Universidade Federal do Paraná
Departamento de Informática

Verificação e Validação

Sistema da Ecomp

Lucas Emanuel de Oliveira Santos
Rafael Urbanek Laurentino
Thiago Fabrício de Mello
Vitor Lorenzo Cumim

CI1163 — Design de Software
Prof. Andrey Pimentel

Curitiba, 2025

1 Introdução

Este relatório apresenta a etapa de Verificação e Validação (V&V) do projeto desenvolvido na disciplina de Design de Software. Seu objetivo é analisar de forma sistemática a consistência entre os artefatos de modelagem produzidos (DSS, Contratos de Operação, Diagramas de Interação e Diagrama de Classes) e a implementação realizada em Java.

2 Verificação e Validação

2.1 ECU001 — Cadastrar Projeto

- **Conformidade com o Diagrama de Interação:** A classe `Ecomp` segue corretamente o fluxo modelado: recebe os dados, instancia um `Projeto` e o adiciona à lista interna.
- **Divergência no DSS (Fluxos Alternativos):** O DSS prevê validação de campos e mensagens de erro via bloco `alt`, mas a implementação não realiza verificações e assume sempre o fluxo de sucesso.
- **Tipagem de Dados:** `dataInicio` e `prazo` são tratados como `String`, o que inviabiliza validações temporais previstas pelo domínio.

2.2 ECU002 — Cadastrar Etapa

- **Conformidade Estrutural:** A controladora `Ecomp` localiza o projeto e delega a criação da etapa à classe `Projeto`, conforme o modelo.
- **Tratamento de Referências Nulas:** O método verifica se o projeto existe antes de prosseguir, embora sem fornecer feedback ao usuário.
- **Nomenclatura:** Há divergência entre o nome projetado (`cadastrarEtapa`) e o nome implementado (`cadastrarEtapas`).
- **Status:** O atributo `status` permanece como `String`, apesar do domínio sugerir o uso de `enum`.

2.3 ECU003 — Alocar Desenvolvedor em Projeto

- **Conformidade Lógica:** O método implementado em `Projeto` evita duplicidade de desenvolvedores, conforme o Diagrama de Interação.
- **Divergência de Assinatura:** O DSS define chamadas por ID; a implementação exige o objeto `Projeto` diretamente.
- **Observação sobre Herança:** A modelagem trata desenvolvedores como papéis, enquanto o código usa herança (`Desenvolvedor extends Ecomper`).

2.4 ECU004 — Cadastrar Atividade

- **Divergência de Fluxo:** O método implementado apenas cria e retorna a atividade, deixando a persistência dependente de outra chamada.
- **Divergência com o DSS (Validação):** Não há verificação de campos obrigatórios.
- **Dados do Domínio:** A classe `Atividade` não possui atributos temporais estruturados, apenas textos.

2.5 ECU005 — Cadastrar Item Fiscal

- **Conformidade de Delegação:** O encaminhamento para a classe `Projeto` segue a modelagem.
- **Divergência de Assinatura:** Assim como em ECU003, a implementação opera com objetos em vez de IDs.
- **Tipagem e Validação:** O atributo `valor` é `float` e não possui validação contra valores inválidos.

2.6 ECU006 — Cadastrar Membro (Ecomper)

- **Conformidade com a Sequência:** O fluxo de criação e adição está alinhado ao Diagrama de Interação.
- **Divergência com o DSS (Regras de Negócio):** Não há validação de unicidade (como CPF) ou de formato dos dados.

2.7 ECU007 — Gerar Relatórios

- **Conformidade com o Diagrama de Integração:** A classe `Relatorio` segue o modelo previsto, e a controladora `Ecomp` delega o processamento corretamente.
- **Lógica de Agregação:** O método `gerarRelatorioItensFiscais` replica os laços aninhados do diagrama, consolidando itens fiscais dos projetos.
- **Fluxos Alternativos previstos no DSS:** Solicitações de "Tipo inválido" geram relatórios vazios, sem mensagens de erro, diferindo da modelagem.