Andreis Gustavo Malta Purim - 213095
Gustavo Purcena de Lima - 198594

Enunciado:

*Nessa avaliação, a **dupla** (ou trio) de alunos deverá descrever um processo, considerando as **abordagens** BizDev e DevOps para o projeto escolhido. Para isso, considere:*

1. *Adotar um modelo prescritivo ou processo existente como **base***
2. *Elencar as **atividades fundamentais** e **guarda-chuva** a serem desempenhadas*
3. *Principais **artefatos** a serem produzidos*
4. ***Papéis** dos envolvidos no processo*

Respostas:

1. Modelo Prescritivo *1*
2. Diagrama Atividades Fundamentais e Guarda-Chuva *3*
3. Artefatos *4*
4. Papéis *4*

1. Modelo Prescritivo

O método prescritivo de Engenharia de Software foi o *método incremental*. As razões para escolhê-lo são:

1. **Começa por identificação do núcleo do produto (requisitos básicos):**

   O nosso sistema visa implementar uma plataforma de formatação de prontuários médicos interoperáveis, mas não sabemos exatamente quais outras *features* podem ser necessárias devido à natureza da área médica (apenas descobriremos conforme formos capazes de validar com possíveis usuários). Além disso, o núcleo base do produto já foi descrito no artigo do ICIPEMIR, basta traduzi-lo para os requisitos básicos de software, e todas as outras etapas serão de melhorias em cima desse núcleo.

2. **Fluxo repetido até que o produto completo seja entregue:**

   Uma vantagem desta implementação é que sempre poderemos testar a base do sistema por completo (a geração dos prontuários), adicionando novos elementos que se apoiam para melhorá-lo. Logo, a ideia de incrementar sequencialmente é natural.

3. **Entregas em incrementos Sequenciais e lineares:**

   Relacionado com a característica anterior. Uma vez que a matéria de MC426 também é sequencial cronologicamente, irá apresentar os diferentes aspectos de Engenharia de Software que aos poucos poderemos integrar no sistema. A figura 1 apresenta de forma abstrata como o projeto será construído com cada feature sendo adicionado.

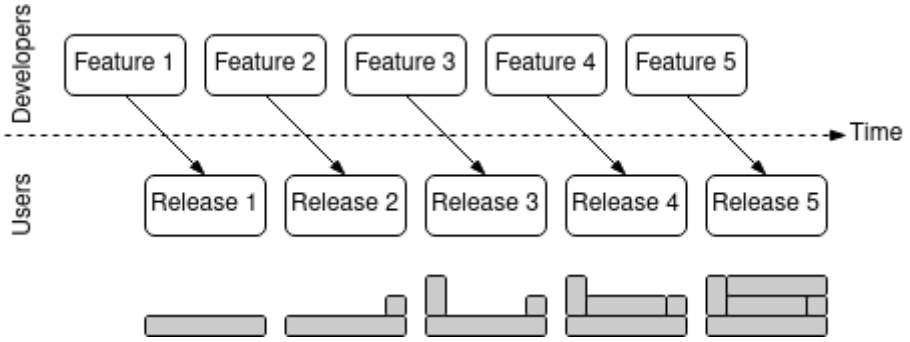


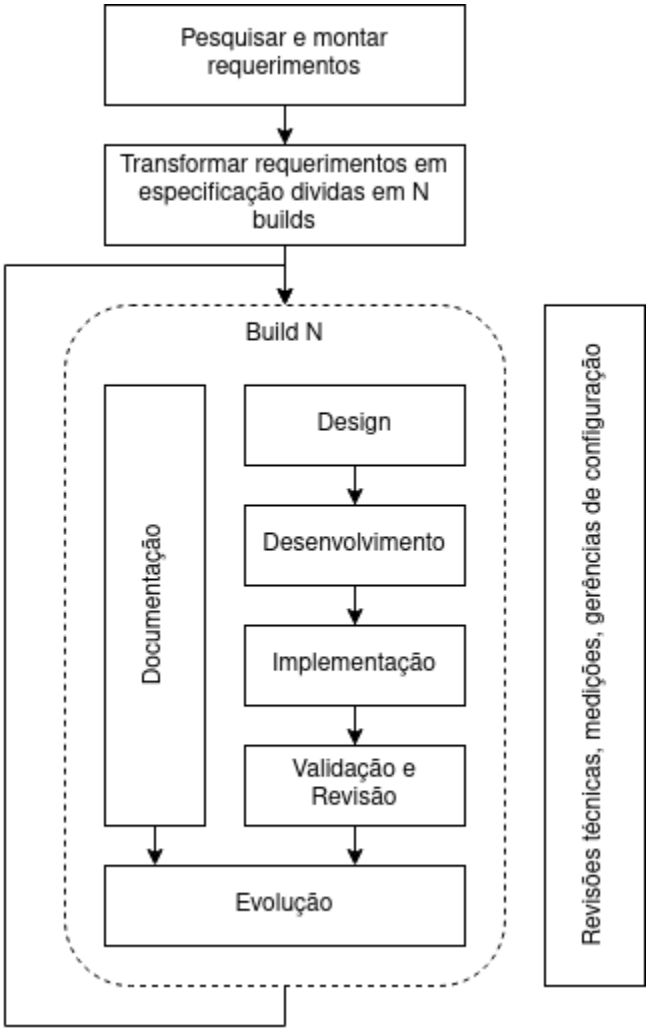
Figura 1. Método incremental pensando nas 5 features de Eng. de Software.

A tabela 1 apresenta as vantagens e desvantagens consideradas para adotar este modelo:

Tabela 1. Vantagens e desvantagens do modelo

Vantagem	**Comentário**
Incrementos podem ser implementados com menos pessoas	Como nossa equipe é de apenas 4 pessoas, precisamos focar em mudanças fáceis de gerenciar
Gerenciar melhor as incertezas (que o cascata) / Entrega e implantação mais rápidas de software útil ao cliente/usuário	As nossas incertezas incluem outras matérias e preocupações no semestre, que podem mudar nossa expectativa de terminar features a tempo. O método incremental assegura que ao menos sempre teremos uma base funcional.
Menor custo e maior facilidade de realizar mudanças em tempo de desenvolvimento	Relacionado ao ponto acima. Precisamos ter incrementos que sejam cabíveis numa escala de tempo do semestre.
Mais fácil de obter feedback do cliente/usuáiio	Outro ponto importante, uma vez que estamos lidando com uma área difícil (biomedicina) e de conhecimento especializado. Logo, sempre precisaremos verificar com possíveis usuários antes de seguir na implementação.
Desvantagem	**Comentário**
Dependendo do tamanho do incremento, espera por versões e retrabalho podem ocorrer	Retrabalho e demoras (“visão de tunel”) são um dos principais perigos no desenvolvimento do projeto. Isso deve ser mitigado com uma divisão bem planejada das features e especificações.
Estrutura do sistema tende a degradar com a adição de incrementos	Para reduzir o *bloat* do código e degradação do sistema, é necessário assegurar um código limpo, modular e bem polido antes de adicionar novas features. Isso foi levado em conta na criação das atividades fundamentais.

2. Diagrama Atividades Fundamentais e Guarda-Chuva



Como o nosso processo é incremental, nosso diagrama foi dividido nas suas N builds apresentados na figura 2 e 3, que repetirão uma certa quantidade de atividades fundamentais. De forma geral, o processo como um todo começa com o **levantamento de requisitos** e o **planejamento das builds incrementais**.

Dentro das builds, as atividades serão de **Design**, que irá transformar as especificações em pontos tangíveis. o **desenvolvimento**, que irá ser a tarefa de programar as features, **implementação**, que inclui testar e integrar no sistema. **Validação e revisão**, cujo o software é verificado para garantir que satisfaz os requisitos (*Quality Control)*, é uma atividade-guarda chuva de cada Build é a **Documentação** para a criação de artefatos. Por fim, a **Evolução** faz ajustes para a próxima build, se necessário.

Além disso, uma atividade guarda-chuva para todas as builds é a de Quality Assurance, revisões técnicas, medições e gerência de configuração.

Figura 2. Diagrama das atividades fundamentais e guarda-chuva

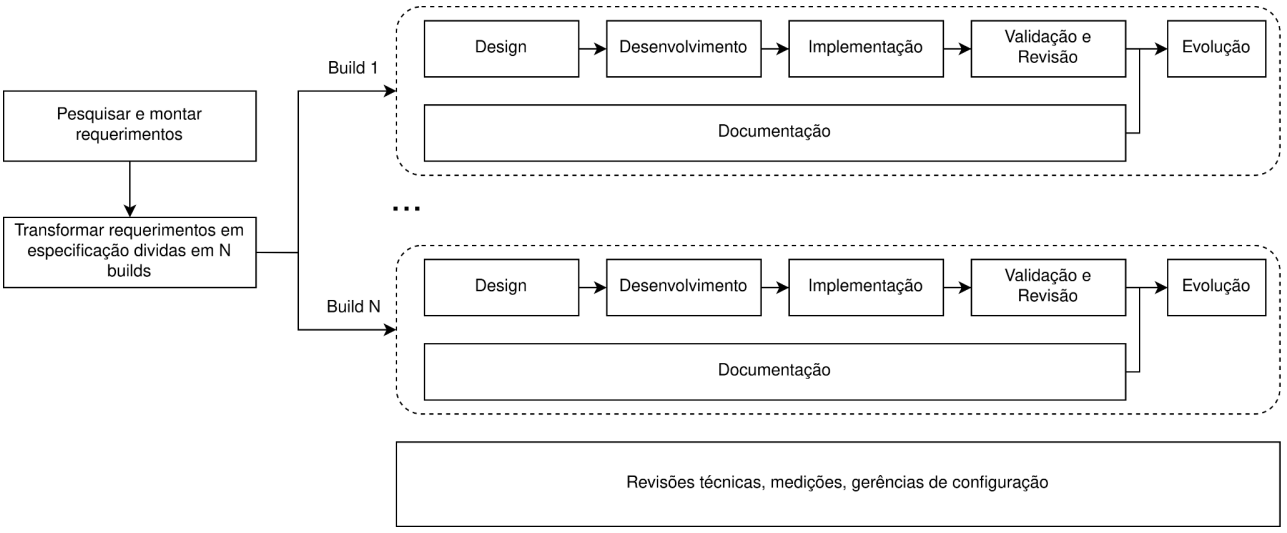


Figura 3. Visão “clássica” do modo incremental.

3. Artefatos

Artefatos são os tipos de subprodutos tangíveis criados durante o desenvolvimento de software.No caso, alguns artefatos podem ser diagramas, modelos, documentação, documentos de design que ajudam a descrever a função e arquitetura do software. Os artefatos que iremos produzir serão divididos em diferentes tipos:

Artefatos dos requerimentos e guarda-chuva: Artefatos que serão criados pela atividade guarda-chuva em comum a todas as builds, por exemplo, o diagrama de Gantt do projeto, diagramas de arquitetura, a Matriz RACI para diferentes tasks, documentos sobre o uso do git, etc…

Artefatos da Build Incremental: Artefatos que são criados durante a build incremental, backlogs do produto (criados durante a atividade de design), quadro de tasks, diagrama de usos e funções do software, documentação das features, documento de validação, etc…

4. Papéis

Como a equipe do projeto é de apenas 4 pessoas, pensamos no menor número de papéis necessários para desenvolver o software de forma eficaz.

Papel	**Descrição**	**Responsável**
Product Owner	Possui conhecimento do projeto e do usuário, entende a perspectiva e as necessidades do cliente e guia o time para a visão e os requisitos em um produto/serviço final.	Andreis Purim
Developer	Responsável por implementar o projeto, programar, adicionar e depurar as funcionalidades	Todos
Quality Control Engineer	Verifica se uma solução desenvolvida atende à especificação exigida, focando na qualidade e projetando documentos para fornecer feedback útil e oportuno	Guilherme Ramirez
User Experience Designer	Responsável por desenhar e desenvolver a interação do usuário com o produto. Atento aos detalhes e possíveis problemas da implementação	Gustavo Purcena
Software Architect	Faz escolhas de design de alto nível com base em requisitos não funcionais e dita padrões de codificação junto com ferramentas e plataformas. Também é responsável por revisar o código, garantindo a qualidade do design, evitando muita complexidade e focando na clareza.	Victor Scholze