



Universidade Federal do Ceará
Campus Quixadá
Curso de Engenharia de Software

Guyllherme Tabosa Cabral

**IMPLANTAÇÃO DE UMA FERRAMENTA DE INTEGRAÇÃO CONTÍNUA EM
UM NÚCLEO DE PRÁTICAS EM INFORMÁTICA: RELATO DE EXPERIÊNCIA**

Quixadá, Ceará

2014

Lista de ilustrações

Figura 1 – Branch no Sistema de Controle de Versão	9
Figura 2 – Sistema de Controle de Versão Local	9
Figura 3 – Sistema de Controle de Versão Centralizado	10
Figura 4 – Sistema de Controle de Versão Distribuído	11
Figura 5 – Processo Lógico de uma Build	12
Figura 6 – Ambiente de Integração Contínua	14
Figura 7 – Processo de Gerenciamento de Configuração	17

Lista de tabelas

Tabela 1 – Cronograma das atividades previstas	19
--	----

Lista de abreviaturas e siglas

NPI	Núcleo de Práticas em Informática
CMMI	Capability Maturity Model Integration
MPS.BR	Melhoria de Processo Brasileiro
GC	Gerência de Configuração
SCV	Sistema de Controle de Versão
SCM	Sistema de Controle de Mudança
IC	Integração Contínua
UFC	Universidade Federal do Ceará
TI	Tecnologia da Informação
PGC	Plano de Gerenciamento de Configuração
SGBD	Sistemas de Gerência de Banco de Dados
CCB	Change Control Board
SBC	Sociedade Brasileira de Computação
MI	Maintainability Index

Sumário

1	INTRODUÇÃO	5
2	TRABALHOS RELACIONADOS	6
3	OBJETIVOS	7
3.1	Objetivo Geral	7
3.2	Objetivos Específicos	7
4	FUNDAMENTAÇÃO TEÓRICA	8
4.1	Gerência de Configuração	8
4.1.1	Plano de Gerenciamento de Configuração	8
4.1.2	Sistema de Controle de Versão	8
4.1.2.1	Sistema de Controle de Versão Local	9
4.1.2.2	Sistema de Controle de Versão Centralizado	10
4.1.2.3	Sistema de Controle de Versão Distribuído	10
4.1.3	Sistema de Controle de Mudança	10
4.1.4	Auditoria de Configuração	11
4.1.5	Ferramentas de Build	12
4.2	Integração Contínua	12
4.2.1	Características de Integração	13
4.2.2	Processo de Integração	13
4.2.3	Benefícios da Integração Contínua	14
4.2.4	Builds Automatizadas	14
4.2.5	Integração Contínua Manual	15
4.2.6	Integração Contínua Automatizada	15
4.2.7	Processo de Escolha da Ferramenta	15
4.3	Processo do NPI	16
4.3.1	Processo de Gerência de Configuração do NPI	16
5	PROCEDIMENTOS METODOLÓGICOS	18
5.1	Cronograma de Execução	19
	Referências	20

1 Introdução

O presente trabalho visa implantar no Núcleo de Práticas em Informática da Universidade Federal do Ceará do Campus de Quixadá (NPI) a utilização de uma ferramenta de Integração Contínua (IC), e relatar a experiência obtida em sua implantação.

Entende-se Integração Contínua como uma ferramenta de gestão de configuração, que auxilia os desenvolvedores e permite que as mudanças realizadas no software sejam imediatamente avaliadas, testadas, verificadas de modo a prover um *feedback* imediato para correção de possíveis erros de integração que somente seriam verificados futuramente após problemas mais complexos de integração (DUVAL; MATYAS; GLOVER, 2007).

Para esta experiência foi utilizado uma ferramenta de integração contínua automatizada, *Jenkins*, como ambiente de aplicação o Núcleo de Práticas em Informática (NPI) da Universidade Federal do Ceará (UFC) do campus de Quixadá.

O NPI é o local onde estudantes que estão em ano de conclusão de curso podem estagiar e aprimorar seus conhecimentos adquiridos no decorrer do curso além de concluir seus componentes curriculares obrigatórios. Este surgiu devido à pouca demanda de empresas de Tecnologia da Informação (TI) na região onde a universidade se encontra, em Quixadá no Ceará.

Dentro do NPI, os projetos desenvolvidos têm como objetivo construir soluções que facilitem as atividades do cotidiano da universidade, esta que tem um grande interesse no desenvolvimento destes projetos, pois consegue reduzir custos ao priorizar construções de sistemas internamente (GONÇALVES et al., 2013). Em paralelo, aumenta a qualidade dos profissionais formados por ela, além de proporcionar um ambiente real de trabalho que facilita a entrada dos concludentes no mercado de trabalho.

No NPI existe um modelo de processo definido em que os desenvolvedores devem seguir para o exercício de suas atividades (GONÇALVES et al., 2013). Entretanto, este não é devidamente seguido, e não contempla dentro do processo a utilização de integração contínua, ocasionando uma despadronização na maneira como estes desenvolvedores trabalham em seus projetos. Somado-se a isto, o NPI apresenta problemas tais como: "Baixa qualidade da documentação dos sistemas; [...] Falta de uma equipe de manutenção; [...] Rotatividade dos profissionais"(PADUELLI; SANCHES, 2006, p. 4).

A experiência em aplicar ferramentas de gestão de configuração foi explanado por Oliveira e Nelson (2005). Eles inseriram a utilização destas ferramentas para evitar a inserção de novos erros proveniente de manutenções realizadas no software. Diferentemente do trabalho a ser desenvolvido nesse projeto que busca relatar a experiência obtida na implantação de uma ferramenta de integração contínua.

2 Trabalhos Relacionados

Nesta seção será descrito trabalhos que influenciaram os conceitos envolvidos neste trabalho além de demonstrar pontos comuns e distintos entre si e o proposto.

3 Objetivos

3.1 Objetivo Geral

Implantar uma ferramenta de integração contínua no Núcleo de Práticas em Informática da Universidade Federal do Ceará do campus de Quixadá.

3.2 Objetivos Específicos

- Selecionar um ferramenta de integração contínua automatizada.
- Coletar experiência obtidas com a implantação da ferramenta.

4 Fundamentação Teórica

Nesta seção será apresentado os conceitos necessários para um completo entendimento deste trabalho. A seção 4.1 conceitos inerentes a Gerência de configuração, juntamente com algumas ferramentas CASE e com ênfase em Integração Contínua descrito na seção 4.2 e por fim na seção 4.3 será descrito o processo de gerência de configuração do NPI.

4.1 Gerência de Configuração

A gerência de configuração é a área da engenharia de software responsável pela evolução do software. Ela atua durante todo o ciclo de vida do produto de software e, por meio de técnicas, ferramentas e metodologias, visa garantir que as mudanças que irão ocorrer dentro do ciclo de vida do desenvolvimento do software sejam identificadas, avaliadas e comunicada a todos os envolvidos através de ferramentas que auxiliam neste processo de evolução. Portanto "o propósito do processo de Gerência de Configuração é estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-la a todos os envolvidos" (SOFTEX, 2013).

4.1.1 Plano de Gerenciamento de Configuração

O Plano de Gerenciamento de Configuração (PGC) descreve todas as atividades de configuração e mudança que serão realizadas durante o projeto. Um conjunto de atividades, responsabilidades, ferramentas, recursos e etc. A gerência de configuração tem como objetivo garantir a integridade dos itens de configuração, que são qualquer artefato que esteja sob custódia da Gerência de Configuração, através do versionamento, da identificação, controlando mudanças e acesso.

4.1.2 Sistema de Controle de Versão

Um sistema de controle de versão: "[...] combina procedimentos e ferramentas para gerenciar diferentes versões de objetos de configurações que são criadas durante o processo de engenharia de software"(PRESSMAN, 2010, p. 927). Atualmente, o uso de sistema de controle de versão se tornou comum nas empresas de grande e pequeno porte. Tais ferramentas permitem que se tenha o controle de diferentes versões de arquivos que estão submetidos ao versionamento, recuperação de versões antigas, visualizar alterações realizadas em arquivos e saber por quem e quando o arquivo foi alterado. Através de comandos (i.e., *check-in*, *check-out*) os usuários conseguem se comunicar com o repositório a

fim de obter os artefatos ali armazenados (MENEZES, 2011). Em situações especiais faz-se necessário que os desenvolvedores trabalhem em uma linha diferente da original chamada de *mainline*, geralmente essa situação ocorre quando tem-se como objetivo consertar bugs de versões anteriores do repositório, nesse caso um *branch*, uma ramificação na linha de desenvolvimento do controle de versão, é criado afim de permitir a realização desta ação, permitindo assim o trabalho em paralelo sobre o mesmo repositório. A figura Figura 1

Figura 1 – Branch no Sistema de Controle de Versão.



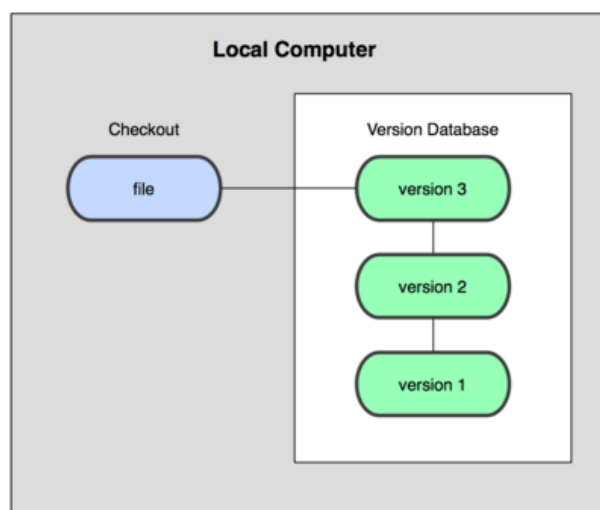
Fonte: Eis (2012).

demonstra a criação de um *branch* paralelo a linha de desenvolvimento principal chamada de *branch feature1* e *branch master* respectivamente, posteriormente as ações realizadas no *branch feature1* são incorporadas ao *branch master*.

4.1.2.1 Sistema de Controle de Versão Local

Um sistema de controle de versão local armazenam todas as informações de um arquivo submetido ao versionamento na máquina local, guardando diferentes versões daquele arquivo localmente como demonstrado na figura Figura 2.

Figura 2 – Sistema de Controle de Versão Local.

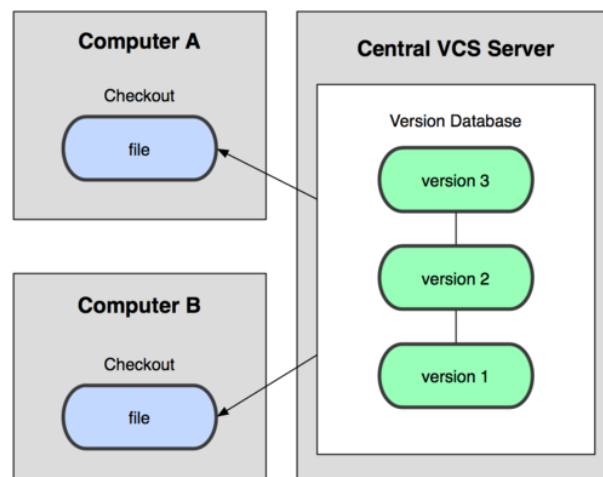


Fonte:(GIT, 20__).

4.1.2.2 Sistema de Controle de Versão Centralizado

Sistema de controle de versão centralizado como o nome diz possuem um único servidor centralizado, como o *subversion*¹, *perforce*², este tipo de padrão de SCV mantém em seu único servidor todos os arquivos versionados. Para cada comando de comunicação realizado nos arquivos versionados, uma requisição deverá ser feita, podendo gerar lentidão ou deixar o servidor fora de funcionamento. No exemplo acima dois desenvolvedores

Figura 3 – Sistema de Controle de Versão Centralizado.



Fonte: GIT (20__).

trabalhando em máquinas diferentes realizam a comunicação com o servidor central para obter o artefato de trabalho.

4.1.2.3 Sistema de Controle de Versão Distribuído

Os sistemas de controle de versão distribuído possuem um servidor central onde os arquivos são submetidos a versionamento, entretanto cada desenvolvedor possui em sua máquina de trabalho as versões que estavam no servidor, tornando cada *workstation* um "servidor", portanto, caso ocorra um problema no servidor central, estes podem ser recuperados via *workstation*, mantendo a integridade dos arquivos e evitando ser um ponto único de falha, como mostra a figura Figura 4.

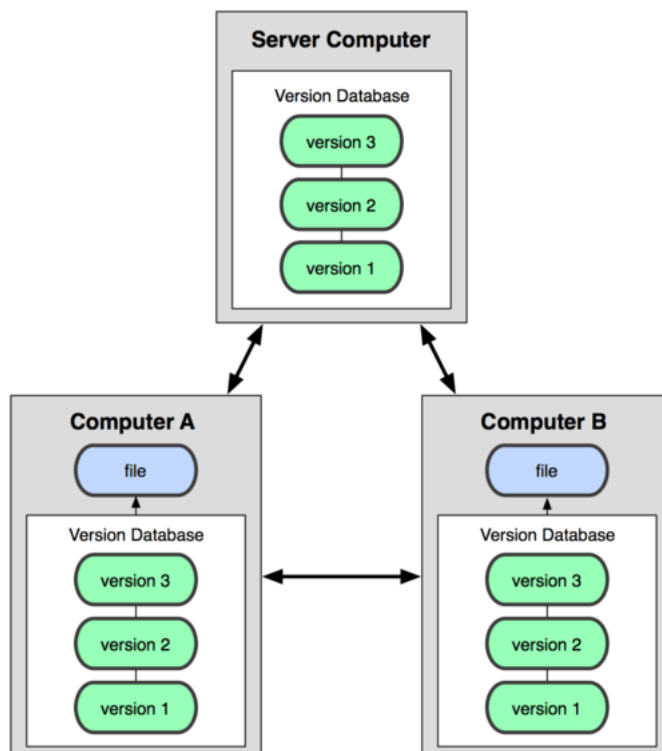
4.1.3 Sistema de Controle de Mudança

Todo software sofre mudanças, lidar com as mudanças é o papel da gerência de configuração, e para isso o gerente de configuração utiliza de um sistemas de controle de mudança. "O controle de mudança combina procedimentos humanos e ferramentas automatizadas para proporcionar um mecanismo de controle de mudança" Pressman

¹ <http://subversion.apache.org>

² <http://www.perforce.com>

Figura 4 – Sistema de Controle de Versão Distribuído.



Fonte: GIT (20__).

(2010, p .930). As mudanças devem ser avaliadas com cautela baseando-se, em seu custo benefício, uma combinação de esforço e *business value*. A mudança tem início quando um "cliente" solicita a mudanças através de um formulário, conhecido como *change request*. Nesse formulário é descrito os aspectos da mudança, após a solicitação ser realizada, esta deve ser avaliada, verificando se a mesma já foi solicitada, ou corrigida em caso de *bugs*. Após a mudança ser validada, uma equipe de desenvolvedores avaliam os impactos que esta mudança têm sobre o sistema, verificando custo/benefício e esforço de realização (SOMMERVILLE, 2011). Posterior a essa análise, a mudança será avaliada por um comitê de controle de mudança (CCB) que avaliará o impacto da perspectiva do negócio, o que decidirá se esta mudança será revisada, aprovada ou reprovada. Alguns sistemas que fornecem este controle sobre as mudança são: *redmine* ³, *GitHub* ⁴ *Jira* ⁵

4.1.4 Auditoria de Configuração

"Uma auditoria de configuração de software complementa a revisão técnica formal ao avaliar um objeto de configuração quanto às características que geralmente não são consideradas durante a revisão" Pressman (2010, p .934). Ela tem como objetivo garantir que mesmo com as mudanças realizadas a qualidade foi mantida. As auditorias se dividem

³ <http://www.redmine.org>

⁴ <http://www.github.com>

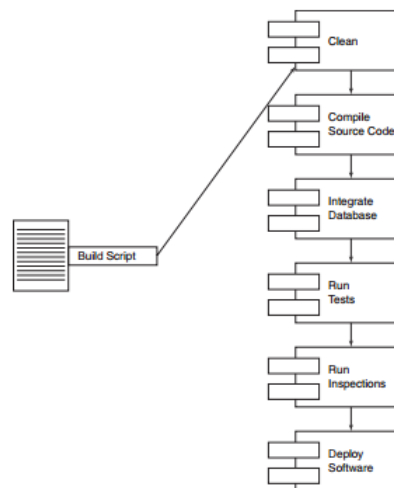
⁵ <https://www.atlassian.com/software/jira>

em dois tipos: auditorias funcionais e auditorias físicas, a auditoria física baseia-se em verificar se os itens de configuração estão devidamente atualizados e se as práticas e padrões foram realizados da maneira correta, enquanto a auditoria funcional, busca verificar os aspectos lógicos dos itens de configuração.

4.1.5 Ferramentas de Build

As ferramentas de build tem como objetivo automatizar processos repetitivos, aumentando a produtividade e facilitando o trabalho do desenvolvedor. Através da definição de uma rotina, ou conjunto de comandos, o desenvolvedor informa a ferramenta que tipo de processo ele deseja automatizar, pode ser desde compilar e testar uma classe, como dropar e criar uma tabela nova no banco de dados, comprimir arquivos css e javascript, cabe ao desenvolvedor definir o escopo da automatização. Alguns exemplo deste tipo de ferramenta são: *Ant*, *Grunt*, *Gulp*, *Maven*.

Figura 5 – Processo Lógico de uma Build.



Fonte: Duval, Matyas e Glover (2007).

Na figura Figura 5 um script foi definido para realizar as seguintes funções, será realizado um clean no projeto, compilará o código fonte, integrará com o banco de dados, executará testes e inspeções no código e por fim será realizado o *deploy* da aplicação.

4.2 Integração Contínua

A integração contínua tem como objetivo identificar erros o mais rapidamente, permitindo que alterações efetuadas e integradas aos repositórios dos sistemas de controle de versão (SCV) sejam posteriormente verificadas e caso erros ocorram, estes sejam notificados imediatamente ao autor da alteração. A melhor definição a cerca de integração contínua foi definida por Fowler (2000)

"[...] uma prática de desenvolvimento de software onde os membros de um time integram seu trabalho frequentemente, geralmente cada pessoa integra pelo menos diariamente – podendo haver múltiplas integrações por dia. Cada integração é verificada por uma build automatizada (incluindo testes) para detectar erros de integração o mais rápido possível. Muitos times acham que essa abordagem leva a uma significativa redução nos problemas de integração e permite que um time desenvolva software coeso mais rapidamente."Fowler (2000, tradução nossa).

4.2.1 Características de Integração

Os requisitos para utilizar uma ferramenta de integração contínua de acordo com Sepalla (2010) são:

- **Uma conexão com um sistema de controle de versão:**

A integração contínua necessita desta conexão, pois ela identifica as alterações ocorridas no repositório e dá início ao processo de integração.

- **A definição de uma build:**

A integração contínua possui uma build privada que será executada assim que o processo de integração for iniciado, e é esta build que define que ações serão realizadas no processo de integração, tais como compilação, testes, análise de código.

- **Um mecanismo de feedback:**

Um dos principais objetivos da integração contínua consiste em seu *feedback* imediato, sendo assim um mecanismo deste tipo é essencial para a ferramenta, tais como e-mail, sms.

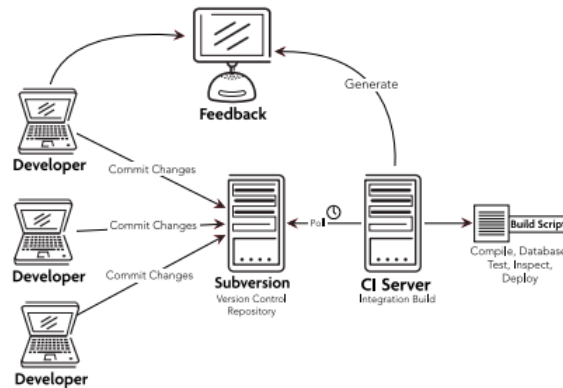
- **Um processo de integração do código:** O processo de integração consiste em como esta será realizada, manualmente ou através de um servidor de integração contínua de forma automatizada.

4.2.2 Processo de Integração

A integração ocorre quando alguma mudança é enviada ao sistema de controle de versão do repositório, que através de um servidor de integração contínua identifica a mudança e executa sua build privada (MRÁZ, 2013).

A figura Figura 6 descreve um ambiente em que um servidor de integração contínua é utilizado. Existem três ambientes de trabalho distintos formado por três desenvolvedores que obtiveram uma cópia do projeto do repositório do SCV para trabalharem em suas *workstation*, durante o trabalho alterações foram realizadas e commitadas ao repositório central, após a inserção junto ao repositório o servidor de integração contínua verifica as alterações e executa uma build de integração, caso exista um problema com a build e esta

Figura 6 – Ambiente de Integração Contínua.



Fonte: Duval (20__).

quebre, o responsável pela alteração será informado sobre a quebra e terá como objetivo consertar a build.

4.2.3 Benefícios da Integração Contínua

As principais vantagens em utilizar um servidor de integração contínua segundo Duval, Matyas e Glover (2007, p. 29) são:

- **Redução de Riscos:** Através da detecção imediata de código quebrados, ou incorretos, reduz-se risco atrelados ao produto.
- **Redução de processos manuais repetitivos:** Um conjunto de tarefas são executadas automaticamente pela build privada do servidor de integração contínua.
- **Permitir melhor visibilidade do projeto:** Através da identificação de informações inerentes ao desenvolvimento, como frequência de códigos defeituosos, módulos mais complexos, permitindo maior gerenciamento do projeto.
- **Estabelecer uma maior confiança no produto do time de desenvolvimento:** Através da visualizações de mudanças bem sucedidas, os desenvolvedores sentem maior confiança ao realizarem mudanças.

4.2.4 Builds Automatizadas

Builds são rotinas de execução definidas com o objetivo de reduzir processos repetitivos. Durante o processo de desenvolvimento de um software muitas ações tendem a serem repetidas por parte dos desenvolvedores, utilizar o tempo para a realização de atividades que poderiam ser automatizadas, de forma manual, reduz a produtividade e preocupações com melhorias devido ao tempo "apertado". Somando-se a isso, uma build garante que tudo que está nela definido será executado, evitando assim, que determinada

ação seja esquecida, ou caso um novo membro entre na equipe uma explicação do que ele deve fazer, ou não esquecer de fazer, não faz-se necessário.

4.2.5 Integração Contínua Manual

Na IC manual o processo de integração é realizado individualmente, possibilitando que apenas um desenvolvedor realize check-in no repositório durante o intervalo de integração Menezes (2011). Este tipo de abordagem como permite que apenas uma pessoa realize o *check-in*, as integrações serão contínuas e seguidas e não paralelas, este tipo de abordagem garante uma maior confiabilidade das integrações, pois segue um padrão de integração, os itens do repositório possuem maior consistência e a garantia da estrutura do repositório é mantida (MENEZES, 2011).

4.2.6 Integração Contínua Automatizada

A integração contínua automatizada é auxiliada pelo uso de um servidor de integração contínua, que obtém do controle de versão as alterações realizadas e executa sua build privada com o objetivo de verificar possíveis erros gerados por essas modificações. Ver seção 4.2 Figura 6

IC Automática possui a vantagem de ser escalável e, deste modo, oferecer maior suporte ao trabalho colaborativo. Com a utilização de Servidores de IC, a responsabilidade de realizar construções da integração é retirada dos desenvolvedores. Portanto, os desenvolvedores podem realizar check-in sem a necessidade de conquistar a vez de integrar. Esse fator é fundamental para que os check-ins continuem sendo verificados sem a necessidade de um desenvolvedor realizar a construção e identificar problemas, resultando na eliminação do gargalo humano. Menezes (2011, p .54).

4.2.7 Processo de Escolha da Ferramenta

- **Suporte a Linguagem:**

O processo de escolha de um sistema de integração continua deve ser escolhido de acordo como a ferramenta da suporte a linguagem, visto que alguns sistemas são construídos para trabalharem como uma linguagem de programação específicas.

- **Suporte ao Sistema de Controle de Versão:**

Como explanado anteriormente a importância do SCV dentro de um sistema de integração contínua, portanto escolher uma ferramenta que integre-se com o repositório é essencial, pois algumas ferramentas fornecem suporte a SCV mais populares, como *Subversion*, *Git*, entretanto pode não haver suporte ao *Mercurial*.

- **Segurança:**

Garantir que somente pessoas autorizadas devem ter acessos aos artefatos existente no servidor de integração contínua.

- **Extensibilidade:**

Capacidade dá ferramenta ter funcionalidades adicionadas por meio de *plugins*, ser extensível.

- **Usabilidade:**

Possui baixa dificuldade de ações dentro da ferramenta, boa aprendizagem, compreensibilidade, GUI.

- **Instalação e Configuração:**

Facilidade de instalação, diferentes ambiente de operação, tais como sistemas operacionais, utilização de recursos. Documentação clara e objetiva do processo de instalação informando dependência existentes.

4.3 Processo do NPI

O NPI possui um modelo de processo definido, este processo é baseado nos modelos e metodologias Scrum, MPS.BR e XP. Este modelo define as práticas e o modelo de trabalho dos envolvidos nas atividades do núcleo. Dentro do modelo de processo definido no NPI ⁶ este trabalho tem como objetivo focar no modelo de processo de gerência de configuração.

4.3.1 Processo de Gerência de Configuração do NPI

O modelo de processo relacionado a gerência de configuração é descrito na figura Figura 7. Este modelo de processo possui duas atividade que serão descritas abaixo:

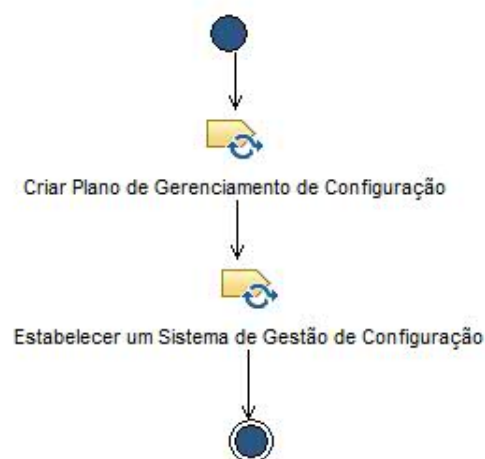
- **Criar Plano de Gerenciamento de Configuração:** Esta atividade é realizada pelo líder técnico da equipe envolvida. Esta atividade subdividi-se em quatro etapas são elas:
 - **Identificar Itens de Configuração:** Esta atividade caracteriza-se pela criação, especificação e seleção dos produtos de trabalho, ferramentas, itens que tem objetivo descrever os produtos de trabalho. Exemplos de itens desta atividade são: Requisitos, Diagramas, Testes.
 - **Atribuir Identificadores únicos para os itens de configuração:** Esta atividade possui um nome bem sugestivo tem como intuito atribuir a cada

⁶ <http://www.npi.quixada.ufc.br/processo/>

item de configuração um identificador único de modo a facilitar a identificação dentro do projeto. O identificador segue o padrão *[PROJETO]-[TIPO]-EXTRA.EXTENSÃO*. Como exemplo um artefato possuiria o seguinte identificador: *[GPA]-[REQ]-Especificacao.doc*

- **Identificar o responsável por cada item de configuração:** Esta atividade tem como objetivo atribuir a cada item de configuração um responsável, permitindo assim, uma maior facilidade na identificação do responsável de um determinado item de configuração.
- **Criar Plano de Gerenciamento de Configuração:** Esta atividade tem como objetivo a elaboração do PGC explicado na subseção 4.1.1 por meio dos dados obtido com as tarefas anteriores. O plano define os responsáveis pelas atividades de Gerência de Configuração, ferramentas e ambientes a serem utilizados e todos os itens de configuração identificados.
- **Estabelecer um sistema de Gestão de Configuração:** Esta atividade tem como requisito que o plano de gerenciamento de configuração esteja concluído, e possui apenas uma etapa:
 - **Estabelecer um sistema de Gestão de Configuração:** Esta atividade tem como objetivo definir as ferramentas de acesso, ambiente de armazenamento e métodos para criação e alteração dos itens de configuração (NPI, 201__).

Figura 7 – Processo de Gerenciamento de Configuração.



Fonte: NPI (201__).

5 Procedimientos Metodológicos

5.1 Cronograma de Execução

Tabela 1 – Cronograma das atividades previstas

ATIVIDADES	2014						
	Mai	Jun	Jul	Ago	Set	Out	Nov
Estudo de Campo	X	X					
Defesa do Projeto		X					
Avaliação do Processo do NPI			X	X			
Avaliação da Manutenibilidade do GAL				X	X		
Implantação da Integração Contínua				X	X		
Treinamento do uso da Ferramenta					X	X	
Avaliação da Manutenibilidade do GAL						X	
Análise do Dados de Manutenibilidade						X	
Revisão final da monografia						X	X
Defesa do Projeto							X

Fonte: Elaborado pelo Autor

Referências

- DUVAL, P. M. Continuous integration: Patterns and anti-patterns. *DZone Refcardz*, n. 1, p. 1–6, 20__.
- DUVAL, P. M.; MATYAS, S.; GLOVER, A. *Continuous Integration: Improving Software Quality and Reducing Risk*. 1. ed. [S.l.]: Pearson Education, 2007.
- EIS, D. Janeiro 2012. Disponível em: <<http://tableless.com.br/introducao-das-premissas-dos-controles-de-versao/>>. Acesso em: 27.3.2014.
- FOWLER, M. Maio 2000. Disponível em: <<http://martinfowler.com/articles/continuousIntegration.html>>. Acesso em: 27.3.2014.
- GIT. 20__. Disponível em: <<http://git-scm.com/book/pt-br/Primeiros-passos-Sobre-Controle-de-Vers~ao>>. Acesso em: 27.3.2014.
- GONÇALVES, E. J. T. et al. Núcleo de práticas em informática: Contribuindo para a formação em sistemas de informação através do desenvolvimento de projetos de software. *XXXIII CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (WEI*, Maceió, n. 1, p. 601–610, 2013.
- MENEZES, G. G. L. de. *OURIÇO: UMA ABORDAGEM PARA MANUTENÇÃO DA CONSISTÊNCIA EM REPOSITÓRIOS DE GERÊNCIA DE CONFIGURAÇÃO*. Niterói: [s.n.], 2011.
- MRÁZ, M. *Use of continuous integration in web application development*. Dissertação (Mestrado) — Masarykova Univerzita Fakulta Informatiky, Brno, 2013.
- NPI: Npi. 201__. Disponível em: <<http://www.npi.quixada.ufc.br/processo/>>. Acesso em: 20.4.2014.
- OLIVEIRA, P. A. de; NELSON, M. A. V. Integração de ferramentas para minimizar erros provenientes de efeitos colaterais inseridos durante a manutenção. *Workshop de Manutenção de Software Moderna*, n. 1, 2005.
- PADUELLI, M. M.; SANCHES, R. Problemas em manutenção de software caracterização e evolução. *III Workshop de manutenção moderna*, 2006.
- PRESSMAN, R. S. *Engenharia de Software*. [S.l.]: Pearson Education, 2010. 1056 p.
- SEPALLA, A. *Improving software quality with Continuous Integration*. Dissertação (Mestrado) — Aalto University, Esbo, Maio 2010.
- SOFTEX. *Guia de Implementacao - Parte 2: Fundamentacao para Implementacao do Nivel F do MR-MPS-SW:2012*. [S.l.], 2013. Disponível em: <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_de_Implementacao_Parte_2_2013.pdf>. Acesso em: 17.4.2014.
- SOMMERVILLE, I. *Engenharia de Software*. 9. ed. [S.l.]: Pearson Education, 2011. 529 p.