

UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE
LABORATÓRIO DE ENGENHARIA E EXPLORAÇÃO DE PETRÓLEO
SETOR DE MODELAGEM MATEMÁTICA COMPUTACIONAL
Disciplina Introdução ao Projeto de Engenharia

Metodologia de Desenvolvimento de Projeto de Engenharia
Versão 2023.2

Prof. André Duarte Bueno

Macaé - 4 de março de 2024

Sumário

1	Metodologia de Desenvolvimento do Projeto de Engenharia	1
1.1	Introdução a metodologia utilizada	2
1.2	Etapa 0 - Definição do desafio tecnológico	6
1.3	Etapa 1 - Elaboração do pré-projeto - esboço	16
1.4	Etapa 2 - Elaboração do contrato - detalhamento do pré-projeto	20
1.5	Etapa 3 - Modelagem de engenharia	23
1.6	Etapa 4 - Ciclos de planejamento, detalhamento e construção/implementação	27
1.7	Etapa 5 - Entrega do produto	32
1.7.1	Cronograma para 24 meses - projeto disciplinas	35
1.7.2	Cronograma para 36 meses - projeto TCC	36

Lista de Figuras

1.1	Etapas de gestão e desenvolvimento do projeto de engenharia - resumo	2
1.2	Etapas de gestão e desenvolvimento do projeto de engenharia	4
1.3	Etapa 0 - Definição do desafio tecnológico	15
1.4	Etapa 1 - Elaboração do Pré-Projeto	17
1.5	Etapa 2 - Elaboração do contrato - detalhamento do pré-projeto	21
1.6	Etapa 3 - Modelagem de Engenharia	24
1.7	Etapa 4 - Ciclos de planejamento, detalhamento e construção	28
1.8	Etapa 5 - Entrega do produto	32

Capítulo 1

Metodologia de Desenvolvimento do Projeto de Engenharia

Apresenta-se aqui a metodologia a ser utilizada no desenvolvimento do projeto de engenharia.

1.1 Introdução a metodologia utilizada

O software a ser desenvolvido utiliza a metodologia de engenharia de software apresentada na disciplina de introdução ao projeto de engenharia, ilustrado de forma compacta na Figura 1.1.

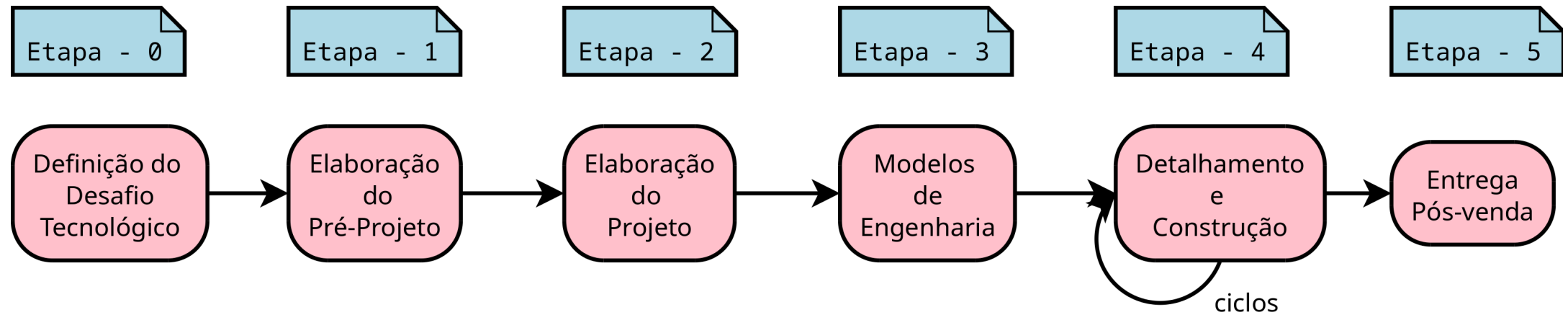


Figura 1.1: Etapas de gestão e desenvolvimento do projeto de engenharia - resumo

Esta metodologia é utilizada nas disciplinas:

- LEP01348 : Introdução ao Projeto de Engenharia.
 - Conceitos de projeto de engenharia, metodologia científica e gestão.
- LEP01447 : Programação Orientada a Objeto em C++.
 - Conceitos de algoritmos e programação.
- LEP01446 : Projeto de Software Aplicado à Engenharia (antiga programação prática).
 - Desenvolvimento prático de um projeto completo.

A Figura 1.2 apresenta uma visão mais detalhada.

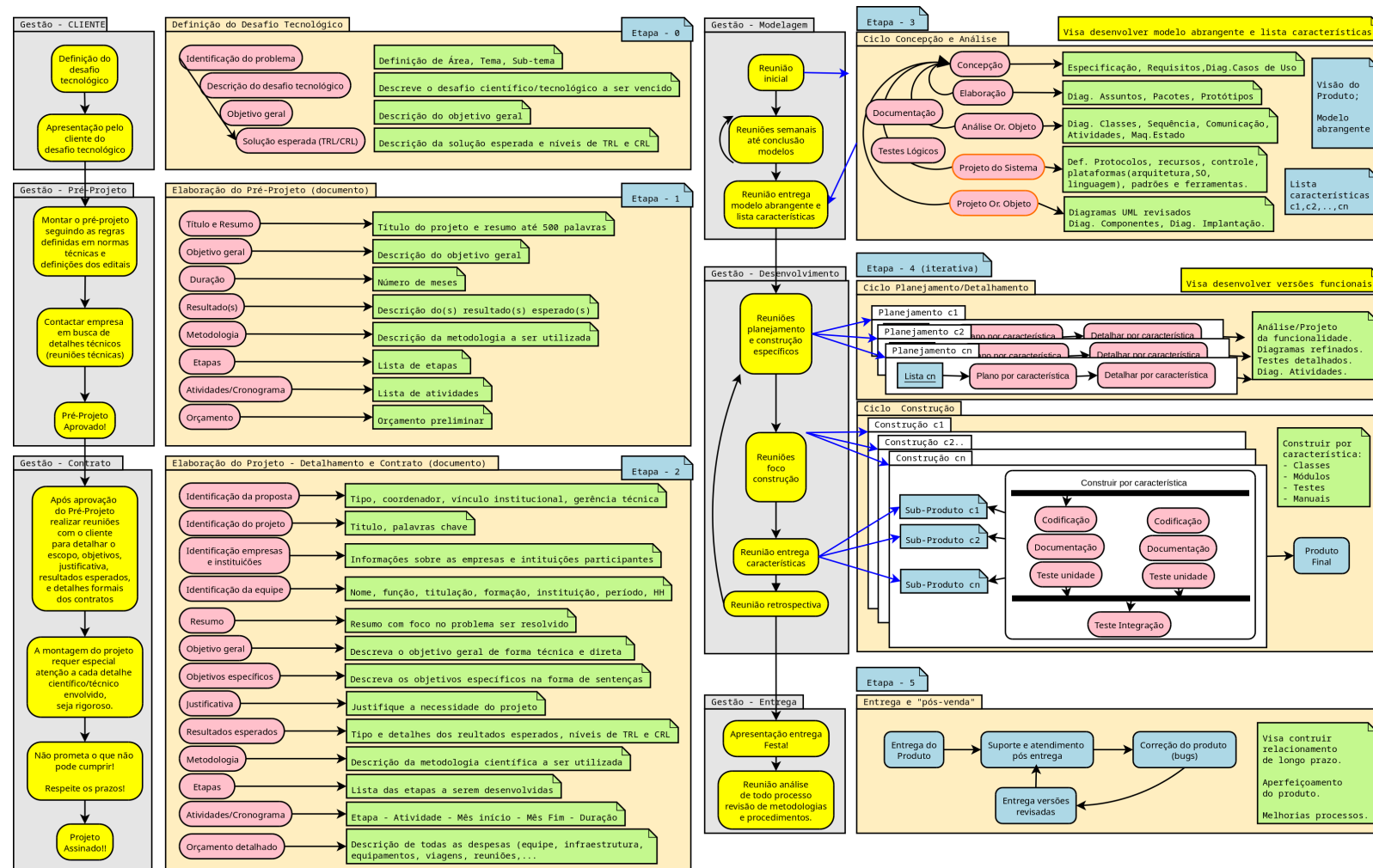


Figura 1.2: Etapas de gestão e desenvolvimento do projeto de engenharia

Entre as referências utilizadas podemos citar:

- UML: [Blaha and Rumbaugh, 2006, Rumbaugh et al., 1994].
- Projetos: [autores, 2017, Inc, 2021, Pires, 2012, Woiler, 1996].
- Gestão de Projetos: [Abrantes, 2020, de Logística e Tecnologia da Informação, 2011, Heldman, 2005, de Moura Menezes, 2018, Pahl, 2005, Valeriano, 2015, Rosa, 2007]
- Produtos: [Abrantes, 2020].

1.2 Etapa 0 - Definição do desafio tecnológico

- A primeira etapa do processo de desenvolvimento de nosso produto tecnológico é a definição pelo cliente (um professor, um engenheiro de empresa) do problema a ser resolvido, o escopo do problema de engenharia.
- Deve apresentar o desafio tecnológico o objetivo geral e a solução esperada.
- Veja Figura 1.3. Veja a seguir as sub-etapas associadas:
- O cliente interessado lê as informações da metodologia de desenvolvimento adotada na disciplina de "Introdução ao Projeto de Engenharia".
 - Entrar no link ProjetoEngenharia-1-TituloProjeto-Modelo e ler o documento
 - * Metodologia Projeto Engenharia (atualizar!).
 - Também pode olhar o modelo padrão para os projetos de engenharia disponibilizado em
 - * Modelo completo - etapas 0-5. (atualizar!)
 - * Modelo específico - etapas 2-5.
 - Se tiver dúvidas sobre a metodologia contactar o professor coordenador da disciplina
 - * e-mail: bueno@lenep.uenf.br.

- Estando em acordo com a metodologia empregada o cliente deve informar o interesse em montar um desafio tecnológico. Neste caso enviar para o email bueno@lenep.uenf.br os seguintes dados:
 - Assunto do email:
 - * Desafio Tecnológico: Título
 - Conteúdo do email:
 - Nome do cliente (professor ou responsável na empresa)
 - * Telefone:
 - * E-mail:
 - * Informar se já tem cadastro no *github* (informe o login no github se já tiver).
 - Título do desafio tecnológico
 - * Breve resumo/rascunho tecnológico [até 500 palavras]:
 - Caso já tenha um ou mais alunos interessados informar o(s) dado(s) do(s) mesmo(s)
 - * Nome do aluno
 - Telefone
 - E-mail
 - Informar login do alunos no *github*

Nota: limite padrão de até 3 alunos por projeto.

- O coordenador da disciplina analisa os dados e dá um retorno.
 - Caso o rascunho da proposta seja rejeitado, explica os motivos, possibilitando correções e reenvio do rascunho da proposta.
 - Um rascunho de proposta que esteja fora do escopo das disciplinas de modelagem computacional será rejeitado.
 - Por exemplo, uma proposta que envolva apenas a parte de experimentos de laboratório sem o desenvolvimento dos softwares associados será rejeitada.
 - Toda proposta deve incluir o desenvolvimento de um software em C++, podendo ser um software educativo (simuladores didáticos), um software que resolva algoritmos de engenharia (algoritmos computacionais), desenvolvimento de uma biblioteca computacional que dê suporte a problemas típicos de engenharia, simuladores de problemas relacionados a engenharia de petróleo, simuladores relacionados a problemas de geologia, geoquímica, geofísica, petrofísica, engenharia de poço, engenharia de elevação e escoamento, engenharia de reservatório; simuladores que tratem de problemas numéricos utilizando volumes finitos, diferenças finitas ou elementos finitos. Simuladores que envolvam o uso de IA - Inteligência Artificial e aplicações. Em resumo, devem tratar da solução de problemas de engenharia de petróleo e/ou modelagem computacional aplicada.
 - Note que a metodologia pode ser utilizada para desenvolver projetos de engenharia que não sejam de software, mas, neste caso, precisa encontrar professores para acompanhar as atividades.

Nota: No PPC - Projeto Pedagógico do Curso de Engenharia de Petróleo, o TCC - Trabalho de Conclusão de Curso, pode ser desenvolvido de diferentes formas: Monografia (indivíduo) ou Projeto de Engenharia (indivíduo ou grupo). Este documento trata apenas do projeto de engenharia. Se estiver interessado em desenvolver IC - Iniciação Científica e um

trabalho no formato de monografia, veja o modelo

<https://github.com/ldsc/ModeloDiretorio-Aluno-Bolsista-TCC-IC-MSD-DSC-Projeto>.

- Caso o rascunho da proposta seja aprovado, o coordenador da disciplina informa o cliente (e alunos caso já tenha).
 - O cliente e alunos que ainda não tem cadastro no *github*, devem se cadastrar e informar ao professor Bueno o login dos membros da equipe.
 - * Enviar email:
 - Assunto: Título do projeto.
 - Conteúdo: nome e login no github.
- O coordenador da disciplina
 - Clona o modelo de projeto da disciplina, criando um projeto novo, com o título do projeto informado.
 - Adiciona no projeto o cliente (professor/engenheiro) e alunos (caso já tenha os alunos).
 - * Note que para adicionar o cliente e alunos no projeto é necessário o cadastro prévio dos mesmos no *github*.
 - Envia para o cliente e alunos o link do *github*, de forma que todos poderão acessar o projeto no modo leitura e escrita.
- O cliente

- Baixa o modelo de projeto criado para sua máquina usando o link informado.
- Acessa o modelo de documento com a descrição do desafio tecnológico.
 - * Se ainda não tem o LyX instalado deve fazer a instalação.
 - * Caso não saiba usar o LyX pode gerar a proposta de desafio tecnológico usando outro editor (como word ou google docs) e repassar para o aluno para que coloque no formato do LyX. Os alunos sabem usar o LyX, vem o mesmo na disciplina de projeto.
- Preenche os dados da proposta do desafio tecnológico.
- Gera o pdf associado.
- Envia os arquivos gerados para o servidor do *github* (lyx e pdf).

* Comandos:

```
git add .  
git commit -m "gerado o desafio tecnologico titulo"  
git push
```

Nota: Se não tiver nenhum conhecimento de *git/github* e *lyx*, pode acessar diretamente o documento 1-DesafioTecnologico-2-DescricaoDaProposta.lyx e editar o mesmo. Após concluir a descrição da proposta enviar para o coordenador da disciplina.

* Note que as propostas de desafios tecnológicos podem ser submetidas sem que tenhamos alunos interessados, neste caso ficam num banco de dados disponibilizado aos alunos no site do *github*. Na prática a intenção é termos a disposição dos alunos uma série de desafios tecnológicos a serem resolvidos, assim a equipe de alunos escolhe um desafio que lhe interessa (semelhante a feira de ideias usada na pós-graduação).

- Requisitos:

- O cliente e alunos devem ter ciência:

- Prazo:

- * O prazo total para execução do projeto é de 18 a 24 meses se estiver limitado ao conjunto de disciplinas de modelagem computacional.

- * Projetos de até 36 meses podem ser desenvolvidos se for o trabalho de TCC na forma de projeto de engenharia. Neste caso teremos mais "ciclos de desenvolvimento" (Figura 1.7).

- Escopo:

- * O projeto deve estar no escopo do curso, competências e habilidades listadas no PPC (para ter uma ideia rápida olhe a lista de disciplinas da grade, olhe os projetos desenvolvidos pelos professores, olhe as atividades de empresas de engenharia que trabalham com inovação científica e tecnológica).

- * O projeto deve ter orientador ou co-orientador do LENEP.

– Softwares utilizados:

- * Os documentos do projeto seguem um modelo pré estabelecido que usa o editor de texto LyX.
- * O Editor LyX é apresentado aos alunos em sala de aula e temos manuais e vídeos disponibilizadas na internet e google classroom.
- * O Editor LyX tem sido usado pelos nossos alunos desde 2004. Seu uso requer leitura e aprendizado do uso do LyX (e conceitos básicos de TeX/LaTeX).
- * Requer o conhecimento do mecanismo de trabalho em equipe usando os softwares *git* e *github*. O software *git* é apresentado aos alunos em sala de aula e temos manuais e vídeos disponibilizadas na internet e google classroom. O software *github* é apresentado aos alunos em sala de aula e temos manuais e vídeos disponibilizadas na internet e google classroom.
- * A parte de modelagem envolve o uso do modelador *umbrello*.
- * A parte de desenvolvimento do software requer conhecimentos de C++, da biblioteca padrão de C++ e ferramentas como compiladores (g++, clang), editores específicos (*emacs*, *vscode*), IDEs.
- * Uma lista com todos os softwares é apresentada na seção 1.7.2.

– Modelagem física-matemática (química-matemática)

- * Os membros da equipe devem compreender os conceitos de matemática, física e de engenharia associados ao desafio tecnológico escolhido.
- * No Capítulo de Elaboração os alunos devem desenvolver todo o raciocínio lógico, apresentar as ideias, as equações e

modelos.

– Equipes:

- * A montagem da equipe é ponto central e deve ser realizada com cuidado. Esteja atento para os seguintes pontos:
- * O tema do projeto deve ser de interesse dos membros da equipe.
- * Fugam de temas que parecem mais fáceis mas não são interessantes (escolha algo que gosta, tem interesse efetivo, pois como é um trabalho de longo prazo, para se manter motivado o tema tem de ser de seu interesse).
- * Os membros da equipe devem conseguir se reunir fisicamente e na forma *online* para esclarecimento das atividades desenvolvidas e apoio mútuo no esclarecimento de dúvidas dos problemas associados (teóricos e práticos).
- * Um aluno que tenha reprovado numa das disciplinas poderá continuar na mesma equipe e projeto.

– Defesas:

- * O projeto tem defesa em equipe (ideia geral) e defesas individuais (para confirmar a efetiva participação e a capacidade de desenvolver novas coisas solicitadas pelo professor).
- * O aluno também poderá mudar de equipe e tema, mas isso deve ser evitado pois usualmente gera atrasos.

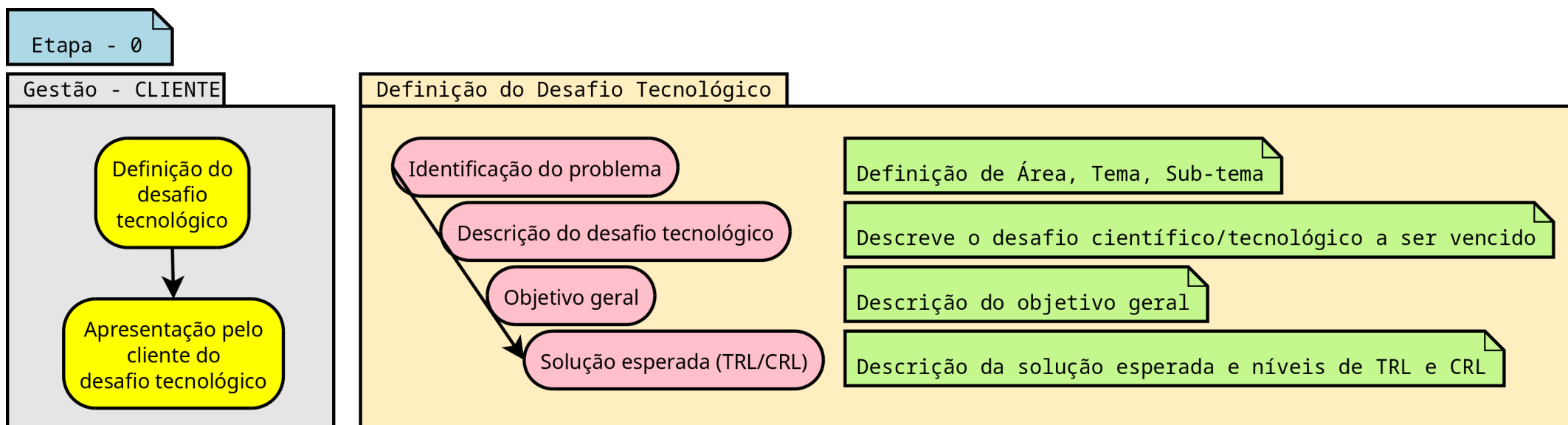


Figura 1.3: Etapa 0 - Definição do desafio tecnológico

1.3 Etapa 1 - Elaboração do pré-projeto - esboço

Após aprovação do desafio tecnológico, criação do projeto no *github* e adição dos membros, a equipe deve elaborar o pré-projeto. A sequência é apresentada na Figura 1.4.

Nota: este modelo segue como exemplo o modelo de desafio tecnológico adotado pela Petrobras.

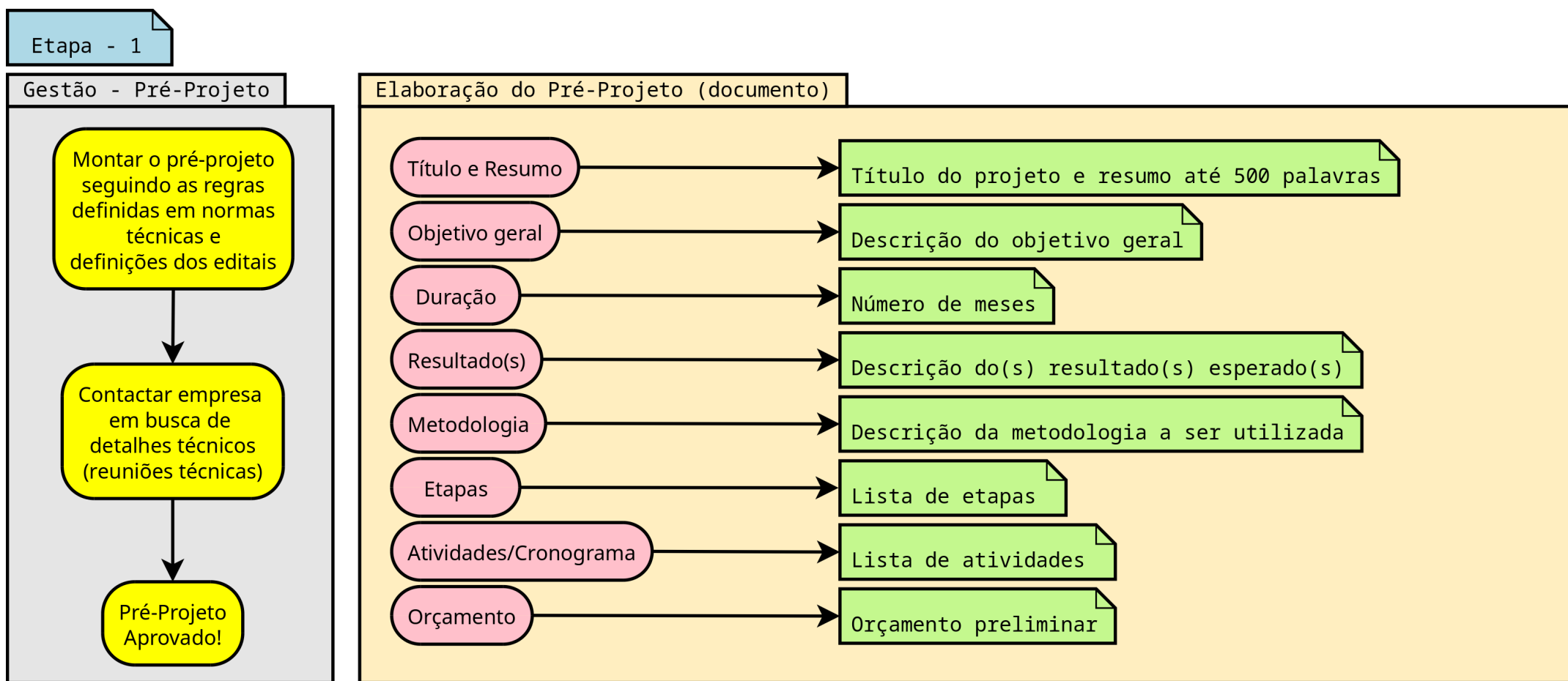


Figura 1.4: Etapa 1 - Elaboração do Pré-Projeto

- A equipe elabora o pré-projeto.
- Os alunos encaminham o pré-projeto.
 - Os encaminhamentos são sempre pelo *github*.
- O pré-projeto deve ser aprovado pelo cliente que elaborou o desafio e pelo coordenador da disciplina.
 - Note que temos um desafio tecnológico definido pelo cliente e pre-aprovado pelo professor e um projeto no github (vazio).
 - Note que os alunos elaboraram um pré-projeto (rascunho de solução preliminar), tendo como base o desafio tecnológico e seu entendimento do que deve ser feito.
 - Note que este pré-projeto deve ser aprovado pelo cliente.
 - Esta análise é necessária para evitarmos problemas de entendimento do que deve ser feito.
 - Tanto o cliente quando a equipe de alunos devem estar de acordo com o que será realizado antes da elaboração do projeto detalhado/executivo¹ e do contrato.

1

* Em engenharia um projeto detalhado é chamado de projeto executivo, o mesmo requer muito trabalho para ser elaborado e normalmente é cobrado.

Notem que temos alguns requisitos que devem ser aprimorados nas disciplinas e ao longo do desenvolvimento do projeto para efetiva capacidade de desenvolver as atividades associadas.

- Uso do *LyX*:
 - Sugere-se ler o tutorial e manual de uso do software.
- Uso do *git/github*:
 - É necessário assistir vídeos e ler material sobre os softwares *git* e *github*.
 - Sugere-se que criem seu próprio repositório e pratiquem os comandos e ações de uso do *git* e *github* antes de executar os comandos diretamente no projeto da disciplina.
 - Mantenham *backups* dos arquivos.
 - Importante executar os comandos com calma, prestando atenção, lendo as mensagens apresentadas.

1.4 Etapa 2 - Elaboração do contrato - detalhamento do pré-projeto

Aprovado o pré-projeto podemos escrever o projeto em sí, o mesmo inclui por exemplo mais detalhes das instituições envolvidas, objetivos específicos, justificativa e resultados esperados, além de cronograma e orçamento. A sequência é detalhada na Figura 1.5.

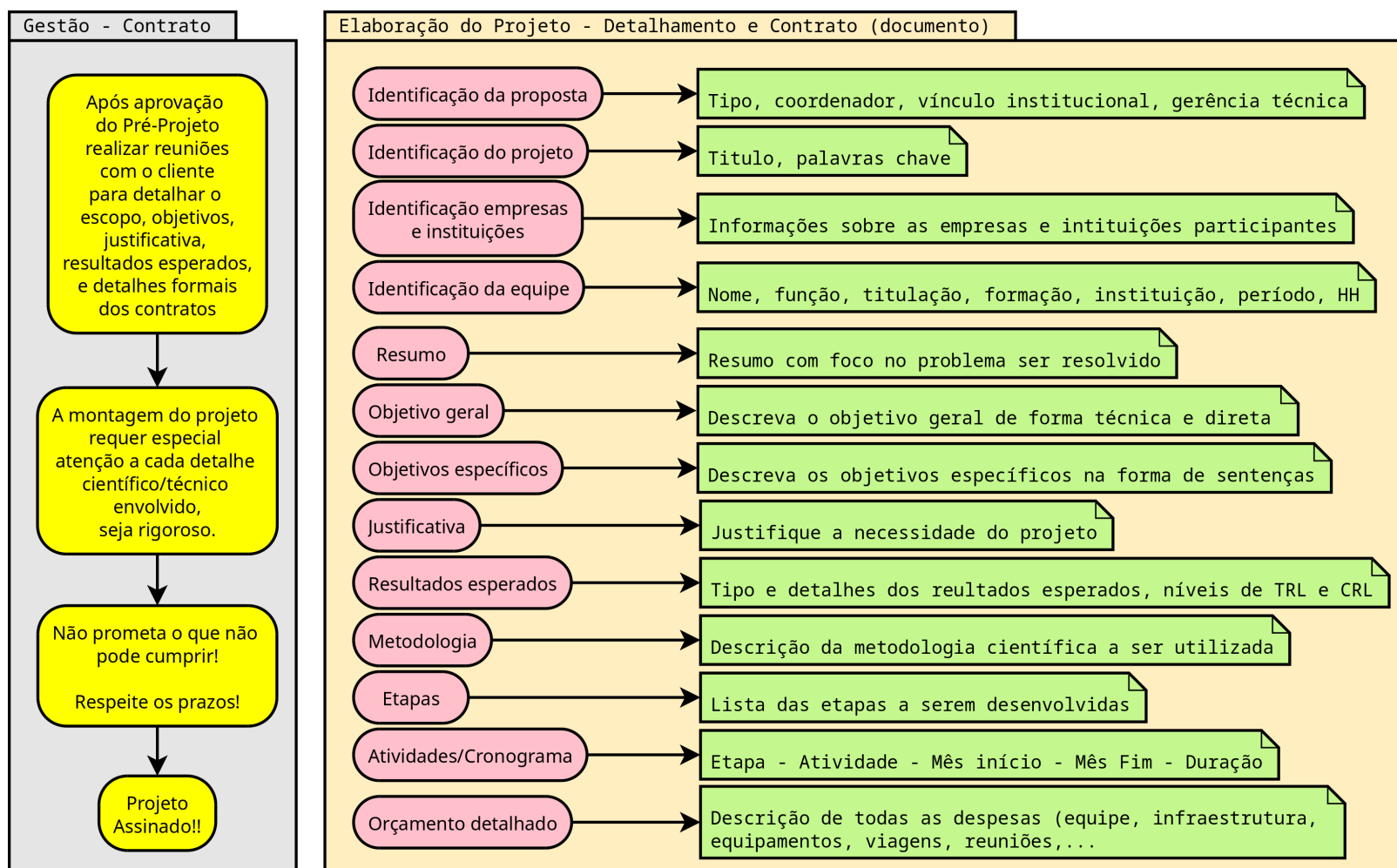


Figura 1.5: Etapa 2 - Elaboração do contrato - detalhamento do pré-projeto

- Após aprovação do pré-projeto os alunos detalham o mesmo gerando o contrato.
 - Os dados do pré-projeto podem ser copiados para o projeto.
 - A seguir tudo deve ser mais detalhado.
 - As etapas associadas devem ser bem definidas.
 - O cronograma deve ser realista.
 - O orçamento deve ser bem realizado. Por exemplo, para realização do projeto serão necessários computadores com determinada configuração, os mesmos devem ser orçados. Esta etapa cobre requisitos da grade associados a temas de administração que foram exigidos para aprovação do projeto pedagógico.

1.5 Etapa 3 - Modelagem de engenharia

A etapa de modelagem de engenharia é apresentada em detalhes na disciplina, os alunos aprendem os conceitos associados e colocam em prática fazendo modelos e diagramas usando o modelador Umbrello. Também é material de trabalhos entregues via *google classroom* e provas.

Nesta etapa a equipe usa o modelador *umbrello* para desenvolver os modelos de software através da metodologia UML - *Unified Modeling Language*, em português *Linguagem de Modelagem Unificada*.

Detalhas da UML consulte o site da organização internacional que a regulamenta <https://www.uml.org/>.

A sequência desta etapa é detalhada na Figura 1.6.

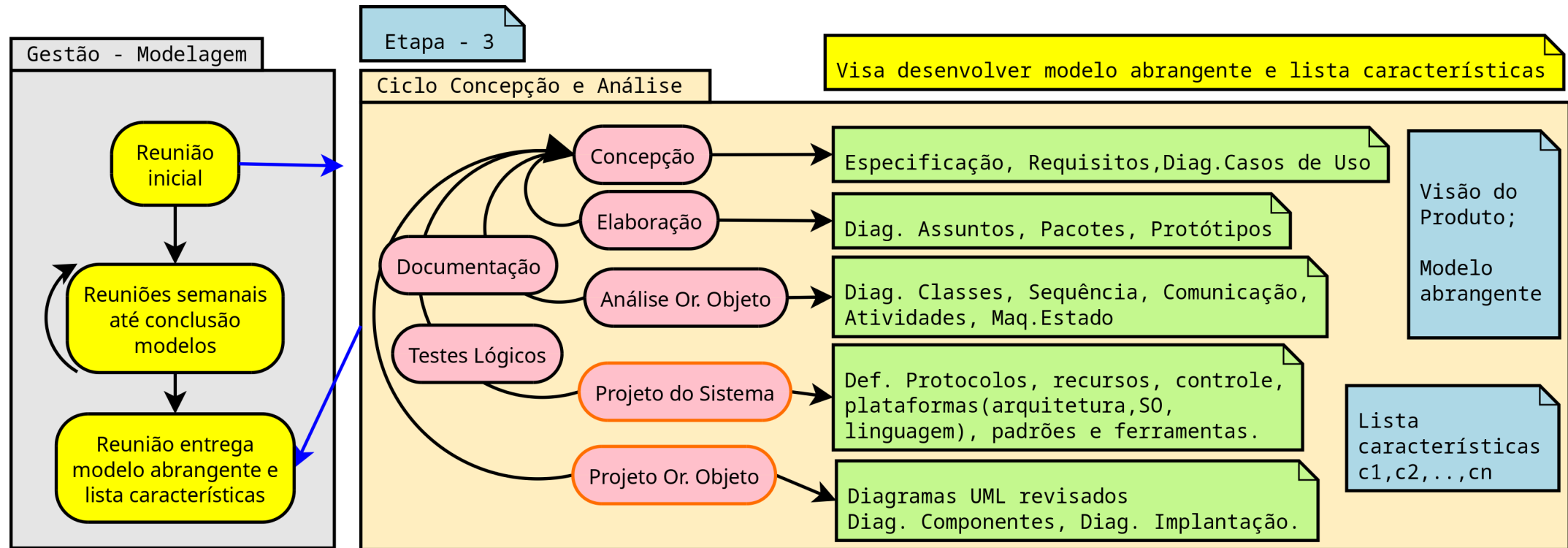


Figura 1.6: Etapa 3 - Modelagem de Engenharia

- Nesta etapa os alunos devem desenvolver um modelo abrangente do sistema de software a ser desenvolvido. Passa pelas etapas de:
 - Concepção
 - Elaboração
 - Análise Orientada a Objeto
 - Projeto do Sistema
 - Projeto Orientado a Objeto
 - Realização de Testes Lógicos
 - Documentação (geração dos documentos de modelagem e diagramas associados).
 - Geração da Lista de Características ou *features* que serão implementados.

- Observações:
 - Um dos objetivos desta modelagem é o aluno, futuro engenheiro, aprender a lidar com o todo e suas partes, com a separação da estrutura (partes estáticas), e a parte dinâmica (objetos se relacionando). Identificar e ver o sistema dos pontos de vista macro e micro.
 - O modelo repassado facilita este trabalho, pois além de já estar formatado contém dicas e sugestões práticas.
 - Temos mais de 40 projetos disponibilizadas no site do ldsc, a grande maioria desenvolvida por ex-alunos.
 - É fundamental gerar uma documentação que dê uma visão clara de todo o sistema, mas, nesta etapa, não é necessário pormenorizar tanto. Por exemplo, os diagramas de máquina de estado e diagramas de atividades aparecem pouco aqui. O detalhamento da parte micro será feita nos ciclos de desenvolvimento.
 - Outras imagens detalhando esta etapa estão disponibilizadas neste link.

1.6 Etapa 4 - Ciclos de planejamento, detalhamento e construção/implementação

Esta etapa é apresentada na Figura 1.7.

Temos duas atividades principais:

- Ciclo planejamento/detalhamento:
 - Usa-se o modelador *umbrello* para desenvolver os modelos micro do software (ex.: diagramas de atividades).
 - É neste momento que os algoritmos de cálculo são detalhados através dos diagramas de atividade (nome classe::nome método).
 - Havendo necessidade detalhe o comportamento dos objetos usando diagramas de máquina de estado (nome classe).
- Ciclo de construção:
 - Note que o modelador UML pode gerar uma casca inicial do software para diferentes linguagens de programação, incluindo C++.
 - Note que o *umbrello* pode exportar e importar códigos, sendo possível atualizar os diagramas após alterações nos códigos.
 - Usa-se uma IDE ou editor de texto para implementar os códigos.
 - Usa-se um compilador.
 - Uma lista dos softwares é encontrada na seção 1.7.2.

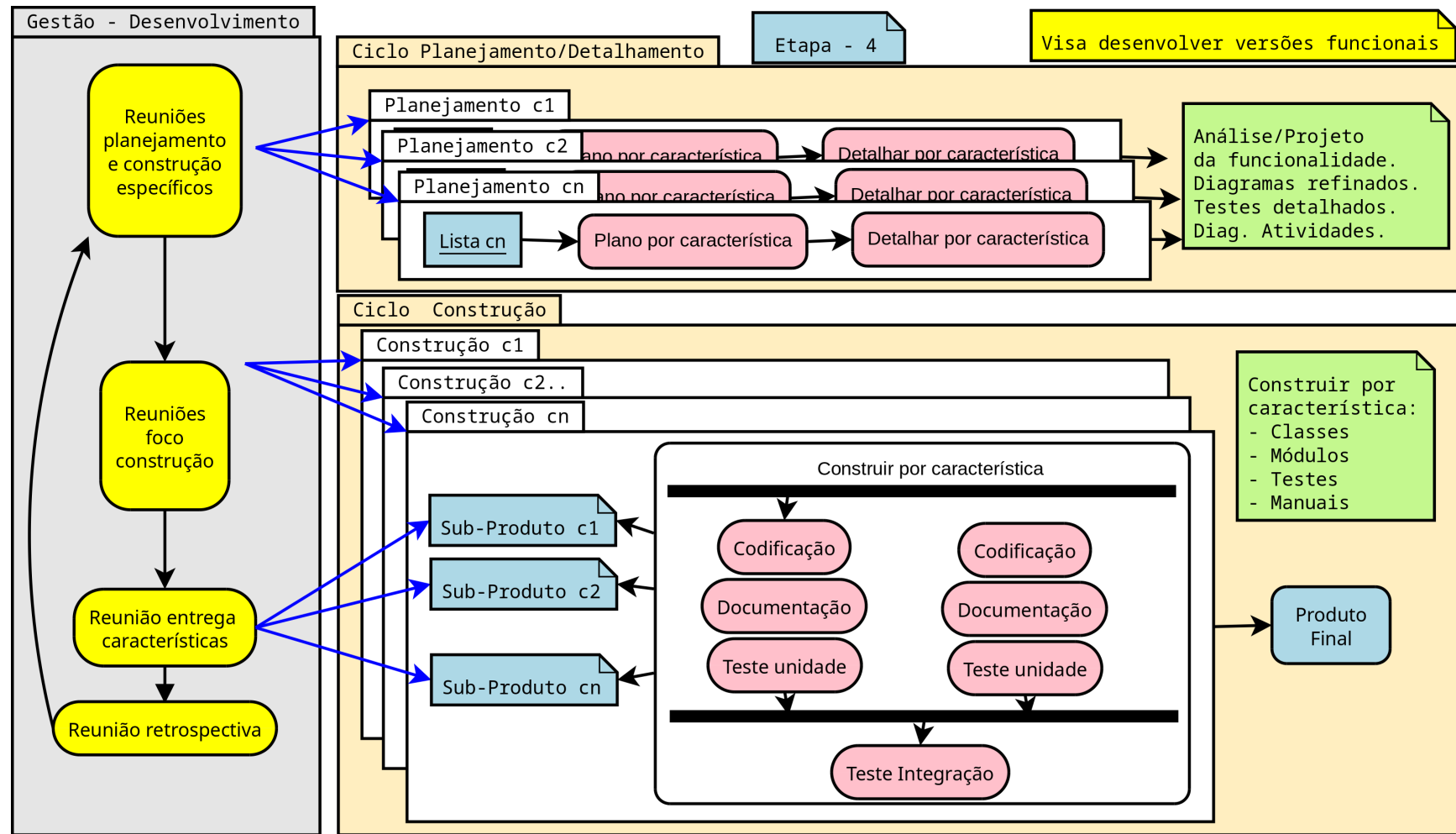


Figura 1.7: Etapa 4 - Ciclos de planejamento, detalhamento e construção

- A cada ciclo de planejamento, detalhamento e construção, a equipe seleciona um grupo de funcionalidades/*features* a serem implementadas.
- O conjunto de funcionalidades a serem implementadas devem "caber" dentro do prazo do ciclo e estar de acordo com o cronograma.
- A seguir é feito um planejamento detalhado das características a serem implementadas, refinando os diagramas e adicionando novos diagramas, como exemplo diagramas de máquina de estado mais detalhados e diagramas de atividades. É neste momento que diversos detalhes micros aparecem e devem ser escritos, tanto a estrutura (diagrama de classes), como a dinâmica através dos diagramas de atividades.

Nota: Lembre-se que os diagramas visam dar aos membros da equipe de engenharia uma visão clara dos conceitos/classes/funcionalidades a serem implementadas. Não é necessário exagerar, criando diagramas para conceitos simples ou algoritmos e rotinas amplamente conhecidos. Uma forma de identificar se determinado diagrama é necessário é verificar se todos os membros da equipe entendem determinado conceito sem os diagramas, caso afirmativo o mesmo é desnecessário. Faça diagramas para partes mais complexas e que requeiram algoritmos mais exigentes.

- A construção ou implementação dos códigos deve ser feita seguindo-se as normas e procedimentos padrões, com uso de ferramentas adequadas, como editores, compiladores, debuggers e *profillers*. Em alguns casos pode-se usar uma IDE como *Qt-Creator*, *Kdevelop*, entre outros.
- A construção dos códigos deve seguir as melhores práticas.

- Na disciplina damos preferência para uso de editores modernos como emacs, vscode, com plugins e funcionalidades adequadas instaladas.
- Leia os manuais e tutoriais de uso do editor ou IDE escolhidos.
- Existem centenas de editores e IDEs, só podemos dar ajuda nos padrões utilizados na disciplina. Isto significa que se a equipe/aluno usar outras ferramentas faz isso por sua conta e risco pois não terá nenhum tipo de suporte.
- Os manuais são gerados obrigatoriamente no LyX.
 - * Visa ensinar a usar editores profissionais.
- Os modelos UML devem ser gerados em plataforma que permita que alunos que venham a fazer a disciplina no futuro possam manipular os arquivos.
 - * A ideia é que teremos diversas versões do software desenvolvidas de forma incremental por diferentes equipes.
 - * Uma versão dos diagramas no *umbrello* é necessária para permitir a continuidade do projeto por outras equipes.
- Após implementar os códigos é preciso realizar os testes, como exemplos os testes de unidade e os testes de integração.
 - Não deixe para fazer os testes do sistema só no final.
 - A cada ciclo de desenvolvimento as *features* desenvolvidas devem ser testadas.
- No final de cada ciclo de planejamento/detalhamento/construção a equipe deve realizar uma reunião física para discutir o andamento do projeto.

- Conversar sobre o que foi feito, o que falta fazer e correções de rumo (ajustes).
- Fazer uma apresentação informal da evolução.
- Esta reunião deve seguir os procedimentos da metodologia SCRUM.

1.7 Etapa 5 - Entrega do produto

A sequência é detalhada na Figura 1.8.

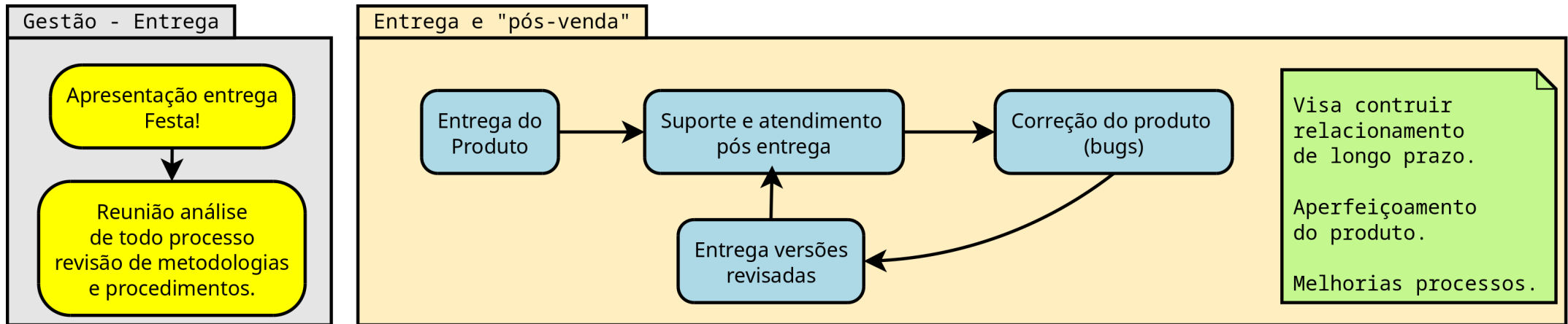


Figura 1.8: Etapa 5 - Entrega do produto

- A versão final do sistema de software deve passar por um sistema de testes pormenorizados.
 - Testar se todas as especificações foram atendidas.
 - Testar se todos os requisitos foram atendidos.
 - Testar se cada uma das classes desenvolvidas atende seus requisitos e esta completamente funcional.
 - Testar exemplos reais aplicados a engenharia.
 - * No mundo real a quantidade de dados a serem processados numa simulação de engenharia costuma ser enorme. Normalmente os alunos não tem em casa computadores para atender esta demanda. A solução é criar um conjunto de dados reduzido e que possa ser utilizado nos testes. Mas para versão final é necessário rodar o simulador desenvolvido com dados reais. Isto permite verificar a estabilidade do sistema e tempos de processamento.
 - Testar se os manuais estão completos dando os mesmos para um terceiro ler, instalar e testar o sistema.
 - * Uma equipe A pode passar seu manual do usuário para a equipe B instalar e testar o software e a equipe B pode passar seu manual para a equipe A instalar e testar.
- No caso da disciplina projeto de software aplicado a equipe deve entregar a versão final com 15 dias de antecedência e marcar a data da defesa.
 - Os 15 dias são necessários pois o professor precisa de tempo para ler, instalar e testar o sistema (executar os testes implementados no manual).

- Caso o aluno ou equipe queiram dar continuidade ao projeto para apresentar o mesmo como TCC, devem considerar a adição de novas funcionalidades/*features*.
 - Um exemplo que já tivemos foi de alunos que na disciplina de programação prática entregaram um sistema totalmente funcional e no TCC adicionaram novas funcionalidades e fizeram versões com interface gráfica amigável em Qt.

Cronograma

1.7.1 Cronograma para 24 meses - projeto disciplinas

- A seguir um exemplo de cronograma. O cronograma efetivo dependerá do projeto e da equipe.
- Os primeiros 4 meses estão relacionados as etapas realizadas na disciplina de Introdução ao Projeto de Engenharia. As demais etapas estão associadas às demais disciplinas (LEP - 01447: Programação Orientada a Objeto em C++ e LEP - 01446: Projeto de Software Aplicado à Engenharia).

Mês	1	2	3	4	5-7	8-10	11-13	14-16	17-19	20-22	23-25	..36 (TCC)
Etapas 0	x											
Etapas 1	x	x										
Etapas 2			x									
Etapas 3			x	x								
Etapas 4.c1					x	x						
Etapas 4.c2						x	x					
Etapas 4.c3							x	x				
Etapas 4.c4								x	x			
Etapas 5									x	x	x	

1.7.2 Cronograma para 36 meses - projeto TCC

- A seguir um exemplo de cronograma. O cronograma efetivo dependerá do projeto e da equipe.
- Os primeiros 4 meses estão relacionados as etapas realizadas na disciplina de Introdução ao Projeto de Engenharia. As etapas estão associadas às disciplinas (LEP - 01447: Programação Orientada a Objeto em C++ e LEP - 01446: Projeto de Software Aplicado à Engenharia) podem atingir 24 meses. Adicionando as atividades do TCC podemos ter até 36 meses.

Mês	1	2	3	4	5-7	8-10	11-13	14-16	17-19	20-22	23-25	..36 (TCC)
Etapa 0	x											
Etapa 1	x	x										
Etapa 2			x									
Etapa 3			x	x								
Etapa 4.c1					x	x						
Etapa 4.c2						x	x					
Etapa 4.c3							x	x				
Etapa 4.c4								x	x			
Etapa 4.c5									X	X		
Etapa 4.c6										X	X	
Etapa 5										x	x	x

Lista de softwares utilizados

- Na disciplina usamos softwares livres que os alunos podem acessar, instalar e usar, sem problemas com direitos autorais e sem pirataria.
- Exemplos de softwares usados em sala de aula e *links* para acesso:
- Sistema operacional GNU/Linux-Fedora:
 - Sistema operacional simples e fácil de usar e configurar.
- Bash - GNU Bourne-Again Shell:
 - Ambiente de *shell* mais utilizado no mundo todo, coloca disposição do aluno um conjunto de aplicativos que permitem melhorar sua produtividade e eficiência na gestão de diretórios, arquivos e configurações de sistemas.
- Git - Distributed revision control system:
 - Software mais usado no mundo no controle de versões de arquivos de projetos.
- GitHub - Where the world builds software:
 - Maior site de armazenamento de projetos de engenharia (em especial de softwares).

- LyX - A document processor:
 - Editor de texto profissional que é utilizado por nossos alunos desde 2004.
 - Interface para uso simplificado do TeX/LaTeX.
- UML Modeller Umbrello UML Modeller - Free UML diagram editor:
 - Modelador UML simples e prático (embora apresente alguns bugs!).
- Kate - A versatile text editor:
 - Editor de texto simples e prático que tem como base a biblioteca Qt.
 - Também é possível usar o *gedit* que tem como base a biblioteca GTK.
- vscode:
 - Editor de texto profissional com centenas de recursos avançados.
 - Seu uso requer treinamento.
- emacs:
 - Editor de texto profissional com centenas de recursos avançados.

- Seu uso requer treinamento.
 - g++ - GNU Compiler Collection
 - Conjunto de compiladores da GNU, compilador mais utilizado no mundo.
 - Clang
 - Fornece um *front-end* de linguagem e infraestrutura de ferramentas para linguagens de programação da C/C++/Java, etc.
 - Caso o cliente queira usar ferramentas proprietárias deverá fornecer as mesmas
 - Deve pagar as licenças associadas com instalação nos equipamentos do LENEP.
 - Não fornecemos nenhum suporte para estes softwares.
 - As versões finais devem ter documentos gerados em formatos compatíveis com softwares livres, garantindo a viabilidade de continuidade do projeto. Ou seja, temos de garantir que futuras equipes possam modificar/atualizar o projeto mesmo sem ter acesso aos softwares proprietários.
- * Nota: Só use software proprietário se o mesmo tiver mecanismos para exportar os arquivos em formato que possa ser lido por um software livre. No caso dos diagramas UML os arquivos devem ser compatíveis com o Umbrello.

Site com referências

- Referências: [Blaha and Rumbaugh, 2006, Rumbaugh et al., 1994]
- Você encontra uma lista de referências no site do professor.

Referências Bibliográficas

[Abrantes, 2020] Abrantes, J. (2020). *Projeto e Engenharia de Produtos*. Ciencia Moderna. ISBN-13 : 978-8539910847. 5

[autores, 2017] autores, V. (2017). *Projetos de engenharia - uma introdução*. LTC. ISBN-13 : 978-8521634454. 5

[Blaha and Rumbaugh, 2006] Blaha, M. and Rumbaugh, J. (2006). *Modelagem e Projetos Baseados em Objetos com UML 2*. Campus, Rio de Janeiro. 5, 40

[de Logística e Tecnologia da Informação, 2011] de Logística e Tecnologia da Informação, S. (2011). *Fundamentos em Gestão de Projetos - Construindo Competências para Gerenciar Projetos BRASIL*. Ministério do Planejamento, Orçamento e Gestão (MPOG). 5

[de Moura Menezes, 2018] de Moura Menezes, L. C. (2018). *Gestão de Projetos*. Atlas. 5

[Heldman, 2005] Heldman, K. (2005). *Gerência de projetos*. Elsevier. ISBN 13 : 978-8535216844, Rio de Janeiro. 5

- [Inc, 2021] Inc, P. M. I. (2021). *A Guide to the Project Management Body of Knowledge and the Standard for Project Management*. PMI Project Management Institute. ISBN 13: 978-1628256642. 5
- [Pahl, 2005] Pahl, G. (2005). *Projeto na Engenharia: Fundamentos do Desenvolvimento Eficaz de Produtos - Métodos e Aplicações*. Blucher. ISBN-13: 978-8521203636. 5
- [Pires, 2012] Pires, A. M. S. (2012). *Projeto de Instalações Elétricas e Telecomunicações*. Instituto Superior de Engenharia de Coimbra. 5
- [Rosa, 2007] Rosa, M. O. (2007). *Gerenciamento de projetos de governo*. PMI-DF - PMInforma. 5
- [Rumbaugh et al., 1994] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. (1994). *Modelagem e Projetos Baseados em Objetos*. Edit. Campus, Rio de Janeiro. 5, 40
- [Valeriano, 2015] Valeriano, D. (2015). *Moderno Gerenciamento de Projetos*. Pearson. 5
- [Woiler, 1996] Woiler, S. (1996). *Projetos: planejamento, elaboração, análise*. Atlas. 5