PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS

ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO.

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS.



PLANO DE GERÊNCIA DE CONFIGURAÇÃO

BRUNO CAMARGO MANSO

JOÃO VICTOR CARDOSO DE OLIVEIRA

NIKOLLY CARDOSO DE FARIA

GOIÂNIA, GO

2020

BRUNO CAMARGO MANSO

JOÃO VICTOR CARDOSO DE OLIVEIRA

NIKOLLY CARDOSO DE FARIA

PLANO DE GERÊNCIA DE CONFIGURAÇÃO

Trabalho que compõe as notas da N2 em

Gerência de Configuração de Software

Orientador: Joriver Canedo

GOIÂNIA, GO

2020

**Resumo**

Este trabalho tem como objetivo elaborar um plano de Gerência de Configuração a fim de ampliar o conhecimento do grupo a respeito de como é elaborado um plano, quais etapas são necessárias e logicamente qual a justificativa para a elaboração do mesmo. Fora elaborado a partir de um *template* guia, para uso com o *Rational Unified Process.*

**Palavras chave:** plano de Gerência de Configuração, etapas, justificativas, template guia.

**Sumário**

[1. Introdução](#_k3ta57wx3hzy) **5**

[1.1. Propósito](#_754x3hbt5f4s) **5**

[1.2. Escopo](#_wfcwtqbglw9a) **6**

[1.3. Definições, Acrônimos, e Abreviações](#_vwklm884f9un) **6**

[1.4. Referências](#_ozjz44pkbycf) **8**

[1.5. Visão Geral](#_apjzhbiyoe6a) **10**

[2. Gerência de Configuração de Software](#_de6p0m92vi04) **10**

[2.1. Organização, Responsabilidades, e Interfaces](#_1bm60zfuli24) **10**

[2.2. Ferramentas, Ambiente, e Infraestrutura](#_o86hhsgmmww7) **12**

[3. O Programa de Gerência de Configuração](#_3b9x1q814k28) **13**

[3.1. Identificação de Configuração](#_bzv701ueevpk) **13**

[3.1.1. Métodos de Identificação](#_uga7jmjori43) **13**

[3.1.2. Linhas básicas do Projeto](#_27ks5ns8jxfm) **15**

[3.2. Controle de Configuração e Mudanças](#_8r3gdzdvmybu) **16**

[3.2.1. Requisição, Processamento e Aprovação de Mudanças](#_ryh1ib1sa4ud) **16**

[3.2.2. Change Control Board (CCB)  - Conselho de controle de mudanças](#_9qasuxn96sit) **17**

[3.3. Contabilização do Status da Configuração](#_a5z45daa8nt7) **17**

[3.3.1. Arquivamento de Mídia do Projeto e Processo de Release](#_15j52pha0o0k) **17**

[3.3.2. Relatórios e Auditorias](#_3zt2bslunp2o) **18**

[4. Milestones](#_gphrixdg5e2x) **19**

[5. Treinamento e Recursos](#_hd1rw8xdd7r6) **19**

[6. Controle de Subcontratante e Vendedor de Software](#_y09mdr6k88g1) **20**

#### **1. Introdução**

O Gerenciamento da Configuração é o processo pelo qual o conteúdo, a mudança e o status de informações em um projeto são geridos e controlados. O sucesso de um projeto é altamente dependente de um bom gerenciamento da configuração. Além disso, mudanças são corriqueiras em projetos de desenvolvimento – à medida que um projeto avança, mudanças no planejamento original ou no produto são naturais e freqüentes; para isso, é necessário um bom planejamento.

Neste documento será apresentado um plano de gerência de configuração, onde, seu 1º tópico traz uma visão geral, seus objetivos, definições relativas ao assunto e referências.

No 2º tópico temos a Gerência de Configuração sendo dividida em duas importantes partes: organização, responsabilidade e Interfaces, e ferramentas, ambiente e infraestrutura; este tópico busca detalhar como acontecem as requisições e análises de mudanças, assim como, a forma que elas serão feitas (conteúdo do tópico 2.2).

Já o tópico 3º, traz informações sobre o programa de gerência de configuração, que pode ser dividido em três partes: 1- Identificação de configuração; 2- controle de configuração e mudanças; 3- Contabilização do status da configuração. Esse tópico visa definir os métodos de identificação dos artefatos de desenvolvimento e Hardware; definir os processos de aprovação de mudanças; e realizar relatórios, documentando tudo o que foi modificado detalhadamente.

No tópico 4º pode-se observar como trabalharemos com os milestones, em resumo, tudo o que foi feito será computado em um novo milestone. “Treinamentos e Recursos” serão tratados no 5º tópico.

E no 6º temos o “Controle de Subcontratante e Vendedor de Software” que pode ser definido como o controle que há ao licitar e contratar as empresas distribuidoras de software.

#### 1.1. Propósito

Este plano de gerência tem por finalidade ser um guia durante o plano de gerenciamento de configuração através do estabelecimento de políticas, procedimentos e documentação para o planejamento, desenvolvimento, gerenciamento, execução e controle do cronograma do projeto. Ele irá abranger exemplos de códigos fontes que precisam ser, de alguma maneira, armazenados e devidamente compartilhados com colaboradores de um mesmo grupo de trabalho.

Também serão apresentadas ferramentas de versionamento e de configuração de software, assim como iremos exemplificar a maneira e as técnicas de como se planeja a gerência de configuração.

#### 1.2. Escopo

Para este plano de gerência inclui-se apenas planos classificados como modificações ‘cosméticas’ da interface do usuário, não está inclusa a verificação de outras camadas, como a camada de persistência e modelo, por exemplo. Qualquer outro tipo de de modificação não será tratado. Será analisado pelo tempo definido no cronograma estabelecido pela gerência em método **ágil.**

#### 1.3. Definições, Acrônimos, e Abreviações

***Check out:*** A retirada de um ou vários itens de configuração que estão localizados em um repositório para trabalhar em cima desses itens é denominada check out, o item continua presente no repositório, é feito apenas um “clone” do mesmo para o espaço de trabalho de quem for mexer em tais itens de configuração.

***Branch:*** é uma ramificação no desenvolvimento, usada para descrever o processo de divisão dos arquivos de um projeto em linhas de desenvolvimento independentes. Podendo servir para teste de uma nova funcionalidade ou para projetos destinados a um cliente específico.

***Open source:*** Software de código aberto (tradução para open source) é o software de computador com o seu código fonte disponibilizado e licenciado com uma licença de código aberto no qual o direito autoral fornece o direito de estudar, modificar e distribuir o software de graça para qualquer um e para qualquer finalidade.

***Back end:*** Back End, como o próprio nome sugere, vem da ideia do que tem por trás de uma aplicação, a maior parte ou se não toda a parte que envolve processamento em aplicações é proveniente do back end.

***Front end:*** Podemos classificar como a parte visual de um site, aquilo que conseguimos interagir. Quem trabalha com Front End é responsável por desenvolver por meio de código uma interface gráfica, normalmente com as tecnologias base da Web.

***Git:*** é um sistema de controle de versão de arquivos. Através deles podemos desenvolver projetos na qual diversas pessoas podem contribuir simultaneamente no mesmo, editando e criando novos arquivos e permitindo que os mesmos possam existir sem o risco de suas alterações serem sobrescritas.

***Confidencialidade****:* propriedade da informação que não estará disponível ou divulgada a indivíduos, entidades ou processos sem autorização; garantia do resguardo das informações dadas pessoalmente em confiança e proteção contra a sua revelação não autorizada.

***Cartões SSD:*** O SSD (solid-state drive) é uma tecnologia de armazenamento considerada a evolução do disco rígido. Ele não possui partes móveis e é construído em torno de um circuito integrado semicondutor, o qual é responsável pelo armazenamento, diferentemente dos sistemas magnéticos.

***Release:*** fim de um ciclo de desenvolvimento; Release é a entrega final de uma versão de um produto, versão que o usuário final irá utilizar.

***Armazenamento a frio:*** se refere a manter os operações armazenadas em meios físicos ou em dispositivos que não precisem estar conectados à internet.

***Fidedignidade:*** Característica do que é confiável, que confirma algo pré-definido.

***Modelo ágil:*** conjunto de técnicas e práticas para gestão de projetos que oferece mais rapidez, eficiência e flexibilidade. Seu objetivo inicial era agilizar o desenvolvimento de softwares, mas esses métodos extrapolaram o setor de tecnologia e hoje revolucionam a gestão de empresas de todas as áreas.

***Workshops:*** evento onde acontece uma reunião de pessoas interessadas em determinado assunto para aperfeiçoar técnicas por meio da explicação de palestrantes e de atividades práticas.

***Milestone:*** "ponto ou evento significativo no projeto, programa ou portfólio". Também podem ser conhecidos como ponto de controle ou ponto de verificação.

***Baseline:***linha de base do projeto, que significa um ponto de partida muito bem definido, ou seja, uma referência para outros projetos, composta pelos itens escopo do projeto, cronograma e orçamento.

***Timeline:*** forma gráfica e linear de representar uma sequência de eventos em ordem cronológica. Timelines - linha do tempo em português - são tipicamente usadas para descrever uma determinada sucessão de fatos.

#### 1.4. Referências

Nesta sessão iremos demonstrar, em tabela, como será feito o registro das atividades e quais documentos são necessários em cada etapa, conterá um código referencial de cada uma das atividades. Todas elas vão gerar documentos específicos e devem ser alimentados conforme modificações são feitas. A tabela também irá conter o cronograma, facilitando procedimentos ágeis. Conterá também qual departamento, ou entidade, responsável por cada documento.

Cada documento incluído será criado um novo *milestone* e no despacho, ou release, de cada artefato modificado, passado o documento de requisição, de análise, relatório de qualidade e de auditoria, logicamente, após passar por todas aprovações devidas, um novo *baseline* será criado que incluirá todo processo documental bem como todos os milestones criados até o término da modificação.

|  |  |  |
| --- | --- | --- |
| Documentos | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Título | Código | Cronograma | Organização da publicação |
| Configuração e produção de documentos | **CPD** | 25/11/2020 | Comitê de gerência de configuração, Comitê de qualidade de software, Comitê de auditoria. |
| *Milestone* | **MS1** | 25/11/2020 | Comitê de gerência de configuração |
| Requisição de Modificação | **RM1** | 27/11/2020 | Cliente |
| *Milestone* | **MS2** | 28/11/2020 | Comitê de gerência de configuração |
| Análise de Modificação | **AM1** | 29/11/2020 | Comitê de gerência de configuração |
| *Milestone* | **MS3** | 30/11/2020 | Comitê de gerência de configuração |
| Relatórios de qualidade | **RQ1** | 01/12/2020 | Comitê de qualidade de software |
| *Milestone* | **MS4** | 02/12/2020 | Comitê de gerência de configuração |
| Relatório de auditoria | **RA1** | 03/12/2020 | Comitê de auditoria |
| *Milestone* | **MS4** | 03/12/2020 | Comitê de gerência de configuração |
| Atualização das *Baselines* | **AB1** | 04/12/2020 | Comitê de gerência de configuração |

#### 1.5. Visão Geral

Em nosso plano de gerência de configuração será exigido os seguintes passos: Solicitação de modiﬁcação; Classiﬁcação da modiﬁcação; Análise da modiﬁcação; Veriﬁcação da modiﬁcação; Impacto de mudanças; Análise de qualidade; Geração de baseline.

O documento está dividido em seis partes: Na primeira parte é dada uma introdução ao assunto, o propósito do plano, suas referências e algumas definições de palavras chave; Na segunda parte temos mais detalhadamente como acontecem as requisições e análises de mudanças; Na terceira parte é onde são detalhados os processos de análise de mudança, assim como sua aprovação, implementação, comunicação e documentação da mudança através de relatórios; Na quarta parte falamos sobre os milestones e como serão utilizados durante o processo; A quinta parte, é referente a treinamentos e recursos; E a sexta aborda a situação de compra de e venda de software, onde as empresas participam de licitações e firmam contratos de prestação de serviços.

#### **2. Gerência de Configuração de Software**

#### 2.1. Organização, Responsabilidades, e Interfaces

**O gerente de configuração de software:**

O gerente de configuração de softwareBruno foi designado para esta ocupação, ele irá filtrar a modificação, como foi dito, o plano de gerência será apenas voltado às modificações com classificação de cunho cosmético. Também iremos gerar alguns documentos e diversas baselines, por exemplo, o documento de análise da modificação, a seguir:

**

**Departamento de qualidade:**

Pessoas que atuam no departamento de qualidade farão a avaliação e a verificação da modificação. João que trabalha no departamento de qualidade ficou responsável de fazer avaliação e verificação da modificação neste projeto.

**Desenvolvedores:**

A desenvolvedora Nikolly irá receber o que será feito, refeito, modificado, adicionado ou excluído, irá posteriormente começar dando um **check out** no código completo ou uma parte dele que será modificada para o seu espaço de trabalho e assim dar início a sua tarefa de implementar a modificação.

**Observações:**

Lembrando que tudo isso será reunido nos milestones e nos relatórios, bem como em planilhas de acompanhamento que serão acompanhadas durante todo o projeto.

#### 2.2. Ferramentas, Ambiente, e Infraestrutura

**Ambiente:**

O ambiente computacional será *home office*, ou seja, cada desenvolvedor estará em suas casas ou local aonde desejam exercer suas tarefas fora da empresa e se necessário por problemas técnicos ou pessoais podem ir até a empresa e exercer suas tarefas de lá, porém o intuito é que façam suas atividades fora da empresa.

Os desenvolvedores devidamente alocados em *home office*em suas máquinas irão produzir branches pessoais e exercer suas atividades dentro do seu **branch** para obter um controle maior de suas responsabilidades.

**Ferramentas:**

Como repositório será usado a plataforma do **github**, onde conta com um servidor em cada máquina e um servidor online onde os códigos versionados serão armazenados. Os passos serão:

* Inicialmente é criado um repositório. Os desenvolvedores irão clonar o projeto principal
* São criados **branches** para edição e modificação
* As tarefas serão então divididas pelo gerente que autoriza as devidas modificações nos códigos
* Em seguida os desenvolvedores irão modificar onde compete a eles

**Eclipse v4.17**

Como desenvolvimento será utilizado a IDE Eclipse, uma ferramenta de desenvolvimento ***open source*** que permite utilizar a linguagem de programação Java que é a linguagem que será utilizada no **back end** e também permite trabalhar com HTML, CSS e JavasCript que serão linguagens que são utilizadas no **front end** da aplicação.

**Windows 10 home atualizado/Linux Ubuntu v20.04.1 LTS**

Como sistema operacional será usado Linux Linux Ubuntu v20.04.1 LTS e Windows 10 home atualizado, a escolha dessas tecnologias cabe ao programador escolher uma das duas e ambas estarão dentro do processo de gerência de configuração.

#### **3. O Programa de Gerência de Configuração**

#### 3.1. Identificação de Configuração

#### 3.1.1. Métodos de Identificação

#### 

Será levado em conta como método de identificação de artefatos de desenvolvimento a primeira e a segunda letra do artefato, acrescidas para maiúsculo para formar o marcador do artefato. Caso já haja algum elemento com identificação de primeira e segunda letras iguais, é pego a primeira e a terceira, caso já haja também, é pego a primeira e a quarta e assim por diante como no modelo abaixo:

|  |  |  |
| --- | --- | --- |
| Identificadores de Artefatos de Desenvolvimento | | |

|  |  |  |
| --- | --- | --- |
| Nome do artefato | Marcador | Numeração |
| Eclipse v4.17 | EC | 4.17 |
| Windows 10 home atualizado | WI | 10 |
| Linux Ubuntu v20.04.1 LTS | LI | 20.04.1 |

Será levado em conta como método de identificação de artefatos de Hardware. Como marcador será dado um nome após a aquisição do equipamento e a numeração será iniciada em 1.1 e irá incrementando de acordo com novas aquisições.

|  |  |  |
| --- | --- | --- |
| Identificadores de Artefatos de Hardware | | |

|  |  |  |
| --- | --- | --- |
| Hardware mínimo | Marcador | Numeração |
| Intel core i3, 4GB de memórias RAM, HD 120GB | Desktop-Dev1 | 1.1 |
| Intel core i3, 4GB de memórias RAM, HD 120GB | Desktop-Dev2 | 1.2 |
| Intel core i3, 4GB de memórias RAM, HD 120GB | Desktop-Dev3 | 1.3 |
| Servidor Dell local | Server-Empresa | 1.4 |

Será levado em conta como método de identificação de artefatos de Gerência. O marcador será dado pelas duas primeiras letras do primeiro nome do artefato mais a primeira letra do segundo nome caso haja, tudo em maiúsculo. A numeração será iniciada em 2.1 e irá incrementando de acordo com novos artefatos forem sendo adicionados.

|  |  |  |
| --- | --- | --- |
| Identificadores de Artefatos de Gerência | | |

|  |  |  |
| --- | --- | --- |
| Descrição | Marcador | Numeração |
| Plano de gerência | PLG | 2.1 |
| Cronograma do projeto | CRP | 2.2 |

#### 3.1.2. Linhas básicas do Projeto

A cada ***release*** será gerada uma baseline nos mesmos critérios adotados neste plano, quando se referir às mesmas modificações.

Ao definir um plano deve ser observado as antigas baselines e qualquer melhoramento será computado de base para os próximos projetos. Se essa alteração funcionar corretamente a mesma se tornará uma nova baseline já implementada com a última alteração de padrão de planejamento.

Uma nova linha de base será autorizada pelo gerente do projeto e irá conter todas as versões de arquivos e de diretórios que foram modificadas no subsistema ou subsistemas.

As baselines do projeto irão conter a seguinte nomenclatura:

<PROJETO>-BL-<VERSÃO><YYYY MM DD>

Onde,

<PROJETO> Identificador(Nome único) do projeto.

<VERSÃO> Versão da baseline de desenvolvimento.

<YYYY MM DD> Ano, mês e data da baseline.

Exemplos:

FRONT\_END\_EMPRESA - GCS-BL-001-20201117

FRONT\_END\_EMPRESA - GCS-BL-002-20201118

#### 3.2. Controle de Configuração e Mudanças

#### 3.2.1. Requisição, Processamento e Aprovação de Mudanças

3.2.1.1. Subprocesso de formalização

Inicia quando o solicitante preenche Formulário de Requisição de Mudança (FRM) e envia o mesmo para gerente do projeto. As mudanças devem ser apresentada e aprovada pelo gerente de configuração da equipe do projeto e pelos impactados pela mudança antes de ter sua implementação realizada.

3.2.1.2. Subprocesso de receber e analisar

Responsável por classificar a prioridade da mudança, no caso deste prono de gerência existe apenas a classificação “cosmética”. Realizar uma validação do correto preenchimento do formulário e do solicitante da mudança. Encaminhar mensagem de aprovado ou não para a solicitação de mudança.

3.2.1.3. Subprocesso de analisar impactos da mudança

É realizada pelo gerente e equipe do projeto. A partir do Formulário de Requisição de Mudança, é criado o Relatório de Controle de Mudança que será utilizado durante todo o tratamento da solicitação. Neste plano será analisado os Impactos de mudanças e análise de qualidade.

3.2.1.4. Subprocesso de aprovar e atribuir mudança

Após a realização da análise de impactos da mudança e seu registro no Relatório de Acompanhamento de Mudança devemos submeter ao solicitante para aprovação antes de iniciar seu desenvolvimento. Desta aprovação participam o gerente e a própria equipe do projeto, além de todos os envolvidos impactados pela mudança.

#### 

#### 3.2.2. Change Control Board (CCB)  - Conselho de controle de mudanças

(💩)

A partir do momento em que o cliente solicita a mudança de algum aspecto do software, emite então um pedido de mudança que será encaminhado para o CCB que é uma comissão de controle e mudança. O gerente é a figura responsável pela aprovação assim que a mesma ‘sobe’ ao quadro de mudanças e irá classificar o tipo de pedido foi feito, tratando as prioridades como altas, médias e baixas. No caso do presente plano, o gerente apenas irá tratar prioridades baixas e médias, pois o escopo das modificações, neste plano, fazem parte de apenas das alterações cosméticas solicitadas pelo cliente.

O gerente então irá aprovar, ou não, as mudanças solicitadas. Quando aprovada, a mesma seguirá em forma de uma solicitação de mudança de engenharia com isso, os devidos itens que compõe o projeto serão **clonados** de seu repositório para a área de trabalho do desenvolvedor escolhido pelo comitê. Um ***branch*** será então criado por ele, para que faça os devidos versionamentos das modificações.

Logo após, atividades avaliativas sobre a qualidade do software são feitas para que tenha as devidas garantias de qualidade nos serviços. Por fim, os itens em *branch* serão re analisados pela gerência do comitê de mudanças e reintroduzidos no repositório principal, logicamente após a aprovação geral da comissão.

#### 3.3. Contabilização do Status da Configuração

#### 3.3.1. Arquivamento de Mídia do Projeto e Processo de Release

Todos processos são sujeitos a alguma falha, podendo ser de **indisponibilidade** de dados, de **confidencialidade** e de **integralidade**. Para mitigar tais problemas toda a produção deverá ser armazenada tanto localmente, quanto em servidores, sendo estes servidores localizados na própria empresa (offline) e também localizados em nuvens de dados, no caso o **GitHub** (online).

Dados contidos em servidores ou em máquinas locais dos desenvolvedores estão devidamente armazenados em **cartões SSD**s. No caso dos servidores, tais cartões são introduzidos apenas quando está ocorrendo um backup ou quando precisa-se restaurar algum dado perdido, como alternativa ao GitHub, se caso haja um problema de indisponibilidade de rede.

Enquanto ao ***release*** dos artefatos, após os devidos versionamentos e após armazenamentos ao serem entregues na plataforma do **GitHub** já será suficiente para alimentar o servidor web (em nuvem). Um cartão SSD com os novos dados serão introduzidos permanentemente no servidor enquanto o que estava anteriormente introduzido será atualizado e desplugado. Chamado **armazenamento a frio** que seria uma analogia ao *cold wallet* do bitcoin porém em formato SSD.

Quaisquer mudanças efetuadas em configurações ou para instalação do artefato serão devidamente reportadas no documento de Readme.md disponível no **GitHub** que sempre será atualizado conforme existem ***releases*** novos, declarando suas mudanças e novos destaques de implementação.

#### 3.3.2. Relatórios e Auditorias

Todos bilhetes de Solicitação de mudanças emitidos, as análises com as devidas classificações de tipo e prioridade, aprovações e reprovações, bem como as soluções encontradas, onde foi alterado, o por quê da alteração, quem alterou e quando foi alterado., tudo isso será reunido em relatórios gerenciais em forma de planilhas. Isso irá garantir a integridade de todo resto dos documentos, sua **fidedignidade**, podendo por fim, alimentar também relatórios de auditoria.

Em caso de defeitos ou falhas graves encontradas, foi adotado o **modelo ágil**: planilhas serão alimentadas por prioridade, com os prazos estabelecidos (inicial e final) calculando, enfim, por quanto tempo tal requisição ficou aberta, dado importante para alimentar relatórios de qualidade e auditorias que fiscalizarão o desempenho sobre solução de problemas.

#### **4. Milestones**

Os marcos ou ***checkpoints*** serão feitos tanto sobre os clientes quanto sobre todo o processo de modificação. O cliente define o problema e também em qual prazo pretende obter a solução dele, com isso, um ***milestone*** é criado com a data do pedido, sua prioridade juntamente com o prazo limite e prazo de tolerância.

A partir daí, tal ***milestone*** aparecerá no quadro da comissão de gerência de modificação. Um novo ***milestone*** interno será assim criado, tendo como base o ***milestone*** do cliente.

Todas as etapas descritas acima, aprovações ou reprovações de solicitação, datas de bilhetes criados, tanto das requisições quando das análises, serão computadas neste novo milestone. A gerência de modificação irá subdividir tarefas, isso servirá para subdividir consequentemente o prazo, e distribuir atribuições entre os desenvolvedores. Então, cada tarefa terá um sub marco dentro dos ***milestones*** internos. Cada entrega aprovada também será um sub marco.

Tais marcações serão necessárias para definir futuras ***baselines*** e também tem objetivo de organização estrutural cartesiana e de impulsionamento do ***release*** produzido, dando um caráter horizontal ao processo com uma ***timeline*** interna como referência.

#### **5. Treinamento e Recursos**

Desenvolvedores usarão massivamente o sistema de versionamento do Git. A cada projeto, após análises das ***baselines*** de projetos anteriores, ***workshops*** serão definidos para a capacitação da equipe. É muito comum a necessidade de re capacitar um desenvolvedor ou até mesmo um gerente quando uma nova tecnologia é necessária. No entanto, é comum a administração de mini ***workshops*** quando, em uma ferramenta que já é conhecida, existe a necessidade de utilização de alguma ferramenta específica dentro dela.

Tais ***workshops*** podem ser ministrados online, em plataformas preexistentes de aprendizado como coursera ou allura, ou até mesmo por membros mais capacitados dentro da empresa.

#### 6. Controle de Subcontratante e Vendedor de Software

Licenças, chaves de ativação de softwares de terceiros que compõe o espectro de ferramentas utilizadas em toda a cadeia de processo serão previamente definidas pela gerência de projetos e aprovadas pelo comitê que inclui idealizadores, sócios e demais stakeholders. Assim, será possível garantir a compra das ferramentas escolhidas tendo o devido respaldo dos executivos envolvidos.

Uma empresa responsável por distribuir softwares será licitada e escolhida, com isso se firmará um contrato de prestação de serviços e que por fim acarretará na entrega e implementação do sistema, também feita por terceiros mas supervisionada pelo departamento de TI, garantindo a devida implementação, a integridade e disponibilidade dos dados preexistentes.

A distribuição do artefato, produto da empresa em questão, neste plano de gerência de configuração, também fica a cargo de outra empresa terceirizada. Que da mesma forma será licitada escolhida e contratada, para que assim efetue a devida distribuição do novo release.