UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE RUSSAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

REBECA MAIA PONTES

STONE: UM PROCESSO DE GERÊNCIA DE CONFIGURAÇÃO DE SOFTWARE PARA PROJETOS ACADÊMICOS

Russas, CE
2019

REBECA MAIA PONTES

STONE: UM PROCESSO DE GERÊNCIA DE CONFIGURAÇÃO DE SOFTWARE PARA PROJETOS ACADÊMICOS

Monografia apresentada ao Curso de Graduação em Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Bacharel em Engenharia de Software.

Orientadora: Prof. Dra. Anna Beatriz dos Santos Marques
Co-orientadora: Prof. Dra. Valéria Lelli Leitão Dantas

Russas, CE
2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

P859s Pontes, Rebeca Maia.
STONE: Um Processo de Gerência de Configuração de Software para Projetos Acadêmicos / Rebeca Maia Pontes. – 2019.
91 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia de Software, Russas, 2019.
Orientação: Profa. Dra. Anna Beatriz dos Santos Marques. Coorientação: Profa. Dra. Valéria Lelli Leitão Dantas.

1. Gerência de Configuração de Software. 2. Processo de Software. 3. Projeto acadêmico. I. Título.

CDD 005.1

REBECA MAIA PONTES

STONE: UM PROCESSO DE GERÊNCIA DE CONFIGURAÇÃO DE SOFTWARE PARA PROJETOS ACADÊMICOS

Monografia apresentada ao Curso de Graduação em Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Bacharel em Engenharia de Software.

BANCA EXAMINADORA

_____

Profa. Dra. Anna Beatriz dos Santos Marques (Orientadora)
Universidade Federal do Ceará (UFC)

_____

Profa. Dra. Valéria Lelli Leitão Dantas (Co-orientadora)
Universidade Federal do Ceará (UFC)

_____

Prof. Ms. José Osvaldo Mesquita Chaves
Universidade Federal do Ceará (UFC)

A Deus.
A minha mãe, Marineide Paula Maia.

AGRADECIMENTOS

Às professoras Dr. Anna Beatriz dos Santos Marques e Dra. Valéria Lelli Leitão Dantas pela excelente orientação.

“O progresso é impossível sem mudança; e aqueles que não conseguem mudar as suas mentes não conseguem mudar nada.” *George Bernard Shaw*

RESUMO

A Gerência de Configuração de Software (GCS) é uma subárea da Engenharia de Software fundamental para o controle de todas as mudanças que ocorrem dentro de um projeto. Embora, sua aplicação traga muitos benefícios, como, por exemplo, maior rapidez na correção de problemas, seu uso não é encorajado nos projetos desenvolvidos por alunos durante seu período na universidade. Com a falta de um processo predefinido que apoie a GCS, estes projetos desenvolvidos frequentemente são entregues fora do prazo esperado, seus artefatos se tornam inconsistentes, ou seja, de alguma forma, a execução das atividades do projeto deixa pontos a desejar. Muitos modelos, como o MPS-BR, PMBoK e CMMI têm sua própria definição do processo de Gerência de Configuração, contudo, sua aplicação prática, normalmente realizada na indústria, necessita de mão-de-obra especializada e uma infraestrutura robusta para suportar as atividades de GCS, dificultando seu emprego em contexto acadêmico. Embora algumas práticas de GCS tenham sido aplicadas em âmbito acadêmico, como a implantação de uma ferramenta de controle de versão para projetos produzidos por alunos, não foi localizado na literatura, trabalhos que propusessem uma solução específica para os problemas causados pela falta de GCS em projetos de software desenvolvidos em ambiente acadêmico. Nesse contexto, este trabalho apresenta um processo de GCS voltado para projetos acadêmicos, a fim de auxiliar os alunos a controlarem mudanças e, consequentemente, evitarem certos tipos de problemas que possam vir a surgir. O processo definido foi baseado em boas práticas do MPS-BR, da ISO 10007 e de outros trabalhos publicados na área. Posteriormente foi realizada uma pesquisa através de questionários com alunos de graduação do curso de Computação e áreas afins para identificar suas principais dificuldades e validar algumas atividades do processo. A diante, foram realizadas alterações no processo, para que este se adeque às necessidades apontadas pelos alunos. Um experimento foi realizado a fim de avaliar o uso de duas ferramentas durante a execução da fase de Controle de Mudanças do processo STONE. Os dados resultantes foram coletados e interpretados, para que o processo fosse melhorado.

Palavras-chave: Gerência de Configuração de Software. Processo de Software. Projeto acadêmico.

ABSTRACT

Software Configuration Management (SCM) is a subarea of Software Engineering that is fundamental to the control of all changes that occur within a project. Although its application has many benefits, such as faster problem correction, its use is not encouraged in projects developed by students during their university period. Lacking a predefined process that supports SCM, these developed projects are often delivered on time, their artifacts become inconsistent, meaning that somehow the execution of project activities leaves points to be desired. Many models, such as MPS-BR, PMBoK, and CMMI have their own definition of the Configuration Management process, however, their practical application, usually performed in the industry, requires skilled labor and a robust infrastructure to support the SCM activities, making it difficult to use in the academic context. Although some SCM practices have been applied at an academic level, such as the implementation of a version control tool for student-produced projects, it has not been found in the literature, papers that proposed a specific solution to the problems caused by the lack of SCM in projects. of software developed in an academic environment. In this context, this paper presents a SCM process focused on academic projects, in order to help students control changes and, consequently, avoid certain types of problems that may arise. The defined process was based on good practices from MPS-BR, ISO 10007 and other works published in the area. Subsequently, a survey was conducted through questionnaires with undergraduate students of the Computer course and related areas to identify their main difficulties and validate some activities of the process. With this information collected, changes were made in the process, so that it suits the needs pointed out by the students. An experiment was performed to evaluate the use of two tools during the execution of the Change Control phase of the STONE process. The resulting data were collected and interpreted so that the process could be improved.

Keywords: Software Configuration Management. Software Process. Academic project.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BPM	Business Process Management
CMMI	Capability Maturity Model Integration
GCS	Gerência de Configuração de Software
IEEE	Institute of Electrical and Electronic Engineers
ISO	International Organization for Standardization
MPS.BR	Melhoria de Processo de Software Brasileiro
PMBoK	Project Management Body of Knowledge
SAAS	Software As A Service
STONE	Software configuraTion maNagement acadEmic
SWEBoK	Software Engineering Body of Knowledge

SUMÁRIO

1 INTRODUÇÃO

Segundo o SWEBOK (2001), a GCS é uma disciplina da Engenharia de Software que tem por objetivo controlar todas as mudanças de um software para que seja mantida a integridade e rastreabilidade dos seus artefatos, durante seu ciclo de vida. Esta definição, embora seja intuitiva e de fácil compreensão, ela dá uma perspectiva restrita do que a GCS aborda no ciclo de vida de um projeto de software.

De uma perspectiva gerencial, a norma ISO/IEC 12207 (1998) estabelece que

> o processo de gerência de configuração é um processo de aplicação de procedimentos administrativos e técnicos, por todo o ciclo de vida de software, destinado a identificar e definir os itens de software em um sistema, e estabelecer suas linhas básicas (baseline); controlar as modificações e liberações dos itens; registrar e apresentar a situação dos itens e dos pedidos de modificação; garantir a completude, a consistência e a correção dos itens; e controlar o armazenamento, a manipulação e a distribuição dos itens.

Com base nessa definição, a GCS é discriminada em atividades gerenciais, fornecendo uma visão mais ampla do processo. No entanto, essas atividades geralmente aplicadas a uma organização que necessita de um certo grau de rigidez na execução das atividades de seus projetos, o que acaba apresentando uma visão mais formal da que visão de GCS que é proposta neste trabalho, portanto não é a definição mais adequada a ser utilizada.

De uma diferente perspectiva, Humble e Farley (2011) afirmam que "*a gerência de configuração se refere ao processo pelo qual todos os artefatos relevantes para o seu projeto e os relacionamentos entre eles são armazenados, recuperados, identificados exclusivamente e modificados*". Para o contexto deste trabalho, será utilizado esta definição informal.

Vários trabalhos da área salientam a necessidade do uso da GCS, a fim de reafirmar a importância de sua aplicação em projetos de desenvolvimento de software. Segundo Khraiwesh (2017)

> a Gerência de Configuração (GC) é uma atividade fundamental presente durante todo o ciclo de vida de desenvolvimento de um software. Ela se torna essencial se qualquer um dos artefatos for alterado durante o ciclo de vida do projeto. Há muitos métodos que podem ser usados em GC para agilizar, com foco na qualidade, o trabalho produzido no projeto de desenvolvimento de um software. Para este propósito, a maioria das empresas adotam práticas de GC com times grandes para implementar estas práticas. É dito por muitos pesquisadores que executar com eficiência os processos de GC não somente aumenta a segurança em organizações, mas também demonstra efeitos positivos sobre o retorno do investimento, os custos do produto, as datas de entrega e a qualidade do produto.

Atualmente, devido ao processo de desenvolvimento de software acelerado e ao trabalho em equipes focadas em produtividade, guardar todo o histórico de alterações do programa passou a ser algo essencial para a gerência da organização. Passou-se a manter não somente o artefato, mas o que foi modificado, quando e como foi modificado, quem modificou, quanto tempo levou a modificação, qual o impacto da mudança, qual a produtividade associada etc. (MOLINARI,2007). A partir disto, surgiu a necessidade de uma área de processo responsável somente para realizar o controle de mudanças que ocorre dentro de um projeto.

Já em contexto acadêmico, a GCS se faz necessária principalmente para os alunos manterem consistência entre os artefatos produzidos no projeto. Apesar de sua aplicação ser menos formal do que na indústria, foi observado, informalmente, que projetos pequenos que não aplicam práticas de GCS frequentemente não correspondem às expectativas do professor, seja por perda de funcionalidades ou mal funcionamento dos componentes do sistema ou redundância de dados, entre outros problemas.

Além disso, quando as práticas da GCS são implementadas, pode-se obter os seguintes benefícios (PMI, 2007):

• Os fluxos de comunicação entre os membros do projeto ficam mais claros;

• Registros são disponibilizados para apoiar a auditoria dos requisitos dos *stakeholders*;

• A integração entre planejamento, execução e controle de mudanças ajuda a manter uma sincronia entre os requisitos e os resultados finais do projeto;

• A garantia de que apenas as mudanças aprovadas serão incorporadas dentro do projeto.

Portanto, a Gerência de Configuração fornece informações completas a respeito de todos os componentes do sistema de serviços que permite a outros processos funcionarem de forma mais efetiva e eficiente (MPS-BR, 2013).

Contudo, embora a GCS tenha importantes benefícios, sua aplicação em contexto acadêmico é raramente utilizada.

O ensino de Engenharia de Software e outros cursos da área da Computação nas universidades estabelece que os alunos devem ter alguma experiência prática nas disciplinas que cursam. Para que os alunos adquiram essa experiência, os professores geralmente passam trabalhos ao fim ou ao longo da cadeira com o intuito de enriquecer o conhecimento destes alunos a respeito de determinado assunto. Contudo, conforme são feitas as entregas e os alunos se aprofundam mais no desenvolvimento das atividades desses trabalhos, seus artefatos passam por mudanças, e quando estas mudanças não são gerenciadas corretamente, os produtos são entregues fora do prazo, não atendem os requisitos definidos pelo professor, falta consistência entre os artefatos, ou seja, os alunos não se tornam aptos a lidar com a capacidade de mutabilidade inerente a todos os projetos de desenvolvimento de software, uma característica essencial entre os profissionais da área de TI.

Para prevenir esses problemas, diversos trabalhos publicados na área propõem atividades e técnicas de Gerência de Configuração de Software, para auxiliar as equipes de desenvolvimento a controlar as mudanças que ocorrem em seus projetos. O trabalho de Duran et. al. (2005), descreve um estudo de caso sobre o diagnóstico do nível de institucionalização do processo de gerência de configuração, baseado nas disciplinas de Gerências de Configuração e Mudança do RUP (RATIONAL, 2002), nas atividades de desenvolvimento de software de uma organização. Em comparação, o trabalho de Makiaho, Poranen e Seppi (2014) faz um estudo sobre ferramentas de controle de versão em projetos de desenvolvimento de software acadêmicos.

Com esses e outros trabalhos pesquisados na literatura, foi observado que não há um trabalho que unifique as principais atividades da GCS a fim de auxiliar sua aderência pelos projetos desenvolvidos pelos alunos durante seu período na universidade. Muitos dos trabalhos de GCS com foco em projetos acadêmicos se preocupam apenas com a fase de codificação e controle de versão do código, negligenciando as outras atividades desenvolvidas pelos alunos, como a documentação dos requisitos do sistema, a elaboração de diagramas de projeto, entre outros.

Portanto, o objetivo deste trabalho é propor um processo de GCS, nomeado STONE (Software configuraTion maNagement acadEmic), voltado especificamente para projetos acadêmicos com o intuito de auxiliar os alunos a gerenciar as mudanças dos artefatos produzidos nas disciplinas, podendo utilizá-lo nos semestres iniciais do curso, além de familiarizá-los com as técnicas e procedimentos de GCS. O propósito deste trabalho é responder as seguintes questões de pesquisa:

Q1. De que maneira a gerência de configuração pode ajudar no desenvolvimento de projetos de software desenvolvidos em âmbito acadêmico?

Q2. Quais os elementos de um processo de GCS tradicional podem ser utilizados para auxiliar os estudantes no controle de mudanças de seus projetos acadêmicos?

Q3. Quais ferramentas são utilizadas em projetos acadêmicos para fazer controle de versões de artefatos de software?

Este trabalho está organizado da seguinte forma: Na Seção 2, os principais trabalhos relacionados à pesquisa são apresentados, descrevendo o objetivo de cada um, de que forma ele contribui para a pesquisa e, descrevendo em que aspecto a proposta se difere do trabalho relacionado. Na Seção 3, é explanado os objetivos geral e específicos deste trabalho. Em seguida, na Seção 4 é apresentado os principais conceitos abordados. Na Seção 5 é definido o procedimento metodológico, contendo o cronograma para execução deste trabalho. Na Seção 6 é feita a descrição do processo de GCS proposto por este trabalho. Na Seção 7 é explanada a análise das necessidades dos alunos em relação a GCS. Na Seção 8 é descrita como foi avaliada duas ferramentas para a fase de ‘Controle de Mudanças’ do processo STONE. Na Seção 9 é feita uma discussão apontando as descobertas feitas para cada questão de pesquisa especificada na introdução. Por fim, na Seção 10 é descrita a conclusão deste trabalho e atividades futuras.

2 TRABALHOS RELACIONADOS

Nesta seção são apresentados alguns trabalhos científicos relevantes a este projeto de pesquisa. Os trabalhos pesquisados foram divididos em quatro tópicos: trabalhos que propõem soluções de GCS em contexto acadêmico (subseção 2.1), trabalhos voltados para o controle de versão (subseção 2.2), para o gerenciamento de releases de projetos (subseção 2.3) e para a gerência de mudanças (subseção 2.4) em projetos de software.

2.1 Trabalhos de GCS em contexto acadêmico

Em seu artigo, Conn (2004) propõe um processo de Engenharia de Software direcionado a projetos acadêmicos. Ele foi projetado para ser facilmente adaptado e modificado a fim de se adequar às necessidades dos alunos. Apesar de abranger cada fase do ciclo de vida do desenvolvimento de software, e conter apenas uma atividade relacionada ao controle da configuração, foi possível adaptar esta atividade de forma que se adequasse ao processo proposto neste trabalho. O processo proposto por Conn (2004) define que o gerente de suporte coloca todos os produtos criados ou modificados sob o controle de configuração. No entanto, o passo-a-passo para atingir este resultado não é estabelecido, dando margem a erros à sua execução. Diferente do processo de Conn (2004), este trabalho fornece atividades e tarefas detalhadas de como colocar os produtos do projeto sob o controle de configuração.

No artigo de Cho, Gray e Sun (2012) são apresentados vários princípios de Engenharia de Software (e.g., gerenciamento de requisitos, design, implementação, teste, gerência de configuração e gerenciamento de *releases*) a serem considerados no desenvolvimento de ferramentas de software para fins de pesquisa acadêmica. Também é realizada uma *survey* das ferramentas que foram apresentadas nas principais conferências para examinar o estado do desenvolvimento de ferramentas de pesquisa acadêmica em termos desses princípios. Com os dados retirados da *survey*, percebeu-se que o uso de ferramentas de GCS aumenta a capacidade de desenvolvimento dos programadores, e consequentemente a qualidade do software, e, portanto, seu uso deve ser encorajado. O trabalho de Cho, Gray e Sun (2012), embora seja aplicado alguns fatores de qualidade para analisar o desenvolvimento dessas ferramentas, não são fornecidos fatores de qualidade de GCS que podem se aplicar às demais fases do processo do ciclo de vida do software, como análise, design. O processo proposto por este trabalho difere do trabalho de Cho, Gray e Sun (2012), pois, além de ser aplicável à etapa de codificação de um software, ele fornece uma visão holística da GCS, com base nas necessidades dos alunos.

O trabalho de Torchiano e Bruno (2018) propõe a integração de uma infraestrutura capaz de suportar cursos OOP a fim de incentivar os alunos a aplicarem práticas-chave de engenharia de software (e.g. testes automatizados, gerenciamento de configuração) – e, consequentemente adquirirem as habilidades básicas para usar ferramentas correspondentes. Como resultado, o nível de satisfação dos alunos foi considerado alto, no entanto, foi apontado que muitos alunos tinham dificuldade em utilizar ferramentas básicas, como por exemplo, ferramentas para controle de versão. O processo de GCS proposto neste trabalho, diferentemente do trabalho de Torchiano e Bruno (2018), suporta o uso de ferramentas básicas de GCS, mas estas ficam à escolha do aluno, portanto o nível de dificuldade de sua utilização será bem reduzido.

O trabalho de Haley, Collins e Coe (2007) propõe uma maneira de documentar mudanças em projetos de software acadêmicos através de wikis. Uma wiki é definida como um "banco de dados *online*" para trabalho. De maneira simplificada, é um site onde é feito o gerenciamento de comunicação entre os participantes do projeto, podendo ser utilizado como plataforma de aprendizagem cooperativa. Esse trabalho, embora proponha uma solução para o gerenciamento de mudanças em projetos de software, utiliza tecnologia que está se tornando obsoleta, portanto, não é adequado seu uso. O processo proposto por este trabalho dá suporte ao uso de ferramentas para o gerenciamento de comunicação entre os membros do projeto, no entanto, a escolha de tal ferramenta fica à escolha dos envolvidos.

O trabalho de Ríos *et al.* (2019) relata um projeto que explora a forma como os alunos resolvem tarefas usando plataformas de desenvolvimento colaborativo e sistemas de controle de versão, como o GitLab, para encontrar padrões e métricas de avaliação que podem ser usadas para melhorar o conteúdo do curso e refletir sobre os problemas mais comuns que os alunos estão encontrando. Diferente dos outros trabalhos, este tem foco em controle de versão e integração contínua aplicados a projetos desenvolvidos na academia, contudo ele não aborda a GCS nas fases de planejamento e controle de mudanças, tornando sua aplicação restrita.

2.2 Trabalhos com foco em controle de versão

Em seu artigo, Makiaho, Poranen e Seppi (2014) fazem um estudo sobre o uso de ferramentas de controle de versão em projetos de desenvolvimento acadêmicos. Eles analisaram os desafios e as principais maneiras de usar controle de versão durante o treinamento, desenvolvimento e gerenciamento do projeto, e a partir deste estudo, os autores observaram que o entendimento dos princípios mais importantes do uso do controle de versão (CV) nem sempre é claro aos estudantes e isso deve ser levado em conta em sala de aula. Além disso, muitos alunos não estavam familiarizados com as ferramentas de CV, ou seja, não conseguiam usá-las de forma eficiente, e os poucos que sabiam, não as empregavam de maneira adequada.

Os trabalhos (KERTESZ, 2015), (FELICIANO, STOREY e ZAGALSKY, 2016), (ARORA, GOEL e MITTAL, 2017) e (TIRKEY e GARY, 2017) apresentam resultados do uso de ferramentas como Git e GitHub por alunos, onde o foco principal estava na interação direta dos alunos no processo de aprendizagem e desenvolvimento de software cooperativo. Uma importante observação em comum desses trabalhos foi que a GCS auxiliou os alunos a aprenderem com os erros dos colegas, deixando-os mais atentos a diferentes soluções para o mesmo problema e levando-os à melhores decisões de como implementar certas tarefas. Tais trabalhos dão suporte a presença remota de professores e estudantes, facilitando a educação à distância, monitoramento e validação de projetos de desenvolvimento de software de maneira remota.

O trabalho de Apel, Leßenich e Lengauer (2012) apresenta uma abordagem para gerenciamento de *merges*, balanceando a precisão da detecção de conflitos e a performance da execução desta abordagem. A fim de se comprovar sua eficiência, foi implementada uma ferramenta que segue esta abordagem. O diferencial desse trabalho é que muitas ferramentas no mercado dão preferência à precisão ao detectar conflitos entre *merges* de diferentes desenvolvedores, o que pode causar problemas de performance. O objetivo era desenvolver

uma abordagem que equilibrasse esses dois princípios. Esta abordagem é ideal para projetos onde muitos desenvolvedores estão envolvidos e continuamente precisam versionar o código que estão programando ou alterando. Um ponto a desejar da ferramenta desenvolvida é que ela suporta apenas programas Java.

O trabalho de Rayana *et. al.* (2016) apresenta uma abordagem composta de modelos de controle de versão e *branching* baseado no Git integrando um conjunto de ferramentas para lidar com problemas, como a necessidade de oferecer suporte a versões antigas de produtos e fornecer correções de erros críticos.

O trabalho de Gowtham (2014) descreve as experiências obtidas ao tentar implementar um Sistema de Controle de Revisão distribuído, Git, como parte do currículo dos cursos de Engenharia e Ciência da Computação nos níveis de graduação e pós-graduação. A partir desta integração, os alunos aproveitaram a oportunidade para continuar trabalhando em diferentes locais de forma eficiente. Além disso, o Git também serviu como *backup* para que professores e alunos não perdessem trabalho semestral devido a falhas nos discos rígidos e / ou computadores.

O trabalho de Krusche, Berisha e Bruegge (2016) descreve uma técnica de revisão informal: revisões de código baseadas em *branches*. Antes de integrar o código-fonte no repositório central, o código é analisado de forma assíncrona pela Internet em um portal de qualidade para identificar defeitos, falhas de design e falhas de código. Após uma avaliação em projetos desenvolvidos por alunos, percebeu-se que a técnica facilita a colaboração e a transferência de conhecimento entre os desenvolvedores, no entanto quando a carga de trabalho aumenta, os alunos não conseguem lidar com isso, acarretando em atrasos de entrega. Também é necessário que os alunos já tenham experiência prévia com desenvolvimento de software. Quanto menos experiência a equipe tiver, menos eficiente será a aplicação da técnica. O processo proposto por este trabalho, diferentemente do trabalho de Krusche, Berisha e Bruegge (2016), dá suporte à aplicação de técnicas de controle de versão de uma maneira mais abstrata, ou seja, o aluno escolhe a técnica que estiver mais familiarizado, diminuindo a probabilidade de erros durante sua aplicação.

O trabalho de Linsbauer, Egyed e Lopez-Herrejon (2016) apresenta um processo de desenvolvimento e ferramentas para gerenciamento de sistemas altamente configuráveis. A abordagem foi avaliada, através de um estudo de caso, com vários domínios e origens diferentes, como sistemas *open source*, acadêmicos e da indústria. A principal contribuição desse trabalho foi um processo de desenvolvimento genérico, incremental, intuitivo e flexível e com suporte a ferramentas que destacam os benefícios de abordagens *ad hoc*.

O trabalho de Estublier e Garcia (2006) propõe um sistema que suporta desenvolvimento de software concorrente, provendo controle sobre custo, qualidade e tempo de desenvolvimento, onde controle de *merge* é a chave.

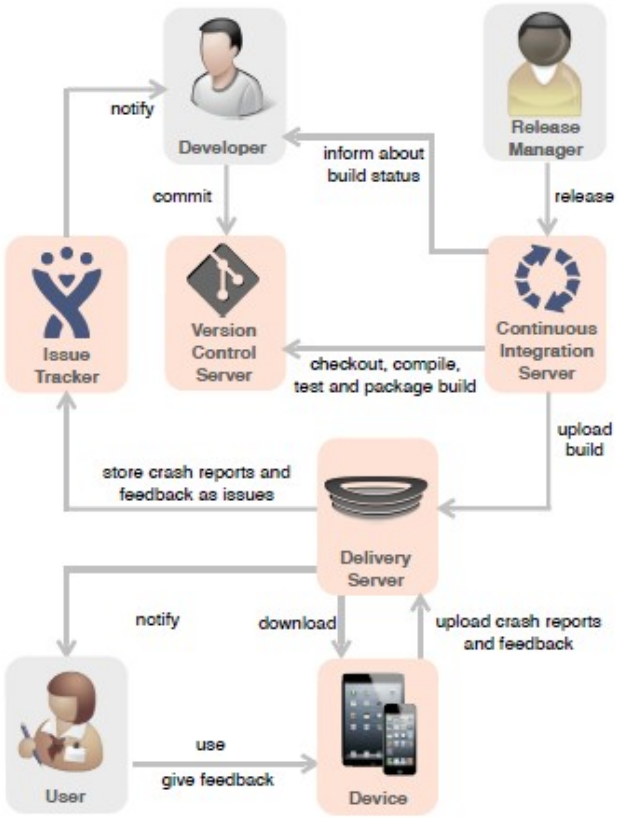
Todos os trabalhos acima se diferem do processo proposto por este trabalho no mesmo ponto. O controle de versão é apenas uma etapa da GCS, ou seja, as demais etapas (e. g. identificação, planejamento da GCS), são negligenciadas, restringindo o escopo do que a GCS pode oferecer ao longo de todo o ciclo de vida do desenvolvimento do projeto.

2.3 Trabalhos com foco em gerenciamento de releases

O trabalho de Agarwal (2011) apresenta uma variação do Scrum com foco no gerenciamento de SAAS. Nessa abordagem, correções de bugs, pequenas melhorias, são lançadas semanalmente, por uma única equipe, fornecendo um ciclo de *releases* rápido.

O trabalho de Krusche *et al.* (2014) apresenta um modelo de processo baseado em entrega contínua, chamado Rubgy. A partir da avaliação desse modelo, foi mostrado que o Rugby melhora a interação entre desenvolvedores e clientes com um mecanismo de *feedback* contínuo, além de melhorar a coordenação da com as partes interessadas através de *releases* baseadas em eventos. A Figura 1 mostra o fluxo de trabalho do Rugby.

Figura 1 – Fluxo de trabalho do Rubgy



Fonte: Krusche *et al.* (2014)

O fluxo de trabalho é iniciado sempre que um desenvolvedor envia o código-fonte para o servidor de controle de versão, levando a um novo *build* no servidor de integração contínua. Se o *build* foi criado com sucesso e se passou em todos os estágios de teste, a equipe pode decidir enviá-la para o servidor de entrega que notifica os usuários sobre um novo *release*. O usuário pode baixar o *release* e reconhecer facilmente quais recursos e bugs foram resolvidos. Ele pode usar um mecanismo incorporado para fornecer *feedback* de maneira estruturada. Esse feedback é coletado no servidor de entrega e encaminhado para *issue tracker* que informa o gerente de *releases*.

Este fluxo de trabalho foi simplificado e adaptado para as fases de Controle de Versão e Gerenciamento de *Releases* do processo proposto por este trabalho. As atividades desta fase estão descritas da seção 6.2.6 à seção 6.2.10.

O trabalho de Ferreira e Langerman (2014) incorpora descobertas onde grandes empresas com projetos globais de TI podem adotar a estrutura conceitual ágil e avalia se diferentes tipos de projetos de TI se beneficiarão de uma abordagem de gerenciamento de *releases* contínuo para a entrega do projeto. Os principais resultados obtidos desse estudo demonstraram que os *releases* frequentes têm um impacto positivo em projetos de software e que os ambientes de trabalhos são os maiores obstáculos para adoção de uma estratégia de gerenciamento de *releases* contínuo.

Os trabalhos supracitados se diferem do processo proposto por este trabalho no mesmo ponto: todos apresentam apenas uma parte da GCS, que é o gerenciamento de *releases*. A perspectiva do gerenciamento de *releases* por si só não é suficiente para abordar as demais fases do ciclo de vida do desenvolvimento de um software, ou seja, a GCS não é aplicada em sua completude, reduzindo seus benefícios a longo prazo.

2.4 Trabalhos com foco em gerenciamento de mudanças

O trabalho de Linsbauer, Berger e Grunbacher (2017) apresenta uma comparação e classificação entre sistemas de controle de variabilidade a fim de compreender as capacidades e vantagens de cada um. Esse trabalho fornece uma classificação de seis sistemas e mostra que estes usam conceitos e abordagens desenvolvidos nas áreas de gerenciamento de configuração de software e engenharia de linhas de produtos de software e com base nos resultados foi discutida as razões que podem impedir o uso em larga escala desses sistemas, como por exemplo a necessidade da execução de um fluxo de trabalho complexo. Como o trabalho de Linsbauer, Berger e Grunbacher (2017), embora apresente uma perspectiva do gerenciamento de mudanças a partir do uso de ferramentas, não mostra resultados favoráveis para o uso dessas ferramentas em projetos acadêmicos, portanto foi incorporado neste trabalho.

O trabalho de Bashir e Qadir (2006) apresenta um levantamento das técnicas de rastreabilidade. O trabalho identificou problemas com a definição de rastreabilidade e os critérios que são usados para avaliar as técnicas de rastreabilidade. A avaliação mostra que as técnicas de rastreabilidade são deficientes e podem causar problemas no gerenciamento dos requisitos. A partir dessa avaliação, foi argumentado que as técnicas existentes podem ser combinadas para abordar as deficiências umas das outras e aproveitar ao máximo os benefícios da rastreabilidade. Este trabalho apresenta soluções de uma atividade do gerenciamento, que é a manutenção da rastreabilidade entre os artefatos do projeto. Isto é feito toda vez que um artefato é criado ou alterado. O processo proposto por este trabalho define tarefas de rastreabilidade, no entanto, não impõe o uso das técnicas específicas.

O trabalho de Fasano (2007) investiga o gerenciamento de artefatos de software através do uso de uma ferramenta, abordando o gerenciamento de configuração de artefatos e o gerenciamento de rastreabilidade para suporte à coordenação de times distribuídos durante a evolução do software. Esta ferramenta gerencia principalmente documentos de software, ou seja, não dá suporte à manutenção da rastreabilidade entre a documentação do projeto e o código-fonte produzido pelos desenvolvedores. No processo proposto por este trabalho, a rastreabilidade entre os artefatos será mantida entre quaisquer tipos de produtos de trabalho, seja documentação e/ou código, a fim de se atingir máxima cobertura.

O trabalho de Crnkovic, Funk e Larsson (1999) apresenta um modelo para gerenciamento de requisitos através de atividades do gerenciamento de mudanças. Esse modelo fornece suporte no controle de como e quando os requisitos são alterados, se estão em implementação ou se já foram implementados. No entanto, o gerenciamento das mudanças de requisitos não funcionais é particularmente difícil de relatar, dando margem a ambiguidades. O processo por este trabalho difere do trabalho de Crnkovic, Funk e Larsson (1999) porque ele não se restringe apenas ao controle de mudanças de requisitos e sim aos demais produtos de trabalhos do projeto.

O trabalho de Shams, Sharif e Kermanshah (2017) apresenta um framework para o gerenciamento de mudanças "absoluto", buscando atingir os principais desafios desta área. A partir da solução desses desafios foi montado um meta-framework. Contudo, tal framework não foi validado então não é possível comprovar sua eficácia. Como o processo proposto por este trabalho tem foco em projetos acadêmicos, os desafios do gerenciamento de mudanças apresentados pelo trabalho Shams, Sharif e Kermanshah (2017) diferem das reais necessidades dos alunos.

2.5 Comparativo entre os trabalhos

Para estabelecer uma analogia entre os trabalhos pesquisados, é apresentada na Tabela 1. Nesta tabela, os trabalhos são comparados através de 3 perspectivas: se propõem algum processo, abordagem ou técnica de GCS; se analisam a aplicação ou propõem uma ferramenta de GCS; e se têm foco em projetos acadêmicos.

Tabela 1 – Comparativo entre os trabalhos relacionados

Título	Processo/ Abordagem/ Técnica	Aplicação/Análise de Ferramenta(s)	Com foco em Projeto/Pesquisa Acadêmico(a)
Este trabalho	X	X	X
Conn (2004)	X		X
Cho, Gray e Sun (2012)		X	X
Makiaho, Poranen e Seppi (2014)		X	X
Kertesz (2015)		X	X
Torchiano e Bruno (2018)		X	X
Haley, Collins e Coe (2007)		X	X
Ríos et al. (2019)		X	X
Feliciano, Storey e Zagalsky (2016)		X	X
Arora, Goel e Mittal (2017)		X	X

Tirkey e Gary (2017)		X	X
Apel, Leßenich e Lengauer (2012)	X	X	
Rayana et al. (2016)		X	
Gowtham (2014)		X	X
Krusche, Berisha e Bruegge (2016)	X		
Linsbauer, Egyed e Lopez-Herrejon (2016)	X	X	
Estublier e Garcia (2006)		X	
Agarwal (2011)	X		
Krusche et al. (2014)	X	X	
Ferreira e Langerman (2014)	X	X	
Linsbauer, Berger e Grunbacher (2017)		X	
Bashir e Qadir (2006)	X	X	
Fasano (2007)		X	
Crnkovic, Funk e Larsson (1999)	X		
Shams, Sharif e Kermanshah (2017)	X		

O diferencial deste trabalho é que este abrange as principais etapas da GCS, aplicadas a todas as fases do ciclo de vida de desenvolvimento de um projeto de software de forma, propondo um processo de forma que os alunos possam aplicá-lo em seus projetos sem maiores dificuldades, dando suporte para o uso de ferramentas conhecidas.

3 OBJETIVOS

Nesta seção serão explanadas as metas para este trabalho. Essas metas estão discriminadas nas subseções 3.1 e 3.2, onde se encontram o objetivo geral e os objetivos específicos, respectivamente.

3.1 Objetivo Geral

O objetivo geral deste trabalho é definir um processo que auxilie os alunos a gerenciar as mudanças que ocorrem em seus projetos acadêmicos.

3.2 Objetivos Específicos

- Definir atividades de planejamento do processo de GCS para que seja adaptável a qualquer tipo de projeto que os alunos estejam desenvolvendo;
- Definir atividades de GCS que possibilitem os controles de mudança e de versão, dos artefatos produzidos ao longo do projeto;
- Sugerir ferramentas que automatizem das atividades de do processo de GCS;
- Propor e definir artefatos específicos para manter o controle dos itens de configuração de projetos acadêmicos;
- Documentar o processo de GCS voltado para projetos acadêmicos de forma que possa ser aplicado por alunos em diferentes tipos de projetos de software.

4 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os principais conceitos abordados pelo trabalho, sendo subdividida em duas subseções. A subseção 4.1 apresenta conceitos específicos da área de Gerência de Configuração de Software. A subseção 4.2 apresenta conceitos necessários para o entendimento de um processo de software, e a subseção 4.2.1 exemplifica diferentes usos da GCS com base em modelos de qualidade conhecidos.

4.1 Gerência de Configuração de Software

A GCS é uma área que utiliza muitos conceitos específicos e para que possamos compreendê-la totalmente é necessário que estes conceitos sejam bem apresentados.

Segundo o SWEBOK (2001), "*um item de configuração (IC) é um item ou agregação de hardware ou software ou ambos que foi projetado para ser gerenciado como uma única entidade. Um item de configuração de software é uma entidade de software que foi estabelecida como item de configuração*". Wazlawick (2013), define item de configuração de software como um elemento unitário para efeito de controle de versão, ou, ainda, um agregado de elementos que são tratados como uma entidade única no sistema de gerenciamento de configuração. Em suma, um IC é um artefato que necessita ser mantido sob o controle de mudanças estabelecido pelo processo de GCS. Ele pode ser qualquer coisa: um executável, uma aplicação corporativa, um documento, entre outros. Tudo depende da necessidade. (MOLINARI, 2007)

Como principal IC para a execução do processo de GCS, é necessário elaborar o Plano de Gerência de Configuração (PGC). O resultado da fase de planejamento da GCS para um determinado projeto é documentado no PGC, um "documento vivo" que serve como referência para o processo de GCS. Ele é mantido, isto é, atualizado e aprovado, conforme o necessário durante o ciclo de vida do software. (SWEBOK, 2001). Ou seja, é no Plano que é descrito a abordagem de GCS adota pelo projeto em que está sendo aplicado, contendo os procedimentos, políticas, cronogramas e responsabilidades.

Uma das tarefas que compõem a elaboração do PGC, é a definição da *baseline*. Segundo Pressman (2011), "*no contexto de Engenharia de Software, definimos uma baseline como um marco de referência no desenvolvimento de um software, que é caracterizado pela entrega de um ou mais itens de configuração*."

A maioria das atividades de GCS são dependentes do conceito de *baseline*. Vários benefícios deste conceito incluem (IEEE Guide to Software Configuration Management, 1988):

1. Distinguir entre diferentes *releases* internas para a entrega ao cliente;
2. Ajudar a garantir uma documentação completa do produto técnico;
3. Reforçar padrões de garantia de qualidade de software;
4. Ser usado como meio de promover, isto é, realizar entregas internas, cada IC de uma fase de desenvolvimento ou teste para outra.

Conforme os ICs forem sendo produzidos, é necessário mantê-los em um repositório. O repositório é uma coleção de todos o os artefatos de software pertencentes a um sistema e a localização/formato na qual tal coleção é armazenada. (ISO/IEC 24765:2009). O propósito deste armazenamento é garantir que o IC não desaparecerá ou será danificado, de maneira que este possa ser encontrado a qualquer momento e entregue nas condições que se deseja encontrar. (MOLINARI, 2007).

A cada alteração que um IC for submetido, é preciso criar uma nova versão para este IC. Segundo o SWEBOK (2001), "*a versão de um item de software é uma instância identificada de um item. Pode ser visto como o estado de um item em evolução*". Comumente, um projeto de software possui inúmeros ICs e cada um destes precisam ter suas versões controladas. O controle de versão fornece os seguintes serviços (MOLINARI, 2007):

- Identificação, armazenamento e gerenciamento dos ICs e de suas versões durante todo o ciclo de vida do software;
- Histórico de todas as alterações efetuadas nos ICs;
- Criação de rótulos e ramificações no projeto;
- Recuperação de uma configuração em um determinado momento desejado.

Quando um IC é liberado ao cliente, ele passa a ser denominado de *release*. Segundo a ISO/IEC 24765 (2009) um *release* "é uma versão entregável de uma aplicação que pode incluir toda ou parte de uma aplicação, uma coleção de ICs novos e/ou alterados que foram testados e introduzido em um ambiente." Ou seja, um *release* é qualquer artefato de software que foi entregue ao cliente. Essas entregas geralmente são feitas incrementalmente.

Como todo IC é passível de sofrer mudanças, é imprescindível que o processo de GCS possua um conjunto de atividades que fique responsável por gerenciar as mudanças desses ICs. Segundo Wazlawick (2013), "*A gerência de mudanças tem por objetivo mapear, para cada mudança efetuada no sistema, qual foi o motivo que gerou esta mudança*". A gerência de mudanças é uma parte importante da gerência de configuração que permite saber por que determinada versão de um IC foi sucedida por outra. Um típico controle de mudança de um sistema de software vai indicar quais funcionalidades foram adicionadas, removidas ou modificadas, quais defeitos foram corrigidos e, eventualmente, quais pendências ainda restam para uma versão futura.

4.2 Processo de software

Segundo Pfleeger (2004), processo é "*uma série de etapas que envolvem atividades, restrições e recursos para alcançar a saída desejada*". Essa definição é muito sucinta, e para o contexto deste é necessário definir o que é um processo de software.

Pode-se conceituar processo de software como uma sequência coerente de práticas que objetiva o desenvolvimento ou evolução de sistemas de software. Estas práticas englobam as atividades de especificação, projeto, implementação, testes e caracterizam-se pela interação de ferramentas, pessoas e métodos. (MAGELA, 2006).

Segundo Pfleeger (2004), atividade é "*alguma coisa que acontecerá em um processo*." Este elemento pode ser iniciado com a entrada de algum dado ou após a realização de uma ação. Ou seja, atividade é conjunto de tarefas que levam a um ou mais artefatos de qualidade controlada pois representa uma evidência de progresso no desenvolvimento, permite o controle da qualidade do resultado e seu esforço é medido por meio das tarefas constituintes.

As atividades de um processo podem ser detalhadas em tarefas individuais. Toda atividade necessita de uma descrição do que precisa ser feito para que ela seja realizada. Isso deve ser feito usando-se um discurso essencial, ou seja, sem entrar em detalhes quanto às tecnologias a serem usadas (WAZLAWICK, 2013). Ou seja, tarefa é ação desempenhada por alguma pessoa visando a realização de uma atividade.

Como resultado da execução de cada atividade, um ou mais artefatos são produzidos. Segundo Pressman (2011), um artefato ou produto de trabalho "*são os programas, os documentos e os dados produzidos em consequência das atividades e tarefas definidas pelo processo*". Ou seja, artefatos são as saídas de uma atividade do processo. Por exemplo, a saída da atividade de projeto arquitetural é o modelo de arquitetura do software.

Essas atividades são executadas por papéis previamente definidos. Segundo Sommerville (2011), "*papéis refletem as responsabilidades das pessoas envolvidas em um processo*". Exemplos de papéis são: gerente de projeto, gerente de configuração, desenvolvedor, etc. Em suma, os papéis são perfis de pessoas ou cargos que realizam uma certa atividade. Na prática, é comum que qualquer atividade tenha um único responsável. Na descrição de um processo, as atividades devem ser atribuídas a perfis ou cargos, e não a pessoas. Apenas quando o processo for usado em um projeto concreto é que deve haver atribuições de atividades a pessoas (WASLAWICK, 2013).

Conforme as atividades forem sendo planejadas, é necessário definir os marcos das principais entregas do projeto, chamados *milestones*. Para Sommerville (2011), "*milestones são estágios-chave no projeto onde o progresso pode ser medido*." Ou seja, é momento específico que marca a finalização de uma atividade importante no decorrer do ciclo de vida de um projeto de software.

5 PROCEDIMENTOS METODOLÓGICOS

Nesta seção será explanada as atividades necessárias para a realização deste projeto de pesquisa.

Inicialmente, foi selecionada a área de pesquisa e definido o problema a ser resolvido, estabelecendo os objetivos e as questões de pesquisa do trabalho. Posteriormente, foi feita uma pesquisa bibliográfica sobre os trabalhos da área de GCS já existentes na literatura. Para a pesquisa destes trabalhos, foi definida a seguinte *string* de busca:

(("configuration management" OR "change management" OR "version control" OR "versioning management" OR "version management" OR "code version" OR "release management") AND (process OR approach OR method OR technique OR tool OR pattern) AND ("small project" OR "student project" OR capstone OR short OR academic OR academia) AND ("software development" OR "application development" OR "system development"))

Essa *string* foi rodada nas bases da ACM e IEEE. Para filtrar os artigos realmente relevantes ao tema, foi feita a leitura do resumo de cada um destes. Para cada trabalho retornado pela *string* foi feita a leitura de seu resumo (*abstract*), e aqueles que passaram por essa filtragem foram catalogados através de um fichamento. Esse fichamento registrou o objetivo de cada trabalho, os pontos positivos e negativos de cada trabalho, o que poderia ser melhorado e quais informações importantes poderiam ser incorporadas no processo.

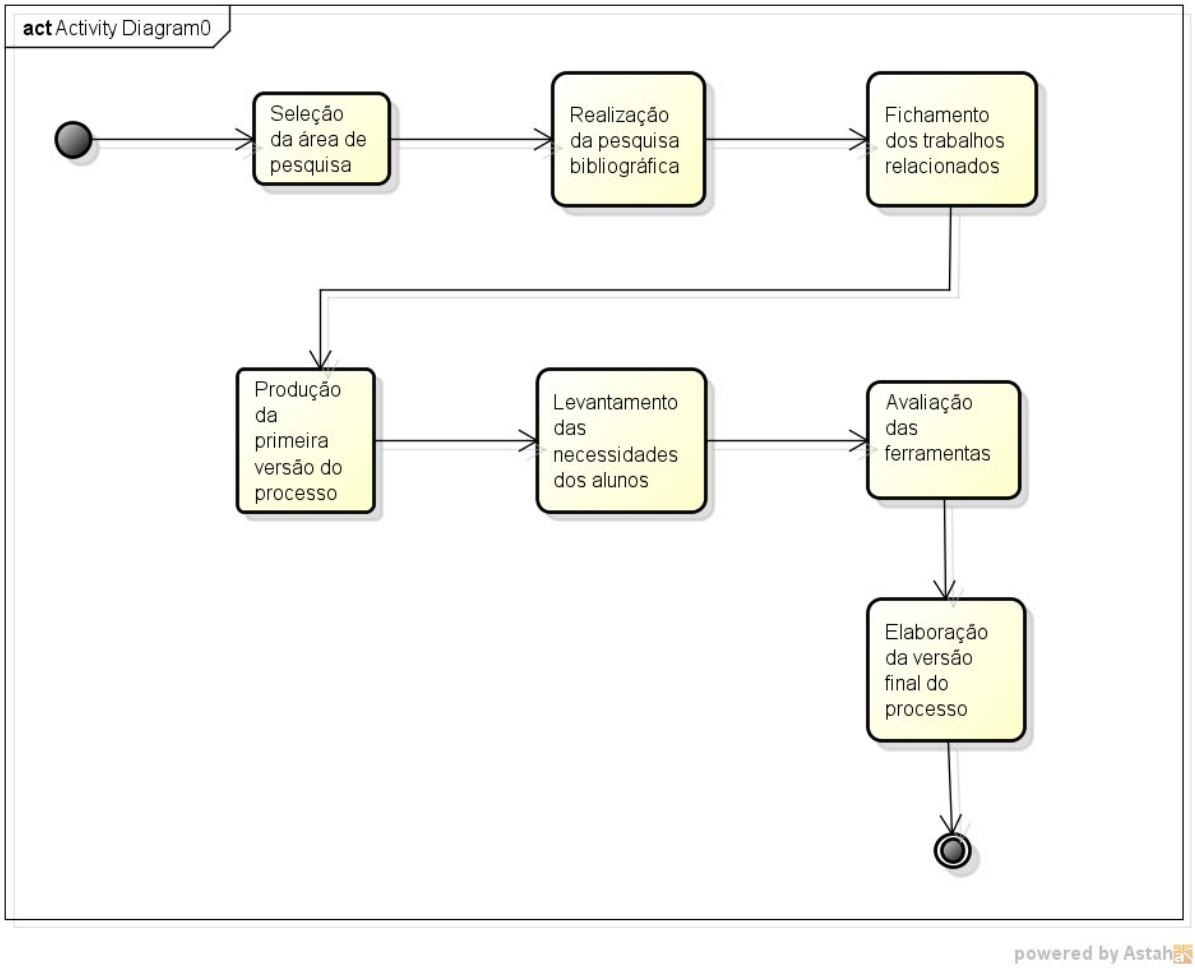
Na base da IEEE, 330 resultados foram retornados, dois quais 18 são relacionados ao tema. Na base da ACM, 136 resultados foram retornados, dois quais 13 são relacionados ao tema. 31 artigos foram selecionados no total.

A partir dos dados retirados do fichamento, foi criado a primeira versão do processo de GCS proposto, juntamente com a descrição do processo em um documento à parte. Esses dados são informações sobre atividades relacionadas à GCS que posteriormente podem ser incorporadas ou adaptadas para à criação do novo processo.

Posteriormente, foi realizado um questionário para levantar as necessidades dos alunos em relação a GCS, identificando as principais dificuldades dos alunos ao lidar com mudanças dentro de um projeto acadêmico. O questionário foi aplicado com alunos dos cursos de Engenharia de Software e Ciência da Computação e Engenharia da Computação da Universidade Federal do Ceará c*ampi* Russas e Pici.

Por fim foi avaliado o uso de duas ferramentas nas atividades da fase de Controle de Mudanças do processo e automatizar tais atividades. Tal avaliação foi conduzida na disciplina Tópicos Avançados em Engenharia de Software do curso de Ciência da Computação Campus do Pici da Universidade Federal do Ceará, a fim de avaliar o uso de duas. Os resultados dessa avaliação foram coletados a partir de um questionário e analisados. Por fim, a versão final do processo foi definida com base nos dados coletados. A Figura 2 demonstra o passo-a-passo das atividades realizadas durante este trabalho.

Figura 2 – Condução das atividades de pesquisa



Fonte: elaborado pelo autor.

6 PROCESSO STONE

Esta seção está organizada da seguinte forma: A subseção 6.1 descreve como o processo STONE foi concebido, com base dos trabalhos e modelos de qualidades de GCS conhecidos na literatura. A subseção 6.2 apresenta o fluxo das atividades do processo, quais ferramentas o processo apoia, quais são os artefatos de entrada e saída, e por fim, quais papéis do processo suporta. Na subseção 6.3 é detalhada cada tarefa que compõe as atividades do processo.

6.1 Concepção do Processo STONE

A norma técnica IEEE 1042 (1988) serviu como base para a definição das fases do processo STONE. Como motivação para a escolha dessa norma, ela fornece orientação às práticas de planejamento de gerência de configuração de software. Este guia discute GCS como um conjunto de disciplinas de gerenciamento dentro do contexto de processo de engenharia de software, em vez de um conjunto de atividades específicas realizadas dentro de uma organização. A GCS é realizada através das atividades de Identificação da Configuração, Gerenciamento de Baselines, Controle e Documentação de Mudanças, Contabilização do Status, Revisões e Auditorias, e Gerenciamento de Releases. Na prática, essas atividades podem ser executadas de forma diferente, dependendo do tipo de software a ser desenvolvido, e podem variar no grau de formalidade de documentação requerida dentro das diferentes fases do ciclo de vida de gerenciamento.

A norma ISO 10007 (1996) serviu como base para a definição da atividade de 'Identificar a Configuração' do processo STONE. Esta norma fornece diretrizes de GC, a fim de prover visibilidade e controle das características funcionais do produto de software. A aplicação dessas diretrizes pode ser adaptada para se adequar a projetos individuais, levando-se em consideração o tamanho, a complexidade e a natureza do trabalho. Na norma, a atividade de Identificação da Configuração se divide nas seguintes tarefas:

- Estrutura de produto e seleção dos itens de configuração: A estrutura do produto deve descrever o relacionamento e a posição dos itens de configuração na decomposição do produto. Convém que os itens de configuração sejam selecionados por meio da aplicação de um processo de decomposição do produto. Para a elaboração do processo STONE, esta tarefa foi adaptada, contendo apenas a seleção dos itens de configuração, sem critérios pré-estabelecidos.
- Documentação de itens de configuração: Todas as características funcionais de um IC, incluindo interfaces, alterações, desvios e concessões devem constar em documentos claramente identificados. Para a elaboração do processo STONE, esta tarefa foi alterada, contendo apenas um documento responsável pela documentação dos ICs, e as informações que devem constar nesse documento são: data de criação, criador do IC, a descrição dos relacionamentos entre os ICs, e a função de cada IC.
- Numeração: convenções de numeração devem ser estabelecidas e aplicadas à identificação de itens de configuração, de suas partes e conjuntos, documentos, interfaces, alterações, desvios e concessões. Para a elaboração do processo STONE, esta tarefa foi alterada, contendo apenas a identificação única de cada IC.

O SWEBOK (2001) serviu como base para a definição da atividade de Elaboração do Plano de Gerência de Configuração, pois este guia apresenta uma visão de GCS que se relaciona intimamente com a área de Garantia de Qualidade de Software (SQA). Conforme definido na área de conhecimento de Qualidade de Software, os processos fornecem garantia de que os produtos de software e processos no ciclo de vida do projeto estejam de acordo os requisitos especificados pelo planejamento, realizando um conjunto de atividades para fornecer uma confiança adequada de que a qualidade está sendo implantada no software. Atividades de GCS ajudam na realização desses objetivos de SQA. No SWEBOK, a fase de Gerenciamento do processo de GCS contém a atividade de Planejamento da GCS, que corresponde à atividade 'Elaborar Plano de Gerência de Configuração' do processo STONE. As seguintes tarefas foram adaptadas para o processo STONE:

- Responsabilidades e Organização da GCS: Para evitar confusão sobre quem realizará determinadas atividades ou tarefas da GCS, as funções organizacionais envolvidas no processo de GCS precisam ser claramente identificadas. Responsabilidades específicas para determinadas atividades ou tarefas de GCS também precisam ser atribuídas a entidades organizacionais, seja por título ou por elemento organizacional.
- Recursos e Cronograma da GCS: O planejamento para a GCS identifica a equipe e as ferramentas envolvidas na execução de atividades e tarefas de GCS. Ele aborda questões de planejamento, estabelecendo sequências necessárias de tarefas de SCM e identificando seus relacionamentos com o cronograma e os *milestones* do projeto. Para a elaboração da tarefa de 'Estabelecer *milestones*' do processo STONE, apenas a identificação dos *milestones* foi considerada.
- Seleção de Ferramentas e Implementação: Quanto a qualquer área de ES, a seleção e implementação de ferramentas de GCS devem ser cuidadosamente planejadas. A GCS geralmente requer um conjunto de ferramentas, ao invés de uma única ferramenta. O tamanho da organização e o tipo de projetos envolvidos também podem afetar a seleção de ferramentas. Para a tarefa de 'Definir ferramentas' do processo STONE, deverão ser definidas ferramentas para comunicação entre os membros da equipe, versionamento dos ICs, armazenamento dos ICs, realização de testes no código-fonte, e por fim, a escolha de checklists de inspeção nos documentos e/ou diagramas do projeto.
- Controle de Interfaces: Quando um IC faz interface com outro item de software ou hardware, uma alteração em um dos itens pode afetar o outro. O planejamento do processo de GCS considera como os itens de interface serão identificados e como as alterações nos itens serão gerenciadas e comunicadas. Para a tarefa 'Fazer rastreabilidade entre itens de configuração' do processo STONE considera apenas a rastreabilidade entre itens de software.
- Elaboração do Plano de Gerência de Configuração: Os resultados da fase de Planejamento da GCS para um determinado projeto são registrados em um Plano de Gerência de Configuração (PGC), um "documento ativo" que serve como referência para o processo de GCS. Ele é mantido (ou seja, atualizado e aprovado) conforme necessário durante o ciclo de vida do software. Na implementação do PGC, normalmente é necessário desenvolver uma série de procedimentos subordinados e

mais detalhados, definindo como os requisitos específicos serão executados durante as atividades diárias. Para a tarefa de 'Documentar Plano de Gerência de Configuração' do processo STONE, esta tarefa será simplesmente o agrupamento das saídas das tarefas anteriores.

A tarefa Controle de Vendas e Subcontratação não foi utilizada, pois em projetos acadêmicos não é necessário se preocupar com a contratação de mão de obra especializada ou gestão das finanças do projeto.

O resultado esperado 'GCO5 - Modificações em itens de configuração são controladas' da área de conhecimento de GCS do Guia de Implementação do MPS-BR (2013) serviu como base para a definição das atividades da fase de Controle de Mudanças do processo STONE. Este resultado esperado especifica que a partir do momento que os ICs passam a fazer parte de uma *baseline*, toda e qualquer modificação deve passar por um processo formal de controle de modificações. Para isso, é necessário que sejam desempenhadas as seguintes atividades:

- Documentação da necessidade de modificação;
- Análise de impacto da modificação;
- Avaliação da modificação (aprovação ou reprovação);

Após a implementação de uma modificação, ao menos os seguintes passos são usualmente registrados (ambas atividades não foram adaptadas para o processo STONE):

- Verificação da implementação;
- Atualização da *baseline* com a modificação;

Para este trabalho, as atividades da fase de “Controle de Mudanças” não utilizam o conceito de *baseline*, a fim de evitar a introdução de conceitos mais complexos e que não seriam de muito uso em projetos mais informais, como os projetos acadêmicos. Apesar das atividades da fase de “Controle de Mudanças” do processo STONE basearem-se neste resultado esperado, as tarefas para a realização de tais atividades foram elaboradas pelo autor.

As atividades das fases de “Controle de Versão” e “Gerenciamento de Releases” do processo STONE basearam-se em atividades propostas pelo modelo de processo Rugby, proposto por Krusche *et al.* (2014). O *workflow* deste modelo de processo de entrega contínua funciona da seguinte forma:

- O desenvolvedor envia um *commit* ao servidor de controle de versão, ou seja, o desenvolvedor está versionando o IC: para o processo STONE, o servidor de controle de versão é a ferramenta que auxilia no versionamento do IC. Dependendo do tipo do IC, o uso da ferramenta é optativo.
- O IC é testado por meio do servidor de integração contínua;
- O gerente de releases, por meio do servidor de integração contínua, informa ao desenvolvedor o status do IC: essa atividade não foi utilizada no processo STONE.
- O servidor de integração contínua faz o upload do IC no servidor de entrega: Nesta atividade, o servidor de entrega é substituído por um repositório.
- Se o servidor de entrega detectar defeitos no release, ele armazena o relatório de defeitos e notifica o desenvolvedor;

- Se o servidor de entrega não detectar defeitos no release, ele notifica o usuário e faz o download do release no dispositivo;
- O usuário dá o feedback do release, e o servidor de entrega o armazena.

Todo o fluxo de atividades descrito acima foi aproveitado para o processo STONE, no entanto, ambos servidores de integração contínua e de entrega contínua não são utilizados no processo STONE, a fim de simplificar o máximo possível o desenvolvimento das tarefas do processo, uma vez que nem todos os alunos têm conhecimento suficiente para utilizarem tais tecnologias em seus projetos acadêmicos, que são de pequeno porte.

6.2 Preâmbulo do processo STONE

O processo proposto foi dividido em 5 fases: Identificação, Planejamento, Controle de Mudanças, Controle de Versão e Gerenciamento de *Releases*. Essa divisão foi realizada com o intuito de adaptar cada fase às etapas do ciclo de vida de desenvolvimento de um projeto. O processo possui 10 atividades. Cada atividade possui tarefas a serem realizadas, onde cada tarefa tem como resultado a criação de um novo artefato ou a atualização de um artefato produzido em alguma atividade anterior ou numa tarefa da própria atividade.

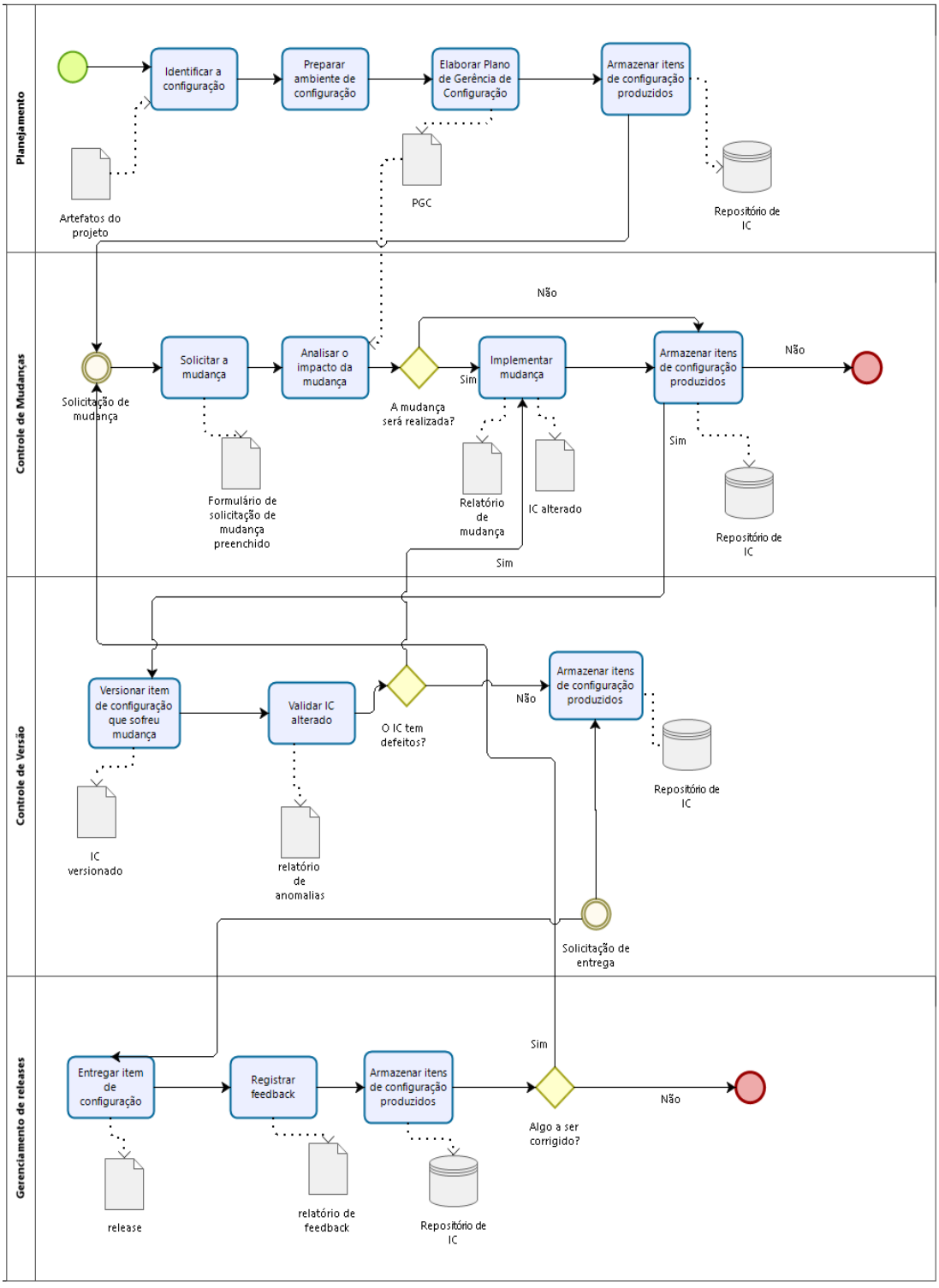
As fases do processo STONE são apresentadas na Tabela 2 descrevendo o que é feito em cada fase.

Tabela 2 – Fases do Processo STONE

Fase	Descrição
Identificação	Nesta fase serão identificados os artefatos do projeto que devem ficar sob o controle de mudanças (item de configuração).
Planejamento	Nesta fase será descrita as principais tarefas para auxílio na execução do processo.
Controle de Mudanças	Nesta fase os itens de configuração terão suas mudanças controladas.
Controle de Versão	Nesta fase os itens de configuração que passaram por mudanças serão versionados e testados adequadamente.
Gerenciamento de releases	Nesta fase as entregas dos itens de configuração serão gerenciadas.

A Figura 3 exibe a representação visual do processo STONE, com cada atividade dividida entre as fases supracitadas.

Figura 3 - Processo STONE



Fonte: elaborado pelo autor.

A fase inicial do processo é a Identificação. Nessa fase os artefatos que devem ficar sob o controle de mudanças são identificados e são definidas tarefas importantes para elaboração do Plano de Gerência de Configuração, o principal artefato do processo. As seguintes atividades fazem parte dessa fase:

1. Identificar a configuração: os itens de configuração (IC) serão selecionados e documentados.
2. Preparar ambiente de configuração: é definido o ambiente de configuração necessário para a implementação do processo de acordo com os ICs identificados na atividade anterior.
3. Elaborar Plano de Gerência de Configuração: será feita a descrição dos papéis dos membros do projeto, a escolha das ferramentas para apoiar o processo, a rastreabilidade entre os ICs e a definição dos *milestones* do projeto.

Na fase de Controle de Mudanças, os ICs que precisarem de mudanças passarão por uma sequência de atividades a fim de controlar as mudanças e manter a consistência entre os demais artefatos do projeto. Ao contrário das outras fases do projeto, esta poderá ser iniciada a qualquer momento durante o projeto, contanto que as atividades das fases de Identificação e Planejamento já tenham sido executadas. As seguintes atividades fazem parte dessa fase:

4. Solicitar a mudança: cada IC que precisar passar por mudanças, deverá ser feito o registro da solicitação da mudança.
5. Analisar impacto da mudança: Após o registro da solicitação da mudança ter sido feito e os demais membros da equipe terem sido notificados sobre esta solicitação, é necessário analisar o impacto da mudança nos demais ICs que serão afetados por esta.
6. Implementar mudança: Caso seja decidido implementar a mudança, o IC que será alterado será encaminhado a um membro da equipe que tenha o conhecimento necessário para implementá-la.

Na fase de Controle de Versão, os ICs que passarem pelo controle de mudanças precisam ter suas versões atualizadas e suas funcionalidades (seu conteúdo) inspecionadas para verificar se a mudança no IC foi implementada corretamente. As seguintes atividades fazem parte dessa fase:

7. Versionar item de configuração: Depois de alterado, o IC precisa ter sua versão atualizada.
8. Validar item de configuração: Depois de versionado, será verificado se o IC em questão foi alterado corretamente.

Na fase de Gerenciamento de releases, os ICs que precisarem ser entregues ao professor, é necessário executar as seguintes atividades:

9. Entregam item de configuração: Se o IC precisar ser entregue ao professor, este será encaminhado pelo canal de comunicação adequado.

Registrar feedback: Depois do IC ter sido entregue, o feedback do professor deverá ser

registrado. Se necessitar de uma nova alteração, este IC irá ser encaminhado para o Controle de Mudanças.

Ao final de cada fase, os itens de configuração serão armazenados no repositório definido durante a atividade 'Preparar ambiente de configuração'. Quando todos os artefatos do projeto (IC's) forem entregues, o processo é finalizado.

Muitos dos artefatos produzidos durante as atividades do processo são utilizados pelo SWEBOK (2001), a norma ISO 10007 (1998) e outros trabalhos publicados na área. Sempre que um artefato for criado ou modificado, ele deverá ser adicionado ao repositório da empresa, permitindo que ele seja utilizado futuramente como auxílio para atividades futuras. A Tabela 3 lista os artefatos utilizados no processo STONE.

Tabela 3 – Artefatos do Processo STONE

Artefato	Características
Plano de Gerência de Configuração	Plano central que define como o processo de gerência de configuração será executado dentro do projeto. Um *template* deste artefato está descrito no Apêndice A.
Formulário de Solicitação de Mudança	Documento que contém a descrição de uma nova solicitação de mudança a um ou mais itens de configuração. Um *template* deste artefato está descrito no Apêndice B.
Relatório de Mudança	Descrição das mudanças que incidiram em um determinado IC. Um *template* deste artefato está descrito no Apêndice C.
Relatório de Anomalias	Detalhamento da execução da atividade de inspeção/teste realizado em um determinado IC. Um *template* deste artefato está descrito no Apêndice D.
Relatório do feedback	Descrição do feedback dado pelo professor sobre determinado artefato. Um *template* deste artefato está descrito no Apêndice E.

Os papéis são funções importantes para o processo STONE, cada ator do processo possui um papel funcional e uma responsabilidade associado a ele. Tais papéis são descritos na Tabela 4.

Tabela 4 - Papéis do Processo STONE

Atores		
	Papel Funcional	**Responsabilidades**
Aluno*	Líder de projeto	Responsável por alterar documentos de gerência do projeto. Ex.: Plano de Gerência de Configuração
	Analista de requisitos	Responsável por alterar documentos das etapas de elicitação e análise de requisitos. Ex.: Documento de Requisitos
	Projetista	Responsável por alterar os diagramas e documentos da etapa de Design. Ex.: Diagrama de classes e Modelo Arquitetural
	Programador	Responsável por alterar o código fonte.
	Testador	Responsável por alterar documentos e códigos etapa de Testes. Ex.: Casos de Teste e Testes Unitários.
	Gerente de *Releases*	Responsável por entregar as *releases* do projeto ao cliente e registrar seu feedback.
Professor	Cliente	Pessoa que requisita um produto específico, que será resultado do desenvolvimento de um projeto. Responsável por dar o feedback dos releases do projeto.
Ambos	Solicitante	Realizar solicitação de mudança sob algum item de configuração.

*Dependendo de quais etapas do processo de ciclo de vida estejam sendo aplicadas no projeto, os alunos podem assumir diferentes tipos de papéis no processo STONE. Se o projeto for composto de apenas um aluno, ele assumirá uma quantidade restrita de papéis.

As ferramentas necessárias para auxiliar a execução das atividades por parte dos papéis envolvidos no processo STONE são descritas na Tabela 5. Para cada ferramenta é descrita é citado seu objetivo e um exemplo.

Tabela 5 – Ferramentas do Processo STONE

Ferramenta	Descrição
Editor de documento	Ferramenta utilizada para a documentação do Plano de Gerência de Configuração. Ex.: Word ou Documentos Google
Ferramenta para comunicação entre os membros da equipe	Ferramenta utilizada para informar a todos os envolvidos no projeto que uma nova solicitação de mudança foi feita. Ex.: WhatsApp, Telegram
Ferramenta para controle de versão	Ferramenta utilizada no versionamento dos itens de configuração. Ex.: Git
Repositório de itens de configuração alterados	Ferramenta que armazena os ICs que sofreram mudanças. Cada membro do projeto deve ter acesso à edição da pasta onde serão mantidos esses itens. Ex.: GitHub, Google Drive
Google Drive	Repositório para armazenamento dos demais itens de configuração, àqueles que não sofreram mudanças. Cada membro do projeto deve ter acesso à edição da pasta onde serão mantidos esses itens.
Ferramenta de testes/inspeção	(Optativa) Uma ferramenta é definida para auxiliar na atividade de teste/inspeção dos ICs que sofreram mudanças. Ex.: JUnit para execução dos testes unitários ou SonarQube para execução da análise estática do código.
Canal de comunicação com professor	São estabelecidas ferramentas por onde os releases do projeto serão entregues ao professor. Ex.: SIGAA, correio eletrônico.

6.3 Atividades

Nesta seção cada atividade do processo STONE será descrita detalhadamente. Para cada tarefa é informado seu nome, sua descrição, seu ator, pré-condições, participantes, entradas, saídas, ferramenta, passo-a-passo e pós-condição.

6.3.1 Planejamento

Esta fase é composta pelas atividades 'Identificar a Configuração', 'Preparar Ambiente de Configuração' e 'Elaborar Plano de Gerência de Configuração'.

6.3.1.1. Identificar a Configuração

Nesta atividade, é feita a documentação dos artefatos que devem ficar sob controle de mudanças, ou seja, a documentação dos itens de configuração.

Nome da Tarefa	**Estabelecer convenção de numeração**
Descrição	Será definida uma regra de como os itens de configuração devem ser identificados e versionados.
Responsável	Gerente de Configuração
Pré-Condições	Algum artefato do projeto já ter sido produzido
Participantes	Nenhum
Entradas	Artefatos do projeto
Saídas	Convenção de numeração
Recursos/Ferramentas	- Editor de texto
Passo a Passo	1. Informar sigla para o IC 2. Informar número para o IC 3. A cada IC que for sendo adicionado, o número será incrementado
Pós-Condições	Nenhuma

Nome da Tarefa	**Documentar itens de configuração**
Descrição	Os ICs serão documentados.
Responsável	Gerente de Configuração

Pré-Condições	Algum artefato do projeto já ter sido produzido
Participantes	Nenhum
Entradas	Artefatos do projeto
Saídas	Registro dos itens de configuração no PGC
Recursos/Ferramentas	- Editor de texto
Passo a Passo	4. Identificar unicamente cada IC. 5. Informar datas de criação de cada IC. 6. Informar criador de cada IC. 7. Descrever a função de cada IC. 8. Especificar localização de cada IC. 9. Informar status de cada IC.
Pós-Condições	Nenhuma

6.3.1.2. Preparar ambiente de configuração

Nesta atividade, é definido o ambiente de configuração no qual o projeto irá ser executado.

Nome da Tarefa	**Definir ferramentas**
Descrição	As ferramentas que serão utilizadas para apoiar a execução do processo ao longo das atividades serão definidas.
Responsável	Gerente de Configuração
Pré-Condições	Os itens de configuração terem sido identificados
Participantes	Demais membros da equipe do projeto
Entradas	Registro dos ICs
Saídas	Descrição das ferramentas
Recursos/Ferramentas	- Editor de texto
Passo a Passo	1. Definir repositório que será utilizado para armazena os itens de configuração. 2. Definir ferramenta(s) para execução das atividades de controle de mudança. 3. Definir canal de comunicação entre os

	membros do projeto. 4. Definir ferramenta de testes. 5. Definir canal de comunicação com o professor.
Pós-Condições	Nenhuma

6.3.1.3 Elaborar Plano de Gerência de Configuração

Nesta atividade, é desenvolvido um plano que estabelece as principais tarefas para execução do processo STONE.

Nome da Tarefa	**Fazer descrição dos papéis dos membros do projeto**
Descrição	Quando o processo estiver sendo aplicado por mais de um aluno, é necessário fazer uma descrição das responsabilidades de cada membro do projeto.
Responsável	Gerente de Configuração
Pré-Condições	Ter identificado os itens de configuração
Participantes	Nenhum
Entradas	Registro de ICs
Saídas	- tabela de responsabilidade
Recursos/Ferramentas	- Editor de documento
Passo a Passo	1. Informar o nome do papel funcional e o membro da equipe correspondente 2. Informar quais atividades cada um é responsável
Pós-Condições	Nenhuma

Nome da Tarefa	**Fazer rastreabilidade entre itens de configuração**
Descrição	Os ICs quem têm dependências entre si terão estes relacionamentos rastreados.

Responsável	Gerente de Configuração
Pré-Condições	Ter identificado os itens de configuração
Participantes	Nenhum
Entradas	Registro de ICs
Saídas	Matriz de rastreabilidade
Recursos/Ferramentas	- Editor de documento
Passo a Passo	1. Relacionar, em uma tabela, cada IC dependente com cada IC de qual ele depende.
Pós-Condições	Nenhuma

Nome da Tarefa	**Estabelecer *milestones***
Descrição	As entregas e principais atividades (marcos) que serão executadas são especificadas, bem como a data de início e fim para cada marco.
Responsável	Gerente de Configuração
Pré-Condições	Ter identificado os itens de configuração
Participantes	Nenhum
Entradas	Registro de ICs
Saídas	Milestones
Recursos/Ferramentas	- Editor de documento
Passo a Passo	1. Definir datas de início e entrega de cada IC. 2. Descrever tarefas em cada marco.
Pós-Condições	Nenhuma

Nome da Tarefa	**Estabelecer estrutura de diretórios**
Descrição	A estrutura dos diretórios/repositórios será definida a fim de facilitar a localização dos ICs.
Responsável	Gerente de Configuração

Pré-Condições	Ter identificado os itens de configuração
Participantes	Nenhum
Entradas	Registro de ICs
Saídas	Estrutura de diretórios
Recursos/Ferramentas	- Editor de documento
Passo a Passo	3. Descrever um diagrama com a hierarquia dos diretórios utilizados no repositório.
Pós-Condições	Nenhuma

Nome da Tarefa	**Definir glossário**
Descrição	Os conceitos utilizados para definição do PGC são especificados.
Responsável	Gerente de Configuração
Pré-Condições	Ter identificado os itens de configuração
Participantes	Nenhum
Entradas	Registro de ICs
Saídas	Glossário
Recursos/Ferramentas	- Editor de documento
Passo a Passo	1. Informar significado de cada conceito específico do projeto.
Pós-Condições	Nenhuma

Nome da Tarefa	**Definir política de aplicação do processo**
Descrição	O escopo e restrições ao qual o processo se aplica ao projeto em questão é especificado.
Responsável	Gerente de Configuração
Pré-Condições	-
Participantes	Nenhum

Entradas	Registro de ICs
Saídas	Política de aplicação do processo
Recursos/Ferramentas	- Editor de documento
Passo a Passo	1.Informar quais atividades serão aplicadas no projeto. 2.Informar restrições do projeto para aplicação do processo.
Pós-Condições	Nenhuma

6.3.2 Controle de Mudanças

Esta fase é composta pelas atividades 'Solicitar a Mudança', 'Analisar o Impacto da Mudança' e 'Implementar a Mudança'.

6.3.2.1 Solicitar a mudança

Nesta atividade é realizado o controle das mudanças que surgem ao longo do projeto.

Nome da Tarefa	**Fazer solicitação de mudança**
Descrição	É feita uma solicitação de mudança a item de configuração.
Responsável	Solicitante
Pré-Condições	Ter executado as atividades da fase de Planejamento do processo STONE.
Participantes	Nenhum
Entradas	Item de configuração
Saídas	- Registro de Solicitação de Mudança
Recursos/Ferramentas	- Editor de texto - Ferramenta para comunicação entre os membros da equipe
Passo a Passo	1. Informar identificador do item de configuração 2. Informar versão do item de configuração 3. Informar identificador da solicitação da

	mudança 4. Informar nome do solicitante 5. Informar data da solicitação 6. Indicar urgência da mudança 7. Descrever a necessidade da mudança 8. Descrever a mudança a ser implementada. 9. Justificar a mudança 10. Informar status de aprovação 11.Informar via ferramenta para comunicação entre os membros da equipe que uma nova solicitação de mudança foi feita.
Pós-Condições	Nenhuma

6.3.2.2 Analisar o impacto da mudança

Nesta atividade é feita a análise do impacto da mudança nos demais itens de configuração do projeto.

Nome da Tarefa	**Analisar impacto**
Descrição	É analisado o impacto da mudança com base na matriz de rastreabilidade.
Responsável	Gerente de Configuração
Pré-Condições	Ter elaborado o PGC e ter registrado a solicitação de mudança.
Participantes	Demais membros do projeto
Entradas	- Formulário de Solicitação de Mudança - Plano de Gerência de Configuração
Saídas	Mensagem aos stakeholders se a solicitação de mudança foi aprovada ou não
Recursos/Ferramentas	Ferramenta para comunicação entre os membros da equipe
Passo a Passo	1. Identificar artefato que sofrerá a mudança. 2. Analisar quais ICs ele afeta.

	3. Decidir junto com resto da equipe se a mudança será implementada ou não.
Pós-Condições	Nenhuma

6.3.2.3 Implementar mudança

Nesta atividade a solicitação de mudança feita inicialmente será implementada por um membro da equipe responsável.

Nome da Tarefa	**Implementar mudança**
Descrição	A mudança é documentada e executada.
Responsável	Aluno
Pré-Condições	Ter analisado o impacto da mudança. Ter sido aprovada a implementação da mudança.
Participantes	Nenhum
Entradas	- Item de configuração - Formulário de solicitação de mudança.
Saídas	- Relatório de mudança. - Item de configuração alterado.
Recursos/Ferramentas	- Editor de documento - Ferramenta para comunicação entre os membros do projeto.
Passo a Passo	1. Encaminhar IC ao responsável pela implementação da mudança. 2. Alterar IC que necessita de mudanças. 3. Alterar demais ICs que foram afetados pela mudança. 4. Atualizar matriz de rastreabilidade referente ao IC que foi alterado. 5. Documentar relatório de mudança.
Pós-Condições	Informar aos demais membros do projeto que a mudança foi implementada.

6.3.3 Controle de Versão

Esta fase é composta pelas atividades ‘Versionar item de configuração que sofreu mudança’ e ‘Validar item de configuração alterado’.

6.3.3.1 Versionar item de configuração que sofreu mudança

O IC que sofreu mudança terá seu histórico de versões atualizado.

Nome da Tarefa	**Versionar item de configuração**
Descrição	Versionar IC de acordo com os procedimentos definidos no PGC.
Responsável	Aluno
Pré-Condições	Ter implementado a mudança em um IC.
Participantes	Nenhum
Entradas	- IC alterado - PGC - Relatório de mudanças
Saídas	- Artefato versionado
Recursos/Ferramentas	- Ferramenta para versionamento do IC.
Passo a Passo	1. Se não for definida ferramenta para versionamento do IC, atualizar histórico de versões do IC manualmente. 2. Se for definida ferramenta para versionamento do IC, realizar esta tarefa por meio da ferramenta em questão.
Pós-Condições	Nenhuma

6.3.3.2 Validar item de configuração alterado

Nesta atividade é realizado o teste ou inspeção do item de configuração que foi versionado.

Nome da Tarefa	**Testar/inspecionar item de configuração**
Descrição	Será verificado de a mudança no item de configuração em questão foi implementada

	corretamente.
Responsável	Aluno
Pré-Condições	Ter versionado o IC.
Participantes	Nenhum
Entradas	- IC versionado
Saídas	- Relatório de anomalias
Recursos/Ferramentas	- Ferramenta de testes
Passo a Passo	1. A inspeção e/ou teste do IC será executada por meio da técnica e/ou ferramenta definida no PGC. 2. A inspeção e/ou teste executado será documentado através do relatório de anomalias. 3. Se o IC não tiver defeitos detectados, este será armazenado até que uma solicitação do professor requisite a entrega do IC. 4. Se o IC tiver defeitos, ele será encaminhado para a fase de controle de mudanças.
Pós-Condições	Nenhuma

6.3.4. Gerenciamento de releases

Esta fase é composta pelas atividades ‘Entregar item de configuração’ e ‘registrar feedback’.

6.3.4.1 Entregar item de configuração

Caso o item de configuração testado deva ser imediatamente entregue ao professor, este deve ser encaminhado pelo canal de comunicação adequado.

Nome da Tarefa	**Enviar IC ao professor**
Descrição	O IC é enviado ao professor
Responsável	Gerente de releases
Pré-Condições	-

Participantes	-
Entradas	- Milestones - IC testado
Saídas	- Release
Recursos/Ferramentas	- Canal de comunicação com o professor
Passo a Passo	1. Conforme o professor apontar, o release deve ser enviado via SIGAA ou correio eletrônico. 2. Se não for necessária uma ferramenta para enviar do release, este será apresentado via notebook ou será feita sua impressão.
Pós-Condições	Nenhuma

6.3.4.2 Registrar feedback

Nessa atividade o feedback do professor é documentado.

Nome da Tarefa	**Registrar feedback**
Descrição	Após o envio do release, o professor fornece um feedback a respeito deste.
Responsável	Gerente de releases
Pré-Condições	Ter enviado a release ao cliente.
Participantes	Professor
Entradas	-
Saídas	- Relatório de feedback
Recursos/Ferramentas	- Editor de documento - Google Drive
Passo a Passo	1. Se o feedback for alguma alteração no release, este será encaminhado à fase de Controle de Mudanças. 2. Se o feedback não se tratar de alterações no release, este feedback será documentado da seguinte forma:

	3. Informar data de envio do feedback. 4. Informar descrição do feedback. 5. Armazenar feedback no Google Drive.
Pós-Condições	Caso todos os ICs do projeto tenham recebido feedback, o processo é finalizado.

6.3.4.3 Armazenar itens de configuração produzidos

Ao final de cada fase do processo STONE, esta atividade é executada seguindo a seguinte tarefa.

Nome da Tarefa	**Armazenar itens de configuração**
Descrição	Esta tarefa tem como objetivo armazenar cada artefato produzido na fase em questão.
Responsável	Aluno*
Pré-Condições	Ter produzido algum IC.
Participantes	-
Entradas	-
Saídas	- IC armazenado
Recursos/Ferramentas	- Repositório
Passo a Passo	1. O PGC é consultado. 2. É identificado o repositório escolhido para armazenar o tipo do IC em questão. 3. O IC é armazenado no repositório apontado.
Pós-Condições	-

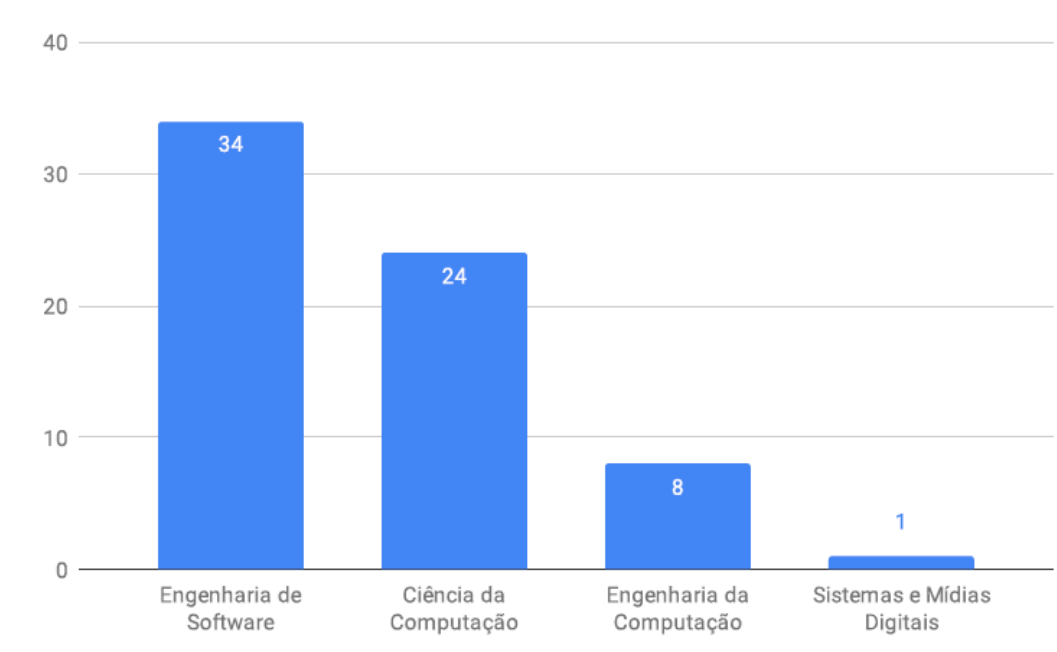
*Qualquer papel que o aluno possa vir a assumir

7 PESQUISA DO USO DA GERÊNCIA DE CONFIGURAÇÃO EM ÂMBITO ACADÊMICO

Após a elaboração da primeira versão do processo STONE, um questionário[1] foi aplicado em turmas do curso de Engenharia de Software, Ciência da Computação e Engenharia da Computação da Universidade Federal do Ceará dos *campi* Russas e Pici. Esse questionário teve o objetivo de identificar como a GCS era aplicada pelos alunos em seus projetos, obtendo no total 67 respostas.

Primeiramente, conforme ilustrado na Figura 4 foi constatado o perfil dos alunos que responderam ao questionário. 34 alunos são do curso de Engenharia de Software (ES), 24 são de Ciência da Computação (CC), 8 são se Engenharia da Computação (EngComp) e 1 é de Sistemas e Mídias Digitais (SMD).

Figura 4 – Curso de graduação dos estudantes



Fonte: elaborado pelo autor.

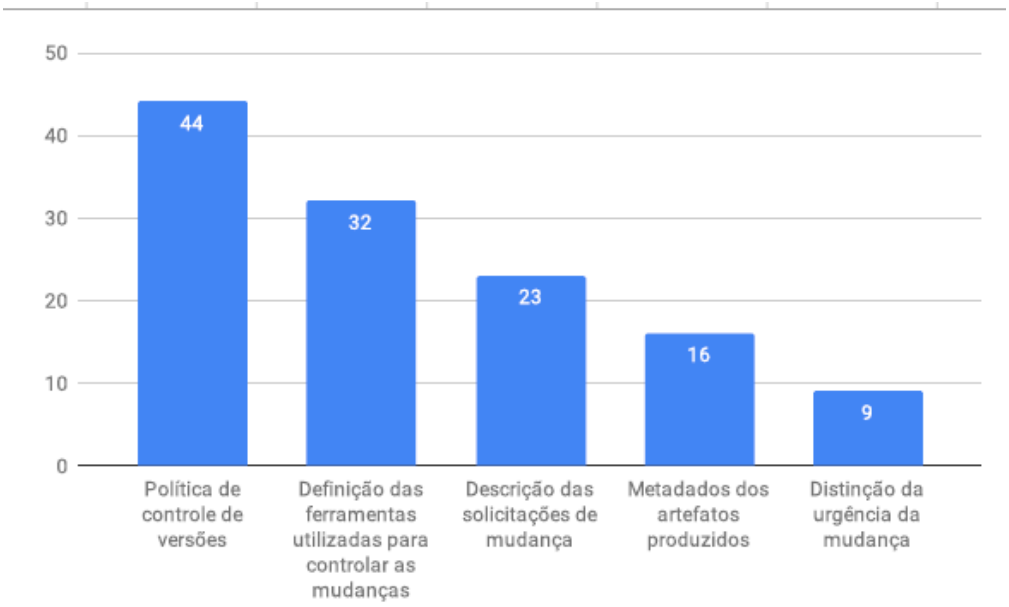
A respeito do conhecimento que os alunos possuem sobre GCS, 66 alunos afirmam que já cursaram uma ou mais disciplinas que introduziam seus conceitos. As disciplinas com maior número de ocorrências foram Processos de Software, Qualidade de Software, Gerência de Projetos e Requisitos de Software. Tais disciplinas são ofertadas a partir do 5º semestre em diante, no entanto, 1 aluno afirmou que nunca cursou cadeiras que introduzissem conceitos de GCS, mesmo estando no 6º semestre.

[1]https://forms.gle/i6HFxb2FBbHVwBiF7

Em relação aos produtos de trabalhos que frequentemente eram entregues durante o projeto, 52 alunos afirmaram que produziam o documento de requisitos, 42 o cronograma de atividades do projeto, 41 a descrição das responsabilidades de cada membro da equipe, 34 o código-fonte, 21 o plano arquitetural e 4 o plano de teste. Outros citaram também o termo de abertura do projeto, plano de projeto e relatório de viabilidade.

A respeito da fase de controle de mudanças, a Figura 5 mostra os artefatos apontados pelos alunos que poderiam ajudar nessa fase. Estes foram a política de controle de versões (44 alunos), a definição das ferramentas para controle de mudanças (32), a descrição das solicitações de mudança (23), os metadados dos artefatos produzidos (16) e a distinção da urgência da mudança (9).

Figura 5 – Informações necessárias para o controle de mudanças

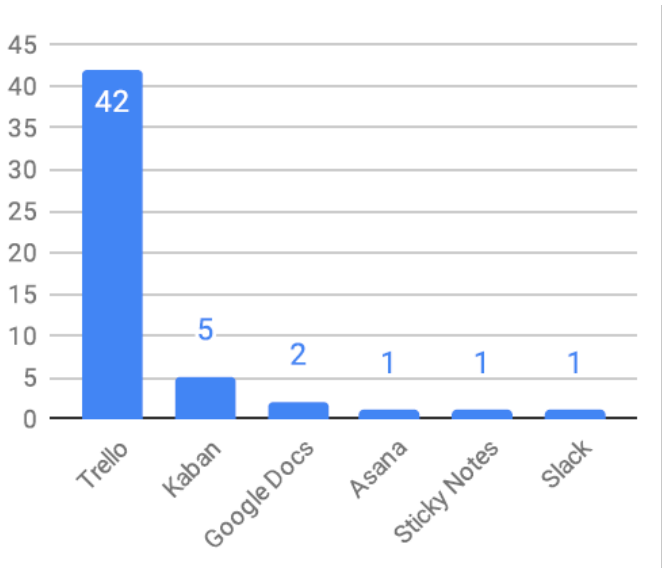


Fonte: elaborado pelo autor.

Uma das questões constatadas foi que a validação dos artefatos produzidos ao longo do projeto geralmente é feita pelos próprios membros da equipe (50 respondentes), no entanto, 17 afirmam que a validação é feita somente pelo professor.

Em relação ao planejamento das atividades do projeto, 53 alunos afirmam que fazem um planejamento prévio, enquanto 14 alunos não realizam. Como demonstra a Figura 6, dos 53 alunos que fazem o planejamento das atividades, 42 alunos afirmam que utilizam o Trello, seguido pelo Kanban (5), Google Docs (2), Asana (1), Sticky Notes (1) e Slack (1).

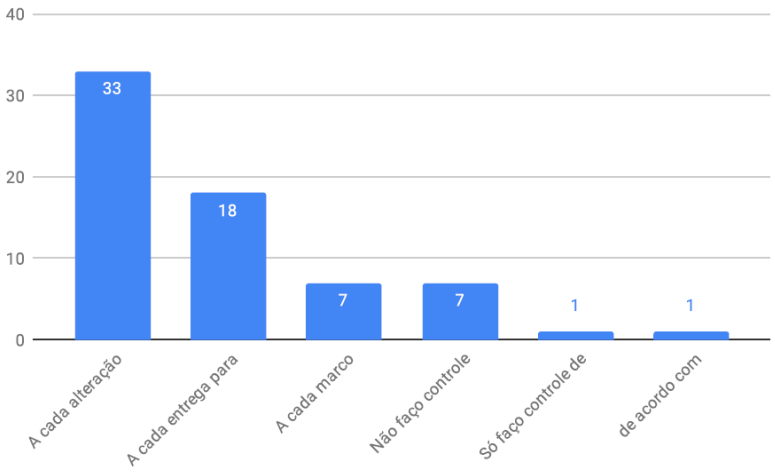
Figura 6 – Ferramentas para planejamento de atividadades



Fonte: elaborado pelo autor.

Em relação à frequência com que os alunos realizam atividades de controle de versão em seus projetos, a Figura 7 demonstra que 33 alunos afirmam que realizam o versionamento dos artefatos a cada alteração, 18 a cada entrega ao professor, 7 a cada marco do projeto, 7 não fazem controle de versão, 1 só faz controle de versão em código fonte e 1 faz de acordo com o desenvolvimento do projeto.

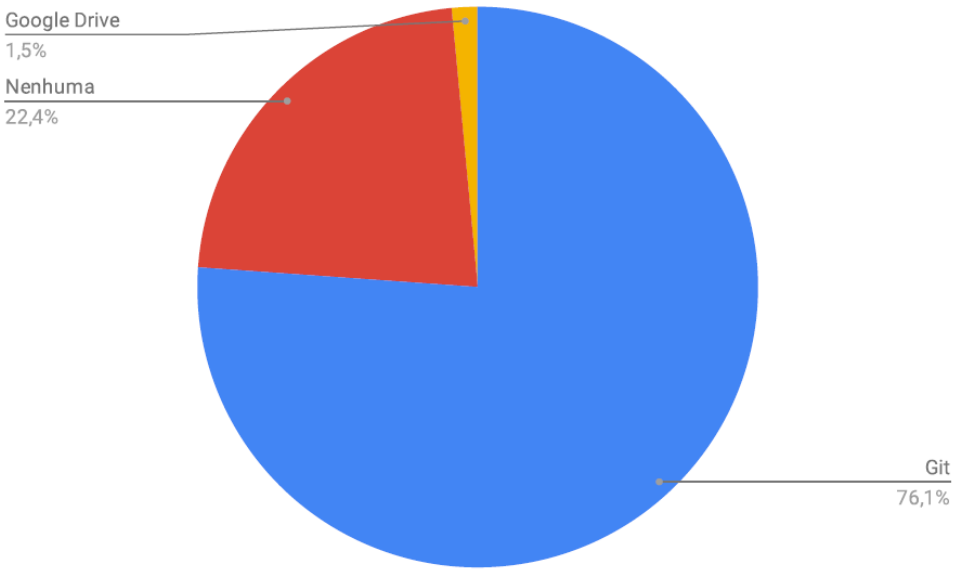
Figura 7 – Frequência de execução do controle de versão



Fonte: elaborado pelo autor.

Conforme a Figura 8 mostra, dos 60 alunos que afirmaram que fazem controle de versão, 51 utilizam o Git, 1 utiliza o Google Drive, e 8 não utilizam ferramentas para apoiar o processo de controle de versão.

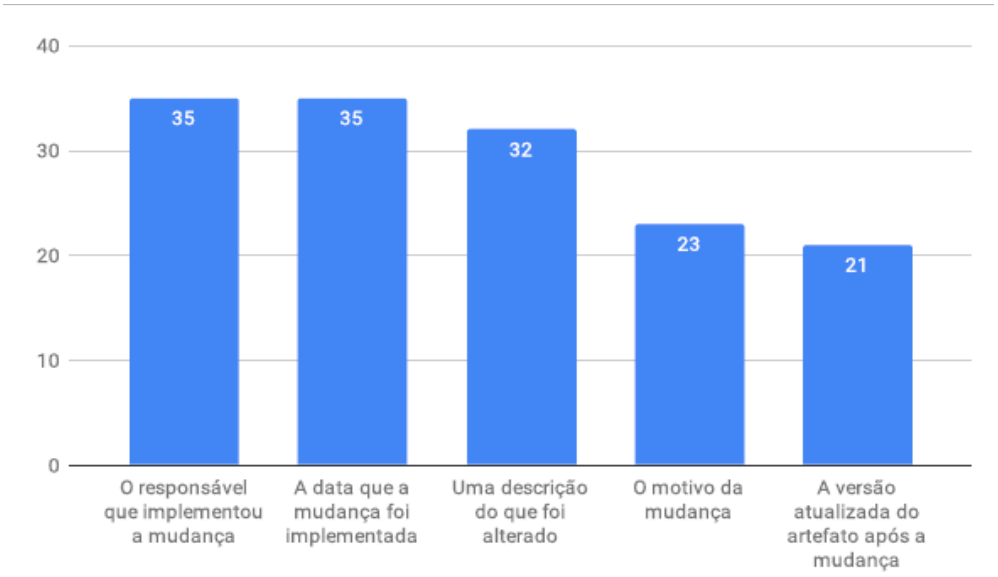
Figura 8 – Ferramentas para controle de versão



Fonte: elaborado pelo autor.

Sobre do histórico de alterações, 38 alunos o utilizam em seus documentos, no entanto 29 não. Dos 38 alunos que utilizam o histórico de alterações, 35 apontam que é necessário informar quem alterou o artefato, 35 a data em que a mudança foi implementada, 32 a descrição da mudança, 23 o motivo da mudança e 21 o número de versão atualizado.

Figura 9 – Dados do histórico de alterações



Fonte: elaborado pelo próprio autor.

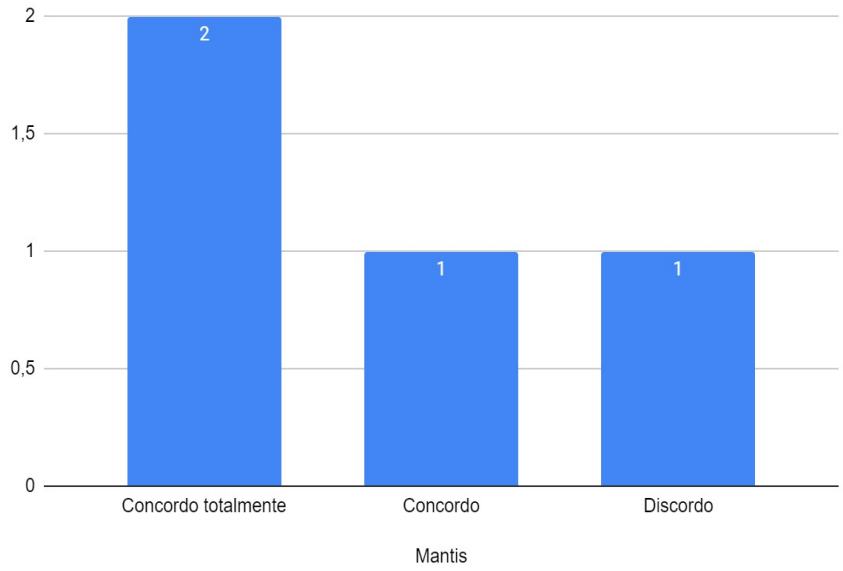
Com o intuito de saber a opinião dos alunos se é melhor atribuir a responsabilidade da aplicação da GCS a uma única pessoa ou à equipe toda. 52 alunos concordaram que é melhor ter um membro do projeto que seja responsável pela aplicação do processo de GCS por motivos de organização. Com uma pessoa coordenando facilita a execução das atividades previstas. Porém, 11 alunos afirmam que todos da equipe devem ser responsáveis pela execução do processo de GCS e não tem necessidade de apenas uma pessoa assumir toda a responsabilidade. 3 alunos afirmam que depende do projeto onde o processo será aplicado, e 1 não soube responder.

8 ANÁLISE DAS FERRAMENTAS DE CONTROLE DE MUDANÇAS

Na disciplina de Tópicos Avançados em Engenharia de Software, ofertada no curso de Ciência da Computação da UFC Campus Pici, foi conduzido um tutorial de duas ferramentas, Mantis e GitLab, a fim de analisar sua aplicação nas atividades da fase de Controle de Mudanças (CM) do processo STONE. Foi fornecido um cenário onde os alunos desempenharam as atividades usando ambas as ferramentas e seus resultados foram comparados através de um questionário. No total 4 alunos participaram desse tutorial e responderam ao questionário[2].

Primeiramente, é questionado aos alunos se a ferramenta melhora o desempenho ao realizar atividades de controle de mudanças. Em relação à ferramenta Mantis (Figura 10.a), 2 alunos concordam totalmente, 1 concorda parcialmente e 1 discorda parcialmente. Já a ferramenta GitLab (Figura 10.b), os 4 alunos concordam totalmente.

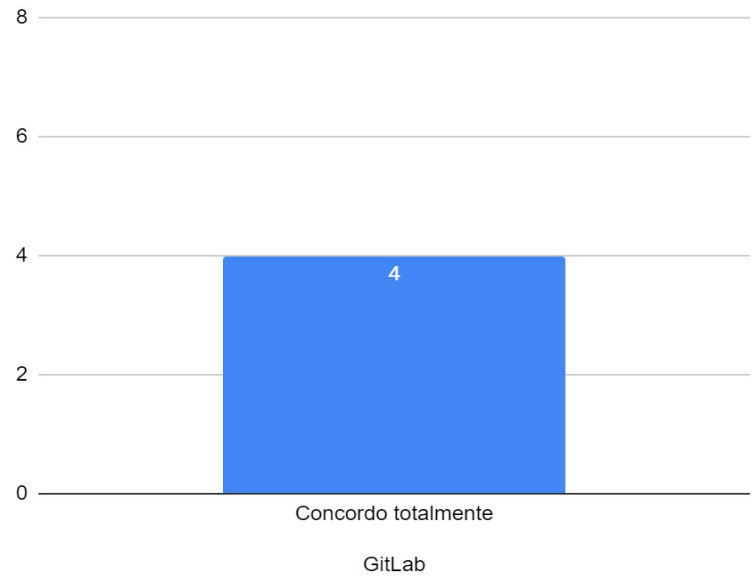
Figura 10.a – Desempenho das atividades de CM no Mantis



Fonte: elaborado pelo autor

[2]https://forms.gle/1LhYS3YAA2yr4dKx8

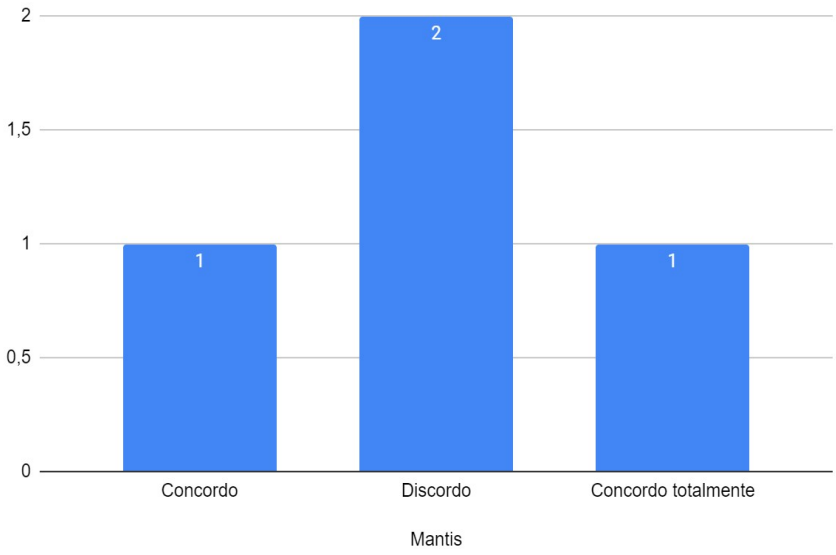
Figura 10.b – Desempenho das atividades de CM no GitLab



Fonte: elaborado pelo autor

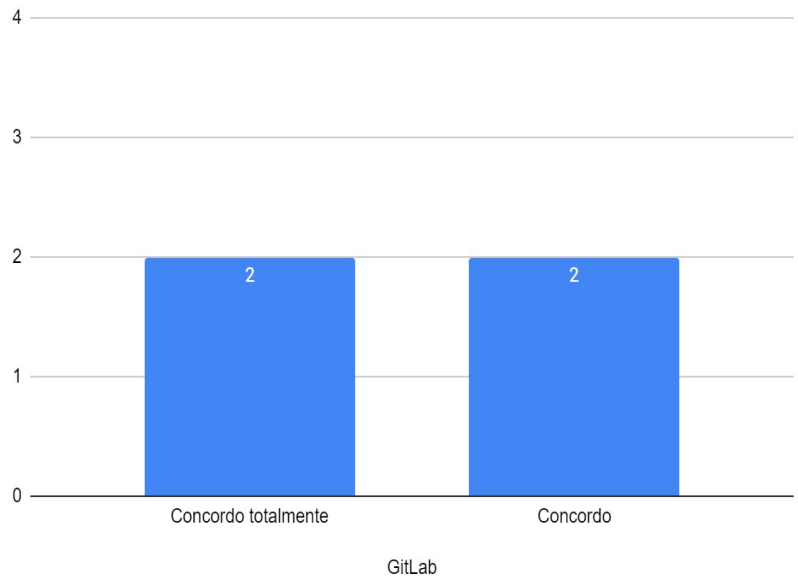
Em relação ao aumento da produtividade ao realizar atividades de controle de mudanças com a ferramenta Mantis (Figura 11.a), 2 discordam parcialmente, 1 concorda parcialmente e 1 concorda totalmente. Já para o GitLab (Figura 11.b), 2 concordam parcialmente e 2 concordam totalmente.

Figura 11.a – Produtividade das atividades de CM no Mantis



Fonte: elaborado pelo autor

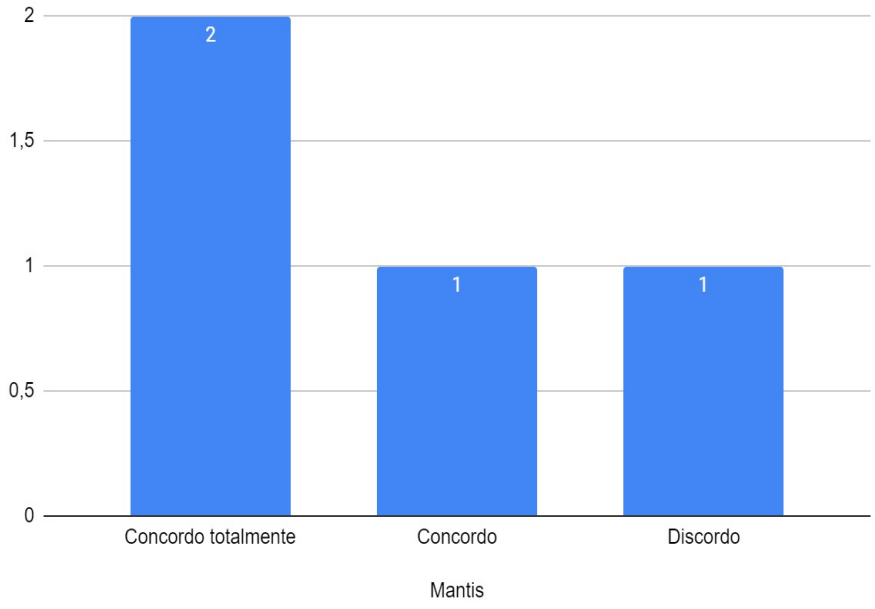
Figura 11.b – Produtividade das atividades de CM no GitLab



Fonte: elaborado pelo autor

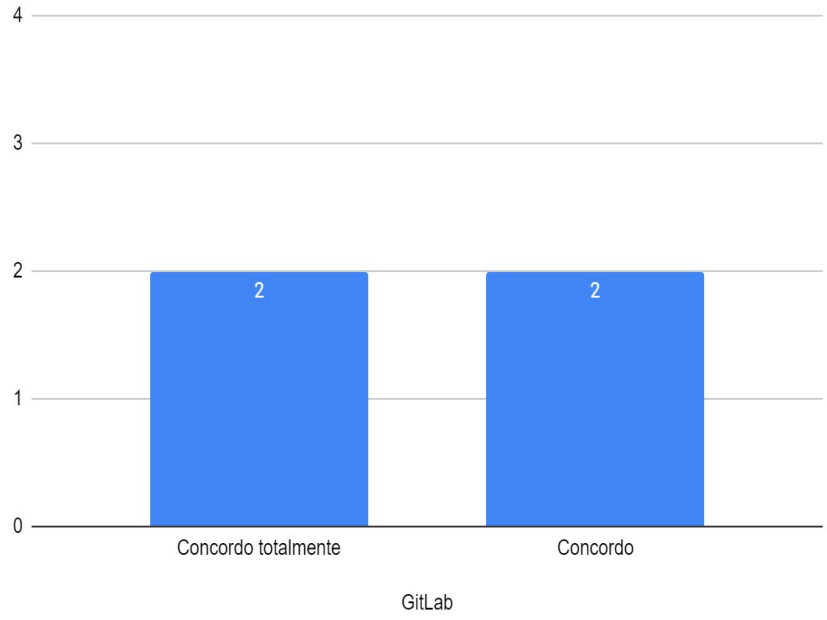
Depois foi questionado se a ferramenta aumenta a eficácia dos alunos ao executarem as atividades de controle de mudanças. Em relação ao Mantis (Figura 12.a), 2 concordam totalmente, 1 concorda parcialmente e 1 discorda parcialmente. Em contraste, o GitLab (Figura 12.b) apresentou respostas mais favoráveis. 2 concordam parcialmente e 2 concordam totalmente.

Figura 12.a – Eficácia das atividades de CM no Mantis



Fonte: elaborado pelo autor

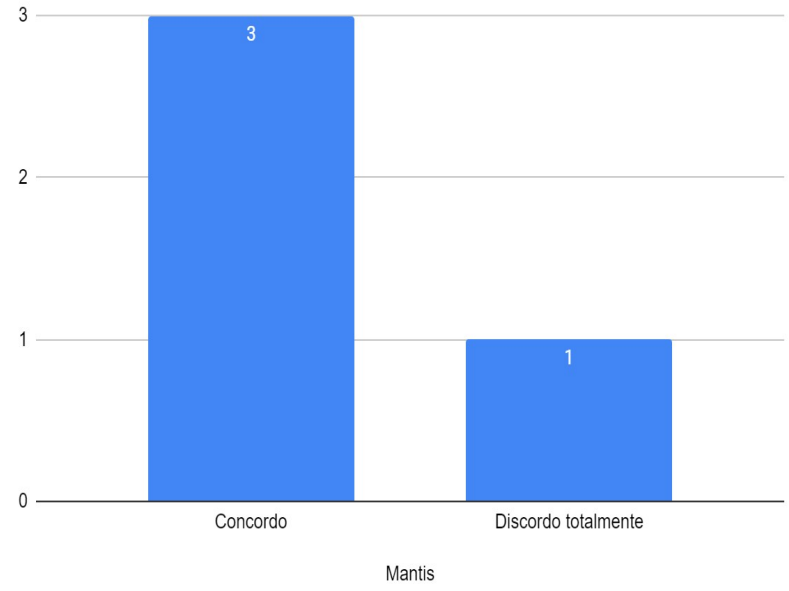
Figura 12.b – Eficácia das atividades de CM no GitLab



Fonte: elaborado pelo autor

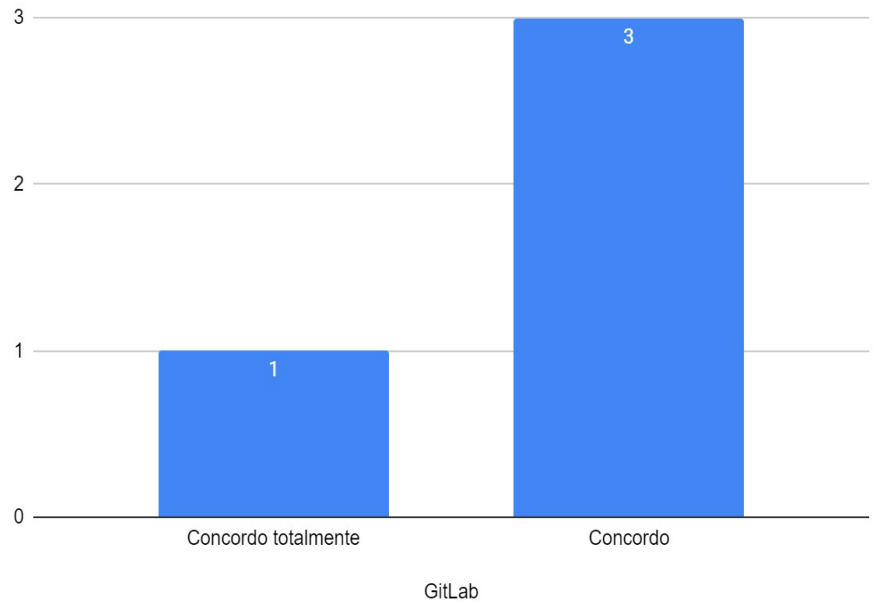
A respeito da experiência do usuário com as ferramentas, foi indagado se a interação com ambas é clara e compreensível. Para a ferramenta Mantis (Figura 13.a), 3 concordam parcialmente e 1 discorda totalmente. Para o GitLab (Figura 13.b), 3 concordam parcialmente e 1 concorda totalmente.

Figura 13.a – Experiência do usuário das atividades de CM no Mantis



Fonte: elaborado pelo autor

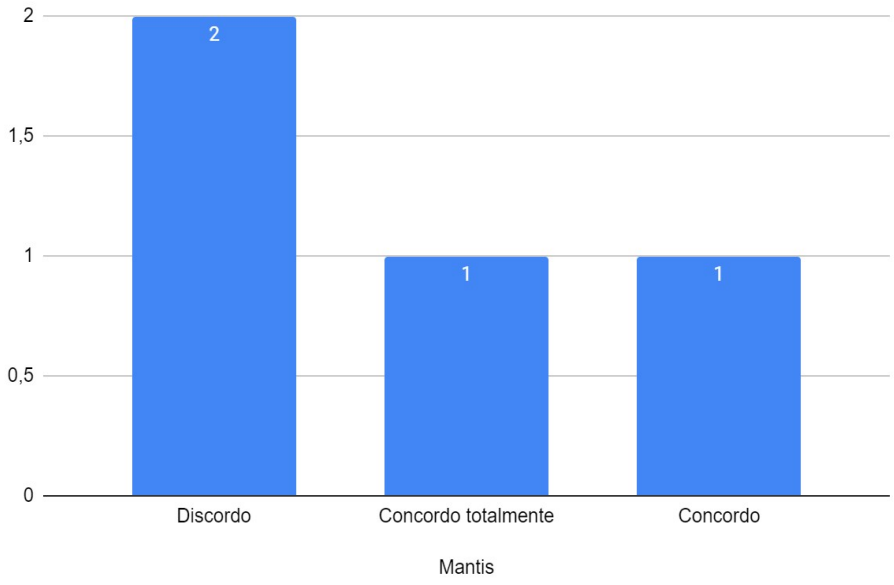
Figura 13.b – Experiência do usuário das atividades de CM no GitLab



Fonte: elaborado pelo autor

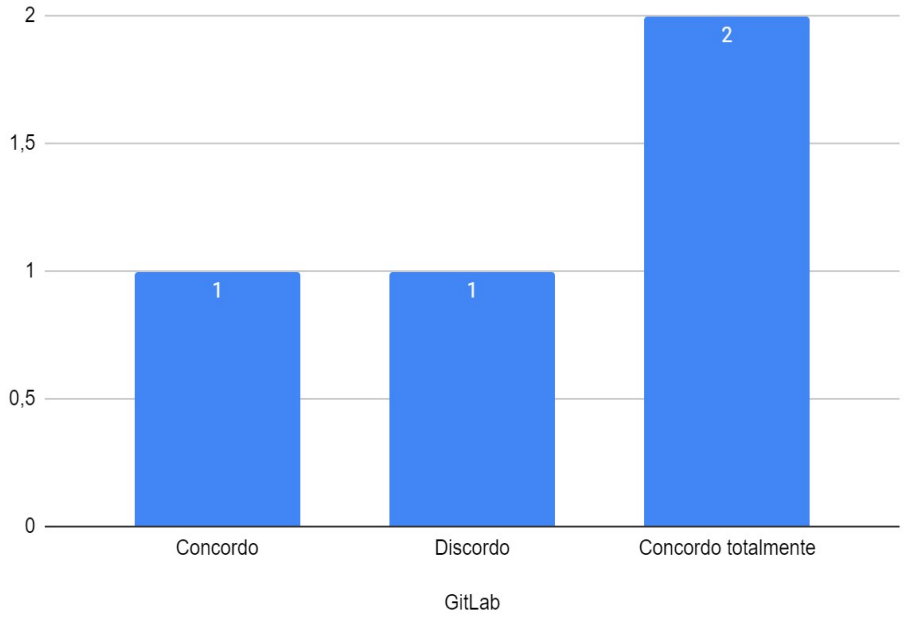
Também foi questionada a facilidade de uso de ambas as ferramentas. Em relação ao Mantis (Figura 14.a), 2 discordam parcialmente, 1 concorda parcialmente e 1 concorda totalmente com essa afirmação. Para a ferramenta GitLab (Figura 14.b), 2 concordam totalmente, 1 concorda parcialmente e 1 discorda parcialmente.

Figura 14.a – Facilidade de uso das atividades de CM no Mantis



Fonte: elaborado pelo autor

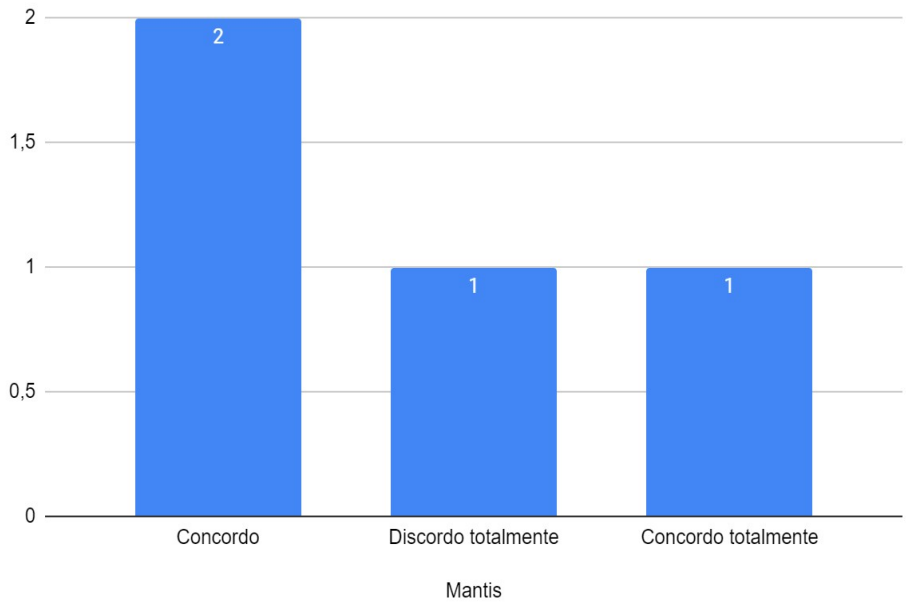
Figura 14.b – Facilidade de uso das atividades de CM no GitLab



Fonte: elaborado pelo autor

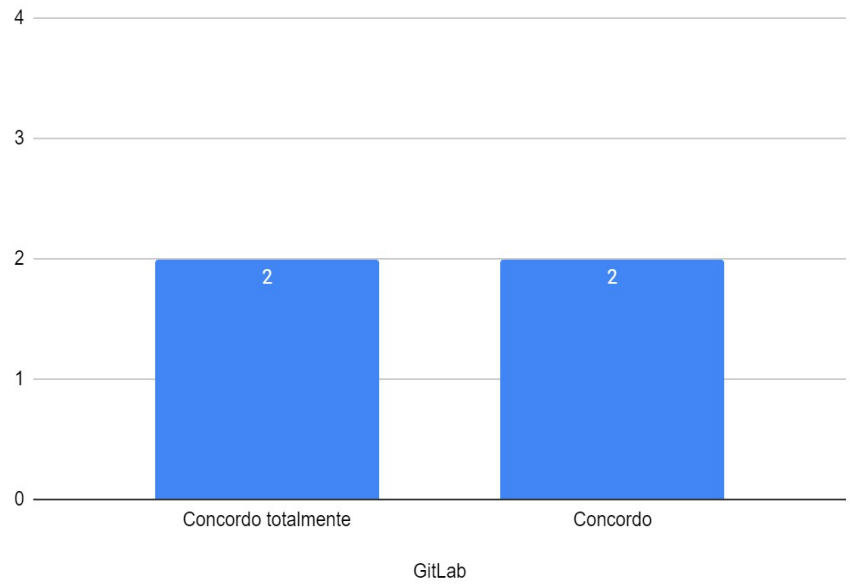
Em relação à experiência de uso da ferramenta Mantis (Figura 15.a), 2 concordam que previram que a utilizarão futuramente, 1 discorda parcialmente e 1 concorda totalmente. Para a ferramenta GitLab (Figura 15.b), 2 concordam parcialmente e 2 concordam totalmente que a utilizarão futuramente.

Figura 15.a – Experiência de uso das atividades de CM no Mantis



Fonte: elaborado pelo autor

Figura 15.b – Experiência de uso das atividades de CM no GitLab



Fonte: elaborado pelo autor

Com esses resultados, observa-se que nesse contexto, a ferramenta GitLab é mais eficiente do que a ferramenta Mantis. Para embasar essa teoria, foi observado que a navegação da ferramenta Mantis é menos intuitiva do que o GitLab, a interface em si não facilita seu uso, a menos que o participante já esteja familiarizado com a ferramenta, contudo uma das vantagens citadas pelos alunos do Mantis é que ele fornece um mecanismo para controle de mudanças e defeitos em qualquer escopo do projeto em que ele está inserido. Sua principal característica é a manutenção do histórico de mudanças e aprovações realizadas ao do ciclo de vida do projeto. O GitLab, apesar de fornecer funcionalidades de controle de mudança voltado principalmente para o código-fonte do projeto, sua execução é mais clara no controle de mudanças e *issues* a nível de código. Vale ressaltar que o GitLab é uma ferramenta que vai além do gerenciamento de mudanças no projeto, ele também incorpora técnicas de integração e entrega contínua.

9 DISCUSSÃO

Esta seção tem como objetivo responder as questões de pesquisa levantadas no início da pesquisa.

A primeira questão indagava de que maneira a gerência de configuração pode ajudar no desenvolvimento de projetos de software desenvolvidos em âmbito acadêmico. Para isso, foi elaborado um processo de GCS com foco em projetos acadêmicos, com as fases de planejamento, controle de mudanças, controle de versão e gerenciamento de *releases*, a fim de auxiliar os alunos a aplicar o processo de GCS no escopo do projeto através de um planejamento prévio, auxiliar os alunos a manter a rastreabilidade das mudanças ocorridas, realizar o versionamento correto nos artefatos do projeto, tanto no documento quanto no código, e, por fim, entregar o produto ao professor de forma a aproveitar o feedback dado por ele, e melhorar o produto que já foi feito. A fim de capturar as necessidades específicas dos alunos em relação a GCS, foi realizado um *survey* e seus dados foram analisados e incorporados no processo.

A segunda questão interpelava quais elementos de um processo de GCS tradicional podem ser utilizados para auxiliar os estudantes no controle de mudanças de seus projetos acadêmicos. Para alterações nos artefatos textuais, geralmente utiliza-se o histórico de alterações para auxiliar a controlar as mudanças. Também pode-se optar pela documentação que cada mudança ocorrida dos artefatos do projeto, através de formulários e relatórios propostos em modelos tradicionais como o IEEE 1042 e ISO 10007. Além dos artefatos, foram incorporados os papéis propostos por esses modelos, como gerente de configuração, solicitante da mudança, testador, entre outros.

Por fim, a última pergunta buscava identificar quais ferramentas são utilizadas em projetos acadêmicos para fazer o controle de versões de artefatos de software. As principais ferramentas identificadas foram, o Google Drive para controle de versão dos documentos produzidos. Para controle de versão em código-fonte, os alunos geralmente utilizam o Git em conjunto com o GitHub para armazenar as modificações dos arquivos-fonte do projeto. Esses dados foram retirados do questionário inicial aplicado com os alunos.

10 AMEAÇAS À VALIDADE

Dentre as atividades do processo e a sua aplicação uma limitação diz respeito à definição do processo para atender às necessidades dos alunos no âmbito acadêmico. Para minimizar essa ameaça foi feito um estudo nas normas técnicas e modelos de qualidade que tratavam GCS para que as atividades básicas fossem contempladas no processo, porém, adaptadas aos projetos acadêmicos. Além disso, foi feita uma pesquisa através de um questionário aplicado aos alunos do curso de CC e áreas afins para que as suas necessidades fossem realmente identificadas. Embora, a quantidade de participantes possa não ter sido expressiva (67 alunos), os alunos eram de cursos, de semestres e de campi diferentes da universidade.

Outra limitação refere-se análise das ferramentas de controle de mudanças em apenas uma disciplina. Para minimizar essa ameaça, foi selecionada uma turma de graduação de uma disciplina optativa em que os alunos apresentam um maior interesse nas atividades extraclasse. Dessa forma, houve adesão da maioria da turma no desempenho das atividades atribuídas durante esta análise.

11 CONCLUSÃO E TRABALHOS FUTUROS

O trabalho realizado consistiu na proposta de um processo de Gerência de Configuração de Software com foco em projetos acadêmicos. Para isso, foi realizada uma revisão bibliográfica dos trabalhos existentes na área, a fim de incorporar e adaptar práticas de GCS que correspondam às necessidades dos alunos. Em seguida, foi aplicado um questionário com alunos, a fim de avaliar as atividades da primeira versão do processo STONE. Após uma análise dos dados coletados do questionário, o processo foi refinado. Em seguida foi executada uma avaliação de ferramentas que podem automatizar as atividades de controle de mudanças do processo STONE. Os dados coletados foram utilizados para a melhoria e definição da versão final do processo.

Para trabalhos futuros pretende-se aplicar o processo STONE em turmas que introduzem conceitos de Engenharia de Software e Gerência de Configuração, a fim de avaliar sua eficiência. Também é necessário avaliar as ferramentas de controle de mudanças em uma amostra maior, com mais alunos a fim de reforçar a validade de seus resultados. Analisar o uso de outras ferramentas, como ferramentas de controle de versão, gerenciamento de releases e planejamento de GCS, com o intuito de automatizar ao máximo as atividades do processo STONE. Para reforçar a aplicação do processo STONE pelos alunos, pretende-se elaborar um jogo educacional, a fim de facilitar o entendimento do processo pelos alunos, e avaliar a eficiência do jogo em turmas do curso de Computação e afins.

REFERÊNCIAS

AGARWAL, Puneet. **Continuous SCRUM: agile management of SAAS products**. In: Proceedings of the 4th India Software Engineering Conference. ACM, 2011. p. 51-60.

APEL, Sven; LEßENICH, Olaf; LENGAUER, Christian. **Structured merge with auto-tuning: balancing precision and performance**. In: Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. ACM, 2012. p. 120-129.

ARORA, Ritu; GOEL, Sanjay; MITTAL, R. K. **Supporting collaborative software development in academic learning environment: A collaborative pair and quadruple programming based approach**. In: 2017 Tenth International Conference on Contemporary Computing (IC3). IEEE, 2017. p. 1-7.

BASHIR, Muhamamd Farhan; QADIR, Muhammad Abdul. **Traceability techniques: A critical study**. In: Multitopic Conference, 2006. INMIC'06. IEEE. IEEE, 2006. p. 265-268.

CHO, Hyun; GRAY, Jeff; SUN, Yu. **Quality-aware academic research tool development**. In: Software Engineering Conference (APSEC), 2012 19th Asia-Pacific. IEEE, 2012. p. 66-72.

Comissão de Estudo de Gestão de Configuração. **NBR ISO 10007: Gestão da qualidade - Diretrizes para a gestão de configuração**. Rio de Janeiro, 1996.

Comitê Brasileiro de Processamento de Dados. **NBR ISO/IEC 12207 - Tecnologia de informação – Processos de ciclo de vida de software**. Rio de Janeiro, 1998.

CONN, Richard. **A reusable, academic-strength, metrics-based software engineering process for capstone courses and projects**. In: ACM SIGCSE Bulletin. ACM, 2004. p. 492-496.

CRNKOVIC, Ivica; FUNK, Peter; LARSSON, Magnus. **Processing requirements by software configuration management**. In: euromicro. IEEE, 1999. p. 2260.

DURAN, Rodrigo et al. **Institucionalização da Gerência de Configuração no Desenvolvimento de Software de uma Organização**. In: VII Simpósio Internacional de Melhoria de Processo de Software, São Paulo, Brazil. 2005. p. 60-85.

ESTUBLIER, Jacky; GARCIA, Sergio. **Concurrent engineering support in software engineering**. In: Automated Software Engineering, 2006. ASE'06. 21st IEEE/ACM International Conference on. IEEE, 2006. p. 209-220.

FASANO, Fausto. **Fine-grained management of software artefacts**. In: Software Maintenance, 2007. ICSM 2007. IEEE International Conference on. IEEE, 2007. p. 507-508.

FELICIANO, Joseph; STOREY, Margaret-Anne; ZAGALSKY, Alexey. **Student experiences using GitHub in software engineering courses: a case study**. In: Proceedings of the 38th International Conference on Software Engineering Companion. ACM, 2016. p. 422-431.

FERREIRA, Natasha N. Vitó; LANGERMAN, Josef J. **Proving that the release management process can enhance throughput in software development projects**. In:

Computer Science & Education (ICCSE), 2014 9th International Conference on. IEEE, 2014. p. 419-424.

GOWTHAM, S. **Revision Control System (RCS) in computational sciences and engineering curriculum**. In: XSEDE. 2014. p. 76:1-76:3.

HALEY, Elise R.; COLLINS, Galen B.; COE, David J. **The wonderful world of wiki benefits students and instructors**. IEEE Potentials, v. 27, n. 2, 2008.

HUMBLE, Jez; FARLEY, David. **Continuous delivery: reliable software releases through build, test, and deployment automation**. Boston: Addison-Wesley, 2011.

IEEE Computer Society**. Guide to Software Engineering Body of Knowledge (SWEBOK)**. Los Alamitos: CS Press, 2001.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **IEEE Guide to Software Configuration Management: An American Standard**. IEEE, 1988.

ISO/IEC/IEEE 24765:2009, Systems and Software Engineering–Vocabulary.

KERTÉSZ, Csaba-Zoltán. **Using GitHub in the classroom—A collaborative learning experience**. In: 2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME). 2015. p. 381-386.

KHRAIWESH, Mahmoud. **Configuration Management Measures in CMMI**. International Journal of Applied Engineering Research, v. 12, n. 18, p. 7546-7557, 2017.

KRUSCHE, Stephan et al. **Rugby: an agile process model based on continuous delivery**. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering. ACM, 2014. p. 42-50.

KRUSCHE, Stephan; BERISHA, Mjellma; BRUEGGE, Bernd. **Teaching code review management using branch based workflows**. In: Proceedings of the 38th International Conference on Software Engineering Companion. ACM, 2016. p. 384-393.

LINSBAUER, Lukas; EGYED, Alexander; LOPEZ-HERREJON, Roberto Erick. **A variability aware configuration management and revision control platform**. In: Proceedings of the 38th International Conference on Software Engineering Companion. ACM, 2016. p. 803-806.

LINSBAUER, Lukas; BERGER, Thorsten; GRÜNBACHER, Paul. **A classification of variation control systems**. In: ACM SIGPLAN Notices. ACM, 2017. p. 49-62.

MAGELA, Rogerio. **Engenharia de Software Aplicada: Princípios** (volume 1). Alta Books: 2006.

MÄKIAHO, Pekka; PORANEN, Timo; SEPPI, Ari. **Version control usage in students' software development projects**. In: Proceedings of the 15th International Conference on Computer Systems and Technologies. ACM, 2014. p. 452-459.

MOLINARI, Leonardo. **Gerência de Configuração: técnicas e práticas no desenvolvimento do software**. Florianópolis: Visual Books, 2007.

PRESSMAN, Roger S. **Engenharia de software: uma abordagem profissional**. Porto Alegre: AMGH,2011.

Project Management Institute (PMI). **Practice Standard for Project Configuration Management**. Project Management Institute, 2007.

QUSEF, Abdallah; ALBADARNEH, Israa; ALBADARNEH, Aalaa. **GitBull: Source code hosting web application**. In: Applied Electrical Engineering and Computing Technologies (AEECT), 2015 IEEE Jordan Conference on. IEEE, 2015. p. 1-6.

Rational Software Corporation. **Rational Unified Process** (RUP). Rational Software Corporation, 2002.

RAYANA, Rayene Ben et al. **GitWaterFlow: a successful branching model and tooling, for achieving continuous delivery with multiple version branches**. In: Proceedings of the 4th International Workshop on Release Engineering. ACM, 2016. p. 17-20.

RÍOS, Julio César Cortés et al. **A methodology for using GitLab for software engineering learning analytics**. In: Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering. IEEE Press, 2019. p. 3-6.

SHAMS, Armin; SHARIF, Hossein; KERMANSHAH, Ali. **Holistic Change Management: Importance and methodological challenges**. In: Technology & Engineering Management Conference (TEMSCON), 2017 IEEE. IEEE, 2017. p. 272-276.

SOFTEX. **MR-MPS-SV – Guia de Implementação – Parte 2**:2013.

SOMMERVILLE, Ian. **Software Engineering**. 9th editon. Boston: Addison-Wesley, 2011.

TIRKEY, Abhishek; GARY, Kevin A. **Curricular change management with Git and Drupal: A tool to support flexible curricular development workflows**. In: Software Engineering Research, Management and Applications (SERA), 2017 IEEE 15th International Conference on. IEEE, 2017. p. 247-253.

TORCHIANO, Marco; BRUNO, Giorgio. **Integrating Software Engineering Key Practices into an OOP Massive In-Classroom Course: an Experience Report**. arXiv preprint arXiv:1804.01700, 2018.

WAZLAWICK, Raul Sidnei. **Engenharia de software: conceitos e práticas**. Rio de Janeiro: Elsevier, 2013.

APÊNDICE A

Trabalho da disciplina XXXXX

PLANO DE GERÊNCIA DE CONFIGURAÇÃO

Membros da equipe:
Rebeca Maia Pontes

Fortaleza, CE
03 de março de 2019

Histórico de versões

Data	Versão	Comentários	Autor
03/03/19	1.0	Criação da estrutura inicial	Rebeca

Índice

1. INTRODUÇÃO

Na introdução é fornecido um overview simplificado das atividades de GCS para que os envolvidos com o projeto possam ter um claro entendimento do Plano. A introdução deve conter quatro seções: o propósito do plano, o escopo, o padrão de numeração dos ICs e o glossário.

1.1. Propósito [1]

O propósito deve explicar brevemente a importância do PGC e apontar à quem ele se destina.

1.2. Escopo [1]

O escopo deve salientar a aplicabilidade da GCS, as limitações e as suposições nas quais o PGC é baseado. Os seguintes itens devem ser incluídos:

a. Descrição do projeto
b. A identificação dos itens de configuração descrita no artefato no artefato.
c. O grau de formalidade, nível de controle, e a parte do ciclo de vida do software na qual a GCS será aplicada nesse projeto
d. Limitações, como restrições de tempo, que se aplicam ao PGC
e. Suposições que podem ter um impacto no custo, cronograma, ou na capacidade de execução das atividades de GCS

1.3. Convenção de numeração [1]

Por convenção, a referência a itens de configuração é feita através da abreviação do nome ‘item de configuração’, seguidos do número do item, de acordo com a especificação a seguir:
[abreviação. número do item]
Ex.: [IC001]
Os itens de configuração devem ser identificados com um identificador único. A numeração inicia com o identificador [IC001] e prossegue sendo incrementada à medida que forem surgindo novos requisitos.

Todos os itens de configuração devem ter um número de versão segundo o seguinte padrão:

X.Y

Em que:

X é o número que representa uma versão final do artefato
Y é o número que representa um draft da versão X do artefato
O número de versão muda de acordo com as regras descritas a seguir:

- A primeira versão do artefato deve ser 0.1
- A cada modificação no artefato, o valor Y deve ser incrementado
- Após cada aprovação do artefato, a versão X deve ser incrementada e o valor de Y volta para 0

1.4. Identificando itens de configuração [1]

A Especificação deve registrar os itens a serem controlados, levando em consideração os artefatos de produto e os artefatos do processo. Cada IC deve conter os seguintes dados:

a. Identificador único
b. Versão mais recente do IC
c. Data de criação
d. Status do IC (A fazer, em desenvolvimento, concluído)
e. Papel funcional do IC (documento de requisitos, código-fonte)
f. Localização do IC
g. Criador do IC

O plano de ensino da cadeira fornecido pelo professor pode servir de fonte para identificação dos itens de configuração, bem como os *milestones* do projeto.

1.5. Rastreabilidade entre itens de configuração

O PGC deve descrever as dependências entre os itens de configuração a serem controlados a fim de manter consistência entre os artefatos do projeto através de uma matriz de rastreabilidade.

1.6. Estrutura de diretórios

O PGC deve conter a estrutura de diretórios/repositórios para a organização dos itens de configuração a fim de facilitar a localização e alteração destes.
Cada diretório deve conter a versão mais recente dos itens de configuração e uma pasta separada com as versões passadas, caso seja necessário recuperá-las.

1.7. Glossário [1]

Termos-chave devem ser definidos conforme eles se aplicam ao PGC aa fim de estabelecer uma terminologia comum entre todos os usuários do plano. O PGC deve incluir um glossário a uma referência ao glossário do projeto.

2. PLANEJAMENTO DA GERÊNCIA DE CONFIGURAÇÃO DE SOFTWARE

2.1. Políticas de aplicação do processo [2]

O processo de GCS incorpora regras, procedimentos, metodologias e recursos para garantir que:

- A configuração do sistema e/ou seus itens são documentados.
- Mudanças feitas aos itens de configuração no decorrer do desenvolvimento são esperadas, e consequentemente, efetuadas sem quaisquer consequências adversas.
- Mudanças são gerenciadas até serem incorporadas em todos os itens de configuração afetados.

As seguintes políticas são inerentes do processo Muutus, e podem ser adaptadas conforme o projeto em que está sendo inserido.

1. Definir a estrutura de diretórios onde os itens de configuração serão armazenados conforme os repositórios selecionados na atividade 'Preparar ambiente de configuração'.
2. Padronizar ferramentas para controle de mudanças, controle de versão e validação dos itens de configuração (quando for o caso).
3. Manter planejamento das atividades e *milestones* constantemente atualizados e disponibilizados a todos os membros da equipe.
4. Caso o projeto tenha atividades mais complexas, definir hierarquias de permissão acesso aos itens de configuração.
5. A cada mudança implementada, alterar histórico de alterações do(s) item(ns) de configuração afetado(s) e manter as versões anteriores armazenadas e organizadas de forma coesa.
6. Para projetos que envolvem a implementação de código-fonte, deve-se definir a estrutura dos branches.

2.2. Política de controle de versão

Esta política estabelece requisitos para o projeto XXXX para gerenciar o código fonte, garantindo um controle de versão adequado.
É exigido que os programadores desenvolvam o código-fonte para:

- Usar e manter um sistema de gerenciamento de código-fonte padrão da indústria seguindo as melhores práticas do setor para controle de código-fonte e gerenciamento de mudanças.

- Manter a coordenação de todos os elementos das alterações do programa no caso de um erro ou falha de produção. Esses elementos incluem, mas não se limitam a releases, branches de código, etc.
- Manter toda a documentação necessária vinculada ou anexada aos ativos/ pacotes de código-fonte apropriados. Essa documentação inclui, mas não se limita a especificações técnicas, requisitos de negócios, notas de release.
- Manter o controle de versão sobre os ativos do código-fonte para oferecer suporte à recuperação de versões anteriores.
- Manter a separação apropriada de responsabilidades entre desenvolvedores e os outros membros da equipe em um nível necessário para proteger o código-fonte contra alterações não autorizadas.
- Manter o controle de versão apropriado entre as bibliotecas de teste e desenvolvimento.

Além disso, proteja o código-fonte contra perda não intencional:

- Garantir que as bibliotecas desenvolvimento e teste, e também o código-fonte, não possam ser alteradas sem as aprovações apropriadas.
- Garantir que as cópias de backup do sistema de gerenciamento de código-fonte sejam feitas diariamente e alocadas para armazenamento protegido fora do ambiente de produção.

2.3. Responsabilidades [1]

A alocação das atividades do processo aos membros do projeto deve ser especificada. Para cada atividade listada no processo, o título do trabalho e o nome do membro da equipe deve ser especificada. Uma matriz que relaciona os papéis definidos acima às atividades e tarefas do processo pode ser útil para documentação das responsabilidades.

2.4. Cronograma [1]

O cronograma estabelece a sequência e a coordenação para as atividades do processo de GCS e para todos os eventos que afetam a implementação do PGC. O plano deve descrever o relacionamento entre atividades-chave de GCS aos milestones do projeto. Os milestones devem incluir procedimentos de implementação de controle de mudanças, datas de começo e término do desenvolvimento dos ICs e as datas para entrega dos releases.

2.5. Recursos [1]

Os recursos devem identificar ferramentas e frameworks de software, técnicas e treinamento necessário para a implementação das atividades de GCS.
As ferramentas podem ser aplicadas para documentação e rastreamento dos ICs, controle e versionamento de código, processamento de solicitação de mudança, comunicação entre os membros do projeto, rastreamento de mudanças e reporte de status, armazenamento, retenção e recuperação dos itens de configuração, ou até o próprio planejamento do processo de GCS.
Algumas sugestões de ferramentas:

- Repositórios para documentos: Google Drive e Dropbox
- Repositórios para código-fonte: Github, Sourceforge e Bitbucket
- Ferramentas para controle de mudanças: Mantis, Bugzilla, Trac, Jira, Bamboo
- Ferramentas para controle de versão: Git, SVN, CVS, Mercurial
- Ferramentas de teste:
    - Checklists para inspeção em documentos arquiteturais [5]
    - Ferramentas para teste de regressão: Selenium
    - Ferramenta para teste estrutural: SonarQube
    - Ferramentas para capture/playback (que gravam tudo que acontece durante o teste - como movimentos de mouse, valores inseridos, teclas usadas, elementos de interface gráfica acionados etc. - e depois podem executar outra vez esse mesmo teste, mas para uma nova versão do programa executável em teste), como STW/CAPBAK[3] ou XRunner[4]
- Ferramenta para comunicação com o professor: SIGAA, correio eletrônico.

2.5.1. Canal de comunicação entre stakeholders

O canal de comunicação deve ser acordado entre os envolvidos no projeto antes do começo das atividades, a fim de evitar divulgação quaisquer informações errôneas, bem como estabelecer o planejamento das atividades do projeto.
A forma como o canal de comunicação será estabelecido dependerá do nível de formalidade do projeto. Para projetos mais informais, comumente utiliza-se Whatsapp ou Discord. Para projetos onde é necessário manter registro das mensagens enviadas, utiliza-se Slack ou correio eletrônico.
Para disponibilização de informações comuns a todos os *stakeholders*, recomenda-se a utilização de uma wiki, como o Github, ou ferramentas como

Slack, Trello e Asana.

REFERÊNCIAS

[1] VIEW, T. IEEE Standard for Software Configuration Management Plans. IEEE Std, v. 2005, p. 1-19, 2005.
[2] HANDBOOK, Military. Configuration management guidance. MIL-HDBK-61A (SE) Department of Defense–United States of America, 2001.
[3] SOFTWARE RESEARCH Inc. STW/coverage tool suite for C. User manual, Release 2, maio de 1994.
[4] Mercury Interactive. XRunner user´s guide. Mercury Interactive Inc., 1994.
[5] BARCELOS, R. F.; TRAVASSOS, G. H. ArqCheck: Uma abordagem para inspeção de documentos arquiteturais baseada em checklist. V Simpósio Brasileiro de Qualidade de Software-SBQS 2006, 2006.

APÊNDICE B

Trabalho da disciplina XXXXX

FORMULÁRIO DE SOLICITAÇÃO DE MUDANÇA

Fortaleza, CE
03 de março de 2019

Histórico de versões

Data	Versão	Comentários	Autor
03/03/19	1.0	Criação da estrutura inicial	Rebeca

1. SOLICITANDO MUDANÇAS [1]

Deve-se especificar os procedimentos para solicitação de uma mudança em um IC e a informação a ser documentada. No mínimo, a informação a ser registrada para a mudança proposta deve conter o seguinte

a. O(s) nome(s) e versão(ões) do(s) IC(s) onde a mudança é desejada.
b. Nome do solicitante
c. Data de solicitação
d. Indicação de urgência
e. A necessidade da mudança
f. Descrição da solicitação de mudança
g. Justificativa da mudança
h. Status de aprovação

Informações adicionais como prioridade ou classificação podem ser incluídos para esclarecer a significância da solicitação e auxiliar na sua análise e avaliação. Outros dados como, número de solicitação de mudança e status podem ser incluídos para rastreamento da mudança.

REFERÊNCIAS

[1] VIEW, T. IEEE Standard for Software Configuration Management Plans. IEEE Std, v. 2005, p. 1-19, 2005.

APÊNDICE C

Trabalho da disciplina XXXXX

RELATÓRIO DE MUDANÇA

Fortaleza, CE
03 de março de 2019

Histórico de versões

Data	Versão	Comentários	Autor
03/03/19	1.0	Criação da estrutura inicial	Rebeca

1. IMPLEMENTANDO MUDANÇAS [1]

Deve-se especificar as atividades para verificação e implementação de uma mudança aprovada. A informação registrada para conclusão da mudança deve conter:

a. A(s) solicitação(ões) de mudança associada(s)
b. Os nomes e versões dos itens afetados
c. Data de alteração
d. Responsável pela mudança
e. Data de entrega
f. Descrição da mudança
g. Identificador da nova versão

Informação adicional como software de suporte utilizado para implementar a mudança pode ser incluída.

REFERÊNCIAS

[1] VIEW, T. IEEE Standard for Software Configuration Management Plans. IEEE Std, v. 2005, p. 1-19, 2005.

APÊNDICE D

Trabalho da disciplina XXXXX

RELATÓRIO DE ANOMALIAS

Fortaleza, CE
03 de março de 2019

Histórico de versões

Data	Versão	Comentários	Autor
03/03/19	1.0	Criação da estrutura inicial	Rebeca

1. INTRODUÇÃO

O relatório de anomalias documenta qualquer evento ocorrido na atividade de teste/validação e que mereça ser investigado. A complexidade da execução desta atividade irá depender do escopo, tempo e mão-de-obra do projeto, portanto este documento deverá ser adaptado para as necessidades do projeto em questão.

2. RESUMO [1]

Descrever de modo resumido o incidente ocorrido. Identificar os itens de teste envolvidos (com versão e nível de revisão). Devem ser feitas referências aos procedimentos de teste, casos de teste e registros de teste associados.

3. DESCRIÇÃO DO PROBLEMA [1]

Fornecer uma descrição do problema, a qual deve incluir os seguintes itens:

- Entradas;
- Resultados esperados;
- Anomalias;
- Data e hora;
- Passo do procedimento;
- Ambiente;
- Tentativas de repetição; e
- Testadores.

Atividades relacionadas e observações que possam ajudar a isolar e corrigir a causa do problema devem ser incluídas.

4. IMPACTO [1]

Caso seja conhecido, indicar que impacto este problema pode vir a causar no plano de teste, no projeto de teste, nos procedimentos de teste, ou nos casos de teste.

5. INSPEÇÃO [2]

O registro de inspeção é utilizado para registrar decisões e ações definidas, para referência futura. Também são úteis para comunicação com os envolvidos no projeto, que não participaram da inspeção.

5.1. Local e Data

Informe o local e data da reunião de inspeção.

5.2. Decisões

Informe quais decisões foram tomadas ao longo da inspeção.

5.3. Não-Conformidades Encontradas

Informe as não-conformidades encontradas, classificando por tipo e severidade. Informar em qual item de configuração foi encontrado o problema e faça uma breve descrição.

ID	**Tipo** (**F**)altando/ (**E**)rrada/ e(**X**)tra/ (**U**)sabilidade/ (**P**)erformance/ (**N**)ão-defeito	**Severidade** (**P**)rincipal / (**S**)ecundária	**Localização**	**Descrição**
1				
2				
3				

REFERÊNCIAS

[1] Villas Boas, André Luiz de Castro - Gestão de configuração para teste de software / André Luiz de Castro Villas Boas. Campinas, SP: [s.n.], 2003.

[2] Sericolla Diniz, Alessandra Ap. – Processo de Inspeção em Artefatos de Teste Funcionais de Software. Dissertação (Mestrado em Engenharia da Computação: Instituto de Pesquisas Tecnológicas do Estado de São Paulo). 94p. São Paulo. IPT, 2008.

APÊNDICE E

Trabalho da disciplina XXXXX
RELATÓRIO DE FEEDBACK

Fortaleza, CE
03 de março de 2019

Histórico de versões

Data	Versão	Descrição	Autor
03/03/19	1.0	Criação da estrutura inicial	Rebeca

1. REGISTRANDO FEEDBACK

Deve-se especificar as atividades para gerenciamento dos releases entregues ao cliente. A informação registrada para registro do feedback deve conter:

a. Identificador do release
b. Os nomes e versões dos ICs afetados
c. Data de entrega
d. Descrição do feedback

Informação adicional como software de suporte utilizado para registrar o feedback pode ser incluída.