interesse transversal. Inclui a definição de um ponto de corte e do adendo associado com esse interesse.
Ponto de junção	Evento em um programa em execução onde o adendo associado com um aspecto pode ser executado.
Modelo de ponto de junção	Conjunto de eventos que podem ser referenciados em um ponto de corte.
Ponto corte	Uma declaração, inclusa em um aspecto, que define os pontos de junção onde o adendo de aspecto associado deve ser executado.
composição	A incorporação do código de adendo em ponto de junção específico por um compositor de aspectos.

FONTE: Sommerville (2011).

Aspectos foram introduzidos originalmente como uma construção de linguagem de programação, mas a noção de interesses vem de requisitos de sistema. Portanto, de acordo Resende (2006) faz sentido adotar uma abordagem orientada a aspectos em todos os estágios do processo de desenvolvimento de software. Sommerville (2011) diz que nos estágios iniciais da engenharia de software, adotar uma abordagem orientada a aspectos significa usar a ideia de separação de interesses como base para pensar nos requisitos e projeto de sistema. Identificar e modelar interesses deve fazer parte dos processos de engenharia de software e de modelagem. Linguagens de programação orientadas a aspectos fornecem, então, o suporte tecnológico para manter a separação de interesses em sua implementação do sistema.

2.2. Ementa da disciplina de Engenharia de Software do IFTO *campus* Porto nacional – TO.

A ementa é como um resumo ou sinopse de determinado tema ou área. Quando se fala em “ementa do curso”, significa a apresentação de um texto que evidencia as principais características deste curso (PORTAL SIGNIFICADOS, 2016).

Da mesma forma, a “ementa disciplinar” ou "ementa escolar" aponta os postos-chaves da matéria a ser apresentada ou das disciplinas que serão ministradas durante o ano letivo, continua afirmando o Portal Significados.

A ementa da disciplina de Engenharia de Software se encontra disponível no Anexo I do Projeto Pedagógico do Curso de graduação de Licenciatura em Computação ofertado pelo *Campus* Porto Nacional, do IFTO. Conforme apresentado na tabela 5 a seguir.

Tabela 5 - Componente Curricular: Engenharia de Software

Componente Curricular: Engenharia de Software
Carga Horária (CH) total: 66,7 CH teórica: 33,3 CH prática: 33,4
Pré-requisitos: Modelagem e Análise de Sistemas.
Ementa
Visão geral e princípios fundamentais da Engenharia de Software. Conhecimentos básicos sobre ciclo de vida de software e seus estágios: requisitos, projeto, implementação, gerenciamento, qualidade. Emprego de metodologias e ferramentas para análise e projeto de sistemas. Documentação de software. Engenharia Reversa. Reengenharia. Ferramentas CASE: conceitos, tipos e exemplos associando com as etapas do ciclo de vida de software.
Competências
• Apresentar, analisar e discutir o corpo de conhecimento que constitui a Engenharia de Software, seus princípios, métodos e ferramentas. • Identificar as técnicas da Engenharia de Software para o desenvolvimento de um produto.
Habilidades
• Ser capaz de descrever os métodos, ferramentas e procedimentos associados; • Conhecer e utilizar as técnicas de trabalho em grupo, especificamente para desenvolvimento de softwares; • Saber identificar os princípios da ética profissional do engenheiro de software.
Bibliografia básica
PRESSMAN, R. S. **Engenharia de Software**. 5ª ed. São Paulo: McGraw-Hill, 2007. PAULA FILHO, W. P. **Engenharia de software: fundamentos, métodos e padrões**. Rio de Janeiro: LTC, 2003. LTC, 2009.
Bibliografia complementar

PRESSMAN, R. **Software Engineering: A Practitioner's Approach.** EUA: G- Hill, 2005.

SOMMERVILLE, I. **Engenharia de Software**. São Paulo: Pearson. 2007.

DAVIS, A. M. **Software requirements**. EUA: Prentice Hall, 1993.

PFLEEGER, S. L. **Engenharia de Software Teoria e Prática.** São Paulo: Prentice Hall, 2004.

MAGELA, R. **Engenharia de softwares aplicada**. Rio de Janeiro: Alta Books, 2006.

Fonte: IFTO (2018).

2.2.1. Plano de ensino da disciplina de Engenharia de Software

O Plano de Ensino é um plano de ação; é o registro do planejamento das ações pedagógicas para o componente curricular durante o período letivo. É um instrumento didático-pedagógico e administrativo de elaboração e uso obrigatórios. Pode-se pensar, primeiramente, que:

> […] planejamento é processo de busca de equilíbrio entre meios e fins, entre recursos e objetivos, visando ao melhor funcionamento de empresas, instituições, setores de trabalho, organizações grupais e outras atividades humanas. O ato de planejar é sempre processo de reflexão, de tomada de decisão sobre a ação; processo de previsão de necessidades (…) visando à concretização de objetivos, em prazos determinados e etapas definidas, a partir dos resultados das avaliações (PADILHA, 2001).

Sendo plano, trata-se de um “documento utilizado para o registro de decisões do tipo: o que se pensa fazer, como fazer, quando fazer, com que fazer, com quem fazer. Para existir plano é necessária a discussão sobre fins e objetivos, culminando com a definição dos mesmos” (PADILHA, 2001).

Dentro dessas definições, o Plano de ensino da disciplina de Engenharia de Software descreve no seu tópico 4 – Bases Tecnológicas, os modelos e processos de software utilizados/ministrados nas aulas da disciplina de Engenharia de Software IFTO (2018), conforme descrito a seguir: Cascata, Prototipação, Evolutivo, Incremental, Transformação e Espiral. Por sua vez, o referido Plano de Ensino da disciplina de Engenharia de Software, segue como anexo 2 deste trabalho.

2.3. Trabalhos correlacionados

2.3.1. Uma Investigação ao inicial da Atividade de Engenharia de Requisitos em Processos Ágeis

Para enfrentar problemas com prazos e complexidade de processos pesados da Engenharia de Software, diversas metodologias de desenvolvimento ´ágil estão sendo utilizadas em projetos de software. Estas metodologias possuem como principal objetivo a satisfação do cliente, se preocupando com a entrega incremental

de software desde as etapas iniciais de desenvolvimento, produtos de trabalho de engenharia de software minimizados e simplicidade global no desenvolvimento. Este artigo procura discutir como as disciplinas tradicionais do processo da Engenharia de Requisitos são consideradas em Metodologias Ágeis, apontando vantagens e deficiências destas novas metodologias, algumas áreas que podem ser exploradas em pesquisas futuras.

2.3.2. Ferramenta de apoio a realização de experimentos em engenharia de software

Este trabalho tem como objetivo apresentar a especificação e implementação de uma ferramenta de apoio a experimentos em Engenharia de Software, que permite a definição dos objetivos, a definição das questões e métricas e a verificação das hipóteses. A ferramenta é direcionada para experimentos relacionados à melhoria de processo.

2.4. Comentários Parciais

Este capítulo discutiu informações sobre elementos que proporcionaram o suporte conceitual necessário para a elaboração do trabalho, os elementos abordados foram: conceitos sobre a engenharia de software; as engenharias que possuem Tendências emergentes de sucesso para desenvolvimento de software; ementas das disciplinas de engenharia de software e plano de ensino. O estudo realizado nesse capítulo deu embasamento teórico para o desenvolvimento do trabalho apresentado metodologicamente no capítulo seguinte. No final do estudo foi realizada uma busca por trabalhos correlacionados.

3. PROCEDIMENTOS METODOLÓGICOS

Este capítulo apresenta a metodologia científica adotada neste trabalho, de forma a sistematizar as tendências emergentes de sucesso para a engenharia de software: Uma revisão sistemática, em seguida consta como foi realizada a pesquisa, confecção e aplicação do questionário.

3.1. Materiais e Métodos

Para a consecução do presente trabalho foi adotado uma classificação de pesquisa de campo, conforme Fonseca (2002), caracteriza-se pelas investigações em que, além da pesquisa bibliográfica e/ou documental, se realiza coleta de dados junto a pessoas, com o recurso de diferentes tipos de pesquisa (pesquisa *ex-postfacto*, pesquisa-ação, pesquisa participante, etc.) Este estudo apresenta o enfoque do método de abordagem qualitativa, que segundo Marconi e Lakatos (1993, p. 8). “Método é o conjunto das atividades sistemáticas e racionais que com maior segurança e economia, permite alcançar o objetivo, conhecimento válido e verdadeiro, traçando o caminho a ser seguido, detectando erros e auxiliando as decisões do cientista” (MARCONI E LAKATOS, 1993, P8).

Embora a pesquisa utilize o método qualitativo para analisar os dados obtidos, vale ressaltar que “mesmo pesquisas com dados qualitativos podem ter um tratamento matemático, especialmente usando-se de análises estatísticas” (ABRANTES et al, 2007, p.13). Desta forma, fez se o uso dos métodos estatísticos, tendo em vista a necessidade de tornar quantificável algumas informações do questionário.

3.1.1. Levantamento das tendências emergentes de sucesso para a engenharia de software

Para tanto, fez se um levantamento teórico, realizado por meio de uma pesquisa bibliográfica, uma vez que o pesquisador busca embasamento na literatura existente. A pesquisa bibliográfica foi realizada a partir do levantamento de referências teóricas já analisadas, e publicadas por meios escritos e eletrônicos, como livros, páginas de web sites e artigos científicos extraídos das bases de dados SCIELO, CAPES, Google Acadêmico, Science Research, BDTD, Eric, Elsevier.

Na opinião de Fonseca, (2002, p. 32). “Qualquer trabalho científico inicia-se com uma pesquisa bibliográfica, que permite ao pesquisador conhecer o que já se estudou sobre o assunto”. Existem, porém, pesquisas científicas que se baseiam

unicamente na pesquisa bibliográfica, procurando referências teóricas publicadas com o objetivo de recolher informações ou conhecimentos prévios sobre o problema a respeito do qual se procura a resposta.

Silvera (2009), afirma que os exemplos mais característicos desse tipo de pesquisa são sobre investigações sobre ideologias ou aquelas que se propõem à análise das diversas posições acerca de um problema.

Em Salomon (2004), lê-se que a pesquisa bibliográfica se fundamenta em conhecimentos proporcionados pela Biblioteconomia e Documentação, entre outras ciências e técnicas empregadas de forma metódica envolvendo a identificação, localização e obtenção da informação, fichamento e redação do trabalho científico. Esse processo solicita uma busca planejada de informações bibliográficas para elaborar e documentar um trabalho de pesquisa científica.

3.1.1.1. Revisão sistemática

Uma revisão sistemática, assim como outros tipos de estudo de revisão, conforme Sampaio (2007), é uma forma de pesquisa que utiliza como fonte de dados a literatura sobre determinado tema.

Esse tipo de investigação na ótica de Basto e Barretos (2019), disponibiliza um resumo das evidências relacionadas a uma estratégia de intervenção específica, mediante a aplicação de métodos explícitos e sistematizados de busca, apreciação crítica e síntese da informação selecionada.

Assim as revisões sistemáticas segundo Bibiano et al (2019), são particularmente úteis para integrar as informações de um conjunto de estudos realizados separadamente sobre determinada terapêutica intervenção, que podem apresentar resultados conflitantes e/ou coincidentes, bem como identificar temas que necessitam de evidência, auxiliando na orientação para investigações futuras.

3.1.1.2. Etapas do levantamento das tendências

A pesquisa deu-se início a partir de um problema diagnosticado que consiste em saber como atualizar a ementa da disciplina de engenharia de software com as atuais tendências de sucesso da Engenharia de Software?

Uma vez elaborada a pergunta que nortearia a seleção dos artigos, faltava definir os termos de busca, os quais deveriam ser precisos, capazes de encontrar um maior número de estudos cujos conteúdos apresentassem informações relevantes sobre as atuais tendências de sucesso da Engenharia de Software. Diante disso,

estabeleceu-se os seguintes termos de busca que foram traduzidos para o inglês em virtude dos artigos disponibilizados nas mencionadas bases estarem escritos no citado idioma:

Tabela 6 Termos de busca usados nas bases de dados

Termos de buscas
TB1 – “engenharia + software “
TB2 – “modelos + desenvolvimento + software “
TB3 – “tendências de sucesso + engenharia de software “
TB4 – “ementas + disciplinas + engenharia de software “

Fonte: O autor (2019)

Depois de elaborado os termos de busca fez-se uma pesquisa nas bases de dados afim de extrair as atuais tendências quanto ao uso das engenharias de desenvolvimento de software. Além disso, definiu-se um critério de importância onde só seriam analisados os 50 (cinquentas) primeiros estudos listados por cada busca, uma vez que foi possível detectar, já na primeira leva de registro de artigos listados, que depois dessa quantidade de artigos listados por cada termo de busca em cada pesquisa, os artigos não mais tratavam sob nenhum aspecto do objeto de estudo pesquisado.

A etapa seguinte dessa revisão foi a escolha criteriosa das bases de dados a serem consultadas no intuito de se obter os artigos especializados sobre o tema da pesquisa, definidas conforme tabela a baixo.

Tabela 7 - Base de dados pesquisadas

Bases de dados
BD1 – ACM Digital Library - http://portal.acm.org
BD2 – ScienceDirect – Elsevier - http://www.elsevier.com
BD3 – IEEE Xplore - http://www.ieee.org/web/publications/xplore/
BD4 – Emerald - http://www.emeraldinsight.com
BD5 – Google Scholar - http://scholar.google.com.br/
BD6 – Microsoft Academic - http://academic.research.microsoft.com/
BD7 – ISI Web of Science - http://www.isiknowledge.com
BD8 – Wiley InterScience - http://www.interscience.wiley.com

Fonte: O autor (2019).

Assim, apenas 83 artigos foram baixados, lidos e analisados para verificar se de fato os referidos artigos tratavam das atuais tendências quanto ao uso das engenharias de desenvolvimento de software.

Após essa leitura completa e criteriosa dos 83 artigos baixados, verificou-se a necessidade de se promover o descarte de mais 36 artigos que embora possuíssem no título ou no resumo alguma palavra que remetia a das atuais tendências quanto ao uso das engenharias de desenvolvimento de software, não tratavam especificamente do tema ora investigado. Depois desse novo descarte, restaram somente 47 artigos classificados selecionados conforme tabela a seguir.

Tabela 8 - Critérios de seleção de artigos

CS1 - Publicado entre o ano de 2009 e 2019
CS2 - O título conter alguma das palavras chaves
CS3 - Abordar algum tipo de tendências de sucesso na área da engenharia de software
CS4 - Artigos disponibilizados nos seguintes bancos de dados: SCIELO, CAPES, Google Acadêmico, Science Research, BDTD, Eric, Elsevier.
CS5 – Estar entre os 50 primeiros listados na busca
CS6 - Estar escrito em inglês ou português

Fonte: O autor (2019).

Após a tradução dos artigos, foram extraídas as atuais tendências de uso das engenharias de desenvolvimento de software, conforme descritos nas no capitulo 2, subitem 2.1.4 deste trabalho.

3.1.2. Pesquisa de campo

Para a consecução do presente trabalho foi adotado uma classificação de pesquisa de campo, conforme Fonseca (2002), pesquisa de campo caracteriza-se pelas investigações em que, além da pesquisa bibliográfica e/ou documental, se realiza coleta de dados junto a pessoas, com o recurso de diferentes tipos de pesquisa (pesquisa *ex-post-facto*, pesquisa-ação, pesquisa participante, etc.)

Desta forma, fez se o uso dos métodos estatísticos, tendo em vista a necessidade de tornar quantificável algumas informações do questionário. O instrumento de coleta de dados escolhido foi a aplicação de um questionário confeccionado para ser aplicados via web aos entrevistados, como descrito no subitem 3.1.3 a seguir.

3.1.3. Confecção do Questionário da Pesquisa.

O questionário é um instrumento ou programa de coleta de dados que segundo Carnevalli e Miguel, (2001), pode ser enviado via correio, fax, Internet, etc., o questionário pode ser estruturado não disfarçado: o respondente sabe qual é o

objetivo da pesquisa, e o questionário é padronizado, usando principalmente questões fechadas.

A confecção é feita pelo pesquisador; o preenchimento é realizado pelo informante. A linguagem utilizada no questionário deve ser simples e direta para que o interrogado compreenda com clareza o que está sendo perguntado (KAUARK, MANHÃES E MEDEIROS, 2010).

O questionário foi confeccionado utilizando a plataforma do google, formulário *Google Form* e utilizado como instrumento de coleta de dados, os links foram enviados via e-mail, seguido de uma nota explicativa, onde, informava a natureza da pesquisa e de cada instrumento aplicado. Pois como afirma Marconi e Lakatos (2003, p. 201):

> Junto com o questionário deve-se enviar uma nota ou carta explicando a natureza da pesquisa, sua importância e a necessidade de obter respostas, tentando despertar o interesse do recebedor, no sentido de que ele preencha e devolva o questionário dentro de um prazo razoável (LAKATOS 2003, p. 201).

Contudo, explicar os objetivos da pesquisa na nota, solicitando colaboração aumenta a influência no retorno dos questionários, pois conforme as autoras anteriormente citadas, deve se despertar o interesse do pesquisado para que ele preencha e devolva-o do mesmo modo que foi enviado.

Nesse sentido, o questionário foi definido em reunião junto ao professor orientador que norteou a confecção do mesmo, assim todas as edições do questionário foi acompanhada pelo orientador.

A edição do questionário foi realizada em alguns passos: o primeiro foi pesquisa sobre como confecciona-los, o segundo passo foi a criação dos escopos para tomada de base na edição, o terceiro passo foi as correções dos escopos, o quarto passo foi a confecção do questionário no Google Docs, ferramenta gratuita para criação de formulários, sendo fundamental para a realização da pesquisa por meio da internet.

A confecção do questionário da pesquisa se deu com base na necessidade de verificar junto aos docentes e discentes do curso de Licenciatura em Computação do Campus Porto Nacional do IFTO, quais tendências seriam mais importantes para o estudo junto a disciplina de engenharia de software. assim a sua elaboração se deu

com base no método avaliador baseado em Carnevalli e Miguel (2001), pode ser questionário sendo este estruturado não camuflado: o respondente sabe qual é o objetivo da pesquisa, tal método garante uma maior precisão na análise e tabulação dos dados obtidos. O questionário de pesquisa construído nesta etapa está disposto no apêndice A desta monografia.

3.1.4. Submissão do Questionário da Pesquisa

Após a confecção foi realizada a submissão do questionário de pesquisa, com o objetivo de coletar os dados para verificar junto aos docentes e discentes do curso de Licenciatura em Computação do Campus Porto Nacional do IFTO, quais tendências seriam mais importantes para o estudo junto a disciplina de engenharia de software, o questionário de pesquisa foi submetido por meio de um formulário Google Docs que permite aos participantes responderem via internet, outra vantagem deste recurso é enviar o link do formulário via e-mail, possibilitando que o mesmo seja respondido com tempo pelo participante e de qualquer lugar que disponha de internet.

3.1.5. Retorno e tabulação das respostas do formulário de pesquisa

Como explicado no item anterior 3.1.4, foi enviado um link do formulário de pesquisa via e-mail, com uma nota explicativa, isso possibilitou o retorno das respostas em tempo que o questionário era respondido pelo universo da pesquisa.

Com os dados brutos em mãos, o processamento dessas informações se deu das seguintes formas: Em primeiro momento as coletas dos dados foram feitas em planilhas eletrônicas do pacote *Google Docs,* no segundo momento os dados foram salvos em planilhas do Microsoft Excel para facilitar a análise e tabulação dos dados, em terceiro momento foi aplicado os métodos estatísticos para a tabulação dos dados. E por fim, os resultados foram impressos em gráficos e tabelas geradas pela ferramenta utilizada, para serem apresentados no capítulo 4 Análises e Resultados dos Dados.

3.2. Comentários Parciais

Neste capitulo foram apresentados os procedimentos metodológicos traçados desde a criação dos formulários até sua aplicação e os métodos utilizados para coleta e análise dos dados utilizados nesta pesquisa. O mesmo foi dividido em duas etapas, a primeira sendo a definição dos procedimentos metodológicos da pesquisa e a segunda sendo a confecção do método de coleta de dados.

Desta forma, foram apresentados na primeira etapa os embasamentos teóricos para realização desta pesquisa, foi realizada a definição do universo da pesquisa, o método a ser utilizado na pesquisa e a escolha da forma da aplicação do questionário de pesquisa.

Na segunda etapa, foi selecionado o instrumento de coleta de dados, seguindo com a sua confecção que no primeiro momento foi elaborado um escopo do questionário da pesquisa, no segundo momento foi confeccionado o questionário da pesquisa, que só então foi aplicado para a coleta dos dados.

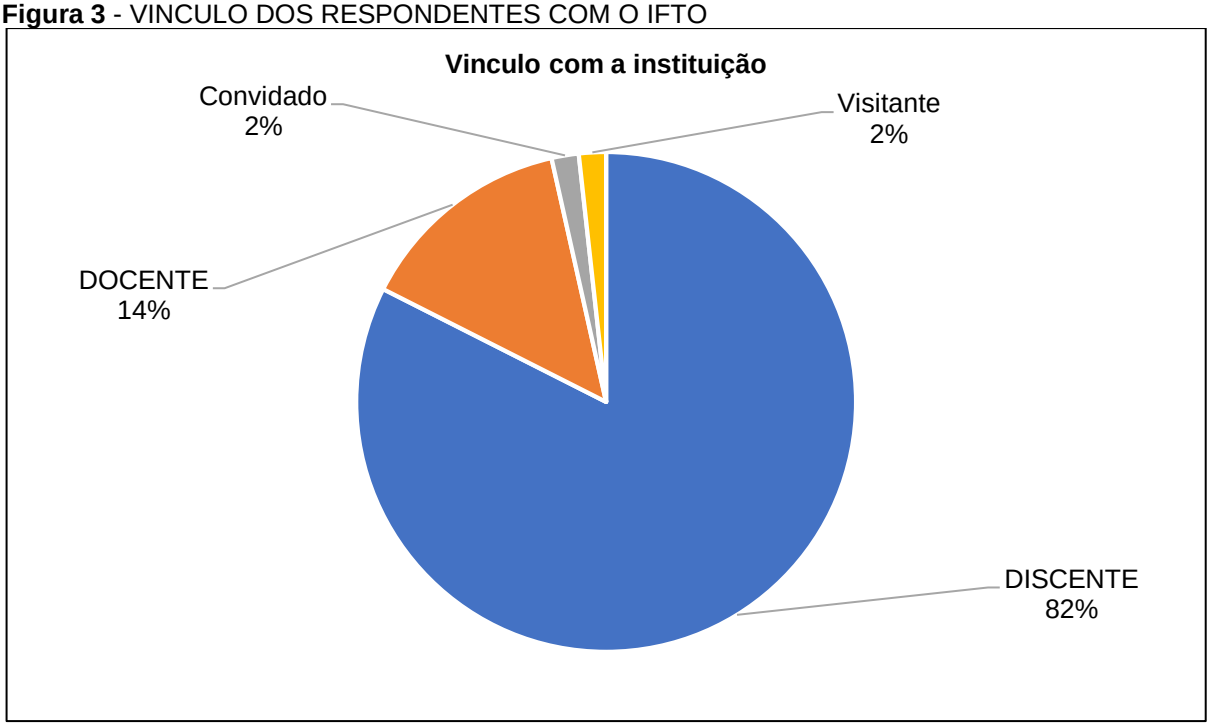
Todas as atividades realizadas neste capitulo foram importantes para alcançar os objetivos desta pesquisa que é Identificar, verificar e propor o estudo das melhores tendências de sucesso como parte da ementa da disciplina de engenharia de software do curso de Licenciatura em Computação do Campus Porto Nacional do IFTO.

4. ANÁLISE E RESULTADOS DOS DADOS

Neste Capitulo são apresentados os resultados e análises obtidas com a pesquisa, Tendências emergentes de sucesso para a engenharia de software: Uma revisão sistemática, foram levantados os seguintes questionamentos: 1). Qual o vínculo do respondente com a instituição IFTO? 2). Qual o grau de conhecimento sobre engenharia de desenvolvimento de software? 3). Quais dessas tendências dentro engenharia de software você julga mais importante para o estudo junto a disciplina de engenharia de software? 4). Caso você conheça outra tendência que não foi citada nesta pesquisa, favor informar. Além de informações pessoais e área de atuação dos respondentes.

Do universo da pesquisa de 96 pessoas, 57 responderam ao questionário, conforme discriminados e apresentados no gráfico da figura – 3, a seguir: 82% discentes, 14% docentes, 2% convidados e 2% visitantes.

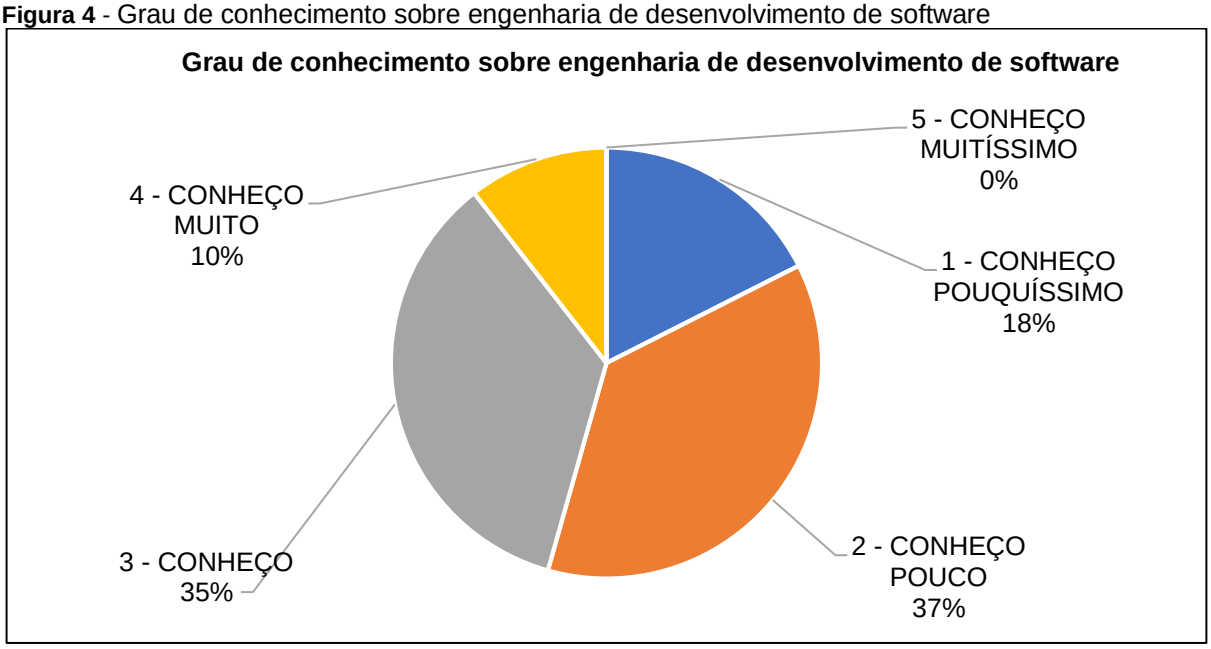
Figura 3 - VINCULO DOS RESPONDENTES COM O IFTO



Fonte: o Autor (2019).

Quando questionados qual o grau de conhecimento sobre engenharia de desenvolvimento de software? 37% responderam que conhece pouco, 35% apenas conhece, 18% conhece pouquíssimo, 10% conhece muito e conhece muitíssimo não pontuou. Esses dados estão apresentados no gráfico da figura – 4.

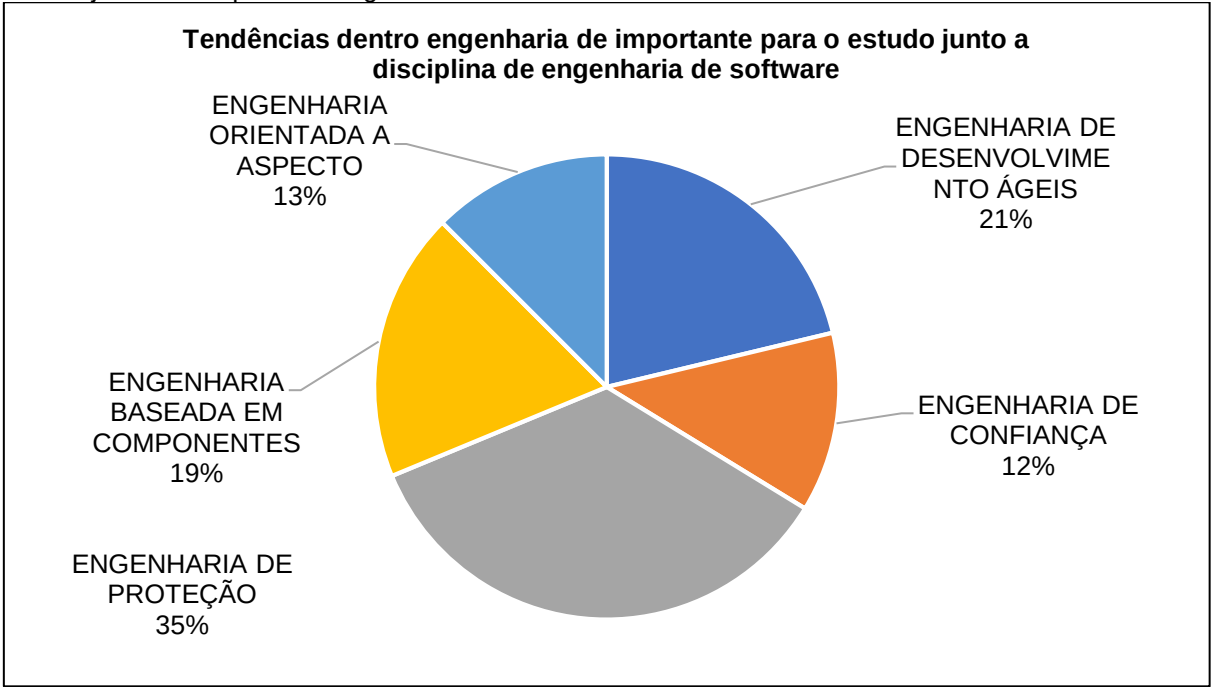
Figura 4 - Grau de conhecimento sobre engenharia de desenvolvimento de software



Fonte: o Autor (2019).

A Pergunta 3 do questionário de pesquisa foi: Quais dessas tendências dentro engenharia de software você julga mais importante para o estudo junto a disciplina de engenharia de software? Assim 35% disseram que a engenharia mais importante é a engenharia de proteção, 21% disseram que é a engenharia de desenvolvimento ágeis, 19% disseram que é a engenharia baseada em componentes, 13% que é a engenharia baseada em aspecto, enquanto que 12% disseram que é a engenharia de confiança. Dados esses apresentados no gráfico da figura 5 a baixo.

Figura 5 - Quais dessas tendências dentro engenharia de software você julga mais importante para o estudo junto a disciplina de engenharia de software



Fonte: o Autor (2019).

4.1. Comentários Parciais

Neste capítulo está sendo apresentados os resultados obtidos da pesquisa que visou Identificar, verificar e propor o estudo das melhores tendências de sucesso como parte da ementa da disciplina de engenharia de software do curso de Licenciatura em Computação do Campus Porto Nacional do IFTO.

Os dados obtidos com a pesquisa permitiram avaliar que a maioria dos envolvidos conhecem pouco os tipos engenharias que tem a tendência de sucesso para o desenvolvimento de software, e que preferem a engenharia de proteção como integrante da ementa da disciplina de engenharia de software do curso de licenciatura em computação do IFTO.

5. CONCLUSÕES E CONSIDERAÇÕES

Diante dos resultados obtidos com a realização desse estudo, nota-se que que a problemática foi resolvida, pois foi identificado na literatura as atuais tendências do uso de engenharias de desenvolvimento de software. A maioria dos respondentes preferiram atualizar a ementa da disciplina de engenharia de software com a engenharia de proteção, respondendo, portanto, a problemática desta monografia: Como atualizar a ementa da disciplina de engenharia de software com as atuais tendências de sucesso?

Podendo também observar que os objetivos foram alcançados, pois a pesquisa permitiu identificar as tendências de sucesso dentro da engenharia de software, verificou-se junto aos docentes e discentes do curso de Licenciatura em Computação do Campus Porto Nacional do IFTO, quais tendências seriam mais importantes para o estudo junto a disciplina de engenharia de software e este estudo propõe as melhores tendências de sucesso como parte da ementa da disciplina de engenharia de software do curso de Licenciatura em Computação do Campus Porto Nacional do IFTO. As quais são: Engenharia de confiança, engenharia de proteção, engenharia de desenvolvimento ágeis, engenharia orientada a aspecto e engenharia baseada em componentes.

A hipótese que solucionaria a problemática deste trabalho foi validada, pois o estudo permitiu a realização de um estudo bibliográfico e de campo, este último dentro do IFTO *campus* Porto Nacional, a fim de identificar as atuais tendências de uso das Engenharias de software e com isso realizar a atualização do plano de ensino dentro da ementa da disciplina.

Todavia a temática desse estudo se torna relevante pois, apresentou uma abordagem multidisciplinar ao analisar as tendências emergentes de sucesso para a engenharia de software: Uma revisão sistemática. Uma vez que, trata-se de um tema pouco explorado pelo universo acadêmico e apresentou uma proposta de melhoria junto a disciplina de Engenharia de Software do *Campus* porto Nacional do IFTO. Com isso, o egresso sairá melhor atualizado com as tendências de engenharias de desenvolvimento de software mais utilizadas no mercado de trabalho.

Assim, o presente trabalho ganha ênfase, pois dentro da literatura pesquisada não há outro igual ou que se assemelhe a esse, quanto a sua proposta, tema, problema e objetivos.

Por fim, este trabalho como um todo propiciou grande experiência ao autor, a principal razão é o envolvimento e foco em todas as etapas de desenvolvimento e escrita do projeto. Com o foco no desenvolvimento, foi possível colocar em prática todos os conceitos levantados durante a pesquisa bibliográfica bem como as habilidades adquiridas nas disciplinas no decorrer do curso.

5.1. Trabalhos Futuros

Como trabalhos futuros o autor pretende desenvolver um artigo sobre o uso do a engenharia de proteção, engenharia essa, apresentada pela literatura e pelos respondentes desta pesquisa como sendo a tendência da atualidade para desenvolvimento de software.

Pretende se também acompanhar essa tendência e observar até quando é viável utilizá-la em cada modelo de desenvolvimento de software e em cada projetos.

A autor pretende ainda aprofundar seus estudos na área de engenharia de software, realizando uma pós-graduação/mestrado nesse segmento. pois pretende se desenvolver softwares e apps, gerenciar projetos ligados aos softwares, arquitetar o design estrutural dos programas, realizar testes nos sistemas e gerenciar bancos de dados.

REFERENCIAS

ABRAHAMSSON, Pekka et al. **Novas orientações sobre métodos ágeis: uma análise comparativa.** In: 25ª Conferência Internacional de Engenharia de Software, 2003. Anais. Ieee, 2003. p. 244-254.

Abrantes PM, Magalhães SMS, Acúrcio FA, Sakurai E. **Avaliação da qualidade das prescrições de antimicrobianos dispensadas em unidades públicas de saúde de Belo Horizonte, Minas Gerais**, Brasil. Cad. Saúde Pública. 2007.

BARROS, Gabryela Santana et al. **Análise experimental entre as técnicas TDD e Test-Last no processo de manutenção corretiva de software**. 2019.

BARTIÉ, Alexandre. **Garantia da qualidade de software**. Gulf Professional Publishing, 2002.

BASTOS, Liliana Paiva; BARRETO, Maria de Lourdes Mattos. UM ESTUDO DE REVISÃO SISTEMÁTICA DA LITERATURA NACIONAL ENTRE 2007 E 2016 SOBRE OS ADOLESCENTES EM CONFLITO COM A LEI. Interfaces Científicas-Humanas e Sociais, v. 8, n. 1, p. 39-56, 2019.

BECK, Kent et al**. Manifesto for agile software development**. Feb. 2001. Disponível em <http:// www.agilemanifesto.org/>. Acesso em janeiro, 2019.

BIBIANO, Alana Maiara Brito et al. Fatores associados à utilização dos serviços de saúde por homens idosos: Uma revisão sistemática da literatura. Ciência & Saúde Coletiva, v. 24, p. 2263-2278, 2019.

BROWN, H. Douglas et al. **Principles of language learning and teaching**. 2000.

CARNEVALLI, José Antonio; MIGUEL, Paulo Augusto Cauchick. **Desenvolvimento da pesquisa de campo, amostra e questionário para realização de um estudo tipo survey sobre a aplicação do QFD no Brasil**. XXI Encontro Nacional de Engenharia de Produção-ENEGEP, 2001.

COHEN, David et al. **Agile software development: a DACS state of art report. NY: Data Analysis Center for Software - Fraunhoufer Center for Experimental Software Engineering Maryland and The University of Maryland**, 2003. Disponível em <http://www.thedacs.com/techs/agile/>. Acesso em abril, 2019.

COLLIER, Paul; HOEFFLER, **Anke. Aluguéis de recursos, governança e conflito. Jornal de resolução de conflitos**, v. 49, n. 4, p. 625-633, 2005.

COLYER, Adrian; CLEMENT, Andy. **Aspect-oriented programming with AspectJ**. IBM Systems Journal, v. 44, n. 2, p. 301-308, 2005.

COLYER, Adrian; CLEMENT, Andy. **Aspect-oriented programming with AspectJ**. IBM Systems Journal, v. 44, n. 2, p. 301-308, 2005.

DANDOLINI, JEISON. **Ferramenta de Apoio a Realização de Experimentos em engenharia de Software. Monografia, Universidade Regional de Blumenau, Ciências da Computação**, 2006.

DE PÁDUA PAULA FILHO, Wilson. **Engenharia de software**. LTC, 2003.

Degoulet, P., Fieschi, M. Introduction to Clinical Informatics. New York : Springer-Verlag, 1997.

DI FRANCESCOMARINO, Chiara; TONELLA, Paolo. **Programação orientada a aspectos cooperativos para processos de negócios executáveis**. In: Anais do Workshop ICSE de 2009 sobre princípios de sistemas orientados a serviços de engenharia. IEEE Computer Society, 2009. p. 91-94.

DOS SANTOS SILVA, Daisy Eliana; DE SOUZA, Ingredy Thaís; CAMARGO, Talita. **Metodologias Ágeis Para O Desenvolvimento De Software: Aplicação E O Uso Da Metodologia Scrum Em Contraste Ao Modelo Tradicional De Gerenciamento De Projetos**. Revista Computação Aplicada-UNG-Ser, v. 2, n. 1, p. 39-46, 2013.

FERREIRA, Aurélio Buarque de Holanda. **Novo Aurélio Século XXI: o dicionário da língua portuguesa**. 3 ed. totalmente rev. e ampl. Rio de Janeiro: Nova Fronteira, 1999.

FONSECA, João José Saraiva**. Metodologia da Pesquisa Científica**. 2002.

HEINEMAN, George T .; COUNCILL, William T. **Engenharia de software baseada em componentes**. Juntando as peças, addison-westley , p. 5, 2001.

HIGHSMITH, Jim. **Agile software development ecosystems**. Boston: Addison-Wesley, 2002.

HIGHSMTIH, Jim et al. **Extreme programming: e-business application delivery**. Feb. 2002. Disponível em <http://www.cutter.com/freestuff/ead0002.pdf>. Acesso em julho, 2019.

JALOTE, P. **An Integrated Approach to Software Engineering**. 3. ed. New York: Springer, 2005, 566p.

KAUARK, Fabiana da Silva; MANHÃES, Fernanda Castro; MEDEIROS, Carlos Henrique. **Metodologia da pesquisa: um guia prático**. 2010.

KICZALES, Gregor et al. **Uma visão geral do AspectJ.** In: Conferência Europeia sobre Programação Orientada a Objetos . Springer, Berlim, Heidelberg, 2001. p. 327-354.

KOSCIANSKI, André; DOS SANTOS SOARES, **Michel. Qualidade de Software-2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software.** Novatec Editora, 2007.

LADDAD, Ramnivas. **AspectJ in action: practical aspect-oriented programming. Dreamtech** Press, 2003.

LAKATOS, Eva Maria; MARCONI, M. de A. **Fundamentos da metodologia do trabalho científico**. São Paulo: Atlas, 1993.

MARCONI, Marina de Andrade. LAKATOS, Eva Maria. **Fundamentos de metodologia científica**, v. 5, 2003.

MIRANDA, Raaby et al. **Uma Análise do Impacto da Filosofia Ágil do Scrum no Sucesso de Projetos de Software**. In: Anais do XXVII Workshop sobre Educação em Computação. SBC, 2019. p. 389-403.

PADILHA, R. P. **Planejamento dialógico: como construir o projeto político pedagógico da escola**. São Paulo: Cortez; Instituto Paulo Freire, 2001.

PEZZÈ, Mauro; YOUNG, Michal. **Teste e análise de software: processos, princípios e técnicas**. Bookman Editora, 2009.

PFLEEGER, C P.; PFLEEGER, S. L. **Security in Computing**. 4. ed. Boston: Addison-Wesley, 2007.

PINTO, Mónica; HORCAS, José M. **Como desenvolver aplicativos seguros com Programação Orientada a Aspectos**. In: 2013 Conferência Internacional sobre Riscos e Segurança da Internet e Sistemas (CRiSIS). IEEE, 2013. p. 1-3.

PORTAL SIGNIFICADOS. **Significado de Ementa**, 2016. Disponível em:<https://www.significados.com.br/ementa/>. Acesso em 23 de jun de 2019.

PRESSMAN, R. S. **Engenharia de Software**. 6. ed. Rio de Janeiro: McGraw-Hill, 2006, 720p.

PRESSMAN, Roger S. **Engenharia de Software**, Sexta Edição. Editora MCGrawHill: Porto Alegre, 2010.

PRESSMAN, Roger. **Engenharia de Software**. São Paulo: McGraw-Hill Brasil, 2006

RAMOS, Ismael Araújo; AMORA, Márcio André Baima. Predição Just-In-Time de Defeitos em Software Utilizando Inteligência Artificial. In: **Anais do XLVI Seminário Integrado de Software e Hardware**. SBC, 2019. p. 113-124.

REZENDE, Denis Alcides. **Engenharia de software e sistemas de informação. Brasport**, 2006.

REZENDE, Denis Alcides. **Engenharia de Software e Sistemas de Informação**. Rio de Janeiro: Brasport, 2005.

RF, SAMPAIO. Estudos de revisão sistemática: um guia para síntese criteriosa da evidência científica. 2007

SALOMON, Valério Antonio Pamplona. **Desempenho da modelagem do auxílio à decisão por múltiplos critérios na análise do planejamento e controle da produção.** Universidade de São Paulo, 2004.

SAMETINGER, Johannes. **Software engineering with reusable components**. Springer Science & Business Media, 1997.

SANCHEZ-Palencia, Paola. **“A ATUAÇÃO DA ENGENHARIA DE SOFTWARE NA SOCIEDADE DO CONHECIMENTO.” A ATUAÇÃO DA ENGENHARIA DE SOFTWARE NA SOCIEDADE DO CONHECIMENTO** (2019): n. pag. 15.

SCARPI, Marinho Jorge. **METRICS DEVELOPMENT FOR THE QUALIS OF SOFTWARE TECHNICAL PRODUCTION**. Revista do Colégio Brasileiro de Cirurgiões, v. 42, p. 73-75, 2015.

SILVEIRA, Denise Tolfo; CÓRDOVA, Fernanda Peixoto. **Unidade 2–A pesquisa científica.** Métodos de pesquisa, v. 1, 2009.

SOMMERVILLE, Ian. Engenharia de Software, 9a. **São Palo, SP, Brasil**, 2011.

SOMMERVILLE, Ian**. Engenharia de Software**. São Paulo: Pearson Addison Wesley, 2005

SZYPERSKI, Clemens; GRUNTZ, Dominik; Murder, Stephan. **Software de componentes: além da programação orientada a objetos**. Pearson Education, 2002.

VICENTE, André Abe. **Uma Investigaç ao inicial da Atividade de Engenharia de Requisitos em ProcessosAgeis**.

APENDICE

Apêndice – 1 Formulário da pesquisa

Tendências emergentes de sucesso para a engenharia de software: Uma revisão sistemática

PARTE - I EXPLICAÇÃO DA PESQUISA

Este questionário é parte integrante de uma pesquisa de elaboração da monografia para obtenção de titulo de licenciado em computação, junto ao Instituto Federal de Educação, Ciência e tecnologia do Tocantins, sob a orientação de Professor Me. Rafael Miranda Correia.
Desta forma, esta pesquisa objetiva Identificar, verificar e propor o estudo das melhores tendências de sucesso como parte da ementa da disciplina de engenharia de software do curso de Licenciatura em Computação do Campus Porto Nacional do IFTO.
Assim, essa ferramenta de coleta de dados foi construída utilizando-se de perguntas fechadas e também com base na escala de Likert de 5 pontos.
Garantido, assim que as informações a serem inseridas serão utilizadas exclusivamente para fins desta pesquisa, não existindo a obrigatoriedade da parte II deste questionário onde são encontradas as perguntas sobre a identificação dos respondentes.

PARTE - II DADOS DOS RESPONDENTES

NOME

Sua resposta

TELEFONE

Sua resposta

SEXO

MASCULINO
FEMININO
Outro:

PARTE - III DADOS DA PESQUISA

QUAL A SUA INSTITUIÇÃO DE VINCULO?

PORTO NACIONAL - TO
OUTRO

VINCULO COM A INSTITUIÇÃO

DOCENTE
DISCENTE
Outro:

QUAL O GRAU DE CONHECIMENTO SOBRE ENGENHARIA DE DESENVOLVIMENTO DE SOFTWARE

1 - CONHEÇO POUQUÍSSIMO
2 - CONHEÇO POUCO
3 - CONHEÇO
4 - CONHEÇO MUITO
5 - CONHEÇO MUITÍSSIMO

QUAIS DESSAS TENDÊNCIAS DENTRO ENGENHARIA DE SOFTWARE VOCÊ JULGA MAIS IMPORTANTE PARA O ESTUDO JUNTO A DISCIPLINA DE ENGENHARIA DE SOFTWARE

ENGENHARIA DE DESENVOLVIMENTO ÁGEIS
ENGENHARIA DE CONFIANÇA
ENGENHARIA DE PROTEÇÃO
ENGENHARIA BASEADA EM COMPONENTES
ENGENHARIA ORIENTADA A ASPECTO
Outro:

CASO VOCÊ CONHEÇA OUTRA TENDENCIA QUE NÃO FOI CITADA NESTA PESQUISA, FAVOR INFORMAR

Sua resposta

ENVIAR

ANEXOS

Anexo – 1 Ementa da disciplina de Engenharia de software

Componente Curricular: Engenharia de Software
Carga Horária (CH) total: 66,7 CH teórica: 33,3 CH prática: 33,4
Pré-requisitos: Modelagem e Análise de Sistemas.
Ementa
Visão geral e princípios fundamentais da Engenharia de Software. Conhecimentos básicos sobre ciclo de vida de software e seus estágios: requisitos, projeto, implementação, gerenciamento, qualidade. Emprego de metodologias e ferramentas para análise e projeto de sistemas. Documentação de software. Engenharia Reversa. Reengenharia. Ferramentas CASE: conceitos, tipos e exemplos associando com as etapas do ciclo de vida de software.
Competências
• Apresentar, analisar e discutir o corpo de conhecimento que constitui a Engenharia de Software, seus princípios, métodos e ferramentas. • Identificar as técnicas da Engenharia de Software para o desenvolvimento de um produto.
Habilidades
• Ser capaz de descrever os métodos, ferramentas e procedimentos associados; • Conhecer e utilizar as técnicas de trabalho em grupo, especificamente para desenvolvimento de softwares; • Saber identificar os princípios da ética profissional do engenheiro de software.
Bibliografia básica
PRESSMAN, R. S. **Engenharia de Software**. 5ª ed. São Paulo: McGraw-Hill, 2007. PAULA FILHO, W. P. **Engenharia de software: fundamentos, métodos e padrões**. Rio de Janeiro: LTC, 2003. LTC, 2009.
Bibliografia complementar
PRESSMAN, R. **Software Engineering: A Practitioner's Approach.** EUA: G- Hill, 2005. SOMMERVILLE, I. **Engenharia de Software**. São Paulo: Pearson. 2007. DAVIS, A. M. **Software requirements**. EUA: Prentice Hall, 1993. PFLEEGER, S. L. **Engenharia de Software Teoria e Prática.** São Paulo: Prentice Hall, 2004. MAGELA, R. **Engenharia de softwares aplicada**. Rio de Janeiro: Alta Books, 2006.

Anexo – 2 Plano de Ensino

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS
CAMPUS PORTO NACIONAL
GERÊNCIA DE ENSINO
COORDENAÇÃO DO CURSO SUPERIOR DE LICENCIATURA EM COMPUTAÇÃO

PLANO DE ENSINO

ENGENHARIA DE SOFTWARE

Componente: **Engenharia de Software**

Carga horária **Total: 80 Carga horária Teórica: 40 Carga horária Prática: 40**

Ano/Semestre: **2019/1 Período: 6º**

Professor(a): **Rafael Miranda Correia** Curso: **Licenciatura em Computação**

1. EMENTA:

Introdução à Engenharia de Software: Conceitos e definições. Sistemas Computacionais. O que é software? O que é engenharia de software? O Ciclo de Vida do Software. Qualidade de Software. Processo de Software: Modelos de processo: Cascata, Prototipação, Evolutivo, Incremental, Transformação, Espiral. Slides. Modelos de processo: Processo Unificado.

Atividades do Processo: Especificação, Design e implementação, Validação e Evolução.

Métricas, Planejamento e

Gerenciamento de Software: Elaboração do cronograma. Planejamento da equipe. Estimativas e Métricas. Análise de riscos. Requisitos de Software: Requisitos e Engenharia de Requisitos. Definindo Requisitos com Casos de Uso. Slides. Modelos de Software. Design de Software: Design Conceitual, Prototipação. Arquitetura de Software conceitos, visão tradicional e visão emergente. Visões arquiteturais. Linguagens de Descrição Arquitetural. Padrões de Projeto. Framework. Verificação e

Validação de Software: Formas de verificação e validação de programas. Técnicas de testes. Manutenção e Evolução de Software.

2. COMPETÊNCIAS:

Apresentar, analisar e discutir o corpo de conhecimento que constitui a Engenharia de

Software, seus princípios, métodos e ferramentas. Identificar as técnicas da Engenharia de Software para o desenvolvimento de um produto; Ser capaz de descrever os métodos, ferramentas e procedimentos associados; Conhecer e utilizar as técnicas de trabalho em grupo, especificamente para desenvolvimento de softwares; Saber identificar os princípios da ética profissional do engenheiro de software.

3. HABILIDADES:

Durante o desenvolvimento desta disciplina o aluno terá a oportunidade de aprender os conceitos e praticas da engenharia de software e gerencia de projetos. Com isso, será capacitado a fazer analise documentada de um softwares por completo e também gerenciar o desenvolvimento de softwares.

4. BASES TECNOLÓGICAS:

4.1. Introdução à Engenharia de Software:

4.1.2. Conceitos e definições.

Entender os conceitos básicos de engenharia de software 4.1.3. Sistemas Computacionais.

Entender os tipos de sistemas computacionais 4.1.4. O que é software?

Entender o que é um software

4.1.5. O que é engenharia de software? Entender o que é a engenharia de software 4.1.6. O Ciclo de Vida do Software.

Entender o funcionamento de cada um dos ciclos de vida de um software 4.1.7. Qualidade de Software. Compreender o a qualidade de software 4.2. Processo de Software:

4.2.1. Modelos de processo:

4.2.2. Cascata, Prototipação,

Entender o tipos de processo de software cascata e prototipação

4.2.3. Evolutivo, Incremental,
Entender o tipos de processo de software evolutivo e incremental

4.2.4. Transformação,
Entender o tipos de processo de software transformação 4.2.5. Espiral.

Entender o tipos de processo de software espiral 4.3. Modelos de processo:

4.3.1. Processo Unificado. Compreender o processo unificado 4.4. Atividades do Processo:

4.4.1. Especificação,
Entender a especificação de processo

4.4.2. Design e implementação,
Entender sobre design e implementação de processos 4.4.3. Validação e Evolução.

Entender como fazer a validação e evolução 4.4.4. Métricas,

Entender as métricas.

4.5. Planejamento e Gerenciamento de Software:

4.5.1. Elaboração do cronograma.Aprender a elaborar cronograma
4.5.2. Planejamento da equipe. Entender sobre planejamento de equipe
4.5.3. Estimativas e Métricas. Entender as estimativas e métricas
4.5.4. Análise de riscos.
Fazer a analise dos riscos

4.6. Requisitos de Software:

4.6.1. Requisitos e Engenharia de Requisitos.Aprender a identificar os requisitos de softwares 4.6.2. Definindo Requisitos com Casos de Uso. Utilizar a uml para definir os casos de uso

4.6.3. Modelos de Software. Entender sobre os modelos de software

4.7. Design de Software:

4.7.1. Design Conceitual, Entender sobre o design conceitual 4.7.2. Prototipação.

Entender sobre prototipação

4.7.3. Arquitetura de Software conceitos,

Entender os conceitos da arquitetura de software 4.7.4. visão tradicional e visão emergente.

Entender sobre a visão tradicional e a visão emergente.

4.7.5. Visões arquiteturais.

Entender sobre as visões de arquitetura

4.7.6. Linguagens de Descrição Arquitetural.Entender sobre as linguagens de descrição da arquitetura 4.7.7. Padrões de Projeto.

Entender sobre os padrões de projetos 4.7.8. Framework.

Entender sobre framework

4.7.9. Verificação e Validação de Software:

4.7.10. Formas de verificação e validação de programas.Entender as formas de verificação e validação de programas 4.7.11. Técnicas de testes.

Entender as técnicas de testes

4.7.12. Manutenção e Evolução de Software. Entender sobre a manutenção e evolução de software

5. METODOLOGIA:

Aula teórica e prática através de exposição dialogada utilizando recurso como pincel, quadro, material impresso, data show, hardwares e computador. Realização de exercícios e trabalhos sobre os conceitos apresentados.

Aulas expositivas

• Aulas práticas e demonstrativas

• Exercícios práticos
• Textos Técnicos

6. RECURSOS DIDÁTICOS:

Pincel, quadro, material impresso, data show, hardwares e computadores

7. AVALIAÇÃO:

1º Bimestre uma prova valendo 7,0 e um trabalho valendo 3,0 assim totalizando 10,0 2º Bimestre uma prova valendo 7,0 e um trabalho valendo 3,0 assim totalizando 10,0

8. BIBLIOGRAFIA:

8.1 Básica

BEZERRA, E. Princípios de análise e projeto de sistemas com UML. 2. ed. Rio Janeiro: Campus, 2006. BOOCH, G.; JACOBSON, I.; RUMBAUGH, J. UML: Guia do Usuário. 2. ed.

Rio Janeiro: Campus, 2006. PAULA FILHO, W. P. Engenharia de software: fundamentos, métodos e padrões. Rio de Janeiro: LTC, 2003. REFERÊNCIAS BIBLIOGRÁFICAS PFLEEGER, S. L. Engenharia de software: teoria e prática. São Paulo: Pearson, 2004.

PRESSMAN, R. Engenharia de software. Rio de Janeiro: MacGraw-Hill, 2006.

SOMMERVILLE, I. Engenharia de software. 8. ed. São Paulo: Addison Wesley, 2007.

8.2 Complementar

FURLAN, J. D. Modelagem de objetos através da UML. São Paulo: Makron Books, 1998. LARMAN, C. Utilizando UML e padrões: um guia para a análise e projeto orientados a objetos. 3. Porto Alegre: Editora Bookman, 2007.

Rafael Miranda Correia

Coordenador do Curso Superior de Licenciatura em Computação

Portaria nº 109/2016/IFTO/ Campus Porto Nacional

Documento assinado eletronicamente por **Rafael Miranda Correia**, **Coordenador**, em 21/02/2019, às 21:55, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.