UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
ESCOLA DE ADMINISTRAÇÃO E NEGÓCIOS

DANILO PALHETA NERY DA SILVA

Proposta de Gestão de Propriedade Intelectual para a Fábrica de Software da Faculdade de Computação da Universidade Federal de Mato Grosso do Sul

CAMPO GRANDE-MS
2018

DANILO PALHETA NERY DA SILVA

Proposta de Gestão de Propriedade Intelectual para a Fábrica de Software da Faculdade de Computação da Universidade Federal de Mato Grosso do Sul

Trabalho de Conclusão Final apresentado ao Programa de Pós-Graduação em Administração Pública em Rede Nacional da Escola de Administração e Negócios da Universidade Federal de Mato Grosso do Sul.

Orientação: Professor Doutor Jeovan de Carvalho Figueiredo.

CAMPO GRANDE-MS
2018

Proposta de Gestão de Propriedade Intelectual para a Fábrica de Software da Faculdade de Computação da Universidade Federal de Mato Grosso do Sul

Trabalho de Conclusão Final apresentado como exigência para a obtenção do título de Mestre em Administração Pública, à banca examinadora, no Programa de Mestrado Profissional em Administração Pública em Rede Nacional - PROFIAP – Universidade Federal de Mato Grosso do Sul, obteve conceito aprovado.

Campo Grande - MS, 27 de fevereiro de 2018.

BANCA EXAMINADORA

_____

Prof. Dr. Jeovan de Carvalho Figueiredo
(UFMS – Esan)
Orientador

_____

Prof. Dr. Marco Antônio Costa da Silva
(UFMS – Naviraí/MS)
Membro interno

_____

Profª. Drª. Ana Paula Martins Amaral

(UFMS – Fadir)
Membro externo

_____

Profª. Drª. Jane Dirce Alves Sandim Eleutério
(UFMS – Facom)

Membro externo

Agradecimentos

Durante a realização deste Trabalho de Conclusão Final, várias pessoas me ajudaram a executá-lo seja dando apoio moral, incentivo ou contribuições intelectuais. Gostaria de agradecer especialmente aos meus pais, Maurício e Sandra, por todo o apoio oferecido a minha formação pessoal, moral e intelectual. Também agradeço profundamente aos meus irmãos, Tiago e Diogo, por todo o apoio, inclusive me ajudando a refletir melhor sobre os temas referentes a este trabalho. Agradeço também a minha namorada, Lena, pela compreensão, carinho e apoio que foram fundamentais para a travessia dessa jornada.

Agradeço imensamente ao meu orientador, professor Jeovan Figueiredo, pela liberdade e apoio concedidos durante a execução deste trabalho. Suas considerações, *insights* e sugestões foram instrumentos valiosos para o desenvolvimento deste trabalho.

Agradeço também aos meus colegas de trabalho da Faculdade de Computação que contribuíram, direta ou indiretamente, para o andamento dos meus estudos no Mestrado Profissional em Administração Pública. Nesse sentido agradeço especialmente aos meus chefes na Secretaria de Apoio Pedagógico, Leonardo e Rosiane, que gentilmente me permitiram eu me ausentar algumas vezes do serviço para poder executar meus estudos e pesquisas no Mestrado. Agradeço também ao Guilherme do DIPT/Aginova pelas importantes contribuições ao trabalho.

Gostaria de agradecer também aos membros da segunda comissão de implementação da Fábrica de Software que me permitiram acompanhar os trabalhos da comissão. Dentre os membros, agradeço especialmente ao professor Bruno Cafeo, que, gentilmente, me concedeu diversas entrevistas/conversas para entender melhor os temas relativos à pratica de desenvolvimento de software.

Por fim, agradeço à oportunidade de ter estudado num programa de mestrado na UFMS, uma instituição pública que oferece ensino gratuito e de qualidade.

RESUMO

O presente Trabalho de Conclusão Final buscou contribuir para a melhoria dos processos gerenciais da Fábrica de Software da Faculdade de Computação da Universidade Federal de Mato Grosso do Sul (FS-Facom), ao propor um modelo de gestão da propriedade intelectual para a fábrica de software. Para alcançar esse objetivo, empreendeu-se uma análise sistemática das normativas referentes aos processos de PD&I da UFMS, bem como um levantamento de boas práticas no que concerne à gestão da propriedade intelectual em Fábricas de Software.Realizou-se uma pesquisa de abordagem qualitativa de caráter exploratório – descritivo, cujos métodos de coleta de dados foram a pesquisa bibliográfica, observação participante e pesquisa documental. Para analisar os dados relativos ao processo de PD&I no âmbito da UFMS, utilizou-se a técnica de análise qualitativa chamada de matriz conceitualmente agrupada (*conceptually clustered matrix*).As contribuições pertinentes à gestão da propriedade intelectual indicam a necessidade de articulação de boas práticas relativas ao gerenciamento ágil de projetos e ao gerenciamento de repositório de artefatos de software internos e externos.

Palavras-chaves: inovação, gestão de projetos,direitos autorais, programa de computador, software livre e de código aberto.

ABSTRACT

This dissertation aimed to enhance the management process of the UFMS Computing Faculty's Software Factory by proposing a software intellectual property management model. To reach this aim, this research undertook a systematic analysis of the UFMS RD&I regulations, as well as a good practice survey about software intellectual property management. The data collection of this qualitative research was undertaken by using documental research and participant observation methods. In order to analyze the gathered data, this research used a qualitative data analysis method called conceptually clustered matrix. The results of this research indicate that it is necessary to articulate good practices of two different dimensions - agile project management and the management of internal and external software artifacts - in order to achieve a sound software intellectual property management model.

Key words: innovation, project management, copyright, computer program, free and open source software

Lista de Figuras

Lista de Quadros

LISTA DE ABREVIATURAS E SIGLAS

BSDBerkeley Software Distribution

FACOMFaculdade de Computação da UFMS

FOSSFree and Open Source Software

FS-FACOM Fábrica de Software da Faculdade de Computação

GPL General Public License

ICTInstituição Científica, Tecnológica e de Inovação

INPI Instituto Nacional de Propriedade Industrial

IVPM2Iterative Visual Project Management Method

LGPL Lessen General Public License

MFEModelo de Fases e entregas do IVPM2

OMC Organização Mundial do Comércio

OMPI Organização Mundial de Propriedade Intelectual

PD&IPesquisa, Desenvolvimento & Inovação

PIPropriedade Intelectual

UFMS Universidade Federal de Mato Grosso do Sul

UFG Universidade Federal de Góias

TCUTribunal de Contas da União

Sumário

1. Introdução

O setor brasileiro de Tecnologia da Informação tem crescido significativamente nos últimos anos, chegando a faturar cerca de US$ 40 bilhões em 2016 (ASSOCIAÇÃO DE EMPRESAS BRASILEIRAS DE SOFTWARE, 2017). Nesse sentido, o setor de Tecnologia da Informação teve aumento de 9,2%, em 2015, apesar de a economia brasileira ter encolhido 3,8%, nesse mesmo ano (OLIVEIRA, 2016; ASSOCIAÇÃO DE EMPRESAS BRASILEIRAS DE SOFTWARE, 2016).

Não obstante esse vertiginoso crescimento do setor de Tecnologia da Informação, essa área tem enfrentado desafios consideráveis, principalmente com relação à escassez de recursos humanos (SOFTEX, 2013; CISCO, 2016). Nesse sentido, o relatório da consultoria IDC, patrocinado pela empresa de telecomunicações CISCO, conclui que, em 2015, o mercado brasileiro de Tecnologia da Informação apresentou *déficit* de cerca de 195 mil profissionais, de modo que as empresas do setor possuem esse montante de vagas ociosas porque não encontram mão de obra qualificada para preenchê-las (CISCO, 2016).

O problema da escassez de profissionais qualificados na área de Tecnologia da Informação apresenta causas de ordem quantitativa e qualitativa (SOFTEX, 2013). Com relação ao aspecto quantitativo, a quantidade de egressos dos cursos de Tecnologia da Informação é menor que a necessária para atender às demandas do setor, principalmente devido aos altos índices de evasão desses cursos de graduação (SOFTEX, 2013). No que se refere ao aspecto qualitativo da escassez de mão de obra, verifica-se que há profissionais teoricamente capacitados no mercado de trabalho, porém tais profissionais não atendem as expectativas das empresas do setor, principalmente aquelas especializadas em software e serviços correlatos (SOFTEX, 2013).

Com o intuito de dirimir esse problema, entidades do setor de software, como a Softex (Associação para Promoção da Excelência do Software Brasileiro), têm sugerido a criação de iniciativas conjuntas entre universidades e empresas do setor visando a aproximar os conhecimentos teóricos da realidade vivenciada no mercado de trabalho (SOFTEX, 2013).

Dessa forma, algumas universidades brasileiras decidiram criar fábricas de software acadêmicas com o objetivo de proporcionar um ambiente favorável para a aplicação prática dos conhecimentos adquiridos em sala de aula, de modo que os alunos possam ser capacitados adequadamente para desenvolver os sistemas de software demandados pelo mercado (ROMANHA, 2016).

Nesse contexto, o Curso de Engenharia de Software da Faculdade de Computação da Universidade Federal de Mato Grosso do Sul também decidiu criar uma fábrica de software acadêmica visando a proporcionar aos alunos experiência prática no desenvolvimento de software (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2015d).

1.1Fábrica de Software da Faculdade de Computação -UFMS

A concepção da Fábrica de Software da Faculdade de Computação da Universidade Federal de Mato Grosso do Sul surgiu com a criação do curso de Bacharelado em Engenharia de Software em 2015, cuja implantação foi autorizada pela Resolução n°. 80/2014 do Conselho Universitário da instituição (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2014). Segundo o Projeto Pedagógico do Curso, aprovado pela Resolução n° 740/2014 da Pró-Reitoria de Graduação, o curso de Engenharia de Software foi criado para atender a crescente demanda, no mercado regional e nacional, por profissionais da área de Tecnologia da Informação qualificados e aptos a analisar, projetar e desenvolver diferentes tipos de aplicações de software para diversos setores da sociedade (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2015).

A primeira menção da Fábrica de Software em documento oficial da Universidade Federal de Mato Grosso do Sul consta no Projeto Pedagógico do Curso de Engenharia de Software, o qual estabelece que ela seria um mecanismo para proporcionar experiência prática aos alunos sobre os problemas e os desafios enfrentados pelas empresas de desenvolvimento de software (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2015b). Naquele documento também já se previa que a Fábrica de Software poderia desenvolver soluções de software para atender as demandas internas da UFMS, bem como firmar parcerias com empresas, como a Embrapa, para desenvolvimento de projetos conjuntos (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2015b).

Com o intuito de implementar a Fábrica de Software da Faculdade de Computação (FS-Facom), a Direção da Faculdade de Computação constituiu duas comissões responsáveis por definir os procedimentos técnicos necessários para esta implementação (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2015c, 2016d).

Enquanto a primeira comissão, criada em 23 de fevereiro de 2015, teve como objetivo definir um modelo de gestão e processo padrão de desenvolvimento de software para a Fábrica de Software (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2015c), a segunda comissão, criada em 7 de Julho de 2016, ficou encarregada de definir processo

operacional para implantação e execução da Fábrica de Software (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2016d).

1.1.1 Primeira comissão de implantação da FS-Facom

A Direção da Faculdade de Computação tomou a primeira iniciativa para concretizar o projeto da Fábrica de Software, por meio da Instrução de Serviço n° 9, de 23 de fevereiro de 2015, que constituiu uma comissão responsável por definir modelo de gestão e processo padrão de desenvolvimento de software da Fábrica de Software (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2015c).

O resultado dos trabalhos dessa comissão se materializou em relatório técnico, datado em 10 de julho de 2015, intitulado "Modelo de Gestão e Processo Padrão de Desenvolvimento de Software da Fábrica de Software da Facom/UFMS" (FACULDADE DE COMPUTAÇÃO, 2015). Em termos gerais, esse relatório técnico estipulou o modelo de gestão de demandas, modelo de gestão de pessoal e o processo padrão de desenvolvimento de software da FS-Facom (FACULDADE DE COMPUTAÇÃO, 2015).

O relatório técnico também estabelece que as atividades da Fábrica de Software seriam desenvolvidas em duas disciplinas obrigatórias, totalmente práticas, denominadas Construção de Software I e Construção de Software II, que seriam conduzidas com a supervisão de um ou dois professores, denominados professores tutores, e, eventualmente,com a participação de profissionais das instituições parceiras formados na área de Computação (FACULDADE DE COMPUTAÇÃO, 2015).

Segundo o relatório técnico, antes de iniciar qualquer projeto de desenvolvimento de software, todos os envolvidos deveriam assinar um **Termo de Responsabilidade,** que visa garantir a confidencialidade das informações do projeto, além de assegurar que os envolvidos que não assinarem o **Termo de Cessão**, até o último dia letivo de cada semestre de execução do projeto, abririam mão da autoria do software[1] (FACULDADE DE COMPUTAÇÃO, 2015).

No que tange à titularidade da propriedade intelectual dos produtos desenvolvidos na Fábrica de Software, o relatório técnico estabelece que deveria constar no edital de demanda externa que os direitos patrimoniais serão repartidos em cinquenta por cento (50%) para instituição parceira e cinquenta por cento (50%) para a UFMS (FACULDADE DE COMPUTAÇÃO, 2015). Cabendo à instituição parceira indicar, no final do projeto, os nomes

[1] Essa questão da possibilidade de abrir mão da autoria do software será melhor analisada nas seções**3.2.1** e **3.2.2**, que abordarão o conceito de autoria e titularidade do programa de computador. Contudo, é pertinente antecipar que a legislação referente à propriedade intelectual de programa de computador não permite a cessão de autoria, mas sim da titularidade dos direitos patrimoniais desse bem imaterial.

dos autores que participarão do registro de software, bem como demais informações necessárias (FACULDADE DE COMPUTAÇÃO, 2015).

O relatório técnico também estipula que a parcela do direito patrimonial que cabe à UFMS será distribuída entre os membros que participaram do projeto de software, ou seja, acadêmicos e professores tutores (FACULDADE DE COMPUTAÇÃO, 2015). Embora o relatório técnico não tenha abordado qual percentual dos direitos patrimoniais da UFMS que será destinado aos autores dos programas de computador desenvolvidos na FS-FACOM, a Resolução n° 31/2004 do Conselho Diretor da UFMS, que regulamenta a política de inovação no âmbito da UFMS, estabelece que a instituição destinará um terço dos resultados financeiros obtidos da exploração de seus direitos de propriedade intelectual aos criadores dessas inovações (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2004).

Segundo o relatório técnico, a porcentagem de participação do acadêmico na produção intelectual do software será determinada pelos professores tutores no final do projeto, conforme a contribuição intelectual de cada acadêmico (FACULDADE DE COMPUTAÇÃO, 2015). Nesse sentido, recomenda-se que os professores utilizem a quantidade e a complexidade (alta, média ou baixa) dos artefatos de software entregues pelo acadêmico como parâmetro na mensuração da contribuição intelectual dos alunos, sendo que todos esses critérios deveriam constar no Termo de Responsabilidade (FACULDADE DE COMPUTAÇÃO, 2015).

Com relação à licença de uso dos softwares desenvolvidos na Fábrica de Software, o relatório técnico propõe que haverá dois tipos de licença, a saber: 1) Softwares desenvolvidos apenas pela UFMS e com qualquer instituição pública; e 2) Softwares desenvolvidos em parceria com instituições privadas (FACULDADE DE COMPUTAÇÃO, 2015).

1.1.2 Segunda comissão de implantação da FS-Facom

Apesar da relevância dos trabalhos realizados pela comissão supracitada no que concerne ao delineamento geral das atividades da Fábrica de Software, a Direção da Faculdade de Computação decidiu constituir outra comissão (doravante comissão de implantação), por meio da Instrução de Serviço n° 55/2016, com o intuito de definir o processo operacional para implantação e execução da Fábrica de Software da Facom (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2016d)[2]. Dessa forma, entre

[2]O autor deste TCF foi designado para participar da comissão instituída pela Instrução de Serviço n° 55, de 7 de julho de 2016.

julho e setembro de 2016, foram realizadas reuniões semanais para discutir e debater esses aspectos operacionais e práticos da FS-Facom[3].

Inicialmente, essa comissão teve como objetivo conceber mecanismos para integrar o conteúdo das disciplinas ofertadas no curso de Engenharia de Software com as atividades da Fábrica de Software, levando em conta seu objetivo pedagógico, bem como buscar meios para atrair a participação de professores na FS-Facom[4].

Uma das primeiras conclusões dos membros da comissão de implantação é que a Fábrica de Software deveria, no curto prazo, atender apenas demandas internas da UFMS, uma vez que, devido ao estágio incipiente da FS-Facom, seria arriscado firmar parcerias externas[5].

Dentre as reuniões realizadas pelos membros da comissão de implantação da FS, destaca-se uma reunião ocorrida no dia 24 de agosto de 2016, por meio de videoconferência, com o coordenador da Fábrica de Software do Instituto de Informática da Universidade Federal de Goiás (UFG), que foi criada em 2010[6]. Nessa reunião, cuja temática principal foi tratar dos desafios enfrentados pela FS-UFG, constatou-se que a falta de harmonia entre as disciplinas ofertadas no curso de Engenharia de Software e as atividades da FS-UFG gerou dificuldades, já que ambas competiam pela atenção e dedicação dos alunos[7].

Dessa forma, visando dirimir os problemas verificados na experiência dos primeiros anos de funcionamento da FS-UFG, os professores do curso de Engenharia de Software da UFG decidiram modificar significativamente o Projeto Pedagógico do Curso, para que as atividades da FS-UFG fossem realizadas no contexto de uma disciplina obrigatória com carga horária de 320 horas, chamada “Prática em Engenharia de Software”, ministrada em regime de internato com dedicação exclusiva do aluno, no último semestre do curso (UNIVERSIDADE FEDERAL DE GÓIAS, 2016).[8]

Inspirados pela experiência da FS-UFG, a comissão de implantação da Fábrica de Software também sugeriu uma série de mudanças relevantes no Projeto Pedagógico do Curso de Engenharia de Software da UFMS, com o intuito de integrar plenamente as atividades da FS-FACOM à vida acadêmica dos alunos[9]. Dessa forma, decidiu-se remover as disciplinas “Trabalho de Conclusão de Curso I” e “Trabalho de Conclusão de Curso II” na matriz

[3] As anotações realizadas por este autor, durante as reuniões da comissão de implantação da Fábrica de Software, serão referenciadas desta forma (NERY, D.P., 2016).
[4]NERY, D.P.Temática discutida entre os membros da comissão de implantação da FS na reunião de 30/06/2016.
[5]NERY, D.P. Temática discutida entre os membros da comissão de implantação da FS na reunião de 30/06/2016.
[6]NERY, D.P. Temática discutida entre os membros da comissão de implantação da FS na reunião de 24/08/2016.
[7]NERY, D.P. Temática discutida entre os membros da comissão de implantação da FS na reunião de 24/08/2016.
[8]NERY, D.P. Temática discutida entre os membros da comissão de implantação da FS na reunião de 24/08/2016.
[9]NERY, D.P. Temática discutida entre os membros da comissão de implantação da FS na reunião de 15/09/2016.

curricular do curso, a fim de substituí-las pelas disciplinas “Prática em Desenvolvimento de Software I” e “Prática em Desenvolvimento de Software I” com carga horária de 272 horas cada uma (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2017e).

Com isso, as atividades da Fábrica de Software transcorrerão no âmbito dessas disciplinas, as quais serão ministradas em regime de internato nos dois últimos semestres do curso, de modo que os alunos tenham uma imersão na vivência prática do desenvolvimento de software[10].

Após essas mudanças no Projeto Pedagógico do Curso de Engenharia de Software, definiu-se que as atividades da Fábrica de Software vão começar no primeiro semestre de 2018, quando alguns alunos da primeira turma ingressante do Curso cursarão a disciplina “Prática em Desenvolvimento de Software I”.

Cabe destacar que, em termos de subordinação hierárquica no âmbito da Faculdade de Computação, a Fábrica de Software é um laboratório desta unidade de administração setorial da UFMS, segundo consta a Instrução de Serviço n° 97/2017, que designa os coordenadores de laboratórios da Faculdade de Computação (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2017b).

Com o intuito de institucionalizar e legitimar o processo de tomada de decisões no âmbito da Fábrica de Software da Facom, a Direção da Faculdade de Computação, por meio da Instrução de Serviço n° 89/2017, constituiu uma Comissão Permanente encarregada de gerenciar a Fábrica de Software (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2017c).

Além disso, providenciou-se a publicação de um edital para seleção do logotipo para a Fábrica de Software da Facom, no qual puderam participar professores, acadêmicos e técnicos-administrativos vinculados à Faculdade de Computação (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2017e). O logotipo selecionado pode ser visualizado na **Figura 1**.

[10]NERY, D.P. Temática discutida entre os membros da comissão de implantação da FS na reunião de 15/09/2016

Figura 1 – Logotipo vencedor do concurso da Fábrica de Software da Facom/UFMS

Fonte: https://www.facom.ufms.br/resultado-do-concurso-de-logotipo-da-fabrica-de-software/

No que se refere especificamente à questão da propriedade intelectual no âmbito da Fábrica de Software da Facom, cabe destacar que esse tema foi discutido numa reunião, em 17 de novembro de 2017, entre alguns membros da comissão de implantação e o responsável pelo NIT da UFMS[11].

Tendo em vista a necessidade de formular um documento formal de cessão dos direitos patrimoniais (essa necessidade será abordada na seção **4.2.4**), foi apresentado nessa reunião um rascunho de um Termo de Compromisso elaborado por este autor[12], no qual os alunos matriculados nas disciplinas "Prática em Desenvolvimento de Software I" e "Prática em Desenvolvimento de Software II" se comprometem a ceder a titularidade dos direitos patrimoniais do software que será desenvolvido nas atividades da Fábrica de Software.

Dessa forma, após colher sugestões do responsável pelo NIT da UFMS e do Colegiado do Curso de Engenharia de Software, o autor deste Trabalho de Conclusão Final elaborou a versão final do Termo de Compromisso, que será analisado com mais detalhe na seção **6.2.2**.

[11] Segundo a Portaria n°489/2017 da Reitoria da UFMS, a Divisão de Propriedade Intelectual e Transferência de Tecnologia (DIPIT) da Agência de Desenvolvimento, Inovação e Relações Internacionais (AGINOVA) foi designada como o Núcleo de Inovação Tecnológica (NIT) da UFMS.

[12] O Termo de Compromisso anexo ao Regulamento das disciplinas "Prática em Desenvolvimento de Software I" e "Prática em Desenvolvimento de Software II" foi elaborado pelo autor deste TCF e constitui-se como um dos resultados da presente pesquisa. Na seção **6.2.2**, este Termo de compromisso será analisado com mais detalhe, principalmente a parte referente à propriedade intelectual.

Além disso, com o objetivo de normatizar as atividades que serão desenvolvidas no âmbito da FS-Facom, o Conselho da Faculdade de Computação emitiu a Resolução n° 146, de 13 de dezembro de 2017, que aprovou o Regulamento das disciplinas "Prática em Desenvolvimento de Software I" e "Prática em Desenvolvimento de Software II" (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2017b).

No regulamento dessas disciplinas, destacam-se a normatização dos seguintes temas: as atribuições do professor supervisor[13] e do professor consultor[14], a composição das equipes de alunos, os critérios de avaliação dos alunos e o Termo de Compromisso que deve ser assinado pelos alunos ao se matricularem nessas disciplinas (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2017b).

Para cumprir com o disposto no Art. 11[15] do Regulamento das disciplinas "Prática em Desenvolvimento de Software I" e "Prática em Desenvolvimento de Software II", que trata da seleção de projetos para a Fábrica de Software, a Comissão Permanente da Fábrica de Software providenciou edital para captação de projetos de software para serem desenvolvidos pelos alunos na Fábrica de Software (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2018).

Nesse primeiro edital de seleção de projetos de software, apenas professores, técnicos-administrativos e alunos vinculados a Faculdade de Computação puderam submeter propostas (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2018).

Essa decisão de inicialmente somente aceitar propostas de desenvolvimento de software do público interno da Facom visa a proporcionar uma experiência de desenvolvimento de software "mais segura", ou seja, evitar compromissos com parceiros

[13]Segundo o §3º do Art. 5° do Regulamento das disciplinas PDS I e II, as atribuições do professor supervisor são as seguintes: I - Acompanhar o projeto de desenvolvimento de software; II - Acompanhar as atividades dos acadêmicos; III - Informar às equipes correções de cada artefato entregue; e IV - Avaliar os acadêmicos.

[14]Segundo o §3º do Art. 6° do Regulamento das disciplinas PDS I e II, as atribuições do professor consultor são as seguintes: I - Participar das reuniões e das atividades de supervisão de equipes, quando solicitado pelo professor supervisor; II - Atuar como consultor nas fases do projeto em que possui maior expertise, visando contribuir com o desenvolvimento do projeto; e III - Avaliar os artefatos entregues pelas equipes, quando solicitado pelo professor supervisor.

[15]Art. 11 do Regulamento das disciplinas PDS I e II: "A comissão permanente da Fábrica de Software deve fornecer uma lista de propostas de projetos de software. É responsabilidade da Comissão Permanente da Fábrica de Software definir os critérios e elaborar a lista com as propostas de projeto de software. Parágrafo único. Cada proposta de projeto de software deve contemplar a descrição do software a ser desenvolvido" (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2017b).

externos, pelo menos, no início do funcionamento da Fábrica de Software da Facom[16]. Dessa forma, à medida que a FS-Facom adquire experiência e expertise no desenvolvimento de software, haverá condições de se firmar parcerias com instituições externas.

Nesse sentido, com o intuito de fornecer subsídios que auxiliem a Fábrica de Software da Facom firmar parcerias externas, o presente Trabalho de Conclusão Final empreenderá uma investigação sobre os processos de Pesquisa, Desenvolvimento & Inovação (PD&I), com foco na questão da gestão da propriedade intelectual.

1.2 Justificativa

Considera-se extremamente relevante tratar da questão da propriedade intelectual dos softwares que serão desenvolvidos na Fábrica de Software, já que a propriedade intelectual, numa análise preliminar, é o direito exclusivo de usufruir determinado bem intangível (ZIBETTI; ZIEGLER FILHO, 2014; BARBOSA, 2011). Dessa forma, examinar detalhadamente as normativas relativas à propriedade intelectual do software é fundamental não somente para assegurar eventuais direitos dos seus criadores (professores e alunos), mas também para fornecer subsídios que contribuam para que a Fábrica de Software da Facom esteja em conformidade com essas normativas.

Esse aspecto da conformidade com a legislação pertinente é essencial para o sucesso das atividades da Fábrica de Software da Facom, uma vez que a mesma é subordinada a uma entidade da Administração Pública Federal, a Universidade Federal de Mato Grosso do Sul (UFMS), que por sua vez, é obrigada a seguir o princípio da legalidade[17] imposto pelo Art. 37 da Constituição Federal (BRASIL, 1988).

Além disso, para aplicar adequadamente os dispositivos previstos na legislação pertinente à propriedade intelectual no contexto da FS-Facom, faz-se necessário compreender um fenômeno mais abrangente, a saber, as normativas relativas aos processos de Pesquisa, Desenvolvimento & Inovação (PD&I) conduzidos em Instituições Federais de Ensino Superior, particularmente a Lei de Inovação (Lei n° 10.973/2004) e as normas da UFMS que regulamentam o tema de PD&I.

---

[16]NERY, D.P. Temática discutida entre os membros da comissão de implantação da FS e o responsável pelo NIT UFMS na reunião de 17/11/2017.

[17] Segundo Di Pietro (2017), o princípio da Legalidade determina que a Administração Pública só pode fazer o que a lei permite. Dessa forma, este princípio constitui uma das principais garantias de respeito aos direitos individuais, visto que a lei, ao mesmo tempo em que os define, estabelece também os limites da atuação administrativa que tenha por objeto a restrição ao exercício de tais direitos em benefício da coletividade (DI PIETRO, 2017).

Dessa forma, este Trabalho de Conclusão Final pretende oferecer uma proposta para a gestão da propriedade intelectual dos softwares desenvolvidos em atividades de pesquisa, desenvolvimento & inovação (PD&I) no âmbito da FS-Facom, conforme os parâmetros das normativas pertinentes. Com isso, o presente estudo buscou responder a seguinte pergunta de pesquisa:quais práticas devem ser adotadas na gestão da propriedade intelectual de uma fábrica de software, considerando as normativas aplicáveis à pesquisa, desenvolvimento e inovação, especificamente para programas de computador ?"

A presente pesquisa tem como objetivo principal propor um modelo de gestão da propriedade intelectual de uma fábrica de software acadêmica vinculada a uma Instituição Federal de Ensino Superior.Para alcançar esse objetivo principal, este Trabalho de Conclusão Final empreendeu os seguintes objetivos secundários: a) frente à possibilidade de firmar parcerias externas, compreender e sistematizar as normativas referentes aos processos de PD&I, com ênfase naquelas normativas vigentes no âmbito da UFMS; b) realizar levantamento de boas práticas no que concerne à gestão da propriedade intelectual em Fábricas de Software; e c) propor mecanismo de integração entre a gestão da propriedade intelectual de software e os processos de PD&I no âmbito de Instituições Federais de Ensino Superior.

Com o intuito de lograr esses objetivos, o presente Trabalho de Conclusão Final realizou uma pesquisa de abordagem qualitativa de caráter exploratório – descritivo, cujos métodos de coleta de dados foram a pesquisa bibliográfica, observação participante e pesquisa documental. Para analisar os dados relativos ao processo de Pesquisa, Desenvolvimento & Inovação no âmbito da UFMS, utilizou-se a técnica de análise qualitativa chamada de matriz conceitualmente agrupada (*conceptually clustered matrix*).

Além desta introdução, o presente Trabalho de Conclusão Final se divide em sete seções nas quais são apresentadas a fundamentação teórica do fenômeno da inovação e suas contribuições para o desenvolvimento; a propriedade intelectual do programa de computador, principalmente no que concerne à sua autoria e titularidade; o conceito de fábrica de software; os procedimentos metodológicos utilizados neste trabalho; proposta de intervenção na FS-Facom por meio de análise qualitativa de dados utilizando uma matriz conceitualmente agrupada, juntamente com as contribuições para a Fábrica de Software da Facom, a proposição de um modelo de gestão de propriedade intelectual para a FS-Facom e as considerações finais.

2. A inovação e a sua contribuição para o desenvolvimento

O fenômeno da inovação tem sido apontado pela literatura como um dos principais fatores que ensejaram as grandes transformações tecnológicas, econômicas e sociais ocorridas na sociedade contemporânea desde meados do século XIX (ROTHWELL,1994; NOBELIUS, 2004; FREEMAN; LOUÇÃ, 2001). Dessa forma, não obstante o fato de o conceito de inovação ser um dos mais estudados pelos pesquisadores da área de Ciências Sociais (FAGERBERG, 2004), imputa-se a esse conceito uma miríade de significados, o que dificulta sua análise criteriosa (GODIN, 2008).

Com o intuito de se analisar adequadamente o fenômeno da inovação, é pertinente diferenciar os conceitos de invenção e inovação segundo os critérios do economista austríaco Joseph Schumpeter (SCHUMPETER, 1982), considerado pela literatura como um dos pioneiros nos estudos sobre a inovação (FREEMAN, 2003; CASTELLACCI, 2004). De acordo com Schumpeter (1982), enquanto a invenção se refere a um ato de criatividade intelectual - descoberta de uma ideia, novo produto ou processo; a inovação se refere à comercialização bem sucedida de um novo produto ou serviço oriundo de descobertas científicas. Desse modo, segundo Schumpeter (1982), o conceito de inovação abrange os seguintes casos: a) criação de novo produto; b) criação de novo método de produção; c) abertura de novos mercados; d) desenvolvimento de novas fontes fornecedoras de matéria primas e outros insumos; e) criação de novas estruturas de mercado em uma indústria.

A inovação, ao engendrar o processo de destruição criativa – na qual produtos e processos defasados são eliminados do mercado por concorrentes mais eficientes, exerce papel fundamental no desencadeamento de novos ciclos econômicos (SCHUMPETER, 1985). Com isso, Schumpeter (1935, p.4) define a inovação como uma nova combinação ou novo uso dos meios de produção existentes na forma de se fazer determinado produto ou processo, ou seja, “essa mudança histórica e irreversível na forma de fazer as coisas”. Ainda segundo Schumpeter (1989), somente a inovação, que é um fator endógeno do sistema econômico, possui o poder de irromper o estado de equilíbrio dos fluxos econômicos previamente estabelecidos. Desse modo, a inovação tem um papel fundamental na própria sobrevivência do sistema capitalista (SCHUMPETER, 1985, 1989).

Nesse sentido, os estudos sobre a inovação realizados por Joseph Schumpeter inspiraram o desenvolvimento de uma nova vertente da teoria do crescimento econômico (AGHION; HOWITT, 2009, 1998; GROSSMAN; HELPMAN, 1993). Ao contrario da

corrente neoclássica de crescimento econômico[18], na qual a dinâmica do progresso tecnológico não é considerada, o paradigma neoschumpeteriano reputa a inovação como a principal fonte do crescimento econômico no longo prazo (AGHION; HOWITT, 2009). Dessa forma, nesse paradigma, a inovação tecnológica é resultado de um esforço deliberado, por parte das empresas, visando à maximização de lucros, de modo que o crescimento econômico é a consequência dos recursos financeiros e humanos despendidos em atividades de Pesquisa e Desenvolvimento (AGHION; HOWITT, 2009, 1998; GROSSMAN; HELPMAN, 1993).

Com o intuito de estudar a dinâmica e os determinantes do processo de inovação tecnológica, emergiu, nos anos 1970 e 1980, estudos conduzidos por economistas neoschumpeterianos, dentre os quais destacaram-se o britânico Christopher Freeman e o dinamarquês Bengt-Åke Lundvall, que investigaram profundamente a formação dos sistemas de Pesquisa e Desenvolvimento dos principais países desenvolvidos, o que esses autores chamaram de "Sistema Nacional de Inovação[19]" (FREEMAN, 1974, 1987, 1995; LUNDVALL, 1988, 1992, 2002). Tal conceito tem como objetivo analisar o desenvolvimento de um arranjo institucional – que envolve firmas, universidades e institutos de pesquisa - que é a principal engrenagem responsável pelas inovações tecnológicas (FREEMAN, 1974, 1987, 1995; LUNDVALL, 1988, 1992). Com isso, os autores argumentam que, ao longo do século XX, a consolidação e profissionalização da atividade de pesquisa científica foram essenciais para a ascensão de uma economia baseada no conhecimento, uma vez que as inovações tecnológicas dependem cada vez mais dos resultados obtidos na pesquisas científicas oriundas tanto das ciências básicas como das aplicadas (FREEMAN, 1974, 1987, 1995; LUNDVALL, 1988, 1992). Nesse sentido, ao estudar a evolução histórica do sistema de Pesquisa e Inovação de países como Estados Unidos e Japão, esses autores defenderam que os governos deveriam priorizar a formulação de políticas públicas para promover a interação entre empresas, universidades e institutos de pesquisa, porquanto esse entrosamento é fundamental

[18] A corrente neoclássica, cujo principal fundamento teórico é o modelo de Solow (1970), afirma que o crescimento econômico é consequência de mudanças em duas variáveis, a saber, o capital (poupança e investimento) e a força de trabalho (através do crescimento populacional). Dessa forma, a corrente neoclássica não explica o fenômeno do progresso tecnológico, pois o considera um fator exógeno ao seu modelo explicativo. Não obstante, a corrente neoclássica reconhece que a interação do progresso tecnológico com aquelas duas variáveis clássicas contribuem para o crescimento econômico. Para mais detalhes sobre esse tema ver MANKIW, N. G. **Princípios de Macroeconomia**. São Paulo: Cengage Learning, 2010; SOLOW, R. M. **Growth theory**: An exposition**.**Oxford: Clarendon Press, 1970.

[19] As definições do conceito de **Sistema Nacional de Inovação** mais usadas na literatura são as de Freeman (1987): "uma rede de instituições nos setores público e privado cujas atividades e interações iniciam, importam, modificam e difundem novas tecnologias"; e a de Lundvall (1992), para quem o **Sistema Nacional de Inovação** é formado por "elementos e relações que interagem na produção, difusão e uso de novos conhecimentos economicamente úteis que estão localizados dentro das fronteiras de um Estado-nação".

para o desenvolvimento econômico e social de qualquer país (FREEMAN, 1987, 1988; LUNDVALL, 1988, 1992, 2007).

2.1 Tríplice Hélice

Outro modelo teórico que busca explicar o fenômeno da inovação resultante das relações entre universidade, empresa e governo é a "Tríplice Hélice", que foi concebida, em meados dos anos 1990, por Henry Etzkowitz e Loet Leydesdorff (ETZKOWITZ; LEYDESDORFF, 1995, 2000). Contudo, diferentemente do modelo do "Sistema Nacional de Inovação", que considera a firma como o principal motor da inovação, a "Tríplice Hélice" reputa à universidade um papel central na inovação em uma sociedade cada vez mais baseada no conhecimento (ETZKOWITZ; LEYDESDORFF, 2000).

Em face da evolução e complexidade dos diferentes tipos de arranjos institucionais que podem existir entre universidade, governo e empresa, o modelo da "Tríplice Hélice" pode ter três configurações: o modelo estatizante, o modelo "*lassaiz faire*" e o modelo da "Tríplice Hélice III" (ETZKOWITZ; LEYDESDORFF, 2000).

No modelo estatista, o governo assume o papel central de coordenar o sistema de inovação, relegando à universidade e à empresa papeis coadjuvantes (ETZKOWITZ, 2008; ETZKOWITZ; LEYDESDORFF, 2000). Segundo o modelo estatista, como o governo é o único ator do sistema de inovação que tem capacidade e recursos para liderar a criação de uma indústria com base tecnológica, as instituições de pesquisa são fortemente subordinadas ao governo central (ETZKOWITZ, 2008). Dessa forma, uma grande fragilidade desse modelo é o desestimulo à inovação *bottom up*, ou seja, aquelas iniciativas inovadoras vindas da base do sistema (ETZKOWITZ; LEYDESDORFF, 2000). Com diferentes graus de intensidade, o modelo estatizante foi aplicado na União Soviética, leste europeu, França e alguns países latino-americanos (ETZKOWITZ, 2008; ETZKOWITZ; LEYDESDORFF, 2000).

O modelo "*lassaiz faire*" tem como premissa uma nítida separação entre as esferas de atuação do governo, indústria e universidade, uma vez que cada esfera tem sua própria atribuição (ETZKOWITZ; LEYDESDORFF, 2000). Nesse sentido, a função da universidade consiste em produzir ciência básica e treinar profissionais, bem como gerar conhecimento através da publicação de artigos científicos (ETZKOWITZ, 2008). Contudo, uma das principais críticas ao modelo "*lassaiz faire*" é a falta de entrosamento entre esferas governamental, industrial e acadêmica (ETZKOWITZ; LEYDESDORFF, 2000).

O terceiro modelo - "Tríplice Hélice III" - é considerado o mais apropriado para gerar inovação, uma vez que enseja a criação de arranjos institucionais com diversos graus de

sobreposição entre as diferentes esferas (ETZKOWITZ, 2008; ETZKOWITZ; LEYDESDORFF, 2000).

No modelo da "Tríplice Hélice III", analisa-se o surgimento de uma infraestrutura de conhecimento, na qual existe uma sobreposição das diferentes esferas institucionais, onde cada organização passa a assumir o papel das demais. Como resultado desse entrosamento, emergem-se organizações híbridas, em que todos assumem as mesmas funções relativas à inovação (ETZKOWITZ; LEYDESDORFF, 2000).

No modelo da "Tríplice Hélice", destaca-se a atuação da chamada "universidade empreendedora" (ETZKOWITZ, 2008). Considera-se uma universidade como empreendedora quando ela é capaz de identificar as áreas de pesquisa e ensino prioritários visando à criação de centros de excelência, com o intuito de atrair recursos externos significativos (ETZKOWITZ, 2008). Além disso, uma universidade empreendedora deve ter a capacidade de compreender e lidar com os problemas e necessidades da sociedade, com o objetivo de criar bases para novos projetos de pesquisa (ETZKOWITZ, 2008).

Dessa forma, as principais características de uma universidade empreendedora são as seguintes:

> 1 - Liderança acadêmica capaz de formular e implementar uma visão estratégica;
> 2 - Controle legal dos recursos acadêmicos, incluindo propriedade física e propriedade intelectual resultante de pesquisa;
> 3 -Capacidade organizacional de transferir através de patentes, licenças e incubação;
> 4 - Espírito empreendedor entre administradores, professores e estudantes (ETZKOWITZ, 2008, p.27).

Nesse sentido, recomenda-se que as universidades empreendedoras identifiquem as áreas de pesquisa e ensino com maior potencial para se criar "ilhas de excelência", que terão maior capacidade de atrair financiamento e apoio externos (ETZKOWITZ, 2008).

2.2 Sistema nacional de inovação: O caso brasileiro

Embora o desempenho do Sistema Nacional de Inovação brasileiro apresente pouco dinamismo em relação aos sistemas dos países desenvolvidos, tal sistema engendrou casos de sucesso que contribuíram significativamente para o processo de industrialização do país durante o século XX, como a indústria aeronáutica, farmacêutica, siderúrgica e agrícola (SUZIGAN; ALBUQUERQUE, 2011). Uma das principais características do sistema brasileiro de inovação é a predominância de universidades públicas, mais especificamente as federais e as estaduais paulistas, nas atividades de Pesquisa e Inovação (POVOA, 2008; SUZIGAN; ALBUQUERQUE, 2011).

Povoa (2008), ao analisar dados referentes ao período entre 1999 e 2003, evidenciou que as universidades públicas eram as maiores depositantes de patentes no sistema brasileiro de inovação. Esse desempenho é bem superior ao das congêneres americanas, uma vez que a maior depositante de patentes brasileira foi a UNICAMP (Universidade Estadual de Campinas) naquele período, enquanto a maior universidade pública americana depositante de patentes - *The University of California* - ocupou a 19ª posição no sistema de inovação dos Estados Unidos.

Com o intuito de comparar o desempenho das universidades públicas brasileiras no que concerne ao registro de patentes de inovações junto ao Instituto Nacional de Propriedade Industrial (INPI), no período entre 1979 e 2004, o estudo de Povoa (2008) identificou que, entre as 41 universidades depositantes de patentes, sete universidades federais estão nas dez primeiras posições, a saber: Universidade Federal de Minas Gerais (UFMG), Universidade Federal do Rio de Janeiro (UFRJ), Universidade Federal do Rio Grande do Sul (UFRGS), Universidade Federal de Pernambuco (UFPE), Universidade Federal de Viçosa (UFV), Universidade Federal de São Carlos (UFSCAR) e Universidade de Brasília (UnB).

Além disso, as universidades financiadas pelo governo federal também se destacam na produção de conhecimento científico, uma vez que essas instituições de ensino superior contribuíram com 45% do total de publicações de artigos publicados em revistas indexadas nacional e internacionalmente em 2008 (CHIARINI, RAPINI, VIEIRA, 2014). Dessa forma, as universidades federais atuam não somente na formação de recursos humanos altamente qualificados, mas também são as principais responsáveis pelas atividades de pesquisa científica no país – juntamente com algumas universidades estaduais (BALBACHEVSKY; SCHWARTZMAN, 2011; CHIARINI; RAPINI; VIEIRA, 2014).

2.3 Relacionamento Universidade + Empresa

Estudos evidenciam que as universidades são atores fundamentais no âmbito do Sistema Nacional de Inovação de qualquer país, uma vez que contribuem significativamente para o processo de inovações das empresas (NELSON, 1990; GEUNA; MUSCIO, 2009). Não obstante, a literatura também aponta que as interações entre universidades e empresas são complexas e variam de acordo com a história e o nível de desenvolvimento de cada país, já que tal relacionamento é um processo dinâmico que reflete a heterogeneidade estrutural de cada Sistema Nacional de Inovação (CHAVES et al., 2015; SUZIGAN; ALBUQUERQUE, 2011).

Dessa forma, considerando-se o contexto latino-americano, o estudo de Arza (2010) propõe um arcabouço conceitual para analisar os tipos de relações existentes entre universidades e empresas, dependendo do tipo de canal pelo qual se desenvolve esse relacionamento, podendo ser um canal bidirecional, tradicional, de serviço ou comercial.

Segundo a pesquisadora argentina, os canais bidirecionais, ao proporcionarem fluxos de conhecimento vindos de ambas as direções, são os mais intensos porque envolvem a criação de novos conhecimentos e inovações tecnológicas por meio de projetos conjuntos de P&D, participação em redes de inovação ou em parques científico-tecnológicos (ARZA, 2010). Já o relacionamento tradicional universidade-empresa implica um fluxo unidirecional de conhecimento, geralmente das universidades para as empresas, por meio da contratação de recém-graduados, conferências e publicações científicas. Outro tipo de relacionamento universidade - empresa é o de serviços, no qual as instituições de ensino e pesquisa, em troca de recursos financeiros, oferecem consultorias, serviços de controle de qualidade, uso de equipamentos e testes para as empresas (ARZA, 2010).

O relacionamento comercial envolvendo universidades e empresas tem como característica primordial a comercialização dos resultados obtidos em pesquisas científicas, principalmente por meio de incubadoras e licenciamento e venda de produtos ou processos patenteados. Ainda segundo Arza (2010), cada tipo de interação universidade – empresa apresenta benefícios e riscos, cabendo aos governantes e autoridades formularem políticas públicas para alcançar o melhor equilíbrio.

Quadro 1 - Benefícios resultantes da interação universidade-empresa

Canais	**Benefícios**			
	Universidades		**Empresas**	
	Intelectual	Econômico	Curto prazo (Produção)	Longo prazo (Inovação)
Serviços		**X**	**X**	
Tradicional	**X**		**X**	
Bi-direcional	**X**			**X**
Comercial		**X**		**X**

Fonte: Extraído de Arza (2010, p. 477)

Baseado na classificação proposta por Arza (2010), o estudo de Chaves et al. (2015) estudou os tipos de interações entre Universidades e Empresas no Brasil, utilizando dados do Censo de 2004 do Diretório de Grupos de Pesquisa do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). Com relação ao perfil da amostra dos 1005 questionários respondidos por líderes de grupos de pesquisa que possuíam algum tipo de interação com empresas, é pertinente ressaltar que 79% desses grupos de pesquisa eram filiados a universidades públicas, ao passo que 13,7% e 6,5% eram filiados a universidades privadas e a institutos de pesquisa públicos, respectivamente (CHAVES et al., 2015).

O estudo de Chaves et al. (2015) mostrou que os principais tipos de interação Universidade - Empresa são projetos conjuntos de P&D de curto prazo (68,7%), consultoria (67.6%), e treinamentos e cursos (62.8%). Contudo, quando se analisa separadamente cada área de conhecimento, verificou-se que, para os grupos de pesquisa das áreas de engenharias, ciências agrárias, ciências da terra e exatas, os projetos conjuntos de P&D de curto prazo são muito mais importantes do que os serviços de consultoria. Diferentemente, o estudo evidenciou que, nas áreas biológicas e de saúde, os serviços de consultoria são predominantes (CHAVES et al., 2015).

Em outro estudo sobre as interações entre universidades e empresas em Minas Gerais, Rapini et al. (2009) demonstrou que as universidades exerciam um duplo papel no sistema de inovação: substituíam e complementavam as atividades de P&D das próprias firmas. Além disso, o estudo mostrou que as empresas consideravam as universidades como importantes fontes de informação com relação à sugestão de novos projetos e conclusão de projetos já existentes (RAPINI et al., 2009).

2.4 Lei de Inovação: Marco na cooperação universidade-empresa no Brasil

Com o intuito de ampliar e aprofundar essa interação entre universidades e empresas, considerada fundamental para o estímulo à inovação no ambiente produtivo do país, o governo federal promulgou a Lei n° 10.973/2004, conhecida como a Lei de Inovação (ARBIX; CONSONI, 2011). Com efeito, a entrada em vigor da Lei de Inovação marcou uma inflexão na política de ciência e tecnologia do Brasil ao regulamentar e incentivar o sistema de gestão da propriedade intelectual e de transferência de tecnologia na universidade brasileira (ARBIX; CONSONI, 2011).

Dessa forma, ao ensejar amparo legal para a criação de alianças estratégicas entre universidades e empresas, a Lei de inovação impulsionou a inovação e comercialização dos resultados das pesquisas científicas e tecnológicas (ARAUJO, 2012; BARBOSA, 2011).

Segundo Barbosa (2011), os 29 artigos da Lei de Inovação podem ser organizados em 4 grandes eixos temáticos:

> **"Constituição de ambiente propício às parcerias estratégicas entre as universidades e empresas**: o motivo condutor é propiciar a cooperação entre os atores do processo inovador, reduzindo as barreiras institucionais que impedem a *via de mão dupla* entre setor privado e ICTs. Trata-se, assim, de articulações *horizontais* entre os dois lados, e não, como no caso do Art. 19 e dos incentivos fiscais, concessão essencialmente unilateral de meios de inovação. A questão é genericamente introduzida pelo Art. 3°e implementado pelos Arts. 5° (Parcerias personalizadas em Sociedades de Propósitos Específicos) e 9° (Parcerias não Personalizadas). O Art. 4° prevê cooperação no uso de equipamentos e instalações.
>
> **Estímulo à participação de Instituições Científicas e Tecnológicas no processo de inovação**: Pelos Art. 6°, 7° e 8°, a lei faculta às ICTs a celebrar contratos de transferência de tecnologia e de licenciamento de patentes de sua propriedade e prestar serviços de consultoria especializada em atividades desenvolvidas no âmbito do setor produtivo. Os Arts. 14 e 15 induzem à mobilidade dos pesquisadores entre ICTs e sua transferência temporária ao setor privado, para os propósitos de estímulo à inovação. Pelo Art. 16 exige-se a criação nas ICTs federais de um órgão gestor das atividades de inovação e articulação, o chamado NIT (Núcleo de Inovação Tecnológica).
>
> **Normas de incentivo ao pesquisador-criador:** Os Arts. 8°, 11, 13 e 15 compreendem normas que se destinam a estimular a natureza especial do trabalho criativo. Os pesquisadores vinculados as ICTs beneficiar-se-ão do resultado financeiro dos serviços prestados sob o Art. 8, além da remuneração normal. Da mesma forma, enquanto criador ou inventor, o pesquisador participará dos ganhos da exploração comercial de sua criação. Prevê-se também bolsa paga diretamente de instituição de apoio ou agência de fomento, envolvida nas atividades de empreendidas em parceria com sua instituição e até mesmo um direito do pesquisador explorar diretamente suas criações.
>
> **Incentivo à inovação na empresa:** Pelo Art. 19, a Lei prevê a concessão, por parte da União, das ICTs e das agências de fomento, de recursos financeiros, humanos, materiais ou de infraestrutura, para atender às empresas nacionais envolvidas em atividades de pesquisa e desenvolvimento [...]. Os recursos financeiros virão como subvenção econômica, financiamento ou participação societária; no caso da subvenção, haverá contrapartida da empresa beneficiária. Pelo Art. 20, a lei introduziu uma modalidade do poder de compra do Estado como meio de incentivo à inovação, posteriormente expandida pela Lei 12.349/2010. Há também a previsão de que as agências de fomento realizarão programas com ações dirigidas especialmente à promoção da inovação nas micro e pequenas empresas. Através do Art. 28, a Lei prevê incentivos fiscais a tais empresas" (BARBOSA, 2011, p. 6 e 7).

Uma vez que a Lei de Inovação oferece diversas formas de cooperação entre universidades e empresas, como acordo de prestação de serviço (Art. 8°) e acordo de parceria (Art. 9°), o presente estudo decidiu analisar somente a possibilidade de cooperação previstas no Art. 9° da Lei de Inovação. Decidiu-se analisar este tipo de cooperação porque, ao contrário do acordo de prestação de serviços, que atribui ao contratante todos os direitos de propriedade intelectual (SILVA, 2011), a parceria prevista no Art. 9° da Lei de Inovação oferece às universidades maior capacidade de negociação no que tange aos direitos de propriedade intelectual (BARBOSA, 2011; SILVA, 2011). Além disso, esse tipo de parceria é

mais apropriada nos casos em que a inovação desejada ensejar riscos, como será analisado mais adiante.

Dessa forma, o Art. 9° da Lei de Inovação possibilita a união de esforços daqueles agentes com o objetivo de realizar pesquisa científica e tecnológica, bem como o desenvolvimento de uma tecnologia, produto ou processo (BARBOSA, 2011). Segundo Pimentel et al. (2010), no caso das ICTs públicas, esse acordo de parceria deve ser materializado na forma de convênio, porquanto os parceiros buscam alcançar o mesmo objetivo: um resultado de PD&I.

Os acordos de parceria podem ser equiparados aos contratos de consórcio ou de sociedade, nos quais a relação é cooperativa no sentido de soma de esforços e recursos em busca de um objetivo comum e convergente a ambas as partes (TEIXEIRA, 2011). Dessa forma, a participação nos resultados, nos lucros e nas perdas é característica fundamental nos acordos de parceria (TEIXEIRA, 2011).

A natureza jurídica desse convênio visando a um acordo de parceria de PD&I é mista: obrigação de dar e fazer (PIMENTEL et al., 2010). Com isso, a alocação de conhecimentos e de recursos humanos, financeiros e materiais necessários à PD&I constituem em obrigação de dar. Assim, caso um dos parceiros não cumpra com suas obrigações, será considerado inadimplente (PIMENTEL et al., 2010), uma vez que há obrigação de prestação de contas dos valores repassados, devendo ser comprovada sua regular aplicação (TEIXEIRA, 2011). Por outro lado, a obrigação de fazer no âmbito da PD&I é uma obrigação de meio e não de resultado, ou seja, caso se finalize uma PD&I sem resultado concreto e satisfatório (como resolução de incerteza científica ou inovação apta a ser protegida por direitos de propriedade intelectual), a obrigação poderá ser considerada cumprida (PIMENTEL et al., 2010), já que esse tipo de acordo pressupõe a partilha de resultados, sejam eles positivos ou negativos (TEIXEIRA, 2011).

Uma vez que o risco é parte intrínseca do acordo de parceria, é necessário que a ICT demonstre que fez todos os esforços possíveis, utilizando-se de todo seu conhecimento técnico para cumprir o plano de trabalho, com o intuito de comprovar que a falta de resultado da PD&I não foi motivada por dolo ou culpa (PIMENTEL et al., 2010). Nesse sentido, para se evitar problemas judiciais no futuro, Pimentel et al. (2010) sugerem:

> "Para que não haja a caracterização do dolo ou da culpa, para efeitos de responsabilidade por perdas e danos, é necessário que a ICT adote na vigência do acordo de parceria de PD&I ferramentas de gestão de projetos visando documentar, gerenciar e resguardar a memória histórica da PD&I, de forma a comprovar formalmente o adimplemento da obrigação. O importante é ficar clara essa

> característica da responsabilidade no acordo, para que possíveis demandas futuras não ocorram" (PIMENTEL et al., 2010, p. 38 e 39).

Além disso, recomenda-se, logo no começo das negociações para se firmar a parceria, determinar como cada parceiro irá alocar seus recursos, o problema a ser solucionado na execução do projeto e como serão tratados os eventuais resultados obtidos da execução do acordo (PIMENTEL et al., 2010). Dessa forma, é fundamental que os parceiros elaborarem um plano de trabalho de forma conjunta, contendo metas, etapas e contrapartidas (PIMENTEL et al., 2010).

Outra característica imprescindível do acordo de parceria, prevista no Art. 8° da Lei de Inovação, é a obrigatoriedade de constar, nesse instrumento jurídico, cláusulas referentes à titularidade da propriedade intelectual [20] e a participação nos resultados das criações resultantes da parceria (BARBOSA, 2011). Nesse sentido, recomenda-se também que os pesquisadores envolvidos no projeto de pesquisa assinem um termo de cessão dos direitos da titularidade da propriedade intelectual, para que esses direitos sejam transferidos para a universidade e o eventual parceiro (PIMENTEL et al., 2010).

Com relação à titularidade da propriedade intelectual sobre os resultados obtidos de acordos de parceria de PD&I, o inciso 2° do Art. 9° da Lei de Inovação estabelece que essa titularidade deve ser partilhada entre os partícipes de modo proporcional à sua contribuição no projeto (BRASIL, 2004):

> "§2º As partes deverão prever, em instrumento jurídico específico, a **titularidade da propriedade intelectual** e a **participação nos resultados da exploração das criações resultantes da parceria**, assegurando aos signatários o direito à exploração, ao licenciamento e à transferência de tecnologia [...]" (BRASIL, 2004, grifo nosso)

Dessa forma, deve-se avaliar detalhadamente a contrapartida de cada parte para o projeto, já que esta avaliação será utilizada como parâmetro para a definição do percentual de cotitularidade da propriedade intelectual da qual cada parceiro poderá ter direito, de modo que tal estimativa seja objetiva e sindicável (BARBOSA, 2011; PIMENTEL et al., 2010).

Com relação à estimativa da contrapartida da ICT no acordo de parceria de PD&I, geralmente, esta se realiza por meio de expedientes não financeiros (PIMENTEL et al., 2010). Nesse sentido, Barbosa (2011) afirma que a contrapartida da ICT é formada principalmente de capital intelectual: laboratórios, materiais de apoio e de escritório, recursos humanos, *know how*, ideias, bens imateriais protegidos por propriedade intelectual, etc.

---

[20] O conceito de titularidade de propriedade intelectual, especialmente no que tange ao software será analisado na seção **3.2.2.**

No que concerne à avaliação da contrapartida da instituição parceira (pública ou privada), recomenda-se buscar todos os documentos necessários que comprovem a efetiva disponibilidade desses recursos para o projeto de PD&I, sejam eles recursos financeiros, bens ou serviços (BARBOSA, 2011; PIMENTEL et al., 2010). Nesse sentido, Rapini, de Oliveira e Silva (2016), em estudo que analisou as modalidades de remuneração de 2.726 grupos de pesquisa do Censo 2008 do Diretório de Grupos de Pesquisa do CNPq, concluiu que as contrapartidas mais utilizadas pelos parceiros são as seguintes: transferência de recursos financeiros do parceiro para as atividades de pesquisa desenvolvidas pelo grupo (30,2%), transferência de insumos materiais para as atividades de pesquisa do grupo (18,1%) e fornecimento de bolsas para os membros do grupo pelo parceiro (12,5%).

Após avaliar as contrapartidas de cada parceiro, deve-se utilizar tal parâmetro para se buscar, de forma equilibrada, a melhor proporcionalidade da quota-parte da titularidade da propriedade intelectual e o resultado da PD&I que cada parceiro tem direito (BARBOSA, 2011). Dessa forma, a proporcionalidade prevista no Art. 9° da Lei de Inovação é sujeita a uma regra de equivalência entre o aportado e o auferido após o trabalho comum, de modo que não há imposição de que haja correlação mecânica entre valores e apropriação (BARBOSA, 2011).

Com o intuito de deslindar mais precisamente a natureza flexível da relação titularidade da propriedade intelectual/resultado da exploração da PD&I, Barbosa (2011) argumenta que:

> "O resultado do esforço comum é distinguido entre titularidade de direitos, de um lado, e resultados, de outro. A equivalência entre investimento e retorno se apurará entre esses conjuntos como um todo. Não há imposição legal, nem razoabilidade, em fazer com que 100 reais de investimento financeiro de uma empresa resultem em cotitularidade meio a meio da patente resultante e meio a meio dos royalties imputados, numa equivalência mecânica. Ao contrário, pode ser de maior interesse para a ICT manter a patente em sua titularidade, reservando a licença exclusiva por todo o prazo da patente para a parceira privada, e recebendo metade dos royalties apurados ou imputados. [...] A norma legal é de razoabilidade, como qualquer norma de proporção, e não de formalidade. Mantidos os princípios de objetividade e sindicabilidade, a eficácia do Art. 9° dependerá do ajustamento da regra da apropriação de resultados às peculiaridades do mercado e do momento" (BARBOSA, 2011, p. 96).

No que tange à relativa liberdade dos parceiros em estipular o grau de participação na titularidade da propriedade intelectual e nos resultados da PD&I, Teixeira (2011) afirma que tal proporção poderá consistir, inclusive, em participação nenhuma, para um dos partícipes, com total apropriação pelo outro, desde que assegurado o direito ao licenciamento das criações e inovações a ambos. Dessa forma, pode-se, ao máximo, obter a total apropriação da titularidade e, no mínimo, ter direito ao licenciamento para uso próprio (TEIXEIRA, 2011).

No que concerne ao licenciamento previsto no inciso 2 do Art. 9°, faz-se necessário analisar o conceito de licença, com intuito de compreender esse possível resultado da PD&I. Em termos gerais, o contrato de licença de software é um negócio jurídico pelo qual o titular de um direito de monopólio de exploração concede a uma pessoa, total ou parcialmente, o gozo de seu direito (BARBOSA, 2011b). Dessa forma, a licença pode ter várias modalidades, podendo ser simples ou exclusiva, dependendo do ajuste entre o licenciante e o licenciado (BARBOSA, 2011b).

A licença simples é a autorização de exploração de software, sem que o licenciante assuma o compromisso de não mais explorar direta ou indiretamente o bem imaterial protegido por propriedade intelectual (BARBOSA, 2011b). A licença exclusiva, que implica em renúncia do direito de exploração do licenciante para terceiros, se aproxima economicamente da venda do direito, embora juridicamente o licenciante continue como titular de direito (BARBOSA, 2011b). Não obstante os parceiros acordarem que um deles terá direito a licença exclusiva, o outro terá direito a explorar o resultado protegido da PD&I para usufruto próprio (TEIXEIRA, 2011).

Em face da grande gama de possibilidades no que tange à participação na titularidade da propriedade intelectual e nos resultados da PD&I, proporcionadas pela Lei da Inovação, pode-se verificar no **Quadro 2** uma lista não exaustiva dessas possibilidades, a qual não incluiu, por exemplo, o caso de parcerias de PD&I que envolvem fundações de amparo à pesquisa e agências de fomento.

Quadro 2 – Tipos de acordos de parceria no que tange à PI e resultados da PD&I

Tipos dos acordos de parceria	**Titularidade da PI**	**Licença**	**Relacionamento com terceiros**
Tipo de acordo de parceria 1	ICT tem direitos exclusivos sobre a PI	*Licença exclusiva* é concedida para parceiro atuar em setor específico da economia	Parceiro não tem direito de negociar seus direitos com terceiros
Tipo de acordo de parceria 2	ICT tem direitos exclusivos sobre a PI	*Licença exclusiva* é concedida para parceiro atuar em setor específico da economia	Parceiro tem direito a negociar com terceiros, ou seja, sublicenciar, com anuência da ICT
Tipo de acordo de parceria 3	ICT tem direitos exclusivos sobre a PI	Concede-se *licença não exclusiva* para parceiro atuar em setor específico da economia	Parceiro não tem direito de negociar seus direitos com terceiros

Tipo de acordo de parceria 4	ICT tem direitos exclusivos sobre a PI	Concede-se *licença não exclusiva* para parceiro atuar em setor específico da economia	Parceiro tem direito a negociar com terceiros, ou seja, sublicenciar, com anuência da ICT
Tipo de acordo de parceria 5	ICT tem direitos exclusivos sobre a PI	Concede-se *licença não exclusiva* para parceiro atuar em setor específico da economia	ICT pode negociar com terceiros, ou seja, sublicenciar, só para campo tecnológico distinto do parceiro
Tipo de acordo de parceria 6	ICT tem direitos exclusivos sobre a PI	Concede-se *licença não exclusiva* para parceiro atuar em setor específico da economia	ICT tem direito a negociar com terceiros, ou seja, sublicenciar, inclusive para mesmo campo tecnológico do parceiro
Tipo de acordo de parceria 7	ICT e parceiro têm cotitularidade da PI	*Licença exclusiva* é concedida para parceiro atuar em setor específico da economia	Parceiro não tem direito de negociar seus direitos com terceiros
Tipo de acordo de parceria 8	ICT e parceiro têm cotitularidade da PI	*Licença exclusiva* é concedida para parceiro atuar em setor específico da economia	Parceiro tem direito a negociar com terceiros, ou seja, sublicenciar, com anuência da ICT
Tipo de acordo de parceria 9	ICT e parceiro têm cotitularidade da PI	Concede-se *licença não exclusiva* para parceiro atuar em setor específico da economia	Parceiro não tem direito de negociar seus direitos com terceiros
Tipo de acordo de parceria 10	ICT e parceiro têm cotitularidade da PI	Concede-se *licença não exclusiva* para parceiro atuar em setor específico da economia	Parceiro tem direito a negociar com terceiros, ou seja, sublicenciar, com anuência da ICT
Tipo de acordo de parceria 11	ICT e parceiro têm cotitularidade da PI	Concede-se *licença não exclusiva* para parceiro atuar em setor específico da economia	ICT pode negociar com terceiros, ou seja, sublicenciar, só para campo tecnológico distinto do parceiro
Tipo de acordo de parceria 12	A ICT tem direitos sobre a PI	Concede-se *licença não exclusiva* para parceiro atuar em setor específico da economia	ICT tem direito a negociar com terceiros, ou seja, sublicenciar, inclusive para mesmo campo tecnológico do parceiro

Fonte: Baseado em Pimentel et al., (2010, p. 88, 89 e 90).

Além das possibilidades listadas acima, a empresa parceira poderá optar por proteger o resultado apenas como “segredo industrial”. Nesse caso, recomenda-se que a ICT negocie uma compensação financeira por não poder proteger tal resultado por direitos de propriedade intelectual, e nem poder licenciá-lo a terceiros (PIMENTEL et al., 2010).

Recomenda-se também que, em quaisquer das possibilidades previstas acima, a ICT e a empresa parceira firmem instrumento jurídico próprio, a partir das regras fixadas no acordo de parceria, que regule detalhadamente as condições de exploração comercial da propriedade intelectual pela empresa, inclusive para tratar da participação da ICT nos resultados da comercialização (PIMENTEL et al., 2010). Nesse sentido, sugere-se também que esses ajustes sejam feitos somente após a obtenção efetiva do resultado de PD&I, quando os parceiros terão condições de estabelecer como deverá ser realizada a exploração comercial (PIMENTEL et al., 2010).

2.5 Participação nos resultados da PD&I

Uma vez alcançado um resultado significativo no projeto de PD&I, como a proteção de uma inovação, pode ser previsto um adiantamento da participação da ICT, baseando-se em uma expectativa estimada de venda ou de prestação de serviços (PIMENTEL et al., 2010). Além disso, segundo Barbosa (2011) e Pimentel et al. (2010) pode se prever pagamento dos direitos de participação a serem realizados posteriormente com base no resultado dos negócios efetivamente realizados (royalties).

Uma vez analisadas as possibilidades oferecidas pelo Art. 9° da lei de Inovação no que tange à parceria entre universidade e empresa para desenvolver produtos inovadores, passa-se a investigar detalhadamente a questão dos direitos de propriedade intelectual, particularmente a propriedade intelectual do software. Para tanto, faz-se necessário, primeiramente, fazer breve histórico do conceito de direito de propriedade intelectual.

3. Propriedade intelectual

Em termos gerais, os direitos de propriedade intelectual implicam a existência de uma espécie de monopólio da reprodução ou emprego de certos bens intangíveis que poderão ser comercializados na forma de produtos ou serviços, já que o titular desses direitos detém o direito exclusivo de explorar essa oportunidade perante o mercado, como por exemplo, o emprego de uma nova tecnologia (BARBOSA, 2003).

A Propriedade Intelectual é um ramo do direito privado historicamente vinculado ao Direito Internacional, uma vez que suas normas fundamentais são baseadas em tratados e convenções internacionais (BASSO, 2004; BARBOSA, 2003). A moderna concepção de propriedade intelectual se consolidou, em 1967, com a criação da Organização Mundial da Propriedade Intelectual (OMPI), agência especializada no âmbito do sistema das Nações Unidas. Desde sua fundação, a OMPI teve papel fundamental no necessário processo de unificação e atualização da Convenção da União de Paris para a Proteção da Propriedade Industrial, criada em 1883, e da Convenção da União de Berna para a Proteção das Obras Literárias e Artísticas, fundada em 1886 (BASSO, 2004).

Na Convenção da OMPI, a definição de Propriedade intelectual engloba os direitos relativos às obras literárias, artísticas e científicas, às interpretações dos artistas intérpretes e às execuções dos artistas executantes, aos fonogramas e às emissões de radiodifusão, às invenções em todos os domínios da atividade humana, às descobertas científicas, aos desenhos e modelos industriais, às marcas industriais, comerciais e de serviço, bem como todos os outros direitos inerentes à atividade intelectual nos domínios industrial, científico, literário e artístico (BARBOSA, 2003).

Outro importante marco para a consolidação da proteção dos direitos de Propriedade Intelectual num mundo cada vez mais globalizado, foi a criação da Organização Mundial do Comércio (OMC), em 1994, principalmente com a assinatura de um de seus principais documentos fundadores, o chamado acordo *TRIPS* (*Trade Related Aspects of Intellectual Property Rights* – Acordo sobre Aspectos dos Direitos de Propriedade Intelectual Relacionados ao Comércio). Em termos gerais, o acordo *TRIPS* estabelece parâmetros mínimos de proteção, pelos quais nenhum país-membro pode tratar diferentemente os demais nem estabelecer desigualdade entre nacionais e estrangeiros (BARBOSA, 2003). Dessa forma, o acordo TRIPS institui princípios básicos; padrões relativos à existência, abrangência e exercício de Direitos de Propriedade Intelectual e de Direitos Conexos; bem como a adoção

de mecanismos de prevenção e solução de controvérsias entre os países-membros (BARBOSA, 2003).

Com o intuito de implementarem as regras estabelecidas pelo acordo *TRIPS,* os países-membros tiveram que adaptar e modificar suas legislações nacionais (BASSO, 2004). No que concerne à legislação brasileira, o governo federal precisou revogar o Código de Propriedade Industrial de 1971 e a Lei de Direitos Autorais de 1973. Desse modo, após aprovação do Congresso Nacional, o Presidente Fernando Henrique Cardoso sancionou a Lei nº 9.279/1996 sobre a Propriedade Industrial e a Lei nº 9.610/1998 de Direitos Autorais (BASSO, 2004).

O acordo *TRIPS,* em seu artigo décimo, estipulou outro dispositivo jurídico importante para os fins do presente estudo, a saber, "programas de computador, em código fonte ou objeto, serão protegidos como obras literárias pela Convenção de Berna" (BRASIL, 1994). Nesse sentido, o governo brasileiro foi impelido a sancionar a Lei nº 9.609/1998, na qual determina que "o regime de proteção à propriedade intelectual de programa de computador é o conferido às obras literárias pela legislação de direitos autorais e conexos vigentes no País". Além disso, a Lei nº 9.609/1998 estabelece algumas regras específicas com relação ao desenvolvimento e à venda dos programas de computador, bem como restringe os direitos morais do autor de programa de computador em relação aos previstos na Lei nº 9.610/1998 de Direitos Autorais (AREAS, 2006).

3.1 Propriedade Intelectual de programa de computador

A legislação brasileira estabelece que a proteção à propriedade intelectual de programa de computador é regulamentada pela Lei nº 9.609/1998 e, subsidiariamente, no que lhe for aplicável, pela Lei dos Direitos Autorais - Lei nº 9.610/1998 (SANTOS, 2008; BARBOSA, 2003b). Nesse sentido, a literatura considera que o regime de proteção à propriedade intelectual de programa de computador é um regime especial oriundo do regime de proteção aos Direitos Autorais de obras literárias, artísticas e científicas (SANTOS, 2008; BARBOSA; PRADO, 2011; BARBOSA, 2003b; HAMMES, 2002).

Com o intuito de se analisar adequadamente a propriedade intelectual do programa de computador, é necessário, primeiramente, diferenciar o conceito de programa de computador e o de software, uma vez que frequentemente são utilizados como sinônimos (SILVA, 2013; SANTOS, 2008; AREAS, 2006; WACHOWICZ, 2004).

O conceito de software é mais abrangente que o de programa de computador, porquanto aquele abrange não somente a linguagem codificada (código-fonte e código-objeto), mas também toda a documentação técnica associada à elaboração do software,

composta pela descrição detalhada do programa com suas especificidades técnicas e funcionalidades, bem como o material de apoio ao usuário (SANTOS, 2008; AREAS, 2006; WACHOWICZ, 2004). Dessa forma, a documentação técnica associada ao software poderá ser protegida apenas pelo regime geral de proteção aos Direitos Autorais previsto na Lei nº 9.610/1998, desde que atendidos os pré-requisitos previstos nessa legislação, como o da originalidade (SANTOS, 2008; AREAS, 2006, 2010; WACHOWICZ, 2004).

Além disso, caso se deseje identificar e distinguir o software como produto ou como serviço, pode-se proteger a marca desse software por meio do regime de proteção intelectual das marcas registradas, conforme previsto no título III (art.122 ao art. 175) da Lei de Propriedade Industrial – Lei n° 9.279/96 (AREAS, 2010; COPETTI, 2008).

Dessa forma, é recomendável que os desenvolvedores de software, como os pesquisadores que desenvolvem esse bem imaterial em processos de P&D, atentem-se para essas diferentes modalidades de proteção jurídica do software, principalmente no que concerne à comercialização do software (AREAS, 2006, 2010).

Não obstante a importância de se estudar os regimes de propriedade intelectual capazes de proteger os diferentes aspectos do software, o presente estudo vai analisar especificamente o regime de proteção à propriedade intelectual do programa de computador previsto nas Leis nº 9.609/1998 e nº 9.610/1998, porquanto esses são os regimes que incidem na linguagem codificada (código-fonte e código-objeto), objeto em que os pesquisadores da área de computação exercem maior influência no processo de desenvolvimento de software conduzidos em universidades federais. Nesse sentido, a lei nº 9.609/1998, em seu primeiro artigo, é clara no que tange ao objeto que deve ser protegido ao definir o programa de computador como:

> A expressão de um conjunto organizado de instruções em linguagem natural ou codificada, contida em suporte físico de qualquer natureza, de emprego necessário em máquinas automáticas de tratamento da informação, dispositivos, instrumentos ou equipamentos periféricos, baseados em técnica digital ou análoga, para fazê-los funcionar de modo e para fins determinados (BRASIL, 1998a).

Dessa forma, a proteção da propriedade intelectual do programa de computador incide somente sobre sua forma de expressão – a linguagem natural (código-fonte[21]) e a codificada (código-objeto[22]).

[21] O código fonte é um conjunto ordenado de instruções, formado por afirmações lógicas, matemáticas e simbólicas, que são facilmente compreendidas e manipuladas, por meio de editores de texto especiais, pelos programadores e especialistas da área de tecnologia da informação (SANTOS, 2008; AREAS, 2006; BARBOSA, 2003b). O código-fonte pode ser codificado através de várias linguagens de programação (AREAS, 2010). Entre as linguagens de programação mais utilizadas destacam-se a Java, C, C++ e Python (TIOBE, 2017).

Uma vez criado o programa de computador, seu criador adquire os direitos de propriedade intelectual referentes à obra, independentemente de registro. Contudo, embora a Lei nº 9.609/1998 (Art. 2°, § 3º) estabeleça que o registro do programa de computador junto ao Instituto Nacional de Propriedade Industrial seja facultativo, o registro oferece segurança jurídica ao titular desses direitos (ZIBETTI, 2008). Além disso, apenas o programa de computador registrado poderá ser comercializado, uma vez que somente por meio do registro emanam-se efeitos jurídicos a terceiros (ZIBETTI, 2008).

3.2 Autoria e titularidade do programa de computador

Para que se possa aprofundar o estudo sobre a propriedade intelectual do programa de computador, é necessário compreender a diferença entre os conceitos de autoria e titularidade no âmbito dos Diretos Autorais, uma vez que a compreensão desses conceitos é fundamental para se analisar adequadamente o caráter dual dos Direitos Autorais, que é formado pelos direitos morais e patrimoniais (ZIBETTI, 2008; AREAS, 2006; LIPSZYC, 2005; BARBOSA, 2003a, 2003b; HAMMES, 2002).

3.2.1 Autoria e direitos morais

O conceito de autor de programa de computador refere-se somente à pessoa física que participou do processo de criação desse bem imaterial (ZIBETTI, 2008; AREAS, 2006; BARBOSA, 2003a). Embora o conceito de autor de programa de computador não esteja expressamente definido na Lei nº 9.609/1998, pode-se tomar emprestado o conceito de autor disposto na Lei de Direitos Autorais (lei nº 9.610/1998), a qual, em seu artigo décimo primeiro, define o autor como a "pessoa física criadora de obra literária, artística ou científica" (ZIBETTI, 2008; AREAS, 2010; BRASIL, 1998b).

Dessa forma, é importante analisar em cada caso qual foi a contribuição intelectual empregada pelos criadores, já que geralmente o programa de computador é resultado de uma obra coletiva (ZIBETTI, 2008).Cabe mencionar que a noção de autoria não pode ser aplicada a pessoas jurídicas, seja de direito público ou privado, uma vez que estas somente podem adquirir direitos patrimoniais, conforme será analisado posteriormente.

A literatura afirma que conceito de autoria é o fundamento jurídico dos direitos morais no âmbito dos Direitos Autorais (SANTOS, 2008; LIPSZYC, 2005; BARBOSA, 2003).

[22] O código-objeto é uma linguagem binária, formada por *bits* e representada pelos números 0 e 1 (SANTOS, 2008; AREAS, 2006). Devido à grande dificuldade de compreensão dessa linguagem pelos seres humanos, programas especiais, chamados de compiladores, traduzem o código-objeto para uma linguagem mais acessível à inteligência humana: o código-fonte (SANTOS, 2008; AREAS, 2010).

Devido ao seu caráter personalíssimo (SANTOS, 2008; AREAS, 2006; BITTAR, 2005; BARBOSA, 2003a), os direitos morais do autor são inalienáveis e irrecusáveis (Art. 27 da Lei nº 9.610/1998). No que tange aos direitos morais do autor do regime geral de Direitos Autorais, regidos pela Lei nº 9.610/1998, o rol desses direitos é bastante amplo:

> Art. 24. São direitos morais do autor:
> I - o de reivindicar, a qualquer tempo, a autoria da obra;
> II - o de ter seu nome, pseudônimo ou sinal convencional indicado ou anunciado, como sendo o do autor, na utilização de sua obra;
> III - o de conservar a obra inédita;
> IV - o de assegurar a integridade da obra, opondo-se a quaisquer modificações ou à prática de atos que, de qualquer forma, possam prejudicá-la ou atingi-lo, como autor, em sua reputação ou honra;
> V - o de modificar a obra, antes ou depois de utilizada;
> VI - o de retirar de circulação a obra ou de suspender qualquer forma de utilização já autorizada, quando a circulação ou utilização implicarem afronta à sua reputação e imagem;
> VII - o de ter acesso a exemplar único e raro da obra, quando se encontre legitimamente em poder de outrem, para o fim de, por meio de processo fotográfico ou assemelhado, ou audiovisual, preservar sua memória, de forma que cause o menor inconveniente possível a seu detentor, que, em todo caso, será indenizado de qualquer dano ou prejuízo que lhe seja causado (BRASIL, 1998b).

Contudo, no que concerne ao regime especial de proteção ao autor de programa de computador, regulado pela Lei nº 9.609/1998, os direitos morais do autor são significativamente restringidos para apenas dois (Art. 2°, § 1°), a saber: a) o direito do autor de reivindicar a paternidade do programa de computador a qualquer tempo; e b) o direito do autor de opor-se a alterações não-autorizadas, quando estas impliquem deformação, mutilação ou outra modificação do programa de computador, que prejudiquem a sua honra ou a sua reputação.

Dessa forma, o rol reduzido dos direitos morais do autor de programa de computador constitui-se uma das principais diferenças entre o regime de proteção à propriedade intelectual de programa de computador e o regime geral de proteção aos Direitos Autorais (SANTOS, 2008; AREAS, 2006). Nesse sentido, o limitado rol de direitos morais é objeto de críticas por parte da literatura, principalmente porque evidencia que o regime de Direitos Autorais não se adéqua eficazmente às particularidades do programa de computador (SANTOS, 2008; PAESANI, 2001; ASCENÇÃO, 1997).

Com relação ao direito moral do autor de reivindicar a paternidade do programa de computador a qualquer tempo, embora esse direito tenha pouco efeito prático (HIDALGO; CUESTA, 2004), Cerqueira (2000) e Areas (2006) afirmam que a reivindicação da paternidade pode ser benéfica ao promover o trabalho do autor de programa de computador frente a um mercado muito competitivo. Além disso, excepcionalmente, esse direito moral

pode favorecer o autor de programa de computador na hipótese de indenização por danos morais ou outros prejuízos quando não houver menção à paternidade do autor (AREAS, 2006).

Com relação ao segundo direito moral previsto na Lei nº 9.609/1998 (Art. 2°, § 1°), o direito do autor de opor-se a alterações não-autorizadas, parte da literatura enfatiza não só a difícil aplicação desse direito, mas também sua inconveniência para a indústria e comércio de software (VIEIRA, 2005; HIDALGO; CUESTA, 2004; CERQUEIRA, 2000).

Apesar da difícil aplicabilidade desse direito moral, Areas (2006) argumenta que:

> "[...] isso não exclui sua validade e a possibilidade de ser reivindicado. Tampouco o fato de existir a complexidade acima referenciada, não faz com que este direito deixe de ser aplicável ao software, o que reforça a necessidade de maior observação ao se confeccionar um contrato no que tange às modificações, alterações, correções, atualizações, etc., não só quanto à titularidade, mas também quanto à autorização para proceder às mesmas" (AREAS, 2006, p. 205).

Dessa forma, Areas (2006), em sua dissertação de mestrado, conclui que os direitos morais do autor de programa de computador são fatores relevantes ao ponto de poderem limitar a autonomia da vontade de agentes econômicos em contratos internacionais de software. Não obstante, a autora reconhece que há divergência da literatura no que se refere à aplicabilidade desses direitos (AREAS, 2006).

3.2.2 Titularidade e direitos patrimoniais

A titularidade de direitos de autor refere-se ao direito exclusivo de exploração econômica de uma obra imaterial, de forma que terceiros apenas poderão utilizar essas obras mediante autorização do titular (BARBOSA, 2003b; AREAS, 2006;LIPSZYC, 2005; ZIBETTI, 2008). Dessa forma, a titularidade de um bem imaterial diz respeito somente aos direitos patrimoniais, ou seja, aqueles advindos da exploração econômica desse bem, excluindo-se, assim, os direitos morais do autor (BARBOSA, 2003b; AREAS, 2006;LIPSZYC, 2005; ZIBETTI, 2008).

Tanto pessoas físicas quanto pessoas jurídicas podem assumir a qualidade de titular de direitos patrimoniais de uma obra imaterial como o programa de computador (AREAS, 2006; ZIBETTI, 2008). De modo geral, a legislação de Direitos Autorais determina que a titularidade de direitos patrimoniais de uma obra recai na figura do autor, quando a criação dessa obra é resultado de sua própria iniciativa, a chamada autoria singular (ZIBETTI, 2008, 2014).

O regime geral de proteção aos Direitos Autorais também prevê a coautoria (ZIBETTI, 2008;ZIBETTI; ZIEGLER FILHO, 2014). Dessa forma, a coautoria ocorre quando um programa de computador é criado em comum por dois ou mais autores. A coautoria é, assim, caracterizada por uma relação horizontal (recíproca) de colaboração (ZIBETTI, 2008;ZIBETTI; ZIEGLER FILHO, 2014). Com relação à titularidade dos direitos patrimoniais, concede-se, de forma igual, a todos os coautores, salvo convenção em contrário(ZIBETTI, 2008;ZIBETTI; ZIEGLER FILHO, 2014).

Não obstante, no caso específico do programa de computador, a legislação também estabelece que a titularidade de direitos patrimoniais possa ser atribuída a pessoas distintas do autor, como no caso de derivação de programa de computador e criações resultantes de relação laboral (ZIBETTI, 2008;ZIBETTI; ZIEGLER FILHO, 2014).

3.2.2.1 A titularidade de derivação de programa de computador

O programa de computador pode ser considerado uma obra derivada quando resultar da transformação de um programa originário (ZIBETTI, 2008). Dessa forma, desde que haja uma autorização do titular do programa de computador originário, pode-se criar uma obra derivada (ZIBETTI, 2008). Nesse sentido, Zibetti e Ziegler Filho (2014) argumentam que:

> "O autor do programa originário não se confunde com o autor das derivações do programa. Será autor de programa de computador derivado aquele que realiza a derivação, resguardando-se a autoria do criador do programa originário. Se o programa originário estiver em domínio público, o titular de programa derivado será, em princípio, o autor do novo programa. Se o programa originário estiver protegido, os direitos sobre as derivações autorizadas pelo titular dos direitos de programa de computador, inclusive sua exploração econômica, pertencerão à pessoa autorizada que as fizer, salvo estipulação contratual em contrário" (ZIBETTI; ZIEGLER FILHO, 2014, p. 305).

Dessa forma, cabe mencionar que as pessoas qualificadas como autores dos programas originário e derivado não se caracterizam como coautores (ZIBETTI; ZIEGLER FILHO, 2014). Com isso, somente poderão concorrer como coautores no caso em que ambas participarem da consecução do novo programa em comum (ZIBETTI; ZIEGLER FILHO, 2014).

3.2.2.2 A titularidade do programa de computador criado por relação laboral

Zibetti e Ziegler Filho (2014), baseados na literatura que trata das patentes de propriedade industrial, argumentam que as criações de programa de computador resultantes de relações laborais podem ser classificadas como "criação de serviço", "criação livre" ou "criação comum". No **Quadro 3** é possível verificar os tipos de criação de programa de computador e suas respectivas titularidades conforme estipulado pela legislação brasileira.

Quadro 3 – Tipos de criação ocorrida durante contrato de trabalho ou de prestação de serviço

Tipo de obra/hipóteses	Tipo de criação/Titularidade
1) Resultar de contrato que tenha por objeto a P&D	Criação de serviço Titular: Contratante Artigo 4º, Lei n. 9.609/1998
2) Resultar da natureza dos serviços para os quais a pessoa foi contratada	Criação de serviço Titular: Contratante Artigo 4º, Lei n. 9.609/1998
3) Estar desvinculada do contrato de trabalho e não decorrer da utilização de recursos da contratante	Criação livre Titular: contratado Artigo 4º, § 2º, Lei n. 9.609/1998
4) Resultar da contribuição pessoal da contratada e de recursos* da contratante	Criação comum Cotitulares: contratante e contratado Art. 4º, § 2º, Lei n. 9.610/1998

Fonte: ZIBETTI; ZIEGLER FILHO, 2014, p. 318.

Com relação às "criações livres", os autores defendem que "são aquelas criações ocorridas durante o contrato de trabalho, mas que não originam-se diretamente do contrato, nem do uso de qualquer tipo de recursos, informações tecnológicas, ou infraestrutura do empregador (ZIBETTI; ZIEGLER FILHO, 2014). Dessa forma, a titularidade dos direitos patrimoniais recai exclusivamente na figura empregado, seja ele prestador de serviço, trabalhador autônomo, ao estagiário ou servidor público (ZIBETTI; ZIEGLER FILHO, 2014). No que tange especificamente ao servidor público, os autores asseveram que:

> [...] ainda que ele tenha colaborado na elaboração de programa de computador de interesse da Administração Pública, mas sem ter sido contratado para isso, a titularidade dos direitos de propriedade intelectual sobre o programa não se desloca para o empregador público. [...] Portanto, se o funcionário público não possui entre suas funções a de desenvolvimento de software ou atividade correlata, caso ele se envolva no processo de criação e elaboração de um programa de computador para a Administração Pública informalmente, sem contrato prévio, a titularidade dos direitos sobre o programa poderá ser atribuída ao funcionário (ZIBETTI; ZIEGLER FILHO, 2014, p. 312, 313).

No que se refere à "criação comum", os autores afirmam que são aquelas decorrentes da contribuição pessoal do empregado juntamente com os recursos, informações tecnológicas, ou infraestrutura do empregador, salvo ajuste em contrário (BARBOSA; PRADO, 2011; ZIBETTI; ZIEGLER FILHO, 2014). Desse modo, os direitos patrimoniais serão repartidos entre ambas às partes de forma igual, configurando-se uma situação de cotitularidade (ZIBETTI; ZIEGLER FILHO, 2014).

Com relação às “criações de serviço”, o regime especial de proteção à propriedade intelectual de programa de computador, regulado pela Lei n° 9609/1998, prevê regras específicas com relação à titularidade dos direitos patrimoniais nos casos de contrato de trabalho, de prestação de serviços ou vínculo estatutário (BARBOSA, PRADO, 2011; AREAS, 2010; ZIBETTI, 2008;ZIBETTI; ZIEGLER FILHO, 2014).

Em seu artigo quarto, a Lei n° 9609/1998 estabelece que a titularidade dos direitos patrimoniais sobre a criação de programa de computador é atribuída unicamente ao empregador (empresa privada ou órgão público) quando “a criação decorrer de contrato de trabalho, vínculo estatutário ou prestação de serviços que tenha por objeto a pesquisa e desenvolvimento ou a atividade inventiva; ou quando ela resultar da natureza dos serviços para os quais foi o empregado contratado” (ZIBETTI, 2008, p. 89). Nesse sentido, é pertinente ressaltar que a legislação (§ 3°, Art. 4° da Lei n° 9609/1998) também aplica essa regra aos programas de computador desenvolvidos por bolsistas, estagiários e assemelhados (BARBOSA, PRADO, 2011; ZIBETTI, 2008).

No que concerne à retribuição do empregado pela criação resultante de relação laboral, a Lei n° 9609/1998 determina que essa compensação limita-se à remuneração ou ao salário convencionado, salvo ajuste em contrário (ZIBETTI, 2008;ZIBETTI;ZIEGLER FILHO, 2014).

Contudo, no que se refere às criações desenvolvidas por pesquisadores vinculados a ICT’s (Instituições Científicas e Tecnológicas) da Administração Pública, como as universidades federais, há regras específicas que regulam essa matéria (BARBOSA, 2011; ZIBETTI, 2008; TEDESCHI, 2011). Dessa forma, a Lei de Inovação (Lei nº 10.973/2004), em seu artigo décimo terceiro, estabelece que:

> “É assegurada ao criador participação mínima de 5% (cinco por cento) e máxima de 1/3 (um terço) nos ganhos econômicos, auferidos pela ICT, resultantes de contratos de transferência de tecnologia e de licenciamento para outorga de direito de uso ou de exploração de criação protegida da qual tenha sido o inventor, obtentor ou autor [...] (Art. 13, BRASIL, 2004)”.

Segundo Barbosa (2011), esse artigo da Lei de inovação implementa o regime especial de pessoal referente ao pesquisador-criador previsto na Constituição federal, cujo Art. 218 prevê o estimulo à inovação por meio de pagamento de parcela dos ganhos obtidos com a criação ao seu autor.

3.2.2.2 Administração Pública: Caso especial de titularidade de PI

Deve-se atentar a uma regra especial relativa à titularidade de propriedade intelectual de programa de computador, quando a Administração Pública (Administração direta e indireta da União, estados, Distrito Federal e municípios) formar parceria com a ICT em projeto de PD&I (PIMENTEL et al., 2010; AREAS, 2010), devido ao disposto no Art. 111 da Lei de Licitações e Contratos (Lei n° 8.666/1993):

> Art. 111. A Administração **só poderá contratar, pagar,** premiar ou receber **projeto ou serviço técnico especializado desde que o autor ceda os direitos patrimoniais a ele relativos** e a Administração possa utilizá-lo de acordo com o previsto no regulamento de concurso ou no ajuste para sua elaboração.
> Parágrafo único. **Quando o projeto referir-se a obra imaterial de caráter tecnológico,** insuscetível de privilégio, **a cessão dos direitos incluirá o fornecimento de todos os dados, documentos e elementos de informação pertinentes à tecnologia de concepção, desenvolvimento,** fixação em suporte físico de qualquer natureza e aplicação da obra (BRASIL, 1993, grifo nosso).

Segundo o Tribunal de Contas da União, no Acórdão 5684/2013, essa regra "somente se aplica aos programas de computador feitos sob encomenda para atender necessidades específicas do usuário", com o intuito de buscar a independência tecnológica da Administração pública frente ao fornecedor (BRASIL, 2013). Dessa forma, a ICT deve observar cuidadosamente esta regra quando pretender firmar acordo de parceria com órgão ou entidade da Administração Pública, de modo que busque compensar sua cessão de direitos patrimoniais.

3.3 Modelos de licenciamento de software: software proprietário e software livre

Antes de distribuir o programa de computador para terceiros, o titular dos direitos patrimoniais deste software deve optar por dois modelos de distribuição e licenciamento: o modelo de software proprietário ou o de software livre (PIMENTEL et al, 2014). Ao optar pelo primeiro modelo, o detentor dos direitos patrimoniais do programa de computador decide por manter seu código fonte "fechado", ou seja, somente o titular terá acesso a este código (PIMENTEL et al, 2014). Dessa forma, em geral, o licenciamento de software proprietário somente permitirá ao licenciado a possibilidade de executar esse programa de computador (VÄLIMÄKI, 2005).

Por outro lado, caso o titular dos direitos patrimoniais do programa de computador pretenda optar pelo modelo de distribuição e licenciamento de software livre e de código aberto, deverá disponibilizar o acesso ao seu código fonte para terceiros (PIMENTEL et al, 2014). Na seção **3.3.1** será analisado com mais profundidade o conceito e o modelo de software livre e de código aberto.

3.3.1 Software livre e de código aberto

O modelo de licenciamento de software livre e de código aberto, mais conhecido pela literatura especializada de língua inglesa como *FOSS (Free and Open Source Software*) [WU et al., 2017; HADDAD, 2016; KEMP, 2009], visa a compartilhar livremente o código fonte entre diversos desenvolvedores de software, com o intuito de melhorar a qualidade do software (KON et al., 2011). Dessa forma, à medida que uma crescente quantidade de desenvolvedores com diferentes habilidades contribuam para um projeto de software *open source,* maior será a capacidade de identificar e corrigir *bugs* (defeitos e falhas do sistema de software) em menos tempo, o que otimiza a produtividade e qualidade do processo de desenvolvimento de software (KON et al., 2011).

Para promover a liberdade de copiar, redistribuir e modificar o software livre, Richard Stallman, um dos pioneiros do movimento de software livre, criou a *Free Software Foundation,* em 1990 (KON et al., 2011; SABINO, 2011). Um dos principais objetivos dessa organização é garantir a integridade das chamadas quatro liberdades fundamentais do software livre (FALCÃO, 2005):

> Liberdade 0: A liberdade de executar o programa, para qualquer propósito;
> Liberdade 1: A liberdade de estudar como o programa funciona, e de adaptá-lo às suas necessidades. O acesso ao código fonte é uma condição prévia para o exercício dessa liberdade;
> Liberdade 2: A liberdade de redistribuir cópias, de modo que você possa auxiliar outras pessoas; e
> Liberdade 3: A liberdade de aperfeiçoar o programa e distribuir esses aperfeiçoamentos para o público, de modo a beneficiar toda a comunidade. O acesso ao código fonte é também uma condição prévia para o exercício dessa liberdade (FALCÃO, 2005, p. 7).

Com o intuito de garantir o cumprimento dessas liberdades básicas do software livre, a *Free Software Foundation* concebeu um instrumento jurídico, chamado de GPL[23] (*General Public License* - Licença Pública Geral), que se trata de uma licença que concede uma série de direitos aos licenciados, desde que estes cumpram com as obrigações estabelecidas pela licença GPL (SABINO, 2011; KON et al., 2011).

Além do chamado software livre, outra importante organização do movimento de software livre e de código aberto, a *Open Source Initiative* (OSI), criada em 1998, estabeleceu uma série de parâmetros para definir um software de código aberto (KON, 2011):

> Livre Redistribuição: Sua licença não pode restringir ninguém, proibindo que se venda ou doe o software a terceiros. A licença não pode exigir que se cobre o pagamento de royalties ou outros valores, embora tal pagamento não seja proibido.

[23] As obrigações da licença GPL serão analisadas com maior detalhe na seção **3.3.3.1.**

> Código-Fonte: O programa precisa obrigatoriamente incluir código-fonte e permitir a distribuição tanto do código-fonte quanto do programa já compilado. Quando o produto não é distribuído junto com seu código-fonte, é
> necessário que uma forma de se obter o seu código-fonte seja anunciada publicamente de uma forma fácil de se obter, preferivemente através de download gratuito da Internet.
>
> Obras Derivadas: A licença deve permitir modificações e obras derivadas e deve permitir que essas modificações sejam redistribuídas dentro dos mesmos termos da licença original.
>
> Integridade do Código do Autor: A licença pode proibir que se distribua o código fonte original modificado desde que, neste caso, a licença permita a distribuição de arquivos de diferenças (patch files) contendo o código fonte que foi modificado. A licença deve então explicitamente permitir a distribuição do software construído através das modificações do código fonte original. A licença pode exigir que esses trabalhos derivados usem um nome ou número de versão diferente do software original.
>
> Não Discriminação Contra Pessoas ou Grupos:
> A licença não pode discriminar contra pessoas ou grupos. Por exemplo, a licença não pode proibir que um programa seja distribuído para pessoas residentes em um determinado país.
>
> Não Discriminação Contra Áreas de Utilização: A licença não pode restringir
> os usuários de fazer uso do programa numa área específica. Por exemplo, não pode proibir que o programa seja usado para fins comerciais, ou para fins militares, por mais nobre que esta última possa ser.
>
> Distribuição da Licença: Os direitos associados ao programa através da licença são automaticamente repassados a todas as pessoas às quais o programa é redistribuído sem a necessidade de definição ou aceitação de uma nova licença.
>
> Licença Não Pode Ser Específica a um Produto: Os direitos associados a um programa não dependem de qual distribuição em particular aquele programa está inserido. Se o programa é retirado de uma distribuição, os direitos garantidos por sua licença continuam valendo.
>
> Licenças Não Podem Restringir Outro Software: A licença não pode colocar
> restrições em relação a outros programas que sejam distribuídos junto com o software em questão. Por exemplo, a licença não pode exigir que todos os outros programas distribuídos no mesmo pacote sejam também software aberto.
>
> Licenças Devem Ser Neutras em Relação a Tecnologias: Nenhuma exigência
> da licença pode ser específica a uma determinada tecnologia ou estilo de interface (SABINO, 2011, p. 9 - 10).

Dessa forma, verifica-se que, enquanto a definição de software livre da *Free Software Foundation* enfatiza a questão da liberdade do usuário, a definição de Software Aberto da *Open Source Initiative* prioriza a utilização deste tipo de software no mundo corporativo (SABINO, 2011). Contudo, em termos práticos, não há diferença entre software livre e de código aberto, uma vez que conjunto de licenças[24] aprovadas pela *Free Software Foundation* e pela *Open Source Initiative* é quase idêntico (SABINO, 2011).

[24] As principais licenças de software livre e de código aberto serão analisadas nas seções **3.3.2, 3.3.3** e **3.3.4**

Uma vez analisados os conceitos de software livre e de código aberto, passa-se a examinar a questão dos direitos autorais deste tipo de software. O detentor da titularidade dos direitos patrimoniais de um software livre e de código aberto concede uma série de direitos e condições para que outras pessoas possam utilizar seu trabalho (SABINO, 2011). O documento que formaliza essa concessão de direitos é a licença, que normalmente é distribuída juntamente com o código fonte (SABINO, 2011).

Segundo Falcão (2005), o licenciamento de software por meio de licenças de software livre e de código aberto se enquadra no artigo 49 da Lei de Direitos Autorais, que regula a transferência dos Direitos de Autor. Dessa forma, as licenças de software livre e de código aberto concedem uma série de direitos de uso, modificação, distribuição e redistribuição do software licenciado (SABINO, 2011; FALCÃO et al., 2005;). Contudo, o licenciado só poderá usufruir esses direitos, caso cumpra com as obrigações contidas no texto desta licença (SABINO, 2011; FALCÃO et al., 2005).

Com relação às obrigações das licenças de software livre e de código aberto, em tese, pode-se redigir o texto dos termos de uso de uma licença de várias formas, porém é mais recomendável escolher alguma das licenças mais utilizadas pelas comunidades *open source* que mais se adéqua aos objetivos do autor (SABINO, 2011).

Dessa forma, essas licenças são classificadas em três categorias (permissivas, recíprocas parciais e recíprocas totais) de acordo com a presença de termos que impõem restrições de licenciamento na redistribuição do trabalho ou criação de trabalhos derivados (SABINO, 2011).

3.3.2 Licenças permissivas

As licenças permissivas impõem poucas restrições aos usuários que obtêm o código fonte. Dessa forma, em termos gerais, esse tipo de licenças não restringe a forma como os trabalhos derivados da licença original serão distribuídos, tanto que é possível redistribuí-los sob uma licença proprietária, ou seja, com o código fechado. (SABINO, 2011).

As licenças permissivas são adequadas para projetos que almejam alcançar o maior número possível de potenciais reutilizadores, sejam eles projetos de software fechados ou livre (SABINO, 2011). Dessa forma, tem aumentado o número de empresas de desenvolvimento de software reutilizam as licenças permissivas (SABINO, 2011; LINDMAN; PAAJANEN; ROSSI, 2010).

Analisar-se-á abaixo as três principais licenças permissivas em projetos de software *open source*: BSD, MIT e Apache.

3.3.2.1 Licença BSD

A licença BSD (*Berkeley Software Distribution*), criada em 1988, é considerada a primeira licença de software livre e de código aberto da história (SABINO, 2011). Essa licença é uma das mais utilizadas pelas comunidades *open source* principalmente devido à sua simplicidade (SABINO, 2011).

A licença BSD possui três versões que apresentam uma pequena variação entre si: a licença BSD original, a licença BSD de três cláusulas (BSD-3 modificada) e a licença BSD de duas cláusulas (BSD-2 simplificada) [SABINO, 2011]. A licença BSD original contém uma cláusula que ficou conhecida como a "cláusula de propaganda da licença BSD", a qual determina que todo material de divulgação relacionado ao software que reutiliza o componente de software licenciado sob a BSD original precisa o seguinte aviso: "este produto inclui software desenvolvido pela Universidade da California, Berkeley e seus contribuidores[25]" (SABINO, 2011).

Contudo, como esta cláusula foi bastante criticada por causar inconvenientes para os desenvolvedores de software, foram criadas as versões BSD-3 e BSD-2, que passaram a ser as mais utilizadas pelas comunidades *open source* (SABINO, 2011).

A única diferença entre essas versões da licença BSD é que a BSD-2 não contém uma cláusula existente na BSD-3, a qual veda utilizar o nome do titular do componente de software licenciante e seus contribuidores para endossar produtos derivados sem sua permissão prévia por escrito[26] (SABINO, 2011).

Com relação às outras obrigações das versões da licença BSD, ao escolher a licença BSD, o autor licenciante permite que outras pessoas usem, modifiquem e distribuam o software (SABINO, 2011). Dessa forma, caso se deseje reutilizar o código fonte ou binário modificado ou não, licenciado sob a BSD, deve-se apenas a mencionar o aviso de *copyright* original e os termos da licença, conforme o **Quadro 4** (SABINO, 2011).

---

[25] No texto original da licença BSD original em inglês este aviso está escrito da seguinte forma: "*All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the University of California, Berkeley and its contributors*".

[26] No texto original da licença BSD-3 em inglês esta cláusula está escrita assim: "3. *Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission*".

Além disso, cabe destacar que, ao reutilizar o componente de software *open source* licenciado sob a licença BSD, o desenvolvedor de software não está obrigado a licenciar o software derivado sob a mesma licença BSD, podendo inclusive licenciá-lo sob uma licença proprietária de código fechado (SABINO, 2011).

Quadro 4– Texto de aviso de direitos autorais da licença BSD-3

```
Copyright <YEAR><COPYRIGHT HOLDER>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

* Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.
* Neither the name of the <organization> nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

Fonte: https://opensource.org/licenses/BSD-3-Clause

Não obstante a simplicidade dos termos da licença BSD, uma de suas desvantagens, do ponto de vista jurídico, é que seus termos são vagos, havendo maior esforço de interpretação da licença para demonstrar que, de fato, a maioria dos direitos autorais do titular estão sendo transferidos aos licenciados (SABINO, 2011).

Dessa forma, a licença BSD é uma boa escolha para os projetos de software *open source* em que não há uma preocupação a respeito de como outras pessoas poderão usar ou modificar o componente de software licenciado sob a BSD (SABINO, 2011). De igual forma, a escolha de componentes de software licenciados sob a licença BSD é muito vantajosa para desenvolvedores e organizações que desejam reutilizá-los para fins proprietários e comerciais.

3.3.2.2 Licença MIT

A licença MIT foi criada por pesquisadores do *Massachusetts Institute of Technology* no final dos anos 1980 (SABINO, 2011). Ao contrário da licença BSD, o texto da licença MIT é mais explícito ao tratar dos direitos que estão sendo transferidos do licenciante ao licenciado (SABINO, 2011). Dessa forma, o software licenciado sob a licença MIT permite que qualquer pessoa que obtém uma cópia desse software possa utilizá-lo sem restrição, bem como seus arquivos de documentação associados (SABINO, 2011).

A licença MIT concede explicitamente, ao licenciado, o direito sem restrição para usar, copiar, modificar, mesclar, publicar, distribuir, sublicenciar e/ou vender cópias do software (SABINO, 2011). Cabe destacar que esse rol de direitos é mais amplo que os concedidos pela licença BSD (SABINO, 2011).

As únicas obrigações impostas pela licença MIT são a de manter o aviso de *copyright* do autor original do software licenciante e a de disponibilizar uma cópia da licença em todas as cópias ou porções substanciais do software (SABINO, 2011).

Dessa forma, a simplicidade do texto e o maior número de direitos explicam a crescente utilização da licença MIT pelas comunidades *open source*, de modo que essa é a licença mais utilizada eplas comunidades *open source* (BLACK DUKE, 2017).

O **Quadro 5** mostra o aviso de direitos autorais que deve acompanhar todo trabalho que reutiliza algum componente de software licenciado sob a licença MIT.

Quadro 5– Texto de aviso de direitos autorais da licença MIT

```
Copyright © [ano] <nome do detentor dos direitos autorais>

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation files
(the "Software"), to deal in the Software without restriction,
including without limitation the rights to use, copy, modify, merge,
publish, distribute, sublicense, and/or sell copies of the Software,
and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

Fonte: https://opensource.org/licenses/MIT

3.3.2.3 Licença Apache

Dentre as licenças permissivas, a licença Apache, versão 2.01, criada em 2004, é a que apresenta o texto mais complexo e preciso no que se refere à concessão de direitos entre o autor licenciante e o licenciado (aquele que vai reutilizar o componente de software) [SABINO, 2011; SINCLAIR, 2010].

Com o intuito de facilitar sua interpretação, a primeira parte do texto da licença Apache contém as definições das palavras-chave que são utilizadas ao longo do texto (SABINO, 2011). Dentre essas definições, cabe destacar as de "fonte" e "Trabalho derivado". A definição de fonte da licença Apache é mais abrangente comparada com outras licenças porque engloba não somente o código fonte, mas também a fonte da documentação e os arquivos de configuração (SABINO, 2011). Com relação à definição de trabalho derivado, a licença Apache considera que qualquer tipo trabalho, seja em código fonte ou objeto, baseado em outro trabalho, cujas modificações resultem em um trabalho original (WEBBINK, 2010). Não obstante, a licença Apache não considera como trabalho derivado aqueles que se mantêm separados do trabalho original ou que meramente são ligados a sua interface (SABINO, 2011).

Outra peculiaridade da licença Apache é existência de uma cláusula[27] que concede, expressamente, ao licenciado, o direito irrevogável de reproduzir, preparar trabalhos derivados, mostrar publicamente, sublicenciar e distribuir o trabalho e seus derivados, na forma de código fonte ou objeto, sujeitos aos termos e condições da licença (SABINO, 2011). Dessa forma, essa cláusula permite que se possa reutilizar software licenciado sob a licença Apache em outros tipos de licença, inclusive proprietárias de código fechado (SABINO, 2011).

Para reutilizar artefatos de software com a licença Apache, segundo Sabino (2011), deve-se seguir as seguintes obrigações:

> - Incluir uma cópia da licença Apache;
> - Incluir avisos em todos os arquivos que foram modificados informando sobre a alteração;
> - Manter na fonte trabalhos derivados todos os avisos de direitos autorais, patentes e marcas que são pertinentes;
> - Se o trabalho incluir um arquivo texto chamado "NOTICE", então qualquer trabalho derivado distribuído deve incluir os arquivos pertinentes contidos nesse arquivo da forma como está detalhado na licença (SABINO, 2010, p. 31)".

[27] No texto original em inglês da licença Apache, essa cláusula está disposta da seguinte maneira: "*2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form*".

No **Quadro 6**, pode-se visualizar o aviso de direitos autorais que deve acompanhar qualquer software que reutilize componentes de software licenciados sob a licença Apache.

Quadro 6– Texto de aviso de direitos autorais da licença Apache

```
Copyright [ano] [nome do detentor dos direitos autorais]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied. See the License for the specific language governing
permissions and limitations under the License.
```

Fonte: https://opensource.org/licenses/Apache-2.0

3.3.3 Licenças recíprocas totais

As licenças recíprocas totais determinam que qualquer trabalho derivado precisa ser distribuído sob os mesmos termos da licença original (SABINO, 2011). Dessa forma, quando se reutiliza qualquer componente de software licenciado sob uma licença recíproca total, deve-se necessariamente licenciar o software derivado sob a mesma licença recíproca total (SABINO, 2011).

Este tipo de obrigação é um dos pilares do conceito de *copyleft,* cunhado pela *Free Software Foundation* com o intuito deliberado de se sobrepor ao conceito de *copyright*, que geralmente é relacionado ao software proprietário (DE LAAT, 2005). Com isso, ao permitir a livre execução, cópia e modificação dos componentes de software, o conceito de *copyleft* das licenças recíprocas totais tem como objetivo disponibilizar as melhorias realizadas no software para toda a comunidade (SABINO, 2011).

Além disso, cabe mencionar que, na maioria dos casos, é possível incorporar componentes de software licenciados sob licença permissiva em projetos de software licenciados com licença recíproca total (SABINO, 2011). Contudo, não se pode realizar o contrário, ou seja, incorporar software licenciado sob licença recíproca total em projetos de software que pretendam continuar licenciados sob licença permissiva (SABINO, 2011).

Todas as licenças recíprocas totais foram elaboradas pela *Free Software Foundation* e fazem parte da família de licenças GPL (*General Public License*): GPL 2.0 e GPL v3 (SABINO, 2011).

3.3.3.1 Licença GPL 2.0

A licença GPL versão 2.0 (GPLv2), criada em 1991, é a segunda licença mais utilizada em projetos de software livre e de código aberto no mundo (BLACK DUKE, 2017). Essa licença é recomendada para projetos de desenvolvimento de software que buscam crescimento por meio de contribuições de terceiros, uma vez que as melhorias feitas no software devem se manter livres para poderem ser distribuídas (SABINO, 2011). Contudo, a licença GPL também pode ser utilizada em um modelo comercial de licenciamento dual, no qual a empresa fornece o software sob a licença GPL, obtendo os benefícios relacionados ao software livre, porém, ao mesmo tempo, disponibiliza o software sob alguma outra licença que permite formas de exploração vedadas pela GPL (SABINO, 2011).

Os usuários que pretendem utilizar, modificar e reutilizar componentes de software licenciados sob a licença GPLv2 devem cumprir uma série de obrigações, dentre as quais se destaca a de disponibilizar o código fonte (SABINO, 2011). Nos termos dessa licença, o código fonte inclui todos os módulos que ele contém, arquivos de definição de interface associados e *script* usados para controlar a compilação e instalação do executável (SABINO, 2011).

Outra obrigação fundamental para o cumprimento dos termos da licença GPLv2 está relacionado ao conceito de *copyleft* (SABINO, 2011). Nesse sentido, qualquer software que contém total ou parcialmente algum componente de software licenciado sob a licença GPL deve estar sujeito às mesmas restrições da licença GPL do trabalho original e, consequentemente, estar licenciado sob a licença GPL (SABINO, 2011). Dessa forma, não é possível que um trabalho derivado de um software com licença GPL seja licenciado com uma licença fechada, permissiva, ou qualquer outra licença, a não ser que haja a opção de utilizar outra licença (SABINO, 2011).

Além disso, a GPLv2 determina que os arquivos modificados (ou sejam os que foram reutilizados) precisam conter avisos proeminentes afirmando que os arquivos foram modificados e a data da modificação, de forma a proteger a reputação do autor do trabalho original (SABINO, 2011).

Afora essas obrigações supracitadas, que são exclusivas às licenças recíprocas totais, ao distribuir software licenciado sob a GPL, deve-se disponibilizar os avisos sobre o direitos autorais (**Quadro 7**) e o texto integral desta licença (SABINO, 2011).

Não obstante, a licença GPL estabelece que essas obrigações não precisam ser aplicadas às seções que possam ser consideradas independentes e não derivadas do programa, desde que tais seções sejam distribuídas em trabalhos separados (SABINO, 2011).

Quadro 7 – Texto de aviso sobre direitos autorais da licença GPLv2

```
One line to give the program's name and a brief idea of what it does.
Copyright (C) <year><name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or (at
your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
USA
```

Fonte: https://opensource.org/licenses/GPL-2.0

3.3.3.2 Licença GPL 3.0

A versão mais recente da GPL, a GPL versão 3.0 (GPLv3), foi criada em 29 de junho de 2007, após um longo período de discussão e revisão pública, com o objetivo de evitar algumas situações consideradas indesejáveis pela *Free Software Foundation* (SABINO, 2011).

Embora a GPL 3.0 e a GPL 2.0 sejam bastante similares, principalmente no que concerne ao princípio do *copyleft*, uma das principais diferenças entre estas licenças é que algumas partes da nova versão foram reescritas visando a adaptar a licença a mudanças de ordem legal e técnica (SABINO, 2011).

Nesse sentido, a GPLv3 trata com maior profundidade a questão de patentes com o intuito de evitar a reincidência de um acordo como o que foi celebrado entre as empresas Microsoft e a Novell relativo à distribuição do SUSE Linux[28], que foi considerado pela *Free Software Foundation* uma afronta aos princípios do software livre (SABINO, 2011). No **Quadro 8**, pode-se visualizar o aviso de direitos autorais que deve acompanhar todo trabalho licenciado sob a licença GPLv.3.

[28]Após alegar que o sistema SUSE Linux, licenciado sob a licença GPLv2 e distribuído pela empresa Novell, infringiu patentes da Microsoft, foi acordado em 2006 que a Microsoft não processaria os usuários desse sistema por infração de patentes desde que a Novell pagasse *royalties* à Microsoft para obter tais direitos (SABINO, 2011).

Além disso, outra mudança relevante foi a reescrita do texto da licença GPLv3, com o intuito de facilitar a compatibilidade com outras licenças, especialmente com a licença Apache 2.0, o que era uma demanda de diversas comunidades de software livre (SABINO, 2011).

Quadro 8: Texto de aviso de direitos autorais da licença GPL 3.0

```
<one line to give the program's name and a brief idea of what it
does.>

Copyright (C) <year><name of author>

This program is free software: you can redistribute it and/or
modify it under the terms of the GNU General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see
<http://www.gnu.org/licenses/>.
```

Fonte: https://opensource.org/licenses/GPL-3.0

3.3.4 Licenças recíprocas parciais

As licenças recíprocas parciais, conhecidas como *weak copyleft* (*copyleft* fraco), estabelecem que as modificações realizadas em um trabalho licenciado sob seus termos devem ser disponibilizadas sob a mesma licença (SABINO, 2011). Não obstante, caso o software licenciado sob uma licença recíproca parcial seja utilizado apenas como um componente de outro projeto de software, esse projeto não precisa estar sob a mesma licença, devendo-se somente disponibilizar o componente de software reutilizado e suas modificações (SABINO, 2011).

Dessa forma, considera-se que essas licenças equilibram adequadamente dois importantes fatores do modelo de software livre: 1) atração de interesse para a comunidade, o que garante a pujança e longevidade do código-fonte disponível; 2) possibilidade para que os desenvolvedores possam utilizar o trabalho original para criar outro sistema de software que será licenciado sob qualquer outra licença (SABINO, 2011).

As principais licenças recíprocas parciais disponíveis são a licença LGPL, Mozilla e a Eclipse.

3.3.4.1 Licença LGPL

A primeira versão da licença LGPL (Lesser General Public License) foi escrita em 1991 pela *Free Software Foundation* (SABINO, 2011). A versão mais recente da licença LGPL passou por grandes modificações no final de 2007 para se adequar à GPL v3 (SABINO, 2011).

Cabe destacar que, devido à complexidade dos termos e obrigações da licença LGPL, é necessário estar muito atento a todas as regras e exceções presentes no texto (SABINO, 2011). Contudo, em termos gerais a licença LGPL estabelece que os trabalhos que apenas usam a biblioteca que está sob essa licença e, por isso são considerados isoladamente, podem ser licenciados de outra forma (SABINO, 2011). Todavia, por outro lado, se o trabalho for baseado na biblioteca de forma mais próxima, ele deve ser licenciado como LGPL ou GPL (SABINO, 2011). Dessa forma, a fronteira exata entre esses dois casos nem sempre está clara, deixando margem para interpretação ambígua (SABINO, 2011).

3.3.4.2 Licença Mozilla

A primeira versão da *Mozilla Public License* (MPL) foi criada em 1998 quando o navegador de internet *Netscape Communicator* passou a ser licenciado como software (SABINO, 2011). Em 2012, foi lançada a versão mais recente dessa licença, a MPL 2.0. A literatura especializada considera que a MPL é u,ma licença recíproca parcial muito bem escrita e possui definições claras quanto às obrigações, o que facilita sua reutilização em outros projetos de software (SABINO, 2011; ROSEN, 2005).

Ao contrário da licença LGPL, a licença Mozilla define claramente que apenas código coberto pela licença deve ser redistribuído pelos mesmos termos da licença Mozilla, todavia, esse código também pode ser utilizado como componente em outros projetos ampliados, que podem estar sob outra licença (SABINO, 2011). Dessa forma, na prática, os arquivos que contém código licenciados sob a MPL devem seguir essa licença, enquanto os demais estão livres para utilizarem a licença desejada (SABINO, 2011).

3.3.4.3 Licença Eclipse

A criação da primeira versão da licença Eclipse, a IBM Public License, em 1999, fez parte do projeto estratégico da empresa IBM de contribuir para o desenvolvimento de projetos de software livre e de código aberto, bem como obter as vantagens proporcionados pelas contribuições desenvolvidas pelas diversas comunidades *open source* (CAPEK, 2005).

A versão mais recente da licença Eclipse, a Eclipse 2.0 (EPL-2.0), determina que as modificações realizadas sobre o código fonte licenciado sob esta licença sejam

disponibilizados, mas permite que o software como um todo possa ser licenciado sob outros termos (SABINO, 2011).

3.4 Compatibilidade entre as licenças de software livre e de código aberto

Uma vez analisadas as principais licenças de software livre e de código aberto (no **Quadro 9** é possível ver um resumo dessas licenças), cabe mencionar que, devido ao grande número de licenças de software livre disponíveis, é fundamental gerenciar possíveis conflitos entre licenças para evitar complicações legais (KON et al., 2011). Dessa forma, ao se iniciar um novo projeto de software, devem ser examinados da melhor forma possível os potenciais problemas de compatibilidade entre as licenças de componentes que serão utilizados (KON et al., 2011).

Para verificar a compatibilidade legal entre componentes é necessário considerar, por um lado, as obrigações que são colocadas nas cláusulas das licenças e, por outro, as diversas formas como componentes de software podem ser utilizados em conjunto e distribuídos (KON et al., 2011).

Quadro 9 – Resumo das licenças de software livre e de código aberto

Licença	Tipo de licença	Facilidade de interpretar a licença
BSD	Permissiva	Média
MIT	Permissiva	Alta
Apache	Permissiva	Alta
GPL v2	Recíproca total	Média
GPL v3	Recíproca total	Média
LGPL	Recíproca parcial	Baixa
Eclipse	Recíproca parcial	Baixa
Mozilla	Recíproca parcial	Baixa

Fonte: Baseado em Sabino (2011)

Nesse sentido, as licenças permissivas são as mais fáceis de compatibilizar com outras licenças, na medida em que permitem que trabalhos derivados sejam redistribuídos e sublicenciados sob outros termos, até mesmo como software proprietário (KON et al., 2011).

Dessa forma, ao escolher um software licenciado sob uma licença permissiva, as diminuem-se chances de incorrer em problemas legais relativos à incompatibilidade de licenças (KON et al., 2011). De qualquer forma, recomenda-se examinar cuidadosamente as obrigações e termos de uso de cada licença utilizada.

Esse tema da reutilização de componentes de software licenciados sob licenças de software livre e de código aberto será analisado com mais detalhe na **seção 5.3**. Não obstante, antes dessa análise, examinar-se-á o conceito de fábrica de software no capitulo 5 sobre Fábrica de Software.

4. Procedimentos metodológicos

O interesse do autor do presente estudo pelo tema da Fábrica de Software da Faculdade de Computação - UFMS teve início quando o mesmo passou a ter contato com esse assunto durante as reuniões do Colegiado de Curso de Engenharia de Software, a partir do segundo semestre de 2015[29].

A ideia de relacionar a questão da implantação da Fábrica de Software da Faculdade de Computação – UFMS com a temática da propriedade intelectual foi concebida e desenvolvida, no decorrer do primeiro semestre de 2016, durante os estudos deste autor relacionados com as disciplinas "Métodos de Pesquisa Aplicados à Gestão Pública" e "Práticas de Produção Técnico-Científica", no âmbito do Mestrado Profissional em Administração Pública (PROFIAP). Nessas disciplinas, este autor entrou em contato com métodos e técnicas de coletas de dados referentes à pesquisa de natureza qualitativa, como a estudo de caso, pesquisa-ação, pesquisa documental e observação participante.

4.1Coleta de dados primários

O presente estudo utilizou a técnica de observação participante como procedimento de coleta de dados primários. Segundo Denzin (1989, p.157-158), a observação participante é "uma estratégia de campo que combina, simultaneamente, a entrevista de informantes, a participação, a observação direta, e a introspecção". Dessa forma, um dos principais benefícios da técnica da observação participante é a habilidade do pesquisador de conseguir permissão para participar de eventos ou de grupos que seriam, de outro modo, inacessíveis à investigação científica (YIN, 2001).

Outro benefício do método da observação participante é a capacidade de proporcionar uma compreensão mais detalhada do fenômeno pesquisado (ABIB; HOPPEN; HAYASHI, 2013).

Adler e Adler (1987) desenvolveram três categorias analíticas de observação participante que se diferenciam pelo grau de participação do pesquisador no fenômeno estudado, a saber: periférica, ativa e completa. Os pesquisadores que utilizam a observação participante periférica fazem parte do grupo estudado, embora exerçam um papel mais marginal, conseguem adquirir informações e *insights* privilegiados (ADLER; ADLER, 1987).

[29] O autor do presente TCF foi designado para secretariar as reuniões do Colegiado do Curso de Engenharia de Software por meio da Instrução de Serviço n° 105, de 1° de outubro de 2015.

Com relação à observação participante ativa, os pesquisadores assumem alguns ou todos os papeis atribuídos aos principais membros do grupo estudado (ADLER; ADLER, 1987). Na observação participante completa, os pesquisadores fazem uma imersão no grupo estudado não somente assumindo papeis centrais, mas também assimilando a identidade do grupo estudado (ADLER; ADLER, 1987). As principais características, vantagens e desvantagens dos três tipos de observação participante podem ser visualizadas no **Quadro 10**.

Quadro 10 -Tipos de observação participante

Tipo de observação	**Principais características**	**Vantagens**	**Desvantagens**
Observação periférica	Reflete uma posição mais marginal e menos comprometida do pesquisador. Envolve contato diário ou quase diário com informantes-chave. O observador possui um papel pouco ativo.	Facilidade em se manter neutro frente a coleta de campo.	Dificuldade em obter a confiança e sua inserção no grupo pesquisado.
Observação ativa	O observador assume um papel mais central e principalmente funcional no grupo.	Mais facilidade na aceitação do observador e na obtenção da confiança por parte do grupo.	Por estar imerso no grupo, buscando uma auto-reflexão, recomenda-se retiradas periódicas do pesquisador do seu campo.
Observação completa	Divide-se em duas categorias: de oportunidades (caso o pesquisador já faz parte do grupo) ou por conversão (quando ele se torna parte efetiva do grupo).	Acesso irrestrito ao ambiente pesquisado com possibilidade de coleta completa de informações e detalhes.	Viés de observação oriundo da vivencia de longo período no Campo.

Fonte: Adler e Adler (1987) Apud Abib; Hoppen; Hayashi (2013, p.607)

Dessa forma, utilizou-se a técnica de observação participante periférica na coleta de dados sobre a Fábrica de Software da Facom-UFMS, uma vez que este autor não possui formação na área de Tecnologia da Informação e a falta de conhecimento técnico impediria o exercício de um papel central no fenômeno estudado. Nesse sentido, DeWalt e DeWalt (2011) afirmam que, geralmente, o pesquisador adota este tipo de observação participante porque será inserido em ambiente no qual não conhece a linguagem e cultura utilizadas no grupo estudado.

Com o intuito de se aproximar do objeto de pesquisa do presente Trabalho de Conclusão Final para realizar uma observação participante periférica, este autor solicitou, junto à Direção da Faculdade de Computação e aos professores do Curso de Engenharia de Software, sua participação na comissão responsável pela definição do processo operacional para implantação e execução da Fábrica de Software da Facom, que foi instituída pela Instrução de Serviço n° 55, de 7 de julho de 2016 (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2016).

Dessa forma, entre julho de 2016 e setembro de 2017, esta comissão realizou reuniões semanais para discutir e debater esses aspectos operacionais e práticos da Fábrica de Software da FACOM-UFMS. Com a finalidade de registrar os assuntos discutidos nessas reuniões, este autor realizou anotações numa agenda que totalizaram dez páginas.

4.2 Coleta de dados secundários

O presente estudo utilizou a técnica de pesquisa documental como procedimento de coleta de dados secundários. A busca por documentos secundários foi empreendida com intuito de aprofundar a compreensão do fenômeno estudado, a propriedade intelectual de programa de computador desenvolvido em universidades federais.

Dessa forma, utilizaram-se documentos internos da Faculdade de Computação, documentos públicos da Universidade Federal de Mato Grosso do Sul, legislação sobre propriedade intelectual de programa de computador e Acórdãos do Tribunal de Contas da União.

4.2.1 Documentos internos

Como resultado da observação participante periférica, obteve-se acesso a documentos internos da Faculdade de Computação relativos à Fábrica de Software. Dentre os documentos internos coletados e analisados, destaca-se o relatório técnico intitulado “Modelo de Gestão e Processo Padrão de Desenvolvimento de Software da Fábrica de Software da Facom/UFMS”, datado em 10 de julho de 2015. Também foram coletados e analisados resumos dos resultados das discussões realizadas no âmbito da comissão de implantação da FS-FACOM, compartilhados via dispositivo *Google Drive* entre os membros da comissão.

4.2.2 Documentos públicos

Utilizou-se a ferramenta de busca do Boletim Interno de Atos Oficiais da UFMS para acessar documentos publicados nessa publicação[30]. Foram coletadas e analisadas a Resolução do Conselho Universitário da instituição que aprovou a criação do curso de Engenharia de Software (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2014); a Resolução do Conselho de Graduação da UFMS que aprovou o Projeto Pedagógico do Curso de Engenharia de Software (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2015); a Resolução do Conselho de Faculdade da Facom que aprovou o Regulamento das disciplinas "Prática em Desenvolvimento de Software I" e "Prática em Desenvolvimento de Software II" UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2017b), as Instruções de Serviço da Direção da Faculdade de Computação que constituíram comissões responsáveis pela implantação da FS-FACOM, bem como a Instrução de Serviço que constituiu a Comissão Permanente da Fábrica de Software (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2015, 2016. 2016b, 2017).

Além disso, com o intuito de compreender e sistematizar as normativas da referentes ao processo de PD&I, este TCF analisou a Lei de Inovação (Lei n° 10.973/2004) e as seguintes normas da UFMS: Instrução Normativa n° 1/2016 da Pró-Reitoria de Planejamento e Orçamento, que normatiza os procedimentos para a formalização, celebração e execução dos Convênios e Congêneres; a Instrução Normativa n°2/2016 da Pró-Reitoria de Pós-Graduação e Pesquisa, que normatiza procedimentos para cadastro, submissão, análise e vigência de projetos de pesquisa coordenados por pesquisadores vinculados à UFMS; a Resolução n° 132/2015 do Conselho Diretor, que regulamenta relações da UFMS com Fundações de Apoio; e a Resolução n° 133/2017 do Conselho Diretor, que regulamenta a concessão de bolsas no âmbito da UFMS.

4.2.3 Legislação sobre propriedade intelectual de programa de computador

Coletou-se a legislação pertinente à propriedade intelectual de programa de computador no endereço eletrônico da Presidência da República. Em seguida, analisou-se a Lei de Programa de Computador (Lei n° 9.609/1998)[31]e, subsidiariamente, a Lei de Direitos Autorais (Lei n° 9.610/1998). [32]

[30] Pode-se acessar a ferramenta de busca do Boletim de Serviço da UFMS neste link:https://bse.ufms.br/
[31]Disponível no link http://www.planalto.gov.br/ccivil_03/leis/L9609.htm

[32]Disponível no link http://www.planalto.gov.br/ccivil_03/leis/l9610.htm

4.2.4Acórdãos do TCU

O presente estudo realizou pesquisa documental em acórdãos do Tribunal de Contas da União (TCU), com o intuito de investigar o entendimento do órgão responsável pela fiscalização da legalidade dos atos da administração pública federal sobre a questão da propriedade intelectual de programa de computador no âmbito de PD&I realizadas em Universidades federais.

Dessa forma, inseriram-se palavras-chave (os termos "Propriedade intelectual", "programa de computador" e "universidade federal") na plataforma de pesquisa da jurisprudência do TCU[33]. Além disso, optou-se por pesquisar acórdãos publicados entre 2005 e 2015, já que, nesse período, estava em vigência a Lei de Inovação (Lei 10.973 de 2004) antes das alterações advindas da Lei nº 13.243/2016. Após adotar os critérios supracitados, encontraram-se dois acórdãos na base de jurisprudência do TCU, a saber, Acórdão 2534/2011 – Plenário e Acórdão 5770/2014 – Segunda Câmara.

O Acórdão 2534/2011 – Plenário foi descartado para a análise documental porque não versa sobre o objetivo do presente estudo, uma vez que se trata de levantamento de riscos nas operações Ministério do Desenvolvimento, Indústria e Comércio Exterior. Nesse Acórdão, as palavras-chave do presente estudo ("Propriedade intelectual", "programa de computador" e "universidade federal") se referem às atividades de registro de propriedade intelectual de programa de computador junto ao Instituto Nacional Propriedade Industrial.

O Acórdão 5770/2014 – Segunda Câmara do TCU, proferido em sessão do dia 14/10/2014, foi considerado para realização de pesquisa documental, já que parte deste acórdão trata de problemas relativos à titularidade da propriedade intelectual do software SIE (Sistema de Informações Educacionais) desenvolvido por servidores da Universidade Federal de Santa Maria (UFSM).

A Segunda Turma do Tribunal de Contas da União, ao analisar contratos de cessão de uso e licenciamento do software SIE firmados entre a Fundação de Apoio à Tecnologia e Ciência, entidade vinculada à UFSM, e diversas instituições federais de ensino superior, concluiu que aquela fundação de apoio se apropriou indevidamente dos direitos patrimoniais do software SIE.

No início da análise da questão da titularidade da propriedade intelectual do software SIE, o Acórdão 5770/2014 consta que essa matéria é regulada pela Lei de Inovação Tecnológica (Lei nº 10.973, de 2 de dezembro de 2004) e pelas normas de proteção da propriedade

[33]A ferramenta de busca de jurisprudência do TCU está disponível no link: https://contas.tcu.gov.br/pesquisaJurisprudencia/#/pesquisa/jurisprudencia

intelectual de programas de computador, a Lei de Programa de Computadore, subsidiariamente, pela Lei de Direitos Autorais.

Ao afirmar que o disposto no art. 4º da Lei 9.609/1998 constitui a regra geral da atribuição da titularidade dos direitos patrimoniais do programa de computador[34], o Acórdão 5770/2014 conclui que os direitos patrimoniais do software SIE pertencem à UFSM. Contudo, segundo esse Acórdão:

> [...] qualquer controvérsia sobre eventual direito das pessoas físicas que criaram o sistema está dirimida pela formalização da Cessão de Direitos Patrimoniais efetuado pelo coordenador do projeto à UFSM, além do registro da propriedade intelectual. Assim, não restam dúvidas quanto à atribuição da titularidade dos direitos patrimoniais do autor à Universidade, ainda que possa ser destinada parte dos royalties ao criador, em caso de regulamentação dessa exceção. O software foi registrado no INPI sob o nº 091434 em 15/9/2008, com a titularidade da UFSM, após a cessão de direitos patrimoniais, do servidor considerado "autor" do software, ocorrida em 11/9/2008 (BRASIL, 2014, p. 14).

Dessa forma, verificou-se a importância de se formular um documento formal que enseje a cessão da titularidade dos direitos patrimoniais dos softwares desenvolvidos no âmbito das atividades da Fábrica de Software da Facom para a UFMS. Nesse sentido, o processo de formulação desse documento será analisado com mais detalhes na seção **6.2.2**.

4.3. Análise de dados qualitativos

Os dados qualitativos coletados neste trabalho foram analisados por meio da técnica de matriz de análise de dados (MILES; HUBERMAN; SALDAÑA, 2014; MILES; HUBERMAN, 1994). Essa técnica tem como objetivo facilitar a visualização de um grande

---

[34] Art. 4° da Lei n° 9.609: Salvo estipulação em contrário, pertencerão exclusivamente ao empregador, contratante de serviços ou órgão público, os direitos relativos ao programa de computador, desenvolvido e elaborado durante a vigência de contrato ou de vínculo estatutário, expressamente destinado à pesquisa e desenvolvimento, ou em que a atividade do empregado, contratado de serviço ou servidor seja prevista, ou ainda, que decorra da própria natureza dos encargos concernentes a esses vínculos.

§ 1º Ressalvado ajuste em contrário, a compensação do trabalho ou serviço prestado limitar-se-á à remuneração ou ao salário convencionado.

§ 2º Pertencerão, com exclusividade, ao empregado, contratado de serviço ou servidor os direitos concernentes a programa de computador gerado sem relação com o contrato de trabalho, prestação de serviços ou vínculo estatutário, e sem a utilização de recursos, informações tecnológicas, segredos industriais e de negócios, materiais, instalações ou equipamentos do empregador, da empresa ou entidade com a qual o empregador mantenha contrato de prestação de serviços ou assemelhados, do contratante de serviços ou órgão público.

§ 3º O tratamento previsto neste artigo será aplicado nos casos em que o programa de computador for desenvolvido por bolsistas, estagiários e assemelhados

volume de informações de cunho qualitativo, com o intuito de proporcionar a sistematização, análise e *insights* significativos acerca dessas informações (MILES; HUBERMAN; SALDAÑA, 2014; NADIN; CASSELL, 2004; MILES; HUBERMAN, 1994).

Uma matriz de análise de dados é formada geralmente por uma tabela composta de linhas e colunas, na qual cada cruzamento dessas linhas e colunas representa uma unidade de análise. (MILES; HUBERMAN; SALDAÑA, 2014; MILES; HUBERMAN, 1994). Dessa forma, o ato de decidir o que as colunas e linhas representam é parte essencial do processo de análise e interpretação dos dados, uma vez que essa decisão é baseada numa análise profunda dos dados pesquisados (MILES; HUBERMAN; SALDAÑA, 2014; MILES; HUBERMAN, 1994).

As matrizes podem ser descritivas ou explicativas (MILES; HUBERMAN, 1994). As matrizes descritivas objetivam fazer com que dados complexos se tornem mais compreensíveis, ao reduzi-los em subpartes, com o intuito de analisá-los com mais detalhe (MILES; HUBERMAN, 1994). As matrizes explicativas visam a explicar o porquê acontecem os fenômenos estudados segundo algumas regras (MILES; HUBERMAN, 1994).

Dentre esses dois tipos de matrizes, a matriz de análise descritiva é a que mais se alinha aos objetivos do presente trabalho, visto que essa matriz permite identificar padrões e categorias existentes, visando a delinear a "estrutura profunda" desses dados (MILES; HUBERMAN; SALDAÑA, 2014).

Dessa forma, o presente trabalho utilizou um tipo de matriz descritiva denominada matriz conceitualmente agrupada - *Conceptually clustered matrix* (MILES; HUBERMAN; SALDAÑA, 2014), com o intuito de sistematizar as variáveis, conceitos e/ou temas das normativas da UFMS referentes ao processo de PD&I.

5. Fábrica de Software

O termo fábrica de software começou a ser utilizado, nas décadas de 1960 e 1970, nos Estados Unidos e Japão, para se referir a certas organizações, dedicadas ao desenvolvimento de software, que experimentaram um grande aumento em sua produtividade (CUSUMANO, 1991a). Dessa forma, o sucesso de fábricas de software norte-americanas e japonesas, como IBM, Toshiba e Hitachi, deveu-se à adoção dos seguintes fatores: 1) Adaptação de métodos, ferramentas, controle de sistemas e padrões para diferentes tipos de produtos; 2) desenvolvimento de ferramentas para automatizar aspectos do gerenciamento de projeto, geração de código-fonte, documentação e testes; 3) refinamento de ferramentas de desenvolvimento de software; 4) busca de altos níveis de integração entre ferramentas através de *software workbenches*[35]*;* e 5) aumento gradual na oferta de produtos desenvolvidos com foco na funcionalidade e facilidade no uso desses produtos (CUSUMANO, 1991b).

Dessa forma, o conceito de Fábrica de Software se modificou ao longo das últimas décadas do século XX, conforme o **Quadro 11**:

Quadro 11 – Evolução do conceito de Fábrica de Software

Fase/período	**Características**
Fase 1: meados nos anos 1960	Gerência da estrutura e Organização básica Objetivos da manufatura de software são estabelecidos; Foco no produto é determinado; começa a coleta de dados sobre o processo.
Fase 2: início dos anos 1970	Padronização e Customização da Tecnologia Objetivos dos sistemas de controle são estabelecidos; Métodos padrões são estabelecidos para o desenvolvimento; Desenvolvimento em ambiente online; Padronização das habilidades por meio de treinamento de empregados; Bibliotecas de código-fonte são introduzidas; surgem metodologias integradas e ferramentas de desenvolvimento.
Fase 3: final dos anos 1970	Mecanização e Suporte ao processo Introdução de ferramentas para apoio ao controle de projetos;

[35] *Software workbenches* são plataformas que oferecem assistência para o desenvolvimento, manutenção e gerenciamento de sistemas de software. As características de um *Software workbenches* são as seguintes: interface gráfica para desenhar diagramas estruturados; repositório de informação central para armazenar e gerenciar informações; conjunto de ferramentas fortemente integrado. Para mais informações sobre esse tema, ver M., Carma, "Software Workbenches:The New Software Development Environment,"National Computer Conference, 1987.

	Introdução de ferramentas para a geração de código, teste e documentação; Integração com ferramentas de banco de dados.
Fase 4: década de 1980	Refinamento do Processo e Extensão Revisão dos padrões; Introdução de novos métodos e ferramentas; Estabelecimento de controle de qualidade e círculos da qualidade; Transferência de métodos e ferramentas para subsidiárias e terceiros.
Fase 5: década de 1990	Automação Flexível Introdução de ferramentas de automação de design; Introdução de ferramentas de apoio à reutilização; Introdução de ferramentas de apoio à análise de requisitos; Integração de ferramentas em plataformas de desenvolvimento; Disseminação de metodologias ágeis de desenvolvimento, como Scrum[36].

Fonte: Fernandes e Teixeira (2004)

As fábricas de software, segundo Fernandes e Teixeira (2004), podem ser definidas como um processo estruturado, controlado e melhorado de forma contínua, orientado para o atendimento a múltiplas demandas de natureza e escopo distintas, visando à geração de produtos de software, conforme os requerimentos documentados dos usuários e/ou clientes, da forma mais eficiente possível.

Recomenda-se que, antes de se implementar uma fábrica de software, é necessário delinear os seguintes aspectos: definição dos perfis funcionais e das respectivas atividades a serem desempenhadas por cada perfil; definição da metodologia de desenvolvimento de software; definição de ferramentas de apoio; definição de um plano de processos que contenha a descrição das atividades, com intuito de relacioná-las com seus respectivos artefatos e perfis funcionais e definir ferramentas de apoio a serem utilizadas (FERNANDES; TEIXEIRA, 2004).

Para que obtenham uma alta performance, Fernandes e Teixeira (2004) sugerem que as fábricas de software tenham os seguintes atributos:

> • Processo definido e padrão (desenvolvimento, controle e planejamento);
> • Interação controlada com o cliente (entradas e saídas da fábrica);
> • Padronização das solicitações de serviço;
> • Estimativas de custos e prazos baseadas no conhecimento real da capacidade produtiva através de registros históricos;

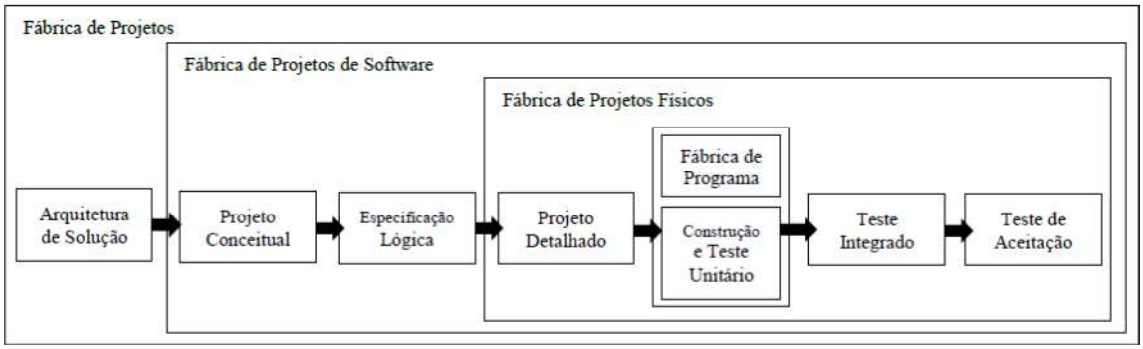
[36] As metodologias ágeis de desenvolvimento de software e, mais especificamente a abordagem Scrum, serão analisadas na seção **5.3.**

> • Controle rigoroso dos recursos alocados em cada demanda da fábrica;
> • Controle e armazenamento, em bibliotecas de itens de software, dos artefatos e do conhecimento produzido em cada etapa;
> • Controle permanente e em tempo real de todas as demandas;
> • Produtos gerados de acordo com os padrões estabelecidos pela organização;
> • Equipe treinada e capacitada nos processos organizacionais e produtivos;
> • Controle da qualidade do produto;
> • Processos de atendimento ao cliente;
> • Métricas definidas e controle dos acordos de nível de serviço (SLA) definidos com o cliente (FERNADES; TEIXEIRA, 2004, p. 116).

5.1 Tipos de Fábrica de Software

Fernandes e Teixeira (2004) propõem quatro tipos de Fábrica de Software, que se diferenciam entre si à medida que aumenta a complexidade e o escopo de fornecimento dos produtos e serviços prestados. Os quatro tipos de Fábrica de Software são os seguintes, em ordem crescente segundo a complexidade: Fábrica de programas, Fábrica de projetos físicos; Fábrica de Projetos de software e Fábrica de Projetos (FERNANDES; TEIXEIRA, 2004).

Figura 2 – Tipos de Fábrica de Software



Fonte: Fernandes e Teixeira (2004)

A Fábrica de programas é a menor unidade de fábrica de software existente, uma vez que visa apenas gerar códigos-fonte e realizar testes unitários[37] em programas de computador (FERNANDES; TEIXEIRA, 2004).

A fábrica de Projetos Físicos tem uma abrangência um pouco maior que o primeiro tipo, englobando também as atividades de projeto detalhado, teste de integração e teste de aceitação[38](FERNANDES; TEIXEIRA, 2004).

[37]O teste unitário é a fase de teste mais básica de um projeto de software porque visa identificar somente erros de lógica e de implementação em cada módulo de software, ao passo que as fases de teste mais complexas, como a de integração e a de sistema objetivam, respectivamente, descobrir erros associados às interfaces entre os módulos e identificar erros de funções de desempenho do sistema. Para mais detalhes sobre as fases de teste de um projeto de desenvolvimento de software ver DELAMARO, M.E; MALDONADO, J. C; JINO, M. **Introdução ao teste de software**. Rio de Janeiro**:** Campu**s**, 2007.

No caso da Fábrica de Projetos de Software, é necessário conhecer as necessidades do cliente, pois engloba as atividades de projeto conceitual[39] e especificação lógica[40] (FERNANDES; TEIXEIRA, 2004). Dessa forma, uma vez que esse tipo de Fábrica de Software requer maior capital intelectual por parte da equipe de desenvolvimento de software, ao contrário dos tipos anteriores que demandam somente habilidades de execução, deve haver uma preocupação permanente em minimizar a rotatividade de sua mão de obra (XAVIER, 2008).

A Fábrica de Projetos Ampliada é o tipo de Fábrica de Software mais completo segundo a classificação proposta por Fernandes e Teixeira (2004), porque oferece um amplo escopo de produtos, abrangendo desde a concepção da arquitetura da solução até a entrega do software pronto e testado. Com efeito, nesse tipo de Fábrica de Software, o software é apenas um dos produtos fabricados, já que são oferecidas soluções mais abrangentes na área de tecnologia da Informação, relativas a configurações de hardware e software básico, redes de comunicação, plataformas de desenvolvimento e de produção, soluções de gerenciamento de bases de dados (XAVIER, 2008).

---

[38]O teste de aceitação é o último teste antes da implementação do projeto de desenvolvimento de software. Esse tipo de teste é realizado com grupo seleto de usuários, geralmente os clientes do projeto de software, com o intuito de verificar se o software atendeu aos requisitos do cliente. Para mais detalhes sobre esse tema, ver R. Miller; C. Collins. **Acceptance testing**. Proc. XPUniverse, 2001. Disponível em: http://www.dsc.ufcg.edu.br/~jacques/cursos/map/recursos/Testing05.pdf

[39]Projeto conceitual ou modelo conceitual é uma das primeiras etapas do projeto de desenvolvimento de software. Com base na análise de requisitos e nas necessidades do usuário, é necessário descrever as informações que o sistema vai gerenciar por meio de representações derivadas do ponto de vista do usuário. De modo geral, a construção do modelo conceitual é formada por três elementos: atributos (informações alfanuméricas simples relacionadas a um conceito); conceitos (representação da informação complexa que agrega atributos, e.g. qualquer substantivo como livro, comprador e vendedor); e associações (tipo de informação que liga diferentes conceitos entre si). Dessa forma, o modelo conceitual, ao analisar o problema que o software pretende solucionar, faz parte do "domínio do problema" no projeto de desenvolvimento de software. Para mais detalhes sobre este tema, ver WAZLAWICK, R. S. **Análisee Projeto de Sistemas de Informação Orientados a Objetos**. 2. Ed. Rev. e Atual. Rio de Janeiro, Rj: Elsevier, 2011.

[40]Documento gerado na fase de análise de problema, no qual se descreve o relacionamento das interfaces internas do sistema, como a estrutura do programa, módulos de código individuais e parâmetros de dados. Para saber mais detalhes sobre este tema ver link disponível em: http://searchsoftwarequality.techtarget.com/definition/functional-specification

5.2 Fábrica de Software Acadêmica

A Fábrica de Software Acadêmica (FSA) é "uma modalidade de FS cujas unidades de produção são compostas principalmente pelo corpo acadêmico de instituições de ensino superior" (ROMANHA, 2016)[41]. Um dos principais objetivos de uma Fábrica de Software Acadêmica é proporcionar aos alunos experiência prática referente às disciplinas da área de Engenharia de Software, de modo que o aluno consiga aplicar em situações reais os conceitos teóricos estudados em sala de aula (ROMANHA, 2016).

Além disso, verifica-se que as instituições de ensino superior que proporcionam ambientes voltados para a vivência prática no desenvolvimento de software são bem avaliadas pelo Ministério da Educação (ROMANHA, 2016). As principais motivações para criação das Fábricas de Software Acadêmicas analisadas por Romanha (2016)[42] podem ser visualizadas na **Figura 3**.

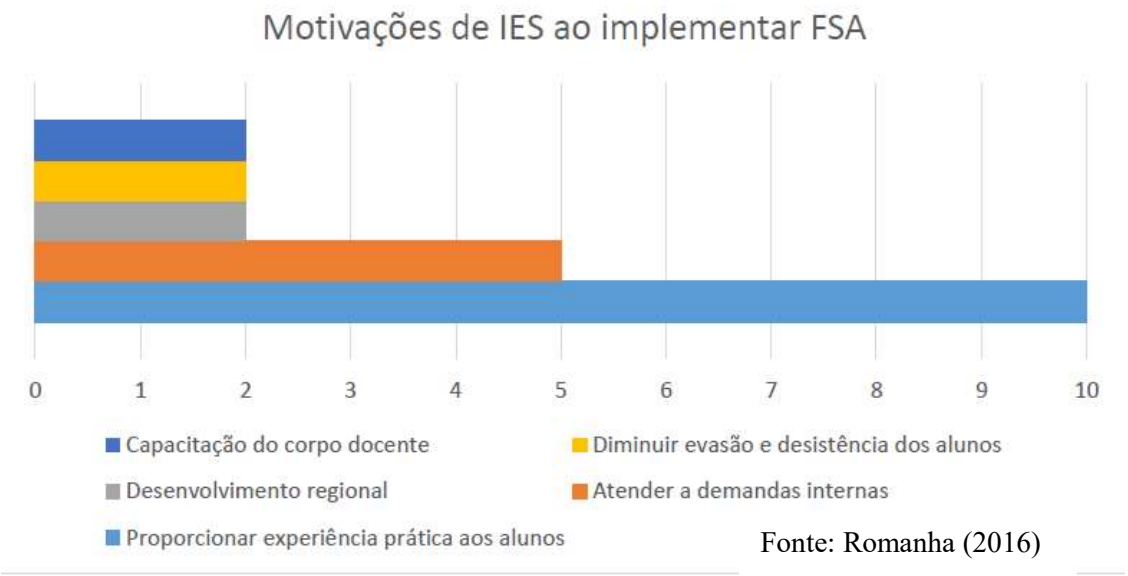
Geralmente, no início de suas atividades, as fábricas de software acadêmicas optam por atender demandas internas de suas próprias instituições, já que é mais fácil atender um cliente menos exigente e mais acessível, antes de se comprometer a atender demandas de clientes externos mais exigentes (ROMANHA, 2016).

Além disso, as soluções produzidas pela fábrica de software acadêmica possibilitam reduzir os custos das instituições de ensino superior com a contratação de empresas da área de tecnologia da informação, permitindo inclusive um alto nível de customização de suas demandas por soluções de software (ROMANHA, 2016).

[41]A presente pesquisa verificou que o conceito de "Fábrica de Software Acadêmica" é pouco explorado na literatura. Após pesquisar os termos "Fábrica de Software Acadêmica" e "Academic Software Factory" na plataforma *Google Scholar*, constatou-se que o estudo de Romanha (2016) foi o único que investigou profundamente esse conceito por meio de análise de revisão bibliográfica de dez artigos científicos que relataram experiências de implementação e gerenciamento de Fábricas de Software em instituições de ensino superior.

[42]As Universidades/Faculdades que foram analisadas pela revisão bibliográfica de Romanha (2016) foram as seguintes: Universidade Federal do Pará, Faculdade Lourenço Filho, Faculdade de Tecnologia de Jundiaí, Universidade Federal de São Carlos, Universidade Federal de Pernambuco, Instituto Federal de Goiás - Campus Inhumas, Instituto Federal de Educação Ciência e Tecnologia do Rio Grande do Sul – Campus Porto Alegre, Universidade Federal de Lavras, Universidade Estadual de Londrina, Universidade de Brasília - Faculdade do Gama.

Figura 3– Motivações para implementar FSA



Fonte: Romanha (2016)

5.2.1 Dificuldades na implementação de FSA

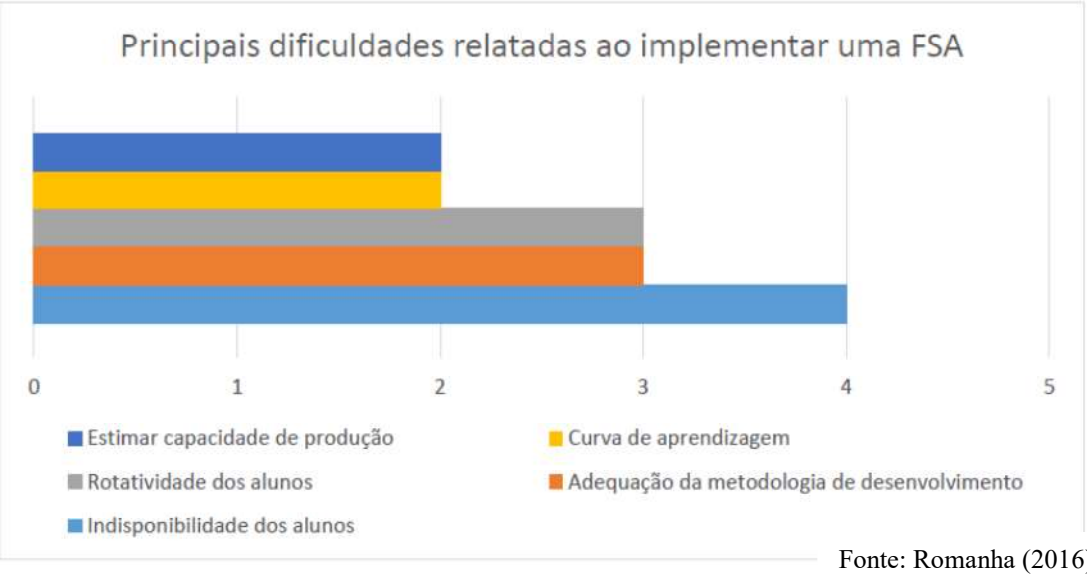
O estudo de Romanha (2016) também identificou as principais dificuldades enfrentadas pelas fábricas de software acadêmicas. Dentre esses problemas, destaca-se a o de conciliar os horários de disponibilidade dos alunos membros da equipe de desenvolvimento, uma vez que, nessas fábricas, além dos estagiários que trabalham em horário fixo, há os alunos que trabalham de forma remota em horários alternativos, o que viabil6iza a participação daqueles que já exercem atividade remunerada em horário convencional (ROMANHA, 2016).

Outro problema existente na Fábrica de Software Acadêmica é a alta rotatividade da mão de obra, composta basicamente por alunos de graduação (ROMANHA, 2016). Os principais motivos responsáveis pela saída dos alunos foram: a conclusão do curso e propostas de emprego no mercado (ROMANHA, 2016).

A curva de aprendizagem dos alunos com as metodologias e ferramentas utilizadas no desenvolvimento de software também gera dificuldades para o pleno funcionamento na Fábrica de Software (ROMANHA, 2016). Com o intuito de minimizar esse problema, Romanha (2016) verificou que as instituições pesquisadas adotam medidas como a estruturação de uma base de conhecimento organizada, *online*, acessível mediante identificação, o chamado repositório de artefatos de software[43].

[43] O repositório de artefatos de software é uma biblioteca virtual onde se armazena os diversos artefatos de software desenvolvidos por uma organização, principalmente o código fonte, com o intuito de reutilizá-los em projetos futuros. Para que funcione eficazmente, o repositório de artefatos deve ter ferramentas que busquem e recuperem adequadamente os artefatos armazenados. Esse assunto será melhor analisado na seção **5.6**.

Figura 4 – Dificuldades para implementar FSA



Fonte: Romanha (2016)

Dessa forma, identificou-se que adoção desse repositório de artefatos facilita e acelera o aprendizado dos novos integrantes das equipes no que concerne às atividades de desenvolvimento de software.

Relacionada com o problema da curva de aprendizagem, a dificuldade de estimar a capacidade de produção de cada equipe de desenvolvimento foi considerada uma tarefa complicada, principalmente durante a elaboração dos primeiros projetos da fábrica, porque não há base histórica para auxiliar nas previsões do esforço que será necessário para a execução dos projetos de desenvolvimento de software (ROMANHA, 2016).

Contudo, esta dificuldade tende a ser minimizada naturalmente com o passar do tempo, à medida que as equipes adquirem experiência, uma vez que o repositório de artefatos reaproveitáveis cresce e a base de conhecimento a respeito dos projetos anteriores permite gerar dados estatísticos de desempenho por tipo de componente criado (ROMANHA, 2016).

Outro problema significativo na implementação de FSA é a dificuldade de se adequar as metodologias de desenvolvimento de software usadas no mercado de trabalho à realidade do ambiente acadêmico, já que essas metodologias foram concebidas para serem aplicadas em um cenário onde a equipe de desenvolvimento é composta por profissionais experientes e cuja jornada de trabalho possui uma alta carga horária, geralmente em horário fixo (ROMANHA, 2016).

Com o intuito de compreender melhor esses problemas identificados na implementação de Fábricas de Software Acadêmicas, analisar-se-á com maior profundidade dois temas cruciais identificados por Romanha (2016): as metodologias ágeis de

desenvolvimento de software, especialmente o método Scrum, e, posteriormente, a reutilização de artefatos de Software.

5.3 Metodologias de desenvolvimento de software ágeis

A utilização de metodologias ágeis de desenvolvimento de software cresceu significativamente a partir da publicação do Manifesto Ágil, em 2001, que foi assinado por dezessete especialistas representantes das principais abordagens ágeis naquele período: XP, Scrum, DSDM, ASD,Crystal, Feature-Driven Development (MANIFESTO AGIL, 2001; HIGHSMITH; COCKBURN, 2001. Nesse manifesto, esses especialistas defendem a adoção de uma série princípios que, se forem respeitados, geram bons resultados em projetos de desenvolvimento de software. (MANIFESTO AGIL, 2001; HIGHSMITH; COCKBURN, 2001).

De modo geral, as metodologias ágeis de desenvolvimento de software prezam pela aproximação com o cliente em todas as etapas do projeto de desenvolvimento de software, por meio de entrega de versões intermediárias do software, com o intuito de obter *feedback* constante do cliente (STELLMAN; GREENE, 2015).

Além disso, outra característica marcante das metodologias de desenvolvimento ágeis é a primazia pela comunicação informal - face a face, entre os membros da equipe do projeto de desenvolvimento de software, através de reuniões periódicas, visando ao compartilhamento de informações e à reflexão sobre o andamento do projeto (STELLMAN; GREENE, 2015). Dessa forma, ao considerar a comunicação informal como o melhor método de transmissão de informações, as metodologias ágeis defendem que a elaboração de documentação formal de software deva acontecer somente nos casos estritamente necessários, como por exemplo, quando o cliente exigir aqueles documentos (STELLMAN; GREENE, 2015).

Com isso, a busca constante pela melhoria dos processos de desenvolvimento de software é outra característica das metodologias ágeis, de modo que a funcionalidade do software é o melhor parâmetro para medir sua qualidade (STELLMAN; GREENE, 2015).

Contudo, embora haja muitas similaridades entre as diferentes metodologias ágeis, cada uma delas possui suas peculiaridades (STELLMAN; GREENE, 2015). Dessa forma, como a abordagem Scrum é a mais utilizada no âmbito das Fábricas de Software Acadêmicas (ROMANHA, 2016), essa abordagem será analisada com maior profundidade.

5.3.1 Abordagem Scrum

A abordagem Scrum de desenvolvimento de software foi concebida por Ken Schwaber e Jeff Sutherland, no início dos anos 1990, com base na metáfora do jogo de Rugby[44] sugerida nos estudos de Takeuchi e Nonaka sobre uma nova abordagem de desenvolvimento de produtos industriais caracterizada pela alta produtividade e flexibilidade (SCHWABE et al., 1995; TAKEUCHI; NONAKA, 1986).

Segundo Schwaber e Sutherland (2013), a abordagem Scrum é "um *framework* dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível[45]".

De modo geral, o funcionamento da abordagem Scrum está baseado no trabalho de um time dividido em papeis que deverá entregar uma série de artefatos (produtos) em períodos previamente estabelecidos (eventos), conforme um conjunto pré-estabelecido de regras. Cada componente do Scrum tem um objetivo específico e é essencial para o uso e sucesso da metodologia (SCHWABER; SUTHERLAND, 2013). As regras do Scrum que administram as relações entre artefatos, eventos e papeis serão analisadas abaixo.

5.3.2 Papeis do Scrum

Um time Scrum é composto pelo *Product Owner* (PO), pela equipe de desenvolvimento e pelo Scrum Master. Uma característica essencial desse time é que ele deve ser auto-organizável e multifuncional (SCHWABER; SUTHERLAND, 2013). Além disso, os times Scrum entregam produtos de modo iterativo[46] e incremental, buscando aumentar as oportunidades de *feedback*. Entregas incrementais de produto 'pronto' garantem que uma versão potencialmente funcional do produto do trabalho esteja sempre disponível (SCHWABER; SUTHERLAND, 2013).

---

[44] Takeuchi e Nonaka (1986), após realizarem estudos de caso em empresas como Honda, Xerox, NEC e Canon, utilizaram a metáfora da posição *scrum* no jogo de Rugby para descrever a emergência de uma abordagem de produção holística, na qual os processos de produção são executados, por pequenas equipes, em múltiplas fases sobrepostas, com o intuito de aumentar a produtividade e flexibilidade dos processos. Takeuchi e Nonaka (1986) também identificaram que as equipes tinham habilidades multidisciplinares e havia constante interação entre seus membros, que trabalhavam juntos do início ao fim do processo produtivo.

[45] Essa definição consta no "Guia do Scrum" que foi elaborado pelos próprios criadores dessa abordagem. Esse manual explica detalhadamente os conceitos e o funcionamento do método Scrum. A versão em língua portuguesa desse manual está disponível no link:
https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf

[46] É importante entender o significado do termo iterativo para não confundi-lo com o termo *interativo*, pois esses termos têm significados diferentes. Segundo o dicionário Houaiss (2010), o adjetivo iterativo significa algo "repetido, feito mais de uma vez". De acordo com o especialista em desenvolvimento de software Emerson Roger Zuliani essa diferenciação é importante para compreender melhor o Scrum. Ver link disponível em: http://emersonzuliani.com.br/desenvolvimento-iterativo-e-incremental/

O *Product Owner*, ou dono do produto, é o responsável por maximizar e avaliar o valor do produto e do trabalho da equipe de desenvolvimento (SCHWABER; SUTHERLAND, 2013). A forma como isso é feito pode variar significativamente entre as organizações, times Scrum e indivíduos. Contudo, é importante ressaltar que o papel de *Product Owner só* pode ser exercido por uma pessoa e não um comitê. Dessa forma, para que o *Product Owner* tenha sucesso, toda a organização deve respeitar as suas decisões. Entre suas atribuições, destacam-se a definição clara dos itens do backlog[47] do produto; e a definição do grau de prioridade desses itens para melhor alcançar as metas e missões.

A equipe de desenvolvimento é formada por profissionais que realizam o trabalho de entregar uma versão potencialmente utilizável do produto por meio de incrementos no final de cada *sprint*[48] (SCHWABER; SUTHERLAND, 2013).

As equipes de desenvolvimento são autorizadas pela instituição para organizar e gerenciar seu próprio trabalho. Com isso, a sinergia resultante aprimora a sua eficiência e eficácia da equipe e dos processos (SCHWABER; SUTHERLAND, 2013). Além disso, as equipes de desenvolvimento são auto-organizadas, de modo que possuem autonomia para transformar o backlog do produto em incrementos[49] de funcionalidades potencialmente utilizáveis (SCHWABER; SUTHERLAND, 2013).

O terceiro papel previsto na metodologia é o Scrum Master, que é o responsável por garantir que o time Scrum cumpra a teoria, práticas e regras da metodologia Scrum. Além disso, o Scrum Master tem a incumbência de ajudar aqueles que estão fora do time a entender quais as suas interações com o time podem ser úteis e quais são desnecessárias (SCHWABER; SUTHERLAND, 2013).

Outra atribuição do Scrum Master é zelar pelos interesses e necessidades do *Project Owner,* buscando técnicas para o gerenciamento efetivo do backlog do produto; comunicando claramente a visão, objetivo e itens do backlog do produto para a equipe de desenvolvimento;

---

[47] O backlog do produto é uma lista ordenada ou priorizada de tudo que deve ser necessário no produto. O *Project Owner* é responsável pela definição do backlog do produto, incluindo seu conteúdo, disponibilidade e priorização. Geralmente, esses itens são ordenados por valor, risco, prioridade e necessidade. Os itens no topo da lista do backlog do produto determinam as atividades de desenvolvimento mais imediatas (SCHWABER; SUTHERLAND, 2013).

[48] *Sprint* é o principal componente da metodologia Scrum. Trata-se de um evento com duração de mais ou menos um mês corrido, durante o qual uma versão incremental potencialmente utilizável do produto é desenvolvida. Uma nova *sprint* somente começa após a conclusão da anterior. Dessa forma, o projeto deve ser dividido em *sprints*, sendo que a quantidade varia de acordo com as demandas do projeto. Cada *sprint* deve ter uma reunião de planejamento, reuniões diárias, trabalho de desenvolvimento, revisão e retrospectiva. Essas etapas da *sprint* serão analisadas mais adiante (SCHWABER; SUTHERLAND, 2013).

[49] O incremento é a soma de todos os itens do backlog do produto concluídos durante a *sprint*, acrescido de todos os outros das *sprints* anteriores. Ao final da *sprint*, um novo incremento deve estar pronto.

ensinando o time Scrum a criar itens de backlog do produto de forma clara e concisa; e facilitando os eventos Scrum conforme exigidos ou necessários (SCHWABER; SUTHERLAND, 2013).

5.3.3 Eventos do Scrum

Os eventos do Scrum são usados para criar uma rotina de trabalho bem estabelecida. Todo evento tem uma duração máxima pré-definida, garantindo que uma quantidade adequada de tempo seja utilizada no planejamento e execução do projeto, evitando perda de tempo e recursos ao longo desse processo. Além disso, cada um dos eventos previstos na *sprint* é uma oportunidade deliberada para inspecionar, adaptar e fazer mudanças necessárias em algum item do projeto (SCHWABER; SUTHERLAND, 2013).

Na reunião de planejamento da *sprint,* deve-se discutir o plano de trabalho a ser executado nos próximos trinta dias. A reunião de planejamento é dividida em duas partes, de modo que cada uma ocupa a metade desse tempo. Na primeira parte da reunião, define-se o que será entregue como resultado do incremento da *sprint*, com o intuito de implementar os itens de backlog do produto selecionados para a *sprint*, o chamado *backlog da sprint*[50]. Na segunda parte da reunião de planejamento, a equipe de desenvolvimento deve discutir como vai executar o trabalho necessário para entregar o incremento (SCHWABER; SUTHERLAND, 2013).

A reunião diária do Scrum é o evento mais frequente da *sprint* e tem duração de no máximo quinze minutos. O objetivo principal dessas reuniões é sincronizar as atividades dos membros da equipe de desenvolvimento e criar um plano para as próximas 24 horas. Nessa reunião, também se realiza uma inspeção do trabalho executado desde a última reunião diária (SCHWABER; SUTHERLAND, 2013).

Uma das principais vantagens das reuniões diárias é a melhoria na comunicação entre os membros da equipe de desenvolvimento, de modo que se identificam e removem impedimentos para o desenvolvimento do projeto. Com isso, o processo de tomada de decisões se torna mais rápido, aumentando o nível de conhecimento da equipe de desenvolvimento (SCHWABER; SUTHERLAND, 2013).

Outro evento previsto na abordagem Scrum é a revisão da *sprint*. Trata-se de uma reunião informal com o *Project Owner* e demais partes interessadas que é realizada no final

[50] O *backlog da sprint* é um conjunto de itens do backlog do produto selecionados para a *sprint*, adicionado do plano para entrega do incremento do produto a ser utilizado. Dessa forma, o *backlog da sprint* é a previsão da equipe de desenvolvimento sobre qual funcionalidade estará no próximo incremento e do trabalho necessário para entregá-la (SCHWABER; SUTHERLAND, 2013).

da *sprint*, com o objetivo inspecionar o incremento, receber *feedback* e adaptar o backlog do produto, caso seja necessário. Dessa forma, durante a revisão da *sprint*, discutem-se os aspectos positivos da *sprint*, os problemas encontrados e como estes foram resolvidos.

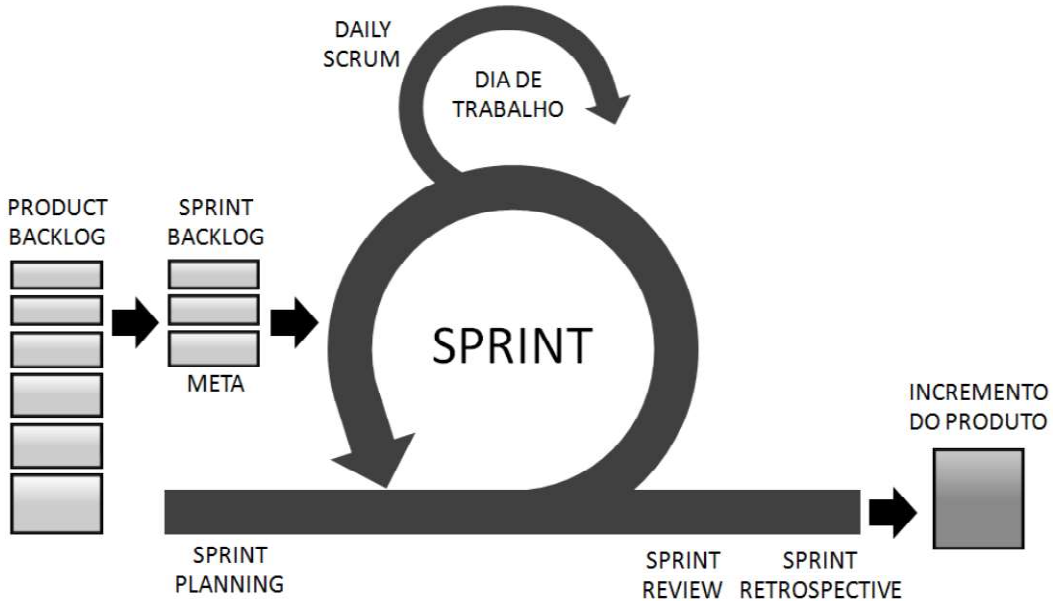
O resultado da reunião de revisão da *sprint* é um backlog do produto revisado e a definição do backlog do produto que será executado na próxima *sprint*. O backlog também pode ser ajustado completamente para atender novas oportunidades e demandas.

O último evento da *sprint* é uma reunião chamada de retrospectiva da *sprint*. Nessa reunião, a equipe de desenvolvimento revisa o trabalho realizado na *sprint* e discute formas de aumentar a qualidade do produto, com o intuito de implementar as melhorias na próxima *sprint*.

Dessa forma, uma vez terminada uma *sprint*, inicia-se outra *sprint*, continuando esse ciclo até a entrega da versão final do produto. Com o intuito de visualizar melhor a natureza cíclica das *sprints*, pode-se ver **Figura 5** que ilustra a dinâmica do funcionamento das *sprints* (STELLMAN; GREENE, 2015).

Uma vez analisada a metodologia de desenvolvimento ágil Scrum, passa-se a investigar outro tema importante para a implementação de Fábrica de Software Acadêmicas segundo o estudo de Romanha (2016): a reutilização de artefatos de software.

Figura 5 – Modelo básico da metodologia Scrum



Fonte: SABBAGH (2014)

Com o intuito de compreender a aplicação da abordagem Scurm, passa-se a analisar o trabalho de Carvalho e Mello (2012), que realizou uma pesquisa-ação para implementar o Scrum em uma pequena empresa de desenvolvimento de software localizada em Itajubá (MG). Após diagnóstico da realidade vivenciada pela empresa e treinamento com uma equipe da empresa composta por seis profissionais, os pesquisadores observaram que seria necessário adotar gradualmente o método Scrum, uma vez que algumas ferramentas, como a estimativa e velocidade do projeto, seriam de difícil implantação no início do projeto (CARVALHO; MELLO, 2012).

Concluído o processo de implantação do método Scrum, Carvalho e Mello (2012) constataram que houve melhoria na comunicação e aumento da colaboração entre envolvidos; aumento da motivação da equipe de desenvolvimento; diminuição no tempo gasto para terminar o projeto (prazo); e diminuição do risco do projeto (menor possibilidade de insucesso). O **Quadro 12** apresenta as principais vantagens e dificuldades encontradas durante a implementação das ferramentas do método Scrum na empresa pesquisada (CARVALHO; MELLO, 2012).

Quadro 12 - Análise da implementação das práticas gerenciais do Scrum

Item	**Vantagens/Facilidades**	**Desvantagens/Dificuldades**
Backlog do produto	Sua adoção melhorou o planejamento do time	O time sentiu dificuldade para elaborar sua primeira versão que geralmente é muito extensa.
Reunião diária	Sua utilização aumentou muito o controle do projeto. Os riscos foram minimizados.	Houve algumas falhas de periodicidade devido a desencontros de horários dos membros da equipe.
Sprint	A divisão do projeto em Sprints aumentou a motivação do time	Devido à inexperiência da equipe, alguns Sprints tiveram duração muito longa.
Planejamento do sprint	Sua realização melhorou o planejamento do Sprint.	As primeiras reuniões foram pouco produtivas devido à inexperiência da equipe.
Backlog do sprint	Sua boa elaboração é crucial para o sucesso ou fracasso do Sprint.	Seu superdimensionamento causou um atraso nos primeiros Sprints.
Revisão do sprint	Importante reunião para o recolhimento das lições aprendidas.	Tende a não ser executado pela equipe quando o time está ansioso para o planejamento do próximo Sprint.
Backlog de	É uma boa ferramenta para	Tende a ser negligenciado pelo próprio

impedimentos	cobrar a resolução de problemas conhecidos, diminuindo os riscos do projeto	Scrum Master, para evitar maiores responsabilidades.
Velocidade e estimativas	Muito importante para o controle do projeto e para o planejamento de custos da empresa	Foi o item mais complexo de ser implementado e que exigiu mais treinamento e novos conhecimentos.
Gráfico burndown	É uma ferramenta visual interessante, que deixa claro o que os números já mostravam.	Tende a ser abandonado, pois, à primeira vista, parece informação redundante.
Dono do produto	Facilita para o time, que passa a ter um representante do cliente dentro da própria empresa.	Não foi o caso do projeto estudado, mas provavelmente é difícil encontrar uma pessoa que entenda realmente o negócio do cliente a ponto de fazer bem este papel.
Scrum master	É imprescindível para tirar dúvidas quanto ao processo de desenvolvimento e liderar as cerimônias do Scrum.	Muitas vezes o time o vê como um gerente do projeto, deixando para ele atribuições que deveriam ser do próprio time.

Fonte: (CARVALHO; MELLO, 2012, p. 569).

5.4 Gerenciamento Ágil de Projetos

O trabalho de Eder et al. (2015) realizou revisão bibliográfica sistemática sobre a teoria tradicional de gerenciamento de projetos e uma nova abordagem, denominada Gerenciamento Ágil de Projetos (GAP), visando a identificar as práticas, ações, técnicas e ferramentas recomendadas por cada uma dessas abordagens. Em seguida, Eder et al. (2015) compararam a adoção dessas ações, técnicas e ferramentas em dois estudos de caso: uma empresa de software reconhecida pelo uso da abordagem tradicional e outra reconhecida pelo uso da abordagem ágil.

Os estudos da teoria tradicional de gerenciamento de projetos, desenvolvidos ao longo das últimas décadas, foram catalogados e sistematizados no reconhecido Guia PMBOK (EDER et al., 2015; PROJECT MANAGEMENT INSTITUTE, 2013). Nesse compêndio, segundo Eder et al. (2015), as boas práticas para o gerenciamento de projetos são compostas por três elementos: a ação em si (algo que gera resultado) e que pode utilizar uma ou mais técnicas (um procedimento sistemático) e uma ou mais ferramentas (artefatos que apoiam a realização da ação, no contexto da técnica).

Por outro lado, em contraposição à teoria tradicional de gerenciamento de projetos, emergiu, no início do século XXI, uma nova abordagem: o gerenciamento ágil de projetos

(EDER et al., 2015). Ao criticar a prática de se definir detalhadamente o escopo logo no início do projeto, o que é considerado uma boa prática segundo a teoria tradicional de gerenciamento de projetos, a abordagem do gerenciamento ágil de projetos defende que, no contexto de projetos inovadores, é necessário mudar o escopo do projeto (EDER et al., 2015).

Eder et al. (2015), após analisarem os dois estudos de caso, concluíram que a principal diferença entre a abordagem tradicional e a ágil reside na forma como cada uma emprega as técnicas e ferramentas para planejar e controlar o projeto. Dessa forma, "é na forma de execução, caracterizada pelas técnicas, que concentram-se as diferenças fundamentais entre as abordagens e não sobre o que é feito" (EDER et al., 2015, p. 494).

Outra importante constatação do estudo de Eder et al. (2015) é que existem seis diferenças fundamentais nas ações praticadas entre a abordagem tradicional e a ágil. Segundo Eder et al. (2015), essas diferenças são as seguintes: 1) a forma de elaboração do plano do projeto; 2) a forma como se descreve o escopo do projeto; 3) o nível de detalhe e padronização com que cada atividade do projeto é definida; 4) o horizonte de planejamento das atividades da equipe de projeto; 5) a estratégia utilizada para o controle do tempo do projeto; e 6) a estratégia utilizada para a garantia do atingimento do escopo do projeto. Pode-se comparar as diferenças entre as abordagens tradicional e ágil de gerenciamento de projetos no **Quadro 13**.

Quadro 13: Diferenças entre as abordagens tradicional e ágil de gerenciamento de projetos

Característica	Abordagem de gerenciamento de projetos tradicional	Abordagem de gerenciamento ágil de projetos
1) A forma de elaboração do plano do projeto	Há um único plano de projeto, que abrange o tempo total do projeto e contém os produtos, entregas, pacotes de trabalho e atividades.	Há dois planos de projeto: a) um plano geral que considera o tempo total de duração do projeto, mas que contém apenas os produtos principais do projeto; b) um plano de curto prazo (iteração) que contém apenas as entregas e atividades referentes a uma fração de tempo do projeto.
2) A forma como se descreve o escopo do projeto	Descrição exata do resultado final por meio de texto, com normas do tipo contratuais, números objetivos e indicadores de desempenho.	Descrição do resultado final de maneira abrangente, desafiadora, ambígua e metafórica.
3) O nível de detalhe e padronização com que cada atividade do projeto é definida	As atividades são descritas de maneira padronizada e organizadas em listas do tipo WBS. Contêm códigos e são classificadas em conjuntos de pacotes de trabalho, entregas e produtos do projeto.	Não há um padrão para a descrição das atividades, que podem ser escritas na forma de estórias, problemas, ações ou entregas. E não há uma tentativa de organização, apenas a priorização do que deve ser executado no momento.
4) O horizonte de planejamento das atividades da equipe de projeto	As listas de atividades são válidas para o horizonte total do projeto.	As listas de atividades são válidas para uma iteração, que é definida como uma fração do tempo total do projeto.
5) A estratégia utilizada para o controle do tempo do projeto	Empregam-se relatórios com indicadores de desempenho, documentos escritos, auditorias e análises de transições de fase. As reuniões da equipe não são frequentes.	Empregam-se dispositivos visuais que indicam entregas físicas do resultado final (cartazes, autoadesivos etc.). As reuniões são curtas e frequentes.
6) A estratégia utilizada para a garantia do atingimento do escopo do projeto	O gerente de projeto avalia, prioriza, adiciona ou altera as atividades do projeto para que os resultados estejam em conformidade com o escopo do projeto assinado com o cliente.	O cliente avalia, prioriza, adiciona ou altera o produto final do projeto, conforme a experiência com os resultados alcançados. A equipe altera as atividades para obter os resultados propostos pelo cliente.

Fonte: (EDER et al., 2015, p. 494)

Após ter analisado os artigos de Carvalho e Mello (2012) e Eder et al. (2015), passa-se a analisar os aspectos relevantes do trabalho de Conforto (2009) no que se refere a ferramentas para auxiliar no planejamento e controle do escopo e tempo de projetos que utilizam métodos ágeis, com o intuito de subsidiar a elaboração de plano de trabalho para os projetos que serão desenvolvidos na Fábrica de Software da Facom.

Conforto (2009) realizou pesquisa-ação em duas empresas pequenas de base tecnológica localizadas em São Carlos (SP). Inicialmente foi realizado diagnóstico dos problemas enfrentados no gerenciamento de projetos por duas empresas de base tecnológica localizadas em São Carlos (SP). Dentre os problemas identificados, destacaram-se as dificuldades em definir o escopo e elaborar cronograma com tarefas e entregas, ausência de uma sistemática para controle de desempenho dos projetos, fases dos projetos não são definidas, ausência de repositório de conhecimentos em gestão de projetos e ausência de metodologia para gestão de projetos (CONFORTO, 2009).

Além disso, baseado em extensa revisão bibliográfica sobre a abordagem tradicional de gerenciamento de projetos e a abordagem ágil, Conforto (2009) propôs um método híbrido que combina elementos dessas duas abordagens, chamado de IVPM2[51] (*Iterative & Visual Project Management Method* - Método iterativo e visual de gerenciamento de projetos), que tem como objetivo auxiliar a gestão ágil do escopo e tempo em projetos inovadores desenvolvidos por equipes pequenas e que são marcados pela incerteza.

5. 5 IVPM2: método de gerenciamento de escopo e tempo para projetos inovadores

O método IVPM2 tem como objetivo propiciar que a equipe do projeto consiga gerenciar, por meio de artefatos visuais, as entregas, permitindo relativa flexibilidade no controle do escopo e do tempo do projeto (CONFORTO, 2009).

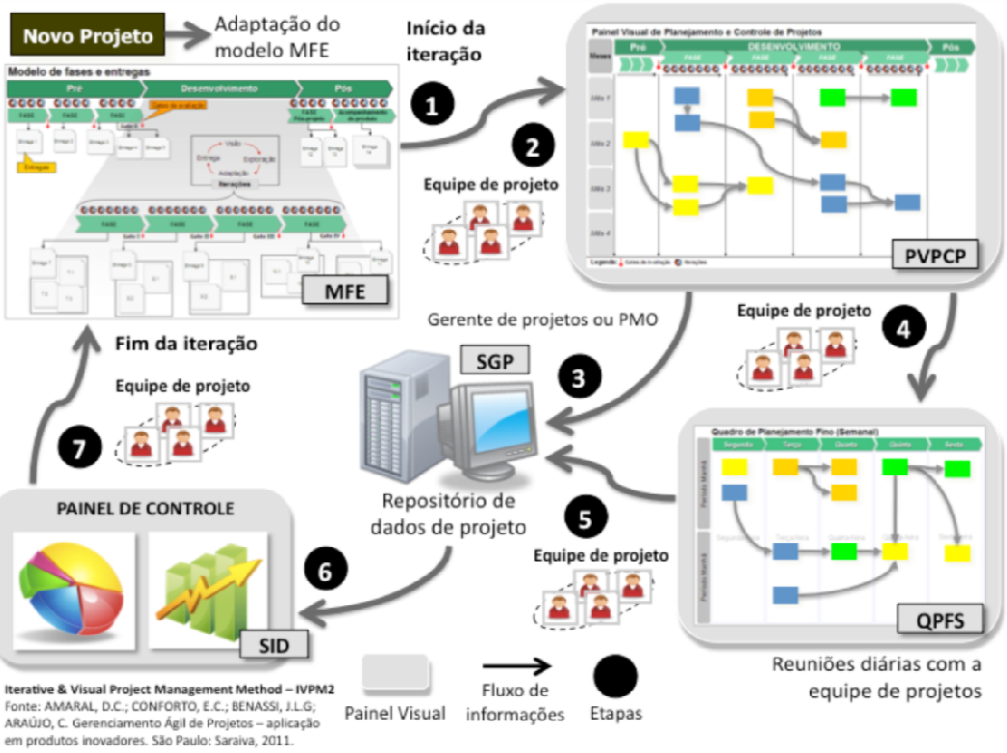
Outra característica fundamental do método IVPM2 é a simplicidade de suas técnicas, o que permite um aprendizado contínuo ao longo do desenvolvimento do projeto (CONFORTO, 2009)

O IVPM2 é um método formado por cinco componentes, que são aplicados ao longo de sete fases, conforme a **Figura 6** (CONFORTO, 2009). Os componentes do IVPM2 são os

[51] O método IVPM2 foi publicado pela primeira vez, em 2009, na dissertação "Gerenciamento ágil de projetos: proposta e avaliação de método para gestão de escopo e tempo" de autoria do pesquisador brasileiro Edivandro Conforto. A pesquisa que originou o método IVPM2 recebeu três prêmios internacionais das principais associações de gerenciamento de projeto do mundo, o *Project Management Institute* (PMI), com sede nos Estados Unidos, o *International project management association* (IPMA), com sede na Holanda, e o PMI *Educational Foundation*. Em 2011, Conforto e outros pesquisadores publicaram o livro "Gerenciamento ágil de projetos: aplicação em produtos inovadores", no qual sintetizaram a pesquisa que desenvolveu o método IVPM2 (CONFORTO et al., 2011)

seguintes: MFE (Modelo de Fases e Entregas); PVPCP (Painel Visual de Planejamento e Controle de Projetos); QPFS (Quadro de Planejamento Fino Semanal); SGP (Sistema para Gerenciamento de Projetos); e SID (Sistema de Indicadores de Desempenho).

Figura 6 – Método IVPM2 de gestão ágil de projetos



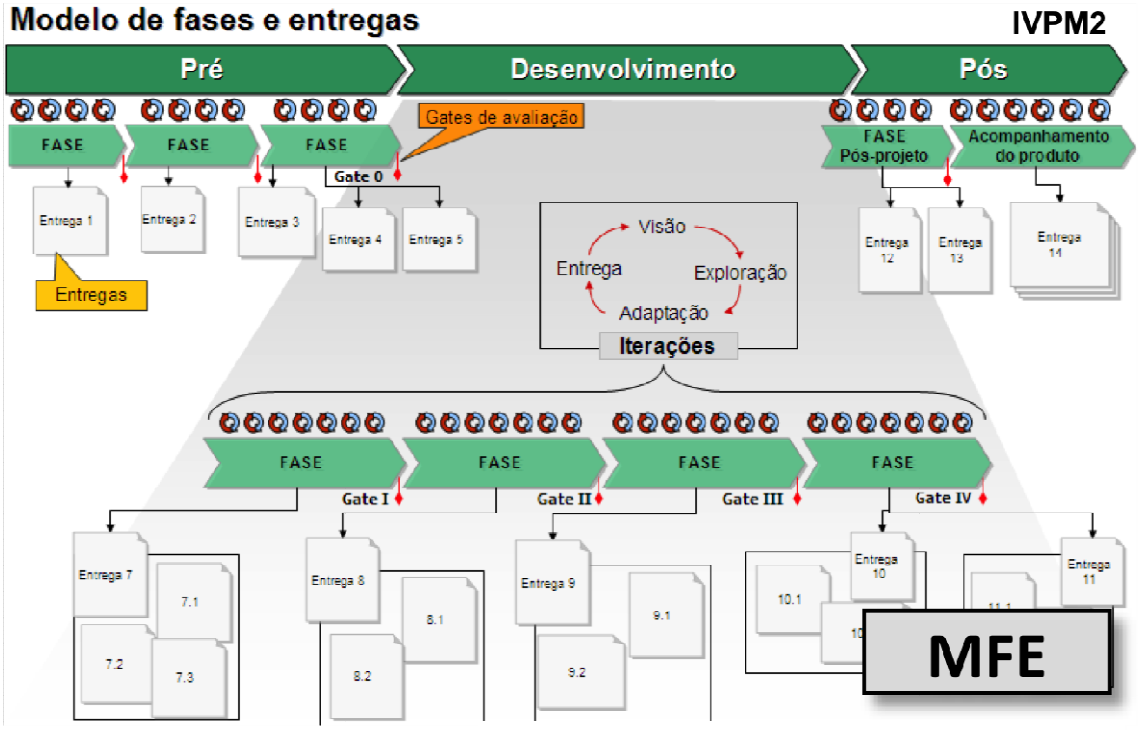
Fonte: Conforto et al. (2011)

5.5.1Modelo de Fases e Entregas (MFE)

O primeiro componente do IVPM2 é o Modelo de Fases e Entregas (MFE), que se trata de um modelo do processo de negócio simplificado (CONFORTO, 2009). O MFE é composto por macrofases (Pré-desenvolvimento; Desenvolvimento e Pós-desenvolvimento), nas quais se deve listar as entregas planejadas para cada fase do projeto, conforme a **Figura 7**.

Cada entrega descrita no MFE representa um resultado, que pode ser medido e avaliado; algo tangível (CONFORTO, 2009). O principal objetivo do MFE é que a equipe do projeto entenda e acompanhe a evolução do projeto (CONFORTO, 2009). A maioria das entregas são documentos, relatórios, procedimentos, esboços ou esquemas, que representem algum resultado do projeto e que agreguem valor para o cliente e para a equipe de projeto (CONFORTO, 2009).

Figura 7 – Modelo de Fases e entregas (MFE)



Iterative & Visual Project Management Method (IVPM2)
Fonte: AMARAL, D.C.; CONFORTO, E.C.; BENASSI, J.L.G.; ARAÚJO, C. Gerenciamento Ágil de Projetos – aplicação em produtos inovadores. São Paulo: Saraiva, 2011.

Fonte: Conforto et al. (2011)

Outra característica importante do MFE é a necessidade de adequá-lo às peculiaridades da organização e da equipe do projeto que o utiliza, bem como primar pela simplicidade (CONFORTO, 2009).

5.5.2Painel Visual de Planejamento e Controle de Projetos (PVPCP)

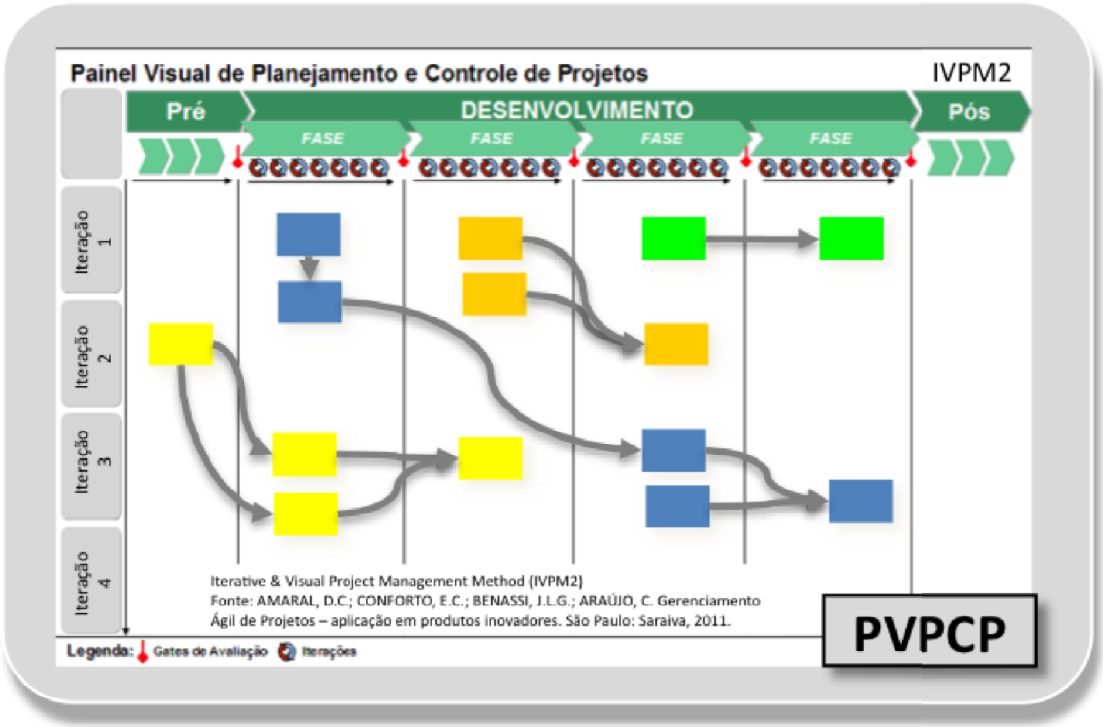
O PVPCP é um painel visual físico que integra, em uma escala de tempo, as fases do projeto com as entregas previstas no MFE (CONFORTO, 2009). Esse componente tem como objetivo definir as entregas do projeto ao invés das atividades. Dessa forma, durante a elaboração do PVPCP, a equipe do projeto não precisa definir exatamente o trabalho que deve ser realizado, mas somente definir as entregas do projeto, uma vez que as tarefas e atividades que precisam ser realizadas ainda não estão claramente definidas (CONFORTO, 2009).

No interior do painel do PVPCP, no espaço delimitado para cada fase, deve-se preencher um cada cartão ou recado autoadesivo com informações relacionadas às entregas do projeto, conforme a **Figura 8**. Sugere-se que cada entrega deve conter as seguintes informações: apelido do projeto (no caso de haver mais de um projeto); responsável pela entrega; nome da entrega e código (exemplo: Plano de projeto, E1); data para conclusão da

entrega (estimada e mais provável); tempo estimado para conclusão, podendo ser em dias, semanas, ou meses (CONFORTO, 2009).

Além disso, no PVPCP existem os *Gates* de avaliação entre as diferentes fases, nos quais são avaliados se os critérios de aprovação definidos para a transição de uma fase para outra foram atendidos (CONFORTO, 2009).

Figura 8 - Painel Visual de Planejamento e Controle de Projetos (PVPCP)



Fonte: Conforto et al. (2011)

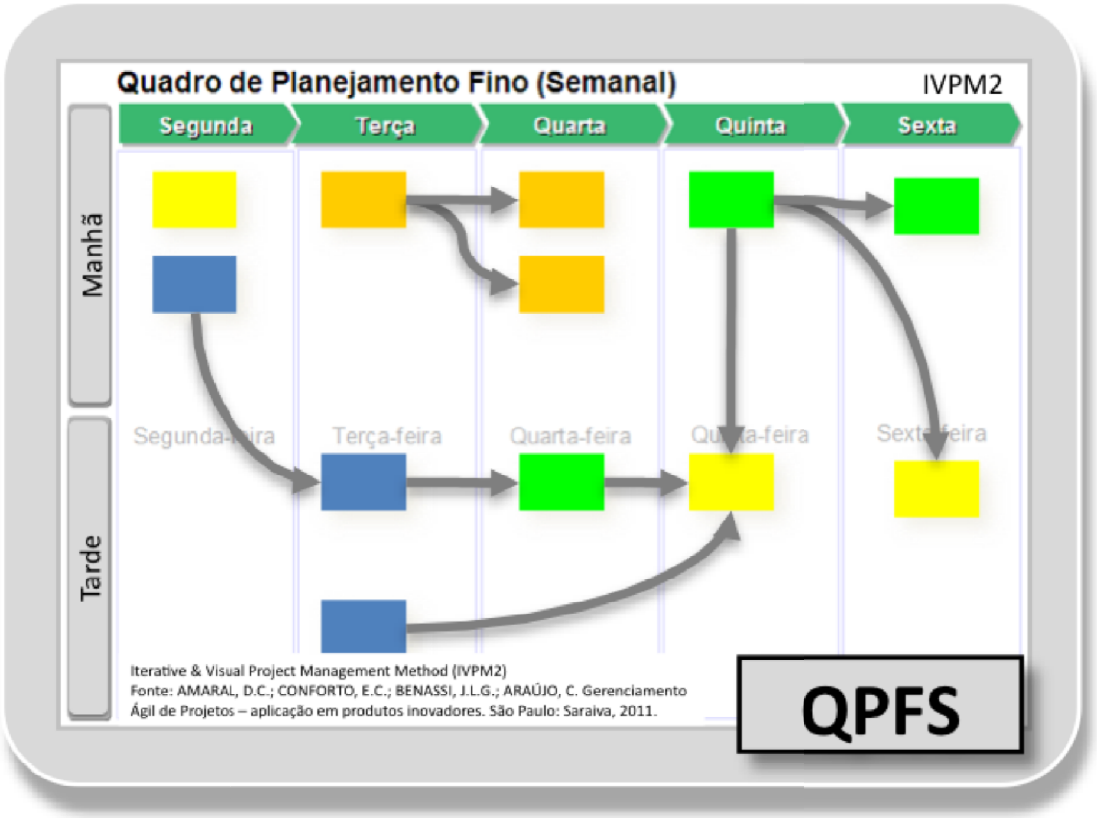
5.5.3 Quadro de Planejamento Fino Semanal (QPFS)

O Quadro de Planejamento Fino Semanal (QPFS) tem como objetivo planejar as atividades e pacotes de trabalho durante uma semana, servindo assim como agenda semanal das atividades do projeto para cada membro da equipe, conforme ilustra a **Figura 9** (CONFORTO, 2009). Dessa forma, o QPFS é um elemento que ajuda a orientar a equipe para a obtenção de resultados tangíveis (CONFORTO, 2009).

As atividades ou pacotes de trabalho devem ser estimados para serem concluídos no máximo em uma semana, conforme acordo prévio da equipe (CONFORTO, 2009). Caso a atividade precise de mais uma semana para ser concluída, deve-se indicar o que será entregue naquela semana, e deixar o restante da atividade para a próxima semana, separando-a em dois cartões distintos (CONFORTO, 2009).

Cada cartão inserido no QPFS deve conter o nome da entrega relacionada, conforme inserida no PVPCP, o nome da atividade ou pacote de trabalho, nome do responsável, e data prevista para sua finalização, sempre dentro do período de uma semana (CONFORTO, 2009).

Figura 9 - Quadro de Planejamento Fino Semanal (QPFS)



Fonte: Conforto et al. (2011)

Pode-se considerar cada semana como um ciclo de desenvolvimento dentro de uma iteração maior cujos resultados rápidos contribuem para a conclusão de entregas definidas no PVPCP (CONFORTO, 2009).

Dessa forma, a utilização do QPFS requer o desenvolvimento da autogestão e auto-organização dos membros da equipe, uma vez que os membros da equipe serão corresponsáveis pelo planejamento e controle das atividades e tarefas semanais do projeto (CONFORTO, 2009).

5.5.4 Sistema para Gerenciamento de Projetos (SGP)

O quarto componente do IVPM2 é um sistema para gerenciamento de projetos (CONFORTO, 2009). Enquanto os primeiros três componentes são artefatos visuais que auxiliam a equipe na rotina de planejamento e controle do projeto, o SGP tem como objetivo

armazenar os dados históricos dos projetos, gerar relatórios de desempenho e fornecer dados para acompanhamento do (CONFORTO, 2009). Dessa forma, a principal função desse componente é ser um repositório de dados dos projetos para geração de indicadores de desempenho e gerar dados para a análise do progresso dos projetos (CONFORTO, 2009).

Existem tanto softwares comerciais como livres que auxiliam no gerenciamento de projetos ágeis. Dentre os softwares pagos, o software *Version One*[52] é um dos mais utilizados, possuindo inclusive um pacote gratuito (CONFORTO et al., 2011). Com relação aos softwares livres, o *Endeavour*[53]e o *Redmine*[54] são uma das ferramentas mais utilizadas (ARAUJO, 2012)

5.5.5 Sistema de Indicadores de Desempenho (SID)

O último componente do IVPM2 é o Sistema de Indicadores de Desempenho (SID). O objetivo do SID no IVPM2 é analisar, do ponto de vista quantitativo, entregas no prazo determinado (CONFORTO, 2009). Dessa forma, deve-se analisar se as datas das entregas definidas no PVPCP ficaram dentro do prazo estipulado no plano de entregas (CONFORTO, 2009). Com isso, é possível avaliar o desempenho das entregas comparando-se as datas reais com as previstas no PVPCP (CONFORTO, 2009).

5.5.6Etapas do método IVPM2

Uma vez apresentados os componentes do modelo IVPM2, é pertinente ressaltar que cada organização deve fazer ajustes para adaptá-los às suas características e particularidades, de modo que se preserve o foco no desenvolvimento iterativo (CONFORTO, 2011). Nesse sentido, deve-se determinar o número e a quantidade de iterações segundo as especificidades do projeto e da organização (CONFORTO et al., 2011).

A seguir, serão apresentadas as sete etapas de aplicação do Método IVPM2 (CONFORTO et al., 2011):

1 Início da iteração – definir plano de iterações e entregas do projeto;

2. Inserir as entregas no PVPCP, utilizando os cartões autoadesivos;

3.Inserir os dados das entregas no SGP;

4. Decompor as entregas do PVPCP em pacotes de trabalho e atividades;

5. Executar os pacotes de trabalho e atividades e atualizar o SGP;

[52]www.versionone.com
[53]www.endeavour-mgmt.sourceforge.net
[54]http://www.redmine.org/

6. Gerar relatórios de desempenho do projeto utilizando o SGP;

7. Analisar resultados da iteração, verificar aprendizado e progresso do projeto.

1 Início da iteração – definir plano de iterações e entregas do projeto;

A primeira etapa requer a participação integral da equipe do projeto, uma vez que todos os membros devem participar da definição do plano de iterações, da organização e das entregas por iteração (CONFORTO et al., 2011). Nessa etapa, com base no MFE, deve-se definir o plano de iterações, contendo uma quantidade aproximada de iterações necessária para finalizar o projeto, a duração de cada iteração (CONFORTO et al., 2011). É necessário que cada iteração seja objetiva, devendo entregar um resultado ou um conjunto de resultados (CONFORTO et al., 2011).

2 Inserir entregas no PVPCP

Uma vez definidas as entregas das iterações, deve-se inserir essas informações, por meio de cartões autoadesivos, no Painel Visual de Planejamento e Controle de Projetos (PVPCP). Dessa forma, a organização dos cartões no PVPCP deve seguir a fase que pertence à entrega e o mês previsto para finalização (CONFORTO et al., 2011)

Outra informação importante, ao organizar as entregas no PVPCP, é a prioridade de execução das diversas entregas, a qual pode ser definida pelo posicionamento das entregas, colocando-as de cima para baixo no quadro (CONFORTO et al., 2011). Também é importante ressaltar que o processo de inserção das entregas no PVPCP deve ser realizado no início do projeto, bem como em todo início de uma nova iteração (CONFORTO et al., 2011).

Além disso, cabe destacar que todos os membros da equipe devem participar desse processo, já que esse envolvimento aumenta o comprometimento dos membros para alcançar as metas e objetivos do projeto (CONFORTO et al., 2011). Esse planejamento deve levar em conta o conhecimento e a experiência dos membros da equipe do projeto, podendo ser consultados especialistas e consultores externos, bem como dados históricos dos projetos realizados anteriormente para aumentar a precisão das estimativas (CONFORTO et al., 2011).

3- Inserir dados das entregas no Sistema para Gerenciamento de Projetos (SGP)

A terceira etapa do método IVPM2 é o registro dessas informações no Sistema para Gerenciamento de Projetos (SGP). Uma vez iniciado o projeto, as entregas inseridas no PVPCP precisam ser inseridas no SGP, indicando nome e código da entrega, iteração a que

pertence, fase, data de início e data prevista para encerramento do projeto (CONFORTO et al., 2011).

Para ser fiel aos princípios da filosofia ágil, recomenda-se que a parametrização dos dados seja feita da forma mais simples possível, com a menor quantidade de atributos, bastando, por exemplo, colocar a equipe e a quantidade de pessoas (CONFORTO et al., 2011). Essas informações já seriam suficientes para calcular o esforço desprendido nas tarefas e, portanto, os cálculos históricos e de capacidade (CONFORTO et al., 2011).

Qualquer alteração realizada no Painel Painel Visual de Planejamento e Controle de Projetos (PVPCP) - alocação da entrega em outra iteração, mudança de datas, quantidade de horas estimadas ou responsáveis por entrega - precisa ser e atualizada no software escolhido para gerenciar o projeto (CONFORTO et al., 2011). Dessa forma, o principal desafio dessa etapa é estabelecer um procedimento simples e prático que assegure a atualização e sincronização dos dados contidos no PVPCP para o software (CONFORTO et al., 2011). Recomenda-se que um membro da equipe seja incumbido de realizar as atualizações no SGP (CONFORTO et al., 2011).

4- Decompor as entregas do PVPCP em pacotes de trabalho e atividades

Na quarta etapa do IVPM2, define-se as atividades ou pacotes de trabalho relativos às entregas inseridas no PVPCP (CONFORTO et al., 2011). Com isso, nessa etapa, utilizam-se ciclos de desenvolvimento em escala semanal, de modo que são definidas, a cada semana, novas atividades e os resultados obtidos na semana anterior são revistos e avaliados (CONFORTO et al., 2011). Dessa forma, busca-se entregar resultados concretos a cada semana.

Para ter sucesso nessa etapa, deve-se preencher o Quadro de Planejamento Fino Semanal (QPFS), que permite o acompanhamento diário da evolução do projeto (CONFORTO et al., 2011). Dessa forma, recomenda-se o uso de reuniões rápidas e diárias entre os membros da equipe de projeto com o intuito de discutir a situação das entregas, atividades e pacotes, bem como relatar problemas prioritários da iteração e trocando sugestões (CONFORTO et al., 2011).

5- Executar as atividades e pacotes de trabalho

Na quinta etapa do IVPM2, os membros da equipe de projeto devem executar as atividades e pacotes de trabalho definidos no QPFS (CONFORTO et al., 2011). Nesta etapa,

não somente se deve atentar na execução da atividade propriamente dita, mas também se deve atentar para mudanças e melhorias que possam ocorrer durante a execução (CONFORTO et al., 2011).

É importante desenvolver as habilidades de autogestão e autodisciplina na equipe para torná-la capaz de reconhecer e se adaptar às mudanças necessárias para o projeto (CONFORTO et al., 2011).

A cada atividade ou pacote de trabalho finalizado, deve-se atualizar a entrega relacionada àquela atividade, segundo PVPCP, no sistema para o gerenciamento de projetos (SGP). Dessa forma, ao finalizar cada atividade ou pacote de trabalho, o cartão físico (recado autoadesivo) de papel deve ser retirado do QPFS e, posteriormente, atualizar essa informação no SGP (CONFORTO et al., 2011).

6 Gerar relatórios de desempenho com o SGP

A sexta etapa do método IVPM2 é a geração de relatórios de desempenho utilizando o SGP (CONFORTO, 2011). Esses indicadores podem ser qualitativos ou quantitativos: entrega no prazo, adequação ao custo, qualidade (a partir dos critérios definidos no MFE), atendimento aos objetivos do projeto, satisfação do cliente, e quantidade de mudanças no projeto (CONFORTO, 2011).

Faz-se necessário definir um conjunto de indicadores padrão para todos os projetos da organização, bem como definir indicadores específicos que poderão ser desenvolvidos para cada tipo de projeto (CONFORTO, 2011).

7 Avaliação dos resultados, tomada de decisão, aprendizado

Na última etapa do método IVPM2, deve-se avaliar os resultados de uma ou mais iterações são avaliados com o apoio dos relatórios de desempenho do projeto (CONFORTO, 2011). Dessa forma, a equipe do projeto deve discutir problemas e desafios enfrentados, resultados alcançados e soluções criativas adotadas que poderão ser úteis em outras iterações e projetos (CONFORTO, 2011).

Uma vez analisada as etapas do método IVPM2, que constituem ferramentas de gerenciamento ágil de projetos para auxiliar na gestão do tempo e escopo de projetos inovadores, como o desenvolvimento de software, passa-se a analisar outro importante fator para o funcionamento de Fábricas de Software: a reutilização de artefatos de software (ROMANHA, 2016).

5.6 Reutilização de artefatos de software

A reutilização de artefatos de software, nos últimos anos, tem se tornado um tema bastante difundido tanto na prática da indústria de software quanto na literatura especializada da área de engenharia de software (BAUER, 2016; OLIVEIRA, 2015; OLIVEIRA; WERNER, 2014). Em termos gerais, pode-se definir o conceito de reutilização de artefatos de software como a "utilização de artefatos de software já existentes para desenvolver novos sistemas de software" (BAUER, 2016, p. 32).

A literatura tem apontado que a reutilização de artefatos de software aumenta tanto a produtividade quanto a qualidade do processo de desenvolvimento de software, por meio da redução do trabalho e do tempo despendidos na elaboração de determinados artefatos de software - principalmente o código fonte, bem como da redução dos esforços de manutenção do sistema de software (BAUER, 2016; BAUER; VETRÓ, 2016; OLIVEIRA, 2015; OLIVEIRA; WERNER, 2014). Além disso, a prática da reutilização de artefatos de software facilita o trabalho dos desenvolvedores recém-chegados, uma vez que estes podem desenvolver artefatos baseados em modificações de trabalhos anteriores (OLIVEIRA, 2015).

Com relação à importância da prática de reutilização de artefatos de software, cabe destacar que organizações responsáveis por certificar a qualidade de empresas de desenvolvimento de software, como o MPS-BR[55], consideram que essa atividade é um importante indicador para se medir a qualidade de um processo de desenvolvimento de software.

Durante o processo de desenvolvimento de software, podem-se reutilizar artefatos internos e/ou externos (BAUER, 2016; BAUER; VETRÓ, 2016; OLIVEIRA, 2015). Os primeiros se referem aos artefatos desenvolvidos pela própria organização, em projetos de software anteriores, os quais são armazenados em um repositório interno de artefatos da organização (BAUER, 2016; OLIVEIRA, 2015). Com relação à reutilização de artefatos de software externos, analisar-se-á esse tema com mais detalhe na seção **5.6.1**.

A prática de reutilização de artefatos de software desenvolvidos internamente é dividida em duas atividades, a saber, as de produção e as de consumo. Na atividade de

[55]O MPS-BR (Melhoria de Processos do Software Brasileiro) é um modelo de qualidade de processo de desenvolvimento de software criado, em 2003, pela Softex (Associação para Promoção da Excelência do Software Brasileiro), com o objetivo de melhorar a capacidade de desenvolvimento de software nacional (SOFTEX, 2012). As certificações do MPS-BR levam em consideração normas e modelos internacionalmente reconhecidos como CMMI (*Capability Maturity Model Integration*) e nas normas ISO/IEC 12207 e ISO/IEC 15504. A certificação do MPS-BR é composta por sete níveis de maturidade, desde o nível G (o mais baixo) até o nível A (o mais alto). A prática de gerência de reutilização de artefatos de software é exigida a partir do nível E de maturidade (SOFTEX, 2012). A título de exemplo, cabe mencionar que a AGETIC (Agência de Tecnologia de Informação e Comunicação) da UFMS recebeu a certificação de nível G de maturidade em 2016.

produção, a organização deve identificar quais artefatos possuem o potencial de serem reaproveitados em outros projetos de software (BAUER, 2016; OLIVEIRA, 2015). Uma vez identificados esses candidatos potenciais, deve-se catalogar a especificação funcional e os possíveis usos desse artefato de software, a fim de armazená-lo no repositório interno da organização (BAUER, 2016; OLIVEIRA, 2015). Dessa forma, à medida que estes artefatos de software são armazenados no repositório, é necessário dispor de ferramentas eficientes para facilitar a busca e recuperação desses artefatos de software (BAUER, 2016; OLIVEIRA, 2015).

No que tange à atividade de consumo na reutilização de artefatos de software, deve-se envidar esforços em identificar quais são os artefatos armazenados no repositório interno que mais se adéquam às necessidades do projeto de desenvolvimento de software (BAUER, 2016; OLIVEIRA, 2015). Em seguida, deve-se recuperar o artefato selecionado e fazer as modificações necessárias para que este artefato seja integrado eficientemente no sistema de software que está sendo desenvolvido (BAUER, 2016; OLIVEIRA, 2015).

Contudo, para que essas práticas de reutilização de artefatos de software sejam implementadas de forma eficaz, é necessário que haja o engajamento da alta cúpula da organização para definir claramente qual estratégia será adotada, uma vez que a reutilização de artefatos de software requer um alto investimento inicial (tempo, recursos humanos e infraestutura) que será compensado a médio e longo prazo (BAUER, 2016; OLIVEIRA, 2015).

Nesse sentido, é preciso determinar os seguintes elementos: quais tipos de artefatos de software devem ser reutilizados (e.g. código-fonte)? Para qual finalidade esses artefatos serão reutilizados? Qual será a infraestrutura de repositório de artefatos de software? (BAUER, 2016; OLIVEIRA, 2015). Com relação ao último elemento, considerado pré-requisito para se implementar a prática de reutilização de artefatos de software (BAUER, 2016; OLIVEIRA, 2015), deve-se selecionar qual tipo de repositório de artefatos será utilizado (e.g. sistema de controle de versão[56], ferramentas de gerenciamento de tarefas e *bugs*[57]).

[56]Sistemas de controle de versão são ferramentas que permitem o armazenamento, em um repositório, de todas as versões dos códigos fonte desenvolvidos em um projeto de software (AQUILES; FERREIRA, 2014). Com isso, ao usar esse sistema, o desenvolvedor pode controlar e monitorar as alterações realizadas no código-fonte, que são chamadas de *commits* (AQUILES; FERREIRA, 2014). Os sistemas de controle de versão mais utilizados são o SVN, Mercurial, Bazaar e Git (AQUILES; FERREIRA, 2014). Dentre estes sistemas, cabe mencionar que o Git, criado pelo fundador do sistema operacional Linux, Linus Torvalds, é um dos mais utilizados no mundo (AQUILES; FERREIRA, 2014). Os desenvolvedores que utilizam o sistema de controle de versão Git podem armazenar as versões dos arquivos de código fonte em repositórios como o GitHub (https://github.com/) ou Gitlab (https://gitlab.com).

Após definir qual estratégia de armazenamento de artefatos internos deve ser adotada, a alta cúpula da organização deve envidar esforços para realizar as mudanças necessárias para implementar eficazmente essa estratégia, principalmente no que concerne à organização do repositório de artefatos (BAUER, 2016; OLIVEIRA, 2015).

Uma vez analisadas as principais características da prática de reutilização de artefatos de software desenvolvidos internamente, passa-se a inquirir o reuso de artefatos externos, mais conhecidos como software *open source*.

5.6.1 Reutilização de artefatos externos

A literatura aponta que a reutilização de componentes de software FOSS (*Free and Open Source Software* – software livre e de código aberto) aumenta significativamente a produtividade e diminui os custos do processo de desenvolvimento de software (WU et al., 2017, BLACK DUCK, 2017).

Os componentes de software FOSS são produzidos por terceiros, geralmente membros de comunidades *open source*, que contribuem para o desenvolvimento de projetos de software *open source*, armazenados em repositórios *online*, como GitHub[58] e SourceForge[59] (WU et al., 2017; MLOUKI; KHOMH; ANTONIOL, 2016; AQUILES; FERREIRA, 2014).

Dessa forma, devido à crescente oferta de componentes de software *open source* disponibilizados nesses repositórios *online,* cada vez mais empresas e organizações de desenvolvimento de software têm reutilizado artefatos de software FOSS para otimizar sua capacidade produtiva (BLACK DUCK, 2017). Contudo, ao contrário da reutilização de artefatos de software internos, o reuso de componentes de software *open source* apresenta uma problemática adicional: a questão da compatibilidade entre as diversas licenças *open source* (WU et al., 2017; MLOUKI; KHOMH; ANTONIOL, 2016).

Conforme visto nas seções **3.3.2**, **3.3.3** e **3.3.4**, ao utilizar componentes de software *open source*,devem-se cumprir as obrigações impostas pelas diferentes licenças, sejam elas permissivas, recíprocas parciais ou recíprocas totais (SABINO, 2011). No entanto, a literatura tem apontado que, não raramente, as organizações de desenvolvimento de software têm

[57]Sistemas de gerenciamento de tarefas e *bugs* são ferramentas que ajudam a gerenciar, monitorar e resolver tarefas relacionadas ao processo de desenvolvimento de software, principalmente aquelas referentes às falhas e defeitos do sistema, os chamados *bugs* (BERTRAM et al., 2010). Dessa forma, à medida que são gerados *tickets* para acompanhar a evolução dessas tarefas, é possível anexar os artefatos relacionados a essas atividades e verificar o histórico dessas atividades no repositório dessas ferramentas (MENEELY; CORCORAN; WILLIAMS, 2010). Os sistemas de gerenciamento de tarefas e *bugs* mais utilizados são *Jira, Redmine* e *Bugzilla* (OLIVEIRA, 2015).

[58]https://github.com/

[59]https://sourceforge.net/

falhado em cumprir com essas obrigações (WU et al., 2017; BLACK DUKE, 2017; MLOUKI; KHOMH; ANTONIOL, 2016).

Nesse sentido, em auditoria realizada em cerca de mil aplicativos de software comerciais, o *Center for Open Source Research and Innovation,* um centro de pesquisa vinculado à empresa de software Black Duke,constatou que 85% desses aplicativos de software continham componentes de software *open source* com licenças conflitantes (BLACK DUKE, 2017). O principal tipo de incompatibilidade de licença identificado – 75% dos casos – foi algum conflito que envolvia licenças da família GPL (BLACK DUKE, 2017). Embora a maioria desses aplicativos tenha utilizado componentes de software com alguma licença da família GPL, somente 45% deles cumpriam efetivamente as obrigações estabelecidas por este tipo de licenças, entre as quais se destaca a obrigação de licenciar todo o sistema de software sob uma licença da família GPL (BLACK DUKE, 2017).

Em outro estudo sobre a incompatibilidade de licenças *open source,* Wu et al. (2017) analisaram diversos tipos de inconsistências de licenças em projetos de software *open source*, com o intuito de identificar se diferentes projetos reutilizaram o mesmo componente de software, mas com licenças discrepantes. Dessa forma, investigou-se uma amostra aleatória de 10514 projetos que utilizaram a linguagem de programação Java e foram hospedados no repositório GitHub (WU et al., 2017).

Após analisarem todos os arquivos de código fonte desses projetos por meio das ferramentas CCFinder[60] e Ninka[61], Wu et al. (2017) constataram que as licenças de alguns arquivos de código fonte foram modificadas pelos projetos de *open source.* Com o intuito de investigar a causa dessas inconsistências, Wu et al. (2017) verificaram manualmente alguns

[60] CCFinder é uma ferramenta de detecção de arquivos de código fonte clonados, ou seja, arquivos idênticos ou semelhantes entre si, que foi desenvolvida por Kamiya, Kusumoto e Inoue (2002). Essa ferramenta permite organizar os arquivos de código fonte semelhantes em grupos. Wu et al. (2017), ao utilizar o CCFinder para analisar todos os arquivos de código fonte da amostra aleatória de 10,514 projetos Java, organizou esses arquivos em 199,284 grupos. O número de arquivos de código fonte nesses grupos variou de 2 a 1514.

[61]Ninka é uma ferramenta capaz de detectar mais de 110 tipos de licenças *open source* contidas em arquivos de código-fonte (GERMAN; MANABE; INOUE, 2010). Para cada arquivo de código fonte analisado, a ferramenta Ninka pode fornecer três resultados: a) apresenta o nome das licenças identificadas; b) identifica a existência de uma licença, mas diz que se trata de uma licença desconhecida pela ferramenta, requerendo assim uma análise manual; ou c) acusa a não existência de licença no arquivo analisado (GERMAN; MANABE; INOUE, 2010). No estudo de Wu et al. (2017), utilizou-se a ferramenta Ninka para identificar as licenças contidas nos arquivos dos199,284 grupos organizados pelo CCFinder. Após essa análise, constatou-se que 13,916 dos grupos, 7% do total, continham arquivos de código fonte com licenças discrepantes. A ferramenta Ninka é um software distribuído sob a licença GNU/GPL v2.0 e está disponível para *download* no link: https://github.com/dmgerman/ninka.

arquivos. Essas modificações foram classificadas em três tipos: mudanças seguras, não seguras e incertas com relação ao cumprimento das obrigações estabelecidas pelas licenças:

> **1-Mudanças seguras**: quando o autor do código fonte ou os desenvolvedores que reutilizaram o código fizeram modificações no texto da licença original, mas estas mudanças respeitaram os termos de condição da licença (WU et al., 2017).
>
> **2-Mudanças inseguras:** os desenvolvedores que reutilizaram o código fonte fizeram alterações não permitidas pela licença original, como por exemplo, mudar a licença original de um componente por outra licença sem o consentimento do autor (WU et al., 2017).
>
> **3-Mudanças incertas:** Não se pode determinar com clareza se a alteração da licença de um arquivo de código fonte infligiu alguma obrigação da licença original. Isso acontece quando, por exemplo, não é possível localizar o repositório do arquivo de código fonte original ou o atalho que leva a este repositório está corrompido (WU et al., 2017).

Em face da problemática da incompatibilidade de licenças em projetos de software *open source* disponibilizados em repositórios *online,* ao reutilizar esses artefatos, recomenda-se verificar se existe incompatibilidade de licenças nos seus arquivos de código fonte (WU et al., 2017).

Com o intuito de ensejar uma estratégia sistemática para realizar esse trabalho de inspeção de licenças de componentes de software *open source*, alguns autores sugerem que as organizações de desenvolvimento de software adotem uma política de gerenciamento para a utilização desses componentes (HADDAD, 2016; KEMP, 2009)

5.6.2 Política de Gerenciamento de Software Open Source

Uma política de gerenciamento de *open source* em organizações tem como objetivo principal diminuir os riscos relacionados à utilização de componentes de software livre e de código aberto, principalmente no que concerne à questão da compatibilidade das diversas licenças desses componentes (HADDAD, 2016; KEMP, 2009). Dessa forma, essa política deve ter os seguintes princípios (KEMP, 2009):

1) Confiança no código aberto utilizado: a organização deve saber a origem dos componentes de software livre e de código aberto que estão sendo reutilizados pelos seus desenvolvedores;
2) Aquisição de código aberto: a organização deve saber quais componentes de FOSS estão sendo usados em seus projetos de desenvolvimento de software;

3) Identificação de código aberto: a organização deve saber qual a funcionalidade dos componentes de FOSS nos softwares desenvolvidos por sua equipe, bem como a localização desses componentes na arquitetura do sistema;
4) Papeis e responsabilidades: a organização deve determinar os papeis de cada membro da equipe responsáveis pela implementação da política de gerenciamento de FOSS;
5) Cumprir com as obrigações das licenças: a organização deve ser capaz de cumprir com as obrigações estabelecidas pelas licenças dos componentes de FOSS utilizados em seu projeto de software.

Embora esses princípios sejam semelhantes para a maioria das organizações que desejem adotar uma política de gerenciamento de FOSS, os elementos e a forma de aplicação dessa política variam de acordo com as especificidades de cada organização (KEMP, 2009). Dessa forma, para que seja implementada de modo eficiente e eficaz, a estratégia de gerenciamento de FOSS deve não somente ser elaborada pela alta cúpula da organização, mas também ser condizente com as operações rotineiras da organização de desenvolvimento de software (KEMP, 2009).

A cúpula da organização deve formular a política de gerenciamento de FOSS de acordo com seus objetivos estratégicos, uma vez que essa deliberação é fundamental para a definição dos aspectos operacionais da utilização de componentes de software de código aberto (HADDAD, 2016; KEMP, 2009). Nesse sentido, para ilustrar a importância dessa decisão, a direção da organização precisa definir com clareza algumas questões, como por exemplo, se seus projetos de desenvolvimento de software serão licenciados como software de código fechado. Em caso afirmativo, a política de gerenciamento de FOSS da organização precisa evitar a utilização de componentes de software com a licença da família GNU/GPL, uma vez que essas licenças são incompatíveis com softwares de código fechado (LINDMAN; PAAJANEN; ROSSI, 2010). Dessa forma, a formulação da política de gerenciamento de FOSS, geralmente, está alinhada com a política de propriedade intelectual da organização (HADDAD, 2016; KEMP, 2009).

No que tange especificamente aos aspectos operacionais de uma política de gerenciamento de softwares livres e de código aberto, é necessário que se leve em conta a rotina de procedimentos e o processo de desenvolvimento de software da organização (HADDAD, 2016; KEMP, 2009). Nesse sentido, recomenda-se que os processos de inspeção das licenças de FOSS utilizadas no projeto de software sejam incluídos, como marcos de transição dos diversos ciclos iterativos do processo de desenvolvimento de software da organização (HADDAD, 2016). Com isso, busca-se evitar que a implementação dessa política

sobrecarregue excessivamente o já complexo trabalho da equipe de desenvolvimento de software (HADDAD, 2016; KEMP, 2009).

Outra medida indispensável para a implementação de uma política de gerenciamento de FOSS é a designação de profissionais responsáveis pela execução e monitoramento dos procedimentos relativos ao bom uso de componentes FOSS nos projetos de software da organização (HADDAD, 2016; KEMP, 2009). Embora esses profissionais não precisem se dedicar exclusivamente a esta atividade, eles necessitam ter conhecimento técnico e legal (relativo às condições de uso das diferentes licenças de software *open source*) para exercer eficazmente o papel de executar e monitorara política de gerenciamento de FOSS na organização (KEMP, 2009).

Além disso, para que a política de gerenciamento de *open source* tenha sucesso, é fundamental que todos os membros da equipe de desenvolvimento da organização recebam treinamento sobre noções básicas das principais licenças de software FOSS e seus termos de uso (HADDAD, 2016; KEMP, 2009). O objetivo desse treinamento é criar uma cultura de conscientização dos riscos e benefícios relacionados à utilização de componentes *open source*, bem como divulgar a política da organização referente a este tema (HADDAD, 2016; KEMP, 2009). Esse treinamento pode ser oferecido na forma de cursos formais ou seminários (HADDAD, 2016).

Uma vez apresentados os pré-requisitos para a formulação de uma política de gerenciamento de componentes FOSS na organização, passa-se a delinear os procedimentos necessários para se implementar efetivamente essa política, conforme as recomendações de Haddad (2016):

1) **Os componentes de software livre e de código aberto devem ser identificados, revisados e aprovados**:

Toda vez que algum membro da equipe de desenvolvimento deseje incorporar um componente de FOSS no projeto de software, deve-se encaminhar essa solicitação para o responsável pela implementação da política de software *open source* da organização (HADDAD, 2016; KEMP, 2009). Essa solicitação pode ser realizada por meio da criação de um *ticket* numa ferramenta de gerenciamento de tarefas e *bugs* (*issue tracking system*).

Dessa forma, pode-se autorizar ou recusar a utilização desse componente de FOSS solicitado, dependendo da conformidade da licença desse componente com a política de gerenciamento de FOSS da organização (HADDAD, 2016). Para realizar essa tarefa de inspeção das licenças de componente de software FOSS, é imprescindível utilizar ferramentas

de identificação automática de licenças, como a ferramenta Ninka[62], uma vez que determinar o número de licenças existentes em arquivos de código-fonte não é uma atividade trivial (GERMAN; MANABE; INOUE, 2010).

Dentre as dificuldades relacionadas à identificação dessas licenças de componentes FOSS, destacam-se a customização indevida do texto dos termos de uso da licença, seja alterando, retirando ou adicionando trechos indevidos, e a própria dificuldade em se localizar esse texto nos arquivos de código-fonte, uma vez que cada componente de software FOSS pode conter centenas de arquivos de código-fonte (GERMAN; MANABE;INOUE, 2010). As principais dificuldades relativas á identificação de licenças podem ser visualizadas no **Quadro 14**.

Quadro 14 - Maiores desafios para identificar licenças de componentes *open source*

Tipo de problema	Desafio para identificar licença
Encontrar a declaração da licença	O aviso de direitos autorais da licença geralmente está misturada com outros textos do código-fonte O aviso de direitos autorais da licença de um arquivo diz que sua licença está localizada em outro arquivo Os arquivos podem conter múltiplas licenças
Problema linguístico	O aviso de direitos autorais da licença pode conter erros gramaticais Uma dada licença é referenciada de formas diferentes O aviso de direitos autorais da licença pode ter sido modificado, alterando-se o texto imposto por determinada licença de *open source*
Customização da licença	Várias licenças devem ser customizadas quando são utilizadas Licenciantes modificam, adicionam ou removem condições de uso de licenças bem conhecidas Licenciantes modificam licenças *open source* por várias razões

Fonte: (GERMAN; MANABE; INOUE, 2010)

Além disso, deve-se inspecionar regularmente o repositório que armazena os códigos-fonte desenvolvidos pela própria organização, com o intuito de verificar se foram incluídos

[62] A utilização da ferramenta Ninka foi analisada na seção **5.6.1**.

equivocadamente componentes de software FOSS (HADDAD, 2016). Recomenda-se também que, ao estabelecer o sistema de controle de código fonte, busque-se separar, em pastas diferentes, os arquivos de código fonte *open source* e os arquivos de código fonte desenvolvidos pela própria organização (HADDAD, 2016).

2) **A implementação do produto deve somente incluir componentes de FOSS com licenças aprovadas:**

Somente quando o profissional responsável pela implementação da política de software *open source* da organização confirmar a aderência da licença do componente solicitado à política da organização, pode-se autorizar a incorporação desse componente de software FOOS no projeto de desenvolvimento de software (HADDAD, 2016). Nesse sentido, deve-se analisar também se todas as licenças de componentes de FOSS utilizadas no projeto de software são compatíveis entre si. Caso haja incompatibilidade de licenças, deve-se recusar a incorporação do componente de FOSS que gerou essa incompatibilidade. Em seguida, após a aprovação da utilização do componente de FOSS no projeto de software da organização, deve-se registrá-lo, no repositório de códigos fonte da organização, como componente dê software FOSS.

3)**Todas as obrigações contidas na licença do componente de FOSS devem ser identificadas e implementadas:**

Após aprovar a incorporação de componente de software FOSS no projeto de software, deve-se providenciar a documentação relativa às obrigações contidas na licença desse componente. De modo geral, essas obrigações consistem em a) reconhecer o uso de componentes de FOSS por meio de avisos de direitos autorais impostos por essas licenças. b) reproduzir o texto integral das condições de uso desses componentes de software *open source* no produto (HADDAD, 2016). Geralmente essas informações relativas aos direitos autorais e termos de uso de licenças *open source* são disponibilizadas, aos usuários do produto final, em interfaces gráficas intituladas "Licença", "Sobre" ou "Informações". Dessa forma, à medida que componentes de software FOSS são incorporados ao projeto de software, deve-se providenciar que os textos dos termos de uso e avisos de direitos autorais dessas licenças *open source* sejam incluídos na documentação do software que está sendo desenvolvido pela organização.

Além disso, as licenças recíprocas parciais, como a família LGPL e a Eclipse, estabelecem que o usuário final deve ter acesso ao código-fonte do componente *open source*

utilizado pelo software da organização (SABINO, 2011). Nesse sentido, recomenda-se que a organização disponibilize, em página na internet específica para este fim, links de acesso a esses componentes de software FOSS (HADDAD, 2016).

4) **Recomenda-se verificar mais uma vez todas as etapas anteriores antes do encerramento do projeto.**

Com o intuito de assegurar o cumprimento da política de gerenciamento de componentes de FOSS na organização, recomenda-se revisar regularmente as etapas descritas anteriormente, principalmente no que concerne à inspeção dos códigos fonte armazenados no repositório de artefatos da organização, sejam os componentes proprietários ou *open source*.

Por fim, sugere-se que, antes de implementar esses procedimentos, a organização identifique a estratégia mais condizente com as suas peculiaridades, principalmente no que tange ao seu processo de desenvolvimento de software (KEMP, 2009).

Uma vez analisadas os fatores que propiciam uma melhoria na gestão de uma Fábrica de Software, principalmente no que concerne à adoção de metodologias ágeis, gerenciamento ágil de projetos e reutilização de artefatos de software, passa-se a formular uma proposta de intervenção da FS-Facom.

6. Proposta de intervenção: Integrando a propriedade intelectual ao modelo de gestão da FS-Facom

As duas comissões de implantação da Fábrica de Software da Faculdade de Computação envidaram esforços para formular o processo padrão e operacional deste ambiente de treinamento em desenvolvimento de software (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2015c, 2016d).

Além disso, como resultado desses trabalhos, chegou-se a vislumbrar a possibilidade de realizar parcerias internas e externas: a primeira envolvendo interações no âmbito da UFMS e a segunda envolvendo parcerias com instituições públicas e privadas (FACULDADE DE COMPUTAÇÃO, 2015).

Nesse sentido, para que essas parcerias sejam concretizadas, em conformidade com o princípio da Legalidade que rege a Administração Pública, é necessário que a FS-Facom cumpra com as obrigações estabelecidas pelas normativas da UFMS referentes aos processos de PD&I.

Dessa forma, com o intuito de fornecer subsídios para a realização de parcerias entre a FS-Facom e instituições públicas e privadas, o presente Trabalho de Conclusão Final empreendeu uma análise sistemática das normativas da UFMS referentes aos processos de PD&I. Para alcançar esse objetivo,realizou-se uma análise qualitativa, por meio da construção de uma matriz conceitualmente agrupada (*Conceptually clustered matrix)* [MILES; HUBERMAN; SALDAÑA, 2014; MILES; HUBERMAN, 1994; NADIN; CASSELL, 2004).

6.1 Construção da matriz conceitualmente agrupada

As fases da construção de uma matriz conceitualmente agrupada são as seguintes: coleta de dados, codificação, identificação das linhas e colunas, preenchimento das unidades de análise (cruzamentos das linhas e colunas) e elaboração do texto analítico que resume e analisa os dados (MILES; HUBERMAN, 1994; NADIN; CASSELL, 2004).

O presente Trabalho de Conclusão Final, por meio de análise documental de normas internas da UFMS e da legislação federal sobre inovação, coletou e codificou conceitos e categorias pertinentes ao processo de pesquisa, desenvolvimento e inovação no âmbito da UFMS. Dessa forma, foram analisados a Lei de Inovação (Lei n° 10.973/2004) e as seguintes normas da UFMS: Instrução Normativa n° 1/2016 da Pró-Reitoria de Planejamento e Orçamento, que normatiza os procedimentos para a formalização, celebração e execução dos Convênios e Congêneres; a Instrução Normativa n°2/2016 da Pró-Reitoria de Pós-Graduação

e Pesquisa, que normatiza procedimentos para cadastro, submissão, análise e vigência de projetos de pesquisa coordenados por pesquisadores vinculados à UFMS; a Resolução n° 132/2015 do Conselho Diretor, que regulamenta relações da UFMS com Fundações de Apoio; e a Resolução n° 133/2017 do Conselho Diretor, que regulamenta a concessão de bolsas no âmbito da UFMS.

Após análise preliminar dessas normativas, decidiu-se que as linhas da matriz fossem formadas pelos criadores dos softwares desenvolvidos na Fábrica de Software da Facom: alunos matriculados nas disciplinas obrigatórias Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II, monitores de graduação dessas disciplinas obrigatórias, professores (sejam eles efetivos, substitutos e voluntários), técnicos e analistas da carreira de Tecnologia da Informação, e participantes externos sem vínculo com a UFMS (aqueles vinculados a eventuais parceiros e aqueles especificamente contratados para o projeto de pesquisa).

Estabeleceu-se também que as colunas da matriz fossem compostas pelos diferentes tipos de parceria de projetos de pesquisa previstos pelas normas da UFMS: Projetos de pesquisa sem fomento, projetos de pesquisa com fomento externo (financiado por agências de fomento como CAPES, CNPq e Fundect), projetos de pesquisa financiados com auxílio da Fundação de Apoio à Pesquisa ao Ensino e à Cultura (FAPEC), Projetos de pesquisa financiados por empresas privadas e organizações públicas.

Dessa forma, uma vez definidas as linhas e colunas da matriz conceitualmente agrupada, buscou-se preencher as unidades de análise com os documentos e informações necessários para a participação dos diferentes atores da FS-FACOM em projetos de pesquisa, com o intuito de facilitar o gerenciamento do processo de PD&I no âmbito da Fábrica de Software, conforme o **Quadro 15**.

Dentre as informações identificadas nas unidades de análise da matriz, cabe destacar o chamado “termo de compromisso*”, que é um documento que visa a oficializar a cessão dos direitos patrimoniais do software desenvolvido no projeto de PD&I para a universidade e o parceiro que financiam o projeto, conforme apontado pelo Acórdão do TCU nº5770/2014 e por Pimentel et al. (2010) como boa prática. Dessa forma, o presente TCF pretende sugerir que os criadores de software da FS-FACOM

Quadro 15: Matriz conceitualmente agrupada dos tipos de parceria possíveis para os Projetos de Pesquisa na UFMS

Vinculado com a UFMS	Criadores	UFMS sem fomento	UFMS com fomento (CNPq, CAPES, Fundect)	Fundação de Apoio	Organização Pública	Empresa
Alunos	**Monitores**	RGA do aluno Termo de Compromisso*	RGA do aluno Termo de Compromisso* Termo de Concessão Art. 18 Res. CD 133/2017	RGA do aluno Termo de Compromisso* Termo de Concessão Art. 26 Res CD 132/2015	RGA do aluno Termo de Compromisso* Termo de Concessão Art. 26 Res CD 132/2015	RGA do aluno Termo de Compromisso* Termo de Concessão Art. 26 Res CD 132/2015
	Alunos matriculados	Matrícula na Disciplina Termo de Compromisso*	Matrícula na Disciplina Termo de Compromisso* Termo de Concessão Art. 18 Res CD 133/2017	Matrícula na Disciplina Termo de Compromisso* Termo de Concessão Art. 26 Res CD 132/2015	Matrícula na Disciplina Termo de Compromisso* Termo de Concessão Art. 26 Res CD 132/2015	Matrícula na Disciplina Termo de Compromisso* Termo de Concessão Art. 26 Res CD 132/2015
Professores	**Efetivos**	Matricula SIAPE Termo de compromisso*	Matricula SIAPE Termo de compromisso* Plano de Trabalho Art. 81 Resolução CD 133/2017 veda pagamento de bolsas para cumprimento de atividades regulares de magistério de graduação	Matricula SIAPE Termo de compromisso* Plano de Trabalho Art. 29 II Res CD 132/2015 veda pagamento de bolsas de ensino para o cumprimento de atividades regulares de magistério de graduação	Matricula SIAPE Termo de compromisso* Plano de Trabalho Art. 29 II Res CD 132/2015 veda pagamento de bolsas de ensino para o cumprimento de atividades regulares de magistério de graduação	Matricula SIAPE Termo de compromisso* Plano de Trabalho Art. 29 II Res CD 132/2015 veda pagamento de bolsas de ensino para o cumprimento de atividades regulares de magistério de graduação
	Substitutos	Contrato Temporário Termo de compromisso*	Contrato Temporário Termo de compromisso* Art. 81 IV Res CD 133/2017 veda pagamento de bolsas cumprimento de atividades regulares de magistério de graduação	Contrato Temporário Termo de compromisso* Art. 29 II Res CD 132/2015 veda pagamento de bolsas de ensino para o cumprimento de atividades regulares de magistério de graduação	Contrato Temporário Termo de compromisso* Art. 29 II Res CD 132/2015 veda pagamento de bolsas de ensino para o cumprimento de atividades regulares de magistério de graduação	Contrato Temporário Termo de compromisso* Art. 29 II Res CD 132/2015 veda pagamento de bolsas de ensino para o cumprimento de atividades regulares de magistério de graduação
	Voluntários	Termo de Adesão (Resolução CD 10/1998) Termo de compromisso*	Termo de Adesão (Res CD 10/1998) Termo de compromisso* Termo de Concessão	Termo de Adesão (Res CD 10/1998) Termo de compromisso* Termo de Concessão Art. 26 Res CD 132/2015	Termo de Adesão (Res CD 10/1998) Termo de compromisso* Termo de Concessão Art. 26 Res CD 132/2015	Termo de Adesão (Res CD 10/1998) Termo de compromisso* Termo de Concessão Art. 26 Res CD 132/2015

Continuação do **Quadro 15**

Vinculado com a UFMS **Técnicos-Administrativos**	**Técnico (Classe D) e Analista (Classe E) de Tecnologia da Informação)**	Matricula SIAPE Termo de compromisso*	Matricula SIAPE Termo de compromisso* Art. 81 VI Resolução CD n° 133/2017 veda pagamento de bolsas aos servidores técnico-administrativos pelo desempenho de atividades administrativas inerentes ao cargo	Matricula SIAPE Termo de compromisso* Art. 29 Resolução CD n° 132/2015 veda pagamento de bolsas por atividades que caracterizem contraprestação de serviços nos projetos para servidores da área-meio da Universidade para desenvolver atividades de sua atribuição regular, mesmo que fora de horário de trabalho	Matricula SIAPE Termo de compromisso* Art. 29 Resolução CD n° 132/2015 veda pagamento de bolsas por atividades que caracterizem contraprestação de serviços nos projetos para servidores da área-meio da Universidade para desenvolver atividades de sua atribuição regular, mesmo que fora de horário de trabalho	Matricula SIAPE Termo de compromisso* Art. 29 Resolução CD n° 132/2015 veda pagamento de bolsas por atividades que caracterizem contraprestação de serviços nos projetos para servidores da área-meio da Universidade para desenvolver atividades de sua atribuição regular, mesmo que fora de horário de trabalho
Participantes Externos (Sem Vínculo com a UFMS)	**Selecionados do Projeto**	Não se aplica	Contrato Termo de compromisso*	Termo de compromisso* Contrato de prestação de serviços por pessoas físicas (Art. 81 parágrafo único da Resolução CD n° 132/2015)	Termo de compromisso* Contrato de prestação de serviços por pessoas físicas (Art. 81 parágrafo único da Resolução CD n° 132/2015)	Termo de compromisso* Contrato de prestação de serviços por pessoas físicas (Art. 81 parágrafo único da Resolução CD n° 132/2015)
	Vinculados aos parceiros	Não se aplica	Contrato com parceiro Termo de compromisso*	Contrato com parceiro Termo de compromisso*	Contrato com parceiro Termo de compromisso*	Contrato com parceiro Termo de compromisso*

Termo de compromisso*:Esse item da matriz conceitualmente agrupada será analisado com mais detalhe *a posteriori* na seção**6.2.2**. Essa decisão foi tomada devido à importância de um documento que formalize a cessão da titularidade dos direitos patrimoniais. Cabe mencionar que, devido à inexistência da previsão desse documento nas normativas da UFMS, o presente TCF formulou um modelo de termo de compromisso que assegure a cessão direitos patrimoniais para a UFMS do software desenvolvido na FS-Facom.

Quadro 16- Texto analítico da Matriz conceitualmente agrupada

Análise por Coluna	**Análise por criador**	***Insights* da Matriz**
Os projetos de pesquisa realizados na UFMS, segundo a Instrução Normativa n° 2/2016 da Pró-Reitoria de Pós-Graduação e Pesquisa, devem ser classificados de duas formas na plataforma do Sistema de Gestão de Projetos (SIGProj): projetos sem fomento ou projetos com fomento externo. O projeto de pesquisa sem fomento não conta com nenhum tipo de financiamento (recursos e bolsas). Já o projeto de pesquisa com fomento externo conta com recursos financeiros provenientes de órgãos oficiais de fomento, como a CAPES, CNPq e Fundect, ou recursos oriundos de convênios, contratos, acordos de parceria ou similares, envolvendo a UFMS em conjunto com um ou mais participes, firmados por meio de instrumento jurídico. No projeto de pesquisa com fomento externo, deve-se cadastrar também no SigProj o Termo de outorga/concessão ou instrumento similar, firmado entre os pesquisadores e o órgão de fomento envolvido. Além disso, em ambos os tipos de projeto de pesquisa, deve ser providenciado Termo de conhecimento ou Parecer do setor responsável pela propriedade intelectual na UFMS, caso o projeto tenha potencial de gerar nova tecnologia, como um programa de computador. Os projetos de pesquisa desenvolvidos na UFMS, segundo a Resolução n° 132/2015 do Conselho Diretor, também podem contar com auxílio da Fapec (Fundação de Apoio à Pesquisa, ao Ensino e à Cultura). A Fapec poderá captar recursos, por meio de convênios, acordos ou contratos, junto a organizações públicas e privadas, com o intuito de proporcionar recursos financeiros aos projetos de pesquisa da UFMS, como a concessão de bolsas. Toda bolsa ou retribuição pecuniária só será concedida após assinatura do termo de concessão por pessoa vinculada à UFMS, a qual aceitará cumprir as metas estabelecidas no Plano de Trabalho do projeto. Os projetos apoiados pela Fapec devem contar com no mínimo dois terços de pessoas vinculadas à UFMS, incluindo docentes, servidores técnico-administrativos, estudantes regulares e bolsistas. Contudo, quando o projeto demandar a contribuição de pessoas não vinculadas à UFMS, será necessário realizar contrato de prestação de serviço de pessoa física, respeitando-se o limite de um terço para a participação de pessoas não vinculadas ao projeto.	Os projetos de pesquisa da UFMS podem ser realizados por professores (efetivos, substitutos e voluntários), alunos (monitores e alunos de graduação matriculados nas disciplinas Prática em Desenvolvimento de Software I e II), técnicos-administrativos (técnico e analista de Tecnologia da Informação) e participantes externos (vinculados ao parceiro do projeto e selecionados especificamente para o projeto). Com relação à participação de alunos em projetos de pesquisa sem fomento, apenas a matrícula nas disciplinas PDS I/II e a assinatura do termo de compromisso* cedendo os direitos patrimoniais são necessárias para comprovar o vínculo do aluno com o projeto. Quando o projeto de pesquisa contar com fomento externo, além dos itens supracitados, é necessário que o aluno preencha o Termo de Concessão para receber a bolsa, conforme o Art. 18 da Resolução 133/2017 do Conselho Diretor, que regulamenta a concessão de bolsas na UFMS. Já os projetos de pesquisa que contarem com auxílio da Fapec também necessitam do preenchimento do Termo de Concessão para receber a bolsa, segundo o Art. 26 da Resolução 132/2015, que regulamenta a relação da UFMS com as fundações de apoio. Com relação à participação de docentes em projeto de pesquisa sem fomento, para comprovar seu vínculo com a UFMS, é necessário que o professor apresente sua matrícula SIAPE (professor efetivo), contrato temporário (professor substituto) ou termo de adesão (professor voluntário), bem como o termo de compromisso* cedendo os direitos patrimoniais. Contudo, no que tange aos projetos de pesquisa com fomento externo, é vedada a concessão de bolsas para o cumprimento de atividades regulares de magistério de graduação, no âmbito da UFMS, segundo o Art. 81 da Resolução n° 133/2017. A mesma resolução também veda a concessão de bolsa aos servidores técnicos-administrativos pelo desempenho de atividades administrativas inerentes ao cargo. Com relação aos projetos de pesquisa amparados pela Fapec, é vedada a concessão de bolsas de ensino para o cumprimento de atividades regulares de magistério de graduação, segundo o Art. 29 da Resolução CD 132/2015. No que tange aos participantes externos nos projetos de pesquisa amparados pela Fapec, em geral, sua quantidade não deve ultrapassar 1/3 da equipe do projeto. Contudo, é possível ultrapassar esse percentual, desde que haja autorização do Conselho Diretor, segundo os incisos 3º e 4° do Art. 19 da Resolução CD 132/2015.	Segundo a Lei de Inovação (Lei n° 10.973/2004), as bolsas de projetos de pesquisa recebidas por servidores e alunos de ICTs devem ser pagas pela própria ICT, Fundação de Apoio ou Agência de Fomento, excluindo-se assim pagamento direto de parceiros públicos ou privados a pesquisadores vinculados às ICTs. A celebração de convênios, acordos ou contratos firmados entre UFMS e organizações públicas e provadas, que resultem em projetos de pesquisa, devem contar com um coordenador do projeto de pesquisa e um gestor do instrumento jurídico, segundo a Instrução Normativa n° 1/2016 da Pró-Reitoria de Planejamento e Orçamento. Constatou-se a importância do Plano de Trabalho para a devida aprovação e execução dos projetos de pesquisa, o qual deverá indicar, entre outras coisas, se há no projeto potencial de criação de objeto sujeito à proteção de propriedade intelectual, segundo o Art. 15 da Resolução CD 132/2015. No caso específico da FS-FACOM, constatou-se que os professores responsáveis pelas disciplinas PDS I/II não poderão receber bolsas devido às vedações contidas nas resoluções CD n° 132/2015 e n° 133/2017.

assinem um Termo de Compromisso com o intuito de oferecer segurança jurídica para a UFMS acerca da cessão dos direitos patrimoniais desse bem imaterial. Com efeito, a preocupação com a questão do termo de compromisso parece ser pertinente, uma vez que não foi identificado, nas normas da UFMS, dispositivo que explicitamente trate da cessão da propriedade intelectual das inovações e criações desenvolvidas na UFMS.

Com relação ao texto analítico da matriz conceitualmente agrupada, verificou-se que um dos principais *insights* ensejados pela matriz é que os professores e técnicos da carreira de Tecnologia da Informação da Facom não poderão receber bolsas pelas atividades desenvolvidas na Fábrica de Software, segundo as normativas que regulamentam a concessão de bolsas (Resolução CD n° 133/2017) e o relacionamento da UFMS com fundações de apoio (Resolução CD n° 132/2015). Como as atividades da FS-FACOM serão desenvolvidas no âmbito de duas disciplinas obrigatórias, o trabalho desempenhado por aqueles profissionais ocorrerá necessariamente no exercício de suas respectivas atribuições funcionais regulares, fazendo com que a concessão de bolsas para esse tipo de atividade seja expressamente vedada pelas resoluções supracitadas (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2015, 2017).

Outro *insight* apontado pelo texto analítico da matriz é a exigência de que cada projeto de pesquisa desenvolvido na FS-FACOM conte com as figuras do coordenador do projeto e do gestor do acordo entre a UFMS e o parceiro. Segundo a Instrução Normativa n° 1/2016 da Proplan, as atribuições do gestor do acordo de parceria tem a atribuição de acompanhar toda a execução do acordo, adotando as medidas administrativas necessárias ao fiel cumprimento das obrigações do acordo. Já o coordenador do projeto é o responsável pelos aspectos técnico-cientificos do objeto do acordo de parceria.

O texto analítico da matriz (**Quadro 16**) também destaca a necessidade do plano de trabalho, como condição fundamental, para execução dos projetos de pesquisa. Segundo o Art. 15 da Resolução CD n° 132/2015, o plano de trabalho deve conter:

> I – objeto, projeto básico (quando for o caso), prazo de execução limitado no tempo, bem como os resultados esperados, metas e respectivos indicadores;
> II – os recursos da UFMS, com ressarcimento pertinentes, nos termos do art. 6º da Lei nº 8.958/1994 quando for o caso;
> III – os participantes vinculados à UFMS e autorizados a participar do projeto, identificados por registro funcional, na hipótese de docentes ou servidores técnicoadministrativos, sendo informados os valores das bolsas a serem concedidas;
> IV – apresentar o valor das despesas com a contratação de pessoas físicas e jurídicas prestadoras de serviços, identificados pelos números de CPF ou CNPJ, conforme o caso, condição e situação;
> V – especificar o processo de divulgação e publicação dos resultados, quando não houver restrição justificada; e
> VI – indicar se há potencial de proteção da propriedade intelectual, a partir dos resultados obtidos no decorrer do projeto (UFMS, 2015).

Ao se analisar o dispositivo supracitado, observa-se que os itens III e IV já foram identificados como relevantes pela matriz conceitualmente agrupada (**Quadro 14**), de modo que não é necessário aprofundar o tema. Contudo, faz-se necessário comentar os aspectos relevantes dos itens I,V e VI do artigo supracitado.

Verifica-se que os itens V e VI estão correlacionados, uma vez que é preciso ter cuidado para evitar a publicação dos resultados do projeto de pesquisa pertinentes a criações/inovações que serão protegidos por algum regime de propriedade intelectual (PIMENTEL et al., 2010). Dessa forma, a equipe do projeto de pesquisa (principalmente o coordenador) precisa estar atenta para a questão da confidencialidade dos resultados que serão objeto de proteção de propriedade intelectual.

No que concerne ao item I do Art. 15 da Resolução CD 132/2015, que trata da questão do prazo de execução e metas do projeto de pesquisa, é necessário analisá-lo à luz das especificidades dos projetos que serão desenvolvidos pela equipe da Fábrica de Software da Facom, principalmente no que tange à utilização de metodologias ágeis.

Nesse sentido, com o intuito de subsidiar a estimação de tempo, escopo e metas para a elaboração de plano de trabalho para os acordos de parceria envolvendo a FS-Facom e instituições públicas e privadas, sugere-se adotar ferramentas de gerenciamento ágil de projetos, como o IVPM2.

Uma vez analisada as normativas da UFMS referentes ao processo de PD&I, por meio da matriz conceitualmente agrupada, passa-se a discorrer sobre as contribuições do presente Trabalho de Conclusão Final no que tange à adoção de boas práticas para a propriedade intelectual na FS-Facom, com o intuito de construir um modelo de gestão de propriedade intelectual.

6.2. Contribuições para Fábrica de Software da Facom/UFMS

Com o intuito de atender ao objetivo principal do presente trabalho, apresentar-se-á uma proposta de gestão de propriedade intelectual que leve em conta as particularidades da Fábrica de Software da Facom/UFMS.

Dessa forma, esse conjunto de ações e práticas propostas busca fazer com que os projetos de desenvolvimento de software da FS-Facom sejam realizados em conformidade com as normativas de propriedade intelectual, reduzindo o volume de consultas e sistematizando as informações pertinentes aos processos de PD&I, contribuindo assim para a maior efetividade da FS-Facom.

6.2.1 Mudanças no Projeto Pedagógico do Curso de Engenharia de Software

A primeira contribuição da presente pesquisa para a gestão da propriedade intelectual da FS-Facom é a sugestão de alterar a ementa de algumas disciplinas no projeto pedagógico do Curso de Engenharia de Software. Sugere-se que se inclua na ementa da disciplina "Prática em Desenvolvimento de Software I" os temas "Noções de Propriedade Intelectual de software" e "Licenças de software livre e de código aberto".

Embora fosse mais apropriado para o aluno ter acesso a estes assuntos nos semestres anteriores às atividades que serão desenvolvidas na FS-Facom, chegou-se à conclusão de que isso seria de difícil aplicação no contexto do Curso de Engenharia de Software, uma vez que a maioria das disciplinas deste curso é ofertada em conjunto com outros cursos da Faculdade de Computação. Dessa forma, como a disciplina "Prática em Desenvolvimento de Software I" é ofertada somente pelo Curso de Engenharia de Software, pode-se incluir mais facilmente aqueles temas no Projeto Pedagógico do Curso, sem depender de alterações dos Projetos Pedagógicos de outros cursos.

6.2.2 Termo de Compromisso

A segunda contribuição da presente pesquisa foi a elaboração de um Termo de compromisso[63] no qual os alunos matriculados nas disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II se comprometem em ceder à UFMS os direitos patrimoniais do software desenvolvido na FS-FACOM. Nesse sentido, o Art. 10 do Regulamento das disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II estabelece como obrigatório o preenchimento do Termo de Compromisso para efetivar a matrícula dos alunos naquelas disciplinas (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2017).

Para fins de validação, cabe destacar que o texto deste termo de compromisso foi revisado pelo Colegiado do Curso de Engenharia de Software, Conselho de Faculdade da Facom e pelo encarregado do Núcleo de Inovação Tecnológica da UFMS. Além disso, na redação da seção do termo de compromisso referente à propriedade intelectual, visando contribuir para a formação dos alunos nesse tema, buscou-se evidenciar a questão dos direitos morais e patrimoniais do programa de computador, previstas na Lei n° 9609/1998, bem como a questão da garantia aos criadores dos ganhos econômicos resultantes da inovação, previstos na Lei de Inovação:

[63] Pode-se ver este Termo de compromisso no Anexo II deste TCF.

> DA PROPRIEDADE INTELECTUAL
>
> Considerando o disposto no §3º do Art. 4° da Lei nº 9.609, de 19/02/98 e as normativas da UFMS referentes à propriedade intelectual, os acadêmicos matriculados nas disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II, doravante denominados autores, cederão à UFMS todos os **direitos patrimoniais** relativos ao(s) Programa(s) de Computador(es) desenvolvido(s) na disciplina Prática em Desenvolvimento de Software ___, no _____ semestre de _____.
>
> Serão garantidos aos alunos autores os **direitos morais** previstos no §1º do Art. 2º da Lei nº 9.609, de 19/02/98; bem como a **participação nos ganhos econômicos**, auferidos pela UFMS, resultantes de contratos de transferência de tecnologia e de licenciamento para outorga de direito de uso ou de exploração de **criação desenvolvida pelos alunos autores**, conforme o Art. 13 da Lei n° 10.973, de 2/12/2004.
>
> A distribuição dos ganhos econômicos assegurados aos alunos autores será repartida de acordo com a contribuição intelectual dos criadores do produto final (UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL, 2017, grifo nosso).

Dessa forma, a assinatura do termo de compromisso vai oferecer maior segurança jurídica para a UFMS e eventuais parceiros no momento do registro desse software junto ao Instituto Nacional de Propriedade Industrial, uma vez que esse termo de compromisso ensejará a cessão da titularidade dos direitos patrimoniais do software desenvolvido na FS-Facom para a UFMS.

6.2.3 Organização do repositório de artefatos

A terceira proposta para a política de gestão de propriedade intelectual da FS-Facom reside na organização do repositório de artefatos. Com efeito, à medida que os alunos matriculados nas disciplinas PDS I e PDS II desenvolverem artefatos de software, principalmente o código fonte, será necessário organizar o repositório desses artefatos. Essa organização do repositório deve visar não somente a reutilização desses artefatos de software, mas também deve buscar identificar eficazmente a autoria desses artefatos. Dessa forma, ao reutilizar os artefatos de software disponíveis em seu repositório interno, no final de cada projeto de desenvolvimento de software, a FS-Facom deve considerar também, como autores de um dado projeto de software, aqueles que desenvolveram os artefatos que foram reutilizados.

6.2.4 Gerenciamento de componentes de software livre e de código aberto

A quarta proposta para o gerenciamento de propriedade intelectual na Fábrica de Software da Facom/UFMS é o estabelecimento de uma política de gerenciamento de componentes de software *open source.* Conforme visto na seção **4.3.1**, deve-se designar um profissional responsável por implementar essa política na organização. No caso da FS-Facom, sugere-se que o professor supervisor ou o professor consultor sejam os responsáveis por autorizar a incorporação de componentes de software FOSS (*Free and Open Source Software*) nos projetos de software. Além disso, sugere-se que seja imputada à Comissão Permanente da Fábrica de Software a tarefa de inspecionar regular e sistematicamente todo o repositório de artefatos da FS-Facom, com o intuito de verificar se existe algum componente de FOSS não autorizado.

Dessa forma, considera-se que a Comissão Permanente da Fábrica de Software seja a instância mais adequada para monitorar se o repositório de artefatos da FS-Facom está em conformidade com as normativas de propriedade intelectual, inclusive as diversas licenças *open source.*

Além disso, no que se refere à reutilização de componentes de software *open source,* sugere-se que os projetos de software da FS-Facom apenas utilizem aqueles componentes licenciados sob licenças permissivas, como a BSD, MIT e Apache. Essa decisão se justifica porque, conforme analisado na seção **3.4**, a reutilização de software sob licenças permissivas apresenta menor risco legal no que concerne à incompatibilidade de licenças.

6.2.5 Método IVPM2

A quinta contribuição desta pesquisa para a FS-Facom consiste na adoção do método IVPM2 (total ou parcialmente) para auxiliar o planejamento e a execução dos projetos de desenvolvimento de software. Embora o método IVPM2 não se destine ao gerenciamento da propriedade intelectual propriamente dito, esse método pode ser muito útil para elaborar o plano de trabalho, item fundamental para a celebração de acordos de parceria com instituições públicas ou privadas, conforme apontado pela matriz conceitualmente agrupada (**Quadro 14**).

Dessa forma, sugere-se que a Comissão Permanente da Fábrica de Software, os professores supervisores e consultores e alunos apliquem, total ou parcialmente, as etapas e ferramentas previstos no *Iterative Visual Project Management Method*, conforme apontado na seção **5.5**. Nesse sentido, recomenda-se priorizar a utilização dos itens "Modelo de Fases e Entregas" e o "Sistema de Gestão de Projetos", com o intuito de integrar

os diversos ciclos iterativos do projeto de desenvolvimento de software ao planejamento geral do mesmo baseado em etapas e metas a serem cumpridas, que são elementos cruciais para a formulação de um plano de trabalho.

Com base nas contribuições supracitadas, apresentar-se-á uma proposta de modelo de gestão de propriedade intelectual para a FS-Facom que visa não somente à melhoria contínua dessas práticas, mas também assegura a definição da autoria e da titularidade do software, de modo que as práticas do processo de desenvolvimento de software da FS-Facom estejam em conformidade com as normativas pertinentes à propriedade intelectual de programa de computador.

6.3.Gestão de Propriedade Intelectual na FS-Facom

A estrutura do modelo de gestão de propriedade intelectual para a Fábrica de Software da Facom se baseia na abordagem sistêmica das organizações (BERTALANFFY, 1975, 1972; KAST; ROSENZWEIG, 1972). Esse modelo é formado por entradas, saídas e processo.

Na teoria sistêmica, a **entrada (input)** é o insumo que fornece material, recursos e informação para a operação do sistema. O **processo** é o mecanismo formado por uma série de interações pelas quais as entradas se transformam em saídas. A **saída** (output) é o resultado das interações ocorridas durante o processo.

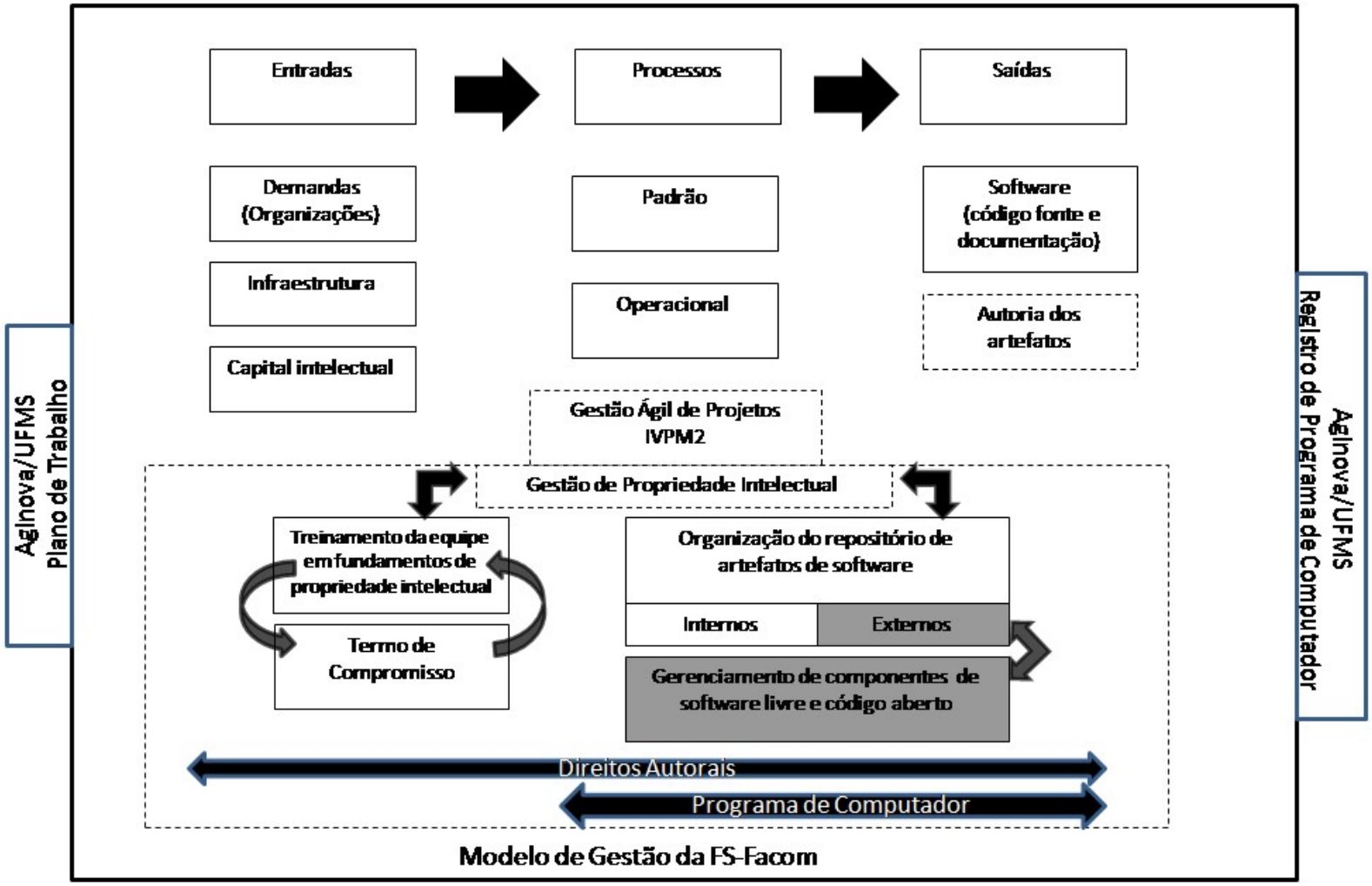
O modelo de gestão de propriedade intelectual proposto pelo presente TCF está inserido num sistema maior, a saber, o modelo de gestão de da Fábrica de Software da Faculdade de Computação. Esse modelo tem como entradas os seguintes elementos: as demandas oriundas de parcerias internas ou externas, a infraestrutura de equipamentos da FS-Facom e o capital intelectual dos professores, alunos e técnicos-administrativos da área de Tecnologia da Informação.

No que concerne ao processo, o modelo de gestão da FS-Facom é composto pelos processos padrão e operacional, que foram elaborados pelas duas comissões de implantação. Figura-se como saída desse modelo, o software desenvolvido no decorrer das atividades da FS-Facom, formado pelo código fonte e sua documentação associada.

Subordinado a esse sistema, encontra-se o modelo de gestão de propriedade intelectual de software proposto pelo presente TCF. Na **Figura 10**, verifica-se que a interface entre modelos está situada no processo, mais especificamente no *gerenciamento ágil de projetos*.

No modelo de gestão de propriedade intelectual para a FS-Facom, as **entradas** são compostas por dois elementos correlacionados, a saber, o treinamento em fundamentos de propriedade intelectual e o *termo de compromisso.*

Figura 10 - Modelo de Gestão de Propriedade Intelectual para a FS-Facom



Fonte: Elaborado pelo autor.

Dessa forma, o *termo de compromisso*, que os alunos deverão assinar ao se matricular nas disciplinas, será fundamental para a conformidade com a legislação pertinente do processo de PD&I que será desenvolvido na FS-Facom. Dessa forma, esse documento oferecerá segurança jurídica para a UFMS e eventuais parceiros no que tange à titularidade dos direitos patrimoniais do software desenvolvido na FS-Facom.

O processo do modelo proposto para a gestão de propriedade intelectual da Fábrica de Software da Facom é composto por duas dimensões correlacionadas, que são fundamentadas respectivamente pela literatura relacionada ao gerenciamento ágil de projetos (CONFORTO et al., 2011; CONFORTO, 2009) e pela literatura referente ao gerenciamento e reutilização de *artefatos internos*(BAUER, 2016; BAUER; VETRÓ, 2016; OLIVEIRA, 2015) e *artefatos externos* – conhecidos como software livre e de código aberto (HADDAD, 2016, KEMP, 2009).

Em termos práticos, a articulação entre essas duas dimensões do modelo de gestão de propriedade intelectual da FS-Facom pode ser intermediada pela ferramenta *Redmine*[64]. Optou-se em recomendar essa ferramenta, principalmente, porque ela é considerada tanto como uma ferramenta de gerenciamento de projetos[65] quanto uma ferramenta de gerenciamento de tarefas e *bugs* (*issue tracking system*)[66]. Além disso, a ferramenta *Redmine* é distribuída sob a licença GPLv2, o que reduz o custo para aquisição dessa ferramenta. Outra importante funcionalidade da ferramenta *Redmine* é a possibilidade de integração com sistemas de controle de versões de código fonte (como Git e SVN*),* o que potencializa as atividades relacionadas à organização e gerenciamento do repositório de artefatos de software.

No que tange à primeira dimensão do processo do modelo de gestão de propriedade intelectual da FS-Facom, a dimensão do gerenciamento ágil de projetos, deve-se primeiramente elaborar um plano de trabalho com metas e prazos, baseado no Modelo de Fases e Entregas (MFE) do método IVPM2, que visa a integrar as tarefas e entregas de cada ciclo iterativo num planejamento macro (CONFORTO et al., 2011; CONFORTO, 2009).

Em seguida, deve-se atribuir na ferramenta adotada para a gestão ágil de projeto as seguintes informações para cada membro da equipe de desenvolvimento de software: tarefas,

---

[64] www.redmine.org

[65] A ferramenta *Redmine* possui a maioria das funcionalidades exigidas por um software de gestão de projetos segundo essa lista comparativa da *Wikipedia* disponível em: https://en.wikipedia.org/wiki/Comparison_of_project_management_software

[66] A ferramenta *Redmine* possui a maioria das funcionalidades exigidas por um software de gerenciamento de tarefas e *bugs*segundo essa lista comparativa da *Wikipedia* disponível em: https://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems

entregas e horas trabalhadas, com o intuito de controlar e monitorar sua evolução, bem como atualizar regularmente a situação desses itens.

Concomitante à primeira dimensão – principalmente na tarefa de atualização do *status* do projeto de software – figura-se a segunda dimensão do modelo do modelo de gestão de propriedade intelectual da FS-Facom, a dimensão da organização e gerenciamento do repositório de artefatos de software.

Nessa dimensão, devem-se realizar as atividades de produção e consumo referentes à reutilização de artefatos, conforme analisado na seção **5.6** (BAUER, 2016; BAUER; VETRÓ, 2016; OLIVEIRA, 2015), utilizando a ferramenta de gerenciamento de tarefas e *bugs* e sua interface com o sistema de controle de versão de código fonte. Durante esse processo é fundamental prezar pela associação dos artefatos de software com seus respectivos autores.

No que tange especificamente à reutilização de artefatos de software externo – aqueles licenciados sob licenças de software livre e de código aberto, devem-se utilizar os procedimentos de gerenciamento para incorporação de componentes de software FOSS, conforme analisado na seção **5.6.2** (HADDAD, 2016).

No caso da FS-Facom, sugere-se que seus projetos de software devem somente reutilizar componentes de software licenciados sob licenças permissivas (BSD, MIT e Apache), com o intuito de dirimir riscos relacionados à incompatibilidade de licenças e eventual obrigação de abrir o código fonte (no caso de reutilizar software licenciado sob a GPL). Nesse sentido, deve-se realizar a inspeção regular de todo o repositório de artefatos da FS-Facom, utilizando alguma ferramenta de identificação de licenças FOSS, como a *Ninka*[67], com o objetivo de identificar componentes de software com licenças não autorizadas.

Uma vez concluídos os processos de transformação que fazem uso das dimensões do *gerenciamento ágil de projetos* e da *reutilização de artefatos internos e externos de software*, passa-se a analisar a **saída** do modelo proposto para a Fábrica de Software da Facom.

Uma **saída** do modelo de gestão da propriedade intelectual da FS-Facom é o software desenvolvido nas atividades das disciplinas Prática em Desenvolvimento de Software I Prática em Desenvolvimento de Software II, formado por seu código fonte e documentação associada. Outra **saída** do modelo proposto é a mensuração da contribuição individual de cada autor no software desenvolvido. Nesse sentido, deve-se utilizar a ferramenta Redmine e sua interface com o sistema de controle de versão de código fonte para auxiliar nessa tarefa de mensurar a participação dos autores no projeto de software. Dessa forma, essas saídas são fundamentais para que se possa realizar o registro desse software junto ao INPI, respeitando-se às normas

[67] https://github.com/dmgerman/ninka

relacionadas à propriedade intelectual de programa de computador e aquelas relacionadas ao processo de PD&I no âmbito de instituições federais de ensino superior, principalmente ao disposto no Art. 13 da Lei de Inovação, que trata da remuneração a que tem direito os autores de criações.

7. Considerações finais

As fábricas de software são espaços organizacionais tradicionalmente associados ao desenvolvimento de programas de computador com qualidade e produtividade. Instituições de Ensino Superior têm buscado implementar fábricas de software acadêmicas, com o intuito de aproximar o conteúdo teórico aprendido em sala de aula com a realidade exigida pelo mercado de trabalho. Sendo que o software é um produto resultante do investimento de capital intelectual, tanto de professores quanto de alunos e técnicos, os gestores das fábricas de software devem realizar a gestão da propriedade intelectual desse bem imaterial, de modo a garantir a reutilização de artefatos com licenças de uso adequadas e ainda, evitar eventuais passivos legais provenientes da não observância de direitos pertinentes à equipe desenvolvedora.

Dessa forma, o Presente Trabalho de Conclusão Final empreendeu uma investigação de caráter qualitativa e exploratória, com o intuito de alcançar o objetivo geral de propor um modelo de gestão de propriedade intelectual para a Fábrica de Software da Facom/UFMS.

Para alcançar esse objetivo geral, buscou-se responder a três objetivos específicos: a) compreender e sistematizar as normativas referentes aos processos de PD&I, com ênfase naquelas normativas vigentes no âmbito da UFMS, frente à possibilidade de firmar parcerias externas; b) realizar levantamento de boas práticas no que concerne à gestão da propriedade intelectual em Fábricas de Software; e c) propor mecanismo de integração entre a gestão da propriedade intelectual e os processos de PD&I no âmbito de Instituições Federais de Ensino Superior.

Com o intuito de alcançar o primeiro objetivo específico, esta pesquisa realizou uma análise sistemática das normativas da UFMS aplicáveis ao processo de PD&I, por meio da construção de uma matriz conceitualmente agrupada, que sistematizou procedimentos e práticas, bem como gerou *insights* sobre as práticas necessárias para a consecução de parcerias com instituições públicas e privadas. Dentre estas práticas, destaca-se a elaboração de um plano de trabalho para cada projeto de software. Acredita-se que essa sistematização de normativas referentes ao processo de PD&I será uma ferramenta útil para os membros da FS-Facom, uma vez que os mesmos poderão verificar quais procedimentos e documentos são necessários para a formalização de parcerias e projetos de pesquisa.

No que concerne ao segundo objetivo secundário, este TCF apresentou uma série de boas práticas ao empreender uma revisão da literatura sobre a propriedade intelectual de

programa de computador, fábrica de software, gerenciamento ágil de projetos e reutilização de artefatos de software.

Fundamentado na análise sistemática e na revisão bibliográfica supracitada, o presente TCF logrou responder ao terceiro objetivo específico, a saber, a proposição de mecanismo de integração entre a gestão da propriedade intelectual de software e os processos de PD&I no âmbito de Instituições Federais de Ensino Superior.

O modelo de gestão de propriedade intelectual proposto foi baseado em uma visão processual e sistêmica do desenvolvimento de softwares. Ao considerar o conjunto de entradas e saídas do sistema, bem como, os processos adjascentes, foi possível determinar que toda a equipe da FS-Facom deve compreender os fundamentos da propriedade intelectual, dado que um dos mais importantes inputs do sistema é o capital intelectual dos desenvolvedores. A negociação preliminar de direitos e responsabilidades entre criadores e titular deve ocorrer antes do inicio do projeto, e não posteriormente, para que não ocorram entraves à realização de processos futuros de licenciamento dos softwares gerados, por ausência de autorizações de desenvolvedores que eventualmente já tenham se desvinculado da fábrica por titulação na universidade à qual se encontrava vinculado.

Na etapa de desenvolvimento do software, artefatos de software podem ser considerados como módulos independentes que podem ser reutilizados em novos projetos. A organização do repositório destes artefatos é outra proposta do modelo apresentado neste trabalho, dado que a mera catalogação dos artefatos de software para reuso é insuficiente para evitar problemas legais futuros e o efetivo licenciamento das novas tecnologias. Faz-se necessário que sejam geridas as licenças de artefatos externos, provenientes de repositórios digitais distintos daqueles existentes na fábrica. Um módulo baseado em artefato externo com licença que impeça o seu uso comercial, como por exemplo, a GLPv2, poderá vetar qualquer forma de licenciamento com ganhos econômicos futuros de um software.

Mediando as dimensões apresentadas e funcionando como integrador de processos, foi proposta a utilização de uma metodologia de gerenciamento ágil de projetos: o método IVPM2. Essa metodologia permite a gestão ágil de projetos, integrando assim os requisitos de informação relacionados a eventuais processos de formalização de parcerias para o desenvolvimento de solução, ao mesmo tempo em que mantém e armazena dados de controle de processo e de resultado, gerando assim os inputs para os processos subsequentes de registro do software e seu eventual licenciamento para organizações e pessoas nele interessadas.

Dessa forma, após adotar as contribuições propostas por esta pesquisa, espera-se que a Fábrica de Software da Facom/UFMS tenha a capacidade de desenvolver projetos de

software em conformidade com as normativas de propriedade intelectual, reduzindo o volume de consultas dos envolvidos e acelerando a curva de aprendizado, contribuindo assim para a maior efetividade das atividades da FS-Facom. Além disso, pretende-se que essas contribuições permitam uma redução de retrabalho e integrem os projetos da FS-Facom com os processos das áreas relacionadas à propriedade intelectual e avenças na UFMS.

Contudo, cabe destacar que a principal fragilidade da presente pesquisa é a falta de testes de validação para testar o modelo proposto. Dessa forma, com o intuito de ampliar a compreensão do fenômeno investigado por este TCF e testar o modelo proposto, sugere-se que pesquisas futuras possam fazer estudo comparativo sobre as práticas de gestão de propriedade intelectual adotadas por outras fábricas de software acadêmicas, principalmente aquelas vinculadas a instituições federais de ensino superior. Nesse sentido, parece ser interessante investigar como as outras fábricas de software acadêmicas lidam com a questão da atribuição de autoria e titularidade nos artefatos internos e externos de software.

8. Referências Bibliográficas

ABIB, G.; HOPPEN, N.; HAYASHI, P. Observação participante em estudos de administração da informação no Brasil. **RAE-Revista de Administração de Empresas**, v. 53, n. 6, 2013.

AGHION, P.; HOWITT, P. **The Economics of Growth**. Cambridge, MA: MIT Press, 2009.

_____. **Endogenous growth theory**. Cambridge, MA: MIT Press, 1998.

ADLER, P. A.; ADLER, P. **Membership roles in field research**. Beverly Hills, CA: Sage, 1987.

AQUILES, A.; FERREIRA, R. **Controlando versões com Git e Github**. São Paulo: Casa do Código, 2014.

ARAÚJO, B. C. **Políticas de apoio à inovação no Brasil: uma análise de sua evolução recente**. Texto para Discussão, Instituto de Pesquisa Econômica Aplicada (IPEA), 2012.

ARAUJO, C. **Uma interface de painel digital interativo para planejamento de projetos**. 2012. Tese (Doutorado em Processos e Gestão de Operações) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2012.

ARBIX, G.; CONSONI, F. Inovar para transformar a universidade brasileira. **Revista Brasileira de Ciências Sociais**, v. 26, n. 77, p. 205-224, 2011.

AREAS, P. O. **Contratos Internacionais de Pesquisa e Desenvolvimento de Software no Direito Internacional Privado Brasileiro e a Política Nacional de Desenvolvimento a Partir da Inovação**. Tese (Doutorado em Direito) – Curso de Pós-Graduação em Direito, Universidade Federal de Santa Catarina, Florianópolis, 2010

_____. **Contratos Internacionais de Software**: o direito moral do autor como limitante da autonomia da vontade. Dissertação (Mestrado em Direito) – Curso de Pós-Graduação em Direito, Universidade Federal de Santa Catarina, Florianópolis, 2006.

ARZA, V. Channels, benefits and risks of public-private interactions for knowledge transfer: conceptual framework inspired by Latin America. **Science and Public Policy**, v. 37, n. 7, p. 473, 2010.

ASCENSÃO, J. O. **Direito autoral**. 2. ed. Rio de Janeiro: Renovar, 1997.

ASSOCIAÇÃO BRASILEIRA DAS EMPRESAS DE SOFTWARE. **Mercado Brasileiro de Software**: panorama e tendências 2017. 1ª. ed. – São Paulo: ABES – Associação Brasileira das Empresas de Software, 2017.

BALBACHEVSKY, E.; SCHWARTZMAN, S. Brazil: Diverse experiences in institutional governance in the public and private sectors. In: **Changing governance and management in higher education**. Dordrecht, Netherlands: Springer, 2011. p. 35-56.

BARBOSA, B. D.; PRADO, E. R. **Quem é dono das criações sob a Lei de Inovação**. In: BARBOSA, B. D. (org.). Direito de inovação: comentários à Lei Federal de Inovação,

legislação estadual e local, poder de compra do Estado (modificações à Lei de Licitações. 2 ed. Rio de Janeiro: Lumen Juris, 2011. p.484-486.BARBOSA, B. D. (org.). **Direito da inovação**: comentários à Lei Federal de Inovação, legislação estadual e local, poder de compra do Estado (modificações à Lei de Licitações. 2 ed. Rio de Janeiro: Lumen Juris, 2011.

_____.Contratos de licenciamento e transferência de tecnologia da Lei de inovação. In: BARBOSA, B. D. (org.). **Direito da inovação**: comentários à Lei Federal de Inovação, legislação estadual e local, poder de compra do Estado (modificações à Lei de Licitações. 2 ed. Rio de Janeiro: Lumen Juris, 2011. p.319-334.

_____. **Uma introdução à propriedade intelectual**. 2 ed. Rio de Janeiro: Lumen Juris, 2003.

_____.**A proteção do software**. Rio de Janeiro: Lumen Juris, 2001.
Disponível em: <http://denisbarbosa.addr.com/77.doc>. Acesso em: 15 de abril de 2017.

_____. **Titularidade das obras produzidas em relação de subordinaçã**o. 1999. Disponível em <http://denisbarbosa.addr.com> Acesso em 14 de abril de 2017.

BASSO, M. A proteção da propriedade intelectual e o direito internacional atual. Revista de Informação Legislativa, v. 41, n. 162, p. 287-309, 2004.

BAUER, V. M. **Analysing and supporting software reuse in practice**. 2016. Tese de Doutorado (Doutorado em informática), Technische Universität München, Munique, Alemanha, 2016.

BAUER, V. M.; VETRÓ, A. Comparing reuse practices in two large software-producing companies. **Journal of Systems and Software**, v. 117, p. 545-582, 2016.

BERTALANFFY, L. V. Teoria Geral dos Sistemas. 2.ed. Petrópolis: Vozes, 1975.

_____. The history and status of General Systems Theory. **Academy of Management Journal**, v. 15, n. 4, p. 407-426, 1972.

BERTRAM D., et al. (2010). Communication, collaboration, and bugs: The social nature of issue tracking in software engineering. In **Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work,** p. 291-300, Savannah, GA, 2010.

BITTAR, C. A. **Direito de autor**. Rio de Janeiro: Forense Universitária, 2005.

BLACK DUKE. **2017 Open Source Security and Risk Analysis.** Center for Open Source Research & Innovation, Burlington, MA, 2017.
Disponível em:
https://www.blackducksoftware.com/open-source-security-risk-analysis-2017

BRASIL. Lei nº 10.973, de 2 de dezembro de 2004. Dispõe sobre incentivos à inovação e à pesquisa científica e tecnológica no ambiente produtivo e dá outras providências. Disponível em:
http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2004/lei/l10.973.htm. Acesso em: 19 de abril de 2017

_____. Lei nº 9.609, de 19 de fevereiro de 1998. Dispõe sobre a proteção da propriedade intelectual de programa de computador, sua comercialização no País, e dá outras providências. Disponível em:
<http://www.planalto.gov.br/ccivil_03/leis/L9609.htm> Acesso em: 19 de abril de 2017.

_____. Lei nº 9.610, de 19 de fevereiro de 1998. Altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências. Disponível em:
<http://www.planalto.gov.br/CCIVIL_03/leis/L9610.htm> Acesso em: 19 de abril de 2017.

_____. Lei nº 8.666, de 21 de junho de 1993. Regulamenta o art. 37, inciso XXI, da Constituição Federal, institui normas para licitações e contratos da Administração Pública e dá outras providências. Disponível em:
<http://www.planalto.gov.br/ccivil_03/leis/L8666compilado.htm> Acesso em: 19 de abril de 2017.

_____. Tribunal de Contas da União. Acórdão nº 5770/2014. Segunda Câmara. Representação. Universidade Federal de Santa Maria/RS. Possíveis irregularidades em contratação firmada com fundação de apoio. Diligências e audiências. Análise de documentos e razões de justificativa. Procedência parcial. Determinação. Recomendação. Ciência. Relator: Ministro José Jorge. Sessão de 14/10/2014. Disponível em:
https://contas.tcu.gov.br/pesquisaJurisprudencia/#/detalhamento/11/*/KEY%3AACORDAO-COMPLETO-1325292/DTRELEVANCIA%20desc/false/1. Acesso em: 14 jul. 2017.

_____. Tribunal de Contas da União. Acórdão nº 5684/2013. Primeira Câmara. Relator: Ministro Walton Alencar Rodrigues. Sessão de 20/08/2013. Representação. Ministério da Saúde. Possíveis irregularidades no fornecimento de sistema de gestão hospitalar. Conhecimento. Parcial procedência. Determinação de medida corretiva e de realização de auditoria operacional. Ciência. Disponível em:
https://contas.tcu.gov.br/pesquisaJurisprudencia/#/detalhamento/11/*/KEY%3AACORDAO-COMPLETO-1283711/DTRELEVANCIA%20desc/false/1. Acesso em: 14 jul. 2017.

_____**. Constituição da República Federativa do Brasil.** Brasília, DF: Senado Federal: Centro Gráfico, 1988.

CARVALHO, B. V de; MELLO, C. H. P**.** Aplicação do método ágil scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica**. Gestão & Produção**,v. 19, n. 3, p. 557-573, 2012.

CAPEK, P. G. et al. A history of IBM's open-source involvement and strategy. **IBM systems journal**, v. 44, n. 2, p. 249-257, 2005.

CASTELLACCI, F. A neo-Schumpeterian approach to why growth rates differ. **Revue Économique**, v. 55, n. 6, p. 1145-1169, 2004.

CERQUEIRA, T. Q. **Software**: lei, comércio, contratos e serviços de informática. Rio de Janeiro: Esplanada, 2000.

CHAVES, C. V. et al. The contribution of universities and research institutes to Brazilian innovation system. **Innovation and Development**, v. 6, n. 1, p. 31-50, 2015.

CHIARINI, T.; RAPINI, M.; VIEIRA, K. Produção de novos conhecimentos nas universidades federais e as políticas públicas brasileiras recentes de CT&I. **Revista Economia & Tecnologia**, v. 10, n. 3, 2014.

CONFORTO, E. C. **Gerenciamento ágil de projetos**: proposta e avaliação de método para gestão de escopo e tempo. Dissertação (Mestrado em Engenharia de Produção). Universidade de São Paulo, São Carlos, SP, Brasil, 2009.

CONFORTO, E. C. et al. **Gerenciamento ágil de projetos**: aplicação em produtos inovadores. São Paulo: Saraiva, 2011.

COPETTI, M. Marcas e programas de computador: seus pontos de intersecção. In: Luiz Otávio Pimentel. (Org.). Série PLATIC - **A proteção jurídica da propriedade intelectual de software**: noções básicas e temas relacionados. 1 ed. Florianópolis: FIESC/ IELSC, 2008, v. 2.

CUSUMANO, M. A. **Japan's software factories: a challenge to US management**. Oxford: Oxford University Press, 1991.

DE LAAT, P. B. Copyright or copyleft?: An analysis of property regimes for software development. **Research Policy**, v. 34, n. 10, p. 1511-1532, 2005.

DELAMARO, M.E; MALDONADO, J. C; JINO, M. **Introdução ao teste de software**. Rio de Janeiro**:** Campu**s**, 2007.

DENZIN, N. K. **The research act**. 3thd. ed. Englewood Cliffs: Prentice Hall, 1989.

DEWALT, K. M.; DEWALT, B. R. **Participant observation:** A guide for fieldworkers. New York: Rowman Altamira, 2011.

DI PIETRO. M. S. Z. **Direito Administrativo.** 30.ed. Rev., atual. e ampl. – Rio de Janeiro: Forense, 2017.

EDER, S. et al. Diferenciando as abordagens tradicional e ágil de gerenciamento de projetos. **Production**, v. 25, n. 3, p. 482-497, 2015.

FACULDADE DE COMPUTAÇÃO. **Modelo de Gestão e Processo Padrão de Desenvolvimento de Software da Fábrica de Software da Facom/UFMS.** Relatório Técnico, Campo Grande, jul. 2015.

FAGERBERG, J. Innovation: a guide to the literature. In: FAGERBERG, J.; MOWERY, D. C., NELSON, R. R. (Orgs.). **The Oxford Handbook of Innovation**. Oxford: Oxford University Press. 2004.

FALCÃO, J. et al. **Estudo sobre o software livre**. Presidência da República, Casa Civil, Instituto Nacional de Tecnologia da Informação. Rio de Janeiro, v. 18, 2005.

FERNANDES, A. A.; TEIXEIRA, D. S. **Fábrica de Software**: Implantação e gestão de Operações, Atlas, São Paulo, 2004.

FREEMAN, C.; LOUÇÃ, F. **As time goes by: from the industrial revolutions to the information revolution**. New York: Oxford University Press, 2001.

FREEMAN, C. The "National System of Innovation" in historical perspective. **Cambridge Journal of Economics**, v. 19, n. 1, 1995.

_____. Japan: a new National System of Innovation? In: DOSI, G.; FREEMAN, C.; NELSON, R.; SILVERBERG, G.; SOETE, L. **Technical change an economic theory**. London and New York: Printer Publishers, 1988. p. 330-348.

_____. **Technology Policy and Economic Performance**: lessons from Japan. London/New York: Pinter Publishers, 1987.

_____. **The economics of industrial innovation**. Harmondsworth, U.K: Penguin Books, 1974.

GERMAN, D M.; MANABE, Y.; INOUE, K. A sentence-matching method for automatic license identification of source code files. In: **Proceedings of the IEEE/ACM international conference on Automated software engineering**. ACM, p. 437-446, 2010.

GEUNA, A.; MUSCIO, A. The Governance of University Knowledge Transfer: A critical review of the literature**. Minerva**, v. 47, n. 1, p. 93-114, 2009.

GODIN, B.T. **Innovation, the History of a Category**. Project on the Intellectual History of Innovation. Working Paper N°. 1, INRS: Quebec, 2008.

GOULD, D. M.; GRUBEN, W. C. The role of intellectual property rights in economic growth. **Journal of Development Economics**, v. 48, n. 2, p. 323-350, 1996.

GROSSMAN H.; E. HELPMAN.**Innovation and growth in the global economy**. Cambridge, MA: MIT Press, 1993.

HADDAD, I. **Open Source Compliance in the Enterprise**. The Linux Foundation, 2016. Disponível em: https://www2.thelinuxfoundation.org/open-source-compliance-ebook

HAMMES, B. J. **O Direito de Propriedade Intelectual***:* Conforme a Lei 9619 de 19.2.1998. 3 ed. São Leopoldo: Unisinos, 2002.

HIDALGO, R.; CUESTA, J. A. **La siempre conflictiva relación del trabajador intelectual y um apunte específico para el creador de "software"**. Leon: Universidad de Leon, 2004.

HIGHSMITH, J.; COCKBURN, A. Agile software development: The business of innovation. **Computer**, v. 34, n. 9, p. 120-127, 2001.

KANWAR, S.; EVENSON, R. Does intellectual property protection spur technological change?. **Oxford Economic Papers**, v. 55, n. 2, p. 235-264, 2003.

KAMIYA T., KUSUMOTO S., INOUE K. CCFinder: A multilinguistic token-based code clone detection system for large scale source code. **IEEE Transactions Software Engineering** 28 (7), p.654–670, 2002.

KAST, F. E.; ROSENZWEIG, J. E. General systems theory: Applications for organization and management. **Academy of management journal**, v. 15, n. 4, p. 447-465, 1972.

KEMP, R. Towards Free/Libre Open Source Software ("FLOSS") Governance in the Organisation. **International Free and Open Source Software Law Review**, v. 1, n. 2, p. 61-72, 2009.

KON, F. et al. **Software Livre e Propriedade Intelectual**: Aspectos Jurídicos, Licenças e Modelos de Negócios. Disponível em: http://ccsl. ime. usp. br/files/slpi. pdf>. Acesso em, v. 2, p. 12, 2012.

LINDMAN, J.; PAAJANEN, A.; ROSSI, M. Choosing an open source software license in commercial context: A managerial perspective. In: **Software Engineering and Advanced Applications (SEAA) 2010 36th EUROMICRO.** IEEE, 2010. p. 237-244.

LIPSZYC, D. **Derecho de autor y derechos conexos**. Paris, Buenos Aires: Ed UNESCO , 2005.

_____. **Nuevos temas de derecho de autor y derechos conexos**. Paris: UNESCO; Bogotá: CERLALC; Buenos Aires: Zavalia, 2004.

LUNDVALL, B. A; FREEMAN, C. **Small Countries Facing The Technological Revolution**. London: Frances Pinter Publishers, 1988.

LUNDVALL, B. A. et al. National systems of production, innovation and competence building. **Research policy**, v. 31, n. 2, p. 213-231, 2002.
_____.**National Innovation System**: towards a theory of innovation and interactive learning. London: Pinter Publishers, 1992.

MENEELY, A.; CORCORAN, M.; WILLIAMS, L. Improving developer activity metrics with issue tracking annotations. In: **Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics**. ACM, 2010. p. 75-80.

MILES, M. B.; HUBERMAN, A. M.; SALDANA, J.. Qualitative data analysis: A method sourcebook. **CA, US: Sage Publications**, 2014.

MILES, M. B.; HUBERMAN, A. Michael. **Qualitative data analysis: An expanded sourcebook**. sage, 1994.

NADIN, S.; CASSELL, C.. Using data matrices. In: CASSELL, C.; SYMON, G. (Ed.). **Essential guide to qualitative methods in organizational research**, p. 271-287, Sage Publications: London, 2004.

NELSON, R. R. Capitalism as an engine of progress. **Research Policy**, v. 19, n. 3, p. 193-214, 1990.

NOBELIUS, D. Linking product development to applied research: transfer experiences from automotive company. **Technovation**, Vol. 24, n. 4, p. 321-334, 2004.

OLIVEIRA, N. IBGE: PIB fecha 2015 com queda de 3,8%. **Empresa Brasileira de Comunicação**, Rio de Janeiro, 3 Mar. 2016. Disponível em: <http://agenciabrasil.ebc.com.br/economia/noticia/2016-03/ibge-pib-fecha-2015-com-queda-de-38>. Acesso em: 3 Ago. 2017.

OLIVEIRA, M. S. **On the Use of Visualization for Supporting Software Reuse**. 2015. Tese de Doutorado (Doutorado em Engenharia de Sistemas e Computação.. Universidade Federal do Rio de Janeiro, 2015.

PAESANI, L. M. **Direito de informática**: comercialização e desenvolvimento internacional do software. 3 ed. São Paulo: Atlas, 2001.

PIMENTEL, L. O. (org.). **Manual básico de acordos de parceria de PD&I:** aspectos jurídicos. Porto Alegre: EDIPUCRS, 2010.

PÓVOA, L. M. C. **Patentes de universidades e institutos públicos de pesquisa e a transferência de tecnologia para empresas no Brasil**. Tese de Doutorado em Economia. UFMG/CEDEPLAR, Belo Horizonte, 2008.

RAPINI, M. S.; DE OLIVEIRA, V. P.; SILVA, T. C**. Como a interação universidade-empresa é remunerada no Brasil:** evidências dos grupos de pesquisa do CNPq**.** Revista Brasileira de Inovação, v. 15, n. 2 jul/dez, p. 219-246, 2016.

RAPINI, M. S. et al. University-industry interactions in an immature system of innovation: evidence from Minas Gerais, Brazil. **Science & Public Policy**, v. 36, n. 5, 2009.

ROMANHA, S. D. **Um Modelo de Fábrica de Software em Instituições de Ensino Superio**r. 2016. Dissertação de Mestrado Profissional (Mestrado Profissional em Engenharia de Produção) – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2016.

ROTHWELL, R. Towards the Fifth-generation Innovation Process. **International Marketing Review**, Vol. 11, n. 1, p.7-31,1994.

SABBAGH. R. **Scrum**: Gestão ágil para projetos de sucesso. São Paulo: Casa do Código, 2014.

SABINO, V. C. **Um estudo sistemático de licenças de software livre**. 2011. Dissertação (Mestrado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2011.

SANTOS, M. J. P. **A proteção autoral de programas de computador**. Rio de Janeiro: Lumen Juris, 2008.

SASSO DE LIMA, T. C.; TAMASO MIOTO, R. C. Procedimentos metodológicos na construção do conhecimento científico: a pesquisa bibliográfica. **Revista Katálysis**, v. 10, 2007.

SCHOTS, M.; WERNER, C. . Characterizing the Implementation of Software Reuse Processes in Brazilian Organizations. Rio de Janeiro: COPPE/UFRJ, 2014 (Relatório Técnico / Technical Report).

SCHWABER. K et al. **Scrum Development Process.** Object-Oriented Programming, Systems, Languages & Applications Conference - Business Object Design and Implementation Workshop, Austin, USA, 1995. Disponível em:

http://www.jeffsutherland.org/oopsla/schwapub.pdf

SILVA, A. P. **Antes de uma fundação, um conceito**: um estudo sobre a disciplina jurídica das fundações de apoio na cooperação entre universidade e empresa. Dissertação (Mestrado em Direito) - FGV - Fundação Getúlio Vargas, São Paulo, 2011.

SILVA, C. E. **Propriedade intelectual de programa de computador desenvolvido para utilização na administração pública**: estudo de caso. Tese (Doutorado do em Direito) – Curso de Pós-Graduação em Direito, Universidade Federal de Santa Catarina, Florianópolis, 2013.

SINCLAIR, A. Licence Profile: Apache License, Version 2.0, **International Free and Open Source Software Law Review** 2*(2), p. 107* – 114, 2010.

SOFTEX. **Mercado de Trabalho e Formação de Mão de Obra em TI**. 2013. Disponível em: <http://www.softex.br/inteligencia/#cadernostematicos>

_____. **Modelo MPS (Melhoria de Processo do Software Brasileiro)**: Guia Geral, Associação para a Promoção da Excelência do Software Brasileiro 2012. Disponível em:https://www.softex.br/wpcontent/uploads/2013/07/MPS.BR_Guia_Geral_Software_2012-c-ISBN-1.pdf

SUZIGAN, W.; ALBUQUERQUE, E. M. The underestimated role of universities for the Brazilian system of innovation. **Revista de Economia Política**, v.31, n.1, p. 03-30, mar. 2011.

SCHUMPETER, J. A. **Business Cycles**: a theoretical, historical and statistical analysis of the capitalist process. Philadelphia: Porcupine, 1989.

_____. **A Teoria do Desenvolvimento Econômico**. 3. ed. São Paulo: Abril Cultural, 1982.

_____. The analysis of economic change. **The Review of Economics and Statistics**, v. 17, n. 4, p. 2-10, 1935.

STELLMAN, A.; GREENE. J. **Learning Agile**: Understand Scrum, XP, Lean, and Kaban. Sebastobol, CA: O'Reilly, 2015.

TEDESCHI, P. P. **Inovação tecnológica e direito administrativo**. Dissertação (Mestrado em Direito. Universidade de São Paulo, São Paulo, 2011

TEIXEIRA, R. L. C. Os impactos da lei 10.973 de 2 de dezembro de 2004 sobre as cláusulas de propriedade intelectual nos contratos e parcerias celebradas entre empresas e instituições científicas e tecnológicas. In: BARBOSA, B. D. (org.). **Direito da inovação**: comentários à Lei Federal de Inovação, legislação estadual e local, poder de compra do Estado (modificações à Lei de Licitações. 2 ed. Rio de Janeiro: Lumen Juris, 2011. p.516-533.

TIOBE. **TIOBE Programming Community Indexfor April 2017**. Acessado em 18 de abril de 2017. Disponível em https://www.tiobe.com/tiobe-index/

THOMPSON, M. A.; RUSHING, F. W. An empirical analysis of the impact of patent protection on economic growth. **Journal of Economic Development**, v. 21, n. 2, p. 61-79, 1996.

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL. Faculdade de Computação. Edital n° 1, de 4 de Janeiro de 2018. Submissão e Seleção de Propostas de Software para a Fábrica de Software da Faculdade de Computação. Boletim de Serviços N° 6703, p.54-59, 5 de jan. 2018.

_____. Faculdade de Computação. Resolução n° 146, de 13 de Dezembro de 2017. Boletim de Serviços N° 6690, p.136-141, 15 de dez. 2017.

_____. Faculdade de Computação. Instrução de Serviço n° 97, de 2 de Outubro de 2017. Boletim de Serviços N° 6640, p.164, 3 de out. 2017.

_____. Faculdade de Computação. Instrução de Serviço n° 89, de 13 de Setembro de 2017. Boletim de Serviços N° 6627, p.170, 14 de set. 2017.

_____. Faculdade de Computação. Edital n° 23, de 13 de Setembro de 2017. Concurso para Criação de Logotipo da Fábrica de Software da Faculdade de Computação. Boletim de Serviços N° 6627, p.205, 14 de set. 2017.

_____. Conselho de Graduação. Resolução Nº 543, de 4 de Setembro de 2017. Boletim de Serviços N° 6621, p.3-73, 6 de set. 2017.

_____. Conselho Diretor. Resolução Nº 133, de 25 de Julho de 2017. Boletim de Serviços N° 6595, p.22-40, 2 de ago. 2017.

_____. Faculdade de Computação. Instrução de Serviço n° 72, de 16 de Setembro de 2016. Boletim de Serviços N° 6385, p.303, 23 de set. 2016.

_____. da Pró-Reitoria de Pós-Graduação e Pesquisa. Instrução Normativa n° 2, de 24 de Agosto de 2016. Boletim de Serviços N° 6363, p.157-165, 26 de ago. 2016.

_____. Pró-Reitoria de Planejamento e Orçamento. Instrução Normativa n° 1, de 24 de Maio de 2016. Boletim de Serviços N° 6341, p.265-336, 27 de jul. 2016.

_____. Faculdade de Computação. Instrução de Serviço n° 55, de 7 de Julho de 2016. Boletim de Serviços N° 6328, p.83, 8 de jul. 2016.

_____. Conselho Diretor. Resolução Nº 132, de 3 de Dezembro de 2015. Boletim de Serviços N° 6189, p.37-50, 17 de dez. 2015.

_____. Faculdade de Computação. Instrução de Serviço n° 105, de 1° de outubro de 2015. Boletim de Serviços N° 6138, p.95, 5 de out. 2015.

_____. Faculdade de Computação. Instrução de Serviço n° 9, de 23 de fevereiro de 2015. Boletim de Serviços N° 5985, p.153, 25 de fev. 2015.

_____. Conselho de Ensino de Graduação. Resolução Nº 740, de 29 de Dezembro de 2014. Boletim de Serviços N° 5966, p.91-139, 27 de jan. 2015.

_____. Conselho Universitário. Resolução Nº 80, de 22 de Outubro de 2014. Boletim de Serviços N° 5902, p.5, 27 de out. 2014.

_____. Conselho Diretor. Resolução n° 31, de 2 de dezembro de 2004. Boletim de Serviços N° 3500, p.1-2, 24 de dez. 2004.

VÄLIMÄKI, M. **The rise of open source licensing:** a challenge to the use of intellectual property in the software industry. Helsinki: Turre Publishing, 2005.

Vieira, J. A. **A Proteção dos Programas de Computador pelo Direito de Autor**. Lisboa: Lex, 2005.

WACHOWICZ, M. **Propriedade intelectual do software & revolução da tecnologia da informação**. Curitiba: Juruá, 2004.

WAZLAWICK, R. S. **Análise e Projeto de Sistemas de Informação Orientados a Objetos**. 2. Ed. Rev. e Atual. Rio de Janeiro, Rj: Elsevier, 2011.

WU, Y. et al. Analysis of license inconsistency in large collections of open source projects. In: **Empirical Software Engineering**, p. 1194–1222, Springer: New York, NY, 2017.

XAVIER. C.D. **Fábrica de Software**: Até que ponto Fordista? Dissertação (Mestrado em Gestão Empresarial) – Curso de Mestrado em Gestão Empresarial, Fundação Getúlio Vargas/Escola Brasileira de Administração Pública e Administração de Empresas, Rio de Janeiro**,** 2008.

YIN, R. K. **Estudo de caso**: planejamento e métodos. (2Ed.). Porto Alegre: Bookman. 2001.

ZIBETTI, F. W.; ZIEGLER FILHO, J. A. Os direitos de propriedade intelectual de programa de computador desenvolvido por servidor público do Estado de Santa Catarina. **Revista da ESMESC**, v. 21, n. 27, p. 299-324, 2014.

ZIBETTI, F. W. **A titularidade sobre os bens imateriais**. 2008. Dissertação (Mestrado em Direito) – Curso de Pós-Graduação em Direito, Universidade Federal de Santa Catarina, Florianópolis, 2008.

Anexo I – Regulamento das Disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software I

REGULAMENTO DAS DISCIPLINAS PRÁTICA EM DESENVOLVIMENTO DE SOFTWARE I E PRÁTICA EM DESENVOLVIMENTO DE SOFTWARE II DO CURSO DE ENGENHARIA DE SOFTWARE DA FACOM

CAPÍTULO I
DOS OBJETIVOS DAS DISCIPLINAS

Art. 1º Para os efeitos deste regulamento, designa-se como Órgão Colegiado Competente o Colegiado de Curso do Curso de Engenharia de Software e em grau de recurso o Conselho da Faculdade de Computação.

Art. 2º As disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II, do curso de Engenharia de Software da Facom, têm por objetivo vivenciar e aplicar, na prática, os conceitos da área de Engenharia de Software, obtidos durante o curso, em projetos de software reais. Ambas as disciplinas devem ser executadas no âmbito da Fábrica de Software da Facom, podendo contar com a participação de instituições parceiras, com o intuito de formar profissionais de excelência, socialmente conscientes e preparados para os avanços tecnológicos e científicos, capazes de criar e aplicar novas tecnologias para o bem estar da sociedade.

CAPÍTULO II
DO REQUISITO PARA MATRÍCULA NAS DISCIPLINAS

Art. 3º Para cursar as disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II, o acadêmico deverá estar apto conforme os pré-requisitos definidos no projeto pedagógico do Curso de Engenharia de Software.

CAPÍTULO III
DOS PROFESSORES SUPERVISORES E CONSULTORES

Art. 4º Qualquer professor da Facom pode ser supervisor e/ou consultor de turmas das disciplinas.

Art. 5º O professor supervisor é um professor alocado em pelo menos uma turma das disciplinas, sendo responsável por orientar e supervisionar uma ou mais equipes de desenvolvimento.

§1º Deverá ser alocado pelo menos um professor supervisor para cada turma.

§2º A carga horária a ser cumprida pelo professor supervisor é de 68 horas para cada turma.

§3º O professor supervisor tem como responsabilidades:

I – Acompanhar o projeto;
II – Acompanhar as atividades dos acadêmicos;
III – Informar às equipes correções de cada artefato entregue; e
IV – Avaliar os acadêmicos.

Art. 6º O professor consultor é um professor alocado em pelo menos uma turma das disciplinas, sendo responsável por apoiar um ou mais professores supervisores.

§1º Poderão ser alocados até três professores consultores para cada turma.

§2º A carga horária a ser cumprida pelo professor consultor é de 17 horas para cada turma.
§3º O professor consultor tem como responsabilidades:
I – Participar das reuniões e das atividades de supervisão de equipes, quando solicitado pelo professor supervisor;
II – Atuar como consultor nas fases do projeto em que possui maior *expertise*, visando contribuir com o desenvolvimento do projeto; e
III – Avaliar os artefatos entregues pelas equipes, quando solicitado pelo professor supervisor.

Art. 7º Professores de outras Unidades da Administração Setorial e de outras Instituições de Ensino Superior podem ser professores supervisores e/ou consultores de turmas das disciplinas, desde que aprovado pelo Órgão Colegiado Competente via Secretaria Acadêmica da Facom.

CAPÍTULO IV
DA COMPOSIÇÃO DAS TURMAS E DAS EQUIPES

Art. 8º Cada equipe das turmas das disciplinas deverá desenvolver um projeto sob orientação de um professor supervisor.

Art. 9º A escolha dos membros que devem compor cada equipe de uma turma das disciplinas deve ser feita pelo professor supervisor, após análise do perfil acadêmico de cada aluno, em busca de formar equipes balanceadas quanto a suas habilidades e conhecimentos.

Parágrafo único. Cada equipe deve ser composta por até cinco acadêmicos.

Art. 10. São obrigações dos acadêmicos matriculados nas disciplinas:

§1º Preencher e assinar o Termo de Compromisso em três vias:
I – O acadêmico somente poderá frequentar as aulas das disciplinas após a assinatura do Termo de Compromisso anexo a este Regulamento;

§2º Manter postura ética e profissional no desenvolvimento das atividades das disciplinas;

§3º Desenvolver as atividades e artefatos conforme definidos no plano do projeto de software; e

§4º Entregar o produto final em execução.

CAPÍTULO V
DA ESCOLHA DO PROJETO DE SOFTWARE

Art. 11. A comissão permanente da Fábrica de Software deve fornecer uma lista de propostas de projetos de software. É responsabilidade da Comissão Permanente da Fábrica de Software definir os critérios e elaborar a lista com as propostas de projeto de software.

Parágrafo único. Cada proposta de projeto de software deve contemplar a descrição do software a ser desenvolvido.

Art. 12. Cabe ao professor supervisor e à equipe a escolha da proposta de projeto de software dentre a lista fornecida. A proposta de projeto de software selecionada deverá ser desenvolvida pela equipe sob orientação do professor supervisor no decorrer da disciplina.

CAPÍTULO VI
DA AVALIAÇÃO

Art. 13. A avaliação das disciplinas deve ser feita em dois formatos: avaliação periódica e defesa de projeto.

Art. 14. A avaliação periódica é de responsabilidade do professor supervisor, e os critérios de avaliação devem ser detalhados no plano de ensino das disciplinas.

§1º Devem ser realizadas ao menos duas avaliações periódicas ao longo do semestre letivo.

§2º As avaliações periódicas podem ser executadas utilizando quaisquer instrumentos de avaliação permitidos conforme o Regulamento dos Cursos de Graduação Presenciais da UFMS.

Art. 15. A defesa de projeto deve ser realizada antes do término do semestre letivo.

§1º A defesa de projeto é oral e pública e será realizada perante uma Comissão Avaliadora composta por, pelo menos, dois professores, tendo o professor supervisor como membro nato e presidente:

I – A composição da Comissão Avaliadora será sugerida pelo professor supervisor e designada pelo Órgão Colegiado Competente;

II – Poderão ser convidados membros externos à UFMS para fazer parte da Comissão Avaliadora para fins de teste de aceitação do produto gerado; e

III – O convite para membros externos não retira a necessidade de ter ao menos dois professores na Comissão Avaliadora, como dispõe o enunciado do presente parágrafo.

§2º A defesa de projeto deve ser realizada mediante a apresentação do produto final em execução.

I – A preparação do ambiente para a realização da execução do produto é de inteira responsabilidade da equipe, que deve ser apoiada pelo professor supervisor.

Art. 16. A avaliação final deve ser uma composição da nota obtida na avaliação periódica e na defesa de projeto, sendo os pesos de cada uma destas avaliações definidas pelo professor supervisor no plano de ensino das disciplinas.

CAPÍTULO VII
DISPOSIÇÕES GERAIS

Art. 17. Os casos omissos serão resolvidos pelo Órgão Colegiado Competente.

Anexo II - Termo de compromisso das disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II

TERMO DE COMPROMISSO DAS DISCIPLINAS PRÁTICA EM DESENVOLVIMENTO DE SOFTWARE I E PRÁTICA EM DESENVOLVIMENTO DE SOFTWARE II

Eu, _____, inscrito(a) no RG sob o nº_____, órgão expedidor _____, portador do CPF/MF sob nº _____, residente e domiciliado(a) no endereço _____ na cidade de _____ no estado de_____, na qualidade de acadêmico matriculado na disciplina Prática em desenvolvimento de Software ....., assumo com a Universidade Federal de Mato Grosso do Sul, neste ato representada pelo(a) Professor(a) Supervisor(a) da disciplina supracitada e pelo(a) Coordenador(a) da Fábrica de Software da Faculdade de Computação, os compromissos descritos neste termo de compromisso.

DAS DISPOSIÇÕES GERAIS

Este termo de compromisso está em conformidade com o Projeto Pedagógico do Curso aprovado mais recentemente, com o Regulamento dos Cursos de Graduação Presenciais da UFMS e com o Regulamento das Disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II.

As disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II do curso de Engenharia de Software da Faculdade de Computação têm por objetivo vivenciar e aplicar na prática os conceitos da área de Engenharia de Software obtidos durante o curso, em projetos de software reais, os quais devem ser executados no âmbito da Fábrica de Software da Faculdade de Computação.

DO OBJETO

Os alunos matriculados nas disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II devem desenvolver as atividades dessas disciplinas conforme o plano do projeto de software aprovado pelo professor supervisor; bem como entregar um produto final funcional.

DA JORNADA

Os acadêmicos matriculados nas disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II do Curso de Engenharia de Software da

Faculdade de Computação devem cumprir carga horária de no mínimo 75% da carga horária total das disciplinas prevista no Projeto Pedagógico do Curso.

DA CONFIDENCIALIDADE

Todas as informações a que os acadêmicos matriculados nas disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II tiverem acesso, decorrentes da participação em projetos de software aprovados pelo professor supervisor daquelas disciplinas, serão informações confidenciais.

Essas informações confidenciais devem ser mantidas em sigilo após o término das atividades daquelas disciplinas, independente da forma ou meio que foram passadas, não podendo repassá-las a quem quer que seja sem autorização feita por escrito por representante legal da UFMS.

DA PROPRIEDADE INTELECTUAL

Considerando o disposto no §3º do Art. 4° da Lei nº 9.609, de 19/02/98 e as normativas da UFMS referentes à propriedade intelectual, os acadêmicos matriculados nas disciplinas Prática em Desenvolvimento de Software I e Prática em Desenvolvimento de Software II, doravante denominados autores, cederão à UFMS todos os direitos patrimoniais relativos ao(s) Programa(s) de Computador(es) desenvolvido(s) na disciplina Prática em Desenvolvimento de Software ___, no _____ semestre de _____.

Serão garantidos aos alunos autores os direitos morais previstos no §1º do Art. 2º da Lei nº 9.609, de 19/02/98; bem como a participação nos ganhos econômicos, auferidos pela UFMS, resultantes de contratos de transferência de tecnologia e de licenciamento para outorga de direito de uso ou de exploração de criação desenvolvida pelos alunos autores, conforme o Art. 13 da Lei n° 10.973, de 2/12/2004.

A distribuição dos ganhos econômicos assegurados aos alunos autores será repartida de acordo com a contribuição intelectual dos criadores do produto final.

E por estarem justas e acordadas, as partes assinam o presente termo de compromisso em três vias de igual teor e forma.

Campo Grande-MS, ____ de _____de _____.

_____

Acadêmico(a)

_____

Professor(a) Supervisor(a)

_____

Coordenador(a) da Fábrica de Software