CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
ULBRA
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Cleydiane Lima de Sousa

Metodologia de Desenvolvimento de Software para a Fábrica de Software do CEULP/ULBRA

Palmas - TO

2012

Cleydiane Lima de Sousa

Metodologia de Desenvolvimento de Software para a Fábrica de Software do CEULP/ULBRA

Trabalho apresentado como requisito parcial da disciplina Trabalho de Conclusão de Curso (TCC) do curso de Sistemas de Informação, orientado pelo Professor Mestre Jackson Gomes de Souza.

Palmas - TO

2012

Cleydiane Lima de Sousa

Projeto de uma Metodologia de Desenvolvimento de Software para a Fábrica de Software do CEULP/ULBRA

Trabalho apresentado como requisito parcial da disciplina Trabalho de Conclusão de Curso (TCC) do curso de Sistemas de Informação, orientado pelo Professor Mestre Jackson Gomes de Sousa.

Aprovada em junho de 2012.

BANCA EXAMINADORA

_____

Prof. M.Sc. Jackson Gomes de Souza

Centro Universitário Luterano de Palmas

_____

Prof. M.Sc. Cristina D'Ornellas Filipakis Souza

Centro Universitário Luterano de Palmas

_____

Prof. M.Sc. Fernando Luiz de Oliveira

Centro Universitário Luterano de Palmas

Palmas - TO

2012

Agradecimento

Deus sempre em primeiro lugar, afinal sem Ele as próximas linhas não existiriam. Agradeço a Ele, por ter me dado força até o presente momento, por Ele ter me guiado até aqui desde o dia do vestibular. 1º de junho de 2007 eu cheguei aqui, alguns dias depois fiz o vestibular, e sem a Sua ajuda, nada até aqui teria acontecido. Os dias de desespero de saudades de casa, Ele me confortou; o desespero na época de prova, na época de coração agoniado, na época de defesas de TCCs, na época que o dinheiro não era suficiente, e principalmente em cada momento que me mostrou o caminho certo a seguir. Saí do curso como uma vitoriosa, como alguém que entrou pra cumprir várias missões e elas foram cumpridas. Deus, sem você nada disso era possível.

Agradeço aos meus pais. Primeiro a minha mãe, que em cada lágrima derramada me dava mais força pra continuar, e em cada palavra querendo me transmitir calma, por eu ser muito agitada. Pelo sorriso e olhar que nunca me deixaram desistir, mesmo eu não acreditando em mim. Por poucas vezes ter vindo me visitar, mas todas as vezes que eu ia pra casa, era tratada como uma verdadeira princesa. Mãe sem ti nada teria sentido, você é minha força maior. Ao meu Herói, que muitas vezes deixou de faltar algo pra si, pra que eu vivesse na capital do Tocantins como uma verdadeira filha do "dono da Cemar", nada me faltou, mesmo que as vezes ele quisesse me regrar financeiramente ele não conseguia. Agradeço por ambos sempre me deixarem livre pra voar, e mesmo com essa liberdade me fizeram ser uma pessoa que nem vocês, determinada. Olha só, eu a primeira filha de vocês formada, e só agradeço a vocês pelo grande esforço de mostrarem pra mim o quanto eu sou capaz e de sempre me apoiar nas minhas decisões. Sei que não foi fácil me manter aqui e eu conseguir formar numa faculdade particular, as dificuldade foram imensas, mas a nossa vontade de vencer superou tudo, eu consegui e é por vocês que eu vivo. E aqui agradeço os meus irmãos, que nunca vieram saber o mundo que vivia, mas eram felizes por eu está viva, assim ausente me deram menos preocupação, amo tanto que não suportaria saber diariamente do sofrimento de vocês. Agradeço a Ana Carla, por ocupar meu espaço enquanto estou fora. Agradeço o Cleyton, porque na hora que eu grito ele tenta me ajudar. Porém um dos melhores presente você me deu nessa caminhada, que foi a dádiva de ter a primeira sobrinha, a Tayla, que me mostrou o verdadeiro amor de uma tia, obrigada.

Agradeço minha Maezinha, meu Paezinho, a Madrinha Dija, o Lucas, que fazem parte da saudade que sinto, e me apoio mesmo estando longe. Agora começo os agradecimentos do inicio da faculdade até chegar na reta final.

Agradeço a Mayene Karen e seu pai, por me conduzirem como entrar na faculdade, foi quando me matriculei fazendo poucas matérias, e pelo apoio dela quando não tinha pra onde ir. Agradeço a Waleria Laurindo, por ter me acolhido na sua casa, somente com meu colchão durante um ano, foi essencial pra caminhada chegar aqui.

Agradeço a Ione o Nairton e Tyago e Nayra, por sempre se fazerem presentes na minha caminhada. E em especial ao "Rato", que não abriu mão de me ajudar financeiramente durante os dois últimos anos. A Ione por sempre ter imposto que não aceitava eu não fazer faculdade e sempre teve presente me ouvindo, tanto as angustias como nas alegrias. E ao Gustavo que é um menino feliz, fez minhas manhãs felizes durante um bom tempo. A Nayra e Tyago por me perturbarem que nem meus irmãos de sangue. A dona vó e seu vô que eram meus avozinhos adotados, pelo animo e apoio que me deram.

Agradeço a Preta, Sofia, Angélica, Cil e o Gean, pela amizade e a força dada durante essa caminhada.

Agradecimentos a pessoas do curso:

- Agradeço ao Leafarneo que nas primeiras aulas de algoritmo sempre foi paciente e sempre ensinou o caminho, mas nunca deu uma resposta pra chegar ao resultado (eu tinha uma raiva disso), e durante todo o curso, sempre amigo.
- Ao Leandro que quando menos eu acreditava em mim, ele acreditava, mesmo me esculachando, me acabando com suas brincadeiras que no final do curso deixou saudades.
- A Valeria que na época que esteve no curso sempre ajudou e depois que formou continuo dando incentivo.
- Ao Van, que me ajudou na pior matéria que eu tive no curso, Tópicos em Web service e pelas conversar e amizades.
- Ao Will, que fomos companheiros e aprendi muito com ele, durante o Proict, fomos guerreiros, as experiências ali foram importantes pra nossa formação. Além disso, agradeço pelas amizades e as conversas jogadas fora.
- Ao Jonatas, que entrou junto comigo no curso, e sempre foi um bom amigo e me apoio muito durante o tempo que esteve comigo.
- Em especial a Milena, que esteve um bom tempo comigo, sonhamos várias vezes em chegar a esse final, mas por motivo maior ela foi embora pra Portugal. Bons tempos de aprendizagem e amizade.

- Ao Roneylson que me ajudou em muitos choros e sorrisos, me acompanhando mesmo antes da faculdade.
- Sará, Kátia, Cris, Jorge, Neuziron, Paulo, Diego, Welligton, Moisés, Andrew, Rauricio, Charles, agradeço vocês por fazem parte de muitas das minhas conquistas, e os que pela correria esqueci de citar. Uma caneta que se empresta, um código que se ajuda a implementar a um ombro amigo, fazem parte da construção dessa formação. Obrigada.

Agradeço a todos do Maranhão: Nyra, Lucidalva, Lucinha, Leonan, Ana Eliza, Dona Rita ("in memória"), Clóres.

Queria agradecer ao meu orientador o Jackson, me faltam as palavras, porque não sei como retribuir tanto zelo pra me orientar. Pela orientação profissional, que de orientador virou amigo. Amigo este que levarei pra sempre. Obrigada pelas conversas em dias de desespero, pela paciência quando eu queria desistir e é a você que devo grande parte do que sou como pessoa. Sem sua orientação profissional eu teria outros caminhos. Isso aqui é pouco pra agradecer tamanha ajuda.

A Mayanne, agradeço pela amizade que ficou maior ainda quando podemos dividir essa época de faculdade juntos. Agradeço pelo cuidado, que sempre foi como uma irmã. Agradeço pelos seus conselhos que muitos foram seguidos, pelo companheirismo nas minhas agonias diárias. Obrigada por ser quem és pra mim.

Agradeço a Luane e a Naara, sim ambas juntas, por terem entrado na minha vida juntas e termos feitos ótimas parcerias. Agradeço a Naara por sua calma e ajuda em tudo que fizemos. A Luane por seu espírito de "velha" que me contagia e me tira grandes sorrisos. Agradeço as duas, por essa grande amizade-irmã que levo do curso.

Agradeço todos os professores, cada um me ajudou nessa jornada. A Mádia pelas conversas, palas ajudas nos eventos, pelas risadas, pelas grandes preocupações comigo, não é atoa que leva o papel de Mãezona do curso. A Parcilene, agradeço por conduzir o curso de uma maneira tão sua e única. Agradeço a Cristina, por ter me ajudado bastante no início do curso, pela amizade, conselhos, as idas ao médico e por ter feito parte da minha Banca de TCC. Agradeço ao Fernando, pelas conversas, as grandes risadas, e por não querer me orientar (brincadeira). Agradeço ao Fabiano pela ajuda no inicio do curso, pelas conversar e conselhos. O Edesilson por ser o professor da minha primeira substituição na primeira matéria que fiz com ele e última, agradeço por me mostrar que as pessoas inicialmente ruins podem se tornar ótimas, basta conhecermos melhor.

Agradeço ao Ricardo, que mesmo sendo meu namorado, foi amigo durante os 5 anos do curso, e tornou-se mais amigo quando lutamos juntos pela finalização do TCC. É no desespero que grandes amizades se fortalecem. Obrigada pelo apoio, companheirismo e paciência durante um ano de namoro e um ano de tentativa até chegamos à reta final.

Agradeço a todos que formaram agora: Douglas Neves (amizade e grandes emoções), Douglas Brito (grande companheirismo nas matérias), Elias (amigo), Iroilton (grande ajuda na reta final), Lucas (muita ajuda e incentivo durante todo o curso), Luane, Marcio, Naara, Ricardo, Thearlismar (da mesma turma, entramos e saímos juntos do curso). Nós unidos vencemos, mas essa etapa. E foi a nossa união que deu força maior pra alguns formarem e fazer a turma 2012-1 ser tão grande.

Enfim agradeço a todos, que ajudaram diretamente ou indiretamente. Isso pra mim é um sonho, na qual nunca esquecerei nada do que foi vivido. Obrigada.

SUMÁRIO

RESUMO

Este trabalho tem por objetivo propor uma metodologia de desenvolvimento de software (MDS) para a Fábrica de Software (FSW) do CEULP/ULBRA. Esta proposta terá como base o uso das melhores práticas de duas metodologias ágeis já existentes: o *Scrum* e o FDD, de acordo com o que for adaptável à realidade da FSW. Algumas das justificativas deste trabalho são: o fato de a equipe da FSW não seguir um padrão de desenvolvimento, a alta rotatividade de funcionários e os prazos curtos. Assim, o presente trabalho apresenta a criação da MDS para a FSW, que contém: processos, papéis, reuniões e artefatos.

PALAVRAS CHAVE: Metodologias Ágeis, Fábrica de Software, FDD, *Scrum*.

LISTA DE TABELAS

LISTA DE FIGURAS

1 INTRODUÇÃO

A década de 1960 foi marcada por um período intitulado pela literatura como “crise do software” (PRESSMAN, 1995, p.22). Os problemas encontrados no desenvolvimento de software levaram os especialistas a repensar o processo de desenvolvimento do software. Um destes especialistas, Friedrich Ludwig Bauer, em 1960, definiu a Engenharia de Software como sendo "a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe em máquinas reais" (PRESSMAN, 1995, p.31). A partir deste período, técnicas de engenharia de software foram criadas e/ou aprimoradas e a disciplina da Engenharia de Software se mantém em constante evolução.

A demanda por software nas organizações cresceu com a evolução das tecnologias e com o aumento das possibilidades que as mesmas oferecem em áreas como administração, acompanhamento e planejamento das empresas. “A exemplo do crescimento e amadurecimento das fábricas de software da Índia as iniciativas brasileiras têm se multiplicado e apresentado um crescimento considerável nos últimos tempos, especialmente devido a fatores competitivos” (KRIPALANI, 2003 apud ROCHA, 2008, p. 11). Esta competividade levou ao surgimento de muitas Fábricas de Software. Segundo Nomura (2008, p. 13) a “fábrica de software” é um conceito que reúne um conjunto de características como: simplificação, integridade conceitual, aderência aos padrões, automação seletiva no processo de desenvolvimento, padronização de tarefas e de controles, divisão de trabalho, mecanização e automotivação.

Com o aumento da concorrência entre as empresas e com a instauração de um mercado mais exigente, no que se refere à qualidade do software, notou-se a necessidade de se utilizarem metodologias de desenvolvimento mais flexíveis e ágeis, para diminuir o tempo de desenvolvimento e, ao mesmo tempo, não afetar a qualidade do produto. Neste sentido concluem Beck et al (2001, online) que “a partir da década de 90, começaram a surgir novos métodos sugerindo uma abordagem de desenvolvimento ágil onde os processos adotados tentam se adaptar às mudanças, apoiando a equipe de desenvolvimento no seu trabalho”. Segundo Gomes (2009, pg. 17) as metodologias que são ágeis possuem algumas características como: processo de desenvolvimento incremental e evolutivo, mudanças de requisitos são bem recebidas etc. Alguns exemplos de Metodologias Ágeis são o FDD (*Feature-Driven Development*) e o *Scrum*.

Conforme informações obtidas junto à coordenação da Fábrica de Software do CEULP/ULBRA os cursos de computação do CEULP/ULBRA, há alguns anos, mantêm uma fábrica de software (FSW). Embora esta FSW desenvolva software há praticamente uma década, ainda não existe uma metodologia formalizada para o desenvolvimento de software. Esta ausência de metodologia formalizada pode atrapalhar os processos internos da FSW em pontos como: gerenciamento de tarefas e prazos, cumprimento de metas, documentação de processos, dificuldade de manutenção e acompanhamento do rendimento do trabalho dos membros da equipe. Com base neste contexto o presente trabalho tem o objetivo de definir uma metodologia ágil para a FSW do CEULP/ULBRA, para conseguir ter um melhor gerenciamento nos produtos desenvolvidos, bem como sua documentação e garantia de melhoria de processo.

Para que fosse possível a concretização deste trabalho foi necessário obter conhecimento sobre os assuntos relacionados. O resultado deste estudo está apresentado neste trabalho da seguinte forma: Referencial Teórico (capítulo 2), onde são apresentados os conceitos relacionados à proposta, expondo a conceituação de "fábrica de software" (seção 2.1); além de Metodologia de Desenvolvimento de Software (seção 2.2), com referência sobre os seguintes assuntos: Ciclo de vida Clássico, Prototipação e Modelo Espiral. Posteriormente, são apresentados os conceitos de Metodologia de Desenvolvimento Ágil (seção 2.3). Depois, são apresentadas as metodologias *Feature Driven Development*; (seção 2.4) e *Scrum* (seção 2.5), com seus respectivos processos e atividades. O trabalho ainda apresenta: Materiais e Métodos (capítulo 3) utilizados durante a realização do trabalho; Resultados e Discussão (capítulo 4), informando os resultados obtidos com a aplicação dos conceitos estudados; Considerações Finais (capítulo 5), informando as conclusões obtidas durante o desenvolvimento do trabalho; Referências Bibliográficas (capítulo 6), que apresentam as referências dos trabalhos utilizados durante o desenvolvimento deste trabalho; e os Apêndices, que apresentam os artefatos desenvolvidos.

2 REFERENCIAL TEÓRICO

Para a criação da metodologia para a Fábrica de Software do CEULP/ULBRA foram estudados os paradigmas clássicos da Engenharia de software (Ciclo de Vida Clássica, Prototipação e Modelo Espiral) e as metodologias ágeis, como FDD e *Scrum*. A seguir serão abordados todos os assuntos citados.

2.1. Fábrica de Software

Um dos significados de "fábrica" é "estabelecimento ou lugar onde se fabrica alguma coisa" (MICHAELIS, online). Um "software" é "(...) qualquer programa ou grupo de programas que instrui o *hardware* sobre a maneira como ele deve executar uma tarefa, inclusive sistemas operacionais, processadores de texto e programas de aplicação (...)" (MICHAELIS, online). Com base na junção desses dois significados, pode-se afirmar que "fábrica de software" (FSW) é o local onde se fabricam programas utilizando *software* e *hardware* para executar uma tarefa. Além disso, tem-se o entendimento de que fabricar *software* vai muito além desta definição, pois existe todo o contexto de como será fabricado, quais linhas de desenvolvimento serão usadas, quem exercerá qual função, quais os conhecimentos adquiridos de uma fabricação para outra dentre outras coisas. Fernandes (2004, p. 13) cita os seguintes elementos que compõem uma FSW:

- Processo definido e padrão;
- Interação controlada com o cliente;
- Solicitações de serviço padronizadas;
- Estimativas de custos e prazos;
- Controle rigoroso dos recursos envolvidos em cada demanda da fábrica;
- Controle e armazenamento em bibliotecas de itens de *software*;
- Controle dos *status* e execução 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 definidos com o cliente.

Uma das grandes necessidades de uma FSW é seguir os padrões, estimar melhor o tempo de entrega de *software* e, com isso, garantir também a qualidade do produto final. Entre outras atividades de controle, como padrões da execução, bom atendimento ao cliente, assim como princípios na gestão de processo, como explica Nomura apud Humphrey (1993, p. 40), há outros pontos:

- **Administração do pessoal**: os profissionais que participam de todo o processo de uma fábrica devem ser estimulados a desempenharem o que têm de melhor, e serem incentivados e ter apoio a cada dia para aumentarem seu desempenho. A gestão deve entender que os erros pessoais que surgem na produção devem ser corrigidos; e esta atividade de correção não deve ser vista como algo que venha a atrapalhar o desenvolvimento.
- **Modelo de processo**: "os processos precisam ser formalmente definidos e ter metas e elementos de mensuração estabelecidos, para que dados estatísticos possam ser colhidos e analisados, permitindo identificar problemas e terminar suas causas";
- **Suporte ao processo**: as pessoas envolvidas no processo precisam ter especializações e treinamentos nas suas áreas de especialidade, para que isso venha a melhorar no desempenho das suas atividades e para que possam manusear melhor as ferramentas que usam.
- **Controle do processo**: práticas de gerenciamento são necessárias para se ter controle sobre o processo, considerando a avaliação dos períodos produtivos e identificando necessidades de melhoria, o que leva a ações corretivas, dando qualidade para o processo.

A junção de gerenciamento, pessoas capacitadas, prazo e controle faz com que a FSW siga um processo com padrões. A seguir serão apresentados paradigmas que são, há bastante tempo, utilizados para desenvolvimento de *software*.

2.2. Metodologia de Desenvolvimento de Software

Segundo Sommerville (2004, p. 4) Metodologia de Desenvolvimento de Software (MDS) dita "práticas e técnicas com a finalidade de obter melhores formas de gerenciar e ordenar o processo através de fases e/ou passos", ou seja, estes elementos representam um conjunto de

práticas e padrões que são adotados por uma equipe de desenvolvimento durante o planejamento e a execução de um projeto.

Uma MDS é organizada para que a equipe alcance um objetivo, que procura satisfazer as necessidades do cliente. Desta forma, o produto se torna o componente mais importante de todo o processo de desenvolvimento (LIMA, s/a, p. 2).

Em uma MDS é importante estabelecer prazos a serem seguidos, pois, a partir disso, "todo projeto de software tem prazos e é comum ter como desafio no contexto do desenvolvimento de software a entrega do produto de forma que esteja dentro do prazo, de acordo com o orçamento previsto e que atenda as necessidades do cliente" (FAGUNDES, 2005, p. 19). Por isso, no início do desenvolvimento do projeto, é necessário se estabelecerem prazos. Além disso, há clientes que exigem prazos curtos, o que também precisa ser levado em consideração durante o desenvolvimento do projeto.

Há metodologias, como *Scrum* e FDD, que estipulam prazos para cada funcionalidade que será desenvolvida e estabelecem um prazo para o projeto como um todo. É de fundamental importância que a equipe de desenvolvimento entregue o produto no prazo correto demonstrando, assim, comprometimento com o cliente. Se o desenvolvimento do produto está demorando mais do que esperado, é necessário fazer novo planejamento para não ocorrerem atrasos. No FDD, segundo Palmer (2002, p. 66, tradução nossa), no terceiro processo do seu desenvolvimento, é feito um planejamento de quanto tempo se levará para desenvolver cada funcionalidade, o que é realizado com base em fatores como: a dependência entre as classes, as dificuldades das funcionalidades e a carga horária dos responsáveis das classes.

A escolha de qual metodologia será utilizada depende de fatores como a própria equipe de desenvolvimento, a quantidade de pessoas, as experiências que elas possuem e do tipo de *software* que será desenvolvido. Então, para saber qual metodologia de *software* será aplicada no desenvolvimento do *software*, é necessário que o responsável pela escolha da metodologia observe o que se tem em mãos, como tempo, equipe, projeto etc.

Existem MDS que possuem diversos tipos de focos, como, por exemplo, *Feature Driven Development* (FDD), que indica a divisão da equipe e desenvolve por funcionalidade. Há metodologias que enfocam em testes, como o *Test Driven Development* (TDD), que requer

que os membros da equipe façam os testes de unidade em todo o código do *software*. Por fim, há metodologias que indicam como organizar cada papel dos membros das equipes, como o *Scrum*, que determinada o papel de cada membro da equipe de desenvolvimento.

O não uso de uma MDS pode representar o surgimento de alguns problemas no desenvolvimento como:

- Falta de organização para realizar as atividades. Muitas vezes a não definição de prioridades faz com que exista uma sobrecarga de trabalho para algum membro da equipe de desenvolvimento;
- A falta de definição de uma metodologia de desenvolvimento faz as pessoas trabalharem sem padrão, o que gera uma dependência de uma pessoa específica para realizar a manutenção do *software*;
- Falta de documentação, que gera dificuldade de manutenção e entendimento do projeto como um todo, o que faz com que os membros da equipe de desenvolvimento dependam diretamente da pessoa que implementou uma funcionalidade;
- Falta de definição de um método que ofereça padronização do desenvolvimento, permitindo que os projetos sejam interpretados com mais facilidade, aumentando o desempenho tanto em sua concepção quanto em atividades de manutenção.

Esses problemas, entre outros, existem desde os primórdios do desenvolvimento de *software*. Por isto, foram criados paradigmas para ajudar a equipe de desenvolvimento de *software* nas suas tarefas. Entre os paradigmas estão: Ciclo de Vida Clássico, Prototipação e Modelo Espiral. Estes paradigmas foram criados na década de 1970 e ainda hoje são base para várias metodologias, como o *Scrum*, que utiliza o Modelo Espiral nas suas iterações. Estes paradigmas serão explicados nas próximas seções.

2.2.1. Ciclo de Vida Clássico

O Ciclo de Vida Clássico, também chamado “Modelo em Cascata”, foi um dos primeiros modelos propostos. “Foi criado por Royce, em 1970, com a finalidade de modificar os vários modelos de processos de engenharia existentes na época, que eram caóticos, e transformá-los em um modelo iterativo, o que o tornava um modelo linear” (LORDELLO, 2010, p. 27). Como vantagem desse modelo tem-se a documentação produzida em cada fase (SOMMERVILLE, 2007, p. 9). Este modelo é semelhante ao ciclo de vida de um produto,

com início, meio e término, no qual cada etapa tem que ser transcorrida completamente para a próxima ser iniciada (PINHEIRO, 2007, p. 5).

Pressman (2003, p. 34) aponta alguns problemas identificados nesta metodologia de desenvolvimento de software:

- Os projetos reais raramente seguem o fluxo sequencial que o modelo propõe. Algumas iterações trazem problemas na aplicação do paradigma;
- É difícil para o cliente declarar todas as exigências explicitamente. O ciclo de vida clássico exige isso e tem dificuldade de acomodar a incerteza natural que existe no começo de muitos projetos;
- O cliente deve ter paciência. Uma versão de trabalho do *software* não estará disponível até um ponto tardio do cronograma do projeto. Um erro grave, se não for detectado até que o programa de trabalho seja revisto, pode ser desastroso.

Despois de serem enumerados alguns desses problemas pode-se mostrar que "o paradigma de ciclo de vida requer uma abordagem sistemática, sequencial ao desenvolvimento do software, que se inicia no nível de sistema e avança ao longo da análise, projeto, codificação, teste e manutenção" (PRESSMAN, 1995, p. 32). O ciclo de vida clássico possui as etapas mostradas na Figura 1.

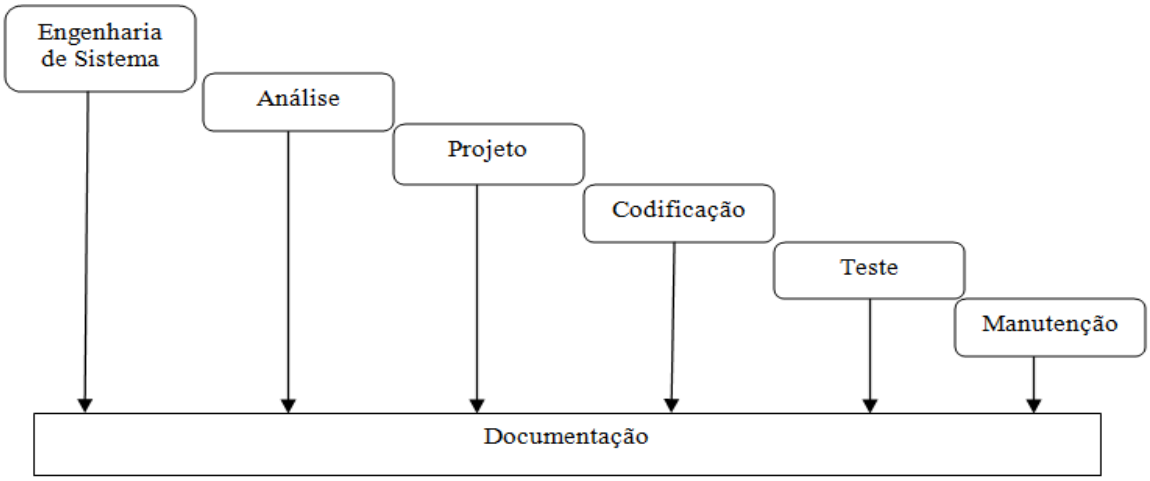


Figura 1 - Ciclo de vida clássico

A Figura 1 mostra as etapas do Ciclo de Vida Clássico. Na figura, as caixas indicam as fases, as setas que encontram uma linha, abaixo, indicam que cada fase inicia quando termina a outra, pois a linha inferior é a linha do tempo. Segue cada fase do processo:

1. **Engenharia do sistema**: onde é feita a análise de requisitos, é estudada a interação entre *software* o *hardware*, pessoas (usuários) e banco de dados.
2. **Análise de requisitos de *software***: onde se entende o funcionamento do sistema e da interface e é feita a documentação dos requisitos. O resultado desta etapa é revisado pelo cliente.
3. **Projeto**: segundo Pressman (1995, p. 32), são atribuídos quatro atributos distintos ao software: estrutura de dados, arquitetura de software, detalhes procedimentais e característica da interface.
4. **Codificação**: fase na qual serão implementadas as funcionalidades planejadas.
5. **Testes**: fase na qual se procura descobrir os possíveis erros na codificação e é verificado se o software está funcionando conforme o que se havia planejado.
6. **Manutenção**: fase que é executada após os testes, para fazer as correções dos erros, ou realizar ajustes de acordo com os pedidos do cliente. "A manutenção do *software* reaplica a cada uma das etapas precedentes do ciclo de vida de um programa existente, e não a um novo" (PRESSMAN, 1995, p. 35). Assim, modifica-se o que já foi feito até chegar nesta fase, não descartando o que já foi testado.

2.2.2. Prototipação

O processo de Prototipação é um ciclo iterativo, com várias partes, diferente do modelo citado anteriormente, que é linear.

"Um protótipo é construído a partir de requisitos. É realizada uma avaliação crítica do protótipo a qual considera os requisitos e os que não foram mencionados inicialmente" (CORDEIRO, 2005, online). Através da avaliação crítica se consegue obter uma maior visão do sistema, conseguindo ter detalhe das funcionalidades que não são mencionadas no início. Segundo Pressman (1995, p. 35) a prototipação pode assumir três formas:

- Um protótipo em papel ou um modelo computacional que mostra a interação do homem com a máquina, de tal forma que o usuário entenda com clareza a interação existente;
- Um protótipo de trabalho que implementa algumas funções que são exigidas pelo sistema desejado;

- Um programa existente que executa parte ou toda a função desejada para o novo sistema, mas com características que poderão ser melhoradas durante o desenvolvimento.

O processo de desenvolvimento conforme a prototipação acontece conforme apresentado na Figura 2:

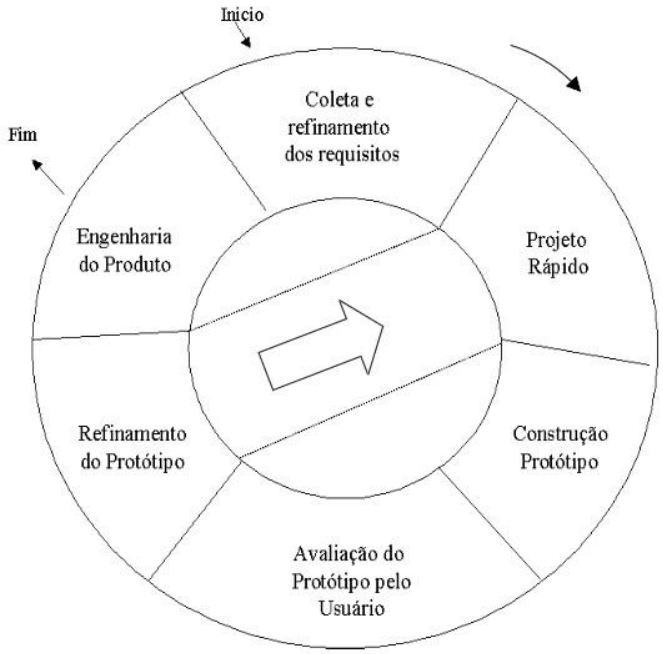


Figura 2 - Prototipação (PRESSMAN, 1995, p. 36)

Na Figura 2 cada fatia do círculo representa uma fase do processo, começando na fatia indicada por "início", seguindo no sentido horário (como indica a seta fora do ciclo). A seta no meio do círculo permite um retorno a fases anteriores, para correções necessárias. Segue cada fase do processo:

1. O processo se inicia com a **coleta e refinamento dos requisitos**, que é feita com o cliente.
2. Depois é feito um **projeto rápido**, que é construído através dos requisitos que foram coletados e refinados. Esse projeto rápido serve para ajudar e dar maior base para a construção do protótipo.
3. Depois é feita a **construção do protótipo**, o qual, sendo finalizado, é avaliado pelo usuário.
4. A partir da **avaliação do protótipo pelo usuário** é feito um **refinamento no protótipo**, levando em conta as considerações do usuário. Após o **refinamento do protótipo** pode-se voltar para o **projeto rápido** e realizar alterações necessárias.

5. “A **engenharia do produto** é encarregada do desenvolvimento e manutenção dos produtos e serviços de software” (VASCONCELOS, 2006, P. 15). É onde será implementado o que foi planejado nas outras fases, fazendo também a correções dos erros que forem encontrados.

2.2.3. Modelo Espiral

O Modelo Espiral é um dos modelos que surgiram para enfatizar o processo iterativo, pois “ao invés de representar o processo de *software* como uma sequência de atividades, a exemplo do Modelo Cascata, ele é representado através de um espiral, onde cada *loop* representa uma fase do processo de software” (SOMMERVILLE, 2007, p. 44).

“Este modelo foi desenvolvido para englobar as melhores características tanto do Modelo Cascata quanto da Prototipação” (MACORATTI, 2005, online). As características como Planejamento e Análise do cliente foram criadas para serem realizadas pelo dono do produto e são executadas até o final do processo de desenvolvimento. Foi acrescentada a fase Análise de risco, pra ter uma analise de cada funcionalidade. Caso, os riscos sejam muito grandes, o projeto é interrompido.

Na Figura 3 é mostrado o processo do Modelo Espiral.

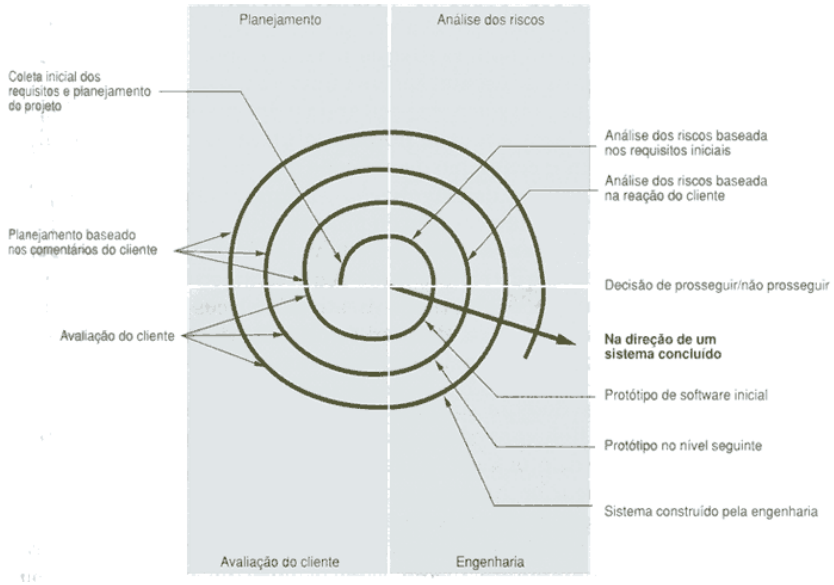


Figura 3 - Modelo Espiral (PRESSMAN, 1995, p. 36)

Na Figura 3, em cada quadrante cinza, estão as fases do modelo e os espirais representam o que vai acontecendo todas as vezes que se passa por cada quadrante. Assim, cada vez que a espiral volta a passar por um quadrante, tem-se um *loop*. O início se dá na ponta interna da espiral, no quadrante "Planejamento", e o término se dá na ponta mais externa da espiral, no quadrante "Engenharia". Segundo Pressman (1995, p. 32) as fases são as seguintes:

- **Planejamento**: determinação dos objetivos, das alternativas e restrições;
- **Análise dos riscos**: análise de alternativas e identificações/resoluções de riscos;
- **Engenharia**: desenvolvimento de produto no "nível seguinte";
- **Avaliação feita pelo cliente**: avaliação dos resultados da engenharia.

Todas as vezes que se realiza um *loop*, uma volta, a equipe precisa se questionar se deve ou não prosseguir. Caso existam muitos riscos, e seus efeitos não possam ser reduzidos, o projeto deve ser interrompido neste momento. "Cada giro ao redor do Espiral requer um trabalho de engenharia (quadrante inferior direito) que passa ser levado a efeito usando-se ou a abordagem de ciclo de vida clássico ou a prototipação" (PRESSMAN, 1995, p. 43).

2.3. Metodologia de Desenvolvimento Ágil

"A partir da década de 90, começaram a surgir novos métodos sugerindo uma abordagem de desenvolvimento ágil onde os processos adotados tentam se adaptar às mudanças, apoiando a equipe de desenvolvimento no seu trabalho" (BECK et al., 2001, online). A partir dessa época surgiram as metodologias ágeis como: TDD, *Scrum* e FDD.

Segundo Gomes (2009, pg. 17) as metodologias ágeis possuem algumas características como:

1. **Processo de desenvolvimento incremental e evolutivo** - Os processos menos ágeis tendem a fazer uma entrega total no final do projeto, enquanto os métodos ágeis promovem entregas constantes e incrementais do produto, promovendo a emergência de novos requisitos e novas soluções;
2. **Processo modular e simples** - Os processos devem ser simples e modulares de forma a poderem responder rapidamente para o desenvolvimento. Além disso, recomendam pequena produção de documentação;
3. **Orientado ao tempo** - Nos processos ágeis o planejamento é orientado a intervalos fixos de tempo (iterações) e não ao volume de esforço necessário.

"Em 2001, uma série de signatários juntou-se para lançar um manifesto, que impulsionou de forma significativa o movimento dos métodos ágeis, através do Manifesto para o desenvolvimento Ágil de Software" (AGILEMAN, 2001, online). São em torno de treze pessoas envolvidas à frente do manifesto, que traz dose princípios para que o processo de desenvolvimento seja ágil. Segundo Lapoll (2011, p. 52), resumidamente, um processo de desenvolvimento ágil deve ter as seguintes características:

- As mudanças de requisitos são bem recebidas, pois representam vantagens competitivas para os clientes;
- Devem existir entregas de software numa base semanal ou mensal.
- As pessoas de negócio e os desenvolvedores devem trabalhar conjuntamente e diariamente durante todo o projeto;
- O trabalho deve ser realizado por indivíduos motivados, sendo que para isso devem ser providenciadas as condições e o suporte necessários, bem como deve ser estabelecida uma relação de confiança;
- A forma mais eficiente de comunicação para e com o projeto é a conversação cara a cara;
- A principal medida de progresso é software que funciona;
- Os processos ágeis promovem um desenvolvimento sustentado. Os *sponsors*, desenvolvedores e utilizadores, devem estar habilitados para manter o mesmo ritmo de desenvolvimento de forma indefinida;
- Uma atenção contínua a bom desenho de soluções e excelência técnica são catalisadores da agilidade;
- A simplicidade é fundamental de forma a minimizar o trabalho a realizar;
- As melhores arquiteturas, requisitos e desenhos emergem de equipes que são autônomas, que se auto-organizam.

Além disso, segundo Gomes (2009, p.17), "em intervalos regulares as equipes refletem em como podem ser mais eficazes e eficientes ajustando os seus processos e métodos de acordo com as reflexões".

Depois do Manifesto Ágil surgiram as demais metodologias ágeis, dentre as quais serão apresentadas FDD e *Scrum*, a seguir.

2.4. Feature Driven Development

"Feature-Driven Development (FDD) é uma abordagem direta criada por Jeff De Luca e Peter Coad para desenvolvimento de software" (FILHO, 2008, pg. 151). É uma metodologia

iterativa e incremental, em que cada iteração corresponde ao desenvolvimento completo de uma funcionalidade. As iterações do FDD geralmente são planejadas para terminar em duas semanas, pois se ultrapassar este tempo, esta iteração deve ser dividida em mais partes. Segundo Filho (2008, p. 151) cada iteração deve apresentar um valor para o cliente, para que ele perceba como o projeto está evoluindo, ou seja, cada iteração deve prever a conclusão de uma parte do *software*, que deve ser entregue ou apresentada ao cliente. Esta característica propicia uma boa precisão na rastreabilidade de relatórios, monitoramento detalhado dentro do projeto, com resumos de alto nível para clientes e gerentes (HIGHSMITH, 2002, p. 30).

O FDD permite que sejam descritas as funcionalidades que o sistema possuirá, tendo a coleta de requisitos na sua primeira atividade.

Segundo Palmer e Felsing (2002, p. 31) o FDD consiste em um conjunto de cinco processos sequenciais, agrupados em dois conjuntos, e, segundo Bonow (2008, p. 30), provê os métodos, as técnicas e diretrizes necessárias para a construção e entrega de um sistema. Os dois conjuntos que agrupam os processos do FDD são: **Concepção e Planejamento** e **Construção**. Na Figura 4 são apresentados os cinco processos do FDD:

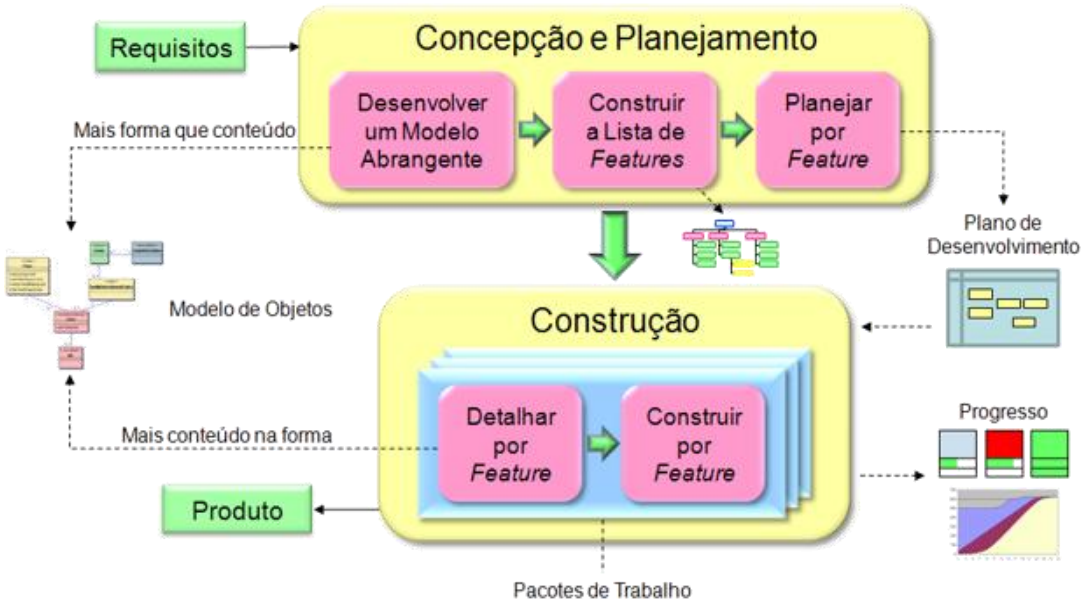


Figura 4 - Processo do FDD (RETAMAL, 2008, p. 2)

Conforme a Figura 4 pode-se perceber que primeiro são recolhidos os requisitos e, em seguida, há dois grupos de processos: **Concepção e Planejamento**, que engloba os processos relacionados ao projeto como um todo, portanto, acontece no início do desenvolvimento, e **Construção**, cujos processos são realizados para cada funcionalidade individualmente,

acontecendo também o rastreio do progresso por meio de relatórios. O grupo **Concepção e Planejamento** contém os processos: **Desenvolver um Modelo Abrangente** (DMA), **Construir a Lista de *Features*** (funcionalidades) (CLF) e **Planejar por *Feature*** (funcionalidade) (PPF). Enquanto isso, o grupo **Construção** contém os processos: **Detalhar por *Feature*** (DPF) e **Construir por *Feature*** (CPF). Na Figura 4 também é demonstrado que os grupos de processos compartilham alguns artefatos: o **Modelo de Objetos** (MO) é construído nos processos DMA e DPF, sendo incrementado do segundo para o primeiro ou a cada ida e volta entre os dois processos; o **Plano de Desenvolvimento** (PD) é resultado do processo PF e é fornecido como entrada para o grupo **Construção**. Cada processo e suas respectivas características serão apresentados em detalhes nas próximas seções deste trabalho.

2.4.1. Papéis

No FDD existem papéis que são distribuídos perante a equipe de desenvolvimento. Os papéis são:

- **Gerente do projeto**: "Responsável por todos os assuntos administrativos ligados ao projeto" (PURIFICAÇÃO, 2010, p. 42). Este papel faz o gerenciamento de todo o projeto, lidera a equipe de desenvolvimento e designa funções.
- **Especialista do negócio**: "Deve usar o conhecimento a respeito do negócio para apresentar à equipe de projeto as necessidades para que o software possa ter real valor para o cliente, deve fornecer o *feedback* sobre as entregas a todos os envolvidos" (PURIFICAÇÃO, 2010, p. 42). É o especialista do negócio que entende todo o produto que está sendo desenvolvido e sua presença é necessária para que o produto final vá de encontro ao desejado pelo cliente.
- **Arquiteto**: "Atua como um consultor da equipe em termos da arquitetura do sistema. Deve ser alguém com experiência em modelagem de objetos que tem a responsabilidade de guiar os desenvolvedores" (NETO, 2008, p. 19).
- **Gerente de desenvolvimento**: "Responsável por liderar o dia a dia do desenvolvimento do produto. Todos os conflitos técnicos entre os programadores-chefes estão sob sua responsabilidade" (PURIFICAÇÃO, 2010, p. 42). É ele que irá lidar diretamente com os programadores chefes, passando as instruções para o desenvolvimento.

- **Programador-chefe**: “Lidera pequenos grupos de desenvolvedores durante a construção das funcionalidades. Possui papel fundamental nas atividades de absorção do conhecimento do negócio e no planejamento das atividades” (PURIFICAÇÃO, 2010, p. 42). Lida com as dificuldades que aparecem dentro do desenvolvimento, orientando e acompanhando o desenvolvimento do projeto, para alcance do objetivo final.
- **Programador (*Class Owner*)**: Compõe as equipes de funcionalidades e realiza atividades de programação, criação de diagramas, testes e documentação de funcionalidades. Para projetos maiores ou complexos, FDD fornece um conjunto maior de papéis, como por exemplo: Gerente de *release* e testadores, (PURIFICAÇÃO, 2010, p. 42).
- **Especialista do Domínio**: “Responsável por prover todas as informações necessárias a respeito da área de domínio, visão geral e modelagem destes” (NETO, 2008, p. 19). Um exemplo é se o projeto tiver domínio acadêmico, ele terá que ser especialista da área acadêmica para oferecer informações sobre aquele domínio.

2.4.2. Desenvolver um Modelo Abrangente

O processo Desenvolver um Modelo Abrangente (DMA) objetiva fornecer à equipe uma compreensão do projeto como um todo. É a primeira etapa do desenvolvimento conforme o FDD. Segundo Leal (2007, p. 3) Modelo Abrangente é “um processo que abrange todo o projeto, sendo portanto executado apenas uma vez no início do mesmo”. Nesta abrangência do todo, o modelo contém uma visão do que será todo o projeto que estará em desenvolvimento. A seguir, a Figura 5 apresenta papéis e atividades do processo.

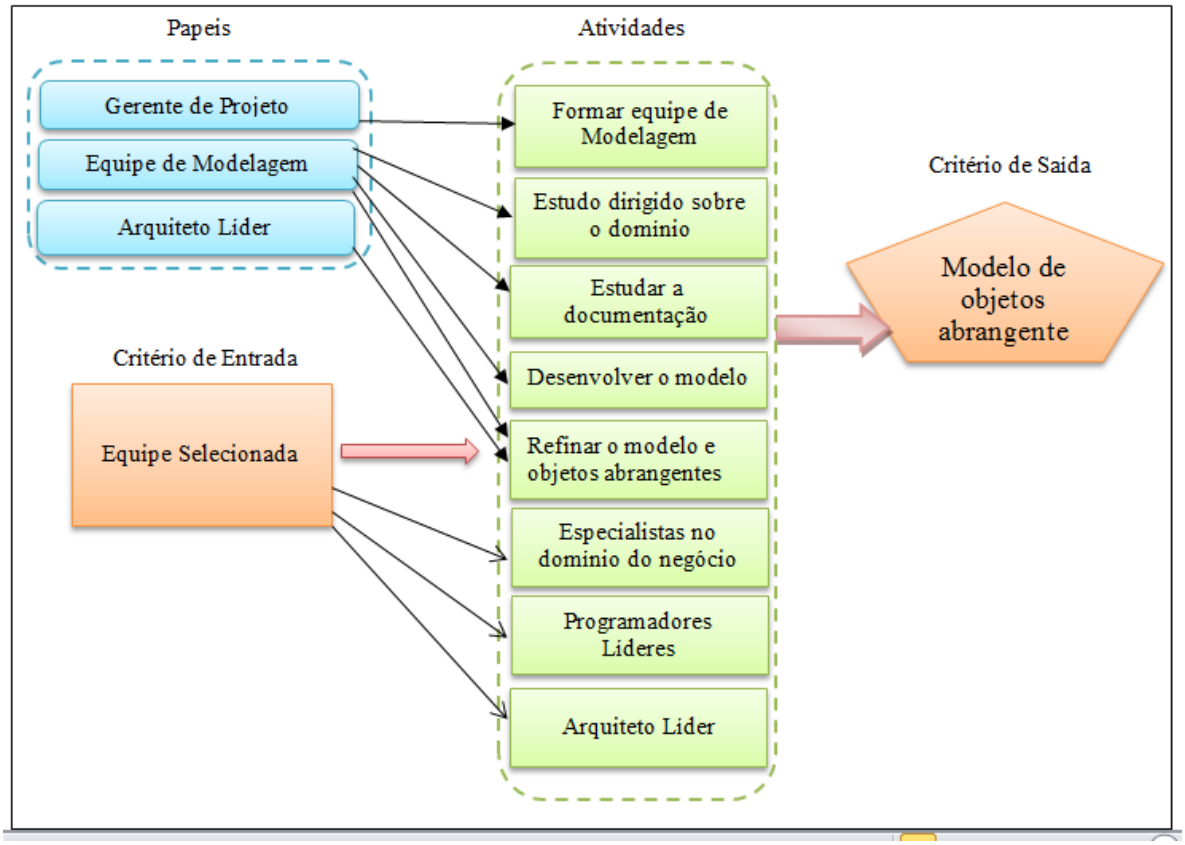


Figura 5 - Detalhamento do processo Desenvolver um Modelo Abrangente

A Figura 5 representa o detalhamento do processo DMA. Nesta fase participam os seguintes papéis:

- **Gerente do Projeto**, que é responsável por formar a equipe de modelagem;
- **Equipe de Modelagem**, cujas funções são estudar a documentação do Modelo de Objetos Abrangente (MOA) junto com o **Arquiteto Líder.**

Ao ser finalizado, o processo DMA tem como critério de saída o **Modelo de Objetos Abrangente**, o qual é composto por vários artefatos que, segundo Retanol (2008, p. 3), são os seguintes:

- Diagramas de classes com foco na forma do modelo;
- Métodos e atributos identificados são colocados nas classes;
- Diagrama(s) de sequência, se houver;
- Comentários sobre o modelo para registrar o motivo pelo qual uma forma de modelo foi escolhida;

No decorrer das próximas seções, serão apresentadas as atividades que ocorrem no decorrer deste processo.

Formar a equipe de modelagem

Segundo Bonow (2008, pg. 31) a atividade **Formar a equipe de modelagem** consiste em formar pequenos grupos com os membros da equipe de modelagem. Eles estudam os documentos que estiverem disponíveis como: modelos de objetos, requisitos funcionais, modelos de dados e guias do usuário. Cada grupo apresenta um modelo que deverá ser seguido pelo resto da equipe, escolhendo o melhor modo por consenso. Não deve haver somente uma equipe de modelagem, mas várias, cada uma com suas responsabilidades dentro do processo de desenvolvimento. Cada grupo fica responsável por uma determinada tarefa. No final, os resultados de cada tarefa serão estudados pelos membros de todas as equipes antes de seguirem para o próximo passo.

Para um bom andamento do projeto, segundo Retamal (2008, p. 2), as equipes de modelagem deverão ser compostas por membros permanentes das áreas do domínio do negócio e de desenvolvimento, especificamente os especialistas no domínio e os programadores líderes. É feito um rodízio com os outros integrantes do projeto através das sessões de modelagem, de modo que todos tenham a chance de participar e ver o processo em ação.

Estudo dirigido sobre o domínio

A atividade **Estudo dirigido sobre o domínio** é realizada pela equipe de modelagem, depois que os requisitos já foram recolhidos, no início do projeto. Um especialista no domínio do negócio, que é o cliente, apresenta uma visão geral da área do domínio que será modelada. Essa apresentação deve também incluir informação que estiver relacionada a esta área do domínio, mas não necessariamente uma parte de sua implementação. O gerente de projeto apresenta para cada membro das equipes de modelagem o que será feito naquela área, para que eles consigam entender o que vai haver naquele domínio. Neste processo, nenhuma funcionalidade é implementada (RETAMAL, 2008, p. 2).

Estudar a documentação

A atividade **Estudar a documentação** é uma etapa opcional. Os integrantes tentam entender em detalhes cada documento que está disponível caso haja dúvida. Exemplos de documentos que podem ser estudados: os documentos de referência ou de requisitos disponíveis, tais como modelos de objetos, requisitos funcionais (tradicionais ou no formato de casos de uso), modelos de dados e guias do usuário (RETAMAL, 2008, p. 2).

Desenvolver o modelo

Desenvolver o modelo é modelar o sistema criando classes sem conteúdo (atributos ou métodos com poucos detalhes), com os requisitos.

Segundo Retamal (2008, p. 3) no processo DMA formam-se grupos com até três componentes. Cada pequeno grupo comporá um modelo, sendo que um grupo se responsabiliza por uma parte do sistema e outro grupo por outras partes, com as quais tenha maior afinidade. Esta etapa ocorre depois que os grupos estudarem os documentos disponíveis (se existentes), para facilitar a criação do modelo. Nessa atividade, para escolha do melhor modelo, ocorrem os seguintes eventos:

1. O arquiteto líder pode propor um modelo base para facilitar o progresso das equipes, que têm a opção de segui-lo ou não;
2. As equipes se reúnem para a elaboração de um modelo abrangente;
3. Um membro de cada grupo apresenta o modelo proposto por seu grupo para a área do domínio;

Depois de ver todos os modelos a equipe de modelagem seleciona um modelo proposto ou compõe um modelo pela combinação das ideias propostas nos modelos apresentados.

Refinar o Modelo de Objetos Abrangente

Frequentemente o Modelo de Objetos Abrangente é atualizado com novas formas de modelo produzidas pelas iterações da atividade **Desenvolver o modelo**, descrita anteriormente (RETAMAL, 2008, p. 3).

Após a realização das atividades do processo DMA, espera-se obter um Modelo de Objetos Abrangente, representando as informações acerca do produto que será desenvolvido.

A partir desse modelo, pode-se passar para o próximo processo, que será abordado em mais detalhes nas próximas seções.

2.4.3. Construir a Lista de Funcionalidades

Segundo Leal (2007, p. 4) no processo **Construir a Lista de Funcionalidades** (CLF) "são utilizadas as partes do produto que foram identificadas no processo anterior, chamadas aqui de áreas de negócio, o grupo deve identificar as atividades dessas áreas e a partir dessas atividades, as funcionalidades que as constituem". A partir das áreas de negócio identificadas, será realizado um detalhamento, resultando em uma hierarquia de funcionalidades que representa o produto a ser construído (TEIXEIRA, 2007, p. 5).

Com base na divisão do domínio, a equipe quebra o domínio em um número de áreas. Cada área é ainda dividida em uma série de atividades (conjuntos de recursos). Cada passo dentro de uma atividade é identificado como um recurso. O resultado é uma hierarquia categorizada, que é a lista de funcionalidades (PALMER, 2002, p. 139, tradução nossa). A Figura 6 apresenta os papéis e as atividades do processo.

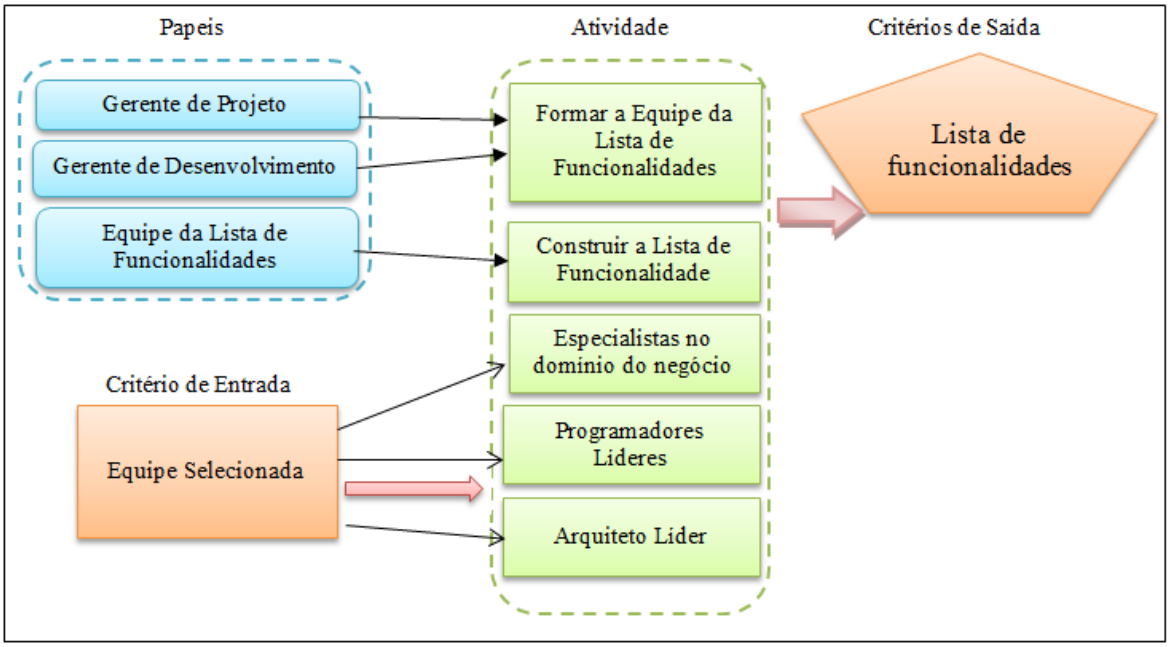


Figura 6 – Detalhamento do Processo Construir Lista de Funcionalidade

Na Figura 6 é apresentado o processo CLF. Os membros do projeto que participam desse processo são:

- **Gerente do Projeto** e **Gerente de Desenvolvimento**, que ficam responsáveis por formar a equipe da lista de funcionalidades; e
- **Equipe da lista de funcionalidades**, constituída pelos programadores líderes do processo anterior. Participam também os membros da equipe de modelagem e especialistas em negócios, para realizarem uma avaliação interna e externa das funcionalidades criadas.

Ao ser finalizado o processo CLF, o critério de saída é a Lista de funcionalidades. As próximas seções abordam as atividades que são realizadas ao longo deste processo.

Formar a equipe da lista de funcionalidades

Nesta atividade participam os programadores líderes, que compõem a equipe da atividade de modelagem do processo DMA (RETAMAL, 2008, p. 4).

Construir a lista de funcionalidades

A construção da lista de funcionalidades acontece através do que foi estudado e entendido no processo DMA. É nesta fase que a equipe de lista de funcionalidades separa as áreas estudadas anteriormente, repartindo o todo em partes menores.

As funcionalidades devem ser escritas seguindo uma nomenclatura que contém, nesta ordem: ação, resultado e objeto. Por exemplo na funcionalidade "calcular preço de venda de do produto de código 001": a **ação** é calcular; o **resultado** é o que será obtido do cálculo (o preço de venda); e o **objeto** é o produto do qual está se calculando o preço (RETAMAL, 2008, p. 5). Sendo assim, as funcionalidades são detalhadas para que possam ser construídas nas fases posteriores.

Segundo Palmer (2002, p. 139, tradução nossa), após a realização dessas atividades, espera-se uma lista de funcionalidade com:

- Uma lista de conjuntos de características principais (áreas);
- Para cada conjunto de características principais, uma lista de conjuntos de recursos (atividades) dentro dessa característica principal;
- Uma lista de recursos para cada conjunto de características (atividade), cada um representando um passo na atividade desse conjunto de recursos.

Após a construção da lista de funcionalidades, ocorre uma fase de planejamento do projeto, que será apresentada junto com suas respectivas atividades na próxima seção deste trabalho.

2.4.4. Planejar por Funcionalidade

No processo Planejar por Funcionalidade (PPF) "é planejada a ordem com que as funcionalidades serão desenvolvidas. É gerado um plano de desenvolvimento baseado na dependência e complexidade das funcionalidades, carga horária e disponibilidade da equipe" (BONOW, 2008, p. 31). Percebe-se que nessa fase do desenvolvimento as funcionalidades já são compreendidas. Sendo assim, é realizada uma organização de como irá ocorrer o desenvolvimento. Depois são atribuídas aos programadores-chefes responsabilidades sobre um conjunto de atividades de negócio. Os programadores-chefes ficam responsáveis por todas as funcionalidades envolvidas, como classes que devem ser construídas e manutenção de classes (LEAL, 2007, p. 4). A Figura 7 apresenta papéis e atividades do processo.

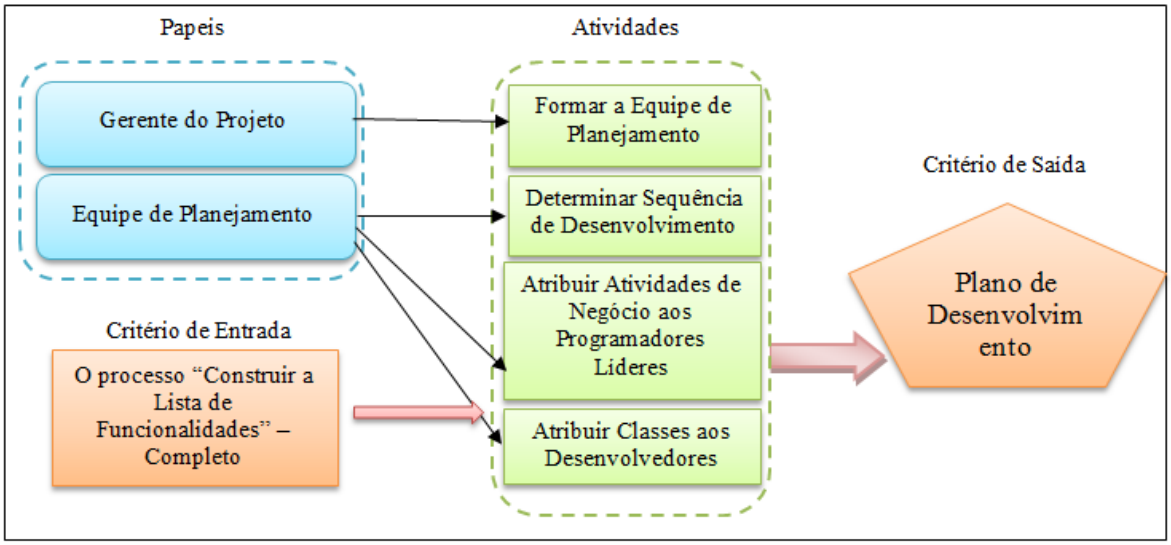


Figura 7 - Detalhamento do Processo Planejar por Funcionalidade

A Figura 7 apresenta o processo PPF. Neste processo, participam:

- **Gerente de Projeto**, responsável por formar a equipe de planejamento, que por sua vez, será composta pelos programadores líderes e gerentes.
- **Equipe de Planejamento**, que irá determinar a sequência de desenvolvimento, atribuir atividades de negócio aos programadores líderes e atribuir classes aos desenvolvedores. Ao ser finalizado, o processo PPF tem como critério de saída o

Plano de desenvolvimento. A seguir, são apresentadas as atividades que ocorrem no processo PPF.

Determinar Sequência de Desenvolvimento

Na atividade Determinar a Sequência de Desenvolvimento a Equipe de planejamento deve definir datas para o término de cada atividade de negócio. Conforme (RETAMAL, 2008, p. 6) para atribuir essas datas é preciso analisar os seguintes fatores:

- Dependências entre as funcionalidades em termos de classes envolvidas;
- Distribuição da carga de trabalho entre os proprietários das classes;
- Complexidade das funcionalidades a serem implementadas;
- Adiantamento das atividades de negócio de alto risco ou complexidade;
- Consideração de qualquer marco externo (visível) do projeto, como versões “beta”, demonstrações pontos de verificação e “todos os produtos” que satisfaçam tais marcos.

Atribuir Atividades de Negócio aos Programadores líderes

A atribuição de atividades de negócio é feita pela Equipe de planejamento, a qual escolhe os Programadores líderes para cada atividade de negócio. Segundo Palmer (2002, p. 67, tradução nossa) para fazer a escolha, analisam-se os seguintes fatores:

- Sequência de desenvolvimento;
- Dependências entre as funcionalidades em termos de classes envolvidas;
- Distribuição da carga de trabalho entre os proprietários das classes (deve-se lembrar que os programadores líderes também são proprietários de classes);
- Complexidade das funcionalidades a serem implementadas.

Atribuir Classes aos Desenvolvedores

A esquipe de planejamento atribui as classes aos desenvolvedores. Palmer (2002, p. 67, tradução nossa) indica que esta atribuição de classes é baseada nos seguintes fatores:

- Distribuição da carga de trabalho entre os desenvolvedores;
- Complexidade das classes;
- Uso das classes;

- Sequência de desenvolvimento.

Segundo Palmer (2002, p. 140, tradução nossa), para a saída do processo PPF, a Equipe de planejamento deve produzir um Plano de Desenvolvimento que esteja de acordo com o que esperam o Gerente do Projeto e o Gerente de Desenvolvimento. O Plano de Desenvolvimento consiste de:

- Conjuntos de atividades de negócio com datas de término (mês e ano);
- Áreas com datas de término (mês e ano), derivadas da data do último término de suas respectivas atividades de negócio;
- Programadores-chefes designados para atividades de negócio;
- A lista de classes com seus respectivos Proprietários.

Após a criação do Plano de Desenvolvimento, inicia-se a fase de **Construção**. Nesta fase nos dois processos (**Detalhar por Funcionalidade** e **Construir por Funcionalidade**) são feitas iterações para construções das funcionalidades. Os detalhes de cada processo e suas respectivas atividades serão apresentados nas próximas seções.

2.4.5. Detalhar por Funcionalidade

No processo **Detalhar por Funcionalidade** (DPF) as equipes detalham os requisitos e outros artefatos para a codificação de cada funcionalidade, incluindo os testes. O resultado do processo é composto pelo Modelo de Objeto Abrangente e pelos esqueletos de código (assinatura do código que possui os tipos de acesso dos métodos, tipo de retorno etc.) prontos para serem preenchidos (TEIXEIRA, 2007, p. 5).

Em seguida é feito, então, um diagrama de sequência relacionado àquela funcionalidade e é refinado o Modelo Abrangente, já incluindo métodos e atributos nas classes. Com essas informações, cada programador cria os prólogos de suas classes, com cabeçalhos de métodos com parâmetros, atributos e outros, não fazendo qualquer implementação (LEAL, 2007, p. 4). A Figura 8 apresenta papéis e atividades do processo DPF.

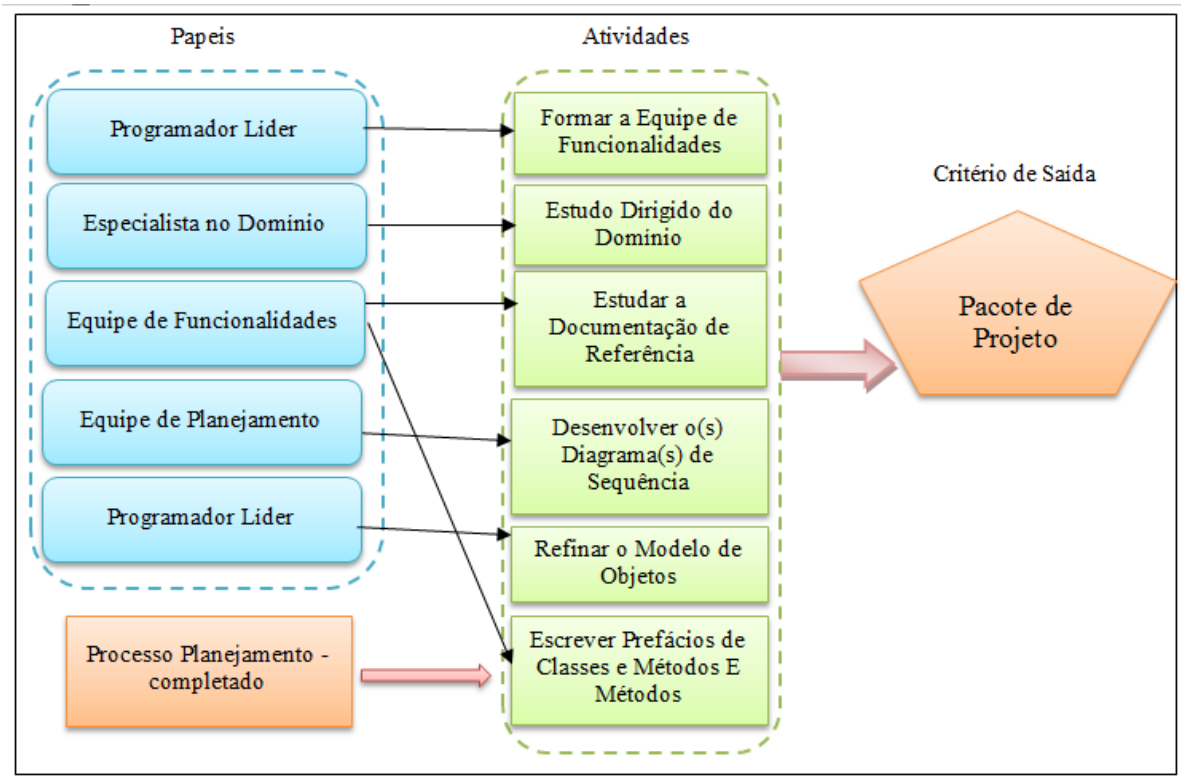


Figura 8 - Detalhamento do Processo de Detalhar por Funcionalidade (DPF)

A Figura 8 apresenta o processo DPF. Nesse processo, participam:

- **Programador Líder** define os **desenvolvedores** que farão parte da Equipe de Funcionalidades e refina o Modelo de Objetos;
- **Especialista de domínio** deve fazer o estudo dirigido do domínio;
- **Equipe de Funcionalidades** estuda os documentos referentes às funcionalidades e escreve prefácios (cabeçalhos contendo descrições) de classes e métodos;
- **Equipe de planejamento** deve criar os Diagramas de Sequência.

Ao ser finalizado, o processo DPF tem como critério de saída um Pacote de Projeto (*Design*) inspecionado com sucesso. A seguir, são descritas as atividades do processo PDF em mais detalhes.

Formar a Equipe de Funcionalidades

Nesta atividade "o programador líder identifica as classes que provavelmente serão envolvidas no projeto deste conjunto de funcionalidades e, consequentemente, atualiza o banco de dados de funcionalidade" (RETAMAL, 2008, p. 8). O Programador Líder, depois de identificar as classes faz a escolha dos programadores que serão responsáveis por cada uma. O

programador faz seu projeto (*design*) para complementar e ajudar com as funcionalidades (RETAMAL, 2008, p. 8).

Estudo Dirigido do Domínio

Esta é uma etapa opcional que é feita caso o Domínio que será implementado seja complexo. “O especialista no domínio apresenta uma visão geral da área do domínio para a funcionalidade a ser projetada” (RETANOL, 2008, p. 8).

Estudar a Documentação de Referência

“A equipe de funcionalidades estuda o(s) documento(s) de referência para a funcionalidade a ser projetada, todos os memorandos de confirmação, desenhos de telas, especificações de interface com sistemas externos e qualquer outra documentação de suporte” (RETAMAL, 2008, p. 9). Isto ocorre para entendimento e detalhamento de cada funcionalidade, para guiar o processo de desenvolvimento.

Desenvolver os Diagramas de Sequência

Nesta etapa são desenvolvidos os diagramas de sequência necessários para a funcionalidade, sendo os diagramas armazenados no repositório, o qual é criado para possuir o versionamento do que for produzido de cada funcionalidade e todos poderem ter acesso (PALMER, 2002, p. 139, tradução nossa).

Refinar o Modelo de Objetos

Assim como nas atividades anteriores nesta atividade ocorre um refinamento do Modelo de Objetos Abrangente. Nesta atividade, a atenção do programador líder é voltada principalmente para as classes correspondentes à funcionalidade que está sendo detalhada. Assim, é possível adicionar ou alterar classes e indicar atributos. Esse refinamento deve apresentar um alto grau de detalhamento, visto que na próxima etapa, a funcionalidade será, de fato, implementada pelos programadores. Esses arquivos devem ser submetidos ao sistema de controle de versões (PALMER, 2002, p. 140, tradução nossa).

Escrever Prefácios de Classes e Métodos

Nesta atividade são utilizados os arquivos-fontes que estão no repositório. A equipe de funcionalidade "escreve os prefácios de classe e métodos para cada item definido pela funcionalidade e pelo(s) diagrama(s) de sequência. Isto inclui tipos de parâmetros, tipos de retorno, exceções e mensagens" (RETAMAL, 2008, p. 9).

Segundo (PALMER, 2002, p. 70, tradução nossa) o resultado do processo DPF (após a realização de todas as atividades descritas nessa seção) é um Pacote de Projeto (*design*) inspecionado com sucesso. O pacote de projeto consiste em:

- Capa com comentários;
- Requisitos referenciados (se houver);
- Diagrama(s) de sequência;
- Alternativas de projeto (*design*) (se houver);
- Modelo de objetos com classes, métodos e atributos;
- Saída gerada para os prefácios de classes e métodos, criados ou modificados por esse projeto (*design*) e uma lista de tarefas e agendamentos para itens de ação nas classes afetadas para cada membro da equipe.

2.4.6. Construir por Funcionalidade (CPF)

No processo **Construir por Funcionalidade** (CPF) cada prefácio (esqueleto de código) que foi criado no processo anterior é preenchido, testado e inspecionado. O resultado é um incremento do produto integrado ao repositório principal de código, com qualidade e potencial para ser usado pelo cliente/usuário (TEIXEIRA, 2007, p. 5). É no processo CPF que será realizada a implementação de tudo que foi planejado e descrito nas fases anteriores. "É produzida uma função com valor para o cliente. Atividades como implementação de classes e métodos, inspeção de código, testes de unidade e atualização da versão, são atribuídas a esta fase" (BONOW, 2008, p. 32). A Figura 9 apresenta papéis e atividades do processo CPF.

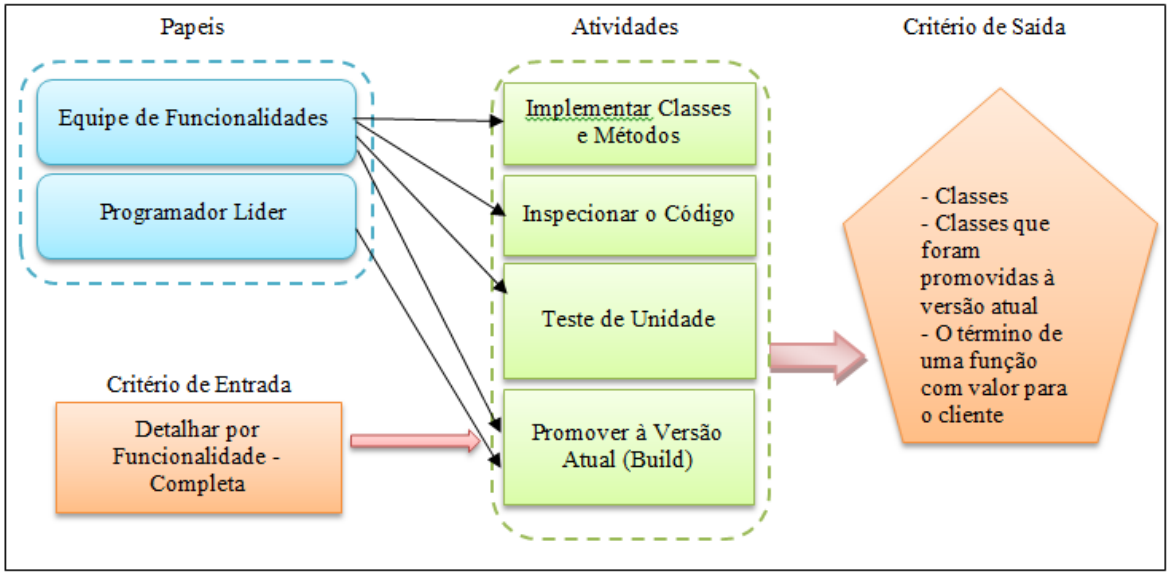


Figura 9 - Detalhamento do Processo de Construir por Funcionalidade (CPF)

A Figura 9 apresenta o processo **Construir por Funcionalidade** (CPF). Neste processo participa a **Equipe de funcionalidades**, que implementa as classes e os modelos, inspeciona os modelos e faz teste de unidade, além de promover a versão atual junto com o **Programador líder**.

Ao ser finalizado, o processo CPF tem como critério de saída **O término de uma função com valor para o cliente,** para que o cliente possa ver e ter acesso a parte do produto esperado. As próximas seções apresentam uma descrição das atividades que ocorrem nesse processo.

Implementar Classes e Métodos

Nesta atividade são implementadas as classes, de acordo o que foi planejado, satisfazendo ao que foi previsto pra cada classe funcionar.

Inspecionar o Código

A equipe de funcionalidades realiza uma inspeção de código, antes ou depois da atividade de Teste de Unidade. O programador líder decide se deve-se inspecionar dentro da equipe de funcionalidade ou com outros membros da equipe. (PALMER, 2002, p. 71, tradução nossa). Nessa verificação, podem-se contemplar alguns pontos, como padrões de

nomenclatura de métodos e variáveis, formatação, endentação do código e lógica utilizada (principalmente nos métodos mais complexos).

Teste de Unidade

O Teste de Unidade é realizado nas classes por seus próprios donos. O programador líder decide quais os níveis de teste para cada classe (RETAMAL, 2008, p. 11). São testados mais vezes os códigos que apresentarem maior complexidade.

Promover à Versão Atual (*build*)

"As classes somente podem ser promovidas para a versão atual (*build*) após uma inspeção de código com sucesso" (RETAMAL, 2008, p. 11). Esta atividade só é realizada depois de terem sido realizados todos os testes para se ter a comprovação de que o código está funcionando conforme o planejado nas fases anteriores.

O critério de saída do processo CPF é composto por (PALMER, 2002, p. 72, tradução nossa):

- Classe(s) e/ou método(s) que passaram pela inspeção de código com sucesso;
- Classe(s) que foi(ram) promovida(s) à versão atual (*build*);
- O término de uma função com valor para o cliente (funcionalidade).

O FDD é uma metodologia que trabalha com foco nas funcionalidades, de maneira que quando as funcionalidades são implementadas o desenvolvedor já as tem bem descritas e detalhadas.

O processo se divide em dois grandes processos: Concepção e Planejamento e Construção, sendo o primeiro se subdivide em Desenvolver um Modelo Abrangente, Construir a Lista de Funcionalidades e Planejar por Funcionalidade. E o segundo processo divide em Detalhar por Funcionalidade e Construir por Funcionalidade.

2.5. Scrum

O *Scrum* é também metodologia de desenvolvimento Ágil, que enfoca no gerenciamento do processo de desenvolvimento. Nesta seção serão seus papéis, artefatos e reuniões realizadas durante esta MDS.

O *Scrum* “é uma metodologia com o foco no gerenciamento da equipe, preocupada na organização dos processos, deixando a cargo dos participantes do projeto escolher a melhor maneira de concluir com sucesso as etapas do desenvolvimento” (SAVOINE et al., 2009). Ele visa à organização de cada membro da equipe de desenvolvimento, as atividades de cada um e o tempo previsto para a realização das mesmas.

O *Scrum* teve suas raízes no Japão, no início de 1986, por Takeuchi e Nonaka com gerenciamento de projeto, pois ele pode ser utilizando tanto pra gerenciar desenvolvimento *software* quanto qualquer outro tipo de projeto, mas começou a ser usado da forma que se conhece hoje apenas em 1996, por Jeff Sutherland e Ken Schwaber” (NETO, 2008, p. 10). Sutherland começou a estudá-lo e a refiná-lo até chegar à forma que ele é aplicado hoje. A composição do *Scrum* e suas características serão apresentadas nas próximas seções.

É natural que ocorram mudanças durante o desenvolvimento do *software*, por isso o *Scrum* permite que sejam realizadas adaptações de modo incremental e alterações nos requisitos durante o ciclo de vida do produto (SCHWABER & BEEDLE, 2002, p. 10). Esta flexibilidade é importante, principalmente, quando se está trabalhando com um contexto de alta complexidade, em que os requisitos podem sofrer alterações.

Segundo Ferreira (2005 apud Bissi, 2007, p. 4) as principais características do *Scrum* são:

- Um processo ágil para gerenciamento e controle de projetos;
- Um processo que controla o “caos”, resultante de necessidades e interesses conflitantes;
- Uma forma de maximizar a comunicação e cooperação de todos os envolvidos no projeto;
- Uma forma de detectar ou remover qualquer impedimento que atrapalhe o desenvolvimento do projeto;
- É escalar, pois pode ser utilizado para projetos pequenos e grandes em toda a empresa;
- Um *wrapper* para outras práticas, ou seja, pode ser associado a outras técnicas de engenharia de software.

Esta metodologia proporciona a comunicação entre os membros da equipe, por meio das reuniões, e assim permite que toda a equipe saiba das dificuldades e dos progressos que surgem no decorrer do projeto. A Figura 10 apresenta o ciclo de desenvolvimento do *Scrum*.

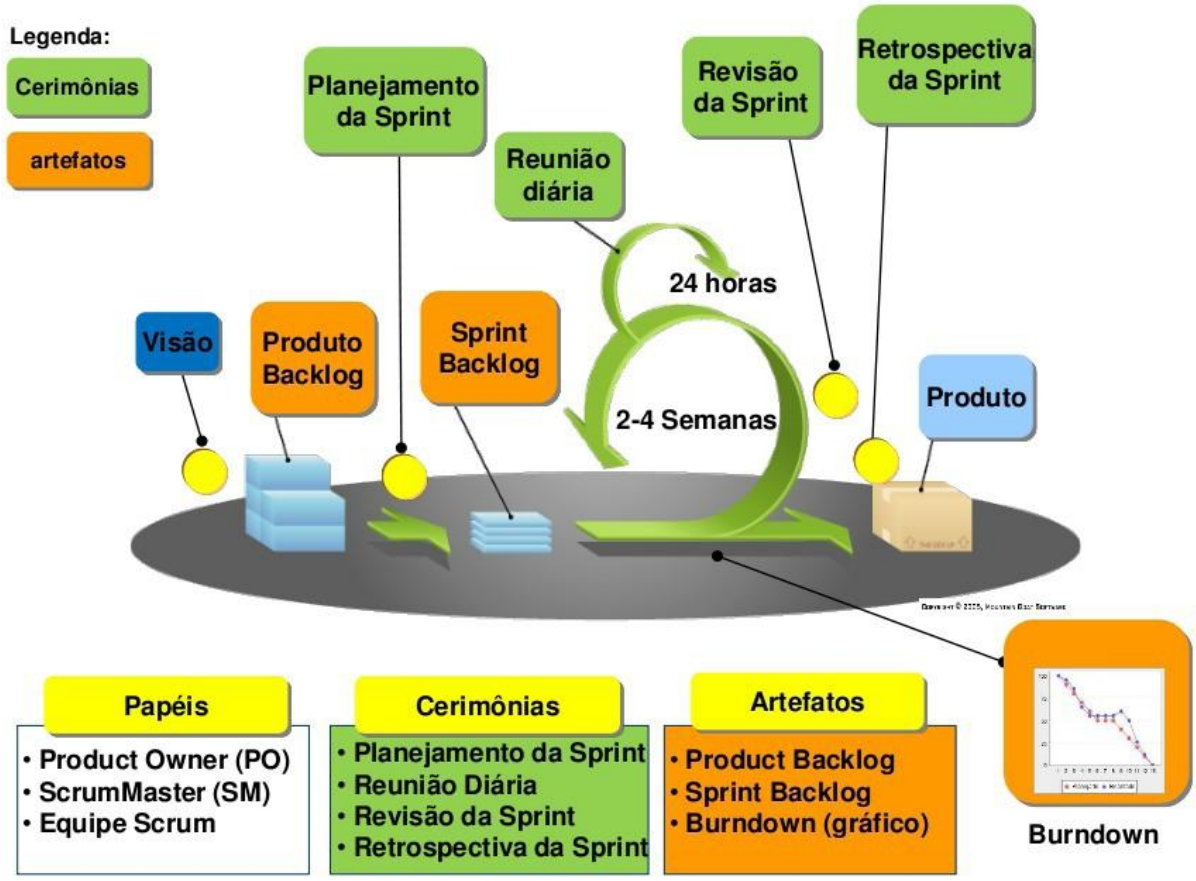


Figura 10 - Processo do *Scrum* (MOUNTAIN GOAT SOFTWARE, 2005)

A Figura 10 apresenta o ciclo de desenvolvimento do *Scrum*: primeiro se tem uma visão do produto, depois é elaborada uma lista de funcionalidade (o *Product Backlog*), depois é realizada uma reunião para o Planejamento do *Sprint* (ou iteração), quando é criado o *Sprint Backlog* (o *Product Backlog* dividido em vários conjuntos de funcionalidades). Durante a implementação de um *Sprint*, são feitas reuniões diárias pra saber o andamento do processo. Assim que o *Sprint* é finalizado, são realizadas outras duas reuniões: uma para Revisão do *Sprint* e outro para Retrospectiva do *Sprint*. Depois que forem implementados todos os *Sprints* obtém-se o produto final. Nas seções seguintes serão apresentadas informações sobre como uma equipe é gerenciada, isto é, como são atribuídos papéis e tarefas para cada membro da equipe de desenvolvimento e como são gerados os artefatos e realizadas as diversas reuniões propostas por esta metodologia.

2.5.1. Papéis e Responsabilidades

No *Scrum* estão definidos papéis e responsabilidades que cada membro da equipe deve assumir: *Product Owner* (o cliente), *Scrum Master* (o líder) e *Scrum Team* (a equipe de desenvolvimento). Uma das características destes papéis, que se assemelham a papéis encontrados em outras metodologias de gerenciamento de projetos, é a existência de um líder para gerenciar a equipe de desenvolvimento. Nas próximas subseções os papéis do *Scrum* serão apresentados em maior detalhe.

Product Owner

O *Product Owner* representa o cliente, pois ele deve saber o que o projeto precisa para alcançar o objetivo final, além de esclarecer qualquer dúvida referente ao projeto. Ele indicará quais funcionalidades deveram ser implementadas primeiro, observando o grau de importância para o projeto. Além do mais, o *Product Owner* segundo (SCHWABER & BEEDLE, 2002, p. 51):

- Gerencia o ROI (Retorno sobre Investimento, do inglês "*Return On Investiment"*), garantindo a lucratividade do produto ao aceitar/recusar os resultados do trabalho desenvolvido;
- É responsável por garantir o investimento no projeto, verificando se cada funcionalidade entregue está de acordo com o que o cliente espera e no tempo que foi designado. Assim, cumprido o que se espera, não há perda do ROI.

Scrum Master

Segundo Bauerle (2011, p. 20) o *Scrum Master* tem as seguintes atribuições:

- Gerência do projeto e da equipe;
- Fazer a comunicação entre o time e o *Product Owner;*
- Garantir a prática do *Scrum*, passando a ensinar o time como utilizá-lo;
- Remover os obstáculos que podem acontecer no decorrer do desenvolvimento;
- Garantir a realização das reuniões;
- Garantir a participação dos membros nas reuniões.

O *Scrum Master* não é o gerente do Projeto, entretanto ele garante que a equipe trabalhe da melhor maneira possível pra atingir o que se espera e ajuda a equipe a terminar o *Sprint* sem que ruídos externos atrapalhem.

Scrum Team

O papel *Scrum Team* é representado pela equipe que desenvolve o projeto. O time é composto por 5 a 10 pessoas. A equipe se organiza entre si, pois a organização é feita de forma que todos participem, o que é feito de maneira auto-gerenciada (pois não há um líder) e multifuncional. A divisão de papeis dentro do time é realizada conforme atribuições e por afinidade do membro do time com a funcionalidade a ser implementada. Além disso, a cada *Sprint* há um membro responsável (LEITÃO, 2010, p. 36). O *Scrum Master* garante que não haverá qualquer fato atrapalhando o *Scrum Team* e toma os procedimentos pra garantir o fim do *Sprint*. Segundo Caixeta (2011, p. 46) o *Scrum Team:*

- Desenvolve as funcionalidades do produto;
- Define como transformar o *Product Backlog* em incremento de funcionalidades numa iteração, gerenciando seu próprio trabalho, sendo responsáveis coletivamente pelo sucesso da iteração e consequentemente pelo projeto como um todo.

Os membros do *Scrum Team* são responsáveis por cada *Sprint* e participam das reuniões diárias (*Daily Meeting*) e notificam ao *Scrum Master* o progresso do desenvolvimento do produto.

2.5.2. Artefatos

No *Scrum* existem artefatos que irão auxiliar no desenvolvimento do projeto, os quais são resultado de uma tarefa que foi desenvolvida. Os artefatos do *Scrum* são apresentado a seguir.

Product Backlog

"O *Product Backlog* trata de uma lista de itens priorizados elencando o que deve ser desenvolvido no *Sprint Backlog* (explicado logo em seguida), associada a um valor de negocio, e que pode ser composta de requisitos funcionais ou não" (LEITÃO, 2010, p. 38). O *Product Backlog* permite ter maior controle do que se tem para fazer durante todo o processo.

“Esta lista pode sofrer alterações durante o decorrer do projeto dependendo da necessidade. Das funcionalidades definidas o dono do produto deve também definir a priorização dos itens da lista” (ALVES, 2010, pg. 4). Os exemplos das próximas seções representam, ficticiamente, artefatos de um projeto de desenvolvimento de um *software* de gerenciamento de vendas. A Figura 11 mostra um exemplo do *Product Backlog*.

	Itens do *Product Backlog*	**Estimativa (H)**
1	Usuário Cadastra	8
2	Usuário loga	4
3	Usuário Pesquisa Produto	4
4	Usuário Seleciona Produto	8
5	Carrinho de Compra	12
6	Pagamento do Usuário	20
7	Cadastro dos Produtos	4
...	...	

Figura 11 - Exemplo do *Product Backlog* de um site de vendas

A Figura 11 ilustra o *Product Backlog* de um projeto, ou seja, uma lista de funcionalidades que faz parte do produto que será produzido. Este conjunto de funcionalidades poderá ser agrupado em iterações (as *Sprints*), o que é resultado do uso do *Sprint Backlog*, que será visto na próxima seção.

Sprint Backlog

Sprint é cada iteração em que se divide o projeto. Se o projeto for divido em 4 etapas, cada um dessas etapas será uma *Sprint*. O projeto é desenvolvido a partir da realização de uma *Sprint* por vez, até que o projeto esteja concluído.

O *Sprint Backlog* contém a lista de funcionalidades do produto (o que está no *Product Backlog*) dividida em iterações. Para cada iteração a realizar, o *Sprint Backlog* lista as funcionalidades a serem implementadas de acordo com o planejado para aquela iteração. “Os itens do *Sprint Backlog* devem ser decompostos. A decomposição deve ser suficiente para que mudanças no progresso possam ser entendidas na *Daily Scrum*” (SUTHERLAND, 2010, pg. 20). A Figura 12 apresenta duas *Sprints* em um projeto.

		Itens do Product Backlog	Estimativa (H)
Primeiro Sprint	**1**	Usuário Cadastra	8
	2	Usuário loga	4
	3	Usuário Pesquisa Produto	4
Segundo Sprint	**4**	Usuário Seleciona Produto	8
	5	Carrinho de Compra	12

Figura 12 - Exemplo de *Sprint Backlog*

A Figura 12 mostra as funcionalidades que serão implementadas na primeira *Sprint* e, quando esta for concluída, começam a ser implementadas as funcionalidades que foram planejadas para segunda iteração.

Burndown Chart

O *Burndown Chart* "é um gráfico que, gerado a partir da lista de tarefas do *Sprint*, mostra o número de horas restantes para a conclusão do projeto. Por este artefato é possível visualizar a velocidade da equipe no Sprint" (ALVEZ, ano, pg.) A Figura 13 ilustra o *Burndown Chart*.

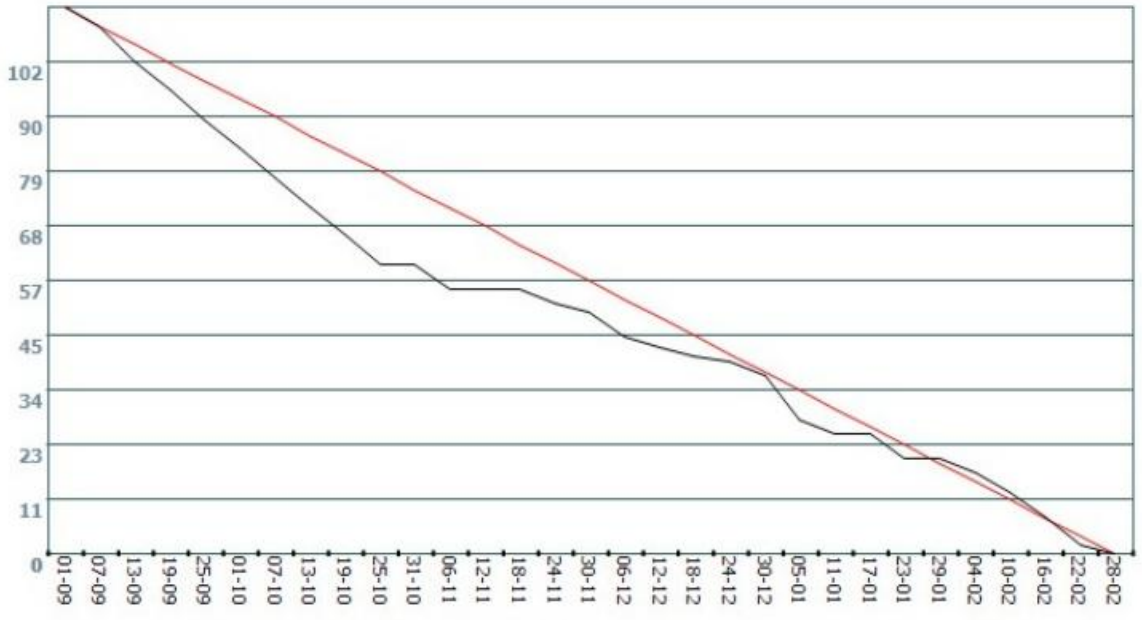


Figura 13 - **Exemplo de um *Burndown Chart* (PINTO, 2010, pg. 49)**

Na Figura 13 o eixo **y** representa as horas gastas, e o eixo **x**, os dias. A linha vermelha é o que se pretende atingir e a linha preta é o que o projeto vem atingindo no decorrer do desenvolvimento. Se a linha preta estiver abaixo da vermelha, o andamento do projeto está melhor que o tempo esperado, porque a equipe está desenvolvendo melhor do que havia planejado.

Taskboard

Taskboard é um quadro de tarefas que é apresentado para todo o grupo do *Scrum* durante o processo de desenvolvimento. Ele é dividido em quatro grupos: as tarefas a serem feitas, em andamento, as impedidas (que estão encontrando problemas na execução) e as finalizadas. O *taskboard* é usado tanto no planejamento da *Sprint* quanto no monitoramento e controle do andamento da *Sprint* atual (PACHECO, 2010, p. 8). A Figura 14 mostra um exemplo de um *Taskboard.*

Figura 14 - Figura Taskboard (PACHECO, 2010, p. 8)

A Figura 14 apresenta o quadro onde estão feitas divisões de tarefas. Dentro das divisões estão os itens do *Product Backlog*. Olhando para ele tem-se a noção de todo o processo que está ocorrendo dentro do *Sprint*, o que está ainda não foi feito, os impedimentos e o que está em andamento. Na próxima seção serão apresentadas as Reuniões que são feitas pra a realização dos Artefatos.

2.5.3. Cerimonial (Reuniões)

Cerimonial representa reuniões que são feitas durante todo o processo de desenvolvimento.
No S*crum* são feitas as seguintes reuniões:

- *Sprint Planning Meeting*
- *Daily Scrum Meetings*
- *Sprint Review Meeting*
- *Sprint Retrospective*

Sprint Planning Meeting

"Na cerimônia *Sprint Planning* são realizadas reuniões de planejamento entre o *Scrum Owner* (cliente) e o *Scrum Team* (equipe)" (BONOW, 2008, p. 24). Essas reuniões devem durar até 4 horas. A equipe e o *Scrum Owner* se reúnem pra definir as prioridades dos itens do *Product Backlog* que foram compilados no início, para começar a primeira *Sprint*. Também se discutem quais itens ficaram em maior ou menor prioridade dentro de uma iteração de acordo com a necessidade do cliente.

Esta é uma das principais práticas em *Scrum*: a equipe decide quanto trabalho que se comprometerá a realizar, ao invés de ter esse parâmetro atribuído pelo *Product Owner* (SCHWABER, 2007, p. 22, tradução nossa). A cada *Sprint*, a equipe começa a ter mais conhecimento do produto e a lista de itens do *Product Backlog* pode ter suas prioridades alteradas caso necessário.

Nesta etapa, para cada item que foi colocado no *Product Backlog*, também é feita a estimativa de tempo que será gasto com cada atividade. O item com tempo mais alto é trabalhado em conjunto, sendo dividido em tarefas individuais, cada qual com estimativas tempos menores (SCHWABER, 2007, p. 22, tradução nossa). O *Product Owner* estará com a equipe o tempo todo de produção para esclarecer dúvidas e o que mais for necessário.

Para fazer as estimativas de cada atividade existe a técnica *Planning Poker*, que é similar a um jogo de cartas. A Figura 15 apresenta as cartas que são utilizas.

Figura 15 - Exemplos das cartas do *Planning Poker* (CRISP, 2008, ONLINE)

A Figura 15 apresenta as cartas do *Planning Poker*. O *Planning Poker* é utilizado pela equipe de desenvolvimento para estimar quanto tempo vai ser gasto para a realização de uma tarefa e, juntamente com o *Product Owner*, tirar as dúvidas que surgirem durante esta atividade. As cartas seguem a sequência de *Fibonacci* (1, 2, 3, 5, 8, 13 ... 100) com cartas adicionais, como 0 (zero), meio (½) , o ponto de interrogação (?) e carta do café. As cartas 1 e o ½ são utilizadas para indicar que uma funcionalidade tem menor tempo que outra que já está estimada com o menor tempo. A carta 0 (zero) é utilizada para alguma funcionalidade que já foi feita ou cuja estimativa não é necessária. A carta com interrogação é utilizada quando não se consegue fazer estimativa para funcionalidade, por não se conhecer como ela poderá ser desenvolvida. A carta com o desenho do café representa uma pausa para o café. Todo o time levanta as cartas ao mesmo tempo e os membros da equipe que utilizaram a menor e a maior carta informam o porquê de terem escolhido tais cartas (COHN, 2007, p. 54, tradução nossa).

Daily Scrum Meetings

A *Daily Scrum Meeting* é uma das características predominantes do *Scrum.* É uma reunião diária, realizada durante 15 minutos com todos os membros que estão envolvidos no projeto. Durante a reunião, os participantes ficam em pé, e a mesma deve ser realizada, de preferência, em locais que comportem todos os membros da equipe de desenvolvimento (LEITÃO, 2010, p. 37). Os participantes da reunião precisam responder a três perguntas (SCHWABER, 2007, p. 13, tradução nossa):

- "O que fiz desde a última reunião?"
- "O que vou fazer até a próxima reunião?"
- "Estou tendo algum impedimento?"

O *Scrum Master* monitora a reunião e certifica que ela será realizada de forma adequada. Seu objetivo é descobrir quaisquer novas dependências, corrigir eventuais necessidades pessoais de indivíduos comprometidos e ajustar o plano de trabalho em tempo real para as necessidades do dia (SCHWABER, 2007, p. 13, tradução nossa). "As reuniões são um termômetro para o projeto, e quando mal utilizadas ocasionam graves problemas" (FIGUEIREDO, 2007, p. 18). Essas reuniões fazem com que o *Scrum Master* observe o que está acontecendo no desenvolvimento do trabalho e com que os membros da equipe possam buscar esclarecimentos e informar o quanto estão progredindo ou regredindo.

Sprint Review Meeting

Após o *Sprint* o cerimonial *Sprint Review Meeting* é realizado em duas etapas. Na primeira etapa os interessados no projeto se reúnem com o *Product Owner* para apresentar a *Sprint* finalizada, para ver se a o produto está saindo de acordo com o que se espera. "O *Product Owner* determina quais itens do *Product Backlog* devem ter sido concluídos no *Sprint*, e discute com a equipe a melhor forma de priorizar o *Product Backlog* para o próximo *Sprint*" (SCHWABER, 2007, p. 255, tradução nossa). Na segunda metade da reunião, o *Scrum Master* se reúne com a equipe de desenvolvimento para discutir as experiências, a maneira que o trabalho foi realizado e como pode ser melhor executado no próximo *Sprint*.

Sprint Retrospective

> A *Sprint Retrocpective* é uma reunião que acontece após o *Sprint.* É "uma revisão do *Sprint* que terminou, apontando pontos positivos e negativos que ocorreram e sugerindo mudanças e adequações para serem resolvidas no *Sprint* seguinte" (FILHO, 2008, p. 153). Esses pontos podem ser apresentados em um quadro e cada item do *Sprint* deve ter seu progresso relatado (SCHWABER, 2007, p. 26, tradução nossa).

Nesta reunião, que dura cerca de 3 horas, o *Scrum Master* incentiva a equipe a rever, no âmbito do processo e das práticas do *Scrum*, seu processo de desenvolvimento, a fim de para torná-lo mais eficaz e agradável para a próxima *Sprint* (LEITÃO, 2010, p. 42).

Durante esta reunião o time pode responder as seguintes perguntas (BISSI2007, p. 5):

1. "O que aconteceu de relevante durante a *Sprint* para a equipe?"
2. "Como é que cada um se sentiu?"
3. "O que podemos concluir disso?"
4. "O que pode ser aplicado para melhorar a próxima *Sprint*?"

Estas perguntas fazem com que o time não cometa erros nas próximas iterações e possa melhorar a cada *Sprint*.

No decorrer desta seção foram apresentados os conceitos sobre Metodologias Clássicas de Desenvolvimento de *Software*, que foram as primeiras metodologias a serem utilizadas por equipes de desenvolvimento de *software* e servem de base para outras metodologias mais recentes, como metodologias ágeis. A seção também apresentou as metodologias ágeis FDD e *Scrum*, que serão a base para a criação de uma metodologia de desenvolvimento de *software* para a Fábrica de Software do CEULP/ULBRA. Na próxima seção serão apresentados os materiais e métodos utilizados no desenvolvimento deste trabalho.

3 MATERIAIS E MÉTODOS

Nesta seção serão apresentados os recursos e métodos utilizados durante o desenvolvimento deste trabalho.

3.1. Local e Período

O desenvolvimento deste trabalho deu-se nas dependências do complexo de informática do CEULP/ULBRA, bem como em residência própria, ambos localizados na cidade de Palmas - TO. O período de desenvolvimento deste trabalho ocorreu durante o primeiro semestre de 2012, como parte das disciplinas “Trabalho de Conclusão de Curso I” e “Trabalho de Conclusão de Curso II”.

3.2. Materiais

Para que fosse possível a realização deste trabalho foram utilizados recursos próprios e outros adquiridos através da internet. Dentre os materiais utilizados no referencial teórico estão: artigos, trabalhos de conclusão de curso, dissertações de mestrado e doutorado e livros que abordam o assunto.

3.3. Métodos

A metodologia deste trabalho seguiu as seguintes etapas:

1. Revisão de literatura e aprofundamento dos conceitos necessários
2. Desenvolvimento
    a. Visão geral da Fábrica de Software do CEULP/ULBRA (FSW) e levantamento de requisitos
    b. Entendimento da metodologia utilizada atualmente na FSW
    c. Comparativo entre a metodologia da FSW e as metodologias ágeis FDD e *Scrum*
    d. Seleção de práticas do FDD e do *Scrum*
    e. Seleção de artefatos do RUP
    f. Definição da metodologia proposta: papéis, processos, atividades e artefatos.

Em um primeiro momento buscou-se um aprofundamento referente aos conceitos relacionados ao trabalho em questão, como: Metodologia de Desenvolvimento de *Software*,

Ciclo de Vida Clássico, Prototipação, Modelo Espiral, Metodologia de Desenvolvimento Ágil, metodologia FDD, *Scrum*, artefatos do RUP, as técnicas e os métodos utilizados nas metodologias citadas.

"O *Rational Unified Process* (também chamado de processo RUP) é um processo de engenharia de software. Ele oferece uma abordagem baseada em disciplinas para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento" (RUP, 1987, online). A Figura 16 apresenta as disciplinas do RUP.

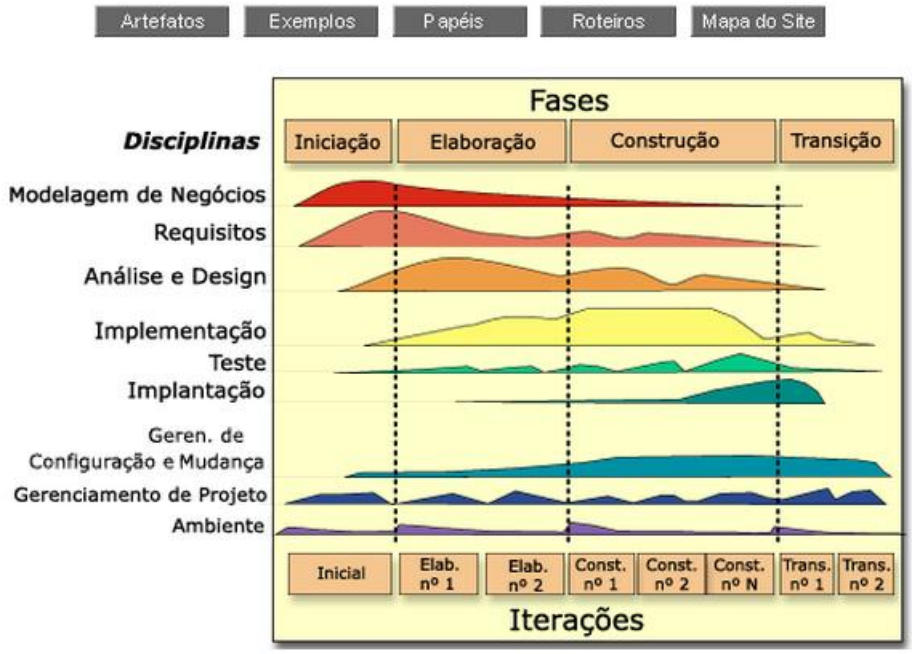


Figura 16 - Processos do RUP

A Figura 16 apresenta as fases e as disciplinas do RUP, sendo as fases: iniciação, elaboração, construção e transição, e as disciplinas:

- Modelagem de negócios;
- Requisitos;
- Análise e design;
- Implementação;
- Teste e Implantação;
- Gerenciamento de configuração e mudanças;
- Gerenciamento de projeto; e
- Ambiente.

Para cada disciplina são definidos atividades, papeis e artefatos a serem gerados durante o desenvolvimento do *software*.

Com o término dos estudos teóricos, foi dado início à fase de desenvolvimento da proposta da metodologia. Para que fosse possível a construção da proposta primeiramente foi realizada uma análise junto à Coordenação da FSW, na pessoa do professor Jackson Gomes de Souza, para obter um melhor conhecimento da realidade da FSW, onde será aplicada a metodologia.

Após a análise da realidade da FSW deu-se início à construção da proposta. Primeiramente foi feito um comparativo entre a maneira como ocorre o desenvolvimento da FSW atualmente e os processos das metodologias FDD e *Scrum*, para saber o que a FSW já estava utilizando das metodologias.

Em uma etapa seguinte, foram escolhidos os processos do FDD e *Scrum* que poderiam ser utilizados para a metodologia da FSW. Esses processos foram escolhidos para já se saber o que das metodologias estudadas seria aplicado, diante da realidade da FSW. Logo em seguida foram analisados artefatos da metodologia RUP, sendo selecionados os que poderiam ser utilizados em conjunto com o que já havia sido selecionado do FDD e do *Scrum*. Junto com a Coordenação da FSW foram atribuídas notas aos artefatos e foram escolhidos os que tiveram notas maiores (demonstrando a sua relevância para o contexto da FSW).

Por fim, foram definidos os papéis, processos e as reuniões, elementos inspirados nas metodologias FDD, *Scrum* e nos artefatos do RUP. A seguir, as definições de cada um dos elementos da metodologia:

- **Papéis**: Analista de Negócio, Gerente de Projeto, Equipe de Desenvolvimento, Arquiteto de Software, Equipe de Modelagem e Equipe de Teste.
- **Artefatos**: Objeto de negócio, Regra de negócio, Lista de Requisitos, Lista de Membros da Equipe de Desenvolvimento e seus Papéis, Plano de Reuniões, Lista de Funcionalidades, Plano de Iteração, Plano de Tarefas, Lista de Testes, Lista de Problemas, Plano de Resolução de Problemas e Correções, Documento de Arquitetura do Software, Documento de Modelagem do Software, Documento de Modelo de Dados e Resultado dos Testes.
- **Processos**: Grupo Planejamento: Entender, Planejar, Arquitetar; Grupo Construção: Projetar, Codificar e Testar.

Os principais processos foram estruturados com base na metodologia FDD, cujas atividades são realizadas por funcionalidade. Assim, todo o processo é desenvolvido por grupo de funcionalidades que pertencem a iterações. Quando cada iteração termina dá-se

início à outra. Foram definidas as reuniões, inspiradas nas reuniões do *Scrum*, que permitem que todos os membros da equipe saibam o que está acontecendo em todo o processo do desenvolvimento. No próximo capítulo a metodologia será apresentada em detalhes.

4 RESULTADOS E DISCUSSÃO

A partir da proposta de uma MDS para a FSW do CEULP/ULBRA, foram definidos os papéis, os processos, os artefatos e os cerimoniais necessários para o entendimento do projeto. As seções a seguir apresentam os passos produzidos durante o desenvolvimento da metodologia, assim como os resultados deste desenvolvimento.

4.1. Contexto Aplicado: Fábrica de Software do CEULP/ULBRA

Para se entender o contexto do trabalho, nesta seção são apresentadas informações coletadas junto à coordenação da FSW do CEULP/ULBRA por meio de entrevistas e reuniões.

Não há uma data específica para ser considerada como a data de criação da FSW do CEULP/ULBRA. O Portal Acadêmico do CEULP/ULBRA foi um dos primeiros projetos de software desenvolvidos por um grupo de alunos e professores do curso de Sistemas de Informação, em 2001. Com o passar do tempo, as necessidades internas do CEULP/ULBRA em relação a *software* aumentaram, principalmente no domínio da *web*. Desta forma, o Portal Acadêmico deixou de ser o único produto mantido pela equipe anterior, que sentiu a necessidade de formalizar seus processos. A partir de então, entre 2005 e 2008 iniciaram-se as atividades para a criação da FSW do CEULP/ULBRA. Atualmente, o Portal Acadêmico é mantido pela FSW, juntamente com outros produtos, apresentados mais adiante. A FSW tem o intuito de fabricar *software* tanto para o CEULP/ULBRA quanto para o público externo, pois isto é uma maneira de os alunos colocarem em prática os conhecimentos adquiridos em sala de aula. Os membros da FSW são alunos e professores, os quais fazem todo o processo de desenvolvimento de *software*.

A equipe da FSW é composta por coordenadores (atualmente, dois professores dos cursos de computação) e desenvolvedores (programadores e *web designer*). Há quatro funcionários (três programadores e um *web designer*) e um estagiário. Os programadores e o estagiário são alunos dos cursos de computação do CEULP/ULBRA, enquanto o *web designer* é aluno do curso de Comunicação Social - Publicidade e Propaganda. A lista a seguir apresenta os produtos já implantados (em produção e em manutenção constante pela equipe da FSW):

- **Central de Gerenciamento de Conteúdo (CGC)** -- *software* para gerenciamento de conteúdo dos sites do Portal Acadêmico do CEULP/ULBRA, dos cursos, de eventos e

sites em geral. Além do CGC (que tem o conteúdo dos sites), há o software para geração da apresentação, do visual ou da interface do usuário**.**

- **Sistema eventos** - *software* para gerenciamento de eventos acadêmicos;
- **Sistema ensino** - *software* para gerenciamento de informações acadêmicas do CEULP, como dados de cursos, professores, turmas, materiais didáticos etc.;
- **Intranet do CEULP** - *software* utilizado por alunos, professores e funcionários de setores acadêmicos do CEULP, como o Núcleo de Apoio Educacional (NAE) e a Assessoria da Direção.
- Portal Acadêmico e *Web sites* (com conteúdo gerenciado pelo CGC):

    - Portal Acadêmico do CEULP/ULBRA - integrado ao Sistema ensino;
    - Sites dos cursos - idem ao Portal Acadêmico do CEULP/ULBRA;
    - Sites de eventos e sites diversos.

No desenvolvimento desses produtos é utilizado um padrão de desenvolvimento que se encontra no seguinte formato:

1. Coleta de requisitos: usa um formato mais explicativo textualmente (modelo de Caso de Uso Expandido ou *User Stories*);
2. Desenvolvimento do protótipo (feito da maneira que deixar mais clara possível a interação entre usuário e sistema);
3. Criação do banco de dados;
4. Codificação da lógica de negócio e testes de unidade;
5. Criação da interface;
6. Realização de testes de uso do *software*;
7. Implantação;
8. Suporte ao usuário.

Segundo o modelo de trabalho atual da FSW, cada desenvolvedor recebe atribuições dos coordenadores da FSW e, dependendo de características do *software*, a tarefa de construí-lo pode ficar a cargo de um desenvolvedor ou de uma equipe. A tarefa de coleta de requisitos é realizada pelos coordenadores ou pelos desenvolvedores, conforme necessidades do projeto. Os coordenadores também se envolvem em tarefas de desenvolvimento, embora, na maioria das vezes, isto seja responsabilidade dos desenvolvedores. Este modelo de desenvolvimento, entretanto, ainda não é uma metodologia formalizada e devidamente testada, verificada e

avaliada. Além disso, há uma preocupação em relação à documentação e ao formalismo utilizado para modelar e descrever o *software*. Desta forma, o presente trabalho se propõe a formalizar e fornecer uma metodologia para a FSW, procurando utilizar boas práticas de metodologias ágeis como o FDD e o *Scrum*. A seguir serão apresentados os resultados do desenvolvimento da metodologia proposta.

4.2. Desenvolvimento da Metodologia Proposta

Com base nos estudos apresentados na Revisão Literária e na metodologia apresentada em Material e Métodos, iniciou-se o entendimento da realidade da FSW. Para dar início à criação da metodologia, foram feitos algumas análises e avaliações com o coordenação da FSW até se chegar no resultado final. Primeiro foi feito um comparativo entre o formato de trabalho da FSW e as metodologias estudadas, para verificar o já seria utilizado dos processos destas metodologias. Em seguida foram discutidas práticas do FDD e do *Srum,* para escolher quais poderiam ser aplicadas na FSW. Por fim foram escolhidos os artefatos do RUP que farão parte da metodologia, que será apresentada com mais detalhes nas próximas seções.

4.2.1. Comparativo das Metodologias FDD e *Scrum* com o formato atual da Fábrica de Software

Para primeira análise foi feito um comparativo entre os processos do FDD e do *Scrum* com o que a FSW utiliza atualmente para o desenvolvimento. Esta análise foi feita para se identificar o que já está sendo usado e o que se poderá acrescentar e/ou diminuir do FDD e do *Scrum* para, enfim, definir uma metodologia de desenvolvimento que seja mais adequada à realidade da FSW. É preciso esclarecer que o entendimento de “adequada à realidade” não significa um simples ajuste do FDD e do *Scrum* à forma de trabalho da FSW, mas, pelo contrário, uma nova proposta de trabalho para a FSW com base nas boas práticas do FDD e do *Scrum*. A Tabela 1 apresenta o referido comparativo.

Tabela 1 - Comparativo entre metodologia atual da FSW e as metodologias FDD E *SCRUM*

	FSW	**FDD**	***SCRUM***
1	Coletar requisitos	Desenvolver um Modelo Abrangente (DMA) Construir por Funcionalidade/Planejar por Funcionalidade (CPF/PPF)	*Product backlog*
2	Criação de Banco Dados	Detalhar por Funcionalidade	-
3	Criação da interface	Detalhar por Funcionalidade	-
4	Testes	Construir por funcionalidade	-
5	Implantação/entrega	Construir por funcionalidade	-

A Tabela 1 é analisada da seguinte forma:

- A atividade "criação de banco de dados" não está diretamente presente no FDD e no *Scrum*, mas no FDD, no processo DPF, o modelo de dados ou objetos é criado.
- Sobre a atividade "criação da interface", no FDD o *design* da interface é feito no processo DPF, mas não é explicado o nível de design, somente que é feito para ajudar os programadores a detalharem as funcionalidades.
- Do *Scrum* é utilizado somente um dos seus artefatos, o *Product Backlog*.

4.2.2. Práticas do FDD e do *Scrum*

Depois de feito o comparativo das metodologias FDD, *Scrum* e a metodologia de desenvolvimento da FSW, foi realizada uma segunda análise nos processos do FDD e do S*crum* para verificar o que de cada metodologia poderia ser aplicado na FSW. A Tabela 2 apresentando as fases do FDD e do *Scrum* que foram selecionadas (explicado em Material e Métodos) para serem aplicadas na FSW diante da sua realidade.

Tabela 2 - Seleção de práticas do FDD e *SCRUM*

Prática	**Metodologia**	***Objetivo/Resumo***
Formação da equipe de modelagem (processo DMA)	FDD	*Formar a Equipe para definir quem fará parte da modelagem*
Levantamento de requisitos (análise de requisitos) (processo DMA)	FDD	*Fazer lavamento de requisitos. Para definição dos requisitos e melhor entendimento do que o cliente espera.*
Revisão de análise de requisitos (processo DMA)	FDD	*Fazer uma analise dos requisitos que foram recolhidos na fase anterior, pra ver a viabilidade do que irá produzir.*
Estudar a documentação (processo DMA)	FDD	*Essa etapa é opcional, e nele estuda o tipo de documentação existente até o presente momento, para facilitar no entendimento do sistema.*
Desenvolver o modelo (fazer a modelagem) (processo DMA)	FDD	*Como já se tem uma equipe de modelagem definida, passa-se a desenvolver a modelagem.*
revisar a modelagem (processo DMA)	FDD	*Fazer uma revisão da modelagem, para ajuste e verificação de falhas.*
Construir a lista de funcionalidades (atividade Construir uma lista de funcionalidade)	FDD	*Fazer a lista de funcionalidades, com detalhamento das mesmas, fazendo uma hierarquia categorizando-as.*
Determinar sequência de desenvolvimento (processo PPF)	FDD	*Fazer a sequência que as funcionalidades serão implementadas, o tempo que se prevê para o termino de cada uma.*

Atribuir classes aos desenvolvedores (processo PPF)	FDD	*Atribui as classes aos desenvolvedores, baseando no termino e na complexidade de cada uma.*
Formar a equipe de funcionalidades (processo DPF)	FDD	*Identifica as classes que fazem parte do conjunto de funcionalidade a ser desenvolvido, escolhendo os programadores que são responsáveis pelas classes e fazendo também o design da mesma.*
Desenvolver o(s) diagrama(s) de sequênci**a** (processo DPF)	FDD	*Feito o diagrama de sequência, possibilitando ver iteração entre as camadas do sistema.*
Escrever prefácios de classes e métodos (processo DPF)	FDD	*Feito o esqueleto do código (incluindo tipos de parâmetros, tipos de retorno, exceções e mensagens)*
Implementar classes e métodos (processo CPF)	FDD	*Fazer a implementação de cada classe*
Inspecionar o código (processo CPF)	FDD	*Fazer a inspeção do código, como nomenclaturas, variáveis etc.*
Teste de unidade (processo CPF)	FDD	*Fazer o teste de unidade, antes ou depois da inspeção do código.*
Promover à versão atual (*build*) (processo CPF)	FDD	*Fazer a atualização da versão final*
Daily meeting	*Scrum*	*Fazer reuniões diárias ajuda a aumentar a comunicação entre a equipe e reduzir o tempo e resolução de conflitos*
Product Backlog	Scrum	*Listar todas as funcionalidades do sistema.*

Sprint Backlog	*Scrum*	*Product Backlog.*
Sprint Planning Meeting	*Scrum*	*Reunião pra definição das prioridades das funcionalidades no Product Backlog*
Sprint Retrospective	*Scrum*	*Reunião para ver os pontos positivos e negativos do Sprint atual, e as possíveis melhorias para os próximos.*

Depois da análise apresentada nesta seção foi estudada a metodologia RUP (*Rational Unified Process*), que serviu como inspiração para os artefatos do presente trabalho.

4.2.3. Artefatos do RUP

Foram estudados artefatos do RUP, pois são bem detalhados. O RUP especifica bem quem faz o artefato, o objetivo, apresenta os *templates*, as entradas para a realização da atividade etc. Depois desse estudo foi feita uma seleção dos artefatos que poderiam ser utilizados para metodologia em criação. A Tabela 3 apresenta os artefatos escolhidos do RUP (1996, online).

Tabela 3 - Artefatos Selecionados do RUP

Artefato do RUP	**Síntese**
Descrição do Negócio (objetivo e regra) - Modelagem de Negócio	Este documento define as Regras de Negócios aplicáveis ao negócio. Se for o caso, podem ser divididas em grupos de assuntos. A Visão do Negócio define o conjunto de metas e objetivos a que se destina o esforço de modelagem de negócios. Captura objetivos de alto nível de um esforço de modelagem de negócios.
Lista de Requisitos	Baseado nos processo de Requisitos
Lista de membros da equipe de desenvolvimento e papéis	Baseado nos processos do RUP
Plano de Reuniões	Baseado nos processos do RUP

Lista de Funcionalidades	Baseado nos processos do RUP
Plano de iterações - Gerenciamento de Projeto	Um conjunto de atividades e tarefas divididas por sequências de tempo, com recursos atribuídos e dependências de tarefas, para a iteração; um plano sofisticado.
Plano de Tarefas	Baseado nos processos do RUP
Plano de Testes - Testes	A definição das metas e dos objetivos dos testes no escopo da iteração (ou projeto), os itens-alvo, a abordagem adotada, os recursos necessários e os produtos que serão liberados.
Lista de Problemas - Gerenciamento de Projeto	A Lista de Problemas fornece ao Gerente de Projeto uma maneira de registrar e acompanhar problemas, exceções, anormalidades ou outras tarefas incompletas que requeiram atenção em termos de gerenciamento do projeto.
Plano de Resolução de Problemas e Correções - Gerenciamento de Projeto	O Plano de Resolução de Problemas descreve o processo usado para relatar, analisar e resolver problemas que ocorrem durante o projeto.

A Tabela 3 mostra os artefatos que foram selecionados do RUP com uma síntese de cada um. Os artefatos foram escolhidos atribuindo notas para cada um. Estas notas foram atribuídas de zero a cinco e, depois, foi feita outra análise para filtrar e ajustar ainda mais os artefatos à realidade da metodologia proposta no trabalho, considerando os artefatos que tiveram nota média acima de três. Depois, continuou-se mais uma análise dos artefatos, mesclando-os com o que já tinham sido escolhidos do FDD e *Scrum*, e definiram-se os artefatos finais da metodologia em desenvolvimento neste trabalho.

Como resultado de todas as análises feitas a próxima seção apresenta a MDS para a FSW.

4.3. Metodologia de Desenvolvimento de Software para Fábrica de Software

A MDS chama-se: Metodologia de Desenvolvimento Software para Fábrica de Software do CEULP/ULBRA. A seguir serão apresentados os Papéis, Processos, Artefatos e Cerimoniais.

4.3.1. Papéis

Os papéis que fazem parte da metodologia são:

- **Cliente:**
  O cliente é quem irá fazer a solicitação do *software*, ou seja, é o dono do *software*. É ele quem dirá o que vai querer no produto. O cliente será entrevistado pelo Analista de Negócio e irá tirar dúvidas da equipe de desenvolvimento, caso haja alguma no decorrer do desenvolvimento do *software*.
- **Analista de Negócio**
  O Analista de Negócio é o responsável por fazer a entrevista com o cliente com o objetivo de descrever o negócio e documentar estas informações para serem repassadas aos demais integrantes da equipe.
- **Gerente de Projeto**
  O Gerente de Projeto é o que fica responsável pelas atividades de gerência do projeto. Ele é quem designará as funções aos membros da equipe, comandará as reuniões e tomará as decisões que surgirem no decorrer do processo.
- **Equipe de Desenvolvimento**
  A Equipe de desenvolvimento é quem vai implementar as funcionalidades do *software*, além de participar de reuniões e atividade que forem a ela designadas.
- **Arquiteto de Software**
  O Arquiteto de Software é o responsável por definir e documentar a arquitetura do *software*.
- **Equipe de Modelagem**
  A Equipe de Modelagem é quem ficará responsável criar e documentar a modelagem do *software*.
- **Equipe de Teste**
  A Equipe de Teste é responsável por fazer os testes no *software*, depois de as funcionalidades estarem implementadas.

Na próxima seção são apresentadas as definições, as principais características e os artefatos dos processos criados para a metodologia proposta no presente trabalho. Vale ressaltar que os artefatos serão detalhados na seção 4.3.3.

4.3.2. Fluxo da Metodologia

A Figura 17 apresenta o processo da referida metodologia.

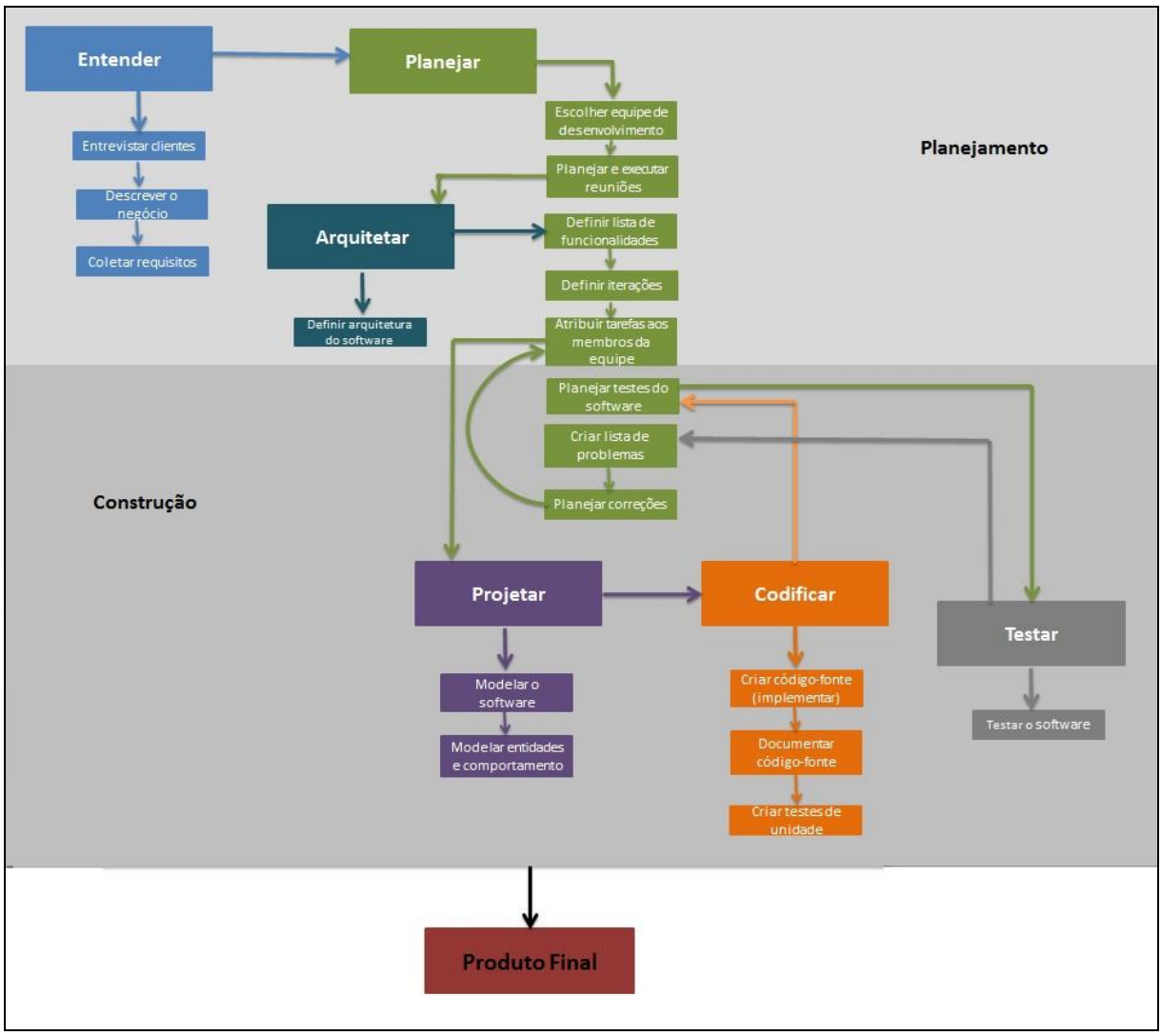


Figura 17 - Processo da Metodologia de Desenvolvimento de Software para a Fábrica de Software do CEULP/ULBRA

A Figura 17 apresenta todo o processo da metodologia. Para iniciar, há dois grupos de processos: **Planejamento** e **Construção.** No grupo **Planejamento** são realizados os planejamentos que englobam o projeto antes de iniciar as fases de implementação. Os processos do grupo **Construção** são realizados por funcionalidade, a exemplo do FDD.

No grupo **Planejamento** há os processos:

1. **Entender**, no qual são realizadas as atividades: Entrevistar clientes, Descrever o negócio e Coletar requisitos. A Figura 16 não apresenta setas de retorno para esta fase, mas é possível retornar a ela quantas vezes for necessário, a partir de

qualquer fase ou atividade. O objetivo deste retorno é o entendimento do negócio e de suas características e regras, sempre buscando atender aos objetivos do cliente com o *software* que está sendo construído.

2. **Planejar**, que tem algumas das suas atividades realizadas no grupo **Planejamento**: Escolher equipe de desenvolvimento, planejar e executar reuniões, definir lista de funcionalidade, definir interações e atribui tarefas aos membros da equipe. Outras atividades deste processo são realizadas no grupo **Construção**.
3. **Arquitetar**, na qual é realizada a atividade de definir a arquitetura do *software*. É importante ressaltar que este processo ocorre em conjunto com o processo **Planejar**.

O grupo **Construção** se inicia quando termina a atividade Atribuir tarefas aos membros da equipe, do processo **Planejar**. A partir de então são realizadas outras fases e atividades dentro deste grupo, sempre por funcionalidade:

1. **Projetar**, na qual são realizadas as seguintes atividades: modelar o *software* e modelar entidade e comportamento.
2. **Codificar**, na qual são realizadas as seguintes atividades: criar código-fonte (implementar), documentar código-fonte e planejar testes unidades.
3. **Planejar**, que tem sua execução continuada, com a realização da atividade Planejar testes do *software*.
4. **Testar**, na qual é realizada a atividade Testar *software*.
5. **Planejar**, com sua execução continuada, com a realização das atividades: criar lista de problemas e planejar correções.

Ao serem finalizados os processos e as atividades do grupo **Construção**, pode-se retornar ao processo **Planejar,** para ser executada novamente a atividade Atribuir tarefas aos membros da equipe. Desta forma, as fases que representam a construção do *software* podem ser executadas novamente, sempre por funcionalidade, seja considerando novas funcionalidades ou correções e soluções de problemas.

Ao término do grupo **Construção** tem-se o produto pronto para ser implantado. Este processo iterativo garante que, ao ser finalizado o grupo **Construção** todas as funcionalidades do produto terão sido devidamente construídas, testadas e, eventualmente, corrigidas. Nas próximas seções cada processo será apresentado com maiores explicações.

Processo Entender

O primeiro processo da metodologia é **Entender**, que faz parte do grupo **Planejamento**. É neste processo que se tem o primeiro contato com o cliente para se entender o domínio e quais os objetivos e as regras do negócio. Não há nada específico, neste sentido, nas metodologias FDD e *Scrum*. Entretanto, existe a entrevista com o cliente no *Scrum*, que tem o *Product Owner* como "cliente", que também responde eventuais dúvidas do modelo do negócio e define as funcionalidades. A Figura 18 apresenta o processo **Entender** em detalhes, com suas atividades, seus papéis e artefatos.

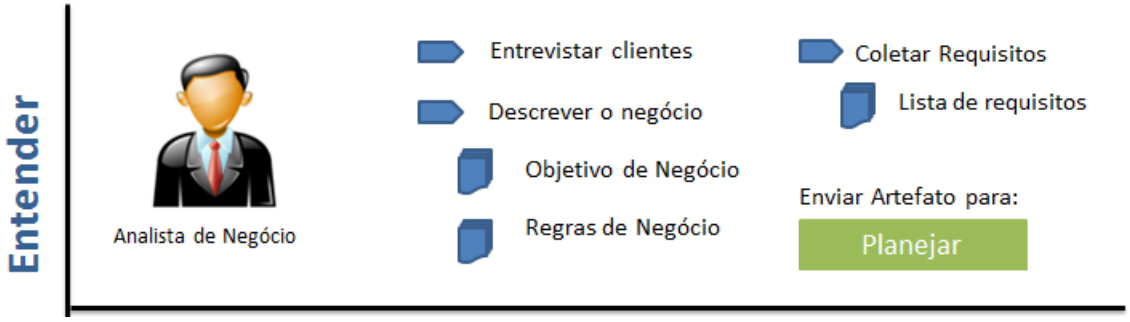


Figura 18 - Processo Entender

A Figura 18 apresenta o papel de quem realiza as atividades do processo **Entender**: o Analista de Negócio. É este papel quem executa as atividades realizadas nesta fase, que são:

- Entrevistar clientes
- Descrever o negócio
- Coletar requisitos

Ao serem finalizadas as atividades os artefatos serão critérios de entrada para o processo **Planejar**. A atividade Descrever negócio gera os artefatos Objetivo de Negócio e Regras de Negócio. A atividade Coletar Requisitos gera o artefato Lista de Requisitos. O processo **Entender** pode ser realizado quantas vezes forem necessárias para que a equipe de desenvolvimento colete a maior quantidade de informação possível para entender o negócio. A seguir serão apresentados detalhes de cada atividade.

Entrevistar clientes

Na atividade Entrevistar clientes o Analista de negócio deve marcar entrevistas com clientes (proprietários ou responsáveis pelo negócio). O objetivo desta atividade é estabelecer um alto nível de diálogo com o cliente desde o início do desenvolvimento, para que seja possível responder possíveis dúvidas da equipe de desenvolvimento em relação aos processos de negócios do cliente. A metodologia proposta não define uma estratégia ou uma metodologia específica para entrevista com cliente, mas é desejável que o Analista de negócio possua características e habilidades para boa comunicação com o cliente.

Descrever o Negócio

O Analista de Negócio, depois de ter feito a entrevista com o cliente e ter recolhido os dados necessários, executa a atividade Descrever o negócio, na qual irá detalhar as características do negócio, como: uma **visão geral**, que é natureza do negócio, hierarquia e ramo de atuação, **objetivos** do negócio ao utilizar o *software*, e **regras** características do comportamento do negócio, como o funcionamento de determinada atividade ou determinado processo do negócio. Ao finalizar essa atividade são gerados os artefatos: Objetivo de Negócio e Regras de Negócio, que serão critérios de entrada pra o processo **Planejar**.

Coletar requisitos

Com base na entrevista com o Cliente, na atividade Coletar requisitos, o Analista de Negócio descreve os requisitos do *software* em desenvolvimento. Ao término da descrição do negócio, é gerado um artefato Lista de requisitos, que será critério de entrada para o processo Planejar.

Processo Planejar

O segundo processo é **Planejar**, o qual se inicia quando termina o processo **Entender** e faz parte do grupo **Planejamento.** Neste processo são feitos os planejamentos antes de executar qualquer outro processo. É neste processo que ocorrerem atividades de planejamento de reuniões, distribuição de tarefas, planejamento de testes etc. A Figura 19 apresenta o processo **Planejar** em detalhes, com suas atividades, seus papéis e artefatos.

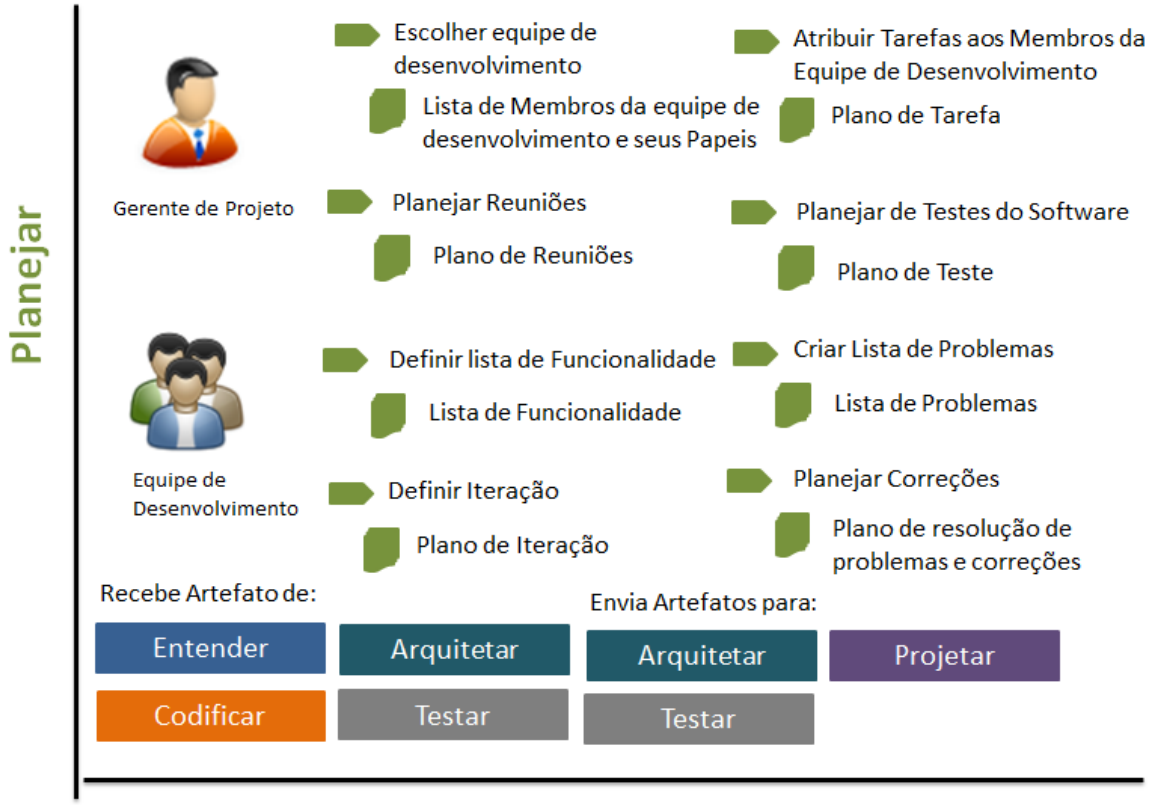


Figura 19 - Processo de Planejar

A Figura 19 apresenta os papéis que executam as atividades do processo **Planejar**: Equipe de Desenvolvimento, que realiza a atividade Definir funcionalidade, juntamente com o Gerente de projeto, que realiza as outras atividades:

- Escolher Equipe de Desenvolvimento
- Planejar e Executar Reuniões
- Definir Lista de Funcionalidade
- Definir Iteração
- Atribuir Tarefas a equipe de Desenvolvimento
- Planejar Teste de Software
- Criar Lista de Problemas
- Planejar Correções

Algumas atividades, antes de serem realizadas, precisam que outros processos estejam finalizados:

- Para se iniciar o processo **Planejar** é preciso que o processo **Entender** esteja finalizado.
- A atividade Definir Funcionalidade acontece em paralelo com o processo **Arquitetar** (será explicado posteriormente).
- A atividade Planejar Testes de Software precisa que o processo **Codificar** (será explicado posteriormente) esteja finalizado.
- A atividade Criar Lista de Problemas só é realizada depois que o processo **Testar** (que será visto adiante) for finalizado.

Os artefatos do processo **Planejar** serão critérios de entrada para os processos: **Arquitetar**, **Projetar** e **Testar**. Este processo possui diversas atividades, como apresentado, e o processo **Planejamento** tem uma interdependência com os demais, por exemplo, para que o processo **Projetar** aconteça a atividade Atribuir tarefas aos membros da equipe, do processo **Planejar**, deve já ter sido finalizada. A seguir as atividades deste processo serão apresentadas com mais detalhes.

Escolher equipe de desenvolvimento

A atividade Escolher equipe de desenvolvimento é a primeira a ser realizada. O Gerente de Projeto escolhe a equipe que irá fazer parte do desenvolvimento e atribui seus papéis dentro do processo. A atividade Escolher Equipe de Desenvolvimento é baseada no FDD – com base nas atividades Formar equipe de Modelagem (DMA), formar Equipe de lista de funcionalidade (CLF), Formar Equipe de Planejamento (PPF) e Formar equipe de Funcionalidade (DPF). Ao finalizar esta atividade será gerado o artefato Lista de Membros da equipe de desenvolvimento.

Planejar reuniões

Na atividade Planejar reuniões o Gerente de Projeto estabelece um cronograma de reuniões com os clientes e com a equipe de desenvolvimento. A atividade Plano de Reuniões é baseada no *Sprint Review Meeting*, do *Scrum*, que faz a apresentação do *Sprint* que foi finalizado para o "cliente". As demais reuniões relacionadas a esta atividade foram também inspiradas em reuniões do *Scrum*.

A lista a seguir contém um conjunto inicial e obrigatório de reuniões (as quais serão vistas na seção posterior). Outras reuniões podem ser realizadas, além destas:

i. ***Reunião de abertura***:
ii. ***Reunião com clientes***:
iii. Reunião de entendimento do negócio e homologação dos requisitos:
iv. ***Reunião de homologação das funcionalidades***:
v. ***Reunião de apresentação das iterações e das tarefas***:
vi. ***Reunião com Clientes para apresentação das funcionalidades****:*
vii. ***Reunião de apresentação da arquitetura do software***:
viii. ***Reunião de apresentação da modelagem***
ix. ***Reunião de apresentação do protótipo***:
x. ***Reunião de homologação da implementação***:
xi. ***Reunião com o Cliente para apresentação preliminar do software***:
xii. ***Reunião para apresentação do plano de testes***:
xiii. ***Reunião de correções de testes***:
xiv. ***Reunião de fechamento***:

Ao término dessa atividade tem-se o artefato Plano de Reuniões.

Definir Lista de Funcionalidade

Nesta atividade o Gerente de Projeto participa junto com a Equipe de Desenvolvimento. O Gerente de Projeto e a Equipe de desenvolvimento se reúnem para definir e selecionar as funcionalidades a serem implementadas no desenvolvimento do *software*. A atividade Definir Lista de funcionalidade é inspirada na atividade Construir lista de funcionalidades, do FDD, no processo CLF. Além disso, é inspirada no *Product Backlog*, do *Scrum*.

A técnica de *Planning poker* é utilizada como forma de medir e estimar a complexidade de cada funcionalidade. Ao terminar a atividade, é gerado o artefato Lista de Funcionalidade.

Definir Iterações

Nesta atividade o Gerente de Projeto projeta as iterações, ou seja, as funcionalidades são divididas em iterações. Esta atividade é baseada na atividade Determinar sequência de

desenvolvimento, do processo FDD e do *Sprint Backlog*, no Scrum. Ao concluir esta atividade é gerado o artefato Plano de iteração.

Atribuir Tarefas aos membros da equipe de desenvolvimento

O Gerente de Projeto atribui tarefas aos membros da Equipe de desenvolvimento e cria o Plano de tarefas. As estimativas para execução das tarefas devem levar em conta a complexidade "votada" pela equipe de desenvolvimento no *Planning poker*. Esta atividade é baseada na atividade Atribuir Atividade de Negócio, do processo PPF, e na atividade Atribuir Classes aos Desenvolvedores, ambas do FDD. Ao ser concluída esta atividade gera o artefato Plano de Tarefas.

Planejar testes do software

Depois de o processo **Codificar** (será explicado posteriormente) ser finalizado o Gerente de teste cria um Plano de testes do *software*. Ao finalizar a atividade é gerado o artefato Plano de teste.

Criar lista de problemas

Depois de o processo **Testar** ser finalizado é executada a atividade Criar lista de problemas. O gerente de projeto cria uma lista com os problemas levantados pela Equipe de teste. Ao término da atividade é gerado o artefato Lista de Problemas.

Planejar correções

Com os Testes de Software e a Lista de problemas finalizados, o Gerente de projeto cria o artefato Plano de Correções, para corrigir as falhas encontradas ao serem executados os testes.

Arquitetar

O processo **Arquitetar** é feito em paralelo com as duas primeiras atividades do processo **Planejar**. É neste processo que serão definidos os componentes do *software*, seu relacionamento com outros componentes de *software* e suas propriedades internas e externas, como plataforma de desenvolvimento, linguagem e arquitetura de camadas. A Figura 20 apresenta o processo **Arquitetar** em detalhes, com suas atividades, seus papéis e artefatos.

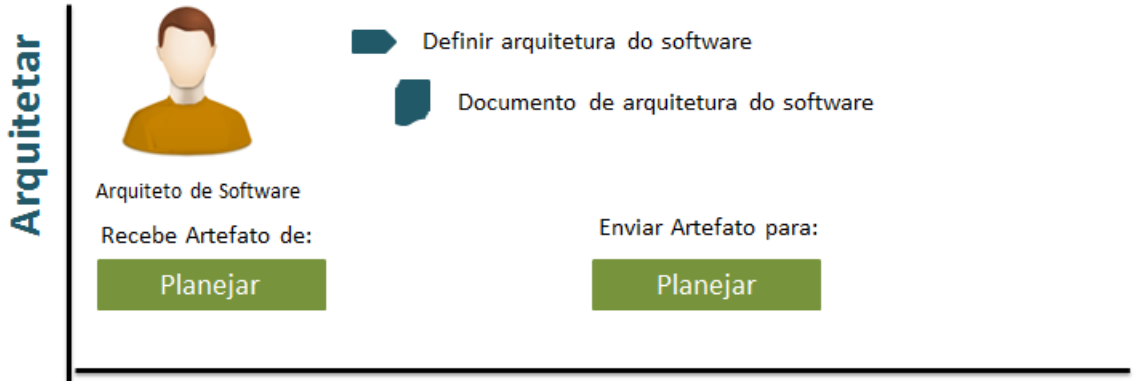


Figura 20 - Processo Arquitetar

A Figura 20 apresenta o papel que realiza atividades no processo **Arquitetar:** Arquiteto de *Software*. O Arquiteto de *Software*, ao terminar a atividade, gerará o artefato Documento de arquitetura do *software*, que é critério de entrada para o processo **Planejar**. A seguir será apresentada com mais detalhe a atividade feita nesse processo.

Definir Arquitetura do *software*

O Arquiteto de *Software* define características primárias do *software* como a arquitetura de camadas, a plataforma de desenvolvimento, os módulos do software e os padrões de projeto a serem utilizados. Como critério de saída existe o Documento de Arquitetura do *software*.

Os processos que serão explicados a seguir são feitos por funcionalidade, ou seja, por cada funcionalidade deve-se: **Planejar** (atividades: Atribuir tarefas aos membros da equipe de desenvolvimento, Planejar testes do software, Criar lista de problemas e Planejar correções), **Projetar**, **Codificar** e **Testar**. Estes processos e estas atividades fazem parte do grupo **Construção**.

Processo Projetar

O processo **Projetar** é o primeiro realizado no grupo **Construir** e é iniciado quando é finalizada a atividade Atribuir tarefas aos membros da equipe, do processo **Planejar**. É neste processo que é feita a modelagem do *software* e a descrição das entidades e do comportamento do negócio. A Figura 21 apresenta o processo **Projetar** em detalhes, com suas atividades, seus papéis e artefatos.

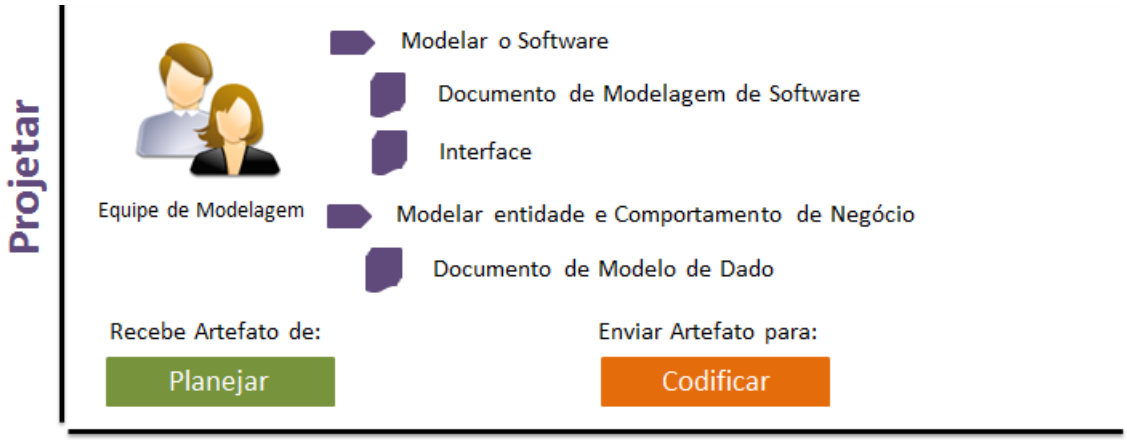


Figura 21 - Processo Projetar

A Figura 21 apresenta o papel que realiza as atividades do processo **Projetar**: a Equipe de modelagem. Este papel executa as atividades:

- Modelar o *software*; e
- Modelar entidade e comportamento de Negócio.

A Equipe de Modelagem, ao finalizar a atividade, gera os artefatos que serão critérios de entrada o processo **Codificar**. A atividade Modelar *software* gera os artefatos: Documento de Modelagem de *Software* e Interface; e a atividade Modelar entidade e Comportamento de Negócio gera o Documento de Modelo de Dados. A seguir as atividades serão apresentadas com mais detalhes.

Modelar o *Software*

Na atividade Modelar o *Software* a Equipe de Modelagem define a estrutura (modelagem) do *software*. Esta atividade compreende:

- A criação de diagramas da UML: diagrama de caso de uso e diagrama de sequência;
- A criação da interface (primeiramente de protótipos das telas ou diretamente utilizando a plataforma de *software* selecionada, como HTML e CSS);
- A criação de Casos de Uso Reais (com um nível de detalhamento adequado para o entendimento aprofundado dos Casos de Uso e do comportamento do usuário e do *software*, e da interação entre usuário e tela do *software*).

A atividade Modelar o *Software* inclui o que é equivalente ao resultado da atividade Desenvolver o Diagrama de Sequência de Projeto, do processo DPF do FDD. Ao terminar esta atividade são gerados os seguintes artefatos:

- Documento de modelagem do *software*; e
- Interface (na forma de protótipo ou diretamente utilizando a plataforma de software selecionada).

Modelar entidade e comportamento do negócio

Na atividade Modelar entidade e comportamento do negócio a Equipe de modelagem, com base nos documentos que descrevem o negócio, nos requisitos e nas funcionalidades, cria o artefato Modelo de dados do *software*, descrevendo as entidades e o comportamento do software, utilizando Casos de Uso Reais e Diagramas de Sequência de projeto. Esta atividade é baseada no FDD na atividade Escrever prefácio de código e método, do processo DPF. Ao término desta atividade é gerado o artefato Documento de modelo de dados.

Codificar

Depois de ser finalizado o processo **Planejar** começa o processo **Codificar.** É neste processo que serão implementadas as funcionalidades que foram planejadas anteriormente. A Figura 22 apresenta o processo **Codificar** em detalhes, com suas atividades, seus papéis e artefatos.

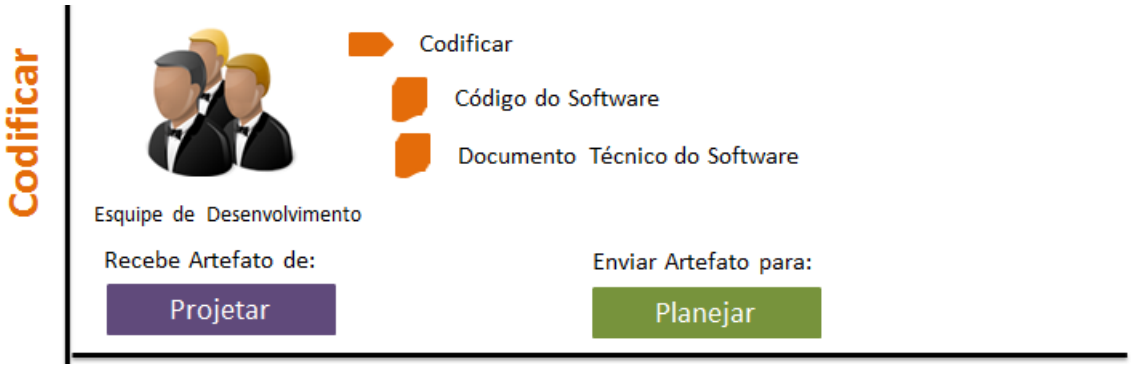


Figura 22 - Processo Codificar

A Figura 22 apresenta o papel que realiza as atividades do processo **Codificar:** Equipe de Desenvolvimento. Este papel executa a atividade Codificar.

Ao ser finalizada a atividade Codificar são gerados os artefatos: Código do Software e Documento Técnico do Software, os quais são critérios de entrada para o processo **Planejar.** A seguir serão apresentados detalhes da atividade.

Codificar

Na atividade Codificar a Equipe de Desenvolvimento, com base nos artefatos criados nas fases anteriores, cria o código do *software*. O código deve ser orientado a testes (utilizando a metodologia TDD). O código deve ser comentado (utilizando definições do arquiteto de *software*, conforme a plataforma de software escolhida). Esta atividade é baseada nas atividades Implementar classes e métodos e Teste de unidade, ambas do processo CPF do FDD. Ao término da atividade são gerados os critérios de saída: Código do *software* e Documentação técnica [do código] do *software*.

Testar

O processo **Testar** é iniciado depois que é finalizada a atividade Planejar teste de *Software*, do processo **Planejar.** É neste processo que serão feitos os testes no *software* para se verificar o que comportamentos e saídas indesejados, incorretos ou inadequados. A Figura 23 apresenta o processo **Testar** em detalhes, com suas atividades, seus papéis e artefatos.

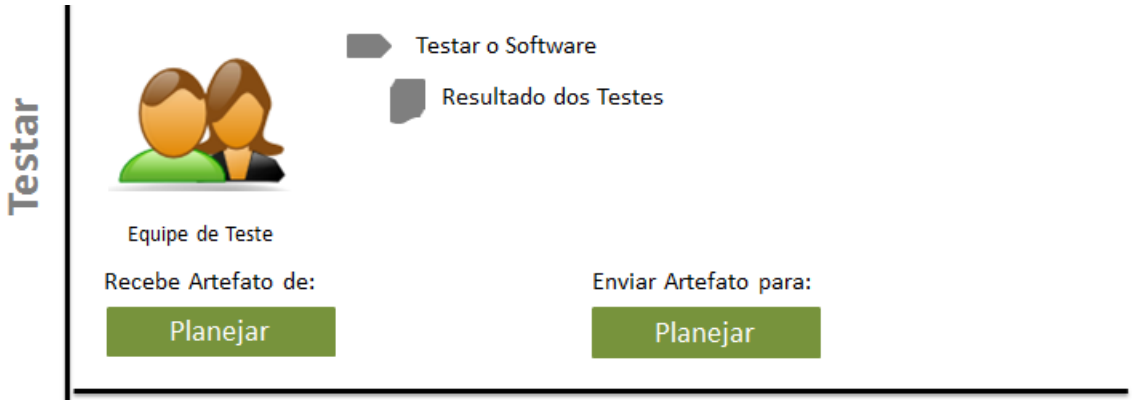


Figura 23 - Processo Testar

A Figura 23 apresenta o papel que realiza atividades do processo **Testar:** Equipe de Teste. É este papel quem executa a atividade realizada nesta fase: Testar o *Software*. Ao ser finalizada a atividade o artefato Resultado dos Testes será critério de entrada pra o processo **Planejar.** A seguir será apresentado com mais detalhe a atividade.

Testar o *Software*

Na atividade Testar *Software* a Equipe de testes, com base no Plano de testes, executa testes de uso do *software*. Ao ser finalizada esta atividade tem-se como critério de saída o artefato Resultados dos testes.

Esta seção apresentou os processos que são realizados na metodologia proposta, assim como seus papéis, artefatos e suas atividades e artefatos gerados, mostrando o ciclo que envolve os seus processos. A seguir são apresentados os Artefatos com maiores detalhes.

4.3.3. Artefatos

Nesta seção serão apresentados os artefatos que são gerados em atividades da metodologia. Os artefatos completos estão apresentados no Apêndice A.

Processo Entender

No processo Entender são realizadas as atividades: Descrever negócio, que gera os artefatos **Objeto de Negócio** e **Regras do Negócio**; a e a atividade Coletar requisito, que gera o artefato **Lista de Requisitos**. A seguir são apresentados os modelos dos artefatos gerados no processo Entender.

Objetivo de Negócio

O Analista do Negócio faz uma descrição do objetivo do negócio. A Figura 24 mostra um exemplo do artefato Objetivo de negócio.

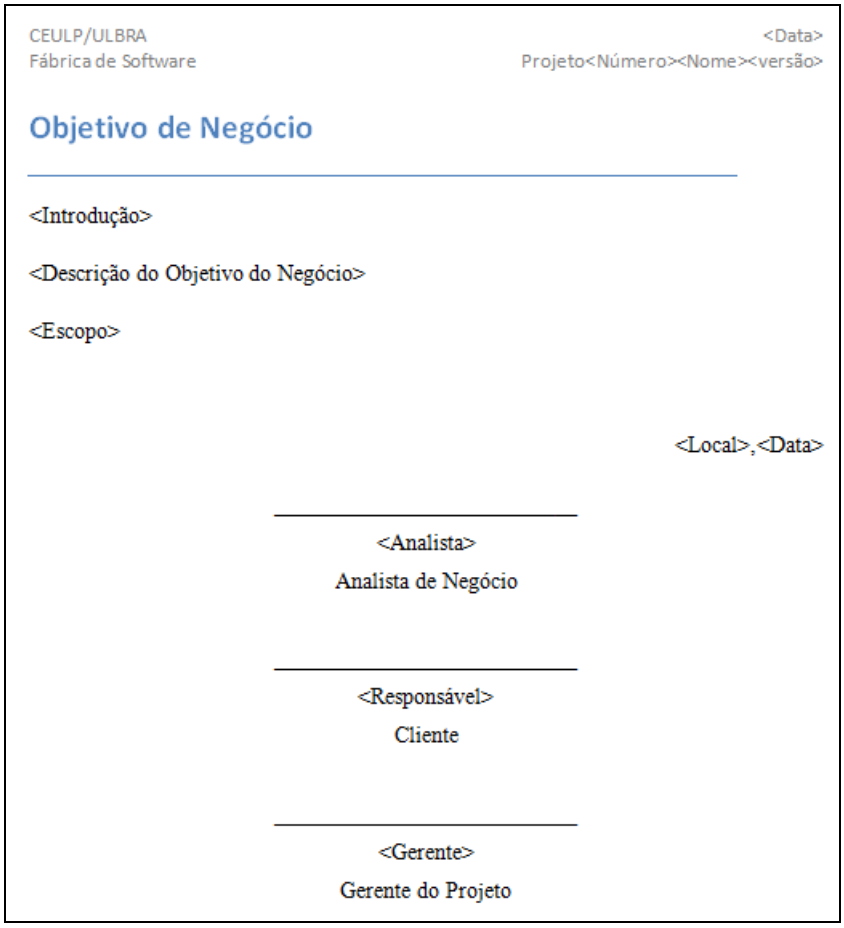


Figura 24 - Artefato Objetivo de Negócio

Conforme pode ser visto na Figura 24, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (analista, cliente, gerente) e dados do próprio artefato:

- **<Introdução>** descreve uma visão geral do artefato
- **<Descrição do Objetivo de Negócio>** faz uma descrição do objetivo do negócio
- **<Escopo>** delimita os objetivos do negócio
- **<Local>** e <**Data**> de geração do artefato
- **<Assinatura>** o(s) envolvido(s) com o artefato

O artefato completo pode ser encontrado no Apêndice 1 (Artefatos – Lista de Requisitos).

Regra de Negócio

O Analista do Negócio descreve as regras do negócio, que são "declarações sobre políticas ou condições que devem ser satisfeitas" (RUP, 1995). A Figura 25 mostra um exemplo do modelo do artefato.

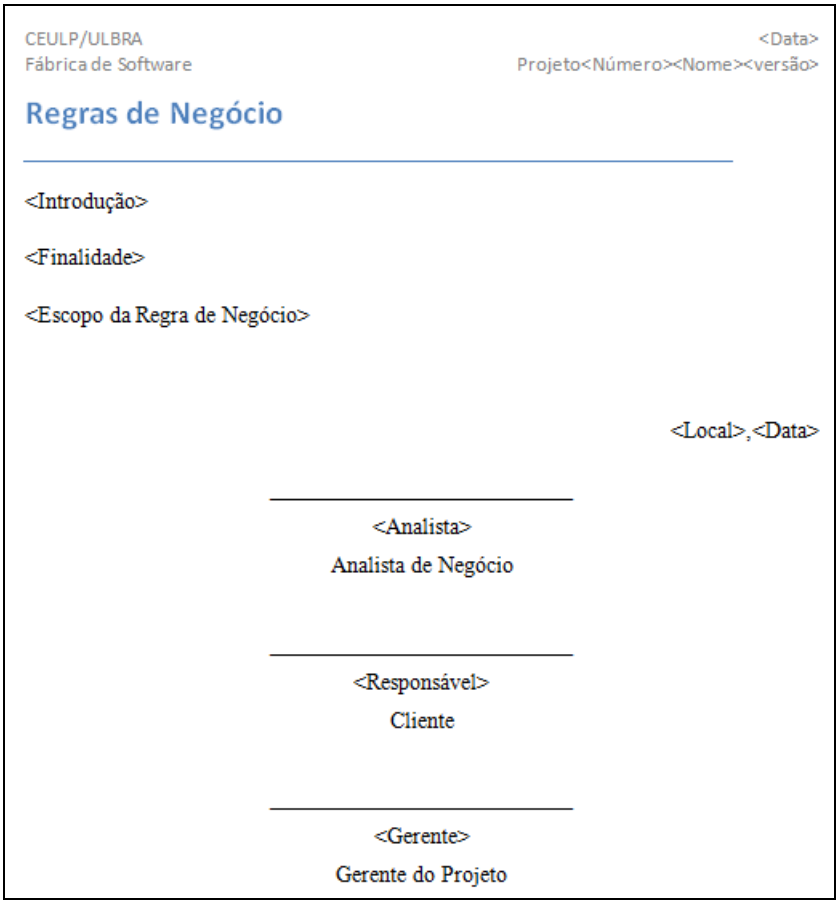


Figura 25 - Artefato de Regra de Negócio

Conforme pode ser visto na Figura 25, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (analista, cliente, gerente) e dados do próprio artefato:

- **<Introdução>** descreve uma visão geral do artefato
- **<Finalidade>** apresenta a finalidade das Regras de Negócio
- **<Escopo>** delimita as Regras do Negócio
- **<Local>** e **<Data>** de geração do artefato
- **<Assinatura>** o(s) envolvido(s) com o artefato

Lista de Requisitos

Após a entrevista com o Cliente, o Analista do Negócio gera a Lista dos Requisitos, baseando-se no que pode obter a partir das suas observações das necessidades do Cliente. A Figura 26 apresenta o artefato Lista de Requisitos.

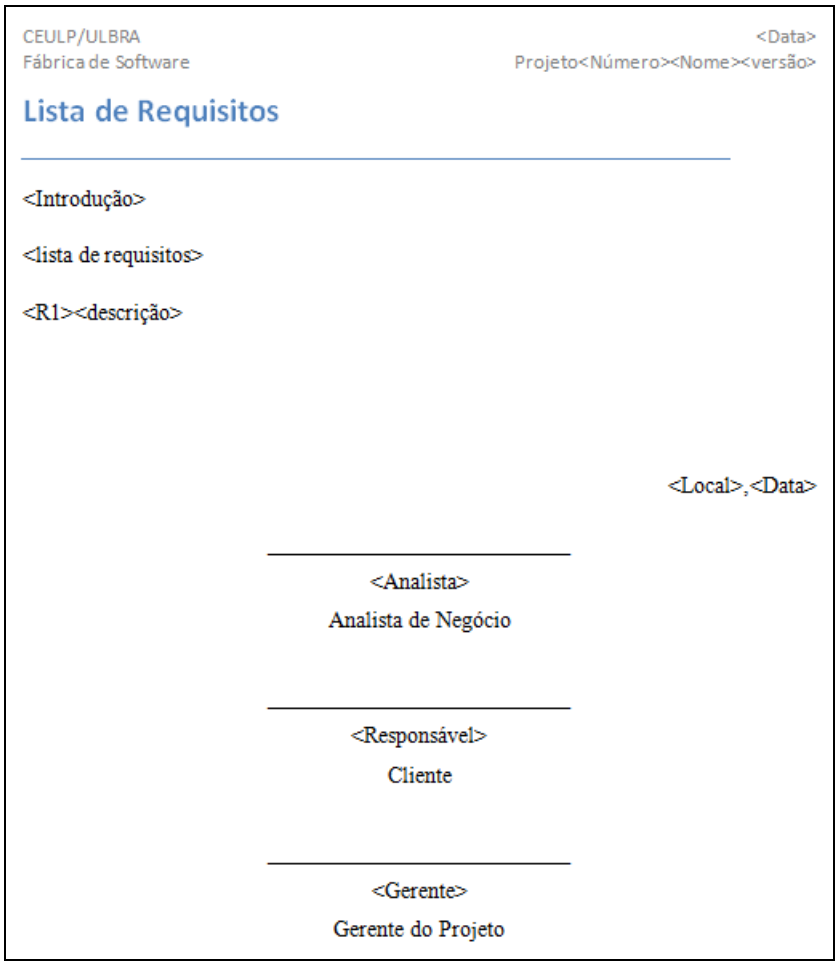


Figura 26 - Artefato Lista de Requisitos

Conforme pode ser visto na Figura 26, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (analista, cliente, gerente) e dados do próprio artefato:

- **<Introdução>** descreve uma visão geral do artefato
- **<Local>** e **<Data>** de geração do artefato

- A lista de requisitos se apresenta no formato:
  R<número sequencial>. <descrição>. O número da identificação do requisito e sua descrição.
- **<Assinatura>** o(s) envolvido(s) com o artefato

Processo Planejar

No processo **Planejar** são executadas as atividades:

- Escolher a Equipe de desenvolvimento, que gera o artefato **Lista de Membros da equipe de desenvolvimento e seus Papéis**.
- Planejar Reuniões, que gera o artefato **Plano de reuniões**.
- Definir lista de funcionalidade, que gera o artefato **Lista de funcionalidade**.
- Definir iteração, que gera o artefato **Lista de Iteração**.
- Atribuir Tarefas aos Membros da Equipe de Desenvolvimento, que gera o artefato **Plano de tarefas**.
- Planejar de Testes do Software, que gera o artefato **Plano de teste**.
- Criar lista de problemas, que gera o artefato **Lista de problemas**.
- Plano de Correções, que gera o artefato **Plano de resolução de problemas e correções**.

Lista de Membros da equipe de desenvolvimento e seus Papéis

Apresenta uma Relação dos Membros da equipe de desenvolvimento e seus respectivos papéis. A Figura 27 apresenta um modelo deste artefato.

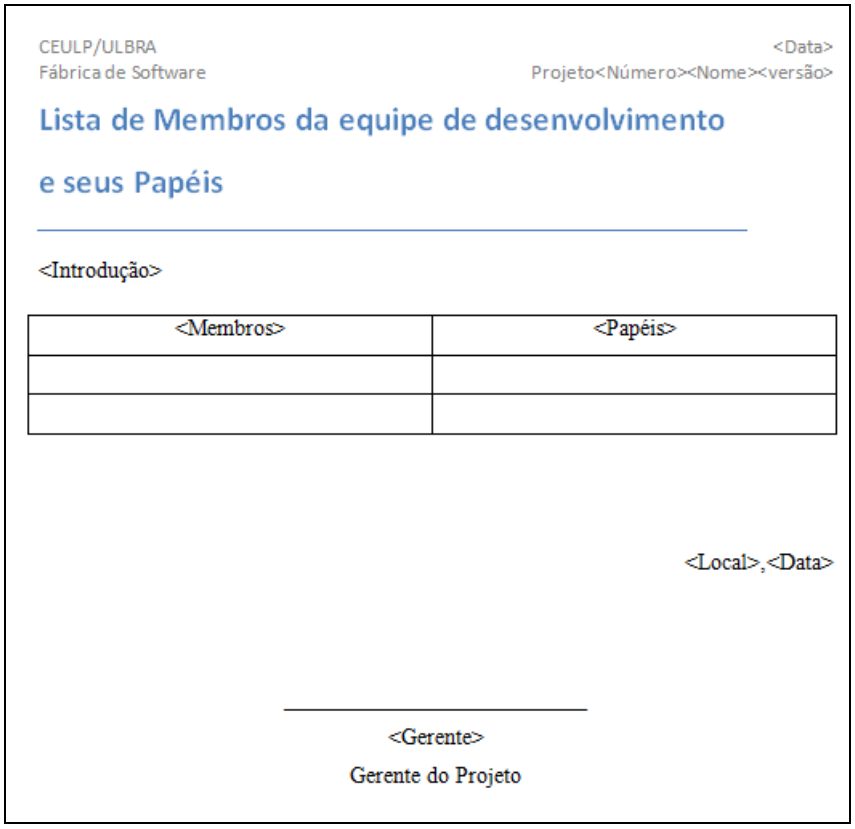


Figura 27 - Lista de Membros da equipe de desenvolvimento e seus Papéis

Conforme pode ser visto na Figura 27, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (gerente) e dados do próprio artefato:

- **<Introdução>** descreve uma visão geral do artefato
- Lista em uma tabela com: **<Membro>** e **<Papéis>**.
- **<Local>** e **<Data>** de geração do artefato
- **<Assinatura>** o(s) envolvido(s) com o artefato

Plano de Reuniões

Apresenta uma lista de todas as reuniões com datas e horários em que irão acontecer. A Figura 28 apresenta um modelo deste artefato.

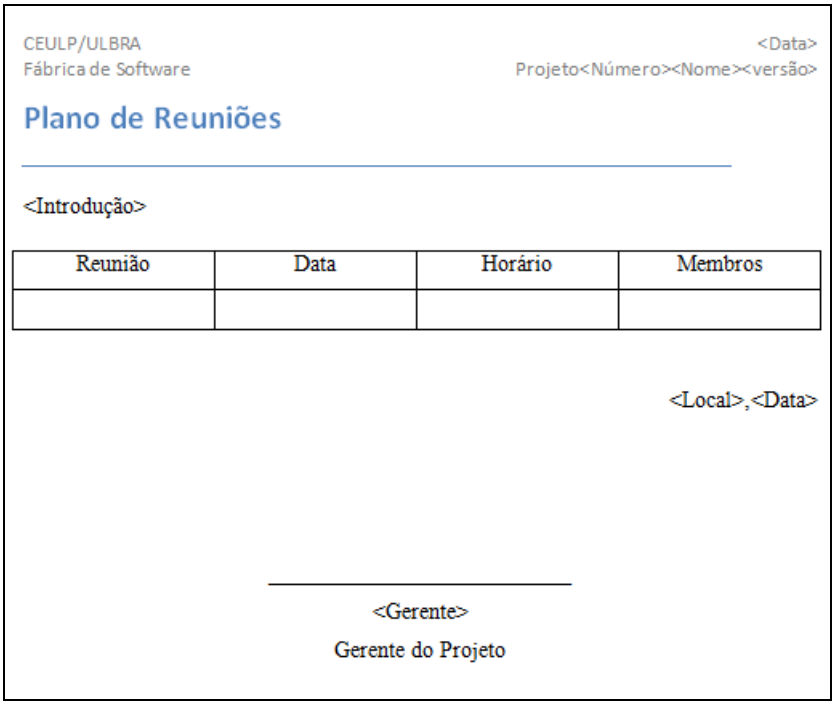


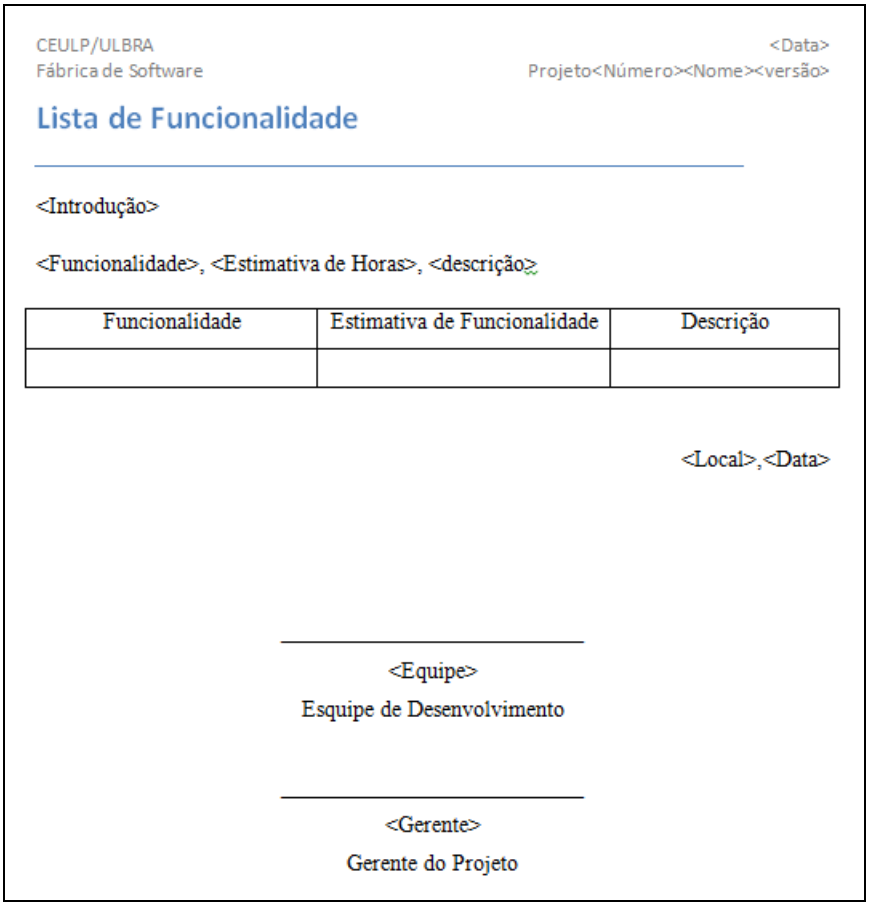
Figura 28 - Plano de Reuniões

Conforme pode ser visto na Figura 28, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (gerente) e dados do próprio artefato:

- **<Introdução>** descreve uma visão geral do artefato
- Lista em uma tabela com: <**nome da Reunião>**, <**Data**>, <**Horário e os membros envolvidos**>
- **<Local>** e **<Data>** de geração do artefato
- **<Assinatura>** o(s) envolvido(s) com o artefato

Lista de Funcionalidades

Apresenta uma lista das funcionalidades do software. A Figura 29 apresenta um modelo deste artefato.


CEULP/ULBRA <Data>
Fábrica de Software Projeto<Número><Nome><versão>

Lista de Funcionalidade

<Introdução>

<Funcionalidade>, <Estimativa de Horas>, <descrição>

Funcionalidade	Estimativa de Funcionalidade	Descrição

<Local>,<Data>

<Equipe>
Esquipe de Desenvolvimento

<Gerente>
Gerente do Projeto

Figura 29 - Lista de Funcionalidade

Conforme pode ser visto na Figura 29, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (equipe, gerente) e dados do próprio artefato:

- <**Introdução**> descreve uma visão geral do artefato
- Lista em uma tabela com: <**Funcionalidade**>, <**Estimativa da Funcionalidade (horas)**> e <**Descrição**>
- <**Local**> e <**Data**> de geração do artefato
- **<Assinatura>** o(s) envolvido(s) com o artefato

Plano de Iteração

Lista as iterações planejadas. A Figura 30 mostra um exemplo deste artefato.

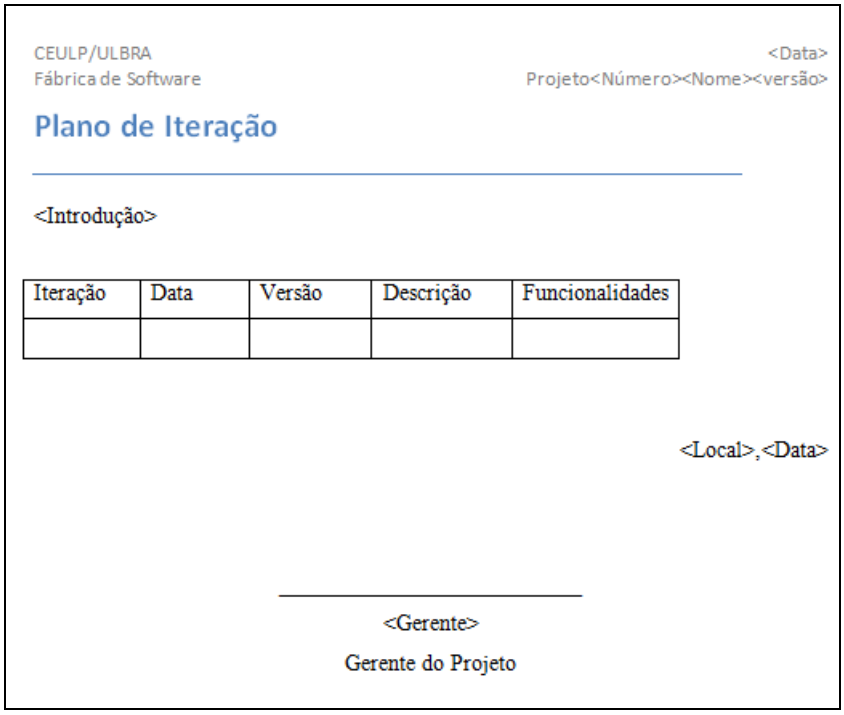


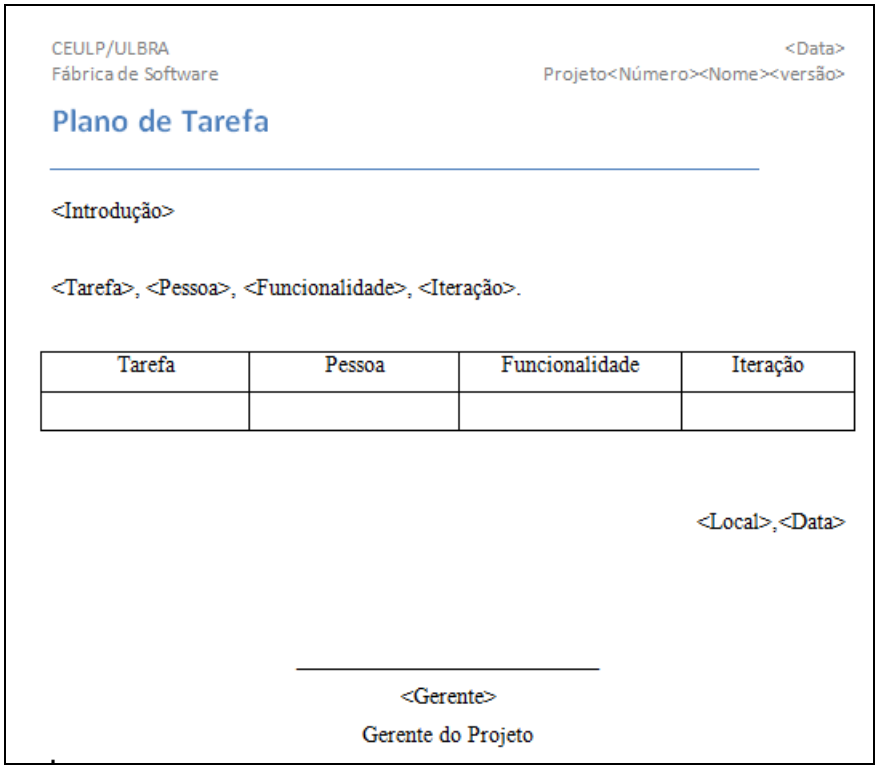
Figura 30 - Plano de Iteração

Conforme pode ser visto na Figura 30, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (gerente) e dados do próprio artefato:

- <**Introdução**> descreve uma visão geral do artefato
- Lista em uma tabela com: <**Iteração**> (identificador, número ou código), a <**Versão**> da iteração ou da funcionalidade, <**Data**> limite para conclusão da funcionalidade e <**Descrição**> e <**Funcionalidade**> nome.
- <**Local**> e <**Data**> de geração do artefato
- **<Assinatura>** o(s) envolvido(s) com o artefato

Plano de Tarefas

Apresenta uma lista de tarefas a serem executadas no decorrer do desenvolvimento. A Figura 31 mostra um exemplo deste artefato.



CEULP/ULBRA <Data>
Fábrica de Software Projeto<Número><Nome><versão>

Plano de Tarefa

<Introdução>

<Tarefa>, <Pessoa>, <Funcionalidade>, <Iteração>.

Tarefa	Pessoa	Funcionalidade	Iteração

<Local>,<Data>

<Gerente>
Gerente do Projeto

Figura 31 - Plano de Tarefa

Conforme pode ser visto na Figura 31, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (gerente) e dados do próprio artefato:

- <**Introdução**> descreve uma visão geral do artefato
- Lista em uma tabela com: <**Tarefa**>, a <**Pessoa**> que vai exercer aquela tarefa, <**Funcionalidade**> (relacionada à tarefa) e <**Iteração**>.
- <**Local**> e <**Data**> de geração do artefato
- **<Assinatura>** o(s) envolvido(s) com o artefato

Lista de Testes

Apresenta uma lista de teste a serem executada. A Figura 32 mostra um exemplo deste artefato.

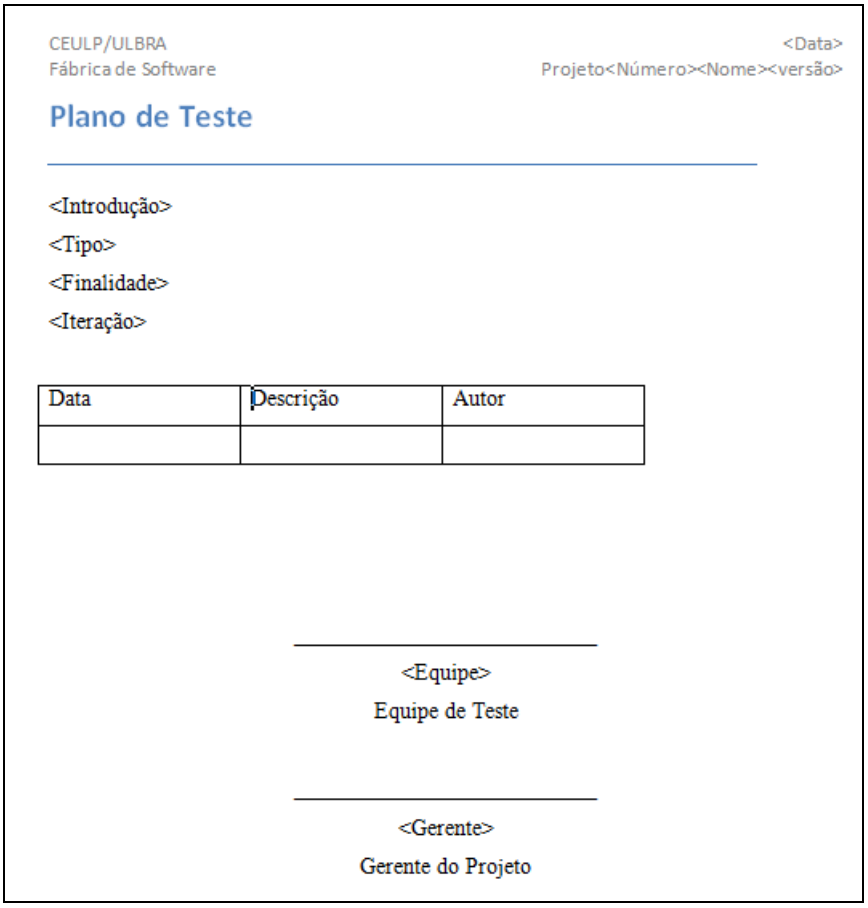


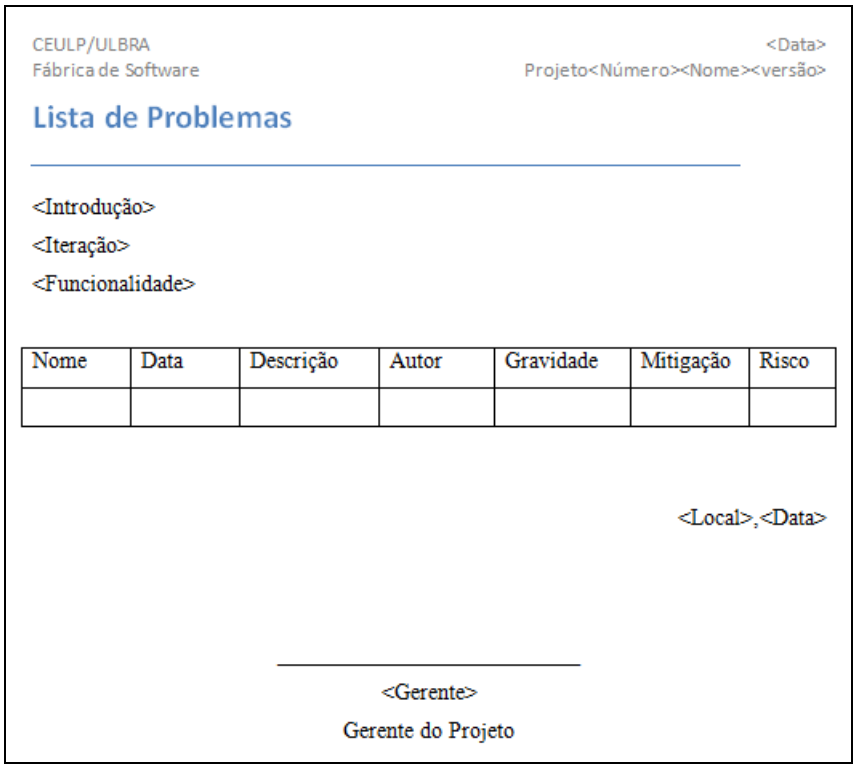
Figura 32 - Plano de Teste

Conforme pode ser visto na Figura 32, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (equipe de teste, gerente) e dados do próprio artefato:

- <**Introdução**> descreve uma visão geral do artefato
- <**Tipo de Teste**> que será aplicado
- <**Funcionalidade**> na qual o teste será aplicado
- Iteração Lista em uma tabela com dados dos testes: <**Data**>, <**Descrição**> e <**Autor**>
- <**Local**> e <**Data**> de geração do artefato
- **<Assinatura>** o(s) envolvido(s) com o artefato

Lista de Problemas

Apresenta os problemas encontrados após a execução dos testes. A Figura 33 apresenta um exemplo deste artefato.


CEULP/ULBRA <Data>
Fábrica de Software Projeto<Número><Nome><versão>

Lista de Problemas

<Introdução>
<Iteração>
<Funcionalidade>

Nome	Data	Descrição	Autor	Gravidade	Mitigação	Risco

<Local>,<Data>

<Gerente>
Gerente do Projeto

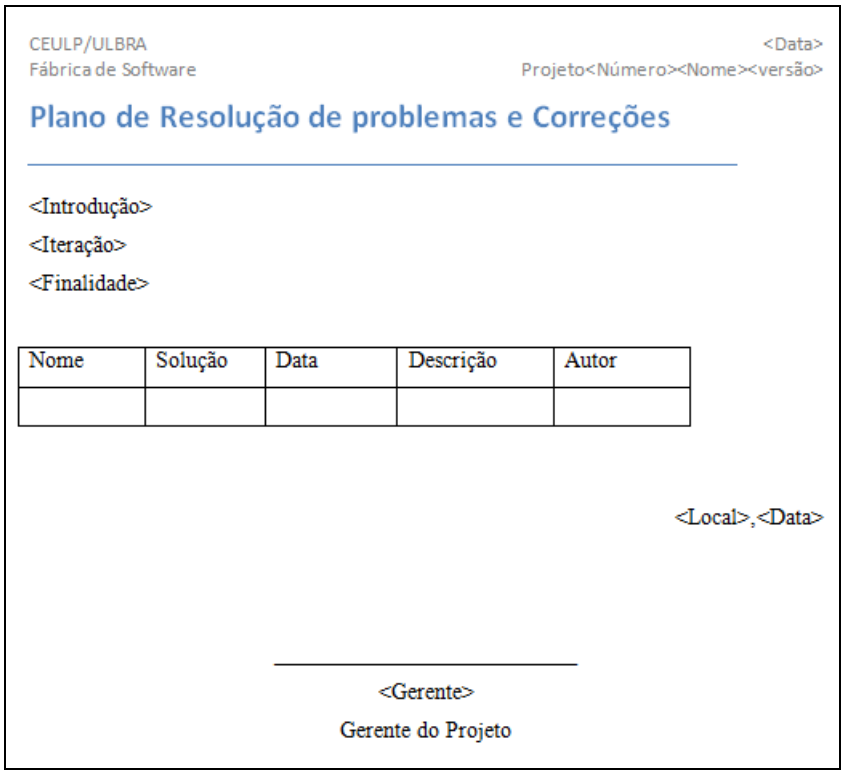
Figura 33 - Lista de Problemas

Conforme pode ser visto na Figura 33, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (gerente) e dados do próprio artefato:

- <**Introdução**> descreve uma visão geral do artefato
- <**Iteração**> que será aplicado o teste
- <**Funcionalidade**> que vai ser aplicado o teste
- Lista em uma tabela com dados do problema: <**Nome**>, <**Data**>, <**Descrição**>, e <**Autor**><**Gravidade**><**Mitigação**><**Risco**>.
- <**Local**> e <**Data**> de geração do artefato
- **<Assinatura>** o(s) envolvido(s) com o artefato

Plano de resolução de problemas e correções

Apresenta o plano de resolução dos problemas encontrados durante os testes e suas correções. A Figura 34 mostra um exemplo deste artefato.


CEULP/ULBRA <Data>
Fábrica de Software Projeto<Número><Nome><versão>

Plano de Resolução de problemas e Correções

<Introdução>

<Iteração>

<Finalidade>

Nome	Solução	Data	Descrição	Autor

<Local>,<Data>

<Gerente>

Gerente do Projeto

Figura 34 - Plano de Resolução de problemas e correções

Conforme pode ser visto na Figura 34, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (gerente) e dados do próprio artefato:

- **<Introdução>** descreve uma visão geral do artefato
- **<Iteração>** que será realizada
- **<Funcionalidade>** que será realizada
- Lista em uma tabela com informações sobre a resolução do problema: <**Nome do Problema**>, a <**Solução do problema**>, <**Data**>, uma <**Descrição do problema**>, e o <**autor**> que irá corrigir.
- **<Local>** e **<Data>** de geração do artefato
- **<Assinatura>** o(s) envolvido(s) com o artefato

Processo Arquitetar

No processo **Arquitetar** a atividade Definir a arquitetura do software gera o artefato Documento do Arquitetura de Software, apresentado a seguir.

Documento de arquitetura do software

Apresentará a arquitetura do software, mostrando os seus módulos, os sistemas que ele se relaciona. A Figura 35 apresenta um exemplo deste artefato.

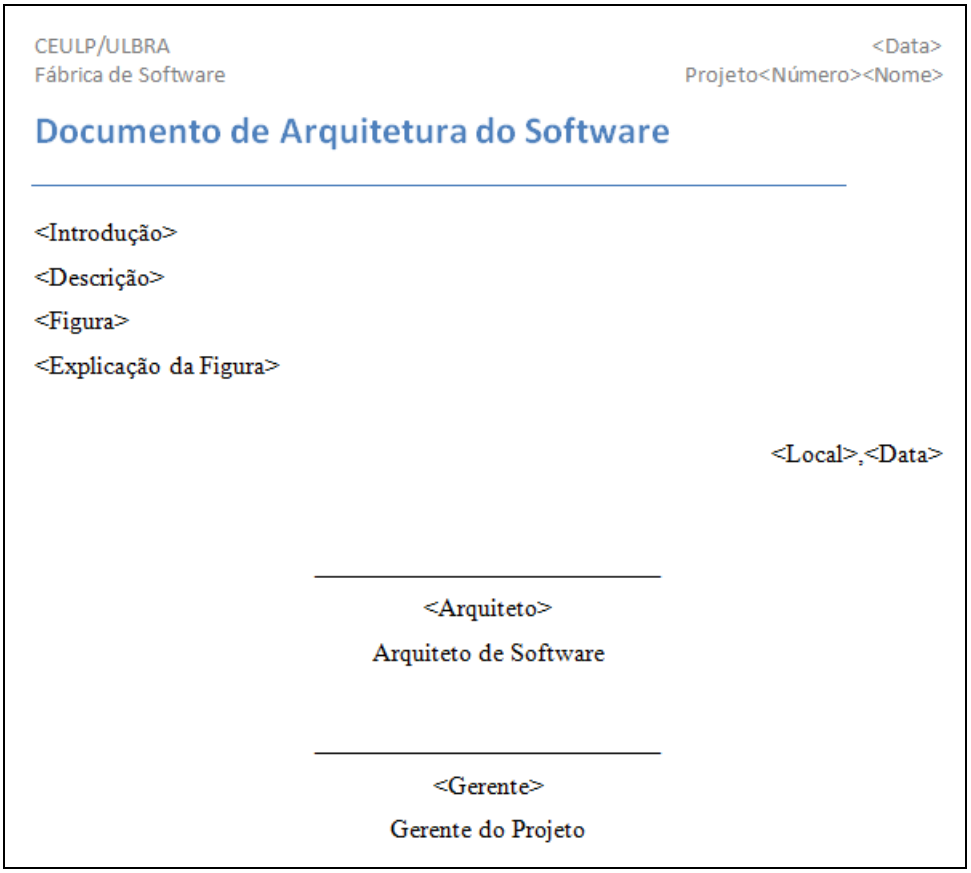


Figura 35 - Artefato de Arquitetura do software

Conforme pode ser visto na Figura 35, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (arquiteto, gerente) e dados do próprio artefato:

- <**Introdução**> descreve uma visão geral do artefato
- <**Descrição do artefato**>
- <**Figura da arquitetura**>
- <**Explicação da Figura**>
- <**Local**> e <**Data**> de geração do artefato
- **<Assinatura>** o(s) envolvido(s) com o artefato

Processo Projetar

No processo **Projetar** a atividade Modelar o software gera o artefato Documento de modelagem do Software e a atividade Modelar entidade e Comportamento de Negócio gera o artefato Documento de Modelo de Dado. A seguir será apresentados estes artefatos.

Documento de Modelagem do Software

Apresenta a modelagem do software, utilizando diagramas da UML: diagrama de caso de uso, diagrama de sequência de projeto e Casos de Uso Reais. A Figura 36 apresenta o modelo deste artefato.

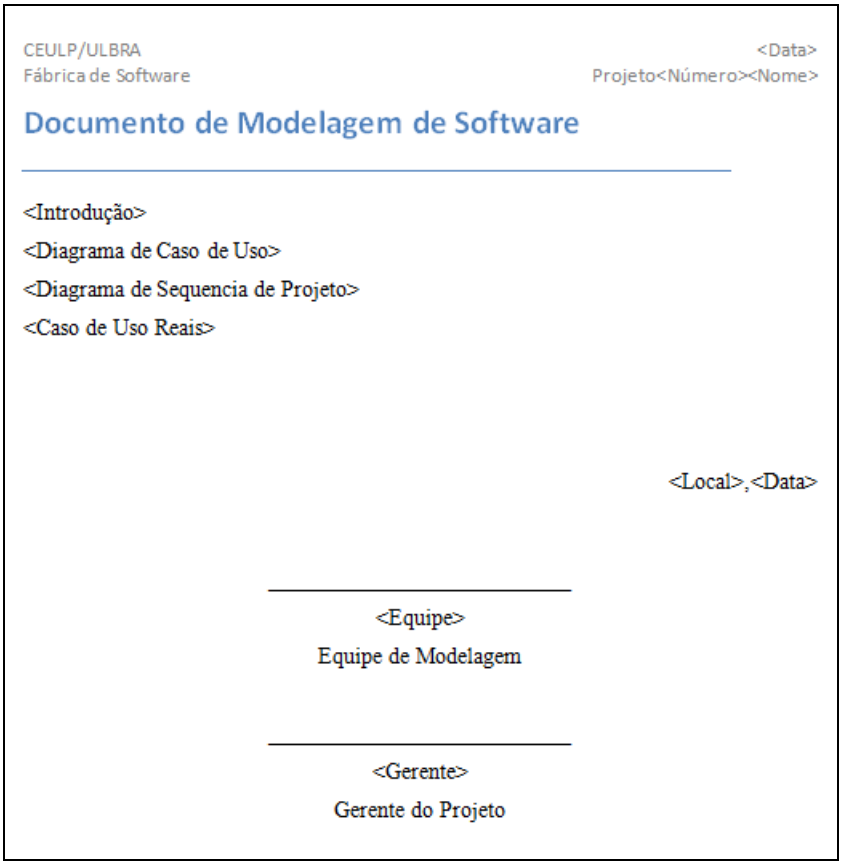


Figura 36 - Artefato Documento de Modelagem de Software

Conforme pode ser visto na Figura 36, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (equipe, gerente) e dados do próprio artefato:

- <**Introdução**> descreve uma visão geral do artefato
- <**Casos de usos feitos**>
- <**Diagramas de sequencia**> de projeto feitos
- <**Casos de uso**> feitos.
- <**Local**> e <**Data**> de geração do artefato

- **<Assinatura>** o(s) envolvido(s) com o artefato

Documento de Modelo de Dado

O documento modelo de dados apresenta: descrevendo as entidades e o comportamento do software. A Figura 37 apresenta um exemplo deste artefato.

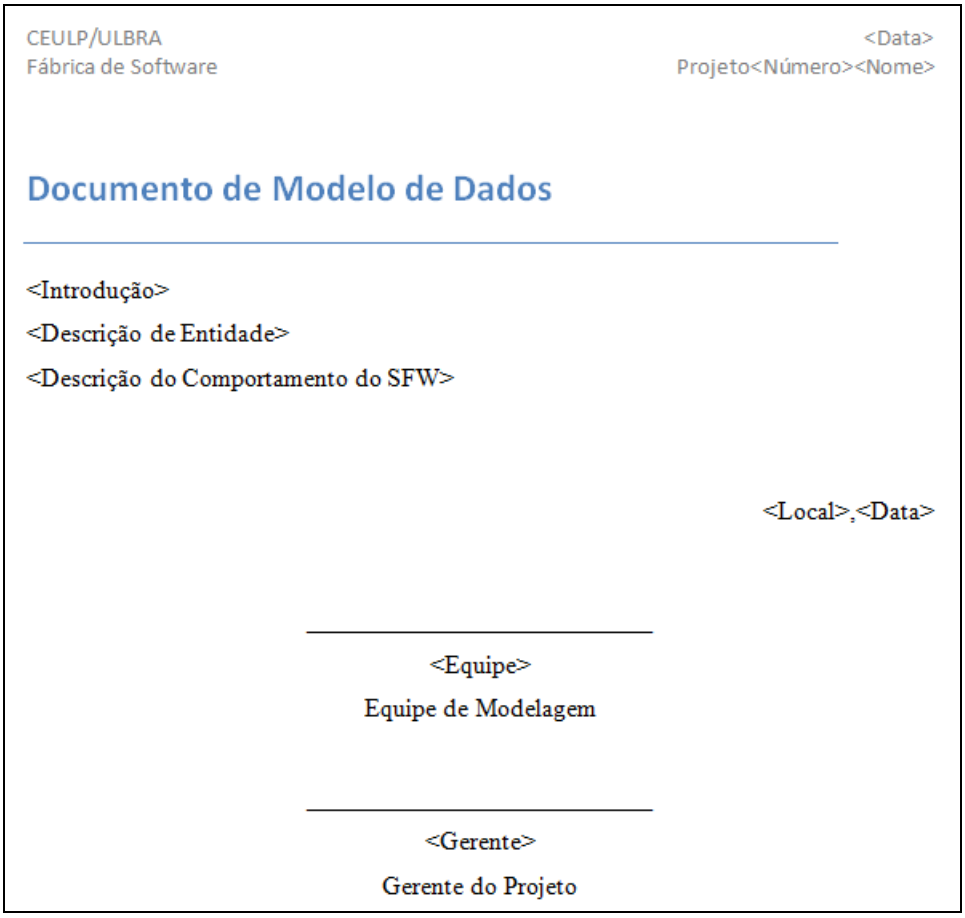


Figura 37 - Documento de modelo de Dados

Conforme pode ser visto na Figura 37, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (equipe, gerente) e dados do próprio artefato:

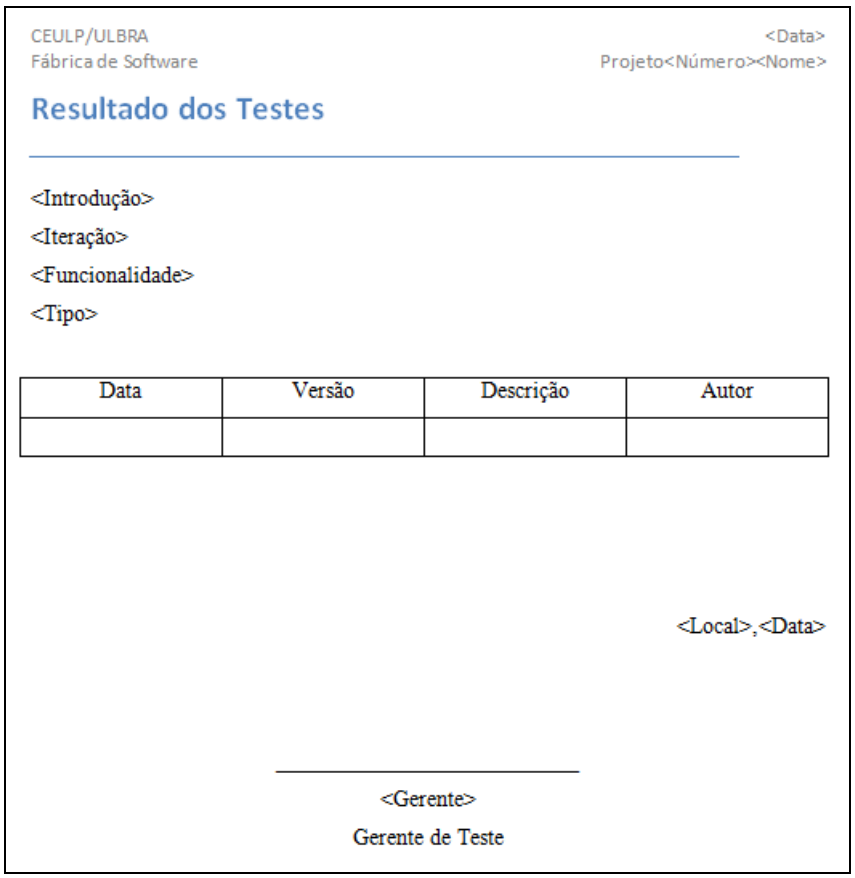
- <**Introdução**> descreve uma visão geral do artefato
- <**Descrição da entidade**>
- <**Descrição do Comportamento do SFW**>
- <**Local**> e <**Data**> de geração do artefato
- **<Assinatura>** o(s) envolvido(s) com o artefato

Testar

O processo Testar na atividade Testar o Software gera o artefato Resultado de testes, apresentado a seguir.

Resultado dos Testes

Apresenta os resultados dos testes que foram executados em processos anteriores. A Figura 38 apresenta um exemplo deste artefato.


CEULP/ULBRA <Data>
Fábrica de Software Projeto<Número><Nome>

Resultado dos Testes

<Introdução>
<Iteração>
<Funcionalidade>
<Tipo>

Data	Versão	Descrição	Autor

<Local>,<Data>

<Gerente>
Gerente de Teste

Figura 38 - Resultado de Testes

Conforme pode ser visto na Figura 38, o artefato contém dados do projeto (número, nome, versão), dados dos papéis envolvidos (equipe, gerente) e dados do próprio artefato:

- <**Introdução**> descreve uma visão geral do artefato
- **<Iteração>**
- <**Funcionalidade**>
- <**Tipo de teste**>
- Lista em uma tabela com dados do teste: <**Data**>, a versão da funcionalidade testada, uma descrição do resultado, e o autor que fez o teste.
- <**Local**> e <**Data**> de geração do artefato

- **<Assinatura>** o(s) envolvido(s) com o artefato

A seguir serão apresentados os Cerimonias que acontecem no decorrer do desenvolvimento do processo da metodologia

4.3.4. Cerimonial

Esta seção apresenta as reuniões realizadas durante o processo de desenvolvimento, além dos objetivos de cada uma e quais papéis que participam das mesmas. A Figura 39 ilustra em que momento do processo de desenvolvimento as reuniões acontecem.

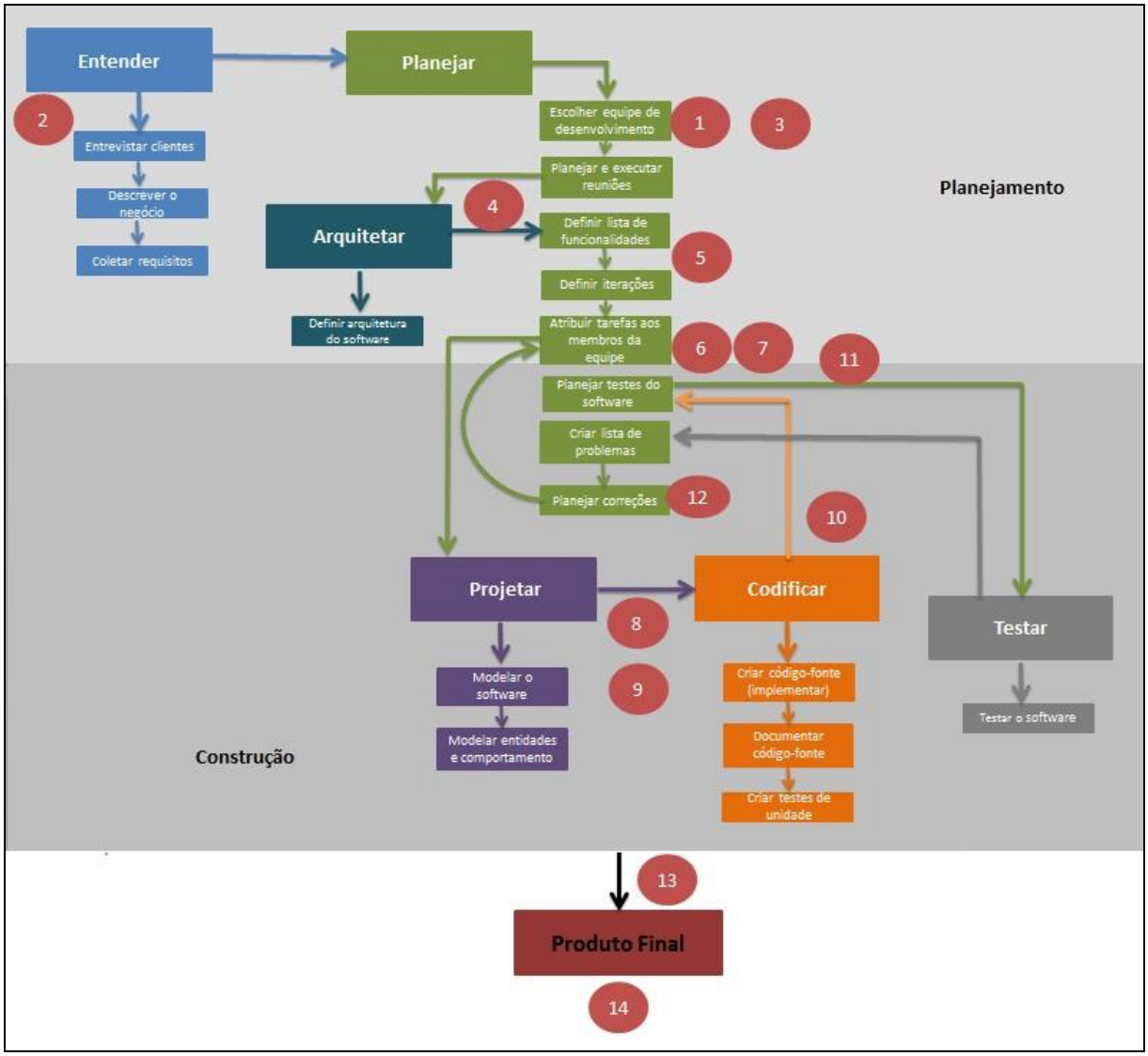


Figura 39 – Reuniões de Desenvolvimento de Software para a Fábrica de Software do CEULP/ULBRA

A Figura 39 apresenta rótulos numerados (de 1 a 14) que demonstram quando as reuniões devem acontecer. As reuniões são:

1. ***Reunião de abertura***: reunião em que o Gerente de projeto seleciona a equipe de desenvolvimento e apresenta seus papéis e suas atribuições. Acontece ao término da atividade Escolher equipe de desenvolvimento, no processo **Planejar**.
2. ***Reunião com clientes***: reunião em que o Analista de negócio coleta informações junto aos clientes sobre o negócio e descreve necessidades e requisitos do *software*. Depois desta reunião seguem-se as demais atividades do processo **Entender**.
3. ***Reunião de entendimento do negócio e homologação dos requisitos***: reunião em que o Analista de negócio apresenta o negócio e suas características para a Equipe de desenvolvimento. Nesta reunião o Gerente de projeto e a Equipe de desenvolvimento homologam o artefato Lista de Requisitos (processo **Entender**).
4. ***Reunião de apresentação da arquitetura do software***: reunião em que o Arquiteto de *software* apresenta à Equipe de desenvolvimento uma visão geral da arquitetura do *software*. Acontece ao término do processo **Arquitetar**.
5. ***Reunião de homologação das funcionalidades***: reunião em que o Gerente de projeto e a Equipe de desenvolvimento aprovam as funcionalidades do *software*, com base na Lista de Requisitos e executam o *Planning poker*, para atribuir pesos às funcionalidades (conforme complexidade ou tempo de desenvolvimento). A Reunião acontece quando a atividade Definir lista de funcionalidade do Processo Planejar estiver finalizada.
6. ***Reunião de apresentação das iterações e das tarefas***: reunião em que o Gerente de projeto apresenta à Equipe de desenvolvimento o Plano de iterações e o cronograma de realização das tarefas, o artefato Plano de tarefas. A reunião acontece logo após as atividades Definir iterações e Atribuir atividades aos membros da equipe, do processo **Planejar**.
7. ***Reunião com Clientes para apresentação das funcionalidades***: reunião em que o Analista do negócio apresenta aos Clientes os artefatos Lista de funcionalidade e o plano de iterações.
8. ***Reunião de apresentação da modelagem***: reunião em que a Equipe de modelagem apresenta à Equipe de desenvolvimento uma visão geral da modelagem do *software*. A reunião acontece depois que o processo **Projetar** tiver sido encerrado.
9. ***Reunião de apresentação do protótipo***: reunião em que o Analista de negócio apresenta aos clientes protótipos das telas do *software*. Acontecerá por funcionalidade.

10. ***Reunião de homologação da implementação***: reunião em que a Equipe de desenvolvimento apresenta uma visão geral da implementação do *software*, demonstrando o mesmo em funcionamento. A reunião acontece depois que o Processo Codificar for finalizado.
11. ***Reunião para apresentação do plano de testes***: reunião em que o Gerente de projeto apresenta à Equipe de desenvolvimento o Plano de testes. A reunião acontece depois que a atividade Planejar teste de *software* do processo **Planejar** for finalizada.
12. ***Reunião de correções de testes***: reunião em que a Equipe de desenvolvimento apresenta resultados dos testes e o Gerente de projeto planeja as correções dos testes. A reunião acontece depois que a atividade Planejar correções do processo **Planejar** for finalizada.
13. ***Reunião com o Cliente para apresentação preliminar do software***: reunião em que o Analista de negócio apresenta aos clientes as funcionalidades implementadas. A reunião fica a critério do Gerente e da Equipe, podendo ser:
    - Por Funcionalidade
    - Por iteração
    - Por data especificada
14. ***Reunião de fechamento***: reunião em que o Gerente de projetos faz um resumo do desenvolvimento e o resultado final do desenvolvimento é apresentado para a Equipe de desenvolvimento. Ao término dessa atividade tem-se o artefato Plano de Reuniões, que acontece no processo planejar.

Este capítulo apresentou os passos produzidos durante o desenvolvimento da metodologia, assim como os resultados deste desenvolvimento, que foi centrado no contexto em que a metodologia será aplicada. O contexto foi favorável para definição dos processos da metodologia, pois atualmente a FSW não possui uma metodologia formalizada. Os detalhes da metodologia proposta foram centrados entre as metodologias FDD e *Scrum*, das quais foram extraídas as principais características que pertencem ao contexto da FSW. A metodologia desenvolvida descreve os papéis, processos, artefatos e cerimoniais, para serem aplicados nos processos de desenvolvimento da FSW.

5 CONSIDERAÇÕES FINAIS

Como apresentando no decorrer do trabalho antes eram usados com maior frequência os paradigmas tradicionais, que são inflexíveis, sempre seguem na linha de desenvolvimento com a mesma sequência e são hierárquicos. Com o passar dos tempos, foram criadas outras metodologias, com base nos paradigmas tradicionais. Surgiram, então, as Metodologias Ágeis, que são adaptáveis a mudanças e menos burocráticas.

Existem diversas metodologias ágeis. O *Scrum*, embora seja uma metodologia ágil completa, possui práticas mais voltadas para gerenciamento e comunicação da equipe. O FDD enfoca o desenvolvimento por funcionalidades: primeiro projeta como fará o seu desenvolvimento, depois implementa cada uma. Devido à realidade apresentada pela Fábrica de Software do CEULP/ULBRA, as duas metodologias foram estudadas para que fosse possível basear-se em seus processos e criar uma metodologia específica para a Fábrica de Software.

O presente trabalho teve por objetivo definir e formalizar uma metodologia a ser seguida no desenvolvimento dos *softwares* da FSW do CEULP/ULBRA. Embora haja algumas práticas neste ambiente de desenvolvimento, não há uma metodologia formalizada e padrões não são seguidos de forma adequada. Como as metodologias existentes não se encaixam por completo à realidade da FSW, a coordenação da FSW decidiu pela criação de uma metodologia própria, a qual é objeto e produto do presente trabalho.

É de fundamental importância a FSW mudar a cultura de desenvolvimento, e seguir uma metodologia formalizada, para ajudar com as dificuldades que possam surgir, como mudança de funcionários, uma documentação com mais detalhes, maior acesso ao que está sendo desenvolvido com todos os *softwares*, seguir a mesma maneira de desenvolvimento etc.

Na FSW a rotatividade de membros (funcionários e estagiários) é frequente, devido ao fato de ser um meio acadêmico, existindo membros que são alunos, que ficam por um tempo e depois saem da FSW. Com a documentação mais detalhada, fica mais fácil o entendimento do desenvolvimento.

A metodologia proposta proporciona um desenvolvimento de *software* que segue processos, e permite que um membro da equipe possa iniciar sua participação em um projeto até mesmo com o desenvolvimento já tendo sido iniciado.

A metodologia para a FSW vai atingir as necessidades esperadas incialmente, no sentido de fornecer documentação de desenvolvimento com mais detalhes, tanto para se ter registrado quanto para que isso auxilie no entendimento do processo.

A metodologia aplica o gerenciamento do *Scrum*, utiliza as reuniões, adaptando-as, e do FDD, aplica desenvolvimento por funcionalidade, alguns dos papéis, bem como alguns dos seus processos. Ao juntar as duas metodologias criou-se esta metodologia da FSW.

A criação da metodologia foi baseada na realidade e em características da FSW. Entretanto, ela pode ser aplicada em outros contextos com estas características:

- **Alta rotatividade** – É um problema que pode ser solucionado seguindo o processo da metodologia, pois a partir do momento que outro membro entra na equipe no meio de um projeto, ele poderá seguir o que a metodologia indica. Além disso, com a documentação definida abrangendo detalhes do processo, o membro que se integrar à equipe e estudar a documentação vai entender do que se trata o desenvolvimento.
- **Poucos integrantes** - Um só membro pode desenvolver mais de um papel. Além de poder está desempenhando funções em mais de um projeto.
- **Exigência de prazos** - A metodologia faz com que se sigam padrões, que são flexíveis, apresenta uma curva de aprendizagem curta que economiza tempo.
- **Projetos paralelos** – A metodologia permite que um mesmo integrante possa participar de mais de um projeto com papéis diferentes.

5.1. Trabalhos futuros

O presente trabalho tem como trabalhos futuros:

- O acompanhamento completo da metodologia na Fábrica de Software;
- Validação da metodologia;
- Aplicar as reuniões diárias e não diárias, pra verificar qual se adapta melhor;

- Nos processos da metodologia adicionar o processo **Guiar**, que representa a criação de uma base de conhecimento, fornecendo um histórico e guias de desenvolvimento. Este processo poderia ter a seguinte atividade:
    - Coletar documentos-guia de desenvolvimento
        - Papéis: Gerente de projeto, Equipe de modelagem, Equipe de desenvolvimento
        - Descrição: são coletados os guias para execução das atividades e criação dos artefatos
        - Artefatos:
            - Guia de Entendimento do negócio
            - Guia de Planejamento
            - Guia de Arquitetura do software
            - Guia de Modelagem
            - Guia de Codificação
            - Guia de Testes
- Estudar as melhores técnicas e práticas para coleta de requisitos e aplicá-las no processo **Entender**.
- Outras metodologias podem ser estudadas e suas boas práticas podem ser adicionadas à metodologia deste trabalho, como TDD (*Test Driven Development*), MSF Ágil (*Microsoft Solutions Framework*) e XP (*eXtreme Programming*).
- A discriminação (os diferentes tipos) de líderes - fazer uma distribuição de líderes dentro das equipes nos processos.

6 REFERÊNCIAS BIBLIOGRÁFICAS

BÄUERLE, Renata Burmeister**. PX:** Um Processo de Desenvolvimento de Software Adaptado para Projetos de Pequeno Porte. 2009. 16 p. Trabalho de Conclusão de Curso (Bacharel em Sistemas de Informação) - Centro Universitário Ritter dos Reis, Porto Alegre, Rio Grande do Sul. Disponível em: http://www.uniritter.edu.br/graduacao/informatica/sistemas/downloads/tcc2k9/TCCII_2009_1_Renata.pdf. Acesso em: 08 de outubro de 2011.

BECK, K. et al. **Manifesto para Desenvolvimento Ágil de Software. 2001. Disponível em:** <http://agilemanifesto.org/iso/ptbr/>**. Acesso em 23/03/2012**

BISSI, Wilson. ***Scrum* – Metodologia de Desenvolvimento Ágil.** Revista Campo

BONOW, Roberto. **Sugestão e modelagem de práticas do desenvolvimento ágil para aplicação em desenvolvimento distribuído *onshore insourcing*.** 2008. 92 p. Monografia (Bacharel em Ciência da Computação) – Universidade Federal de Pelotas, Pelotas, Rio Grande do Sul.

Digital. Volume 2, número 1, 2007. Disponível em: http://revista.grupointegrado.br/revista/index.php/campodigital/article/view/312/146. Acesso em 20 de setembro de 2008.

CAIXETA, Marcelo**. Investigação para uma Proposta de um Método Ágil para Identificação de Riscos em Projetos de TI.** 2011. 222 p. Dissertação (Mestrado Profissional em Desenvolvimento Regional) – Faculdade Alves Faria, Goiânia.

COHN, Mike. **Agile Estimating and Planning.** Prentice Hall: New Jersey, 2007. 368 p.

CORDEIRO, Edson dos Santos. **Introdução a Ciclo de Vida do Software.** Disponível em: http://www.cordeiro.pro.br/aulas/engenharia/processoDeSoftware/ciclos.pdf. Acesso em: 26 de fevereiro de 201

CRISP. ***Planning Poker***. 2008. Disponível em: http://www.crisp.se/planningpoker. Acesso em 28 de setembro de 2008.

FAGUNDES, Priscila Basto. **Framework para Comparação e Análise de Métodos Ágeis.** 2005. 134 p. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina.

FERNANDES, A. A.; TEIXEIRA D. S. **Fábrica de Software**: Implantação e Gestão de Operações. São Paulo,Editora Atlas, 2004 - acha esse trabalho pra mim.

FIGUEIREDO, Alexandre Magno. *Scrum* e as Armadilhas das Reuniões Diárias. Visão Ágil – **Revista Visão Ágil**. Edição 001, 2007. Disponível em: http://www.visaoagil.com/downloads/edicoes/VA_01.pdf. Acesso em 25 de dezembro de 2008.

GOMES, Pedro Miguel Ribeiro Veloso. **Integração de modelos de desenvolvimento de software mais e menos ágeis.** 2009. 54 p. Dissertação (Mestrado em Engenharia Informática e Computação) – Faculdade de Engenharia da Universidade do Porto, Porto, Portugal.

GONÇALVES, Eduardo dos Santos; REIS FILHO, Heitor Boeira dos. **Ferramenta para Gerenciamento de Requisitos em Metodologias Ágeis**. **Hífen,** Uruguaiana, v. 32, n. 32, 148-155, 2º Semestre de 2008.

HIGHSMITH, Jim. **Agile software development ecosystems.** Boston: Pearson Education, 2002. 404 p.

LEAL, Flávio; SETTI, Rodrigo; GAMEIRO, Lucas; BOSCARIOL, Leandro. **Feature Driven Development**. Disponível em: http://www2.dc.uel.br/~rlarruda/trab/fdd.pdf. Acesso em: 14 de outubro de 2011.

LEITÃO, Michele de Vasconcelos. **Aplicação de Scrum em Ambiente de Desenvolvimento de Software Educativo.** 2010. 72 p. Monografia (Bacharel em Engenharia da Computação) – Escola Politécnica de Pernambuco, Recife.

LIMA, Ricardo Roberto de. **Metodologia de Desenvolvimento de Sistemas de Informação baseados em OO.** Disponível em: http://www.i2p.com.br/ricardo/artigo-unibratec-rrl-metodologia-desenvolvimento-web.pdf. Acesso em: 24 de fevereiro de 2012.

LORDELLO, Elizabeth Torres. **MPS.BR Modelo de Qualidade de Software e a sua Aplicabilidade à Realidade Brasileira.** 2010. 82 p. Monografia (Pós-Graduação em Engenharia de Sistemas) – Escola Superior Aberta do Brasil – ESAB, Rio de Janeiro.

NETO, Erasmo Isotton. ***Scrumming* - Ferramenta Educacional para Ensino de Práticas do *SCRUM*.** Faculdade de Informática – Pontifica Universidade Católica do Rio Grande do Sul (PUCRS), 2008. Disponível em: http://www.inf.pucrs.br/~rafael/Scrumming/Scrumming.pdf. Acesso em 18 de outubro de 2011.

NOMURA, Luzia. **Definição e estabelecimento de processos de fábrica de software em uma organização de ti do setor público**. 2008. 237 p. Tese (Mestrado em Engenharia) – Universidade de São Paulo, São Paulo.

NOMURA, Luzia; SPINOLA, Mauro de Mesquita; HIKAGE, Osvaldo; TONINI, Antonio Carlos. FS-MDP: **Um Modelo de Definição de Processos de Fábrica de Software**. In: ENEGEP, 26, 2006, Fortaleza – Ceará. ***Anais...*** Fortaleza: ABEPRO

MOUNTAIN GOAT SOFTWARE. Disponível em: http://www.mountaingoatsoftware.com/scrum/figures. Acesso em: 28 de maio de 2012

PACHECO, Guilherme Furtado. **Desenvolvimento de um Software Touch Screen para Auxiliar na Gerência do Taskboard**. 2010. 96 p. Monografia (Bacharel em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina.

PALMER S. R., FELSING J. M. **A Practical Guide to Feature-Driven Development.** Prentice-Hall: USA, 2002. 304 p.

PINHEIRO, Lívia Figueiredo Rodrigues. **Modelagem do domínio DAA utilizando a tecnologia P3TECH©.** 2007. 196 p. Monografia (Graduação em Ciência da Computação) – Faculdade Farias Brito, Fortaleza, Ceará

PINTO, Miguel Alexandre Pimpim. **Gestão de Projectos com Processos Ágeis**. 2010. 83 p. Dissertação (Mestrado em Engenharia Informática e de Computadores) – Universidade Técnica de Lisboa, Lisboa, Portugal.

PRESSMAN, Roger S. **Engenharia de Software**. 3. ed. São Paulo: Makron Books, 1995. 720 p.

PURIFICAÇÃO, Mauricio Cesar Santos da. **FDWS:** Uma Metodologia para Gerência e Desenvolvimento de Projetos Ágeis de Business Intelligence. 2010. 156 p. Monografia (Bacharel em Ciência da Computação) – Universidade Federal da Bahia, Salvador, Bahia.

RETAMAL, Adail Muniz**. FDD – Feature-Driven Development** : Descrição de Processos. Disponível em: http://www.heptagon.com.br/files/FDD-Processos.pdf. Acesso em: 14 de outubro de 2011.

ROCHA, Thayssa Águila da; OLIVEIRA, Sandro Ronaldo Bezerra; VASCONCELOS, Alexandre Marcos Lins de. **Adequação de Processos para Fábricas de Software**. In: Simpósio Internacional de Melhoria de Processos de Software, 6, 2004, São Paulo. Anais. São Paulo: SIMPROS, 2004. p. 131-142

RUP: Rational Unified Process. Site Oficial em Português. Disponível em: http://www.wthreex.com/rup/portugues/index.htm. Acessado em: 24 de junho de 2012.

TEIXEIRA, Maria. **O Processo De Desenvolvimento De Software.** Disciplina de Engenharia de Software. Universidade Federal do Espirito Santo.2009. Disponível em: http://www.ceunes.ufes.br/downloads/2/mariateixeira-EC.Engenharia%20de%20Software.Conte%C3%BAdo%203.2009.1.pdf. Acesso em: 14 de outubro de 2011.

TREVISAN, Rosana (Coord.). **Michaelis Moderno Dicionário da Língua Portuguesa**. São Paulo: Editora Melhoramentos. Disponível em: http://michaelis.uol.com.br. Acesso em: 19 de março de 2012.

SAVOINE, M.; et al. **Análise de Gerenciamento de Projeto de Software Utilizando Metodologia Ágil XP e *Scrum*:** Um Estudo de Caso Prático. In: Encontro de Estudantes de Informática do Tocantins, 11, 2009, Palmas. **Anais**. Palmas: Centro Universitário Luterano de Palmas, 2009. p. 93-102. Disponível em: <http://tinyurl.com/yfsvnwz>. Acesso em 17/05/2010.

SETTI, Rodrigo; GAMEIRO, Lucas; BOSCARIOL, Leandro; **LEAL, Flávio. Feature Driven Development. Disponível em: http://www2.dc.uel.br/~rlarruda/trab/fdd.pdf.** Acesso em: 14 de outubro de 2011.

SCHWABER, K.; BEEDLE, M. **Agile Software Development With *Scrum***. Prentice-Hall, Nova Jersey, 2002.

SESHADRI, N.; WINTERS, H. Jack. **Two signaling schemes for improving the error performance of frequency-division-duplex (FDD) transmission systems using transmitter antenna diversity.** Disponívell em:http://www.jackwinters.com/00507522.pdf. Acesso em: 16 de março de 2012.

SETTI, Rodrigo; GAMEIRO, Lucas; BOSCARIOL, Leandro; LEAL, Flávio. **Feature Driven Development**. Disponível em: http://www2.dc.uel.br/~rlarruda/trab/fdd.pdf.Acesso em: 14 de outubro de 2011.

SOMMERVILLE, I. **Engenharia de Software.** 8. ed. São Paulo: Pearson Addison-Wesley, 2007. 568 p.

SUTHERLAND, Jeff; SCHWABER, Ken. **The *Scrum* Papers:** Nuts, Bolts, and Origins of an Agile Process. Disponível em: http://www.crisp.se/scrum/books/ScrumPapers20070424.pdf. Acesso em: 24 de outubro de 2011.

SCHWABER, K.; BEEDLE, M. **Agile Software Development With *Scrum***. Prentice-Hall, Nova Jersey, 2002.

TAVARES, Ana Lúcia de Oliveira; ECKEL, Ana Paula; SCARPA, Cateane; VENDRAME, Závia Roselita. **Engenharia de Software:** uma visão geral. Disponível em:http://www.joinville.udesc.br/sbs/professores/caliari/materiais/Artigo1.pdf. Acesso em: 13 de março de 2012.

VASCONCELOS, Alexandre Marcos Lins de; ROUILLER, Ana Cristina; MACHADO, Cristina Ângela Filipak; MEDEIROS, Teresa Maria Maciel de. **Introdução à engenharia de software e à qualidade de software.** 2006. 157 p. Monografia (Especialização em Melhoria de Processo de Software) – Universidade Federal de Lavras, Lavras, Minas Gerais.

APÊNDICES

Apêndice A – Artefatos da Metodologia de Software da Fábrica de Software do CEULP/ULBRA

Objetivo de Negócio

<Introdução>

<Descrição do Objetivo do Negócio>

<Escopo>

<Local>,<Data>

_____

<Analista>

Analista de Negócio

_____

<Responsável>

Cliente

_____

<Gerente>

Gerente do Projeto

Regras de Negócio

<Introdução>

<Finalidade>

<Escopo da Regra de Negócio>

<Local>,<Data>

_____

<Analista>

Analista de Negócio

_____

<Responsável>

Cliente

_____

<Gerente>

Gerente do Projeto

Lista de Requisitos

<Introdução>

<lista de requisitos>

<R1><descrição>

<Local>,<Data>

_____

<Analista>

Analista de Negócio

_____

<Responsável>

Cliente

_____

<Gerente>

Gerente do Projeto

Lista de Membros da equipe de desenvolvimento e seus Papéis

<Introdução>

<Membros>	<Papéis>

<Local>,<Data>

_____

<Gerente>

Gerente do Projeto

Plano de Reuniões

<Introdução>

Reunião	Data	Horário	Membros

<Local>,<Data>

_____

<Gerente>

Gerente do Projeto

Lista de Funcionalidade

<Introdução>

<Funcionalidade>, <Estimativa de Horas>, <descrição>

Funcionalidade	Estimativa de Funcionalidade	Descrição

<Local>,<Data>

_____

<Equipe>

Esquipe de Desenvolvimento

_____

<Gerente>

Gerente do Projeto

Plano de Iteração

<Introdução>

Iteração	Data	Versão	Descrição	Funcionalidades

<Local>,<Data>

_____

<Gerente>

Gerente do Projeto

Plano de Tarefa

<Introdução>

<Tarefa>, <Pessoa>, <Funcionalidade>, <Iteração>.

Tarefa	Pessoa	Funcionalidade	Iteração

<Local>,<Data>

_____

<Gerente>

Gerente do Projeto

Plano de Teste

<Introdução>

<Tipo>

<Finalidade>

<Iteração>

Data	Descrição	Autor

_____

<Equipe>

Equipe de Teste

_____

<Gerente>

Gerente do Projeto

Lista de Problemas

<Introdução>

<Iteração>

<Funcionalidade>

Nome	Data	Descrição	Autor	Gravidade	Mitigação	Risco

<Local>,<Data>

_____

<Gerente>

Gerente do Projeto

Plano de Resolução de problemas e Correções

<Introdução>

<Iteração>

<Finalidade>

Nome	Solução	Data	Descrição	Autor

<Local>,<Data>

_____

<Gerente>

Gerente do Projeto

Documento de Arquitetura do Software

<Introdução>

<Descrição>

<Figura>

<Explicação da Figura>

<Local>,<Data>

_____

<Arquiteto>

Arquiteto de Software

_____

<Gerente>

Gerente do Projeto

Uml

- diagrama de caso de uso, diagrama de sequência de projeto;

Casos de Uso Reais

Documento de Modelagem de Software

<Introdução>

<Diagrama de Caso de Uso>

<Diagrama de Sequencia de Projeto>

<Caso de Uso Reais>

<Local>,<Data>

_____

<Equipe>

Equipe de Modelagem

_____

<Gerente>

Gerente do Projeto

Documento de Modelo de Dados

<Introdução>

<Descrição de Entidade>

<Descrição do Comportamento do SFW>

<Local>,<Data>

_____

<Equipe>

Equipe de Modelagem

_____

<Gerente>

Gerente do Projeto

Resultado dos Testes

<Introdução>

<Iteração>

<Funcionalidade>

<Tipo>

Data	Versão	Descrição	Autor

<Local>,<Data>

_____

<Gerente>

Gerente de Teste