UNIVERSIDADE FEDERAL DO PAMPA

Vitor Hugo Maciel dos Santos

Uma Estratégia para Introduzir Pipelines de IC em Disciplinas de Resolução de Problemas V

Alegrete
2021

Vitor Hugo Maciel dos Santos

Uma Estratégia para Introduzir Pipelines de IC em Disciplinas de Resolução de Problemas V

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Prof. Dr. Fábio Paulo Basso

Alegrete
2021

Vitor Hugo Maciel dos Santos

UMA ESTRATÉGIA PARA INTRODUZIR PIPELINES DE IC EM DISCIPLINAS DE RESOLUÇÃO DE PROBLEMAS V

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Trabalho de Conclusão de Curso defendido e aprovado em: 01 de Outubro de 2021.

Banca examinadora:

_____

Prof. Dr. Fábio Paulo Basso

Orientador

Unipampa

_____

Prof. Dr. Elder de Macedo Rodrigues

Unipampa

_____

Prof. Dr. Maicon Bernardino da Silveira

Unipampa

Assinado eletronicamente por **MAICON BERNARDINO DA SILVEIRA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 01/10/2021, às 21:07, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **FABIO PAULO BASSO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 01/10/2021, às 21:08, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **ELDER DE MACEDO RODRIGUES**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 01/10/2021, às 21:52, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0629350** e o código CRC **5E7D698E**.

AGRADECIMENTOS

Primeiramente, gostaria de expressar minha profunda gratidão aos meus pais, cujo amor, incentivo e apoio incondicional foram fundamentais ao longo de toda minha jornada. Sem a dedicação e o carinho deles, este trabalho não seria possível. Em seguida, agradeço ao Dr. Fábio Paulo Basso pela orientação valiosa e pelo constante apoio ao longo do desenvolvimento deste TCC. A presença e o suporte dessas grandes pessoas foram essenciais para que eu chegasse até aqui.

“Estamos todos indo em direção ao mesmo
fim inevitável. Não precisa ter pressa!
(Taric, Personagem do jogo League of Legends)

RESUMO

A configuração de pipelines de Integração Contínua (IC) é uma tarefa complexa e crucial nas práticas de gerenciamento de configuração. Ela exige que engenheiros de software possuam habilidades técnicas para configurar servidores ou serviços de IC. Esses profissionais precisam selecionar recursos intensivos em conhecimento para automatizar processos de software. No contexto do gerenciamento de configuração, isso significa ter a capacidade de transformar esses recursos em tarefas de Integração, Implantação e Entrega Contínua, organizadas em pipelines. Essa habilidade requer um conhecimento especializado em ferramentas para gerenciar configurações em diversos projetos de software.

A busca por esse conhecimento é um dos objetivos das disciplinas de Resolução de Problemas (RP), especialmente da RP 5, do curso de Bacharelado em Engenharia de Software da Unipampa. Motivado por essa necessidade, este TCC explora as dificuldades que os alunos enfrentam na configuração de pipelines de IC e propõe uma estratégia de ensino.

As principais contribuições deste TCC são: 1) um estudo de mineração de repositórios de código-fonte, derivado da disciplina RP 5 e realizado em 2018; 2) uma pesquisa longitudinal (survey) com alunos de quatro edições dessa disciplina (2015, 2016, 2017 e 2018), que destaca a necessidade de uma estratégia de ensino que inclua elementos de gerenciamento de configuração; 3) um estudo de mapeamento sistemático da literatura sobre o tema, que evidencia o estado da arte e posiciona este TCC em termos de inovação; 4) uma análise prática dos servidores/serviços de IC e suas sintaxes de configuração, com o objetivo de motivar os alunos, apresentando exemplos de pipelines em três níveis de dificuldade; 5) uma estratégia para que alunos e professores possam introduzir pipelines de IC nas disciplinas de RP; e 6) um relato da implementação dessa estratégia na disciplina RP 5, realizada em 2021.

ABSTRACT

Configuring Continuous Integration (CI) pipelines is a complex and crucial task in configuration management practices. It requires software engineers to have the technical skills to configure CI servers or services. These professionals need to select knowledge-intensive resources to automate software processes. In the context of configuration management, this means having the ability to transform these resources into Continuous Integration, Deployment and Delivery tasks, organized into pipelines. This skill requires specialized knowledge of tools for managing configurations in various software projects.
The search for this knowledge is one of the objectives of the Problem Solving (PR) subjects, especially RP 5, in the Bachelor's degree course in Software Engineering at Unipampa. Motivated by this need, this Capstone explores the difficulties students face in configuring CI pipelines and proposes a teaching strategy.
The main contributions of this TCC are: 1) a source code repository mining study, derived from the RP 5 course and carried out in 2018; 2) a longitudinal survey of students from four editions of this course (2015, 2016, 2017 and 2018), which highlights the need for a teaching strategy that includes elements of configuration management; 3) a systematic mapping study of the literature on the subject, which highlights the state of the art and positions this TCC in terms of innovation; 4) a practical analysis of CI servers/services and their configuration syntaxes, with the aim of motivating students by presenting examples of pipelines at three levels of difficulty; 5) a strategy for students and teachers to introduce CI pipelines in RP courses; and 6) a report on the implementation of this strategy in RP course 5, which took place in 2021.

Key-words: continuous integration . CI. Pipeline.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O gerenciamento de configuração trata do desenvolvimento e o uso de padrões e procedimentos para o gerenciamento de sistemas de software em desenvolvimento (SOMMERVILLE, 2010). Os requisitos de um sistema sofrem mudanças constantemente durante a produção e uso, e isso acaba resultando na necessidade de se adicionar alterações ou novos requisitos em novas versões do sistema. Neste sentido, faz-se necessário gerenciar os sistemas em produção, pois sem isso é comum para a equipe de desenvolvimento perder a rastreabilidade do que foi alterado e em qual versão do sistema. Falha na gerência de configurações acarreta na perda de esforço das equipes.

O gerenciamento de alterações em software incorpora gerenciamento de mudanças, de correções de defeitos e de adaptações para diferentes hardwares e sistemas operacionais. Em uma organização de grande porte, há muitas versões de código em desenvolvimento operados por uma mesma equipe para diferentes clientes. Levando em consideração isso, a falta de um processo bem definido para gerenciamento de configuração implica em desperdício de tempo, seja na modificação da versão errada do sistema, e/ou entregas equivocadas de requisitos de sistema aos clientes, e/ou ineficiência na rastreabilidade de alterações em código-fonte e, finalmente, altos custos para a produção do sistema devido à improdutividade.

No gerenciamento de configuração existem procedimentos que definem como registrar e processar mudanças no sistema. Tais procedimentos também informam como relacioná-las aos componentes do sistema e os métodos usados para identificar diferentes versões e *releases*. O mesmo sistema pode ter diferentes configurações, que podem ser produzidas para diferentes ramificações de múltiplas entregas de produtos tetáveis aos seus usuários. Neste sentido, é possível usar ferramentas de gerenciamento de configuração com intuito de armazenar as versões de componentes do sistema, sistemas construídos com base nesses componentes, e rastrear os construtores das *releases* das demandas de clientes de sistemas que já se encontram em uso.

De modo à prover um desenvolvimento incremental, algumas organizações vem desenvolvendo uma abordagem automatizada para o gerenciamento de configuração: suportar a automação, como de diversos tipos de testes de sistema, construção e implantação de aplicações, em diversas etapas do desenvolvimento de software. Por exemplo, de modo à garantir a qualidade no processo, busca-se automatizar tarefas que permitam executar construções diárias do sistema, por meio de serviços de *build* automático integrado. Estes, que quando integrados com serviço para execução de testes automatizados, aumentam as chances de se encontrar defeitos no sistema em produção.

Estes serviços são comumente programados para execuções periódicas automáticas. Por exemplo, uma vez que ao fim de cada dia de trabalho, os testes deverão ser executados, serviços de *build* são configurados para colocar em execução diferentes *workflows* de *build* e de tetes. Cada *workflow* pode testar diferentes versões do sistema em produção, o que

permite gerenciar a qualidade do software para múltiplas *releases*.

Estes serviços reduzem a probabilidade de um defeito passar despercebido pelo time antes do início do seu trabalho diário. No entanto, apesar destes serviços serem essenciais nas fábricas de software modernas, a configuração destes serviços requer um alto *know how* das equipes de desenvolvimento. Assim, para se obter um uso bem sucedido destas construções diárias, é necessário primeiro de um processo de gerenciamento de mudanças bem definido, e em segundo momento a sua automação em serviços oferecidos por servidores de Integração Contínua (IC).

1.1 Motivação

Com o aumento da concorrência no mercado de desenvolvimento de software, as organizações alocam recursos para desenvolver e entregar software de alta qualidade a um ritmo muito acelerado (PHILLIPS et al., 2015). A integração contínua é uma das práticas da gerência de configuração destinadas a ajudar as organizações a acelerarem o desenvolvimento e entrega de recursos de software sem comprometer a qualidade do produto (HUMBLE; FARLEY, 2010). Assim, é esperado por estas organizações que a adoção das práticas contínuas ofereçam várias vantagens, sendo elas: 1) Obter feedback mais rápido no processo de desenvolvimento de software; 2) Ter implantações frequentes e confiáveis, que levem a uma melhor satisfação do cliente e qualidade do produto e 3) Conexão entre equipes de desenvolvimento e operações fortalecida pela entrega contínua e as tarefas manuais eliminadas (LEPPÄNEN et al., 2015; CHEN, 2015).

Apesar destes benefícios serem um mantra de fábricas de software em geral, adotar práticas contínuas não é uma tarefa trivial. Um das dificuldades recai na falta de sistematização das atividades, uma vez que os processos, práticas e ferramentas organizacionais podem não estar prontos para suportar a natureza altamente complexa e desafiadora de um processo automatizado. Essa dificuldade se observou na disciplina de Resolução de Problemas V de 2018/1, em que apenas um grupo conseguiu configurar corretamente um servidor de Integração Contínua(IC). Esta experiência ressalta e evidencia a dificuldade que é configurar um servidor/serviço de IC, portanto motivando esse trabalho exploratório na direção de se compreender sobre IC e seus servidores/serviços.

1.2 Objetivos

O tema principal para o trabalho de conclusão de curso é "Assistindo a configuração de pipelines de integração contínua em turmas de Resolução de Problemas V". Este trabalho foca em parte da execução de estudos neste tema, focados na execução do trabalho de conclusão de curso I. Assim, para o TCC I, o objetivo principal é aprofundar os conhecimentos sobre a prática de Integração Contínua (IC).

Objetivos específicos são: 1) Identificar os requisitos de gerência de configuração para a iniciação dos alunos da Engenharia de Software em práticas de integração contínua; 2) Caracterizar os desafios de aprendizado de IC para com disciplinas de Resolução de Problemas V; 3) Levantar o conhecimento de um público especifico das disciplinas de RPV, conduzidas entre 2016 e 2019, de modo à identificar suas limitações de conhecimentos para a configuração de pipelines de IC; 4) Identificar como sua experiência anterior à RPV permitiu a configuração de servidores de IC, por meio de uma análise de seu desempenho em projetos de software, e; 5) Analisar três servidores de integração contínua, um estudo exploratório e prático que busca: 5.1) identificar semelhanças e diferenças em configurações para; 5.2) mapeá-las para determinadas necessidades de pipelines de automação, encontradas em blogs e listas de discussões técnicas.

1.3 Contribuições

Esse trabalho apresenta uma análise sobre o cenário de integração contínua no curso de Engenharia de Software e seus desafios por meio de duas contribuições: 1) Um estudo exploratório por meio de mineração de repositórios de software de alunos de RPV, e um sequente *survey* com seus desenvolvedores, com as seguintes contribuições: caracterização do perfil de alunos de RPV, das dificuldades para configurar um servidor de IC, e uma análise de seus conhecimentos anteriores ao uso de IC; 2) Um segundo estudo exploratório por meio analítico-prático de três servidores de IC, resultando num material didático preliminar, que quando refinado serviu para instruir os usuários nas configurações destes servidores ao longo do estudo de avaliação. Tais contribuições nos auxiliam à obter outras contribuições ao longo do TCC II, como um *guideline* para a configuração de servidores de IC e um estudo de mapeamento sistemático.

1.4 Organização

A organização deste trabalho se dá conforme a seguir:

- O capitulo 2 apresenta a metodologia de pesquisa, seguindo uma abordagem para *design science* (PEFFERS et al., 2007);
- O capitulo 3 apresenta em embasamento teórico referente aos conceitos e termos relacionados a esta monografia, incluindo o mapeamento sistemático da literatura da área;
- O capítulo 4 detalha o estudo executado para a derivação do *guideline* proposto, que é avaliado e discutido no Capítulo 5;
- O capitulo 6 apresenta a conclusão desse trabalho e os trabalhos futuros.

2 METODOLOGIA

Este capítulo apresenta a metodologia de pesquisa, seguindo uma abordagem para *design science* (PEFFERS et al., 2007), e caracteriza as atividades executadas e planejadas da Figura 1 em duas fases. Este TCC apresenta os resultados da primeira e segunda fase.

2.1 Protocolo de Design Science

Por se tratar de um estudo que envolve muito protocolos científicos, a metodologia de pesquisa associada ao trabalho de conclusão de curso é embasada em Piffers et al. (PEFFERS et al., 2007). Logo, as macro-atividades da metodologia são discutidas como segue:

2.1.1 Identificar o problema e definir os objetivos da solução

Fase I: A identificação do problema deu-se de modo pragmático, após a execução de uma atividade para configuração de um servidor de integração contínua na disciplina de Resolução de Problemas (RPV) em 2018. Assim, a partir e 22 de dezembro de 2018 alinhou-se um projeto de pesquisa para investigar as dificuldades na configuração destes servidores.

Fase II: Para a identificação do problema na segunda fase, fez-se uso dos estudo aqui apresentados. Por fim, um estudo ad-hoc de literatura foi iniciado, permitindo a descoberta de alguns trabalhos relacionados que devem ser usados como entrada para outro estudo de mapeamento sistemático executado no TCC II.

2.1.2 Projeto e desenvolvimento da solução

Fase I: O projeto e desenvolvimento da solução segue como meta satisfazer os requisitos para representação de pipelines de IC identificados na turma de RPV de 2018, e aprofundados num estudo analítico prático de três opções discutidas no Capítulo 4.3.1. Uma vez que este capítulo apresenta resultados que permitiram que os objetivos específicos do TCC I fossem atingidos, traçou-se novos objetivos futuros como a execução de um mapeamento sistemático e de uma avaliação do *guideline* proposto em uma turma de graduação do curso de Engenharia de Software.

2.1.3 Demonstração conceitual

Fase I: Esta atividade foi iniciada e conta com uma demonstração técnica no Capítulo 4.3.1. No entanto, as demonstrações conceituais do nosso objeto principal competem ao cronograma do TCC II. Nesta atividade foi executado o mapeamento sistemático de literatura a fim de determinar o estado da arte referente a automatização de pipelines de Integração, Entrega e Implantação Contínua.

Fase II: Após a execução do mapeamento sistemático de literatura o próximo passo foi escrever um artigo científico com os resultados deste mapeamento.

2.1.4 Avaliação

Fase I: O TCC I apresentou duas avaliações que caracterizam o problema: No Capítulo A dois estudos complementares identificam a necessidade de desenvolver uma estratégia para a introdução de pipelines de Integração Contínua nas disciplinas de Resolução de Problemas V, enquanto que o Capítulo 4.3.1 demonstra a complexidade para realizar esta tarefa.

Fase II: Já o TCC II avaliou estes resultados a fim de definir uma estratégia para a introdução a configuração de pipelines de Integração Contínua nas disciplinas de Resolução de Problemas V e avaliados por meio de um *survey* dentro do contexto da disciplina de 2021/1 onde os resultados da avaliação foram apresentados no Capítulo 5.

2.1.5 Comunicação

Fase I: A Figura 1 mostra um diagrama de atividades cujos objetos são materiais publicáveis. Assim, uma vez que o conteúdo de ambos os trabalhos estiverem concluídos ambos serão posteriormente submetidos.

Fase II: Remete-se a conclusão de ambos os trabalho e o refinamento contínuo para publicação.

2.2 Atividades Executadas

1. **Atividade 1** - Mineração de repositórios no Bitbucket. Este foi o primeiro estudo conduzido e tinha como objetivo identificar se, de fato, os alunos de RPV de 2018 conseguiram configurar o CircleCI, o servidor de integração contínua adotado pela turma.

2. **Atividade 2** - Leitura de projeto de pesquisa e Tese do orientador. Uma vez que se decidiu por um TCC focado em integração contínua, buscou-se uma aproximação com os temas de pesquisa do orientador. Logo, este trabalho é também associado com um projeto de pesquisa do grupo LESSE, denominado "Fundamentação para a Transferência de Tecnologia no MDE como um Serviço", em que o proponente atua como um volutuário desde de 22 de dezembro de 2018.

3. **Atividade 3** - Análise do servidores de IC. O objetivo foi de identificar a sintaxe de configuração dos servidores Jenkins, Hudson IC e Circle CI, e encontrar as possíveis configurações de melhor caso, caso médio e pior caso que servirão para orientar ao desenvolvimento da estratégia para introdução de pipelines de Integração Contínua em disciplinas de Resolução de Problemas V

4. **Atividade 4** - Desenvolvimento e execução do Survey com alunos de RPV sobre dificuldades para configurar servidores de Integração Contínua em RPV baseando-se nos pipelines do Circle CI.

5. **Atividade 5** - Analisar duas DSLs do grupo de pesquisa que podem ser utilizadas no contexto de pipelines.

6. **Atividade 6** - Busca ad-hoc de artigos sobre estudos tratando de abordagens de Integração, Entrega e Implantação Contínua com o objetivo de identificar trabalhos relacionados.

7. **Atividade 7** - Escrita e Defesa do TCC I, ao foi necessário elaborar uma apresentação para utilizar no dia 28/06/2018.

8. **Atividade 8** - Condução do mapeamento sistemático de literatura. O mapeamento tem como objetivo destacar o estado da arte no tema e fornecer elementos de posição para este trabalho em relação á sua novidade.

9. **Atividade 9** - Escrita de um artigo relatando sobre o Mapeamento Sistemático de Literatura desenvolvido na atividade 8.

10. **Atividade 10** - Execução de um Survey com alunos de RPV do ano de 2021 sobre a Base Necessária para Automação dentro do contexto de Integração, Entrega e Implantação Contínua.

11. **Atividade 11** - Desenvolvimento de uma estrategia para introdução de IC em disciplinas de RPV e escrita de um artigo sobre a mesma.

12. **Atividade 12** - Escrita e Defesa do TCC II.

13. **Atividade 13** - Refinamento do TCC II para os resultados do estudo de caso da aplicação da estratégia na disciplina de Resolução de Problemas V.

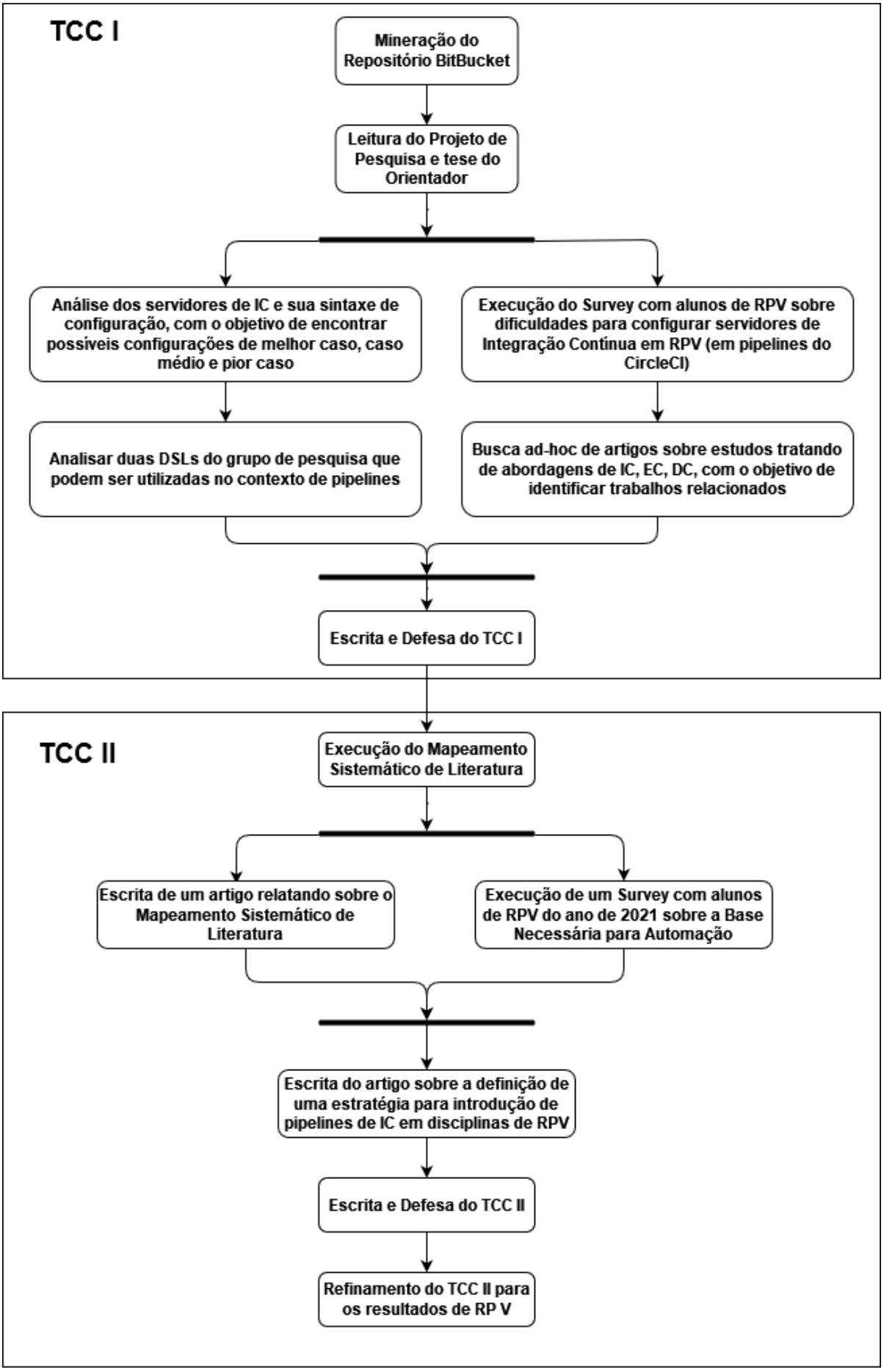

Figura 1 – Atividades divididas em duas fases para o TCC I e TCC II

3 EMBASAMENTO TEÓRICO

3.1 Gerenciamento de Configurações

Segundo (SOMMERVILLE, 2010) O gerenciamento de configuração é o desenvolvimento e uso de padrões e procedimentos para o gerenciamento de sistemas de software em desenvolvimento. Os procedimentos de gerenciamento de configuração definem como registrar e processar mudanças de sistema, como relacioná-las aos componentes do sistema e os métodos usados para identificar diferentes versões dele.

3.2 Gerenciamento de Mudanças

Mudanças ocorrem frequentemente no desenvolvimento de software e o gerenciamento de mudanças é o que garante que essas mudanças serão aplicadas ao sistema de forma controlada com um conjunto de procedimentos de gerenciamento de mudanças apoiados por ferramentas (SOMMERVILLE, 2010). Cada pessoa envolvida no processo de gerenciamento de mudanças é responsável por alguma atividade. Elas completam essas atividade e passam os formulários e itens de configuração associados para os outros. Segundo (SOMMERVILLE, 2010) o procedimento natural desse processo significa que um modelo de processo de mudanças pode ser projeto e integrado com um sistema de gerenciamento de versões.

3.3 Gerenciamento de Versões

O Gerenciamento de Versões envolve o gerenciamento de grande quantidade de informações e garante que as mudanças de sistema sejam registradas e controladas (SOMMERVILLE, 2010). A maioria dos sistemas de gerenciamento de versões fornecem pelo menos um conjunto padrão de funções, sendo elas: 1) Identificação de versões e releases; 2) Gerenciamento de armazenamento; 3) Registro do histórico de mudanças; 4) Desenvolvimento independente e 5) Suporte a projetos.

A primeira funcionalidade consiste em identificar versões gerenciadas quando são enviadas ao sistema. A segunda fornece recursos de gerenciamento de armazenamento de maneira que as versões sejam descritas pelas suas diferenças em relação a uma versão mestre com intuito de reduzir o espaço de armazenamento exigido. A terceira registra e lista toda a evolução do projeto. Assim, alterações sobre arquivos devem ser salvas de forma que seja possível saber o que foi feito, quando foi feito e onde foi feito. A quarta funcionalidade possibilita que vários desenvolvedores trabalhem em paralelo nos mesmos arquivos, sem que um apague/sobrescreva o código do outro. Na quinta funcionalidade fornece suporte a múltiplos projetos/arquivos.

3.4 Gerenciamento de Releases

Uma release de sistema é uma versão do sistema, distribuída para os clientes (SOMMERVILLE, 2010). Os gerentes de releases de sistemas são responsáveis por decidir quando um sistema pode ser liberado para os clientes, gerenciar o processo de criação do release. É papel do gerente decidir o meio de distribuição e documentar o release para assegurar a recriação do sistema exatamente como foi ele foi distribuído. O gerenciamento de releases apresentado em (SOMMERVILLE, 2010) mostra três etapas dentro do Gerenciamento de Releases: A) A primeira é a tomada de decisão para uma release onde uma release é preparada. B) A segunda etapa é a criação da release que é um processo de criação de arquivos e documentos que inclui todos os componente do release do sistema. C) A terceira é a documentação de releases onde após a release do sistema ter sido produzida, ela deve ser documentada de forma que deve-se registrar as versões específicas dos componentes de código-fonte usados para criar o código executável.

3.5 Construção de Sistemas

De acordo com (SOMMERVILLE, 2010), a construção de sistemas é um processo de compilação e ligação dos componentes de software em um programa que executa determinada configuração. Em algumas linguagens, um script de configuração é criado automaticamente, enquanto em outras faz-se necessário a criação manual, o que pode levar ao erro humano. O script fornece as dependências entre componentes e informações para que o sistema possa definir e decidir quando o código-fonte dos componentes deve ser recompilado e quanto o código-objetivo atual pode ser reusado.

3.6 Integração Contínua

De acordo com (FOWLER, 2006), Integração Contínua(IC) é uma prática de desenvolvimento de software onde os membros de um time integram seu trabalho frequentemente. Geralmente, cada pessoa integra código pelo menos diariamente, podendo haver múltiplas integrações por dia. Cada integração é verificada por um *build* automatizado, que inclui na sequência a execução de testes para detectar erros de integração .

3.6.1 Implantação Contínua

Implantação Contínua é uma atividade sequente à integração contínua. Com o objetivo de minimizar o tempo de espera, i.e., o tempo decorrido entre o desenvolvimento de uma nova linha de código e o novo código sendo usado por usuários ao vivo, a implantação contínua permite alocar automaticamente o sistema para um estágio de produção (CONECTING, 2017).

3.6.2 Entrega Contínua

Entrega Contínua (EC) é a capacidade de obter, em estágio de produção, mudanças de todos os tipos, incluindo novas funcionalidades, alterações de configuração e correções de bugs, de forma segura e rápida (FARLEY, 2007). Dessa forma, é garantido que o código esteja sempre em um estado implantável, mesmo diante de equipes de milhares de desenvolvedores fazendo alterações diariamente nos código fontes do sistema.

3.7 Trabalhos Relacionados

Para a busca dos trabalhos relacionados foi feita uma busca *ad-hoc* na literatura. Assim, considerou-se como trabalhos relacionados os estudos que, de alguma forma, falavam sobre Integração, Entrega ou Implantação Contínua. Para o TCC II, espera-se extrair termos comuns destes trabalhos e planejar um protocolo de mapeamento sistemático sobre o assunto.

O trabalho de (CLEMENCIC; COUTURIER, 2015) fala sobre a implementação de uma linguagem específica de domínio para configurar e executar a integração contínua do LHCb (*Large Hadron Collider beauty*). Trata-se de um estudo aplicado ao desenvolvimento de software científico para a Física usando integração contínua. Logo, LHCb é experimento desenvolvido para fazer medidas precisas da violação da simetria CP e decaimentos raros de mesões com o quark b.

O trabalho de (O'CONNOR; ELGER; CLARKE, 2017) fala sobre uma abordagem emergente para o desenvolvimento de software: a Engenharia de Software Contínua (CSE em inglês). Essa abordagem permite que um software operacional possa ser entregue com muita frequência. Essa abordagem faz uso de um conjunto complexo de ferramentas independentes, que juntas reduzem o tempo de espera no fornecimento de software comercial. Porém, os autores relataram também que essa abordagem pode não ser apropriada para todas as formas de desenvolvimento de software, como por exemplo em casos onde um número elevado de funcionários introduz mudanças no ambiente de desenvolvimento e implantação, nesse caso será necessário introduzir mecanismos para evitar conflitos na construção e implantação de forma a evitar falhas.

Tratando-se de estudos experimentais, o trabalho de (LAUKKANEN; ITKONEN; LASSENIUS, 2017) avaliou os problemas enfrentados ao adotar a entrega contínua, por meio uma revisão sistemática de literatura. Além disso, os autores identificaram causas e soluções para os problemas encontrados. Os autores dividiram os problemas em sete temas sendo eles: 1) Construção; 2) Sistema; 3) Integração; 4) Teste; 5) Versão; 6) Humano e Organizacional e 7) Recursos. No primeiro tema foram abordados problemas causados por decisões referentes a construção do sistema(*Build*). No segundo foram abordados problemas causados por decisões de projeto. No terceiro foram abordados problemas que surgem quando o código-fonte é integrado à linha principal do desenvolvimento de

software. O quarto tema foi relacionados aos problemas decorrentes no teste de software. O quinto refere-se aos problemas relacionados aos problemas decorrentes após a entrega da versão de implantação do sistema. O problemas do sexto tema não estão relacionados ao desenvolvimento especificamente e sim aos aspectos humanos e organizacionais na adoção da entrega contínua. O sétimo e último tema está relacionado aos recursos disponíveis dentro da organização, já que entrega contínua consome muitos recursos. Nesse trabalho também se relatou que, ao adotar a entrega contínua, esses problemas são comuns, críticos e pouco estudados.

Outro trabalho empírico é apresentado por (ZHAO et al., 2017a), que caracteriza a inovação promovida pela Integração Contínua e sobre sua a necessidade de adaptação das práticas existentes para aproveitar ao máximo o seu potencial e as "melhores práticas". Dentro desse contexto, o trabalho se foca em um estudo sobre o impacto da adoção da TRAVIS CI, nas práticas de desenvolvimento baseadas numa coleção de projetos GitHub, além de identificar os desenvolvedores responsáveis pela introdução do TRAVIS nesses projetos.

O trabalho de (KONAT et al., 2018) apresentou a PIE, uma Linguagem Específica de Domínio(DSL em inglês) para o desenvolvimento interativo de pipelines. Esta processam eventos em tempo real para o desenvolvimento de software. Segundo os autores o PIE fornece um modelo de programação simples que permite uma representação direta e concisa de pipelines sem informações desnecessárias, reduzindo assim o esforço na criação e manutenção de pipelines. Os programas dos pipelines compilados podem ser inseridos em ambientes interativos, como editores de código-fonte e IDEs, permitindo assim um feedback oportuno sem um alto custo.

O trabalho de (SHAHIN; BABAR; ZHU, 2017) apresentou uma revisão sistemática de literatura para elencar o estado da arte de práticas contínuas(Integração Contínua, Entrega Contínua e Implantação Contínua). Nessa revisão foram identificadas trinta abordagens e ferramentas que facilitam a implementação de práticas contínuas(Integração Contínua, Entrega Contínua e Implantação Contínua). Dos trabalhos abordados a maioria foram pesquisas de validação (34,7%) e avaliação (36,2%). Assim, classificaram abordagens e ferramentas, identificando desafios e práticas neste contexto, bem como lacunas para pesquisas futuras. Segundo os autores pode-se destacar os seguintes tópicos de melhoria para a pesquisa como em aberto: 1) Melhorar a captura e o relato de informações contextuais nos estudos que relatam diferentes aspectos das práticas contínuas; 2) Obter uma compreensão profunda de como os sistemas de software intensivo devem ser re-arquitetados para suportar práticas contínuas e 3) Abordar a falta de conhecimento e ferramentas para projetar processos de engenharia de concepção e executar pipelines.

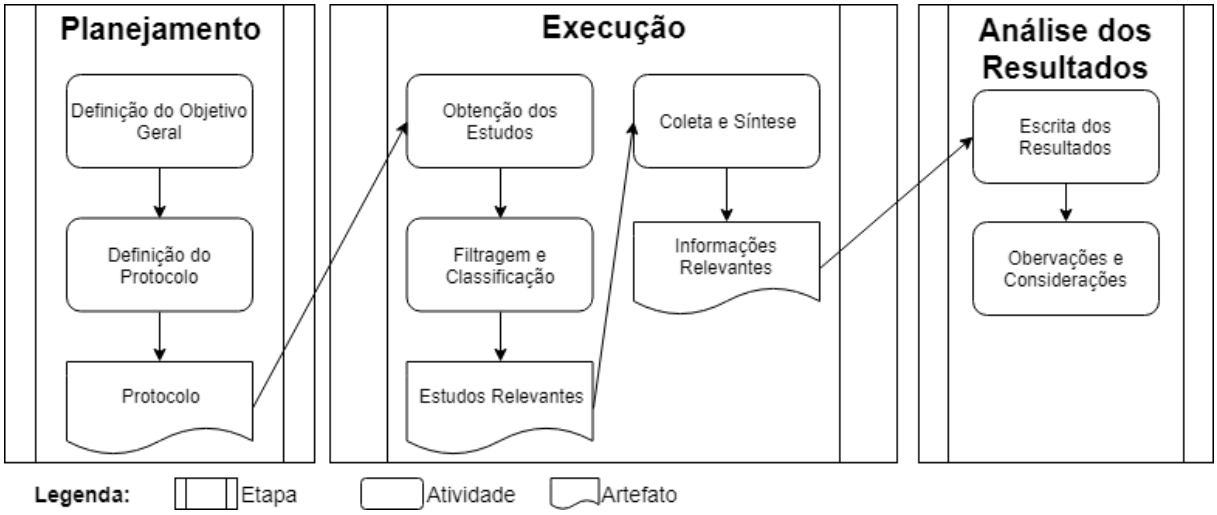

Figura 2 – Processo de mapeamento utilizado

3.8 Mapeamento sistemático de literatura

3.8.1 Processo de Mapeamento Sistemático

Para a execução da revisão sistemática de literatura é importante ter um processo bem definido. Para este trabalho, foi usado uma adaptação do processo apresentado por (NAKAGAWA et al., 2017), que pode ser visto na Figura 2. As atividades realizadas foram agrupadas em três grupos, sendo eles: planejamento, condução e análise dos resultados.

Na etapa de planejamento, definiu-se o objetivo geral da revisão (seção 3.8.1.1) e o protocolo do mapeamento, contendo informações sobre questões de pesquisa (seção 3.8.1.2), as estratégia para conduzir o mapeamento (seção 3.8.1.3), critérios de aceitação, exclusão (seção 3.8.1.4) e qualidade (seção 3.8.1.5), como os dados foram extraídos e apresentados (seção 3.8.1.7). Na etapa de condução os estudos relacionados foram identificados e aplicados os critérios de aceitação e exclusão, os estudos foram classificados utilizando os critérios de qualidade (seção 3.8.3) e por fim foi realizada a coleta e síntese das informações apresentadas nos trabalhos relacionados. Na etapa de análise dos resultados, as informações mais relevantes obtidas durante a condução foram escritas neste artigo, a fim de responder as questões de pesquisa (seção 3.8.4).

3.8.1.1 Escopo e Objetivo

Neste trabalho foi conduzido um mapeamento com intuito de identificar técnicas, metologias, métodos, abordagem ou processos para configuração de pipelines de Integração Contínua, Entrega Contínua e Implantação Contínua em contexto reais da indústria ou casos simulados bem como suas vantagens, desvantagens e dificuldades.

3.8.1.2 Questões de Pesquisa

As questões de pesquisa (QP) definidas neste mapeamento são:

QP01: Quais os desafios de pesquisa na área de gerenciamento de configuração?

QP02: Quais foram seus pontos positivos e negativos observados no estudo?

QP03: Quais são os contextos das empresas ou organizações que utilizam IC/DC/EC?

QP04: Quais estudos apresentam ferramentas de suporte para a configuração automatizada de pipelines de Integração Contínua (IC), Deploy Contínuo (DC) e Entrega Contínua (EC) e quais são as ferramentas utilizadas?

3.8.1.3 Processo de Pesquisa

Neste mapeamento, foi adotada a estratégia automatizada para busca de trabalhos relacionados utilizando seis repositórios, sendo eles *IEEE Xplorer*, *ACM Digital Library*, *Scopus*, *Engineering Village*, *Springer Link* e *ScienceDirect*.

Para a definição da string de busca, foram utilizados as palavras e sinônimos apresentados na Tabela 1, juntamente com os operadores lógicos *OR* entre os sinônimos e *AND* entre os termos. Para a definição desta string foram realizados diversos estudos piloto, com a finalidade averiguar se a string de busca retornava estudos relacionados ao tema de interesse, utilizando o motor de busca Scopus. Nestes estudos, aplicou-se os critérios de inclusão CI01 e CI02 (veja 3.8.1.4) atrávez da leitura do *abstract*. Desta forma, quando um novo sinônimo para algum dos termos foi encontrado o mesmo foi adicionado na string de busca e em casos onde nenhum trabalho era perdido ao remover o termo os trabalhos que não seria apresentados sem o termo seriam averiguados individualmente e caso nenhum trabalho fosse válido o mesmo era marcado para remoção. Os termos marcados para remoção eram validado em outras bases e caso não houvesse nenhum válido o mesmo era removido resultando na string final da Tabela 2

3.8.1.4 Critérios de Inclusão e Exclusão

A definição de critérios de inclusão (CI) e exclusão (CE) durante o planejamento do mapeamento é importante para filtrar os artigos obtidos através das buscas nas bases de dados. Estes critérios ajudam a selecionar apenas artigos relevantes, pois filtram os mesmos com base no seu conteúdo. Neste mapeamento, para um artigo ser aceito o mesmo deve contemplar ao menos um CI. Caso o artigo contemple um CE, o mesmo é excluído do mapeamento, mesmo que atenda a um ou mais CI.

Os seguintes critérios de inclusão foram definidos:

- **CI01** - O estudo apresenta uma técnica/metodologia/método/ferramenta/processo/relato para a definição e implantação de pipelines de Integração Contínua, Implantação Contínua ou Entrega Contínua.

Termo	Sinônimo
Automated	Adaptative
	Automated
	Automatic
Continuous Integration	CI
	CircleCI
	Continuous Integration
	Hudson
	Hudson CI
	Jenkins
Generation	Construction
	Creation
	Generation
	Production
Pipelines	Configuration Files
	Files
	Pipelines

Tabela 1 – Termos utilizados na string de busca

String de Busca
("continuous integration"OR "continuous deployment"OR "continuous delivery" OR "software delivery"OR "continuous release") AND ("pipeline"OR "integration process"OR "CI process"OR "deployment process") AND ("CD process"OR "delivery process"OR "release process"OR "build process")

Tabela 2 – String de busca final

Já os critérios de exclusão definidos foram:

- **CE01** - Estudo que não provê acesso ao seu conteúdo na íntegra.
- **CE02** - Estudos do tipo editoriais, artigos de posição, revisões, tutoriais, demonstrações e resumos de *proceedings*.
- **CE03** - O estudo é de uma área não relacionada à software/sistema.
- **CE04** - O estudo não está disponível para download.
- **CE05** - O estudo não foi escrito na língua inglesa.
- **CE06** - O estudo não se enquadra no critério de inclusão.
- **CE07** - O estudo possui cinco ou menos páginas.

3.8.1.5 Critérios de Qualidade

Neste trabalho a fim de quantificar a qualidades dos trabalhos previamente selecionados foram definidos quatro critérios de qualidade. Para cada critérios de qualidade

existem de três a cinco opções com valores unitários de 0 a 2 sendo assim as pontuações mínima e máximas são respectivamente 1 e 8.

- **QA01** - Qual a relevância do estudo em termos de novidade ferramental para a área?
  - **QA01 A** - O estudo apresenta um novo suporte ferramental - 2.0
  - **QA01 B** - O estudo apresenta a integração de um suporte ferramental existente -1.5
  - **QA01 C** - O estudo apresenta um incremento para um suporte ferramental existente - 1.0
  - **QA01 D** - O estudo utiliza de um suporte ferramental existente - 0.5
  - **QA01 E** - O estudo não utiliza suporte ferramental - 0.0
- **QA02** - Qual a relevância do estudo para se entender como a proposta é aceita na prática?
  - **QA02 A** - O estudo apresenta um relato completo de aplicabilidade num cenário real. - 2.0
  - **QA02 B** - O estudo apresenta um relato superficial de aplicabilidade num cenário real. - 1.5
  - **QA02 C** - O estudo apresenta um relato completo de aplicabilidade num cenário ilustrativo. - 1.0
  - **QA02 D** - O estudo apresenta um relato superficial de aplicabilidade num cenário ilustrativo. - 0.5
  - **QA02 E** - O estudo não apresenta um relato de aplicabilidade. - 0.0
- **QA03** - Qual é a qualidade do estudo de avaliação da proposta?
  - **QA03 A** - O estudo é do tipo evaluation research (Controlled Experiment, Quasi-Experiment, Real-world case study, Real-world action research). - 2.0
  - **QA03 B** - O estudo é do tipo validation research (Analytical study, Survey, Simulation, Focus group). - 1.0
  - **QA03 C** - O estudo é do tipo solution proposal (Proof of concept, Conceptual demonstration). - 0.5
  - **QA03 D** - O estudo de avaliação inexiste (um pequeno relato, um position paper, artigo de opinião)). - 0.0
- **QA04** - Qual é a relevância do estudo para a definição de um pipeline completo de implantação contínua?

- **QA04 A** - O estudo apresenta um pipeline completo de Implantação Contínua. - 2.0
- **QA04 B** - O estudo apresenta um pipeline de Entrega Contínua. - 1.5
- **QA04 C** - O estudo apresenta um pipeline de Integração Contínua. - 1.0

3.8.1.6 Processo de Seleção

A seleção dos artigos de interesse para o mapeamento foi realizada por todos os autores do estudo. Este processo de seleção foi realizado em 5 etapas: (1) obtenção dos artigos nas bases de dados, (2) eliminação de duplicatas, (3) seleção inicial, (4) aplicação de critérios de inclusão e exclusão, (5) aplicação dos critérios de qualidade e (6) sincronização dos resultados. Após a realização destas etapas obteve-se a lista de artigos selecionados.

Obtenção dos artigos nas bases de dados: trabalho em conjunto onde os pesquisadores geraram strings específicas para cada base de dados com base na string padrão. Foi realizada a busca nestas bases e os estudos obtidos foram compilados.

Eliminação de duplicatas: ainda em conjunto, os pesquisadores procuraram e removeram artigos duplicados na lista gerada na etapa anterior. Como resultado, foi criada uma lista livre de duplicatas. Cada pesquisador criou uma cópia desta lista para utilizar nas etapas a seguir.

Seleção inicial: cada pesquisador, agora individualmente, realizou uma seleção inicial dos artigos, atreves da leitura do título e *abstract* (quando necessário ocorreu a leitura da introdução e conclusão também) para remover artigos que fugissem do contexto de interesse. Também, foram aplicados os CE (Critérios de Exclusão) quando implícitos.

Aplicação dos critérios de inclusão e exclusão: ainda individualmente, foi feita a leitura (parcial ou completa) dos artigos que restaram, aplicando quando cabível todos os critérios de inclusão e exclusão. Assim, foram definidos os artigos que fariam parte do mapeamento.

Aplicação dos critérios de qualidade: individualmente, os pesquisadores realizaram a leitura completa dos artigos aceitos para classificá-los de acordo com os CQ.

Sincronização dos resultados: por fim, os pesquisadores comparam os resultados da execução das etapas anteriores para verificar se a avaliação realizada pelos mesmos foi similar. Em caso de divergência, onde um ou mais artigos foram selecionados por um pesquisador ou algum critério foi selecionado por um pesquisador, porém não pelo outro, foi realizada uma avaliação e discussão em conjunto para reaplicar os critérios de inclusão, exclusão e de qualidade de forma a determinar um resultado.

3.8.1.7 Estratégia de Extração de Dados

Para extrair as informações relevantes para este mapeamento foi utilizado um formulário com os seguintes itens, que ajudaram a responder as questões de pesquisa e

verificar os critérios de qualidade:

- Título
- Autor(es)
- Ano
- Em que conferência/revista foi publicado?
- Qual é o tipo de publicação?
- Observações gerais
- Qual o contexto/domínio de aplicação do estudo?
- Quais são as etapas principais do pipeline proposto/utilizado no estudo? Informe as etapas na sequência correta e indique entre parênteses, para cada uma, se é automática ou manual.
- Apresenta incremento ferramental para:
- Quais são as ferramentas utilizadas no estudo e qual o propósito específico de cada uma? (ex: configuração de pipeline, controle de versão, análise do código, build, servidor CI, teste, servidor CD, etc.)
- Licenças de uso
- Observações sobre melhores práticas
- Método de avaliação
- Comente se e como é utilizada uma arquitetura de referência e/ou estilo arquitetural e/ou padrão arquitetural
- Pontos positivos observados no estudo
- Pontos negativos observados no estudo
- Desafios apontados pelo estudo
- Informações do estudo sobre o retorno de investimento
- O que os estudos apresentam sobre a temática de Engenharia de Software Contínua
- Quais são os desafios de pesquisa na configuração de pipelines de IC/DC/EC?

3.8.2 Resultados do Estudo de Mapeamento Sistemático

Estudo	**Descrição do estudo realizado**
S01	No trabalho *Closing the feedback loop between UX design, software development, security engineering, and operations*(NGUYEN; DUPUIS, 2019) não é apresentada uma configuração de Integração, Entrega ou Implantação Contínua apenas é relatado que serão usadas práticas contínuas no modelo de desenvolvimento de software proposto para economizar tempo e continuar com qualidade garantida principalmente em UX design com testes de aceitação.
S02	No trabalho *Enabling devops collaboration and continuous delivery using diverse application environments*(WETTINGER; ANDRIKOPOULOS; LEYMANN, 2015) é proposto um conceito de requisitos de um ambiente de aplicação com requisitos não funcionais específicos que foram mapeados entre topologia de aplicação e requisitos do ambiente de aplicação. Para a resolução destes requisitos foi proposto o uso de uma base de conhecimento em combinação com um processo de resolução com tarefas distintas.
S03	O trabalho *Introducing a deployment pipeline for continuous delivery in a software architecture course*(GREISING; BARTEL; HAGEL, 2018) faz uma introdução ao conceito de Entrega Contínua executado dentro de sala de aula onde foi apresentado aos alunos o que era pipeline de Integração e Entrega Contínua e seus passos, no caso aplicado haviam alunos que possuíam conhecimento distintos sobre programação e testes de software porém ao final foi feita uma reavaliação e os alunos que não sabiam sobre o tema tiveram uma evolução no conhecimento e os alunos que já tinham conhecimento não sabiam o tanto que pensavam que sabiam.
S04	O trabalho *Modern Release Engineering in a Nutshell - Why Researchers Should Care*(ADAMS; MCINTOSH, 2016) apresenta os conceitos de Engenharia de Liberação (*Release engineering*) onde algumas de suas áreas se remetem a Integração Contínua e Entrega Contínua, apesar de não definido especificamente quais etapas o pipeline tem e quais são os reais benefícios da aplicabilidade desses conceitos em um cenário real.

S05	O trabalho *Towards a deployment system for cloud applications*(LUO; YE; ZHANG, 2015) faz menção sobre a implantação de aplicativos distribuídos em nuvem, propõe um modelo baseado em componentes que visa a configuração e implantação automatizadas em um contêiner leve. Baseado nesse modelo proposto derivam um sistema de gerenciamento de aplicativos e um sistema de implantação que poderia processar as dependências e o relacionamento interconectado dos componentes automaticamente.
S06	O trabalho *DevOps for containerized applications*(BIENER; CRAWFORD, 2019) não faz uma boa explicação sobre o tema, mas faz a apresentação de DevOps para aplicação de contêiner , explicando as ferramentas que se pode utilizar. Falta um aprofundamento maior sobre a área de pesquisa, mas ele é um artigo bom que da uma ideia sobre o assunto de DevOps ao leitor.
S07	No trabalho *Security support in continuous deployment pipeline*(ULLAH et al., 2017) é relatada a preocupação com segurança na implementação de um novo pipeline de implantação contínua, tendo em vista os diversos riscos para a segurança que podem surgir no desenvolvimento do mesmo.
S08	O trabalho *ICDO: Integrated cloud-based development tool for DevOps*(AHMADIGHOHANDIZI; SYSTÄ, 2015) relata a criação de um protótipo que se trata de um junção de várias ferramentas de DevOps para implantação em servidores fazendo com que o desenvolvedor use uma única ferramenta até a implantação do produto. Os desenvolvedores também conseguem gerenciar aplicativo já implantados e economizar tempo com a automação.
S09	No trabalho *Designing an android continuous delivery pipeline*(ZACHOW, 2016) é apresentado um estudo de caso sobre um sistema de entrega para auxílio em Integração Contínua visando o uso de tecnologias que a equipe do projeto já conhecia dentro de uma arquitetura de microsserviços onde relatam a dificuldade de implementar Integração Contínua e Entrega Contínua por conta do esforço necessário para tal além de sugerir a adoção de prática ágeis em conjunto com práticas contínuas.

S10	No trabalho *Continuous Testing in the Development of IoT Applications*(GUşEILă; BRATU; MORARU, 2019) é apresentado um processo, método e ferramenta para introdução da automação de integração, implantação e testes contínuos em um projeto sugerindo também adotar que seja um processo ágil de desenvolvimento de software de aplicativos **IoT**(Internet das Coisas). No trabalho é relatado ser vantajoso a adoção de integração contínua e entrega contínua em conjunto com práticas ágeis para lidar com requisitos em constante mudanças no desnevolvimento de software de aplicativos **IoT**(Internet das Coisas).
S11	No trabalho *From the edge to the cloud: A continuous delivery and preparation model for processing big IoT data*(SáNCHEZ-GALLEGOS et al., 2020) é apresentado um modelo implementado como uma estrutura que permite às organizações construir grandes soluções de dados flexíveis de IoT para processamento de dados, de maneira ininterrupta, da borda à névoa, à nuvem aos dispositivos dos usuários finais, e tornando viável fornecer essas soluções com esquemas de segurança, eficiência e confiabilidade de maneira econômica.
S12	O trabalho *Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible*(MYSARI; BEJGAM, 2020) relata como funciona o Jenkins Ansible falando sobre os passos de instalação, como funciona, como um usuário novo entra , fala também que o pipeline com Jenkins e mostrando suas fases , comenta também que existe uma falta de automação entre *delay* e testes, que se perde muito tempo, mas essas estrategias de de IC/DC ajudam a melhorar isso.

S13	No trabalho *Improving the delivery cycle: A multiple-case study of the toolchains in Finnish software intensive enterprises*(MäKINEN et al., 2016) é relatado um estudo de entrevista realizado em organizações finlandesas de desenvolvimento de software onde foi constatado a proeminência de ferramentas em diferentes estágios do desenvolvimento. Nestes estágios algumas destas ferramentas poderiam ser consideradas mais importantes em termos de implantação contínua do que outras, logo sendo mais importantes para esta etapa de implantação. Após uma análise executada no trabalho os pesquisadores constataram que uma infraestrutura de ferramentas bem definida suporta o envio de lançamentos de software continuamente. Adotando ferramentas que vão desde a facilitação do gerenciamento de requisitos, facilitando as atividades diárias de desenvolvimento, garantindo a qualidade, agilizando a implantação, até o tratamento rápido de feedback do usuário, melhorando o ciclo de entrega. Sendo assim os pesquisadores relataram que organizações que empregam essas ferramentas na prática podem obter um impulso extra para o desenvolvimento de software entregando o produto mais rápido.
S14	O trabalho *Test framework for Jenkins shared libraries*(ANTUNES; NAVARRO; HANAZUMI, 2018) apresenta uma implementação de um *framework* de testes para Jenkins, ou seja, neste trabalho ajuda a facilitar a criação de testes voltados para Integração Contínua e Entrega Contínua nas empresas. O trabalho faz uma boa divisão dos assuntos a serem a apresentados, mas em momento algum é apresentada uma versão para download do *framework* somente são apresentadas *screenshots* do código do Jenkins.
S15	No trabalho *Implementation of a DevOps pipeline for serverless applications*(IVANOV; SMOLANDER, 2018) foram relatadas melhores práticas para adoção de Integração Contínua e Entrega Contínua em um contexto de *serverless* em um cenário fictício onde relatam facilidade na adoção de Integração Contínua e Entrega Contínua diferente dos outros trabalhos anteriormente relatados e propõem vinte e seis boas práticas para adoção de **CI** e **CD**.

S16	No trabalho *Continuous Delivery: Overcoming adoption challenges*(CHEN, 2017) foram relatados os principais desafios da adoção de Entrega Contínua em uma organização com faturamento anual de aproximadamente sete bilhões de euros chamada Paddy Power a qual teve como principal desafio a adesão das esquipes de software as práticas de Entrega Contínua sendo quase resolvidas ao apresentar as vantagens que os desenvolvedores teriam ao aceitar essas mudanças. Outro ponto interessante foi relatar que das mais de seiscentas aplicações a qual a organização trabalhava a aplicação de Entrega Contínua era bem mais eficiente nas aplicações que geravam mais renda a organização de forma a justificar os gastos e ter mais lucro que benefícios.
S17	No trabalho *Dyn Tail - Dynamically Tailored Deployment Engines for Cloud Applications*(WETTINGER; BREITENBüCHER; LEYMANN, 2015) foi relatado a construção do *DYNTAIL framework* para automatização da implantação de *releases* em ambiente de produção fazendo uso de serviços cloud e uma arquitetura de microsserviços de forma a otimizar o tempo de entrega. Foi relatada uma melhor no tempo de implantação e ressaltado como problema a implantação de múltiplos serviços em múltiplos ambientes. No trabalho não houve aplicações em ambiente real de trabalho.
S18	O trabalho *Building lean continuous integration and delivery pipelines by applying devops principles: A case study at varidesk*(DEBROY; MILLER; BRIMBLE, 2018) comenta sobre os problemas na adoção de CI e CD nas empresas, pois quando foi escrito o artigo existiam poucas pesquisas na área oque dificultava muito a adoção destas praticas nas empresas, mas essas dificuldades podem ser superadas aplicando o princípios de DevOps. Mas o artigo reforça que que deve-se estudar mais e eles querem que essa pratica de CI e CD seja aplicadas na industria e na academia.
S19	O trabalho *Vulnerabilities in Continuous Delivery Pipelines? A Case Study*(PAULE; DüLLMANN; HOORN, 2019) apresenta as diversas vulnerabilidades presentes em um pipelines já existente de Entrega Contínua e executa diversas verificações a intuito de encontrar, mapear as vulnerabilidades presentes no pipelines da organização a fim de definir melhores práticas e corrigir as vulnerabilidades encontradas. Uma das principais sugestões foi a definição de permissões para cada usuário e agente, além da conteinerização de cada execução de pipeline.

S20	No trabalho *Designing a next-generation continuous software delivery system: Concepts and architecture* (STEFFENS; LICHTER; DöRING, 2018) é apresentada uma solução para implementação de integração contínua e entrega contínua dentro de uma organização pequena usando tecnologias que a equipe já conheça para evitar a necessidade de despender tempo para aprendizagem de muitas novas tecnologias economizando no tempo requerido e consequentemente no custo. A solução foi implementada em protótipo chamado JARVIS e validada por um estudo de caso que mostrou resultados promissores para os pesquisadores que redigiram o trabalho como melhorar o feedback rápido na decorrência de erros, resultando em uma melhora no desempenho da equipe. O trabalho também recomenda o uso de práticas contínuas em conjunto com desenvolvimento ágil.
S21	O trabalho *Continuous integration and continuous delivery pipeline automation for agile software project management*(ARACHCHI; PERERA, 2018) comenta do surgimento de CI e CD até sua evolução nos dias atuais e fala que a fusão das duas praticas juntas é um grande avançando para área, pois conseguimos juntas os seus benefícios em ao logo de toda execução de um projeto.
S22	O trabalho *Performance monitoring in SAP HANA's continuous integration process*(REHMANN et al., 2016) relata a criação de uma abordagem de Integração Contínua para o sistema de banco de dados relacional HANA DB visando aplicar o quanto antes os testes de desempenho e identificar anomalias. A abordagem tem como foco os testes de desempenho o quanto antes no ciclo de desenvolvimento de software a fim de identificar o mais cedo possível anomalias no desempenho do banco de dados já que é uma de suas principais qualidades. Foi relatado que houve uma redução no tempo de entrega de novas mudanças sem perder a garantia de qualidade, apesar de não mencionado o quanto melhorou.

S23	O artigo *Continuous and Integrated Software Development using DevOps*(AGARWAL; GUPTA; CHOUDHURY, 2018) traz uma proposta de mudar a forma que os testes são executados em deixar de se basear nos atores, mas se basear nas interações do usuário com o sistema. O trabalho faz o uso de ferramentas como Selenium e relata uma carência nas ferramentas de testes, pois atualmente elas só testam as aplicações, mas não fazem testes. É comentando que devemos ficar longe de ramificações de longa duração, que existe uma falta de experiência e habilidades dentro do contexto de Integração Contínua, falta de testes de aceitação e automatização dos processos.
S24	O trabalho *Towards Cloud Native Continuous Delivery: An Industrial Experience Report*(HäKLI; TAIBI; SYSTA, 2018) apresenta um novo suporte ferramental para implementação de Integração Contínua e Entrega Contínua em Nuvem em um cenário real da indústria usando **AWS** (*Amazon Web Services*) onde relata como vantagem agregada ao cliente as entregas contínuas de produto/mudanças que podem ser continuamente avaliadas pelo cliente em troca de um custo maior apesar de não informado qual seria esse custo.
S25	O trabalho *Perceived Benefits of Adopting Continuous Delivery Practices*(ITKONEN et al., 2016), foi feito em um ambiente real, explica que deve-se ter uma boa comunicação entre o time de desenvolvimento e os clientes e também que os times de desenvolvimento dentro da mesma empresa(time) também devem ter uma boa comunicação para não evitar problemas, por exemplo alguém do time pode ficar parado, pois não ocorreu a comunicação de passar tarefa, mas não foi mostrado como é o pipeline, mas foi mostrada a evolução da empresa com a implementação de CD. O trabalho relata como é bom ter uma comunicação entre dos desenvolvedores e os clientes e entre outros desenvolvedores dentro da mesma empresa e time(por exemplo os times de desenvolvimento e testes se comunicarem para não ter problemas e terem que ficar parado um tempo). O trabalho também relata que a adoção de CD nas empresas é muito útil, mas ainda está no inicio isso.

S26	No trabalho *Automated testing in the continuous delivery pipeline: A case study of an online company*(GMEINER; RAMLER; HASLINGER, 2015) é relatado sobre a a aplicação de um pipeline de Entrega Contínua em uma empresa de desenvolvimento de softwares para web, relatam no início usarem cascata e conforme o crescimento da empresa e a volatilidade dos requisitos foi iniciado a implementação do pipeline de Entrega Contínua a fim de preparar os recursos do sistema de software para o teste de aceitação do usuário com a maior qualidade possível. Foram relatados ganhos na aplicação da Entrega Contínua que superam o tempo gasto e ressaltado mesmo havendo uma etapa específica para testes de aceitação do cliente outros testes ainda são necessários anteriormente a essa etapa, além disso relatam a necessidade de futuramente adicionarem testes voltados a segurança e aumento na cobertura dos testes unitários.
S27	O trabalho *A Pilot Study on Introducing Continuous Integration and Delivery into Undergraduate Software Engineering Courses*(EDDY et al., 2017) relata um estudo feito em uma curso de engenheira de software com o intuído de ensinar eles sobre pipeline, **EC**, **IC**, pois é comentado que nos anos anteriores da graduação é muito focado em programação, mas não nesta parte de entregas continuas e os alunos não entendem muito bem como é esse funcionamento, ao decorrer do trabalho é são nos apresentados os conceitos de **EC** e **IC**, pipelines e suas fases, mas também são apresentadas perguntas que foram respondidas ao decorrer do trabalho, fazendo comparações do que os alunos sabiam antes e depois dos estudos, os alunos não tiveram que desenvolver uma aplicação do zero, mas sim foi fornecida uma aplicação onde foi informado que pontos do código deveriam ser modificadas.

Tabela 3 – Breve descrição dos artigos e seus focos de estudos

3.8.3 Condução do Mapeamento

A condução do mapeamento ocorreu entre os dias 22 de fevereiro de 2021 e 28 de agosto de 2021 e se deu conforme as etapas apresentadas na seção 3.8.1.6.

3.8.3.1 Pesquisa nas Bases de Dados

Algumas bases de dados permitem adicionar filtros adicionais para as pesquisas. Para a base de dados *Scopus*, a busca foi limitada para apenas o título, resumo e palavras-

Estudo	Descrição da métrica de avaliação do estudo realizado
S03	O trabalho lista como métrica o Tempo, relata que grandes empresas estão começando a usar entrega continua, pois se ganha tempo e consegue competir no mercado com mais efetividade e menor custo.
S07	Foi usado o *Goal Structuring Notation* (GSN) para avaliação qualitativa da eficácia das táticas de segurança propostas. Para o teste Quantitativo foram usadas duas ferramentas de varredura sendo elas o *Qualys OWASP Scan e OWASP Zed Attack Proxy (ZAP).*
S11	Nos estudos de caso realizados nas duas fases desta avaliação experimental foram considerados o tempo de resposta, o tempo de serviço e o throughput métricas para medir o desempenho de cada configuração e solução implementada nos protótipos e/ou no simulador.
S13	No trabalho foi quantificado o tempo necessário para o Ciclo real de entrega de software, o Ciclo de entrega real e o tempo necessário para o software mínimo implantável. O Ciclo real de entrega software é um ciclo real durante o qual a organização de desenvolvimento produz um artefato que poderia ser, em princípio, liberado para a utilização. O Ciclo de entrega real é a frequência com que o software é realmente lançado. E o Tempo para o software mínimo implantável é o tempo para o o software mínimo implantável sem considerar o tempo de implantação.
S15	Monitorar o pipeline de design e implementação
S21	Neste trabalho, latência e a taxa de transferência(*throughput*) são usadas como métricas no benchmarking, enquanto a carga do sistema é aumentada.
S27	Existem limitações neste campo, como por exemplo: ensinar. Neste trabalho foi pego uma sala de aula com vários alunos com conhecimentos sobre enrega continua e outros não e foi ensina desde o inicio sobre isso, mas não foi falado sobre métricas.

Tabela 4 – Achados sobre as métricas de avaliação discutidas nos estudos

chave e apenas para a área "Computação". Ainda, na *Scopus* e *Engineering Village*, a busca foi restringida apenas para artigos em inglês.

Ao realizar a busca utilizando as strings de busca em cada base de dados, foram obtidos no total 702 estudos. A Tabela 5 apresenta a distribuição dos artigos por base de dados. Na identificação das duplicatas, foram removidos 363 artigos. Desta forma, restaram 339 artigos para serem analisados.

Após a etapa de seleção inicial, o pesquisador 1 selecionou 33 artigos entre os 363, e ao aplicar os critérios de exclusão e inclusão esse número diminuiu para 22. Já o pesquisador 2 selecionou inicialmente 30 artigos, e após a aplicação dos critérios restaram 27 artigos. Como houve diferença no número de artigos aceitos, ao realizar a sincronização dos resultados, 4 dos trabalhos aceitos pelo pesquisador 2 também foram incluídos pelo outro pesquisador. Para o trabalho restantes, fez-se a leitura em conjunto do mesmo e após diversas discussões decidiu-se que o trabalho estava de acordo com o critério de inclusão, logo, foi mantido resultando em um total de 27 trabalhos. A Tabela 6 apresenta os artigos aceitos.

Base de Dados	Quantidade	Porcentagem
ACM	57	8,12%
Engineering Village	208	29,63%
IEE	103	14,67%
Science Direct	28	3,99%
Scopus	302	43,02%
Springer Link	4	0,57

Tabela 5 – Base De Dados

3.8.3.2 Aplicação dos Critérios de Qualidade

A Tabela 6 apresenta os trabalhos aceitos e a avaliação realizada sobre os critérios de qualidade. Vale ressaltar que nenhum trabalho foi excluído por conta dos critérios de qualidade.

Após a leitura do trabalho foi identificado usando como base o formulário de extração que para grande parte dos trabalhos todas as etapas definidas nos pipelines eram automatizadas e sendo comum em todos eles terem a etapa de *build* onde o projeto é construído e a etapa de *tests* onde os testes unitários são executados. Houveram também vários casos onde não foram definidas detalhadamente quais seriam as etapas, supondo-se assim baseado no relatado no texto que teriam pelo menos as etapas de *build* e *tests* se baseando na análise onde há algum momento que o projeto é construído e testado automaticamente. A Exceção foi os trabalhos (MYSARI; BEJGAM, 2020) e (AGARWAL; GUPTA; CHOUDHURY, 2018) que possuíam uma etapa semi-automatizada e manual respectivamente. No trabalho (MYSARI; BEJGAM, 2020) a etapa semi-automatizada seria definida como *Manual approval*, no contexto do trabalho se entende que nesta fase ocorre a aprovação manual da etapa por algum membro da equipe, para dar continuidade ao fluxo do pipeline. No (AGARWAL; GUPTA; CHOUDHURY, 2018) foi relatado um estudo sobre empresas que aplicavam integração, entrega e implantação contínua. Na maior parte das empresas o fluxo era semi-automatizado de 40% a 60% do fluxo. Em dois casos o fluxo era totalmente manual.

Na Tabela 7 é apresentado o incremento ferramental relatado nos trabalhos, dentre estes trabalho vale ressaltar o (LUO; YE; ZHANG, 2015), (SáNCHEZ-GALLEGOS et al., 2020), (MäKINEN et al., 2016), (IVANOV; SMOLANDER, 2018), (CHEN, 2017), (PAULE; DüLLMANN; HOORN, 2019), (REHMANN et al., 2016) que não possuem nenhum incremento ferramental proposto e fazem uso apenas de ferramentas já existentes ou não relatam a criação de novo ferramental. Já os trabalhos (WETTINGER; ANDRIKOPOULOS; LEYMANN, 2015), (GREISING; BARTEL; HAGEL, 2018), (DEBROY; MILLER; BRIMBLE, 2018) e (ARACHCHI; PERERA, 2018) apresentam incremento ferramental para Configuração de Pipelines, Deployment Contínua, Entrega Contínua, Integração Contínua.

Estudo	**Autor**	**CQ01**	**CQ02**	**CQ03**	**CQ04**	**Avaliação**
S01	(NGUYEN; DUPUIS, 2019)	D	D	D	C	2.0
S02	(WETTINGER; ANDRIKOPOULOS; LEYMANN, 2015)	B	B	D	D	3.5
S03	(GREISING; BARTEL; HAGEL, 2018)	D	C	C	B	3.5
S04	(ADAMS; MCINTOSH, 2016)	D	D	C	A	3.5
S05	(LUO; YE; ZHANG, 2015)	B	D	C	C	3.5
S06	(BIENER; CRAWFORD, 2019)	C	D	B	C	3.5
S07	(ULLAH et al., 2017)	B	D	B	C	4.0
S08	(AHMADIGHOHANDIZI; SYSTÄ, 2015)	B	D	B	C	4.0
S09	(ZACHOW, 2016)	D	C	B	A	4.5
S10	(GUşEILă; BRATU; MORARU, 2019)	C	C	B	B	4.5
S11	(SáNCHEZ-GALLEGOS et al., 2020)	B	C	B	B	5.0
S12	(MYSARI; BEJGAM, 2020)	B	C	C	A	5.0
S13	(MäKINEN et al., 2016)	B	C	C	A	5.0
S14	(ANTUNES; NAVARRO; HANAZUMI, 2018)	B	C	B	B	5.0
S15	(IVANOV; SMOLANDER, 2018)	D	C	A	A	5.5
S16	(CHEN, 2017)	D	A	A	B	6.0
S17	(WETTINGER; BREITENBüCHER; LEYMANN, 2015)	A	C	B	A	6.0
S18	(DEBROY; MILLER; BRIMBLE, 2018)	B	B	A	C	6.0
S19	(PAULE; DüLLMANN; HOORN, 2019)	C	B	A	B	6.0
S20	(STEFFENS; LICHTER; DöRING, 2018)	A	B	A	C	6.5
S21	(ARACHCHI; PERERA, 2018)	B	C	A	A	6.5
S22	(REHMANN et al., 2016)	B	A	A	C	6.5
S23	(AGARWAL; GUPTA; CHOUDHURY, 2018)	A	C	A	A	7.0
S24	(HäKLI; TAIBI; SYSTA, 2018)	B	B	A	A	7.0
S25	(ITKONEN et al., 2016)	B	A	A	B	7.0
S26	(GMEINER; RAMLER; HASLINGER, 2015)	B	A	A	B	7.0
S27	(EDDY et al., 2017)	B	A	A	A	7.5

Tabela 6 – Artigos Aceitos e Critérios de Qualidade Avaliados

Study	CP	DC	EC	IC	NP
S01	-	-	-	-	X
S02	X	X	X	X	-
S03	X	X	X	X	-
S04	-	X	X	X	-
S05	-	-	-	-	X
S06	-	-	X	X	-
S07	-	X	-	X	-
S08	-	X	X	X	-
S09	X	-	X	X	-
S10	X	-	X	X	-
S11	-	-	-	-	X
S12	X	X	-	X	-
S13	-	-	-	-	X
S14	X	X	-	X	-
S15	-	-	-	-	X
S16	-	-	-	-	X
S17	-	X	X	X	-
S18	X	X	X	X	-
S19	-	-	-	-	X
S20	-	X	X	X	-
S21	X	X	X	X	-
S22	-	-	-	-	X
S23	-	X	X	X	-
S24	-	X	X	X	-
S25	-	-	-	X	-
S26	X	X	X	X	-
S27	X	-	X	X	-

Tabela 7 – Incremento ferramental proposto. *Configuração de pipelines* (CP),*Deployment Contínuo* (DC), *Entrega Contínua* (EC), *Integração Contínua* (IC), *Nenhum incremento ferramental proposto* (NP)

3.8.4 Respostas das Questões de Pesquisa

QP01 - Quais estudos apresentam ferramentas de suporte para a configuração automatizada de pipelines de Integração Contínua (IC), Deploy Contínuo (DC) e Entrega Contínua (EC) e quais são as ferramentas utilizadas?

O trabalho (STEFFENS; LICHTER; DöRING, 2018) propõe um sistema de integração e entrega contínuas, a solução foi implementada e mostrou resultados promissores aos pesquisadores que redigiram o tralho como a melhoria no *feedback* de erros. A grande maioria dos estudos usa um ferramental já existente, usando principalmente como servidor de automação o Jenkins.

QP02 - Quais são os contextos das empresas ou organizações que utilizam IC/DC/EC?

Em grande parte dos artigos que foram lidos na construção deste trabalho, foi notado que existe uma carência nesta área tanto de profissionais quanto de empresas que utilizem práticas contínuas no desenvolvimento, em basicamente quase todos os trabalhos que foram aplicados na indústria os problemas comumente apresentados são o custo alto de implantação de práticas contínuas, o tempo necessário para a adequação da equipe e a dificuldade da adesão da equipe.

O (EDDY et al., 2017) fez um estudo dentro da sala de aula com alunos que um grupo de alunos que pensavam que sabiam sobre o conteúdo, outro grupo que não sabiam sobre oque se tratava. Porém os artigos [(HäKLI; TAIBI; SYSTA, 2018), (DEBROY; MILLER; BRIMBLE, 2018), (ANTUNES; NAVARRO; HANAZUMI, 2018), (STEFFENS; LICHTER; DöRING, 2018), (MäKINEN et al., 2016), (ITKONEN et al., 2016), (GMEINER; RAMLER; HASLINGER, 2015), (REHMANN et al., 2016), (CHEN, 2017)] fizeram testes dentro de empresas que queriam implantar IC/DC/EC, mas a maioria dos artigos fez testes em ambientes controlados se preocupando apenas em demonstrar o funcionamento de IC/DC/EC. Em (CHEN, 2017) foi feito um estudo de caso em uma organização com faturamento anual de aproximadamente sete bilhões de euros chamada *Paddy Power* e teve como principal desafio a adesão das equipes de software as práticas de Entrega Contínua.

3.8.4.1 Vantagens e Desvantagens

Esta seção tem por objetivo responder a **QP03 - Quais foram as vantagens e desvantagens observados no estudo?** , caracterizando a literatura da área como segue:

3.8.4.1.1 Vantagens

- Melhoria na qualidade de software - Estudos mapeados (CHEN, 2017);
- Maior automação para práticas ágeis - a automação de aplicações em múltiplas camadas, serviços e componentes garante a operacionalidade de práticas ágeis;
- Fornecer software de qualidade com mais rapidez - a automação promovida por estes sistemas, por sua vez, permitem que software sejam produzidos em ritmos mais acelerados, aumentando o rendimento e qualidade das equipes. Estudos mapeados (CHEN, 2017);
- Aumentar o desempenho da organização como um todo - Sistemas integrados de IC, DC e EC integram ferramentas da produção à gerência DevOps, como de construção e teste, visualização detalhada de pipelines e organização estrutural para suportar as crescentes demandas e tempo de mercado;

- Melhoria da qualidade - As abordagens incentivam a implantação de testes, que ainda são muito negligenciados nas equipes de desenvolvimento;
- Suporte para integração multi-plataforma - Sistemas integrados de integração, implantação e entrega tendem à garantir a operacionalidade em uma gama de tecnologias adotadas em ecossistemas de fábricas de software;
- Flexibilidade para configuração em arquiteturas modernas de serviços - Relatos apontam a utilização de arquiteturas distribuídas e em micro-serviços, implantadas localmente e na nuvem;
- Governança pelo fluxo de automação no pipeline - As entradas de qualidade, a visibilidade e as aprovações tendem à garantir que apenas as configurações testadas sejam liberadas;
- Auditoria associada com visibilidade - O engenheiro de software tem clara visibilidade dos aplicativos implantados, garantindo conformidade entre as atividades de implantações e a auditoria de qualidade;
- Relato em tempo real - as ferramentas, uma vez que integram atividades de testes com implantação, permitem aos engenheiros de software detectar problemas em tempo real (CHEN, 2017), antes que o processo de entrega ocorra em pipelines mais avançados;
- Menor risco no desenvolvimento de software como um todo - Uma vez que primam pela garantia da qualidade, tecnologias para IC, DC e EC apresentam aos que às adotam um menor risco em relação à problemas derivados da chamada crise do software[1].

3.8.4.1.2 Desvantagens

- Requer muito conhecimento técnico -
- Requer um grande investimento inicial para operacionalizar o ambiente -
- Não são ferramentas adaptadas para a alta gestão -
- Alta curva de aprendizado nas disciplinas de Engenharia de Software -
- O retorno de investimento (ROI) não é muito perceptível quanto ao aumento de receitas - A adoção de práticas contínuas em produtos não geram diretamente receita, uma vez que não tem por objetivo direto a melhoria nos negócios de fábricas

[1] https://cienciacomputacao.com.br/tecnologia/o-que-foi-a-crise-do-software-e-o-inicio-da-engenharia-de-software/

de software (CHEN, 2017). Assim, temos observado que o cômputo de ROI nunca é mencionado pelos trabalhos analisados, o que gera insegurança por parte das fábricas de software quanto à planejamento estratégico para adoção.

3.8.4.2 Desafios de Pesquisa

QP04 - Quais os desafios de pesquisa na área de gerenciamento de configuração? Os principais desafios relatados foram a quantidade de tempo e recursos humanos qualificados necessários para a configuração dos pipelines. Outro ponto interessante relatado em (CHEN, 2017) é dificuldade da adesão das equipes de software as práticas de Entrega Contínua que como relatado no trabalho foram quase resolvidas ao apresentar as vantagens que os desenvolvedores teriam ao aceitar essas mudanças.

Além disso, a viabilidade da implantação/entrega contínua também depende de escolhas arquiteturais adequadas para o sistema. Nesse sentido, uma arquitetura baseada em microsserviços é frequentemente considerada central para a aplicação de CD e *DevOps*, pois ela entrega pequenos átomos de funcionalidades auto-contidos. A autonomia técnica de microsserviços permite criar pipelines de implantação independentes para cada serviço. Assim, as mudanças podem ser entregues rapidamente à produção, uma vez que somente os serviços afetados precisam ser construídos e testados e vários pipelines podem ser executados em paralelo. A escalabilidade, flexibilidade e portabilidade são alguns benefícios frequentemente associados com este estilo arquitetural. OBS.: Todo serviço deve ser uma unidade autônoma, independentemente implantável, de tamanho gerenciável que interage com outros serviços somente por meio de interfaces agnósticas à tecnologia, tal como APIs Web RESTful (HEINRICH et al., 2017).

3.8.5 Ameaças à Validade

As principais ameaças que podem comprometer a validade do nosso mapeamento sistemático são:

1. Não estamos cientes dos vieses que podemos ter ao analisar e categorizar os artigos, porém o leitor deve estar ciente do possível impacto de nossos próprios interesses nas análises. De forma a tentar minimizar essa ameaça definimos critérios de inclusão e exclusão. Em conflitos onde as avaliações sobre um trabalho qualquer foi diferente, ambos os pesquisadores discutiram seus pontos e caso não fosse resolvido um terceiro pesquisador deveria ser consultado.

2. Definimos nossa *String* de busca baseada em apenas alguns trabalhos e nosso conhecimento sobre o tema e algumas buscas onde validávamos os títulos de alguns trabalhos, então mesmo tendo sido feitas várias revisões em cima da *String* não foi possível garantir que todos os termos estariam na *String*.

3. É possível que algum trabalho não esteja disponível nas bases usadas ou que não estão disponíveis, devido isso não podemos garantir que todos os trabalhos sobre o assunto foram mapeados.

4. Nenhum dos pesquisadores tem experiência em mapeamento sistemático, devido isso pode ocorrer de existir outros problemas que não foram identificados/corrigidos/mitigados.

3.8.6 Discussões

Nesse contexto, um pipeline de entrega contínua representa o processo de caminho único até a implantação em produção pelo qual todas as mudanças de um dado sistema devem passar, seja uma mudança de código-fonte, configuração de infraestrutura ou ambiente, teste automatizado, *script* de banco de dados ou da cadeia de ferramentas (HUMBLE; MOLESKY, 2011). Um pipeline típico de implantação/entrega contínua consiste de um ambiente de desenvolvimento (e sistema de controle de versão), um ambiente de construção (*build*), um ambiente de validação (*staging*) e o ambiente de produção (operação normal). Então, por exemplo, quando um desenvolvedor faz um *commit* de código no sistema de controle de versão (ambiente de desenvolvimento), este passa automaticamente pela integração e testes automatizados (de unidade, integração, regressão e sistema) no ambiente de *build*, passa, na sequência, por outros testes no ambiente de validação (testes de aceitação do usuário, simulações e testes de performance) até que é finalmente implantado em produção (por demanda ou não) (BASS, 2018). Com o gerenciamento automatizado do pipeline, se tem rastreabilidade completa e habilidade de auditoria para todas as mudanças, assim como toda a configuração e passos necessários para recriar o ambiente correto para um incremento de sistema (mudança) são armazenados (HUMBLE; MOLESKY, 2011). Ou seja, para todo incremento em produção deve ser possível rastrear seus elementos constituintes, bem como os casos de teste que foram aplicados, as dependências do serviço e as versões das ferramentas no pipeline que foram usadas para a implantação.

No contexto desse pipeline para acelerar a entrega de produtos de software, a Integração Contínua (IC) é um ativador chave. Ela surgiu inicialmente na comunidade de *Extreme Programming* e é descrita como uma prática de desenvolvimento de software onde membros de uma equipe integram seu trabalho com frequência, pelo menos diariamente, levando a múltiplas integrações por dia, onde cada integração é verificada por um *build* automatizado (incluindo testes) para detectar erros de integração o mais rápido possível (FOWLER, 2006).

Ferramentas de CI constroem e testam automaticamente uma base de código de um projeto, isoladamente, com cada mudança (por *default*) (ZHAO et al., 2017b) e tem sido amplamente adotadas na prática, especialmente aquelas de código aberto baseadas na nuvem, compatíveis com o sistema de controle de versões *GIT*, como *TRAVIS CI*,

CLOUDBEES e *CIRCLECI* (ZHAO et al., 2017b).

A IC requer que todos os testes executados sejam automatizados. Portanto, um desafio nesse contexto é a seleção rápida e eficaz, pela equipe de desenvolvimento, do conjunto de casos de teste que serão aplicados de modo automatizado por uma ferramenta de IC. Os casos de teste devem ser completos o suficiente para ter uma boa cobertura das funcionalidades do sistema e seus requisitos de qualidade (por exemplo, performance e segurança) e ter um número reduzido para não consumir muito tempo de processamento (BASS, 2018).

4 DESENVOLVIMENTO E EXECUÇÃO DE UMA ESTRATÉGIA PARA INTRODUZIR PIPELINES DE INTEGRAÇÃO CONTÍNUA EM DISCIPLINAS DE RESOLUÇÃO DE PROBLEMAS V

As disciplinas de RP da Unipampa utilizam a Abordagem Baseada em Problemas (**ABP**), do inglês *Problem-Based Learning* (**PBL**). Esta é uma abordagem de aprendizagem ativa que possibilita aos alunos vivenciar experiências semelhantes à realidade do contexto profissional através da resolução de problemas reais, que são encontrados neste contexto. A disciplina de RP V, mencionada anteriormente, foi escolhida como alvo para a pesquisa apresentada nesse documento tendo em vista que toda a motivação se deu na prática, cujos proponentes ou participaram como ouvintes ou planejadores da disciplina.

4.0.1 A Resolução de Problemas como Componentes Curriculares

O curso de Engenharia de Software da Unipampa contém 6 disciplinas ou componentes curriculares que adotam a Abordagem Baseada em Problemas (ABP). Estas disciplinas são denominadas Resolução de Problemas (RPs) e abordam diferentes conteúdos do curso propondo a resolução de problemas muito semelhantes àqueles que os alunos encontrarão no mercado de trabalho. As RPs da Unipampa cobrem todas as áreas centrais da Engenharia de Software. A disciplina de Resolução de Problemas V, que é o foco principal dos estudos reportados nesse artigo, busca fortalecer o conhecimento dos alunos no que se refere a práticas de manutenção de software, incluindo sub-problemas que envolvam práticas de refatoração, teste, e automação da gerência de configuração. Para cursar esta disciplina, os alunos já deveriam ter realizado anteriormente outras disciplinas do curso que abordam cada um dos aspectos do Gerenciamento de Configuração.

4.1 Caracterização do Cenário das Disciplinas de Resolução de Problemas V

Esta seção aborda um estudo realizado com o intuito de identificar as características da disciplina de Resolução de Problemas 5 com a intenção de identificar gargalos de aprendizado na automação de atividades de gerência de configuração.

4.1.1 Survey com Alunos de RP V

Um estudo do tipo survey foi realizado com alunos de diferentes turmas da disciplina de Resolução de Problemas V com o intuito de identificar o perfil destes alunos e, a partir disto, traçar estratégias para a definição de uma estratégia de ensino-aprendizagem de apoio à configuração de pipelines de integração contínua. Para tanto, foi elaborado e aplicado um questionário usando o *Google Forms*.

O *survey* teve dois principais objetivos específicos:

(1) Caracterizar o conhecimento de alunos das disciplinas de RP V, conduzidas entre 2016 e 2019, de modo a identificar suas limitações quanto ao aprendizado da gerência

de configuração (tema abordado em disciplinas anteriores do curso de Engenharia de Software);

(2) Caracterizar os desafios do aprendizado de IC nas disciplinas de Resolução de Problemas V, através de uma atividade de configuração de um pipeline de Integração Contínua dentro da disciplina.

ID	Questão de pesquisa
QC1	As disciplinas anteriores forneceram a base necessária de "gerência de configuração"para utilizar uma ferramenta de versionamento de código (como o GIT, SVN ou CVS)?
QC2	As disciplinas anteriores forneceram a base necessária de "gerência de configuração"para utilizar uma ferramenta de automação da implantação/*deploy* (como ANT ou MAVEN)?
QC3	As disciplinas anteriores forneceram a base necessária para desenvolver e automatizar testes de unidade?
QC4	As disciplinas anteriores forneceram a base necessária para desenvolver e automatizar testes de integração?
QC5	As disciplinas anteriores forneceram a base necessária para desenvolver e automatizar testes de regressão?
QC6	As disciplinas anteriores forneceram a base necessária para desenvolver e automatizar testes de aceitação?
QC7	As disciplinas anteriores forneceram a base necessária para gerenciar alterações no software decorrentes de sua evolução?
QC8	As disciplinas anteriores forneceram a base necessária para gerenciar mudanças de requisitos de software?
QC9	As disciplinas anteriores forneceram a base necessária para realizar a auditoria e relatório das mudanças de requisitos?

Tabela 8 – Lista de Questões de Caracterização (QC) do perfil dos alunos respondentes

Como ilustram as Figuras 33 e 34, a maioria (mais da metade) dos respondentes cursou a disciplina de RP V em 2018/1. Mesmo nesta disciplina, onde se cobrou a configuração de um servidor de IC, os alunos tiveram dificuldades na implementação do problema proposto pelos professores.

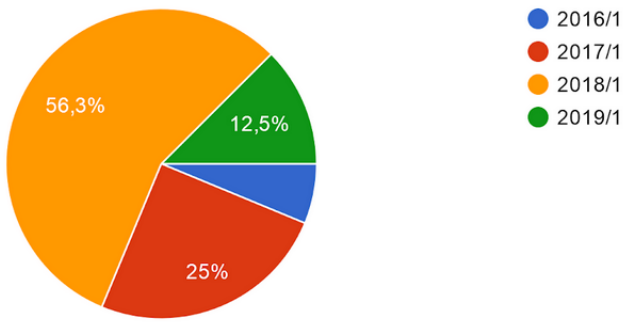

Figura 3 – Ano/Período em que os respondentes cursaram Resolução de Problemas V

Na segunda seção foi questionado sobre os conceitos e práticas anteriores à disciplina. As questões eram sobre se as disciplinas anteriores a RP V forneciam a base

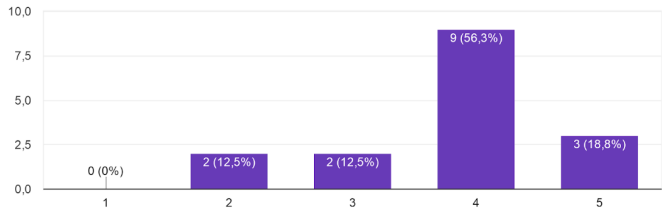

Figura 4 – Nível de dificuldade para a implementação do problema proposto em Resolução de Problema V

necessária para: 1) gerência de configuração; 2) desenvolver e automatizar testes de testes de unidade, integração, regressão e aceitação; 3) gerenciar alterações em software decorrentes de uma nova *Sprint*; 4) gerenciar mudanças de requisitos de software; 5) realizar a auditoria e relatório das mudanças de requisitos.

Nas perguntas sobre gerência de configuração, a maior parte dos respondentes do primeiro grupo (Geral) relatou que as disciplinas anteriores a RP V forneceram as bases necessárias para utilizar uma ferramenta para versionamento de código. Porém, a maior parte discordou totalmente referente as disciplinas anteriores a RP V fornecerem a base para utilizar uma ferramenta para automação da implantação. Portanto, há um forte indício de que a estratégia deva reforçar a automação da implantação, ou que este conteúdo seja incluído em alguma das disciplinas da Engenharia de Software.

A Tabela 8 apresenta as questões de pesquisa e os resultados do survey são apresentados na Figura 5.

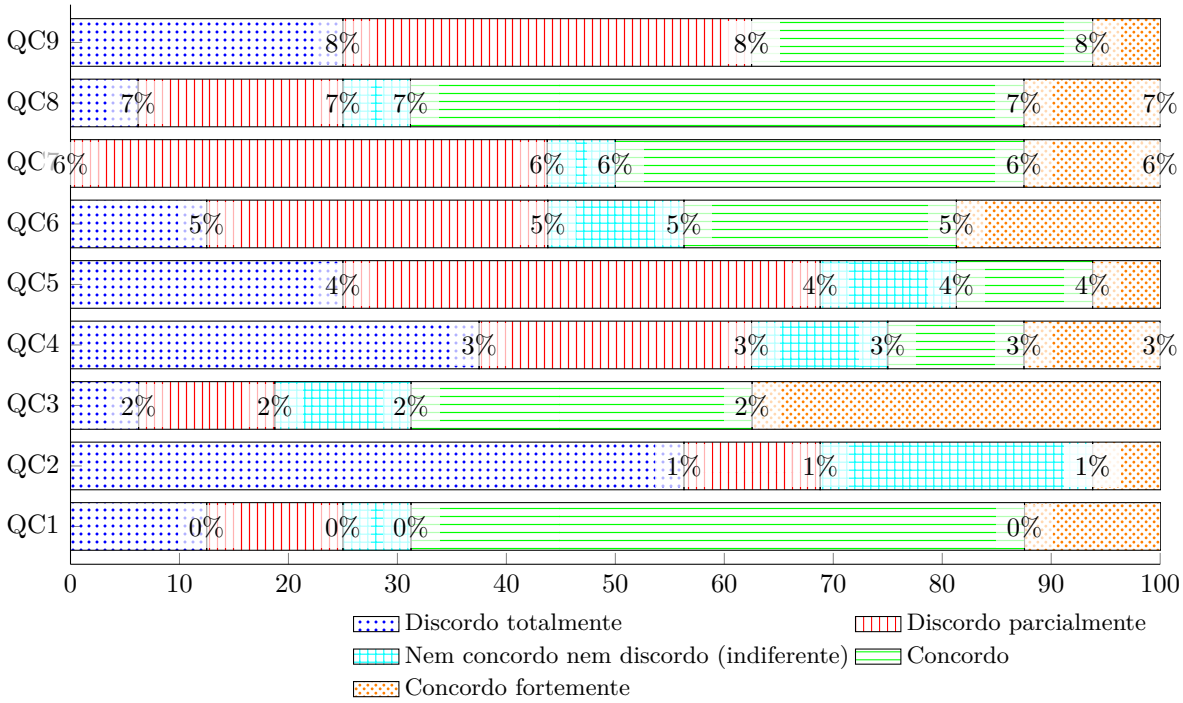

Figura 5 – Resultados das questões de caracterização com os gargalos de aprendizado.

Nas questões sobre desenvolvimento e automatização de testes a maioria dos entre-

vistados concordou que as disciplinas anteriores a RP V forneciam a base para desenvolver e automatizar testes de unidade e integração. Porém, discordaram que elas forneceram a base referente aos testes de regressão.

Sobre os testes de aceitação, a opinião dos entrevistados ficou dividida. Ou seja, há a necessidade de uma estratégia de ensino auxiliar na inclusão dos testes Selenium nos pipelines de IC. Quanto aos testes de aceitação, há a necessidade de reforço em disciplinas que apresentem checklists pra execução de testes de aceitação. Quanto à estratégia de ensino, uma vez que um teste de aceitação é manual, é preciso prever o uso de uma atividade que utilize gatilhos/*triggers* no pipeline de IC.

Os entrevistados também foram questionados sobre dificuldades no gerenciamento de alterações em software dentro de uma *Sprint*. A maioria dos respondentes concorda que essa fundamentação foi transmitida em disciplinas anteriores ou na disciplina de RP V corrente.

Sobre mudanças de requisitos de software, a maioria dos entrevistados concordou que as disciplinas anteriores forneceram a base necessária. Porém, sobre a base necessária para realizar a auditoria e o relatório das mudanças de requisitos, a maioria discordou que fundamentos tenham sido transmitidos em disciplinas anteriores.

4.1.2 Estudo Exploratório de Mineração de Repositórios

Também foi realizado uma mineração dos repositórios da disciplina de Resolução de Problemas V conduzida em 2018. Trata-se de um estudo de caso exploratório no estilo (*Mining Software Repository*) conduzidos sobre repositórios de software da turma de 2018 de RP V. Foi realizada uma mineração daquilo que foi configurado pelos cinco times na representação de um pipeline de integração contínua. Para auxiliar à delinear gargalos de ensino e aprendizagem, foi feita uma análise nos arquivos de configuração usados pelos grupos em seus respectivos projetos no *BitBucket*. Informalmente, também houveram questionamentos aos grupos sobre a configuração de seus servidores de IC.

A mineração observou várias inconsistências de configuração em todos os repositórios. Isso demonstra que os alunos não sabiam exatamente o que estavam fazendo, o que sugere que a introdução do problema de modo gradual assistido por referencial teórico e prático é desejável. Por fim, os resultados do estudo observacional de mineração de repositórios de software apontou que não é de fácil aprendizado de tarefa que visam automatizar processos de gerência de configuração, em especial a de configurar um servidor/serviço de integração contínua.

4.1.3 Aprendizados para Guiar as Próximas Disciplinas

O *survey* bem como a prática de buscar a automação de processos de integração contínua nos levaram aos seguintes aprendizados em nível de objetivos micro. Então, além do estudo de *survey* relatado, que teve como propósito identificar gargalos de ensino

em disciplinas de Resolução de Problemas 5, o mesmo estudo também aprofundou a entrevista para identificar gargalos nas práticas de integração contínua associadas com os objetivos micro de automação de processos para gerência de configuração de software. Na sequência, se executou um estudo de mineração de repositórios. Optamos por sumarizar os aprendizados destes estudos nessa seção, como segue.

Dificuldade de configuração: Após a análise dos resultados do *survey*, foi identificado que a maioria dos grupos tiveram bastante dificuldade para a criação e configuração dos arquivos de configuração. Além disso, várias redundâncias também foram observadas. Quatro dos cinco grupos não conseguiram configurar corretamente o servidor de IC *Circle CI*, como requerido pelos professores. Apenas um grupo conseguiu configurar corretamente o servidor de IC, sendo que de seus quatro membros, apenas um deles desempenhou o papel de gerente de configurações. Todos os repositórios que foram analisados, ou não possuíam nenhum arquivo de configuração, ou usavam o padrão do servidor *Circle CI* sem customizações.

Inconsistências entre prática relatada e prática detectada: Também foi observado que haviam várias inconsistências nos arquivos de configuração. As mais comuns são a configuração de um banco de dados, que nunca era utilizado, caminhos errados para o diretório de trabalho, onde os testes deveriam ser executados, e erros simples de sintaxe. Vale ressaltar que a configuração do banco de dados não utilizado não impediu o *Cicle CI* de funcionar. Porém, o diretório incorreto e os pequenos erros sintáticos impediram. Outro ponto interessante é que a maioria dos arquivos de configuração tinham semelhanças com a de um grupo em específico, como se tivessem sido derivados do arquivo de configuração desse grupo específico e posteriormente alterados para os outros projetos. Mesmo assim, os arquivos de configuração analisados estavam incorretos e apresentavam redundâncias, o que demonstra que os gerentes de configuração não sabiam exatamente o que estavam realizando.

4.1.4 Lições para Aprofundamento da Experiência Individual

Após análise das configurações dos cinco projetos, alguns participantes foram questionados individualmente, de forma informal, sobre: 1) quais são suas dificuldades para configurar o *Circle CI*?, e 2) por quê não conseguiram implementar o servidor de Integração Contínua?

A maioria das respostas convergiram em: 1) porque que existe grande dificuldade para oferecer uma solução para o projeto na implantação de um pipeline de Integração Contínua em um curto espaço de tempo 2) a carga de atividades executadas em paralelo com o desenvolvimento do software, o que impediu que os gerentes pudessem focar na aprendizagem da Integração Contínua, e 3) questão de prioridade, pois o tempo era limitado e os gerentes consideravam mais importante entregar o projeto.

Esta entrevista informal levou ao desenvolvimento de um questionário formal. Ou

seja, buscando-se caracterizar melhor as dificuldades observadas pelos professores e alunos de RP V 2018/1, o seguinte estudo foi executado.

4.1.5 Aprendizado para Aprofundamento da Experiência Coletiva

Dificuldades para planejar a atividade de automação em nível ténico/micro. Há grande dificuldade para estudantes de Engenharia de Software compreender os requisitos e assimilarem os conceitos de integração contínua em ambiente acadêmico. Primeiramente, não existem soluções "plug-and-play"para integração de recursos em servidores de IC, já que cada projeto de software demanda uma configuração diferente em nível micro de objetivos. Além disso, os três servidores estudados também apresentam diferenças para a configuração de pipelines, o que dificulta a troca de infromação entre os times de desenvolvimento. Somando à isso, também se observou que, mesmo dedicando um tempo para a pesquisa desses servidores, o conhecimento especializado dessas ferramentas não é sistematizado nos materiais com cunho didático disponíveis na web. Ou seja, ele permanece tácito na mente dos desenvolvedores. Em adição, por meio de uma entrevista, identificou-se muitos elementos que tornam difícil a configuração destes serviços, resultando neste documento. Portanto, entende-se que estas dificuldades são um grande obstáculo para a formação de gerentes de configuração em nosso curso de graduação.

Alunos gostariam de materiais mais instrutivos para a condução da disciplina de Resolução de Problemas 5. Uma vez demonstrado que configurar um servidor/serviço de Integração Contínua não é uma tarefa fácil nem trivial, abre-se espaço para contribuições que resolvam estas dificuldades. Para superar este obstáculo, é preciso primeiro sistematizar as atividades para configuração desses serviços em problemas que possam gradualmente ser introduzidos na disciplina de RP V. Em um segundo momento, pode-se buscar um nível de dificuldade maior em pipelines do que as implementações disponíveis nos servidores de IC, como caracterizadas aqui em nível macro e intermediário de objetivos. Portanto, uma vez que superar tais dificuldades é de grande valia para o curso de Engenharia de Software, o advento destes documentos instrutivos pode contribuir com a redução da curva de aprendizado de alunos em futuras disciplinas de RP V.

Falta um problema que exija a automação em pipelines de integração contínua. Nos problemas anteriores de RP 5, os objetivos de automação não eram claramente definidos, e eles são fundamentais para garantir ao time a evolução de software com eficiência. Para caracterizar o problema, o survey que foi aplicado em turmas de RP V identificou a relevância de um espaço onde múltiplos conhecimentos do curso são praticados e, dentre eles, a gerência de configuração e automação de testes. Uma vez que os alunos focavam na entrega de produtos e não na assimilação de práticas, era necessário definir um problema que necessitasse da introdução de práticas de integração contínua. Para a turma de REsolução de Problemas 5 de 2021, o problema em questão foi focar o Marco 3 na evolução de banco de dados.

4.1.6 Considerações Finais

Concluiu-se que a maioria dos respondentes não consegue configurar corretamente estes serviços que buscam a automação de atividades de gerência de configuração de software.

4.2 Estratégias Graduais de Problematização na Automação de Processos de Gerenciamento de Configuração

À seguir, apresentam-se alguns elementos que podem influenciar na estratégia da adoção de práticas tanto pelo time como pelo professor tutor da disciplina de Resolução de Problemas V. Optou-se por distribuir as estratégias em nível macro, onde consideram-se elementos de tomada de decisão em nível organizacional das fábricas de software, em nível intermediário, onde níveis de maturidade de times ágeis são buscados como objetivo, e em nível micro, onde o time, selecionando práticas ágeis que necessitam de integração contínua, buscam implantar processos de automação de pipelines. Este documento apresenta uma contribuição para decisão em nível micro.

4.2.1 Objetivos de Automação em Nível Macro

Esta seção apresenta elementos que podem ser considerados como requisitos macro no planejamento da automação, i.e., para adoção gradual de abordagens de automação de processos de desenvolvimento de software. Abaixo, elencam-se os requisitos de automação macro, porém ainda sem um *guideline* ou *framework* de referência para auxílio. Este *guideline* de nível macro está em desenvolvimento, e também será disponibilizado para a turma de Resolução de Problemas 5.

- **Planejamento para o DevOps.** DevOps é a junção dos termos "development" (Dev) e "operations" (Ops) para descrever a combinação de filosofias culturais, metodologias, ferramentas e práticas para: a) a integração das áreas de desenvolvimento (desenvolvedores de softwares), b) operações (sysadmin ou infraestrutura) e c) controle de qualidade (QA – Quality Assurance). Ela tem o propósito de entregar de maneira ágil e contínua, valor e melhor experiência ao cliente.

- **Planejamento para a Engenharia de Software Contínua.** Esta consiste em um conjunto de práticas e ferramentas que apoiam uma visão holística do desenvolvimento de software com o objetivo de torná-lo mais rápido, iterativo, integrado, contínuo e alinhado ao negócio. Ela entende que o processo de desenvolvimento de software não é uma sequência de atividades discretas, realizadas por equipes distintas e desconectadas. Visa estabelecer um fluxo contínuo entre as atividades relacionadas ao software, levando em consideração todo o ciclo de vida do software. É um tópico recente que procura transformar práticas de desenvolvimento

discreto em alternativas mais iterativas, flexíveis e contínuas, mantendo o objetivo de construir e entregar produtos de qualidade de acordo com o tempo e os custos estabelecidos.

4.2.2 Objetivos de Automação em Nível Intermediário de Práticas Ágeis

A avaliação do nível de maturidade do time na adoção de práticas mais pontuais, denominadas neste documento como em nível micro, deve ocorrer conforme a dimensão do problema buscado em cada Sprint e os objetivos de melhoria do time. Esta seção apresenta como sugestão a adoção de um framework que apresenta tais objetivos, porém na forma de níveis de maturação das equipes ágeis.

Os pipelines de integração de implantação contínua sugeridos na Figura 6 também foram elaborados conforme níveis de maturidade que podem ser obtidos pelos times ágeis ao longo do processo de aprendizagem. Para tal, sugere-se a utilização do framework "How Agile Are You? Let's Actually Measure It!" (BALBES, 2015a). Este framework apresenta várias práticas para diagnótico de maturidade em equipes ágeis, foi utilizada durante a disciplina de Resolução de Problemas 5 de 2018 e, inclusive, tem motivado pesquisas no tema.

O framework estabelece um conjunto de práticas ágeis mapeadas para áreas do conhecimento da Engenharia de Software. Cada área agrupa um conjunto de práticas cuja maturidade é gradualmente percebida no time. O autor também apresenta uma descrição da metodologia de pontuação de 0-5 usada como referenciada, indicando a pontuação 0 como um nível de maturidade baixo e 5 alto grau de maturidade em determinada prática. O valor zero deve ser usado quando não houver no time capacidade em qualquer área específica.

Em especial, para a definição de estratégias que visem a automação de processos com IC, DC e EC, práticas que podem ser introduzidas gradualmente no time e que podem ser também avaliadas pelo professor tutor incluem:

- **Planejamento para Assimilação de Práticas Tratando do Artesanato Técnico (*Technical Craftsmanship*)**, cujo material de suporte é dividido em Página 1 (BALBES, 2015d) e Página 2 (BALBES, 2015e). O artesanato técnico busca fomentar práticas como testes unitários automatizados, testes não funcionais, desenvolvimento dirigido à testes, integração contínua, programação em pares, decompor os problemas em problemas menores (chamado de *spiking*), controle de versionamento e *banches*, gerenciamento de versões, padronização de codificação, alinhamento com atividades do processo de desenvolvimento, compartilhamento de código, e gerenciamento de mudanças de software e;

- **Planejamento para Assimilação de Práticas Tratando de Advocacia da Qualidade (*Quality Advocacy*)**, também dividida em Página 1 (BALBES, 2015b)

e Página 2 (BALBES, 2015c), que busca fomentar práticas como foco na qualidade do código, definição do processo, medição da qualidade interna do time, medição da qualidade externa do time, apropriação do time por elementos de qualidade, gerenciamento de defeito e medição, testes de aceitação do usuário e o uso de testes exploratórios.

4.2.3 Objetivos de Automação em Nível Micro de Práticas de Gerenciamento de Configuração

Uma vez que o time seleciona, do nível intermediário, algumas práticas ágeis que pretende introduzir no time, as decisões em nível micro buscarão aprofundar práticas de gerenciamento de configuração com objetivos para a sua automação. Neste sentido, os seguintes perfis que podem ser aperfeiçoados pelos alunos:

- **Planejamento para Assimilação de Práticas Tratando de Testes de Unidade e Integração**. As práticas de tese de unidade e integração são essenciais para garantir que práticas de refatoração ocorram sem traumas para o time;

- **Planejamento para Assimilação de Práticas Tratando de Testes de Regressão**. Testes automatizados de regressão, utilizando o *Selenium* por exemplo, ou alguma alternativa para *Behavioural-Driven Development*, como *Cucumber*, são bastante eficientes para implantar um desenvolvimento evolutivo de software;

- **Planejamento para Assimilação de Práticas Tratando de Testes de *Staging* e de Aceitação**. Os testes de encenação podem ser positivos para se introduzir elementos de manutenção num estágio anterior aos testes de aceitação. Problemas de evolução de banco de dados podem explorar bem estas possibilidades;

- **Planejamento para Assimilação de Práticas Tratando de Automação de Evolução**. Aqui podem ser considerados tanto problemas que visem a evolução decorrente de novas *Sprints*, como evolução para modernização. Portanto, planejamentos podem ser realizados considerando tanto práticas de refatoração como de reengenharia;

- **Planejamento para Assimilação de Práticas Tratando de Gerenciamento de Mudanças de Requisitos de Software**. É fundamental que o time acompanhe e monitore as mudanças decorrentes de novas *Sprints* ou mesmo de novos requisitos, realizando a auditoria e o relato das mudanças;

- **Planejamento para Assimilação de Práticas Tratando de Gerenciamento de Versionamento**. É importante que o aluno seja capaz de exercitar tanto o versionamento de código como o versionamento de *Releases*. Gerenciamento de *releases*

podem explorar entregas de produtos testáveis pelo cliente final. Dependendo do nível de automação que o time consiga obter, *releases* podem até mesmo ser definidas e gerenciadas de modo contínuo;

- **Planejamento para Assimilação de Práticas Tratando de Automação de Construção do Software**. Práticas de visam alinhar as configurações para construção de software são fundamentais para que o time consiga implementar algum mecanismo de automação no processo de software;

- **Planejamento de Pipelines.** Um pipeline de Integração/Entrega Contínua consiste em uma série de passos para a entrega de uma nova versão de software, em um mesmo pipeline podem haver etapas de monitoramento e automação de etapas principalmente nos estágios de integração e teste. Os estágios comumente usados em pipelines são: 1) Compilação, 2) Teste, 3) Lançamento, 4) Implantação, e 5) Validação e conformidade.

- **Planejamento para a Integração Contínua.** A Integração Contínua consiste em integrar continuamente o código fonte em um repositório central e executar alguns testes a fim de encontrar e investigar *bugs* mais rapidamente, melhorar a qualidade do software e reduzir o tempo que leva para validar e lançar novas atualizações de software.

- **Planejamento para a Implantação Contínua.** Esta é uma pratica de *DevOps* que tem por objetivo automatizar as etapas de construção, testes e implementação ao máximo. Se a alteração no código passar com sucesso por todas as etapas do pipeline, essa mudança vai ser implantada automaticamente no ambiente de produção sem qualquer intervenção manual. Com isso, é possível entregar novos recursos aos usuários o mais rápido possível. A implementação continua é sustentada por uma integração continua bem testada e etapas de entrega continua. Pequenas alterações de código feitas de forma periódicas ao *branch master*. Após, são submetidas a um processo de automatização de construção e testes, passando por ambientes de pré-produção nas etapas de Integração e Entrega Contínua. Caso esta automação não encontre problemas, o serviço é implantado no ambiente de produção.

- **Planejamento para a Entrega Contínua.** A Entrega Contínua consiste em testar e carregar automaticamente em um repositório A, as alterações do código fonte feitas por uma equipe de desenvolvimento em uma aplicação, em um repositório B, onde podem ser implantadas em ambiente de produção em tempo real pela equipe de operações com objetivo de garantir o mínimo de esforço na implantação de novos códigos.

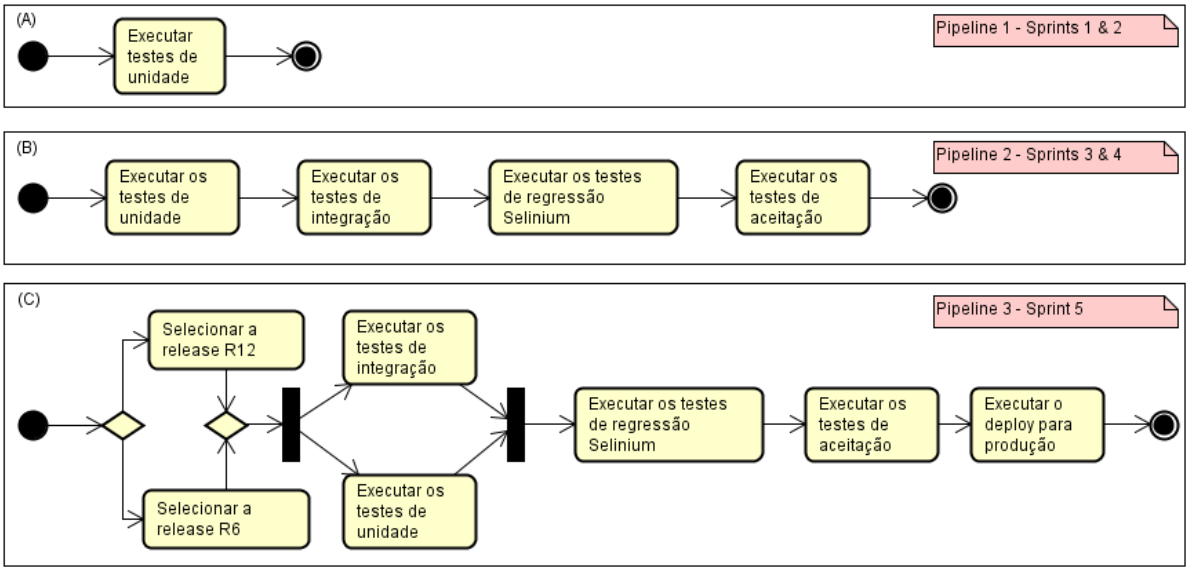

Figura 6 – Três cenários de pipelines de IC à serem introduzidos gradativamente na execução de disciplinas que envolvam atividades de gerência de configuração.

4.3 Delineamento de Problema de Automação de Processo

Para a resolução do problema anteriormente introduzido na evolução de banco de dados em um cenário de integração contínua, o primeiro passo é planejar cenários de integração em níveis de dificuldade, como fácil, médio e difícil. Também é preciso distribuir estes cenários para que englobem etapas da curva de aprendizado dos alunos em uma disciplina de RP V, que contam com 5 *Sprints* de três semanas. Isso precisa ser realizado em conjunto com professores que já ministraram esta disciplina.

De momento, propõem-se um desafio composto de três pipelines de integração contínua, como ilustrados na Figura 6. São três exemplos de configuração mapeados para níveis de dificuldade, sendo o primeiro, Figura 6 (A), o mais simples e o terceiro, Figura 6 (C), o mais complexo.

Também foram sugeridas as *Sprints* para a execução de cada nível e as atividades que compõe os pipelines. Uma vez que testes de aceitação são tarefas manuais, a última atividade do nível médio, Figura 6 (B), requer o uso de gatilhos para concluir o pipeline, além da configuração de três bancos de dados (*development*, *staging* e *production*). Já o nível difícil inclui atividades executadas em paralelo, além do uso de gerenciamento de *releases* no sistema de controle de versão adotado, e da implantação automática da aplicação em ambiente de produção, como requerido em práticas de entrega contínua.

A Tabela 9 descreve uma sequência de tarefas para configurar minimamente um servidor de Integração Contínua. Ou seja, trata do cenário fácil. Este servidor leva em conta um projeto de software semelhante aos que os alunos de RP V 2018 utilizaram, cujas tarefas de teste e construção são especificadas em *Java Maven*. Cabe salientar que, mesmo neste nível, a turma de RP V que mais se dedicou ao aprendizado de IC não foi

Tabela 9 – Tarefas no cenário mais simples para configurar um servidor de IC

Tarefa 1 (Setar quais são as ferramentas necessárias para a construção do projeto) Nessa tarefa são definidas as ferramentas necessárias para a construção do projeto, como por exemplo no *CircleCI* é setado a versão do *JDK* e do *MySQL* pelo próprio arquivo de configuração e no *Jenkins* e no *Hudson CI* é feita pela própria ferramenta.
Tarefa 2 (Importar o projeto para o servidor de IC) Nessa tarefa o projeto deve ser importado para os servidores de IC, no caso do *Jenkins* e do Hudson o servidor de IC fica na sua máquina, porém no caso do *CircleCI* para importar o projeto é necessário apenas autorizar o acesso ao repositório *Bitbucket* ou *GitHub* ou *GitLab*.
Tarefa 3 (Criar o arquivo de configuração) Nessa tarefa são criados os arquivos de configuração e definidos neles os testes a serem executados.
Tarefa 4 (Fazer um *commit* inicial) Nessa tarefa é feito o primeiro *commit* que passará pelos servidores de Integração Contínua mostrando os primeiros resultados e verificando se o servidor foi configurado corretamente.

capaz de realizar todas as tarefas previsas nesta tabela.

Outro ponto que deve ser considerado é o controle das *releases* do software, e este controle afeta diretamente como os alunos irão construir os pipelines de automação. O gerenciamento de *release* apresentado em (SOMMERVILLE, 2010) é composto por três etapas:

- A primeira é a tomada de decisão para uma *release*, etapa em que ela é preparada;
- A segunda etapa é a criação da *release*, que envolve o desenvolvimento de arquivos e documentos incluindo todos os seus componentes, e;
- A terceira etapa consiste na documentação da *release*, ou seja, registrar as versões específicas dos componentes de código-fonte usados para criar o código executável.

Este é um exemplo do que se pode explorar em termos de problema para superar os gargalos de gerência de configuração identificados. Ou seja, o material proposto serve tanto para alunos utilizarem na configuração de pipelines de integração continua, em três níveis de dificuldade organizados de modo gradual, como pelos professores, que podem fornecer recomendações na introdução de práticas de IC ao longo das *Sprints* de um processo de desenvolvimento que envolva *PBL*.

Cabe salientar que esta estratégia proposta é um trabalho em andamento. Assim, outras tarefas precisam ser bem elaboradas no escopo de um exemplo de nível médio e difícil, conforme a percepção dos professores na assimilação dos conceitos pelos alunos em sucessivas execuções da disciplina.

4.3.1 Objetivos de Automação em Nível Micro para IC e CD

Uma vez que objetivos de automação macro (práticas gerais) e objetivos intermediário (de práticas ágeis) tenham sido definidas e exploradas pelos time e tutor, o próximo passo é buscar alinhar os objetivos de aprendizado com a automação de pipelines de integração contínua e implantação contínua, que atende um objetivo de nível micro.

A seguir, apresenta-se um estudo exploratório, conduzido com o seguinte objetivo: Compreender as semelhanças e diferenças na sintaxe de servidores de IC, de modo à caracterizar recursos disponíveis, e esboçar níveis de dificuldades de configurações em três pipelines que devem ser utilizados na criação de uma estrategia de automação. Tal material serve como suporte para os times realizarem o objetivo de nível micro: *Planejamento para a Implantação Contínua*.

4.3.1.1 Circle CI

O primeiro servidor/serviço de Integração Contínua analisado foi o *Circle CI*, a fonte usada para estudo foi a documentação do CircleCI (CIRCLECI, 2019). Segundo a sua documentação, ele se destaca com um alto desempenho, um controle completo de como deseja trabalhar e sua flexibilidade incomparável. Para a configuração do servidor CircleCI, faz-se necessária a criação de uma pasta na raiz do projeto com nome de *.circleci* usando um arquivo de configuração chamado *config.yml*. Portanto, a configuração do servidor é feita dentro desse arquivo.

A primeira configuração a ser feita no arquivo é determinar qual versão no *CircleCI* utilizar. Nesse trabalho, foi usada a versão 2.1 por ser a última versão estável. Logo após, usa-se a tag *jobs* para definir seus trabalhos. Foram usados quatro configurações de trabalhos diferentes, sendo eles: 1) *build*; 2) *test1*; 3) *hold* e 4) *deploy*, como podem ser vistos na Figura 7

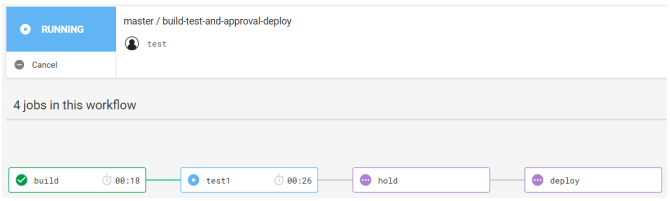

Figura 7 – Fluxos de Trabalho sendo executados no servidor/serviço de IC do CircleCI após um commit

No trabalho denominado *build*, o projeto Java Maven foi construído e variáveis de ambiente, como de versão do banco de dados, foram setadas para a construção do projeto. Em cada trabalho é necessário setar estas configurações. Após, o projeto é construído, o banco de dados é inicializado e populado, como pode ser visto na Figura 8.

No trabalho denominado *test1*, como pode ser visto na Figura 9, primeiramente é setada a imagem à ser usada nessa etapa. Em seguida, utiliza-se a tag *working_directory*

```
version: 2.1
jobs:

  build: #primeiro trabalho que inicializa as coisas
    docker:
      - image: circleci/openjdk:8-jdk #imagem do jdk 8
        environment: #variaveis de ambiente, nesse caso os dados do banco de dados
          MYSQL_HOST: 127.0.0.1
          MYSQL_DB: mydb
      - image: circleci/mysql:5.7 #imagem do mysql 5.7
        command: mysqld --character-set-server=utf8mb4 --collation-server=utf8mb4_bin --innodb
        environment:
          MYSQL_USER: root
          MYSQL_ALLOW_EMPTY_PASSWORD: true
    steps:
      - checkout
      - run: sudo apt install -y mysql-client #install mysql client
      - checkout
      - run: mysql -u root < Gamming/Documentação/Sprint\ 4/Banco\ de\ Dados/mydb_completo.sql
      #Populando banco
```

Figura 8 – Trabalho Build sendo configurado dentro do arquivo de configuração do servidor/serviço CircleCI

para determinar o repositório padrão para aquele trabalho. Depois, as dependências do projeto são construídas e o local de carregamento/armazenamento dos artefatos do CircleCI são setados. Por último, os testes são executados.

```
test1: #Primeira bateria de testes
  docker:
    - image: circleci/openjdk:8-jdk #JDK dnv pois se trata de outro trabalho(job)
  working_directory: ~/unipampa-ales-rpv-g2 #setando diretorio padrão
  steps: #etapas a serem executas nos tests

    - checkout

    - run: cd Gamming/RPV-G2-Gamming2 && mvn dependency:go-offline #Contruindo

    - store_test_results: #carrega os metadados de teste no "target" diretório para que ele possa ser exibido
        path: Gamming/RPV-G2-Gamming2/target/surefire-reports

    - store_artifacts: #armazena um uberjar como um artefato no "target"
        path: Gamming/RPV-G2-Gamming2/target/RPV-G2-Gamming2-1.0-SNAPSHOT.jar


    # Executa os testes, já que não tá no diretório do working_directory faz-se necessário usar o cd ali
    - run: cd Gamming/RPV-G2-Gamming2 && mvn clean test
```

Figura 9 – Trabalho Test1 sendo configurado dentro do arquivo de configuração do servidor/serviço CircleCI

No trabalho denominado *hold*, é configurado um trigger. Assim, assuma que para que todo commit, o trabalho precise ser aceito manualmente para que esta etapa seja dada como concluída. Caso o time não aceite o commit, esta atividade permanece em espera. Este exemplo é aplicável em cenários de teste de aceitação, incluindo no pipeline um trabalho do tipo *hold*.

Por último no trabalho *deploy* o commit do código fonte é aceito e a aplicação é implantada em determinado servidor de aplicações. Essas duas etapas não possuem uma configuração extra e fazem apenas isso: 1) aguardam uma confirmação; e 2) realizam uma implantação.

As tags usadas no arquivo de configuração, que podem ser vistas na Figura 10, são as mínimas configurações necessárias para que uma etapa de trabalho possa ser setada. Caso o pipeline necessite mais configurações específicas para cada projeto, outras tags podem ser colocadas nessas etapas.

O CircleCI fornece também a possibilidade de definir o fluxo de trabalho, chamado de pipeline. Usando a tag *workflows*, como pode ser visto na Figura 11, é possível definir a ordem de execução dos trabalhos e suas dependências. Neste exemplo, foi configurado um pipeline para executar o trabalho *build*, depois o *test1*, o *hold* e por último o *deploy*. Assim, cada trabalho depende do seu anterior no pipeline. Caso algum trabalho não

```
hold: #Trabalho que exige a aceitação manual do usuário(Simulando o teste de aceitação)
  docker:
    - image: circleci/openjdk:8-jdk
  steps:
    - run: cd Gamming/RPV-G2-Gamming2 && ls #Aqui deveria vim qualquer coisa que faz-se necessário

deploy:
  docker:
    - image: circleci/openjdk:8-jdk
  steps:
    - run: ls #Aqui deveria vim qualquer coisa que faz-se necessário durante essa etapa
```

Figura 10 – Trabalho Hold e Deploy sendo configurado dentro do arquivo de configuração do servidor/serviço CircleCI

consiga cumprir seu objetivo ou algum teste falhe, o próximo trabalho do pipeline não será executado.

```
workflows: #Fluxos de trabalho, no caso vai executar um de cada vez sendo que o próximo sempre é dependente
  version: 2
  build-test-and-approval-deploy:
    jobs: #executando na ordem build, test1, hold e deploy com os requerimentos
      - build
      - test1: # Customização dos trabalhos(nesse caso adição dos requires)
          requires: # test1 não será executado caso o build não tenha sido executado.
            - build
      - hold: # Aqui é o teste de aceitação
          type: approval
          requires:
            - test1
      - deploy: #executando a última etapa, que no caso é o deploy
          requires:
            - hold
```

Figura 11 – Fluxos de trabalho sendo configurados para o servidor/serviço CircleCI

Outro ponto importante é a utilização da tag *type*, ilustrada na Figura 12. Esta segue com o valor *approval*, que determina que o trabalho de *hold* precisa ser aceito manualmente, para que o fluxo do pipeline continue sua execução.

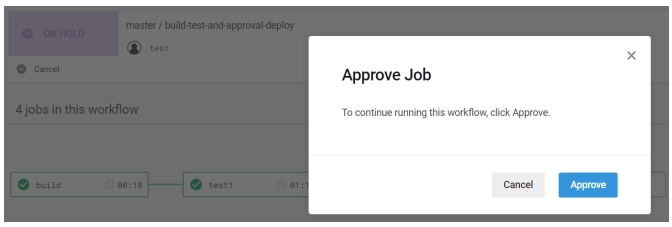

Figura 12 – Aceitando manualmente o fluxo de trabalho hold no servidor/serviço CircleCI

4.3.1.2 Hudson CI

O segundo servidor/serviço de Integração Contínua analisado foi o Hudson CI. A fonte usada para estudo foi a documentação do Hudson CI (HUDSON, 2019). Em relação ao CircleCI, o Hudson CI possui menos funcionalidades, menor flexibilidade e alguns trabalhos são engessados. Por isso, esse servidor/serviço de IC não foi bem avaliado.

Um ponto positivo em termos de sintaxe de pipelines é que o léxico do Hudson CI é bem organizado, e segue a seguinte ordem: 1) Job; 2) Build; 3) Artifact; 4) Workspace; 5) Status e 6) Trend. O primeiro termo é onde se pode definir os trabalhos a serem executados, de forma bem semelhante ao CircleCI, porém sem paralelismo e teste de aceitação. Por ter uma etapa definida para o Build, Hudson CI também não possibilita setar trabalhos deste tipo.

O segundo termo remete-se a construção do projeto. No caso dos alunos de RP V 2018/1, já foi usado para construção um projeto Java Maven para build e configuração de test suites do projeto. O terceiro termo é onde os artefatos são gerados. O quarto termo define onde o projeto será implantado e onde serão armazenados os artefatos. O quinto termo é sobre os status dos trabalhos, que podem ser classificados como Bem sucedido (tudo ocorreu bem), Instável (a compilação foi bem sucedida porém há falhas), Falhou (a construção falhou) e Desativado (O projeto nunca foi construído antes ou está desativado).

É possível criar trabalhos no Hudson CI. Usando a tela como a mostrada na Figura 13, cria-se o tipo de trabalho mais usado segundo (CI, 2017), que é o "Build a free-style software project". Caso já exista um projeto semelhante à uma compilação existente do Hudson CI, é possível selecionar a opção "Copy an existing job". Após selecionado, a tela respondente é mostrada na Figura 14, onde selecionam-se os detalhes da criação do novo trabalho.

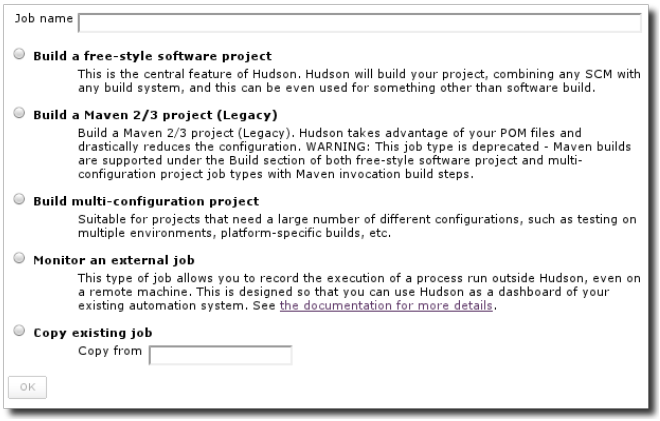

Figura 13 – Criando um novo trabalho na interface gráfica do Hudson CI

Para a configuração do servidor, faz-se necessário primeiro definir suas configurações globais, como mostradas na Figura 15: 1) Home Directory; 2) System Message; 3) of Executors; 4) Quiet period; 5) SCM checkout retry count; 6) Enable Security; 7) Prevent Cross Site Request Forgery exploits e 8) Help make Hudson better.

O primeiro parâmetro o caminho de instalação absoluto do sistema Hudson atualmente em execução e não é configurável em tempo de execução. O segundo parâmetro é a mensagem é exibida por Hudson na tela principal acima da lista de projetos(mensagem de boas vindas basicamente). O terceiro controla o número de compilações simultâneas que o Hudson está configurado para executar. O quarto define um número especifico de segundos antes que uma construção acionada seja iniciada. O quinto determina o número de tentativas que o Hudson faz para verificar quaisquer atualizações ao pesquisar o sistema SCM em busca de alterações e definir o sistema como indisponível. O sexto é uma caixa de seleção que se assinalada exigirá nome de usuário e senha para qualquer acesso para executar compilações ou alterar configurações do Hudson e construir projetos. O

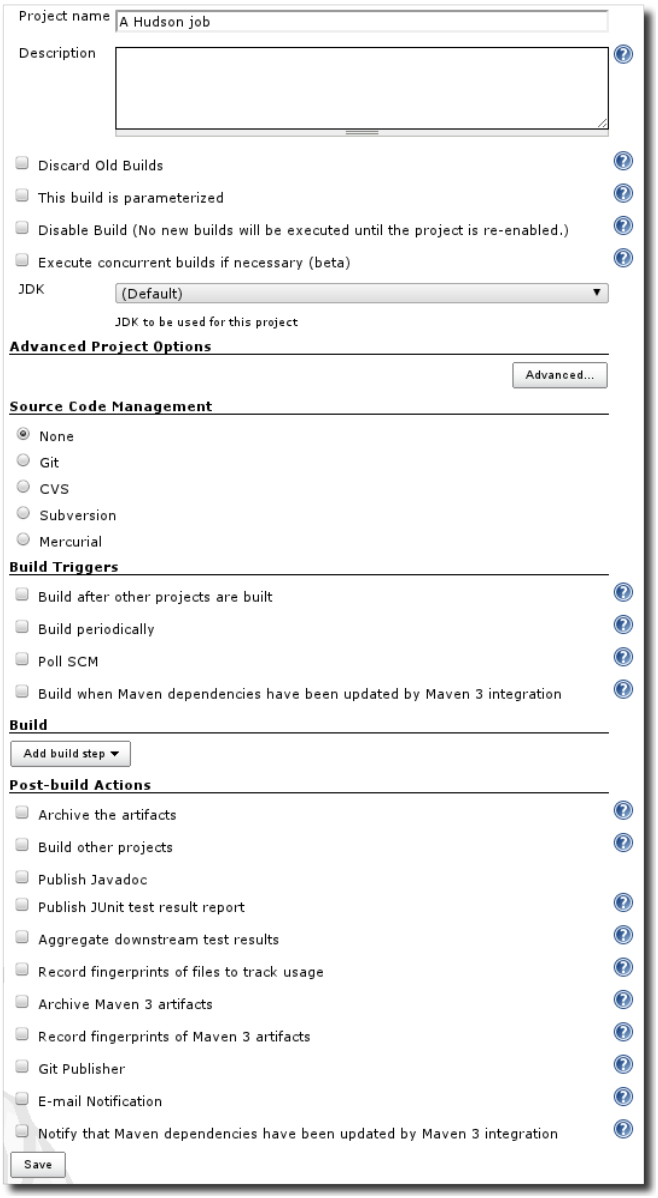

Figura 14 – Resultado da seleção do trabalho na interface gráfica do Hudson CI

sétimo permitirá a segurança aprimorada contra explorações de Cross Site *Request Forgery* e é recomendável que seja ativado quando a instância do Hudson estiver disponível publicamente na internet. O oitavo e último, é para concordar que suas estatísticas de uso anônimas sobre a instalação do Hudson sejam criadas e enviadas com segurança à equipe de desenvolvimento do Hudson e disponibilizadas para a comunidade de usuários.

Após faz-se necessário configurar o JDK da aplicação, como mostrado na Figura 16. Depois, a instalação do configuração do Maven, como mostrado na Figura 17. A próxima etapa é para configurar notificações por e-mail, porém ela não será abordada neste documento. Logo após, o arquivo Maven deve ser configurado apenas setando o arquivo "pom.xml". Por último, configure as exibições de listas de projetos globais e individuais da sua preferência.

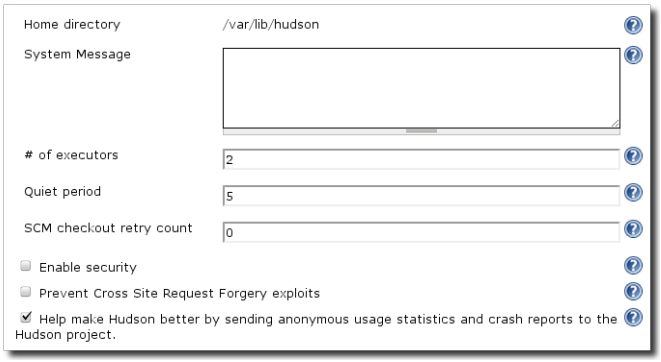

Figura 15 – Setando as configurações globais do Hudson CI

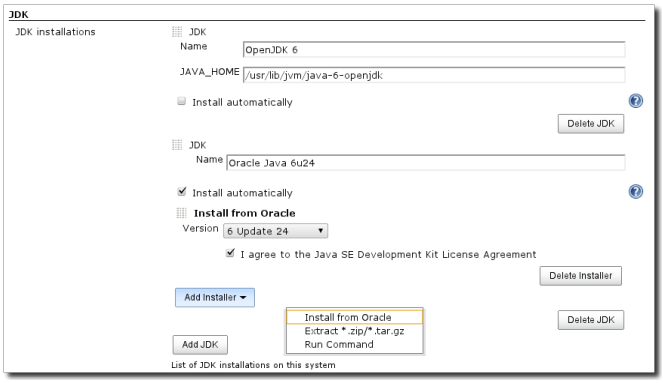

Figura 16 – Configurando JDK na interface gráfica do Hudson CI

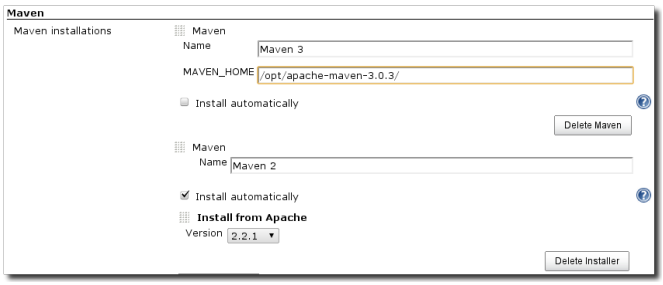

Figura 17 – Configurando Maven na interface gráfica do Hudson CI

4.3.1.3 Jenkins

O terceiro servidor/serviço de Integração Contínua analisado foi o Jenkins. A fonte usada para estudo foi a documentação do Jenkins (JENKINS, 2019). Em comparação ao CircleCI, o Jenkins possui todas as funcionalidade apresentadas. Jenkins também tem suporte a paralelismo e *"Workflows"*, assim como o CircleCI, porém se destaca na sua ampla variedade de plugins suportados.

Os principais plugins encontrados (LIMA, 2017) foram: 1) *JUnit Plugin*; 2) *Test Results Analyser* e 3) *Failure Cause Manager*. O primeiro plugin apresenta um gráfico simples, porém muito útil, da tendência dos testes automatizados. Ela é obtida com base nas últimas execuções do *Job*. Já que ele aparece logo após a inicializar o Jenkins, é

possível ter uma visão geral rápida sobre o número de testes que passaram, falharam e foram executados. O segundo mostra de forma fácil e simples os resultados tabulares de todos os testes em todos os *Builds* de um *Job*. O terceiro permite registrar no Jenkins as causas na qual um Job ou testes falharam. Apesar da diversidade de plugins, nenhum foi usado nesta análise.

Para a criação e configuração do arquivo de configuração do Jenkins foram definidos três trabalhos e também o contêiner à ser usado para a execução, como pode ser visto na Figura 18. O container foi definido dentro da tag "agente"pela tag "docker"onde é usada a imagem da versão 3 do Maven. Os trabalhos são nomeados pelo Jenkins como estágios. Porém, para facilitar o entendimento, considerou-se estágios e trabalhos como sendo a mesma coisa.

No trabalho "Build", como mostrado novamente na Figura 18, é construído o projeto. No trabalho "Test", como mostrado na Figura 19, é definido um novo trabalho em que os testes são executados. Logo após, define-se o local onde o relatório JUnit XML deve ser guardado. Caso não seja definido o local, o relatório não é armazenado.

Por último, como mostrado na Figura 20, é criado o trabalho Deliver. Este executa o script de Shell "deliver.sh". Para esse trabalho, nada foi setado dentro do script. Para fins ilustrativos, apenas a função foi apresentada. Já para trabalhos específicos, deve-se especificar a execução de um script de Shell no Jenkins.

```
agent {
    docker {
        image 'maven:3-alpine' 1
        args '-v /root/.m2:/root/.m2' 2
    }
}
stages {
    stage('Build') { 3
        steps {
            sh 'mvn -B -DskipTests clean package' 4
        }
    }
}
```

Figura 18 – Configurando contêiner a ser usado e o trabalho Build dentro do arquivo de configuração do servidor/serviço Jenkins

```
stage('Test') {
    steps {
        sh 'mvn test'
    }
    post {
        always {
            junit 'target/surefire-reports/*.xml'
        }
    }
}
```

Figura 19 – Configurando o trabalho Test dentro do arquivo de configuração do servidor/serviço Jenkins

```
stage('Deliver') {
    steps {
        sh './jenkins/scripts/deliver.sh'
    }
}
```

Figura 20 – Configurando o trabalho Deliver dentro do arquivo de configuração do servidor/serviço Jenkins

4.3.2 Recomendação de Ferramental

Esta seção faz uma recomendação aos times, porém a definição do servidor de integração contínua também dependerá da escolha do aparato ferramental utilizado para gerenciar o versionamento dos códigos. Dos servidores/serviços de IC apresentados, o *CircleCI* se destacou por sua facilidade de uso e flexibilidade. Isto se deve, principalmente, pelo fato de não haver necessidade de instalar nenhuma ferramenta, já que para a configuração basta apenas adicionar seu projeto na aplicação Web no *CircleCI*, de adicionar o arquivo de configuração a raiz do repositório. Além disso, o *CircleCI* fornece uma ótima interface em sua aplicação WEB que também é bem simples de usar. Porém o *Jenkins* não fica atrás e fornece uma ampla variedade de *Plugins* a serem usados, o que abre novas possibilidades e mais funcionalidades do que as presentes no *CircleCI*, justificando assim ser mais difícil usar a ferramenta. Já o Hudson não se destacou em nada, é uma ferramenta antiga e não se recomenda a sua utilização.

5 AVALIAÇÃO

O Apêndice A apresenta os dois estudos conduzidos ao longo do TCC 1, que serviram como base de motivação para a definição de uma estratégia de ensino e aprendizagem de Integração Contínua em futuras disciplinas de RP V. Este capítulo apresenta um pequeno relato de um estudo em andamento, que busca avaliar se a estratégia de ensino e aprendizagem contribuiu para a disciplina de RP V conduzida em 2021/1.

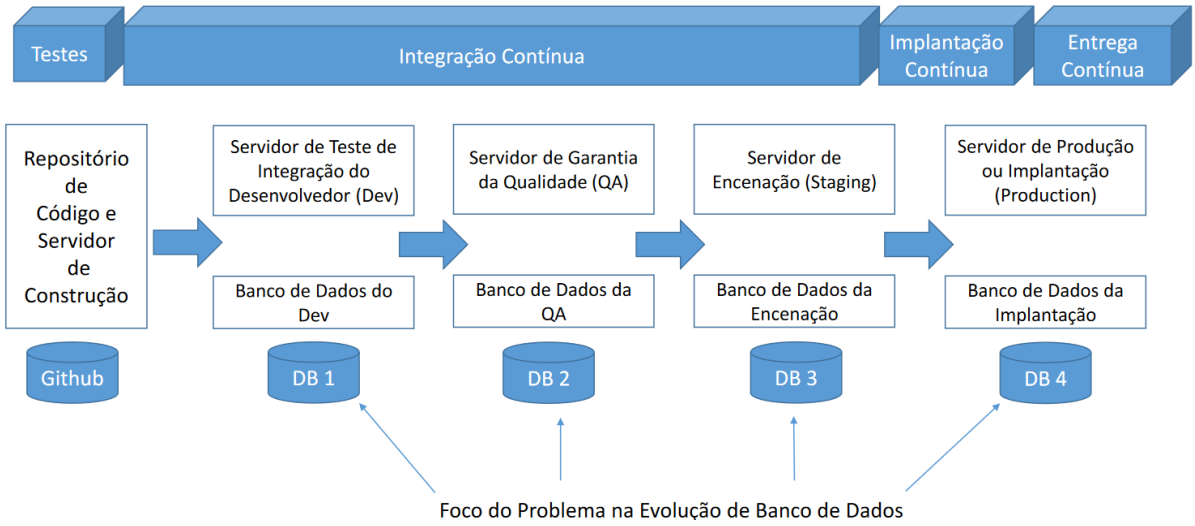

Figura 21 – Estratégia introduzida em Resolução de Problemas V 2021/1

Durante a disciplina de Resolução de Problemas V foi aplicado um *Survey* com questões de caracterização focadas em integração e implantação contínua para iniciar em automação de processos de evolução de software. Para este *Survey* foram definidas treze questões, sendo a primeira delas apenas sobre os dados gerais onde o aluno deveria preencher seu nome e matrícula. As outras questões serão discutidas nas seções abaixo, porém uma ressalva seria que da terceira a décima terceira questão as opções de resposta são apenas cinco, sendo elas Discordo totalmente, Discordo parcialmente, Nem concordo nem discordo (indiferente), Concordo, Concordo fortemente. Nos gráficos apresentados abaixo para reduzir o tamanho da opção "Nem concordo nem discordo (indiferente)"a mesma foi alterada para "Indiferente".

5.1 Disciplinas de Resolução de Problemas já cursadas pelos entrevistados

A segunda questão "Disciplinas de Resolução de Problemas Já Concluídas"tem como objetivo conhecer um pouco mais sobre a jornada do aluno e quais disciplinas de Resolução de Problemas ele já concluiu. Como pode ser visto na figura 22 todos os entrevistados por exceção um cursaram as disciplinas de Resolução de Problemas I, II e III e que 61,54% dos alunos cursaram a disciplina Resolução de Problemas IV dando a entender que boa parte deles possui uma base referente aos RP's anteriores ao RPV.

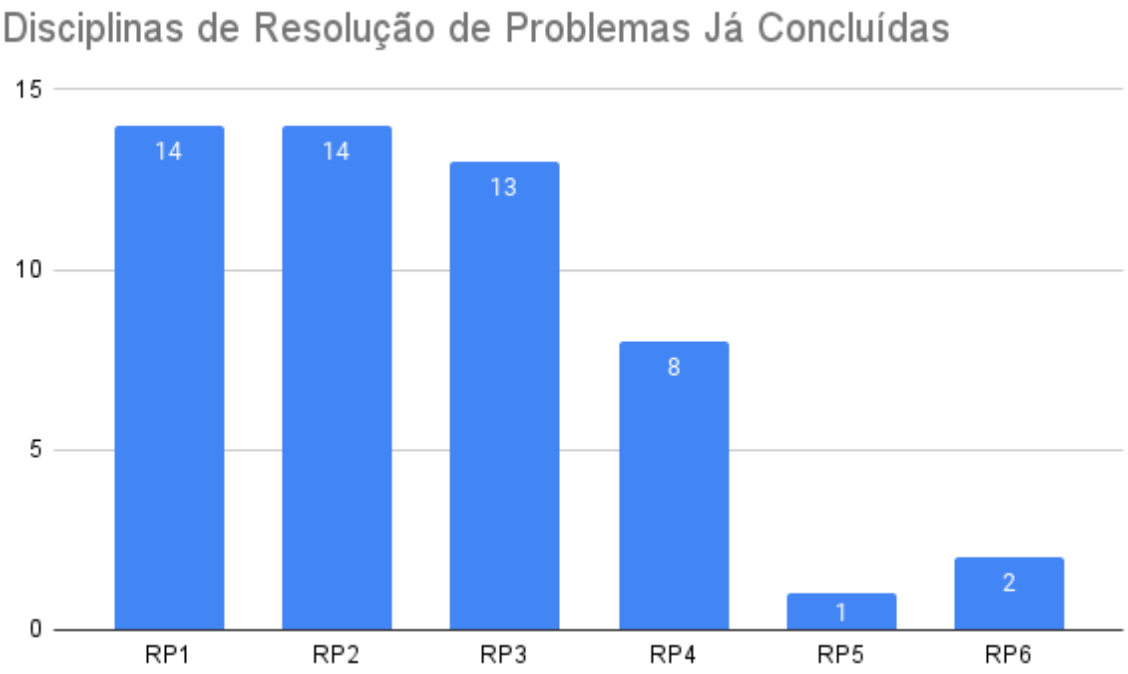

Figura 22 – Disciplinas de Resolução de Problemas Já Concluídas

5.2 Gerência de Configuração

A terceira questão "As disciplinas anteriores forneceram a base necessária de "gerência de configuração"para utilizar uma ferramenta de versionamento de código (como o GIT, SVN ou CVS)"tem como objetivo entender se as disciplinas anteriormente cursadas pelo entrevistado fornecem a base necessária a utilização de uma ferramenta de versionamento de código. Como pode ser visto na 23 a maior parte dos alunos (78,6%) Concorda Fortemente ou Concorda que a disciplinas anteriores fornecem essa base, além dos 14,3% indiferente. Baseado nesses dados pode-se concluir que as disciplinas anteriores forneceram a base necessária de Gerência de Configuração para que os alunos consigam utilizar ferramentas de versionamento de código.

A quarta questão "As disciplinas anteriores forneceram a base necessária de "gerência de configuração"para utilizar uma ferramenta de automação da implantação (deploy) do software (como ANT ou MAVEN)"tem como objetivo validar se as disciplinas anteriormente cursadas pelo aluno forneceram a base necessária para a utilização de uma ferramente de automação de implantação de software. Como pode ser visto na 24 35,7% dos alunos Discorda totalmente ou parcialmente sobre as disciplinas anteriores terem fornecido essa base, apesar de somando os que concordaram totalmente, ou parcialmente também temos o valor de 35,7%. Por fim se tem 28,6% dos entrevistados respondendo com Indiferente.

Figura 23 – As disciplinas anteriores forneceram a base necessária de "gerência de configuração

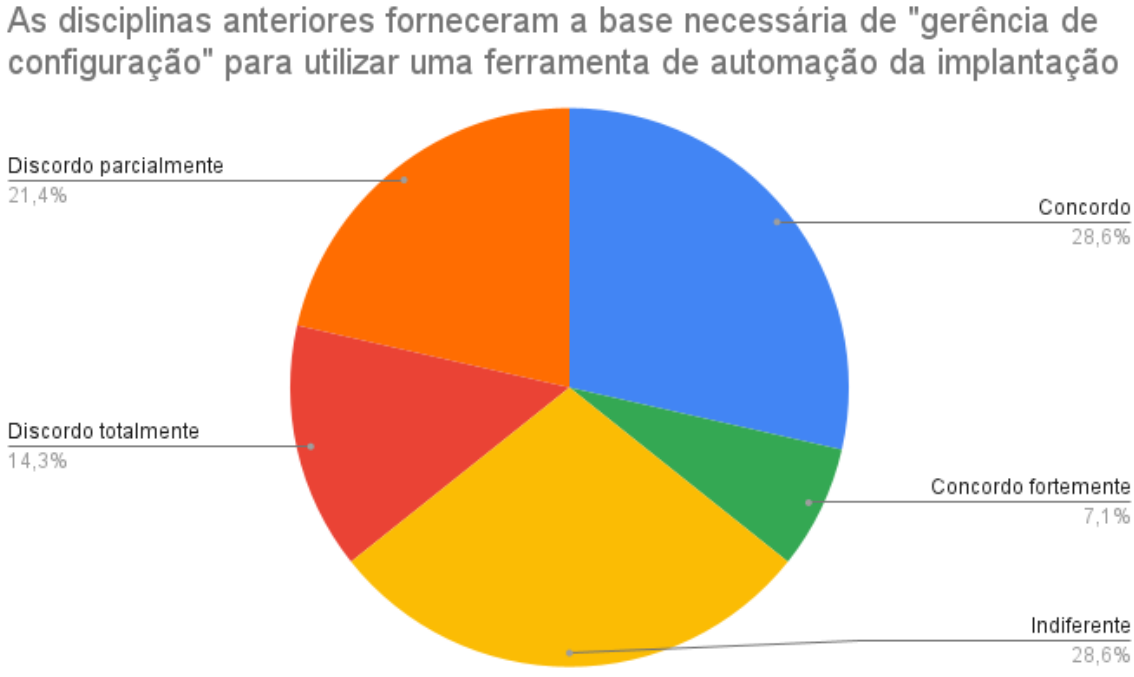

Figura 24 – As disciplinas anteriores forneceram a base necessária de "gerência de configuração"para utilizar uma ferramenta de automação da implantação (deploy) do software (como ANT ou MAVEN)

Figura 25 – As disciplinas anteriores forneceram a base necessária para desenvolver e automatizar testes de unidade

5.3 Desenvolver e Automatizar Testes

A quinta, sexta, sétima e oitava questão teve como objetivo validar se os alunos possuem a base necessária para o desenvolvimento de testes, sendo eles testes de Unidade, Integração, Regressão e Aceitação respectivamente apresentados em 25, 26, 27, 28. Referente a base para os testes de Unidade e Integração mais de 70% dos entrevistados Concordou, Concordou Fortemente ou foi Indiferente e a maior parte Concordou ou Concordou fortemente, baseado nisso conclui-se que para os testes de Unidade e Integração a maior parte dos entrevistados considerou que as disciplinas anteriores forneceram essa base. Porém para os testes de Regressão e Aceitação nenhum entrevistado marcou a opção "Concordo Fortemente"e especificamente sobre o teste de Regressão a maior parte Discordou fortemente ou parcialmente concluindo assim que seriam pontos a serem revisados ou introduzidos.

5.4 Gerencia de Alteração de Software e Mudanças de Requisitos

A nona questão "As disciplinas anteriores forneceram a base necessária para gerenciar alterações no software decorrentes de sua evolução incremental"tem como objetivo verificar se as disciplinas anteriormente cursadas pelo entrevistado forneceram a base necessária para gerenciar alterações de software decorrentes de sua evolução incremental que naturalmente ocorrerá na disciplina de Resolução de Problemas V. Como visto na 29 a

Figura 26 – As disciplinas anteriores forneceram a base necessária para desenvolver e automatizar testes de integração

Figura 27 – As disciplinas anteriores forneceram a base necessária para desenvolver e automatizar testes de regressão

maioria dos entrevistados (57,2%) Concorda ou Concorda Fortemente que as disciplinas

Figura 28 – As disciplinas anteriores forneceram a base necessária para desenvolver e automatizar testes de aceitação

anteriores forneceram essa base.

A décima questão "As disciplinas anteriores forneceram a base necessária para gerenciar mudanças de requisitos de software"verificar se o aluno cursou anteriormente disciplinas que forneceram a base necessária para gerenciar mudanças nos requisitos de software. Baseado no apresentado na 30 grande parte dos correspondentes Concordou que as disciplinas anteriores forneceram essa base.

Baseado na nona e décima questão conclui-se que a maioria dos entrevistados tem a base necessária para gerenciar alterações de software e mudanças de requisitos apesar do valor de 21,5% de correspondentes que marcaram a opção Indiferente para a nona questão.

5.5 Auditar e gerar relatórios de mudanças de requisitos

A décima primeira questão "As disciplinas anteriores forneceram a base necessária para realizar a auditoria e relatório das mudanças de requisitos"verifica se os alunos cursaram disciplinas que proveram a base necessária para auditar e produzir relatórios das mudanças de requisitos solicitadas. Baseado no apresentado em 31 71,4% dos entrevistados concordaram que as disciplinas anteriores proveram essa base, baseando-se nisso conclui-se que os alunos possuem a base necessária auditar e gerar relatórios de mudanças de requisitos.

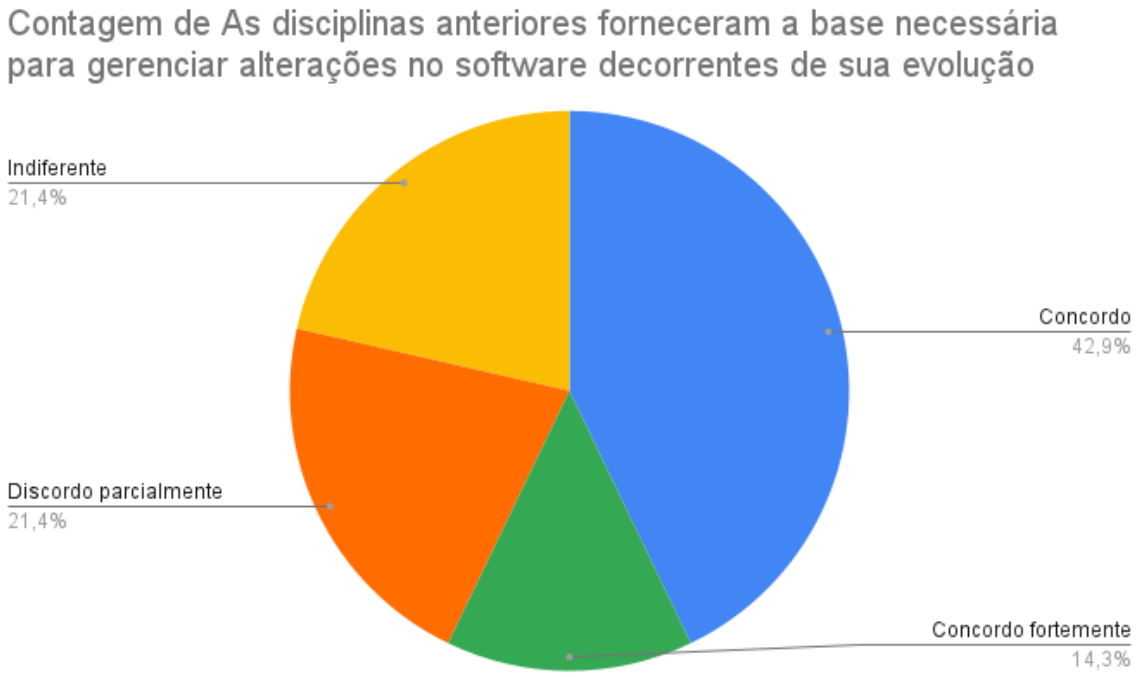

Figura 29 – As disciplinas anteriores forneceram a base necessária para gerenciar alterações no software decorrentes de sua evolução incremental

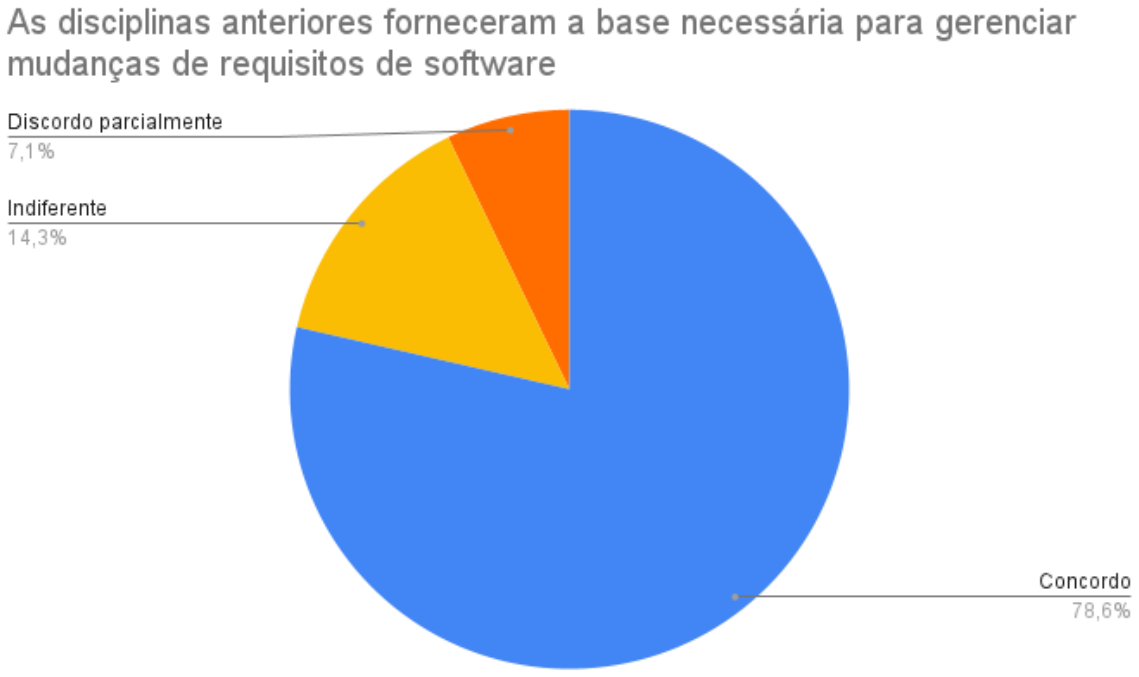

Figura 30 – As disciplinas anteriores forneceram a base necessária para gerenciar mudanças de requisitos de software

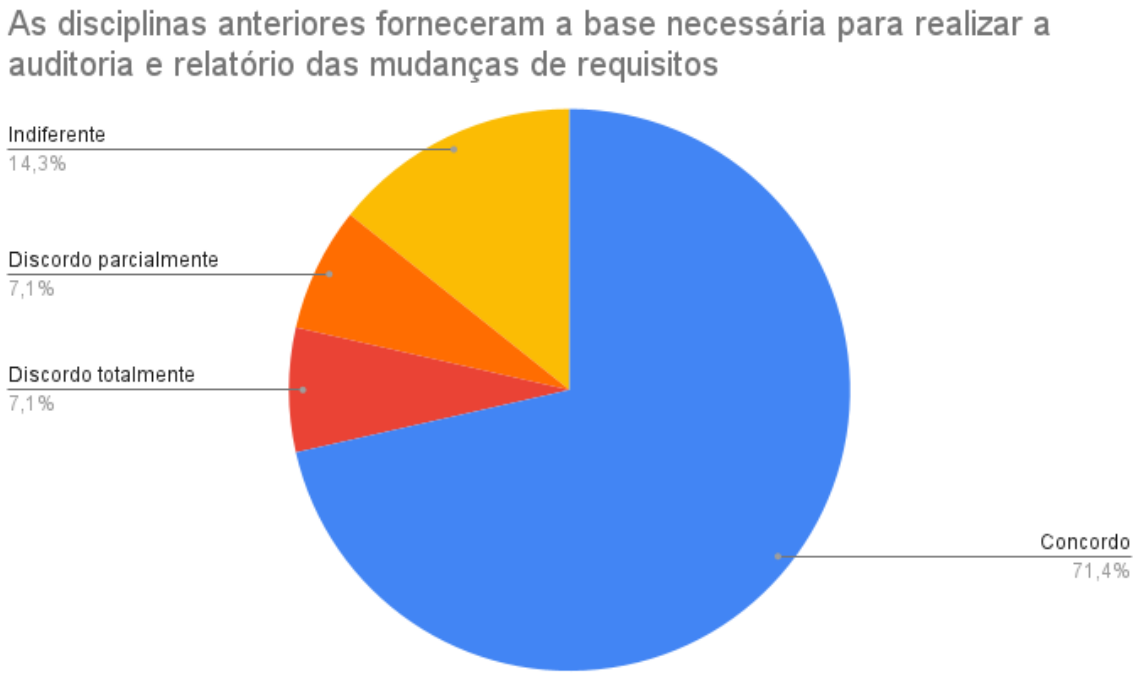

Figura 31 – As disciplinas anteriores forneceram a base necessária para realizar a auditoria e relatório das mudanças de requisitos

5.6 Tenho interesse em me aprofundar em leituras em material específico do tema de integração e implantação contínua

A décima segunda questão "Tenho interesse em me aprofundar em leituras em material específico do tema de integração e implantação contínua"foi proposta para entender o interesse dos entrevistados em se aprofundarem no tema de Integração e Implantação Contínua. Baseado no visto em 32 28,5% dos entrevistados discordaram parcialmente ou fortemente, porém 21,4% dos correspondentes marcaram Indiferente.

5.7 Comentários gerais e observações sobre cada questão

A última questão "Comentários gerais e observações sobre cada questão"tinha como objetivo abrir um espaço para comentários dos entrevistados porém não teve nenhuma resposta.

5.8 Avaliação da Estratégia

Está sendo executado um estudo de caso sobre a aplicação da estratégia que será finalizada dia 05 de outubro de 2021, por conta disso o resultado do estudo de caso será relatado posteriormente a essa data.

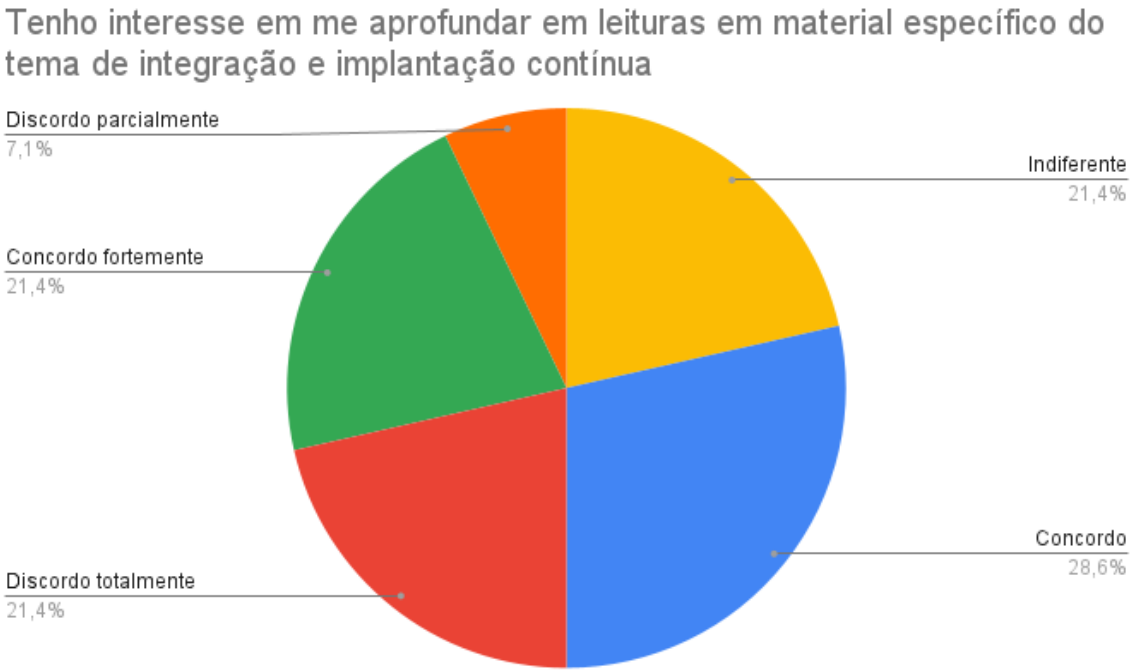

Figura 32 – Tenho interesse em me aprofundar em leituras em material específico do tema de integração e implantação contínua

5.9 Conclusões preliminares da Avaliação

Baseado no apresentado nas seções anteriores conclui-se que a maior parte dos alunos possui as bases necessárias para integração e implantação contínua em automação de processos de evolução de software. Sendo assim apesar de algumas ressalvas como no caso dos testes de Regressão e Aceitação um material específico do tema de integração e implantação contínua com os principais conceitos para a configuração de um pipeline de Integração e Implantação Contínua provavelmente seria benéfico para o aprendizado dos alunos.

6 CONCLUSÕES E TRABALHOS FUTUROS

Há grande dificuldade para estudantes de Engenharia de Software compreender os requisitos e assimilarem os conceitos de integração contínua em ambiente acadêmico. Primeiramente, não existem soluções "plug-and-play"para integração de recursos em servidores de IC, já que cada projeto de software demanda uma configuração diferente. Além disso, os três servidores estudados também apresentam diferenças para a configuração de pipelines. Somando à isso também se observou que, mesmo dedicando um tempo para a pesquisa desses servidores, o conhecimento especializado dessas ferramentas não é sistematizado nos materiais com cunho didático disponíveis na web. Ou seja, ele permanece tácito na mente dos desenvolvedores. Em adição, por meio de uma entrevista, identificou-se muitos elementos que tornam difícil a configuração destes serviços. Portanto, entende-se que estas dificuldades são um grande obstáculo para a formação de gerentes de configuração em nosso curso de graduação.

Uma vez demonstrado que configurar um servidor/serviço de Integração Contínua não é uma tarefa fácil nem trivial, abre-se espaço para contribuições que resolvam estas dificuldades. Para superar este obstáculo, é preciso primeiro sistematizar as atividades para configuração desses serviços. Em um segundo momento, pode-se buscar um nível mais alto de representação de pipelines do que as implementações disponíveis nos servidores de IC. Portanto, uma vez que superar tais dificuldades é de grande valia para o curso de engenharia de Software, o advento destes dois produtos pode contribuir com a redução da curva de aprendizado em futuras disciplinas de RP V.

Para caracterizar o problema, um survey foi aplicado em turmas de RP V, espaço onde múltiplos conhecimentos do curso são praticados e, dentre eles, a gerência de configuração. Concluiu-se que a maioria dos respondentes não consegue configurar corretamente estes serviços. Além disso, por meio de mineração de cinco repositórios no BitBucket da turma de RP V 2018/1, observou-se várias inconsistências de configuração, o que demonstra que os alunos não sabiam exatamente o que estavam fazendo. Por fim, os resultados apontam que não é de fácil aprendizado a tarefa e configurar um servidor/serviço de integração contínua.

Com a leitura dos artigos que foram extraídos no mapeamento, foi apresentado que a utilização de integração contínua, entrega contínua e implantação contínua se torna muito útil nas empresas, pois com ela pode-se diminuir o tempo e frequência de entrega de mudanças nas aplicações com uma maior qualidade e eficiência e resultando também em uma melhor comunicação entre os times. Configurando pipelines para isso é possível também validar a aplicação com testes antes de subir para a versão em produção, apesar disso também ter sido mencionado em grande parte dos trabalhos acaba resultando em um custo grande para as empresas coma a exigência recursos humanos bem capacitados e problemas de aceitação da equipe na adoção das práticas contínuas. Grande parte das organização possui dificuldades na comunicação entre times, principalmente quando a

equipe de testes encontra um *bug* ao qual é reportado a equipe de desenvolvimento que resolve o problema e depois reenvia a equipe de testes gerando assim uma perca de tempo muito grande que pode ser minimizada pela adoção de testes automatizados na execução do pipeline.

Portanto, por meio dos dados coletados, é possível concluir que o desenvolvimento de um guideline e suporte ferramental para configuração de servidores de integração contínua seria interessante para o escopo da Acadêmia e, provavelmente, da Indústria. De forma a melhorar o aprendizado sobre Integração Contínua e auxiliar na configuração desses servidores, tanto em casos simples quanto em casos mais específicos, o próximo trabalho apresentará uma solução.

Para trabalhos futuros seria interessante uma avaliação em uma organização real visando os ganhos de tempo e consequentemente monetários da configuração de pipelines Integração Contínua(CI), Entrega Contínua(CD), Implantação Contínua(CE) a fim de determinar em quais casos essas configurações realmente trouxeram benefícios. Outro ponto importante seria identificar o ganho por entregar o produto de forma mais rápida e frequente com métricas bem definidas e precisas.

REFERÊNCIAS

ADAMS, B.; MCINTOSH, S. Modern release engineering in a nutshell – why researchers should care. In: **2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)**. [S.l.: s.n.], 2016. v. 5, p. 78–90. Citado 2 vezes nas páginas 41 e 51.

AGARWAL, A.; GUPTA, S.; CHOUDHURY, T. Continuous and integrated software development using devops. In: **2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)**. [S.l.: s.n.], 2018. p. 290–293. Citado 3 vezes nas páginas 47, 50 e 51.

AHMADIGHOHANDIZI, F.; SYSTÄ, K. Icdo: Integrated cloud-based development tool for devops. In: **SPLST**. [S.l.: s.n.], 2015. Citado 2 vezes nas páginas 42 e 51.

ANTUNES, R. V.; NAVARRO, G. M.; HANAZUMI, S. Test framework for jenkins shared libraries. In: **Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing**. New York, NY, USA: Association for Computing Machinery, 2018. (SAST '18), p. 13–19. ISBN 9781450365550. Disponível em: <https://doi.org/10.1145/3266003.3266008>. Citado 3 vezes nas páginas 44, 51 e 53.

ARACHCHI, S.; PERERA, I. Continuous integration and continuous delivery pipeline automation for agile software project management. In: **2018 Moratuwa Engineering Research Conference (MERCon)**. [S.l.: s.n.], 2018. p. 156–161. Citado 3 vezes nas páginas 46, 50 e 51.

BALBES, M. **How Agile Are You? Let's Actually Measure It!** 2015. <https://adtmag.com/articles/2015/12/15/balbes-agile-model-0-intro.aspx>. Acessado em: 31.08-2021. Citado na página 66.

BALBES, M. **How Agile Are You? Let's Actually Measure It! (Part 2: Quality Advocacy) (Page 1)**. 2015. <https://adtmag.com/Articles/2015/12/15/Balbes-Agile-Model-2-Quality.aspx?Page=1>. Acessado em: 31.08-2021. Citado na página 66.

BALBES, M. **How Agile Are You? Let's Actually Measure It! (Part 2: Quality Advocacy) (Page 2)**. 2015. <https://adtmag.com/Articles/2015/12/15/Balbes-Agile-Model-2-Quality.aspx?Page=2>. Acessado em: 31.08-2021. Citado na página 67.

BALBES, M. **How Agile Are You? Let's Actually Measure It!(Part 1: Technical Craftmanship)(Page 1)**. 2015. <https://adtmag.com/Articles/2015/12/15/Balbes-Agile-Model-1-Technical.aspx?Page=1>. Acessado em: 31.08-2021. Citado na página 66.

BALBES, M. **How Agile Are You? Let's Actually Measure It!(Part 1: Technical Craftmanship)(Page 2)**. 2015. <https://adtmag.com/Articles/2015/12/15/Balbes-Agile-Model-1-Technical.aspx?Page=2>. Acessado em: 31.08-2021. Citado na página 66.

BASS, L. The software architect and devops. **IEEE Software**, IEEE Computer Society, Los Alamitos, CA, USA, v. 35, n. 01, p. 8–10, jan 2018. ISSN 1937-4194. Citado 2 vezes nas páginas 56 e 57.

BIENER, A. S.; CRAWFORD, A. C. Devops for containerized applications. In: AHRAM, T. Z. (Ed.). **Advances in Artificial Intelligence, Software and Systems Engineering**. Cham: Springer International Publishing, 2019. p. 35–44. ISBN 978-3-319-94229-2. Citado 2 vezes nas páginas 42 e 51.

CHEN, L. Continuous delivery: Huge benefits, but challenges too. **IEEE Software**, IEEE, v. 32, n. 2, p. 50–54, 2015. Citado na página 24.

CHEN, L. Continuous delivery: Overcoming adoption challenges. **Journal of Systems and Software**, v. 128, p. 72–86, 2017. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121217300353>. Citado 6 vezes nas páginas 45, 50, 51, 53, 54 e 55.

CI, H. **The Hudson Book**. 2017. Disponível em: <https://www.eclipse.org/hudson/the-hudson-book/book-hudson.chunked/index.html>. Citado na página 74.

CIRCLECI. **CircleCI Docs**. 2019. Disponível em: <https://circleci.com/docs/2.0>. Citado na página 71.

CLEMENCIC, M.; COUTURIER, B. Implementing a domain specific language to configure and run LHCb continuous integration builds. **Journal of Physics: Conference Series**, IOP Publishing, v. 664, n. 6, p. 062007, dec 2015. Disponível em: <https://doi.org/10.1088%2F1742-6596%2F664%2F6%2F062007>. Citado na página 33.

CONECTING, G. **O que é continuous deployment?** 2017. Disponível em: <https://gaea.com.br/o-que-e-continuous-deployment/>. Citado na página 32.

DEBROY, V.; MILLER, S.; BRIMBLE, L. Building lean continuous integration and delivery pipelines by applying devops principles: A case study at varidesk. In: **Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering**. New York, NY, USA: Association for Computing Machinery, 2018. (ESEC/FSE 2018), p. 851–856. ISBN 9781450355735. Disponível em: <https://doi.org/10.1145/3236024.3275528>. Citado 4 vezes nas páginas 45, 50, 51 e 53.

EDDY, B. P. et al. A pilot study on introducing continuous integration and delivery into undergraduate software engineering courses. In: **2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE T)**. [S.l.: s.n.], 2017. p. 47–56. Citado 3 vezes nas páginas 48, 51 e 53.

FARLEY, J. H. e D. **Continuous Delevery**. [S.l.]: Addison-Wesley Professional, 2007. Citado na página 33.

FOWLER, M. **Continuous Integration**. 2006. Disponível em: <https://martinfowler.com/articles/continuousIntegration.html>. Citado 2 vezes nas páginas 32 e 56.

GMEINER, J.; RAMLER, R.; HASLINGER, J. Automated testing in the continuous delivery pipeline: A case study of an online company. In: **2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)**. [S.l.: s.n.], 2015. p. 1–6. Citado 3 vezes nas páginas 48, 51 e 53.

GREISING, L.; BARTEL, A.; HAGEL, G. Introducing a deployment pipeline for continuous delivery in a software architecture course. In: **Proceedings of the 3rd European Conference of Software Engineering Education**. New York, NY, USA: Association for Computing Machinery, 2018. (ECSEE'18), p. 102–107. ISBN 9781450363839. Disponível em: <https://doi.org/10.1145/3209087.3209091>. Citado 3 vezes nas páginas 41, 50 e 51.

GUşEILă, L. G.; BRATU, D.-V.; MORARU, S.-A. Continuous testing in the development of iot applications. In: **2019 International Conference on Sensing and Instrumentation in IoT Era (ISSI)**. [S.l.: s.n.], 2019. p. 1–6. Citado 2 vezes nas páginas 43 e 51.

HEINRICH, R. et al. Performance engineering for microservices: Research challenges and directions. In: . [S.l.: s.n.], 2017. Citado na página 55.

HUDSON. **Hudson-ci documentation**. 2019. Disponível em: <https://wiki.eclipse.org/Hudson-ci/documentation>. Citado na página 73.

HUMBLE, J.; FARLEY, D. **Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)**. [S.l.]: Pearson Education, 2010. Citado na página 24.

HUMBLE, J.; MOLESKY, J. Why enterprises must adopt devops to enable continuous delivery. v. 24, p. 6–12, 08 2011. Citado na página 56.

HäKLI, A.; TAIBI, D.; SYSTA, K. Towards cloud native continuous delivery: An industrial experience report. In: **2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)**. [S.l.: s.n.], 2018. p. 314–320. Citado 3 vezes nas páginas 47, 51 e 53.

ITKONEN, J. et al. Perceived benefits of adopting continuous delivery practices. In: **Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement**. New York, NY, USA: Association for Computing Machinery, 2016. (ESEM '16). ISBN 9781450344272. Disponível em: <https://doi.org/10.1145/2961111.2962627>. Citado 3 vezes nas páginas 47, 51 e 53.

IVANOV, V.; SMOLANDER, K. Implementation of a devops pipeline for serverless applications. In: KUHRMANN, M. et al. (Ed.). **Product-Focused Software Process Improvement**. Cham: Springer International Publishing, 2018. p. 48–64. ISBN 978-3-030-03673-7. Citado 3 vezes nas páginas 44, 50 e 51.

JENKINS. **Jenkins User Documentation**. 2019. Disponível em: <https://jenkins.io/doc/>. Citado na página 76.

KONAT, G. et al. Pie: A domain-specific language for interactive software development pipelines. **arXiv preprint arXiv:1803.10197**, 2018. Citado na página 34.

LAUKKANEN, E.; ITKONEN, J.; LASSENIUS, C. Problems, causes and solutions when adopting continuous delivery—a systematic literature review. **Information and Software Technology**, Elsevier, v. 82, p. 55–79, 2017. Citado na página 33.

LEPPÄNEN, M. et al. The highways and country roads to continuous deployment. **Ieee software**, IEEE, v. 32, n. 2, p. 64–72, 2015. Citado na página 24.

LIMA, M. **Testes automatizados no Jenkins: recursos, plugins e dicas para aumentar a produtividade**. 2017. Disponível em: <https://medium.com/cwi-software/testes-automatizados-no-jenkins-recursos-plugins-e-dicas-para-aumentar-a-produtividade-1685ffa1e9db>. Citado na página 76.

LUO, R.; YE, W.; ZHANG, S. Towards a deployment system for cloud applications. In: **SEKE**. [S.l.: s.n.], 2015. p. 122–127. Citado 3 vezes nas páginas 42, 50 e 51.

MYSARI, S.; BEJGAM, V. Continuous integration and continuous deployment pipeline automation using jenkins ansible. In: **2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)**. [S.l.: s.n.], 2020. p. 1–4. Citado 3 vezes nas páginas 43, 50 e 51.

MäKINEN, S. et al. Improving the delivery cycle: A multiple-case study of the toolchains in finnish software intensive enterprises. **Information and Software Technology**, v. 80, p. 175–194, 2016. ISSN 0950-5849. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584916301434>. Citado 4 vezes nas páginas 44, 50, 51 e 53.

NAKAGAWA, E. Y. et al. **Revisão Sistemática da Literatura em Engenharia de Software: Teoria e Prática**. [S.l.]: Elsevier Brasil, 2017. Citado na página 35.

NGUYEN, J.; DUPUIS, M. Closing the feedback loop between ux design, software development, security engineering, and operations. In: . [S.l.: s.n.], 2019. ISBN 978-1-4503-6921-3. Citado 2 vezes nas páginas 41 e 51.

O'CONNOR, R. V.; ELGER, P.; CLARKE, P. M. Continuous software engineering—a microservices architecture perspective. **Journal of Software: Evolution and Process**, Wiley Online Library, v. 29, n. 11, p. e1866, 2017. Citado na página 33.

PAULE, C.; DüLLMANN, T. F.; HOORN, A. V. Vulnerabilities in continuous delivery pipelines? a case study. In: **2019 IEEE International Conference on Software Architecture Companion (ICSA-C)**. [S.l.: s.n.], 2019. p. 102–108. Citado 3 vezes nas páginas 45, 50 e 51.

PEFFERS, K. et al. A design science research methodology for information systems research. **J. Manage. Inf. Syst.**, v. 24, n. 3, p. 45–77, dez. 2007. ISSN 0742-1222. Citado 2 vezes nas páginas 25 e 27.

PHILLIPS, A. et al. The it manager's guide to continuous delivery: Delivering business value in hours, not months. **XebiaLabs**, 2015. Citado na página 24.

REHMANN, K.-T. et al. Performance monitoring in sap hana's continuous integration process. **SIGMETRICS Perform. Eval. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 43, n. 4, p. 43–52, fev. 2016. ISSN 0163-5999. Disponível em: <https://doi.org/10.1145/2897356.2897362>. Citado 4 vezes nas páginas 46, 50, 51 e 53.

SHAHIN, M.; BABAR, M. A.; ZHU, L. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. **IEEE Access**, IEEE, v. 5, p. 3909–3943, 2017. Citado na página 34.

SOMMERVILLE, I. **Software Engineering (9th Edition)**. [S.l.]: Addison-Wesley, 2010. Citado 4 vezes nas páginas 23, 31, 32 e 70.

STEFFENS, A.; LICHTER, H.; DöRING, J. S. Designing a next-generation continuous software delivery system: Concepts and architecture. In: **2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE)**. [S.l.: s.n.], 2018. p. 1–7. Citado 4 vezes nas páginas 46, 51, 52 e 53.

SáNCHEZ-GALLEGOS, D. D. et al. From the edge to the cloud: A continuous delivery and preparation model for processing big iot data. **Simulation Modelling Practice and Theory**, v. 105, p. 102136, 2020. ISSN 1569-190X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1569190X20300757>. Citado 3 vezes nas páginas 43, 50 e 51.

ULLAH, F. et al. Security support in continuous deployment pipeline. In: . [S.l.: s.n.], 2017. Citado 2 vezes nas páginas 42 e 51.

WETTINGER, J.; ANDRIKOPOULOS, V.; LEYMANN, F. Enabling devops collaboration and continuous delivery using diverse application environments. In: SPRINGER. **OTM Confederated International Conferences"On the Move to Meaningful Internet Systems"**. [S.l.], 2015. p. 348–358. Citado 3 vezes nas páginas 41, 50 e 51.

WETTINGER, J.; BREITENBüCHER, U.; LEYMANN, F. Dyn tail - dynamically tailored deployment engines for cloud applications. In: **2015 IEEE 8th International Conference on Cloud Computing**. [S.l.: s.n.], 2015. p. 421–428. Citado 2 vezes nas páginas 45 e 51.

ZACHOW, M. Designing an android continuous delivery pipeline. In: **Software Engineering**. [S.l.: s.n.], 2016. Citado 2 vezes nas páginas 42 e 51.

ZHAO, Y. et al. The impact of continuous integration on other software development practices: a large-scale empirical study. In: IEEE. **2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)**. [S.l.], 2017. p. 60–71. Citado na página 34.

ZHAO, Y. et al. The impact of continuous integration on other software development practices: A large-scale empirical study. In: **2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)**. [S.l.: s.n.], 2017. p. 60–71. Citado 2 vezes nas páginas 56 e 57.

Apêndices

APÊNDICE A – CARACTERIZAÇÃO DO PROBLEMA

Para a caracterização do problema, executou-se dois estudos como segue.

A.1 Mineração de Repositórios no Bitbucket

Trata-se de um estudo de caso exploratório no estilo (*Mining Software Repository*). Ou seja, uma mineração daquilo que foi configurado pelos cinco times na representação de um pipeline de IC ao final da disciplina de Resolução de Problemas V de 2018/1. Para a análise, foi feita uma análise nos arquivos de configuração usados pelos grupos em seus respectivos projetos. Informalmente, também houveram questionamentos aos grupos sobre a configuração de seus servidores de IC.

Após a análise, foi identificado que a maioria dos grupos tiveram bastante dificuldade para a criação e configuração dos arquivos de configuração. Além disso, várias redundâncias também foram observadas. Quatro dos cinco grupos não conseguiram configurar corretamente o servidor de IC CircleCI, como requerido pelos professores. Apenas um grupo conseguiu configurar corretamente o servidor de IC, sendo que de seus quatro membros, apenas um deles desempenhou o papel de gerente de configurações. Todos os repositórios que foram analisados, ou não possuíam nenhum arquivo de configuração, ou usavam o padrão do servidor CircleCI sem customizações.

Também foi observado que haviam várias inconsistências nos arquivos de configuração. As mais comuns são a configuração de um banco de dados, que nunca era utilizado, caminhos errados para o diretório de trabalho, onde os testes deveriam ser executados, e erros simples de sintaxe. Vale ressaltar que a configuração do banco de dados não utilizado não impediu o Cicle CI de funcionar. Porém, o diretório incorreto e os pequenos erros sintáticos impediram. Outro ponto interessante é que a maioria dos arquivos de configuração tinham semelhanças com a de um grupo em específico, como se tivessem sido derivados do arquivo de configuração desse grupo específico e posteriormente alterados para os outros projetos. Mesmo assim, os arquivos de configuração analisados estavam incorretos e apresentavam redundâncias, o que demonstra que os gerentes de configuração não sabiam exatamente o que estavam realizando.

Após análise das configurações dos cinco projetos, alguns participantes foram questionados individualmente, de forma informal, sobre: 1) quais são suas dificuldades para configurar o Circle CI?, e 2) por quê não conseguiram implementar o servidor de Integração Contínua?

A maioria das respostas convergiram em: 1) porque que existe grande dificuldade para oferecer uma solução para o projeto na implantação de um pipeline de Integração Contínua em um curto espaço de tempo 2) a carga de atividades executadas em paralelo com o desenvolvimento do software, o que impediu que os gerentes pudessem focar na aprendizagem da Integração Contínua, e 3) questão de prioridade, pois o tempo era limitado e os gerentes consideravam mais importante entregar o projeto.

Esta entrevista informal levou ao desenvolvimento de um questionário formal. Ou seja, buscando-se caracterizar melhor as dificuldades observadas pelos professores e alunos de RP V 2018/1, o seguinte estudo foi executado.

A.2 Survey com Alunos de RP V

Com intuito de entender sobre como foi experiência dos alunos, derivada do desafio de uma atividade para a configuração de um pipeline de Integração Contínua dentro da disciplina de Resolução de Problemas V, foi elaborado e aplicado um questionário usando o Google Forms. Este formulário segue no Anexo I deste documento.

O principal objetivo deste estudo foi identificar o perfil dos alunos para traçar estratégias na definição do guideline proposto. Assim, buscou-se compreender se existiam lacunas da gerência de configuração observadas também nas disciplinas que antecederam RP V 2018/1. Uma vez que se detecte tais lacunas, espera-se usá-las como base para propor determinadas atividades em um guideline que integre os conhecimentos necessários, de modo gradual.

Uma vez que a configuração de um servidor de integração contínua é a última etapa de aprendizado na gerência de configuração, os seguintes objetivos específicos foram elaborados: 1) Identificar quais práticas foram introduzidas anteriormente para controle de versão; 2) Identificar quais práticas foram introduzidas anteriormente para testes de software; 3) Identificar quais práticas foram introduzidas anteriormente para gerenciamento de mudanças em software; 4) Identificar quais práticas foram introduzidas anteriormente para automação de deploy; 5) Identificar quais práticas foram introduzidas anteriormente para Integração Contínua; 6) Identificar as atividades para Integração Contínua executadas individualmente, e 7) Identificar a relevância da nossa proposta de um sistema para assistir tais configurações no contexto da disciplina de RP V.

O questionário foi dividido em quatro seções, sendo elas: 1) Cabeçalho; 2) Questionário Sobre Conceitos e Práticas Anteriores à RP V; 3) Integração Contínua, e 4) Questões gerais.

Devido a maioria dos respondentes terem cursado RP V no primeiro semestre de 2018, e outros terem pouco contato com Integração Contínua nos outros semestres, agrupou-se as respostas em três categorias: 1) Geral; 2) 2018 e 3) Outros, exceto 2018. Para o primeiro grupo, analisam-se todos os dados. Já para o segundo, apenas os estudantes que cursaram Resolução de Problemas V em 2018/1 e o terceiro grupo todos os correspondentes exceto os que cursaram Resolução de Problemas V em 2018/1.

O survey foi dividido em três seções. Na primeira, foram apresentados os objetivos e motivações do questionário, solicitando informações como nome, e-mail, o ano em que cursou a disciplina, domínio e dificuldade referente ao conteúdo programático do plano de aula. Para o restante, notou-se que um respondente preencheu o questionário duas vezes, com exatamente as mesmas respostas. Assim, devido à ambas terem sido exatamente

iguais, descartou-se uma das respostas.

Como ilustram as Figuras 33 e 34, a maioria (mais da metade) dos respondentes cursou a disciplina de RP V em 2018/1. Mesmo nesta disciplina, onde se cobrou a configuração de um servidor de IC, os alunos tiveram dificuldades na implementação do problema proposto pelos professores.

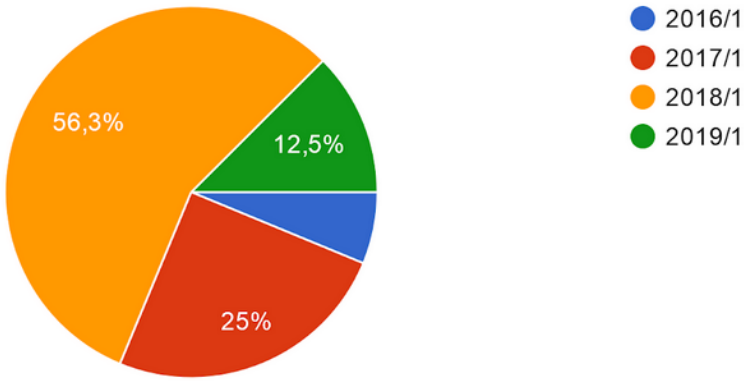

Figura 33 – Ano/Período em que os respondentes cursaram Resolução de Problemas V

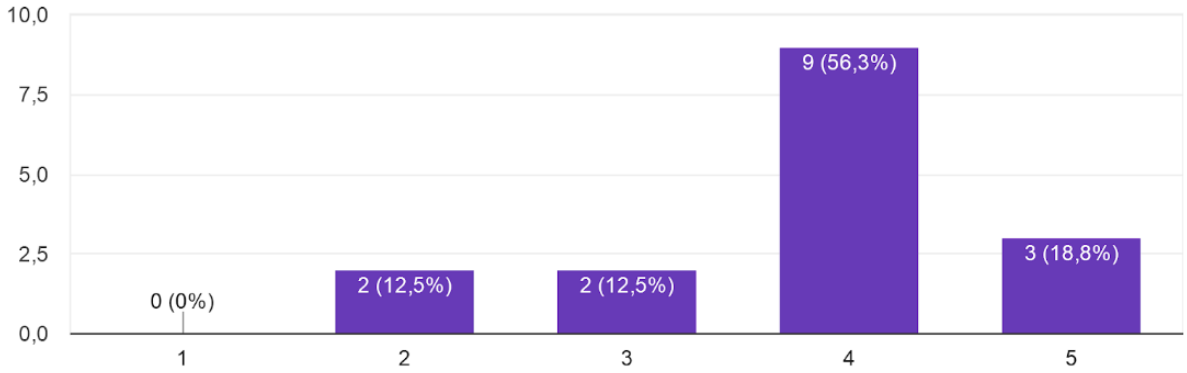

Figura 34 – Nível de dificuldade para a implementação do problema proposto em Resolução de Problema V

A.2.1 Segunda Seção de Perguntas

Na segunda seção foi questionado sobre os conceitos e práticas anteriores à disciplina. As questões eram sobre se as disciplinas anteriores a RP V forneciam a base necessária para: 1) gerência de configuração; 2) desenvolver e automatizar testes de testes de unidade, integração, regressão e aceitação; 3) gerenciar alterações em software decorrentes de uma nova Sprint; 4) gerenciar mudanças de requisitos de software; 5) realizar a auditoria e relatório das mudanças de requisitos.

Nas perguntas sobre gerência de configuração, como pode ser visto pela Figura 35, a maior parte dos respondentes do primeiro grupo (Geral) relatou que as disciplinas

anteriores a RP V forneceram as bases necessárias para utilizar uma ferramenta para versionamento de código. Porém, a maior parte discordou totalmente referente as disciplinas anteriores a RP V fornecerem a base para utilizar uma ferramenta para automação do "deploy", como pode ser visto na Figura 36. Portanto, há um forte indício de que o guideline deva reforçar a automação da implantação, ou que este conteúdo seja incluído em alguma das disciplinas da Engenharia de Software.

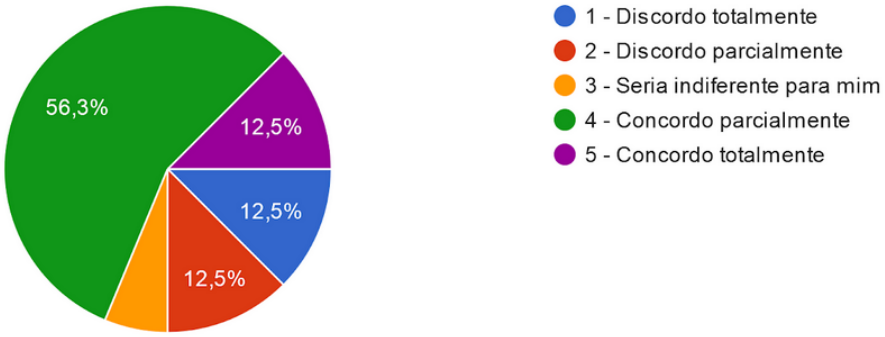

Figura 35 – Levando em conta as disciplinas anteriores à RP V, elas me forneceram a base necessária de "gerência de configuração"para utilizar uma ferramenta para versionamento de código, como o GIT, ou SVN, ou CVS ou outra alternativa.

Nas questões sobre desenvolvimento e automatização de testes a maioria concordou que as disciplinas anteriores a RP V forneciam a base para desenvolver e automatizar testes de unidade e integração, como podem ser vistos nas Figuras 37 e 38. Porém, discordaram referente aos testes de regressão, como pode ser visto na Figura 39.

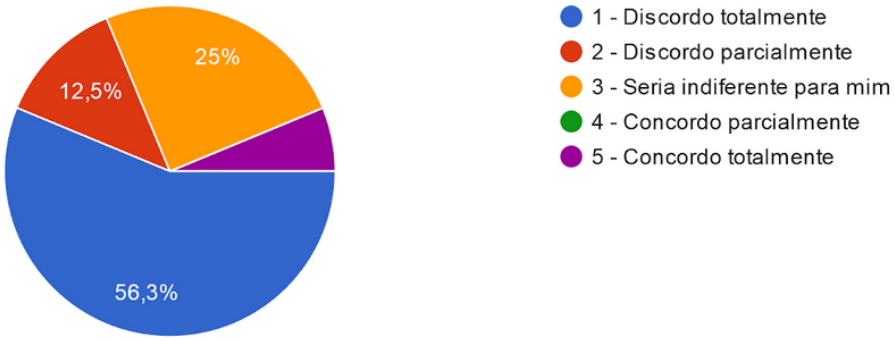

Figura 36 – Levando em conta as disciplinas anteriores à RP V, elas me forneceram a base necessária de "gerência de configuração"para utilizar uma ferramenta para automação do "deploy"como ANT, MAVEN ou outra alternativa.

Sobre os testes de aceitação, a opinião ficou dividida 40. Ou seja, há a necessidade de guideline auxiliar na inclusão dos testes Selenium nos pipelines de IC. Quanto aos testes de aceitação, há a necessidade de reforço em disciplinas que apresentem checklists

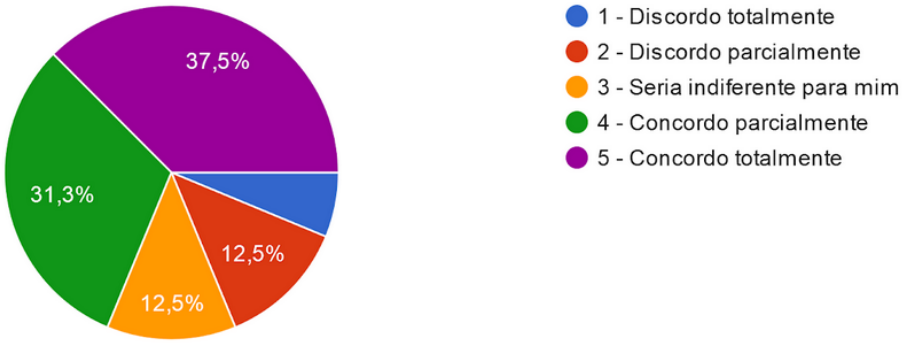

Figura 37 – Levando em conta as disciplinas anteriores à RP V, elas me forneceram a base necessária para desenvolver e automatizar testes de unidade.

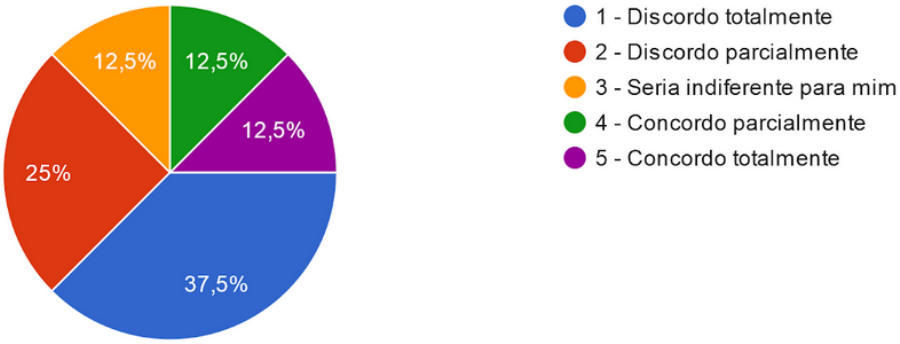

Figura 38 – Levando em conta as disciplinas anteriores à RP V, elas me forneceram a base necessária para desenvolver e automatizar testes de integração.

pra execução de testes de aceitação. Quanto ao guideline, uma vez que um teste de aceitação é manual, é preciso prever o uso de uma atividade que utilize trigger no pipeline de IC.

A Figura 41 mostra as respostas para a pergunta referentes à gerenciamento de alterações em software dentro da Sprint. A maioria dos respondentes concorda que essa fundamentação foi transmitida em disciplinas anteriores ou na disciplina de RP V corrente.

Sobre mudanças de requisitos de software, a maioria concordou que as disciplinas anteriores forneceu a base necessária, como pode ser visto na Figura 42. Porém, sobre a base necessária para realizar a auditoria e o relatório das mudanças de requisitos, a maioria discordou que fundamentos tenham sido transmitidos, como ilustra a Figura 43.

A.2.2 Terceira Seção de Perguntas

Na terceira seção, foi questionado sobre os conceitos e práticas em Integração Contínua. A primeira questão já visava identificar se o respondente havia se envolvido na aprendizagem da Integração Contínua ao final das últimas Sprints. A segunda questão buscou entender sobre o número de horas dedicadas para o aprendizado do conceito de In-

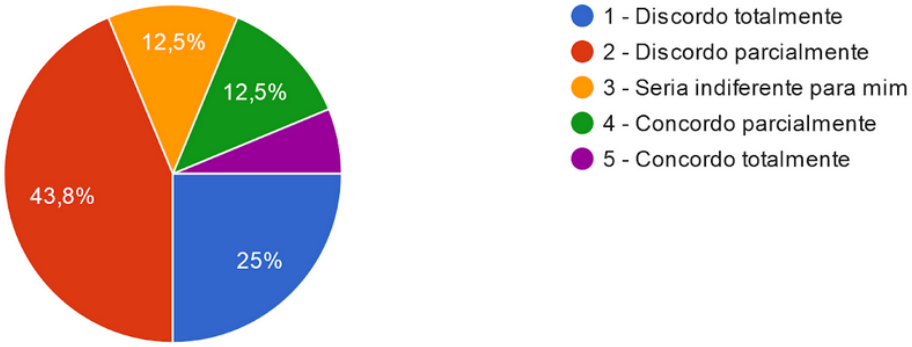

Figura 39 – Levando em conta as disciplinas anteriores à RP V, elas me forneceram a base necessária para desenvolver e automatizar testes de regressão.

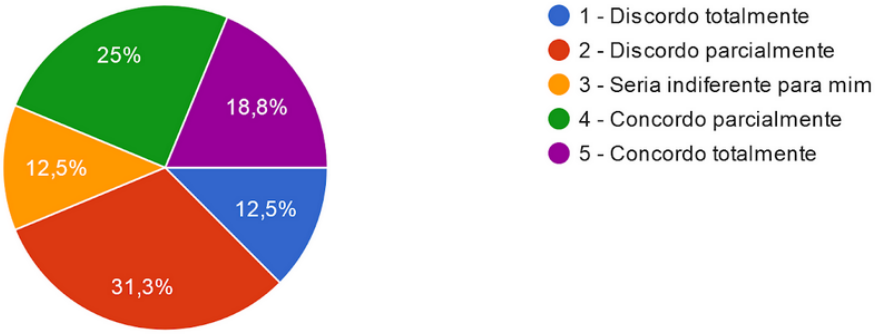

Figura 40 – Levando em conta as disciplinas anteriores à RP V, elas me forneceram a base necessária para desenvolver e automatizar testes de aceitação.

tegração Contínua. A terceira questão entrevistou como respondente havia feito/ajudado a configurar o servidor de Integração Contínua.

As próximas questões eram sobre qual servidor e quantas horas haviam dedicado para configuração. As questões posteriores eram sobre a dificuldade de configurar o servidor/serviço, e sobre se o respondente se considera apto a utilizar/configurar servidores de Integração Contínua no mercado de trabalho.

Por fim, as últimas questões perguntavam aos respondentes se eles consideram útil: 1) uma ferramenta para modelagem e configuração de pipelines de integração contínua, assistida por meio de geração de código; 2) uma ferramenta para modelagem e configuração de pipelines de integração contínua, assistida por meio de geração de código; 3) uma ferramenta desse tipo para questões educacionais na gerência de configuração; 4) uma ferramenta desse tipo para fábricas de software/indústria na gerência de configuração.

A.2.2.1 Análise do Primeiro Grupo de Respostas

O primeiro grupo analisado foi o grupo Geral(todos os respondentes). Referente à primeira questão, a maioria concordou parcialmente ou totalmente, como pode ser visto na

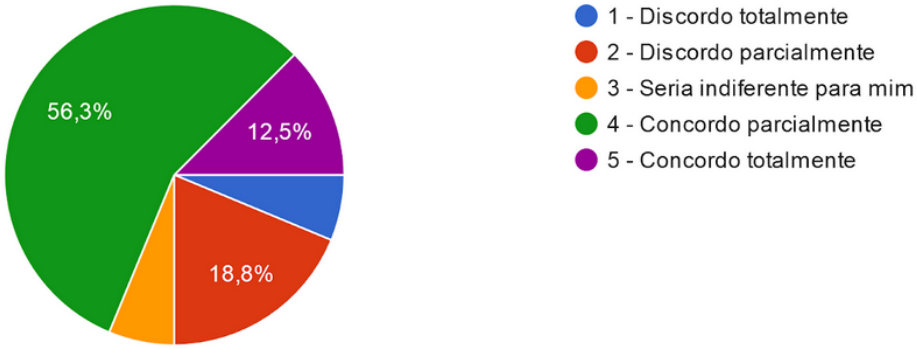

Figura 41 – Levando em conta as disciplinas anteriores à RP V, elas me forneceram a base necessária para gerenciar alterações em software decorrentes de uma nova Sprint.

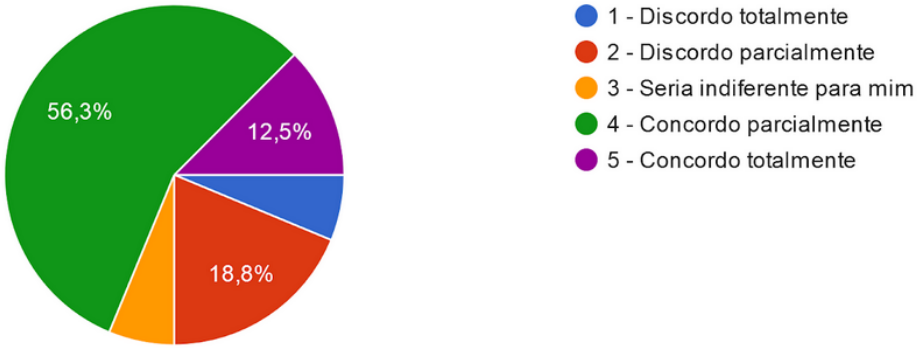

Figura 42 – Levando em conta as disciplinas anteriores à RP V, elas me forneceram a base necessária para gerenciar mudanças de requisitos de software.

Figura 44. Este gráfico mostra que maioria dos entrevistados pesquisou sobre Integração Contínua.

No entanto, na segunda questão mostrada na Figura 45, a maioria dedicou menos de 10 horas para a aprendizagem dos conceitos de IC.

Na terceira questão mostrada na Figura 44, mais de 60% ao menos ajudou outros grupos a configurar um servidor de IC.

A questão quatro e cinco tratam de qual servidor e quantas horas os entrevistados haviam dedicado para realizar a configuração. Das respostas mostradas na Figura 47, 50% respondeu que o servidor/serviço usado foi o CircleCI e 0% havia usado Jenkins ou Hudson CI. Além disso, 37,5% não usou servidor algum e 37,5% dedicou menos de 10 horas na configuração, como pode ser visto na Figura 48.

A sexta questão buscou respostas sobre a dificuldade de configurar um servidor de IC, como mostra a Figura 49. Aqui, 56,3% dos entrevistados consideraram difícil ou muito difícil a tarefa, e apenas 37,5% respondeu "Não tentamos/Não se aplica". Isto é, portanto, um forte indício de acerto na escolha da análise por grupos separados. Como o grupo 2018 possui apenas membros de 2018/1 que tiveram mais contato com IC, esperava-se

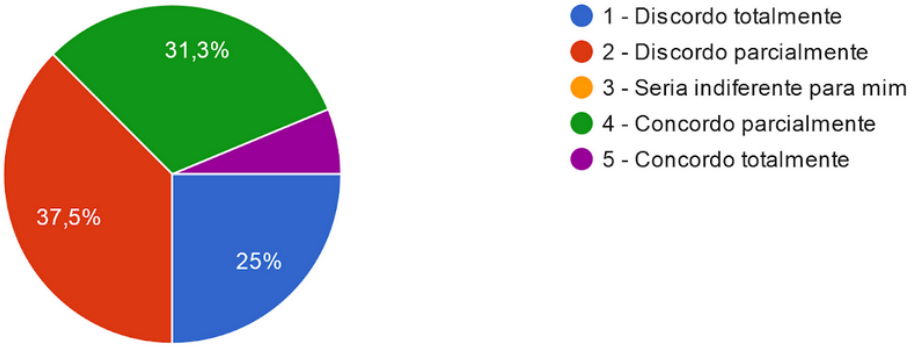

Figura 43 – Levando em conta as disciplinas anteriores à RP V, elas me forneceram a base necessária para realizar a auditoria e relatório das mudanças de requisitos.

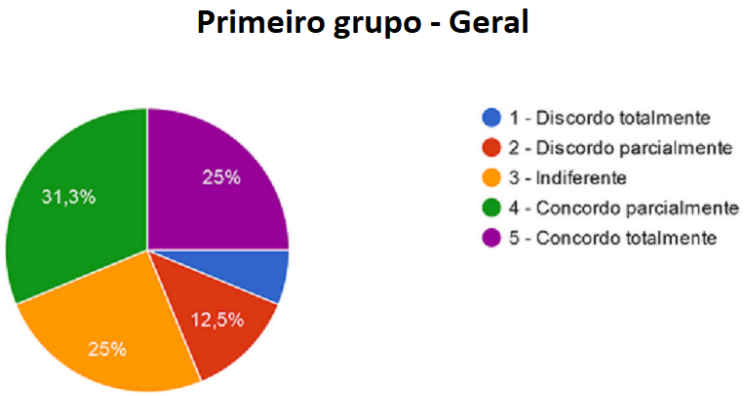

Figura 44 – Eu me envolvi no aprendizado da integração contínua ao final das últimas Sprints.

que este teria uma quantidade menor de "Não tentamos/Não se aplica".

Na sétima e oitava questões, como mostrado respectivamente na Figura 50, observou-se que apenas 31,3% dos respondentes concordaram parcialmente ou totalmente em estarem aptos a utilizar/configurar um servidor de IC. A maioria, 50% dos entrevistados, discordaram totalmente ou parcialmente sobre estarem aptos a utilizar/configurar um servidor de IC.

Referente às quatro últimas questões, cujas respostas podem ser observadas na Figura 51, a maioria concordou parcialmente ou totalmente sobre a utilidade das ferramentas.

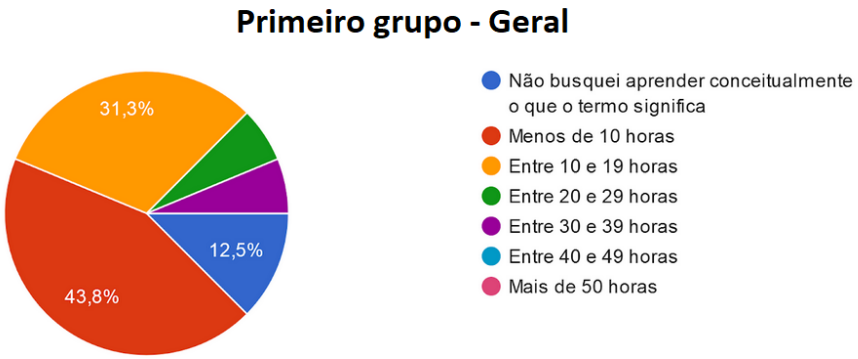

Figura 45 – Informe o número de horas dedicadas para o aprendizado sobre o conceito de integração contínua

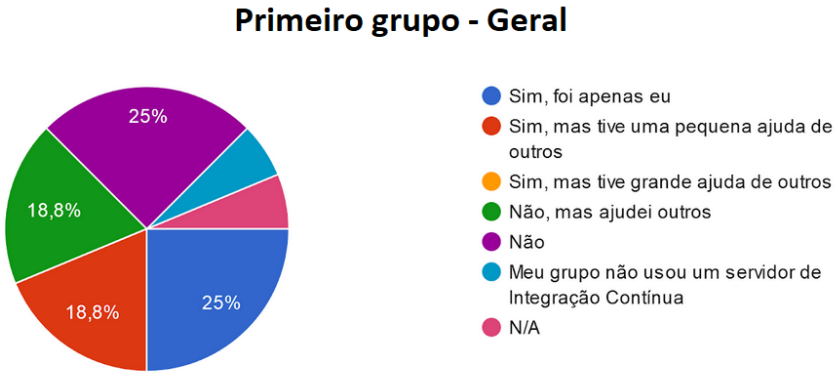

Figura 46 – Durante Resolução de Problemas V em 2018 foi você quem configurou o servidor/serviço de Integração Contínua?

A.2.2.2 Análise do Segundo Grupo de Respostas

O segundo grupo analisado foi o grupo 2018, que inclui todos os respondentes que fizeram RP V em 2018/1. Para a geração de gráficos desse grupo, foi usado o Google Planilhas (Spreadsheets).

Referente à primeira e segunda questões, cujas respostas são mostradas na Figura 52, observou-se que 66,6% dos respondentes concordaram parcialmente/totalmente sobre terem participado do aprendizado de IC. Estas respostas acabam sendo maiores que os 56,3% do grupo geral, o que demonstra pelo menos o envolvimento de mais pessoas além do líder técnico do grupo, já que eram cinco grupos.

Além disso, houve aumento na quantidade de horas dedicas, como pode ver na Figura 53. Isto demonstra que os alunos tem noção da relevância que a integração contínua tem para a indústria, e de que à consideram como importantes para as demandas de

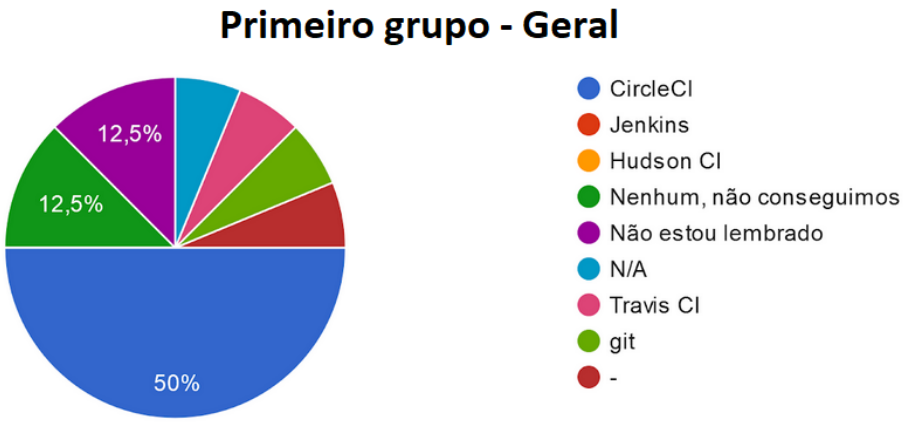

Figura 47 – Em caso afirmativo, qual servidor/serviço de integração contínua você ou seu grupo usou?

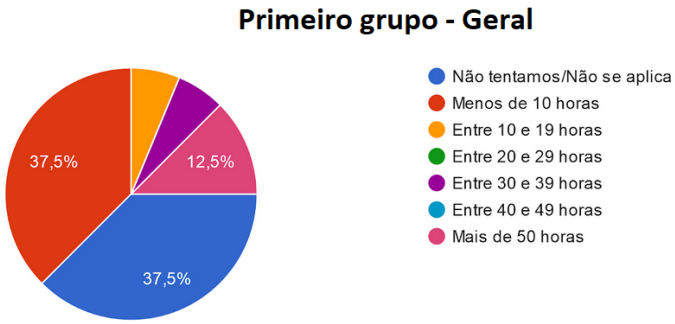

Figura 48 – Informe o número de horas dedicadas para configurar o serviço/servidor de integração contínua.

oportunidades de empregos mais recentes.

As respostas para a terceira questão podem ser vistas na Figura 54. Apenas 11,1% dos respondentes não usou um servidor de IC, enquanto que todos os outros ou ajudaram alguém ou configuraram o seu próprio servidor de IC.

Nas questões quatro e cinco, observa-se que 88,9% usou CircleCI, como pode ser visto na Figura 55, enquanto que 22,2% gastou mais de 50 horas nesta tarefa, como pode ser visto na Figura 56. Ou seja, mesmo utilizando um servidor tido como de fácil utilização em blogs e listas de discussões, a atividade de configuração do CircleCI ainda é considerada muito difícil pelos respondentes.

Essa observação é complementada com as respostas para a sexta questão. Nela observa-se que 88,9% dos respondentes achou difícil ou muito difícil configurar um servidor de IC como pode ser visto na Figura 57. Neste sentido, até este ponto, cabe destacar que

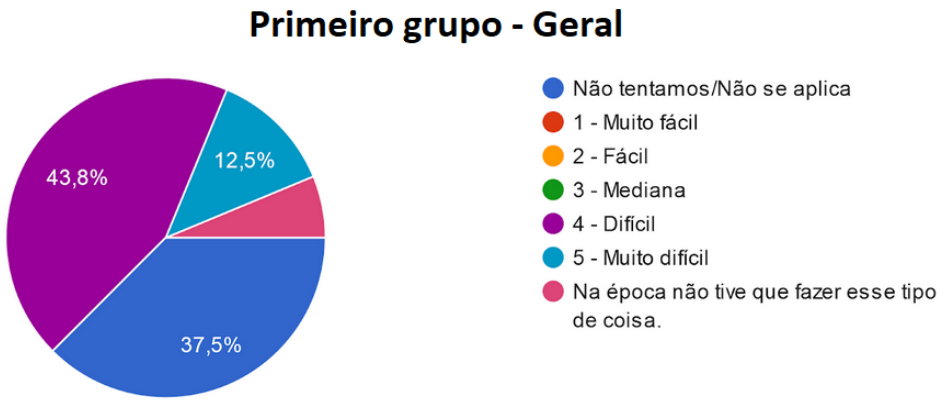

Figura 49 – Levando-se em consideração as questões anteriores, pode-se afirmar que para você configurar o servidor/serviço de integração contínua foi

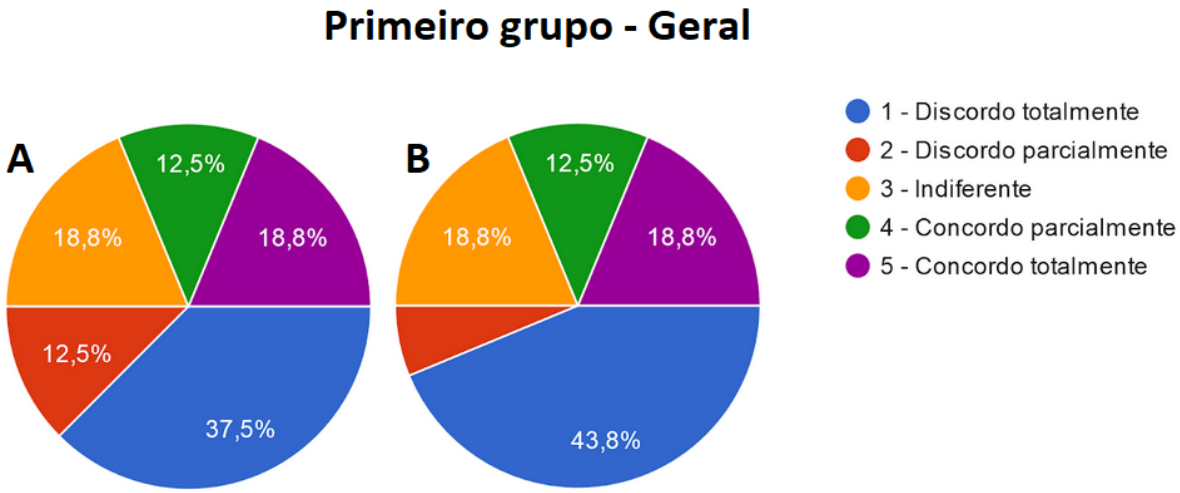

Figura 50 – Eu me considero apto para UTILIZAR um servidor de integração contínua no mercado de trabalho.

as dificuldades podem ser tanto metodológicas como ferramentais.

As respostas para a sétima e oitava questões são mostradas nas Figuras 58 e 59. Apenas 33,3% dos respondentes se considera apto para usar e configurar um servidor/serviço de IC no mercado de trabalho. Isso demonstra que devemos dar mais atenção para esta prática da gerência de configurações em disciplinas de RP V.

Referente às quatro últimas questões, como mostrado em Figura 60, em média 83,35% dos respondentes concordaram completamente/parcialmente com a utilidade de ferramentas que facilitem a configuração de integração contínua.

A.2.2.3 Análise do Terceiro Grupo de Respostas

O terceiro e último grupo analisado foi o E2018. Levando em consideração a diferença do desempenho do segundo grupo (2018) com o primeiro grupo (Geral), espera-

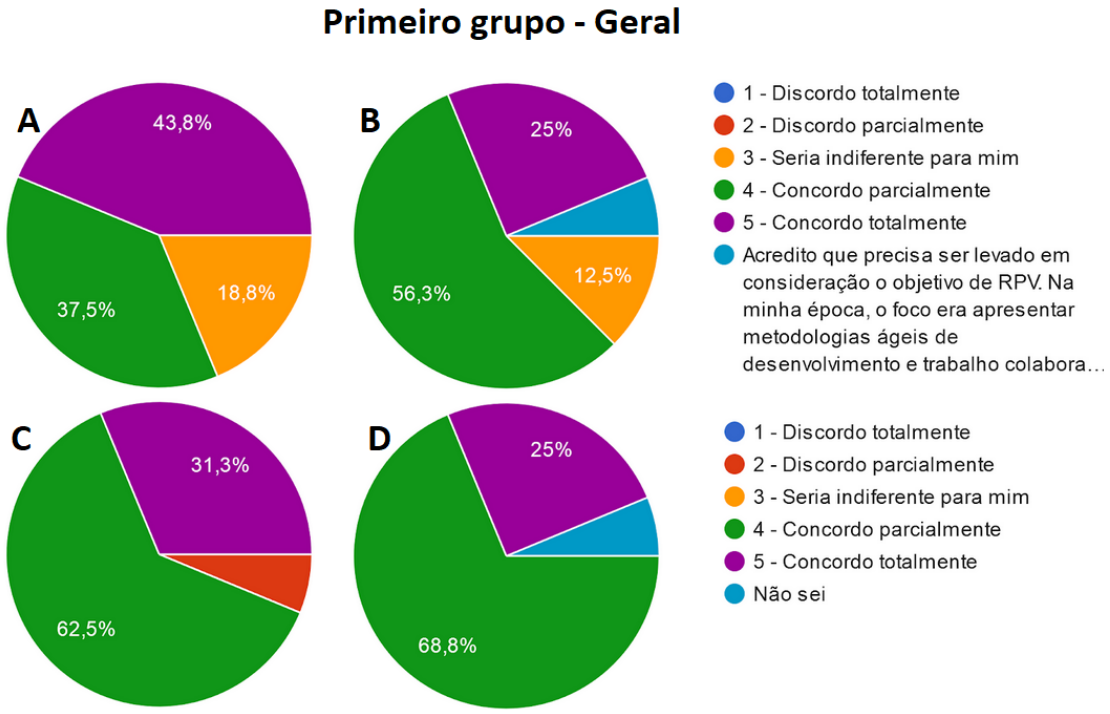

Figura 51 – Respostas referente a respectivamente se o respondente considera útil uma ferramenta para para modelagem e configuração de pipelines de integração contínua, assistida por meio de geração de código levando em consideração Objetivo profissional/Objetivos do seu grupo nas atividades da disciplina de RP V/Questões educacionais na gerência de configuração/Fábricas de software na gerência de configuração

se que o conhecimento dos alunos referente a Integração Contínua seja pior. Portanto, nesse caso, uma visão geral referente as respostas será apresentada no final ao invés de comentar cada questão, ainda por cima é possível ter uma visão com as Figuras 61, 62, 63, 64, 65, 66, 67, 69 referente aos resultados obtidos.

Como já era de se esperar, nas questões de IC o terceiro grupo teve um desempenho pior, provavelmente pelo fato de não terem sido tão incentivados a usar/aprender sobre IC como os respondentes que cursaram RP V em 2018/1. Porém, paradoxalmente, mesmo o grupo de 2018 teve apenas um arquivo de configuração funcional, como explicado na seção A.1.

A.2.3 Quarta Seção de Perguntas

A quarta e última seção é sobre questões gerais onde os respondentes foram questionados sobre: 1) sua disposição a testar uma ferramenta de modelagem e geração de código para servidores de integração contínua, participando assim de um experimento controlado em meados de novembro de 2019, mas executado fora de uma disciplina de Engenharia de Software; 2) se quanto às práticas fundamentais da "gerência de configu-

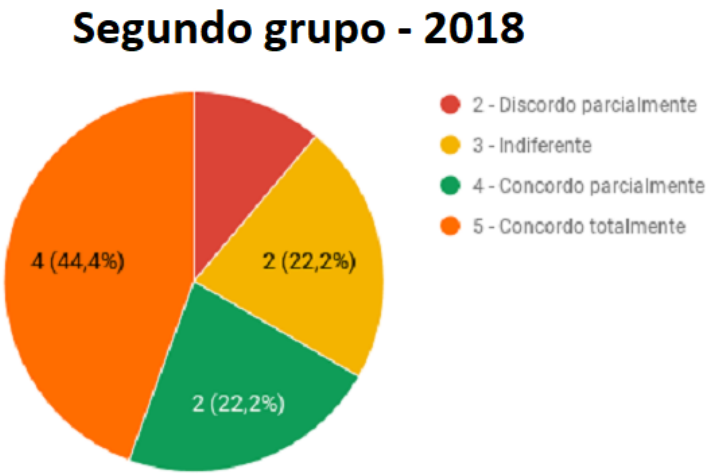

Figura 52 – Eu me envolvi no aprendizado da integração contínua ao final das últimas Sprints

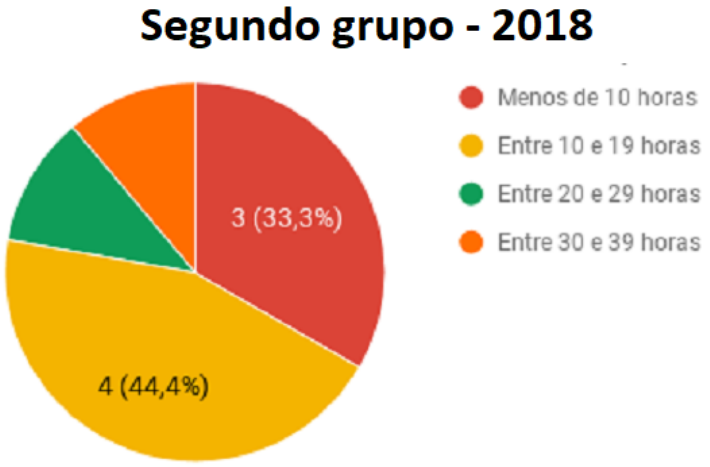

Figura 53 – Informe o número de horas dedicadas para o aprendizado sobre o conceito de integração contínua

ração"apresentadas anteriormente, o respondente se considera apto a executá-las hoje no mercado de trabalho em empresas como DELL, HP, Amazon, ThoughtWorks, etc e 3) sugestões que o respondente traz para as próximas versões da disciplina de RP V ou RP VI, focadas no exercício de práticas de gerência de configuração. Sobre a primeira questão a maioria concordou em participar ou estava em dúvida e solicitou um e-mail na data do experimento, referente a segunda a maioria discordou e sobre a terceira as opiniões foram bem variadas.

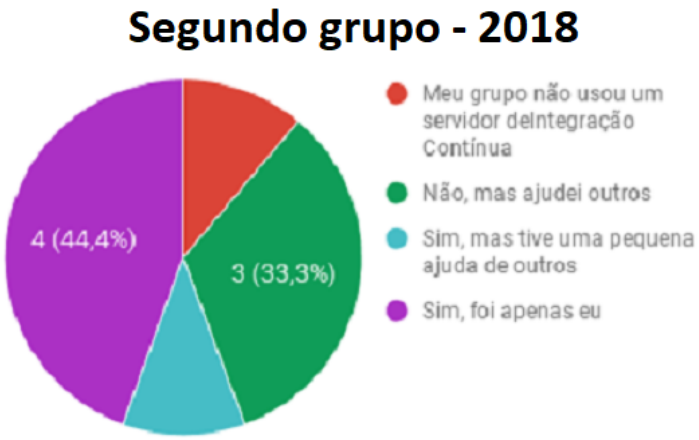

Figura 54 – Durante Resolução de Problemas V em 2018 foi você quem configurou o servidor/serviço de Integração Contínua?

Figura 55 – Em caso afirmativo, qual servidor/serviço de integração contínua você ou seu grupo usou?

A.3 Resultados Preliminares

Como visto nas seções A.1 e A.2 a Integração Contínua não é algo trivial. Dos repositórios analisados, em apenas um arquivo uma pipeline muito simples foi configurada corretamente. Além disso, os outros projetos possuíam várias inconsistências.

Quanto ao survey apresentado na seção A.2, a divisão em grupos resultou em uma análise melhor, tornando possível notar que em 2018 todos usaram CircleCI, mesmo quando o professor ofereceu duas outras opções. Destaca-se neste survey o seguinte achado: apenas 31,3% dos respondentes concordaram parcialmente ou totalmente nas questões que investigam se eles conseguiriam utilizar/configurar um servidor/serviço de IC no mercado de trabalho. Portanto, trata-se de um número relativamente baixo, que justifica o seguimento da nossa proposta.

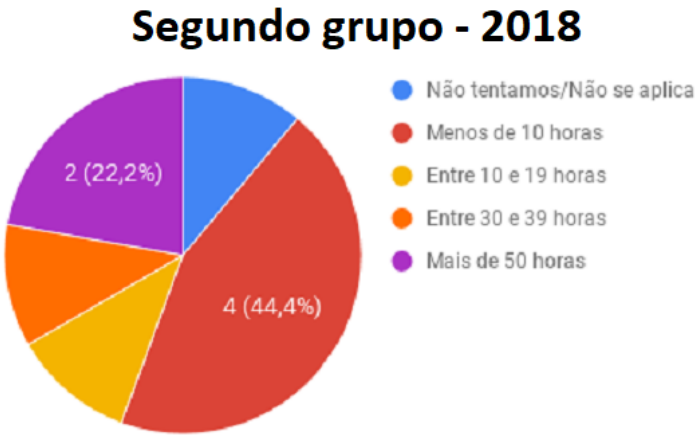

Figura 56 – Informe o número de horas dedicadas para configurar o serviço/servidor de integração contínua

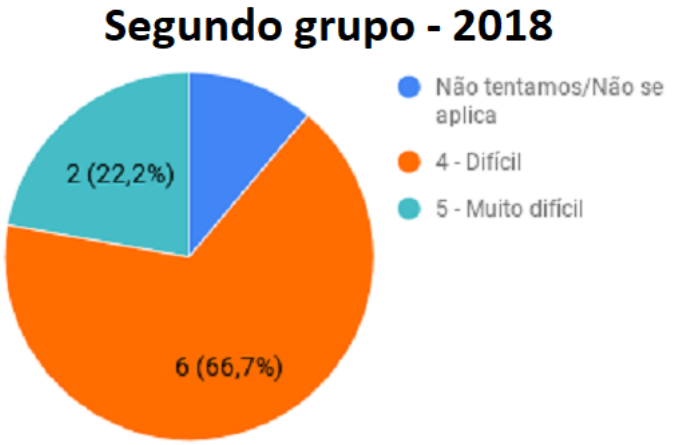

Figura 57 – Levando-se em consideração as questões anteriores, pode-se afirmar que para você configurar o servidor/serviço de integração contínua foi

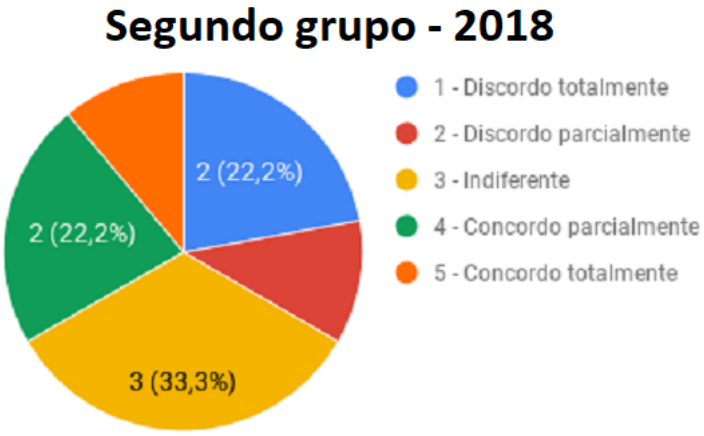

Figura 58 – Eu me considero apto para UTILIZAR um servidor de integração contínua no mercado de trabalho

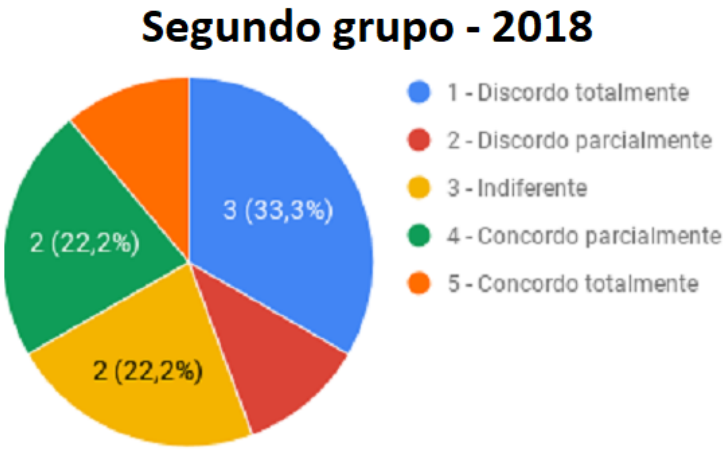

Figura 59 – Eu me considero apto para CONFIGURAR servidores de integração contínua no mercado de trabalho

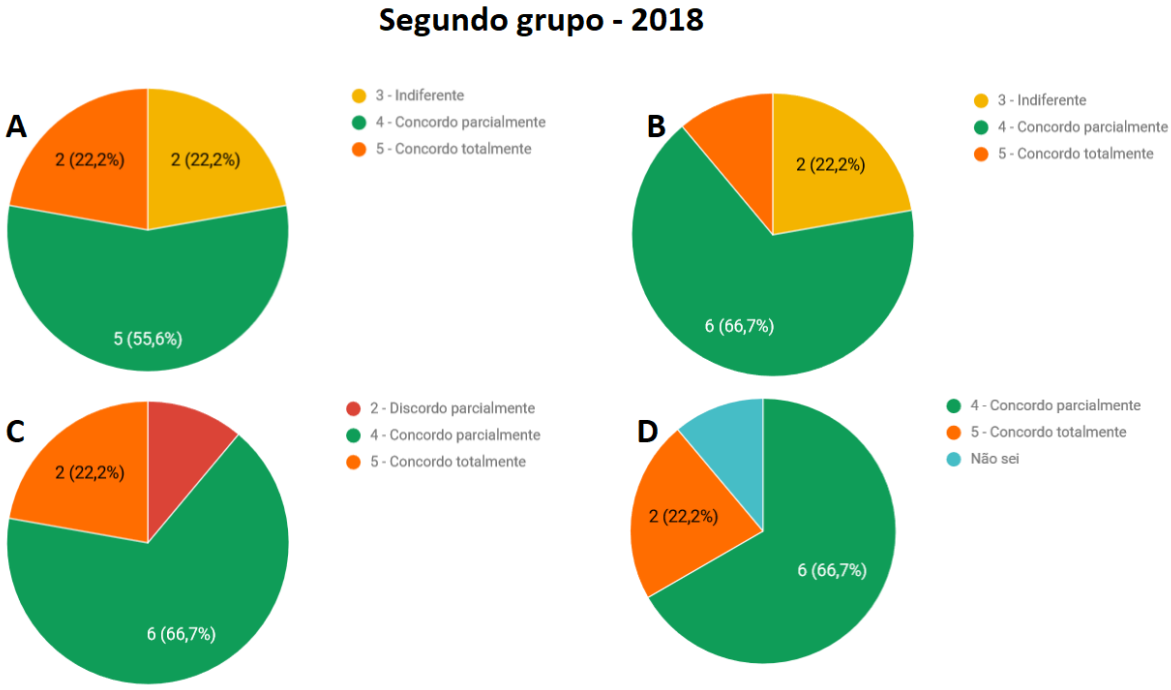

Figura 60 – Respostas referente as questões 09, 10, 11 e 12 da seção de IC do grupo de 2018

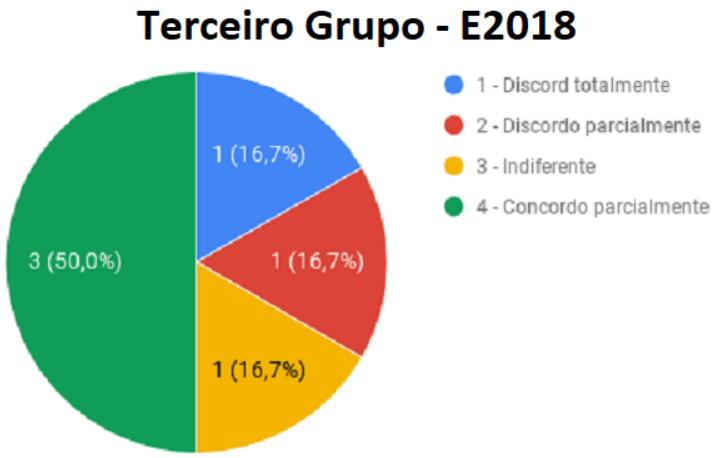

Figura 61 – Eu me envolvi no aprendizado da integração contínua ao final das últimas Sprints

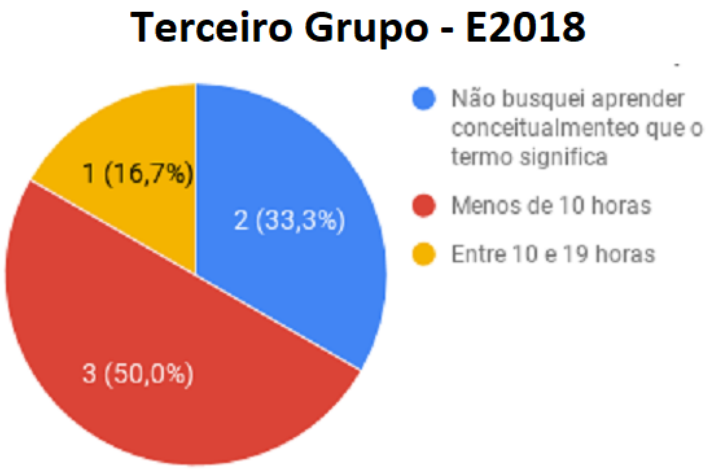

Figura 62 – Informe o número de horas dedicadas para o aprendizado sobre o conceito de integração contínua

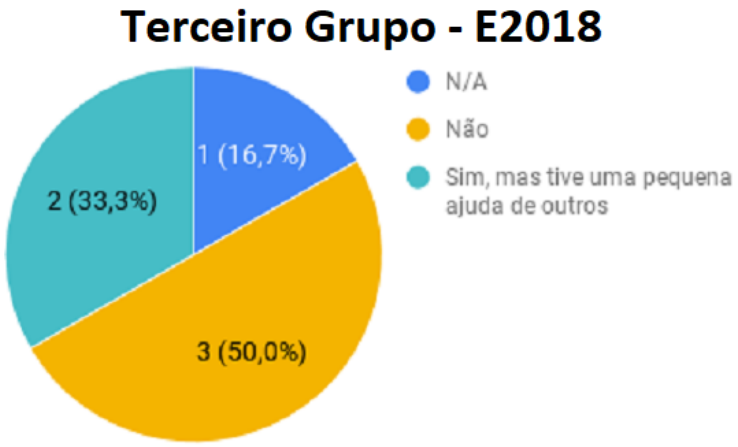

Figura 63 – Durante Resolução de Problemas V em 2018 foi você quem configurou o servidor/serviço de Integração Contínua?

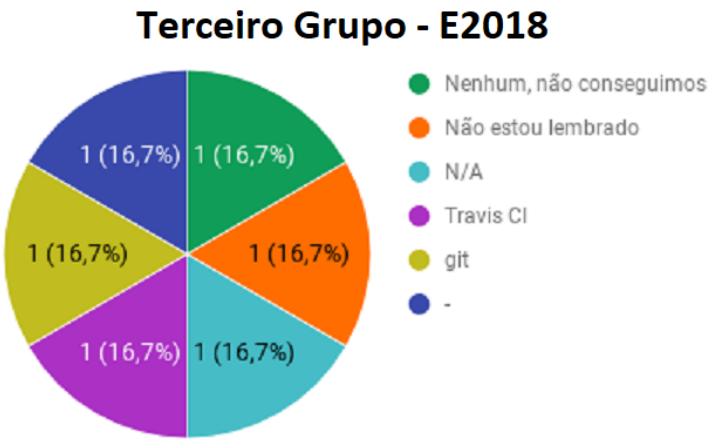

Figura 64 – Em caso afirmativo, qual servidor/serviço de integração contínua você ou seu grupo usou?

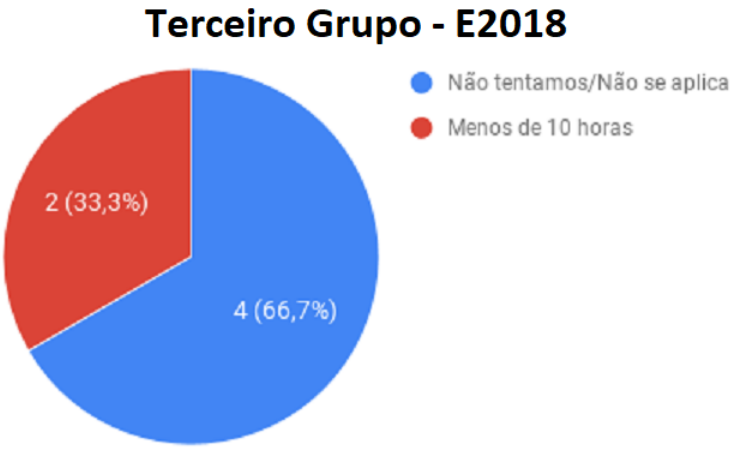

Figura 65 – Informe o número de horas dedicadas para configurar o serviço/servidor de integração contínua

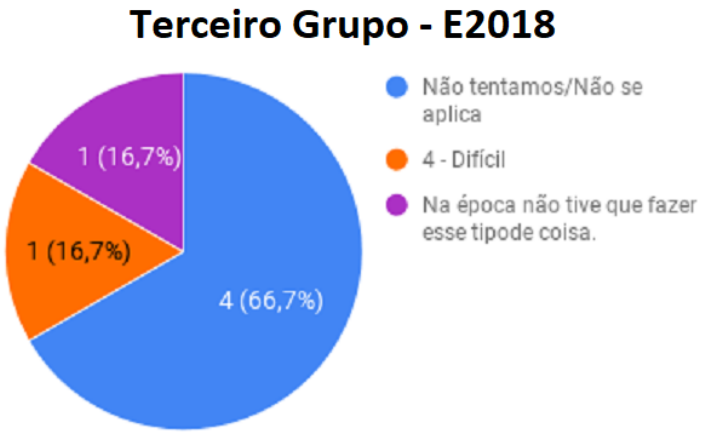

Figura 66 – Levando-se em consideração as questões anteriores, pode-se afirmar que para você configurar o servidor/serviço de integração contínua foi

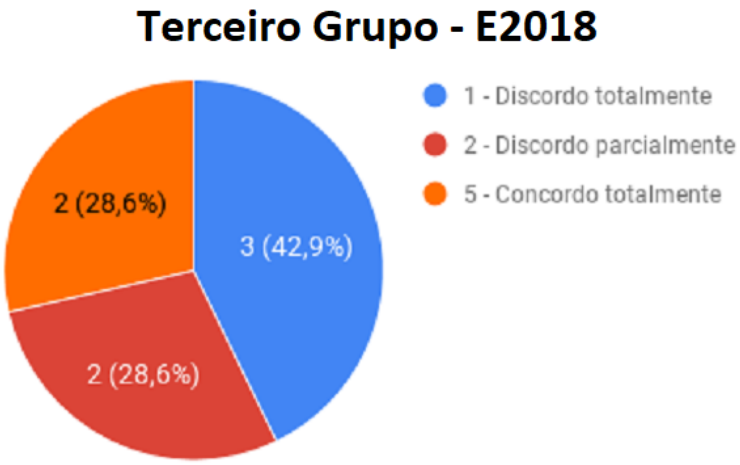

Figura 67 – Eu me considero apto para UTILIZAR um servidor de integração contínua no mercado de trabalho

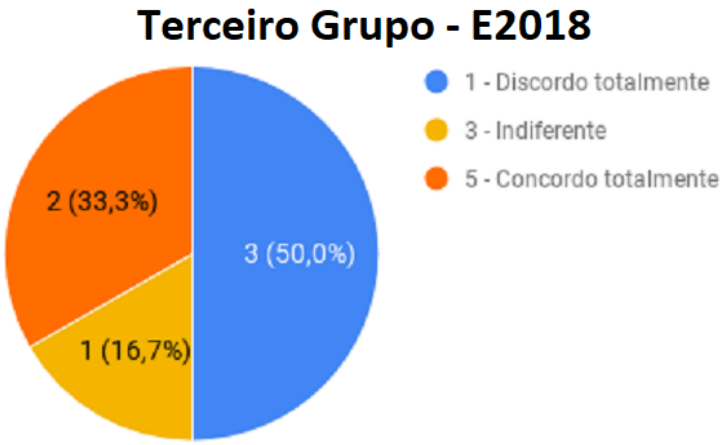

Figura 68 – Eu me considero apto para CONFIGURAR servidores de integração contínua no mercado de trabalho

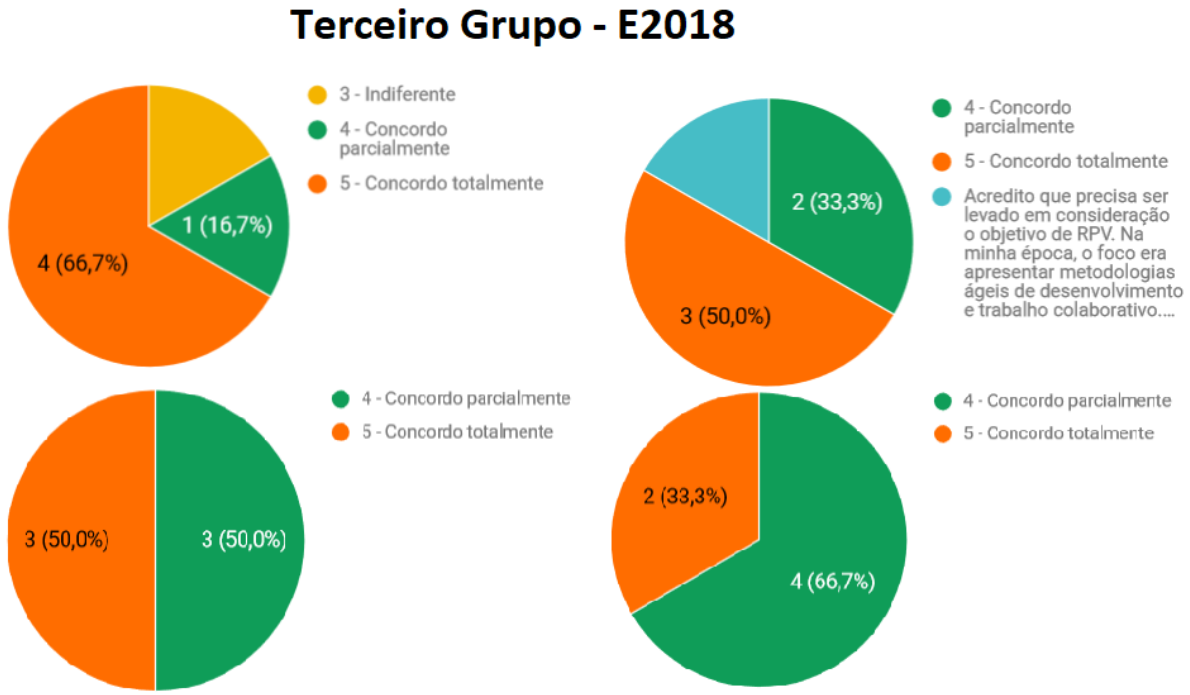

Figura 69 – Respostas referente as questões 09, 10, 11 e 12 da seção de IC do grupo E2018