UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE
LABORATÓRIO DE ENGENHARIA E EXPLORAÇÃO DE PETRÓLEO
SETOR DE MODELAGEM MATEMÁTICA COMPUTACIONAL

Disciplina Introdução ao Projeto de Engenharia

Metodologia para Desenvolvimento do Projeto de Engenharia

Versão 2025.1

Prof. André Duarte Bueno

Macaé - 3 de abril de 2025

Sumário

Lista de Figuras

Capítulo 1

Metodologia para 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.

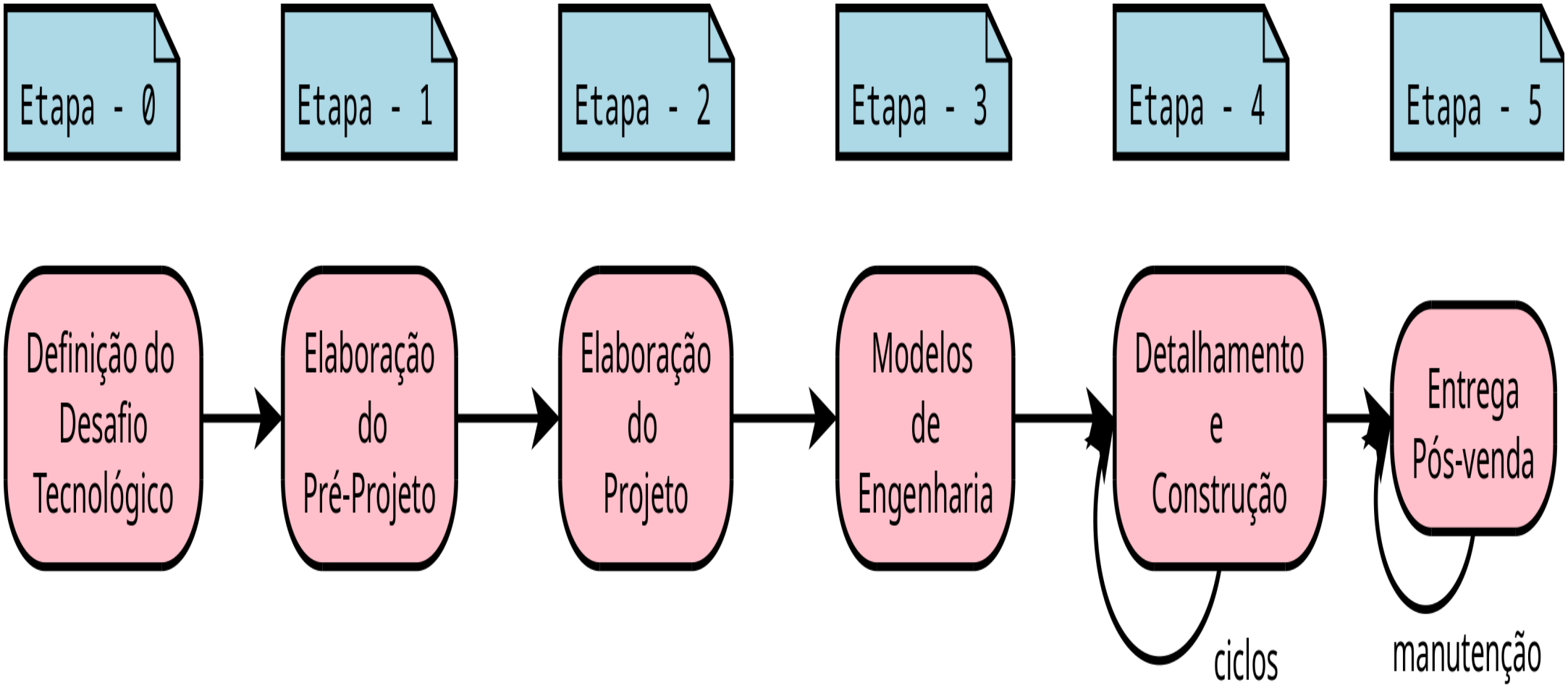

Figura 1.1: Etapas de gestão e desenvolvimento do projeto de engenharia - resumo

Esta metodologia é utilizada nas disciplinas:

- LEP01348 : Introdução ao Projeto de Engenharia.
  - Introdução a modelagem e projeto de engenharia (conceitos e definições);
  - Introdução à metodologia científica e a pesquisa científica e tecnológica (inovação);
  - TRL/CRL, O ciclo do projeto de engenharia e do método científico e tecnológico;
  - Administração e gestão de projeto (planejamento, execução, controle);
  - Administração – uso de metodologias ágeis; ferramentas git, github, github projects (kanbam);
  - Exemplos aplicados ao TCC, engenharia de software e a engenharia de petróleo.
  - Ciclo Planejamento e Detalhamento (concepção - especificação e requisitos, elaboração, análise, projeto) e Ciclos de Construção (do software, testes do funcionamento do software; manutenção e documentação do software desenvolvido).
  - Desenvolvimento das etapas: 0,1,2,3 (Figura 1.1).
- LEP01447 : Programação Orientada a Objeto em C++.
  - Conceitos de algoritmos e programação, sintaxe de C++.
  - Aprender a programar vendo a teoria e a prática com exemplos aplicados.
  - É possível dar desenvolvimento a partes da etapa: 4 (detalhamento e construção - ciclo 1) (Figura 1.1).

- LEP01446 : Projeto de Software Aplicado à Engenharia (antiga programação prática).
  - Desenvolvimento prático de um projeto de engenharia completo.
  - Ciclo Planejamento e Detalhamento (concepção - especificação e requisitos, elaboração, análise, projeto) e Ciclos de Construção (do software usando C++, testes do funcionamento do software; manutenção e documentação do software desenvolvido).
  - Uso de softwares: modeladores (dia, umbrello), editores (kate, gedit, emacs), compiladores (gcc/g++), montadores (make, cmake). IDEs (DEVC++, kdevelop, QtCreator). Sistemas de Controle de Versões (git, github). Entre outras ferramentas usadas no desenvolvimento de softwares de engenharia.
  - Desenvolvimento das etapas: 4,5.

A Figura 1.2 apresenta uma visão mais detalhada.

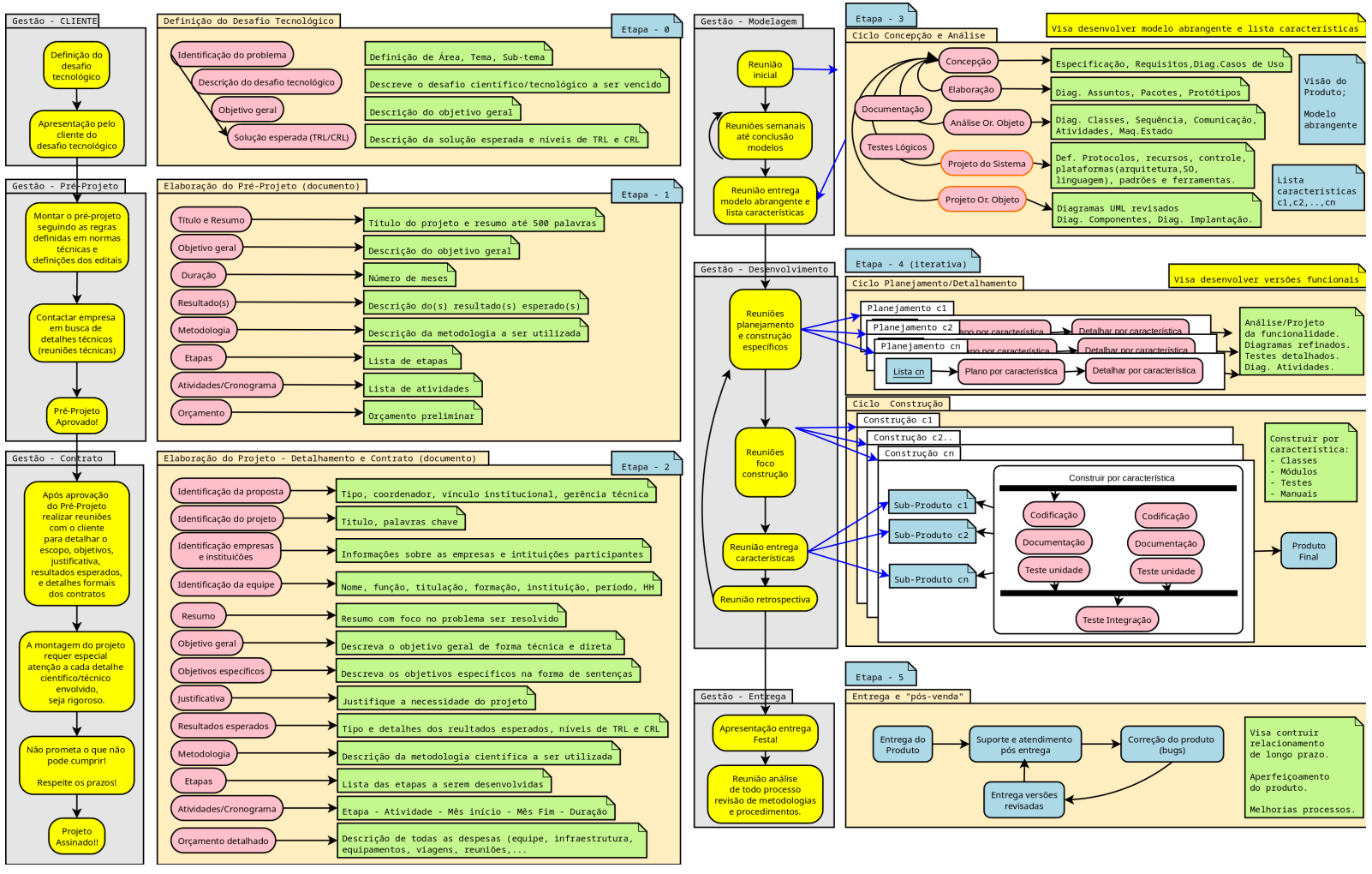

Figura 1.2: Etapas de gestão e desenvolvimento do projeto de engenharia

Entre as referências utilizadas podemos citar:

- UML: [Fowler and Scott, 2005, Rumbaugh et al., 1994, Blaha and Rumbaugh, 2006, Sonerviile, 1993].
- 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].

Note que existe uma relação direta com o conceito de Nível de Maturidade Tecnológica ou TRL (*Technology Readiness Level*) apresentado na Figura 1.3.

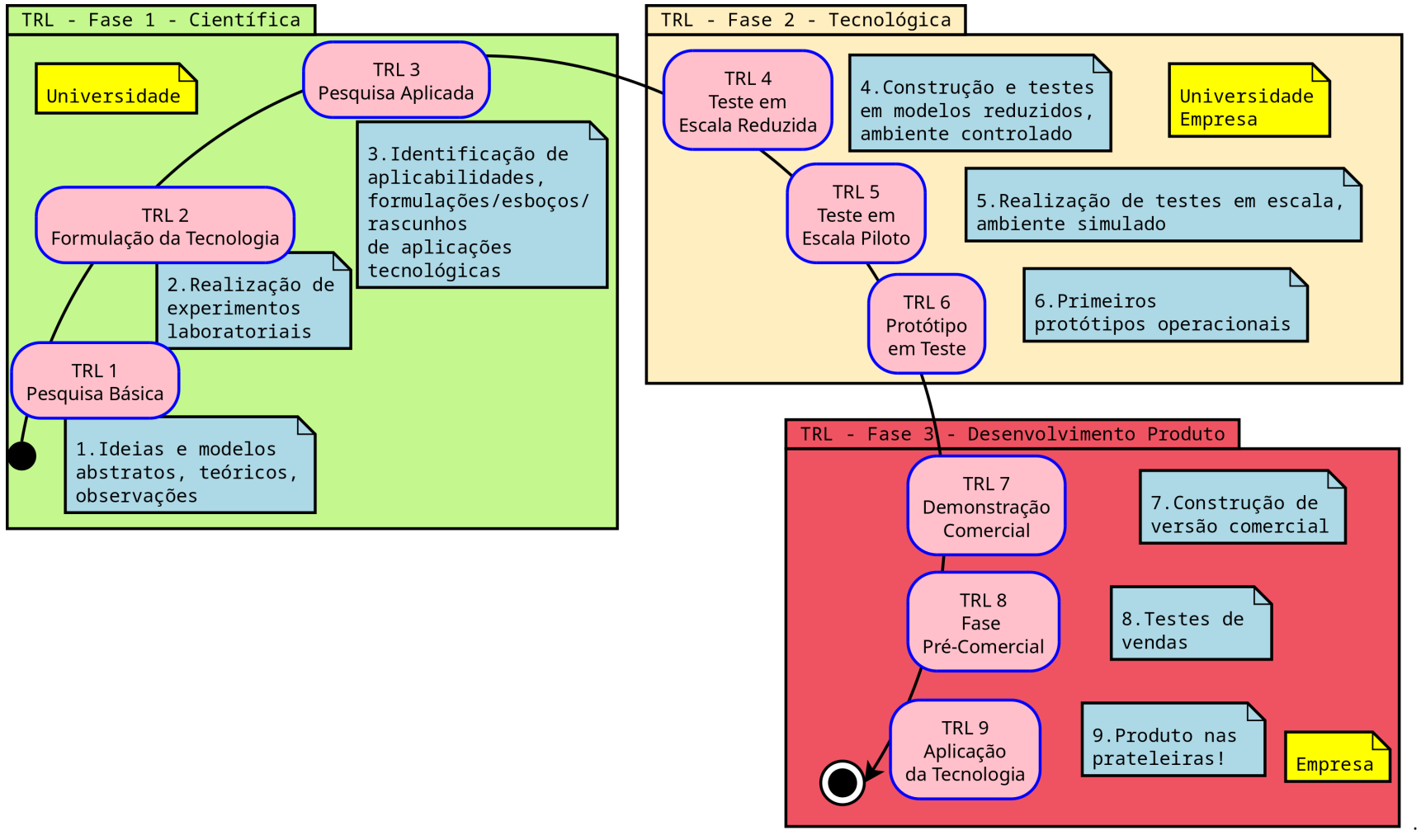

Figura 1.3: Nível de Maturidade Tecnológica ou TRL (*Technology Readiness Level*)

1.2 Etapa 0 - Definição do desafio tecnológico

- A primeira etapa do processo de desenvolvimento do 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 1.4.

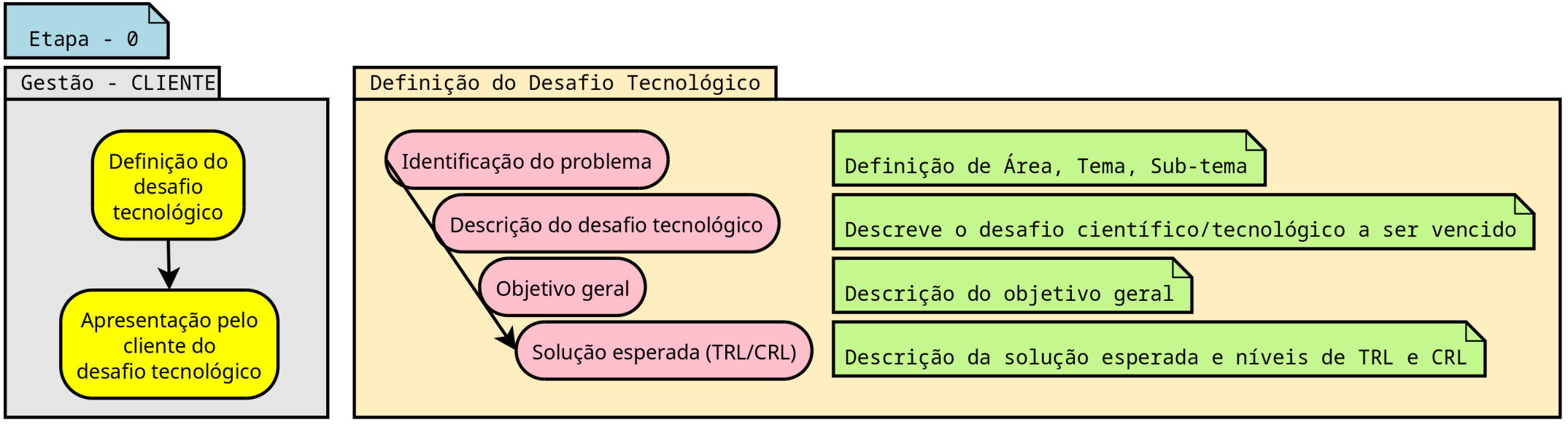

Figura 1.4: Etapa 0 - Definição do desafio tecnológico

- Veja a seguir as subetapas associadas:

- O cliente interessado lê as informações da metodologia de desenvolvimento adotada na disciplina de "Introdução ao Projeto de Engenharia".
  - Metodologia para desenvolvimento do projeto de engenharia (padrão Petrobras) (este documento)
  - Para entender os detalhes da metodologia pode olhar o modelo padrão para:
    * "Proposta de Desafio Tecnológico" disponibilizado em https://github.com/ldsc/LDSC-ProjetoEngenharia-1-PropostaDeDesafi TituloDoDesafioTecnologico (este documento segue um padrão Petrobras).
    * "Modelo para o desenvolvimento do software de engenharia" disponibilizado em https://github.com/ldsc/LDSC-ProjetoEngenh 2-Software-TituloProjeto-ModeloCompleto (inclui diretórios e documentos de suporte para desenvolvimento de todo projeto de engenharia incluindo códigos e testes).
    * Se ainda 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 a proposta de desafio usando este formulário. Os seguintes dados são solicitados:
  - Proposta de Desafio Tecnológico
    * Título do desafio tecnológico:
    * Nome do cliente (professor ou responsável na empresa):

    - Telefone:
    - E-mail:
    - Login cadastrado no site *github.com* (se já tiver; vai precisar se quiser acompanhar o desenvolvimento do produto!).
    - 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 aluno no *github*:

    Nota: limite padrão de até 3 alunos por projeto.

    - Após enviar os dados pelo formulário comunicar o professor enviando email para bueno@lenep.uenf.br.

- O coordenador da disciplina de introdução ao projeto de engenharia analisa os dados e dá um retorno.

    – Toda proposta deve incluir o desenvolvimento de um software em C++ com uso de orientação a objetos. Pode 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 à engenharia.

- Uma 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.
- Caso o rascunho da proposta de desafio tecnológico seja rejeitado, o professor explica os motivos, possibilitando correções e reenvio da proposta[1].

  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 (individual) ou Projeto de Engenharia (individual ou grupo). Este documento trata apenas do projeto de engenharia. Se estiver interessado em desenvolver sua IC - Iniciação Científica no formato tradicional de monografia, veja o modelo https://github.com/ldsc/ModeloDiretorio-Aluno-Bolsista-TCC-IC-MSC-DSC-Projeto.

- Caso a proposta de desafio tecnológico seja aprovada, o coordenador da disciplina informa o cliente (e alunos caso já tenha).

[1] ∗ Note que esta metodologia pode ser aproveitada para desenvolver projetos de engenharia que não sejam de software, mas, neste caso, precisa encontrar professores para acompanhar as atividades e não será valido para a disciplina Projeto de Software Aplicado à Engenharia.

  - 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 desafio tecnológico (Título do projeto).
      · Conteúdo: nome, email e login no *github* dos membros da equipe.

- O coordenador da disciplina
  - Clona o modelo de projeto da disciplina, criando um projeto novo, com o título do projeto informado (armazenado no site https://github.com/ldsc).
  - 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 (veja LyX).

    - 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 do desafio tecnológico.
  - Gera o pdf associado.
  - Envia os arquivos gerados para o servidor do *github* (lyx e pdf).
    - Comandos:

```
cd diretórioComOProjeto
git add .
git commit -m "Gerado o desafio tecnológico: titulo"
git push
```

    - 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/ldsc*. 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 criada pelo Prof. Carlos Dias e 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 1.8). Ou seja, um projeto de engenharia desenvolvido nas disciplinas de programação poderá ser um TCC se atender as exigências, normalmente o projeto da disciplina é ampliado e melhorado.
  * Por exemplo, os projetos de engenharia dos alunos Guilherme, Nikolas, Nathan, posteriormente viraram o TCC deles. Adicionaram novas funcionalidades e interface gráfica em Qt.
- 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 coorientador do LENEP.
  * O projeto deve envolver o desenvolvimento de um software orientado a objeto em C++.
- Softwares utilizados:
  * LyX: Os documentos do projeto seguem um modelo pré estabelecido que usa o editor de texto LyX.

- O software editor LyX é apresentado aos alunos em sala de aula e temos manuais e vídeos disponibilizadas na internet e *google classroom*.
- O software 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).
- É um sistema eficiente e produtivo, pois os alunos recebem modelos prontos, se preocupando apenas com o conteúdo.

* Git/Github: 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*.
  - Sugere-se a equipe treinar os comandos básicos, testar seu uso, antes de executar no projeto em si.
* Umbrello: A parte de modelagem envolve o uso do modelador *umbrello*.
  - Com o umbrello os alunos conseguem fazer os diversos tipos de diagramas da UML (macro e micro, estruturais e dinâmicos).
* Compilador: A parte de desenvolvimento do software requer conhecimentos de C++, da biblioteca padrão de C++ e ferramentas como compiladores (gcc/g++, clang), editores específicos (emacs, vscode), IDEs (Qt-Creator).
* Uma lista com todos os softwares é apresentada na seção 1.7.2.

- Modelagem física-matemática, modelagem química-matemática, modelagem numérica-computacional, modelagem de engenharia
  - Os membros da equipe devem compreender os conceitos de matemática, física, química, algoritmos 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. Inclui a revisão bibliográfica.
- 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.
    - Fujam de temas que parecem fáceis mas não são interessantes (escolha algo que gosta, que 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 (mas a defesa final individual só ocorre quando matriculado oficialmente na disciplina).
    - O aluno também poderá mudar de equipe e tema, mas isso deve ser evitado pois usualmente gera atrasos.

        - Um bom projeto de engenharia é uma porta de entrada para vagas de emprego em diversas empresas de engenharia e de desenvolvimento de software.

- 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).

Nota: é possível dar continuidade a um projeto antigo, já desenvolvido, neste caso consulte a página https://github.com/ldsc/LDSC-ProjetoEngenharia-0-Metodologia-Instrucoes-Etapas e o site https://github.com/ldsc para ver lista de projetos existentes.

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 1.5.

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

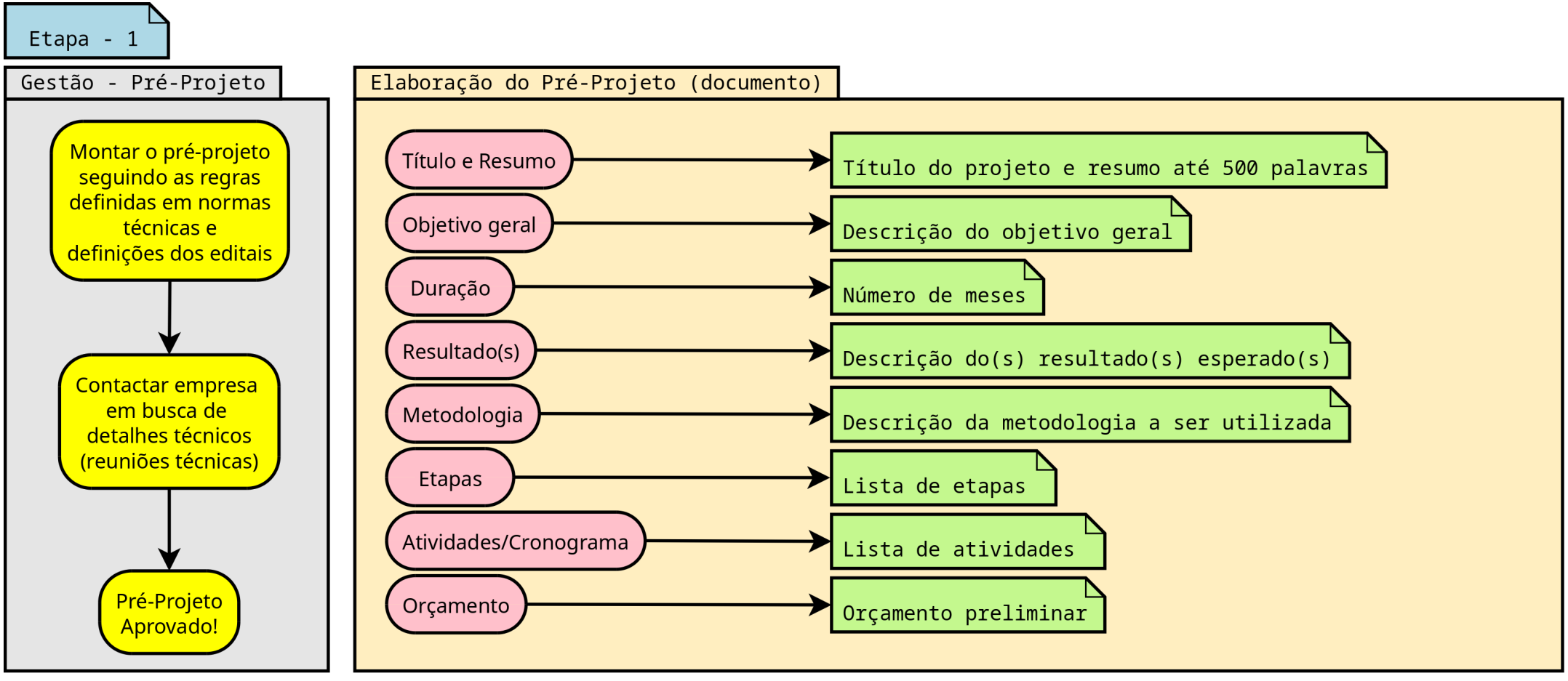

Figura 1.5: Etapa 1 - Elaboração do Pré-projeto

- Comunicação
  - Após aprovação do desafio tecnológico montamos um grupo do telegram que é usado para troca de informações exclusivas do projeto.
- A equipe elabora o pré-projeto.
- Os alunos encaminham o pré-projeto.
  - Os encaminhamentos são sempre pelo *github* seguidos de comunicado pelo telegram (e/ou email).
- 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 pré-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/executivo[2] e do contrato.

[2]

    * Notem que esta sequência é a sequência adotada pela Petrobras no seu sistema de desafios. Somente depois de aprovado o pré-projeto é que temos acesso ao sigitec para montagem do projeto com definição de equipamentos, pessoal, orçamentos e cronogramas.

Notem que temos alguns requisitos que devem ser aprimorados nas disciplinas e ao longo do desenvolvimento do projeto para efetivo aprendizado das competências e habilidades associadas ao desenvolvimento de projetos de engenharia.

- 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.
  - Matenha *backups* dos arquivos.
  - Importante executar os comandos com calma, prestando atenção, lendo as mensagens apresentadas.

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

- Uso do *github/Projects* ou *trello* para gestão das atividades:
  - Conceitos de administração/gestão usando uma metodologia *kanbam*.

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

Aprovado o pré-projeto podemos escrever o projeto em si, o mesmo inclui mais detalhes das instituições envolvidas, objetivos específicos, justificativa e resultados esperados, além de cronograma e orçamento.

A sequência é detalhada na Figura 1.6.

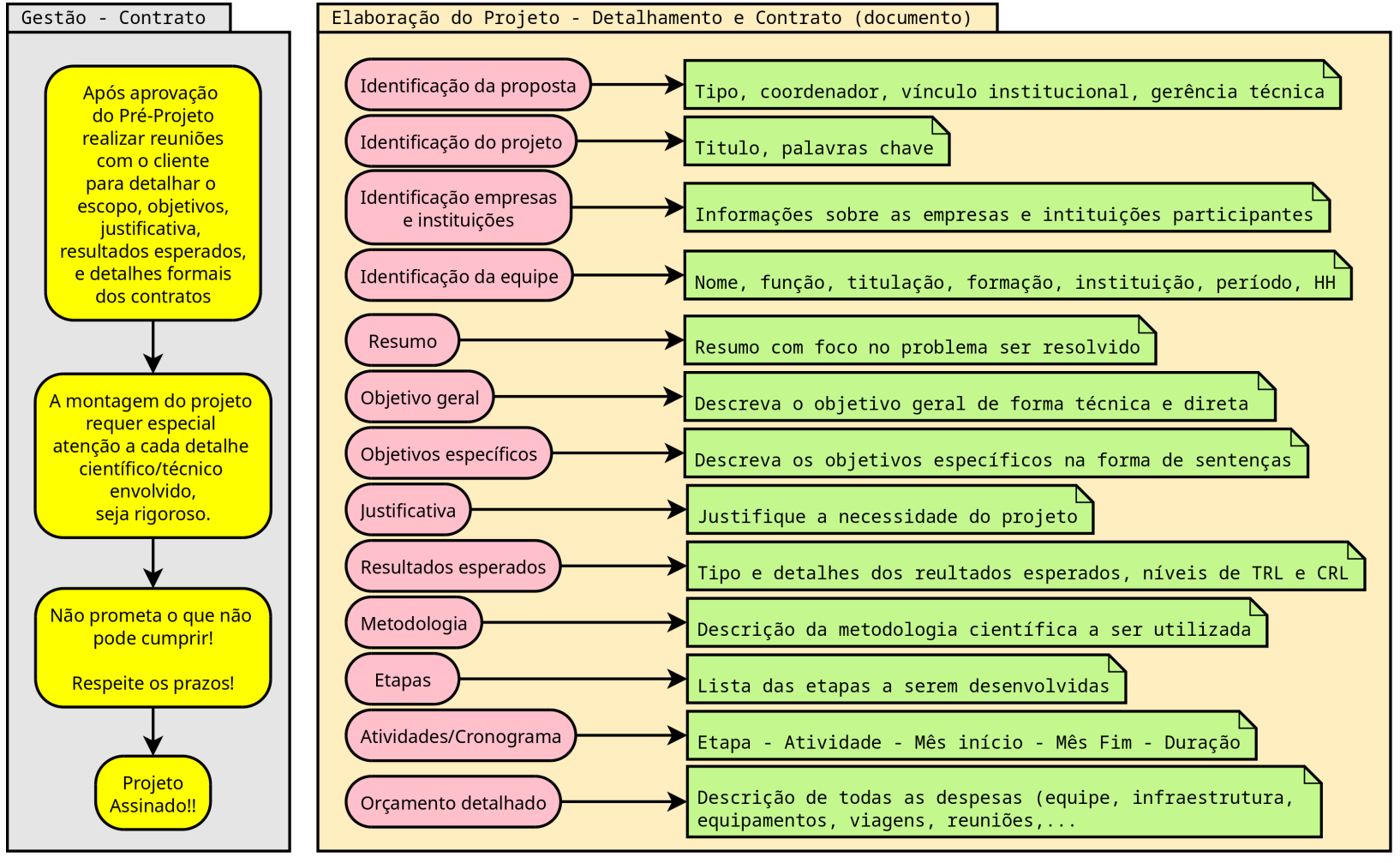

Figura 1.6: Etapa 2 - Elaboração do contrato - detalhamento do pré-projeto

- 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[3].
  - Toda parte conceitual precisa ser melhor detalhada, não pode haver dúvida sobre o que de fato vai ser feito.
  - Havendo possibilidade a empresa interessada poderá pagar bolsas aos alunos durante o período de execução do projeto (modelo de empresa júnior).

[3] ∗ 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 *Introdução ao Projeto de Engenharia*, os alunos aprendem os conceitos associados e colocam em prática fazendo modelos e diagramas. 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*. Para maiores detalhas consulte o site da organização internacional que a regulamenta a UML - https://www.uml.org/.

A sequência desta etapa é detalhada na Figura 1.7.

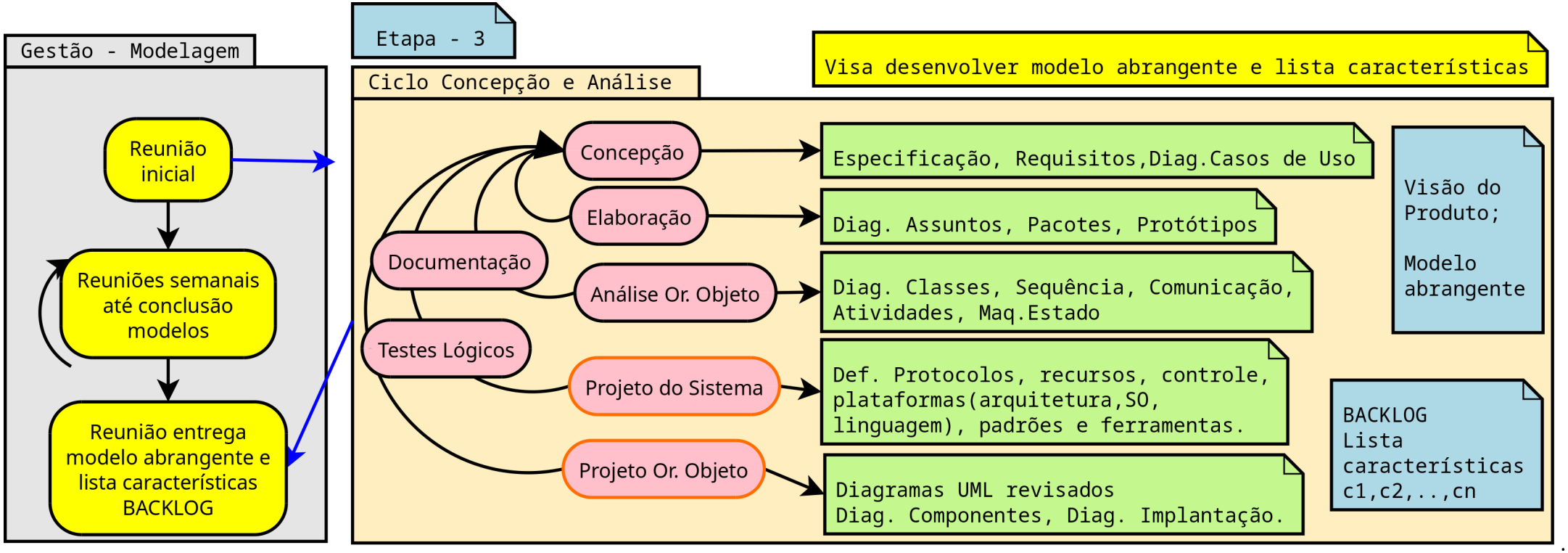

Figura 1.7: Etapa 3 - Modelagem de Engenharia

- Nesta etapa os alunos devem desenvolver um modelo abrangente do sistema de software a ser desenvolvido. Passa pelas etapas de:
  - **Concepção**: especificação, requisitos e diagramas de caso de uso.
  - **Elaboração**: estudo do problema em livros de engenharia. Desenvolvimento de revisão bibliográfica, equações e modelos.
  - **Análise Orientada a Objeto**: desenvolvimento dos modelos (diagramas UML).
  - **Projeto do Sistema**: definição de arquiteturas, protocolos, recursos, controle e plataformas, padrões e ferramentas.
  - **Projeto Orientado a Objeto**: revisão dos modelos considerando as decisões do projeto do sistema.
  - **Realização de Testes Lógicos**: testar logicamente os modelos desenvolvidos.
  - **Documentação**: geração dos documentos de modelagem e diagramas associados.
  - **Geração da Lista de Características ou *features* (BACKLOG)**: lista das funcionalidades 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 aos alunos facilita este trabalho, pois além de já estar formatado contém dicas e sugestões práticas.
  - Temos mais de 50 projetos disponibilizadas no site do https://github.com/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 1.8.
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::NomeMétodo).
  - Havendo necessidade detalhe o comportamento dos objetos usando diagramas de máquina de estado (nome classe).
- Ciclo de construção:
  - Use o modelador UML para 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.
  - Use uma IDE ou editor de texto para implementar os códigos (ex: *vscode*, *emacs*).
  - Use um compilador.
  - Uma lista completa dos softwares utilizados é encontrada na seção 1.7.2.

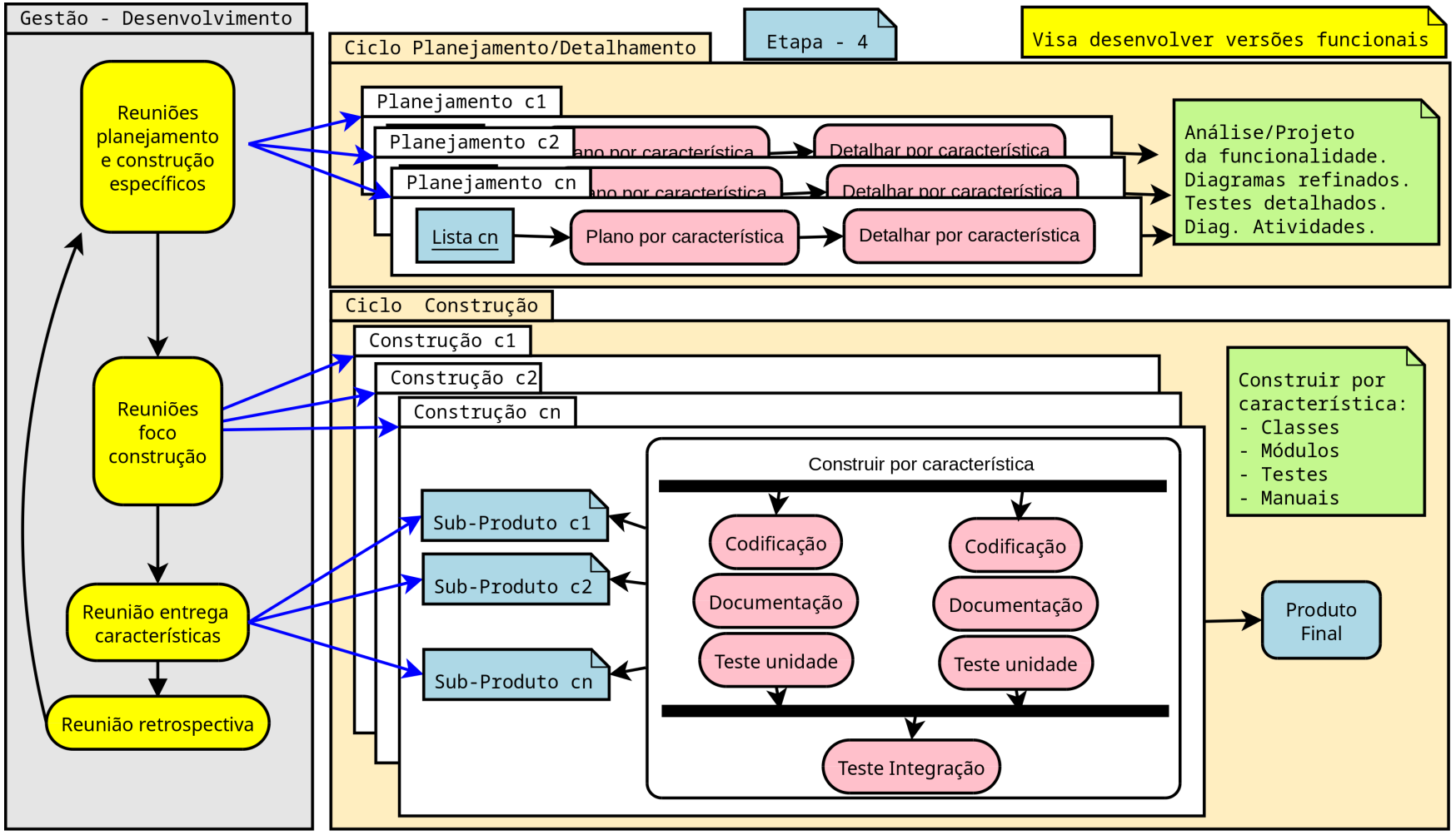

Figura 1.8: Etapa 4 - Ciclos de planejamento, detalhamento e construção

- 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 (diagramas de comunicação, máquina de estado e atividades) são revistos.

  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 diagrama é desnecessário. Faça diagramas para as partes mais complexas e que requeiram algoritmos mais exigentes. Na documentação você pode acrescentar link para sites que incluem informações detalhadas sobre determinado ponto do projeto.

- A construção ou implementação dos códigos deve ser feita seguindo-se as normas e procedimentos padrões usando as melhores práticas:

  – Na disciplina damos preferência para uso de editores modernos como emacs, vscode, com plugins e funcionalidades adequadas

instaladas. Escolha um editor, a seguir leia os manuais e tutoriais de uso para um melhor rendimento.

  - Existem centenas de 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.
    * Visa permitir que outras equipes deem continuidade ao projeto usando as ferramentas ensinadas.
  - 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 que é desenvolvido de forma incremental por diferentes equipes.
    * Uma versão dos diagramas no *umbrello* é necessária para permitir a continuidade do projeto por outras equipes.
    * O umbrello é software livre e existe a muitos anos.

- Após implementar os códigos é preciso realizar os testes.
  - Implemente os testes de unidade (teste das funções e classes desenvolvidas). Veja este vídeo.
    * Ferramentas usadas para teste:
    * Catch2 (arquivo único, simples e expressiva, boa para testes de unidade e comportamente (BDD))
    * Google Test (excelente para iniciantes)

    - Boost.Test (mais pesada e completa, requer experiência)
    - CTest (vinculada ao cmake, é indicada para projetos que já usam o cmake, realizada testes de integração e testes de sistema, não é muito efetiva para testes de unidade; é mais complexa)
  - Implemente os testes de integração (módulos e componentes do sistema, ajuda).
  - A cada ciclo de desenvolvimento as *features* desenvolvidas devem ser testadas.
  - Não deixe para fazer os testes do sistema só no final.

- No final de cada ciclo de planejamento/detalhamento/construção a equipe deve realizar uma reunião física para discutir o andamento do projeto e realizar a entrega da versão (subproduto).
  - Conversar sobre o que foi feito, o que falta fazer e correções de rumo (ajustes na metodologia de trabalho).
  - Fazer uma apresentação informal da evolução do sistema.
  - Esta reunião deve seguir os procedimentos da metodologia SCRUM, o cliente deve participar.
  - Idealmente o cliente deve realizar testes com o software desenvolvido e dar um retorno/*feedback*.

A etapa 4 é repetida várias vezes, gerando as diferentes versões do sistema:

- No nosso caso, como é um sistema desenvolvido com fins didáticos, criamos diretórios para cada versão desenvolvida:

- 4-Execucao-Detalhamento-Construcao-Versao-1
- 4-Execucao-Detalhamento-Construcao-Versao-2
- 4-Execucao-Detalhamento-Construcao-Versao-3
- 4-Execucao-Detalhamento-Construcao-Versao-...

1.7 Etapa 5 - Entrega do produto

A sequência é detalhada na Figura 1.9.

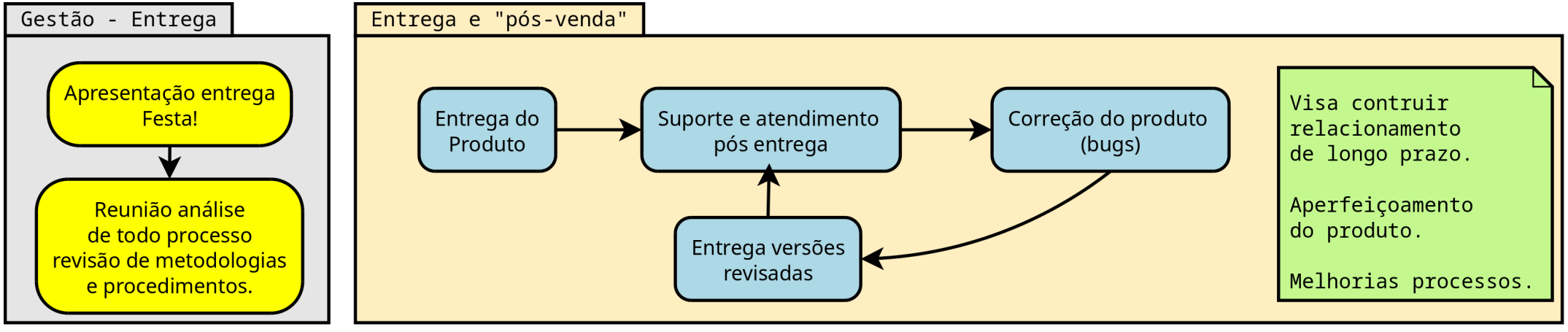

Figura 1.9: Etapa 5 - Entrega do produto

- A versão final do sistema de software deve passar por um sistema de testes pormenorizados.
  - Testar se todas as especificações foram atendidas.
  - Testar se todas 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.
    * Outras pessoas devem ter acesso ao projeto, realizar a leitura, instalação e testes do sistema.
    * O ideal é que este teste de instalação e uso seja feito por 3 pessoas diferentes em sistemas diferentes (ex: Windows, GNU/Linux, Mac OS X).
    * 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 defenderam.
  - Depois adicionaram novas funcionalidades e uma interface gráfica amigável em Qt e defenderam como TCC. Ou seja, é necessário ampliar o escopo e atingir um nível técnico adequado a um TCC.

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)
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 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.
- Note que temos duas defesas, a primeira esta associada a disciplina LEP - 01446: Projeto de Software Aplicado à Engenharia e a segunda é a defesa do TCC um ano depois.

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 - TCC									X	X		
Etapa 4.c6 - TCC										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 a 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.
  - O aluno recebe modelos prontos e funcionais, com layout definido, basta preencher o conteúdo.
    - O aluno se preocupará apenas com o que interessa, o problema de engenharia.
- 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 e casa dos alunos.
  - Não fornecemos nenhum suporte para estes softwares.
  - As versões finais devem ter documentos gerados em formatos compatíveis com softwares livres, garantindo a viabilidade da 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. Dê preferencia para imagens no formato png.

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. 7

[autores, 2017] autores, V. (2017). *Projetos de engenharia - uma introdução*. LTC. ISBN-13 : 978-8521634454. 7

[Blaha and Rumbaugh, 2006] Blaha, M. and Rumbaugh, J. (2006). *Modelagem e Projetos Baseados em Objetos com UML 2*. Campus, Rio de Janeiro. 7, 47

[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). 7

[de Moura Menezes, 2018] de Moura Menezes, L. C. (2018). *Gestão de Projetos*. Atlas. 7

[Fowler and Scott, 2005] Fowler, M. and Scott, K. (2005). *UML Essencial*. Bookman, São Paulo, 3 edition. 7

[Heldman, 2005] Heldman, K. (2005). *Gerência de projetos*. Elsevier. ISBN 13 : 978-8535216844, Rio de Janeiro. 7

[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. 7

[Pahl, 2005] Pahl, G. (2005). *Projeto na Engenharia: Fundamentos do Desenvolvimento Eficaz de Produtos - Métodos e Aplicações*. Blucher. ISBN-13: 978-8521203636. 7

[Pires, 2012] Pires, A. M. S. (2012). *Projeto de Instalações Elétricas e Telecomunicações*. Instituto Superior de Engenharia de Coimbra. 7

[Rosa, 2007] Rosa, M. O. (2007). *Gerenciamento de projetos de governo*. PMI-DF - PMInforma. 7

[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. 7, 47

[Sonerviile, 1993] Sonerviile (1993). *Engenharia de Software*. MacGraw-Hill, São Paulo. 7

[Valeriano, 2015] Valeriano, D. (2015). *Moderno Gerênciamento de Projetos*. Pearson. 7

[Woiler, 1996] Woiler, S. (1996). *Projetos: planejamento, elaboração, análise*. Atlas. 7