Engenharia de Software

Projeto do Sistema

AULA 04

PROFª OTILIA

Revisão

Aula 01

- Processo de Fabricação do hardware x Processo de Desenvolvimento de Software – Engenharia de Software
  - Imagine que você tem um terreno e deseja construir uma casa. Qual o primeiro passo? Você iniciará pelas paredes, pela fundação ou pelo projeto? E quando você tem um problema e deseja construir um software para solucioná-lo, por onde você deve começar?

Revisão

Aula 02

- Desenvolvimento de software como um projeto e as Atividades de gerenciamento: planejamento e gerenciamento de riscos
  - Se você deseja aprender um novo idioma e de vez em quando assiste algumas aulas na internet, ou faz algumas leituras esporádicas, isso pode ser considerado um projeto pessoal? Suponha agora que você tenha planejado estudar duas horas por dia, três vezes na semana durante 18 meses, e após esse período irá fazer uma viagem de 30 dias para exercitar a conversação. Isso pode ser considerado um projeto pessoal?

Revisão

Aula 03

- Engenharia de Requisitos e Análise de Sistemas
  - Quando um cliente deseja um software, como nós sabemos quanto tempo demorará para ser desenvolvido? Como nós descrevemos para os programadores todas as características que devem ser implementadas? De que forma são listadas todas as funcionalidades que o software deve possuir? Resposta: Através dos requisitos!

Objetivos

- Conhecer os conceitos, elementos e procedimentos centrais da atividade de análise e projeto de sistemas.

Situação problema

- Quando um cliente explica o problema que o software deve solucionar, o que o analista de TI deve se preocupar em anotar:
  - O que sistema irá fazer?
  - Ou como o sistema irá fazer?

Resposta: o que o sistema irá fazer

Projeto de Software

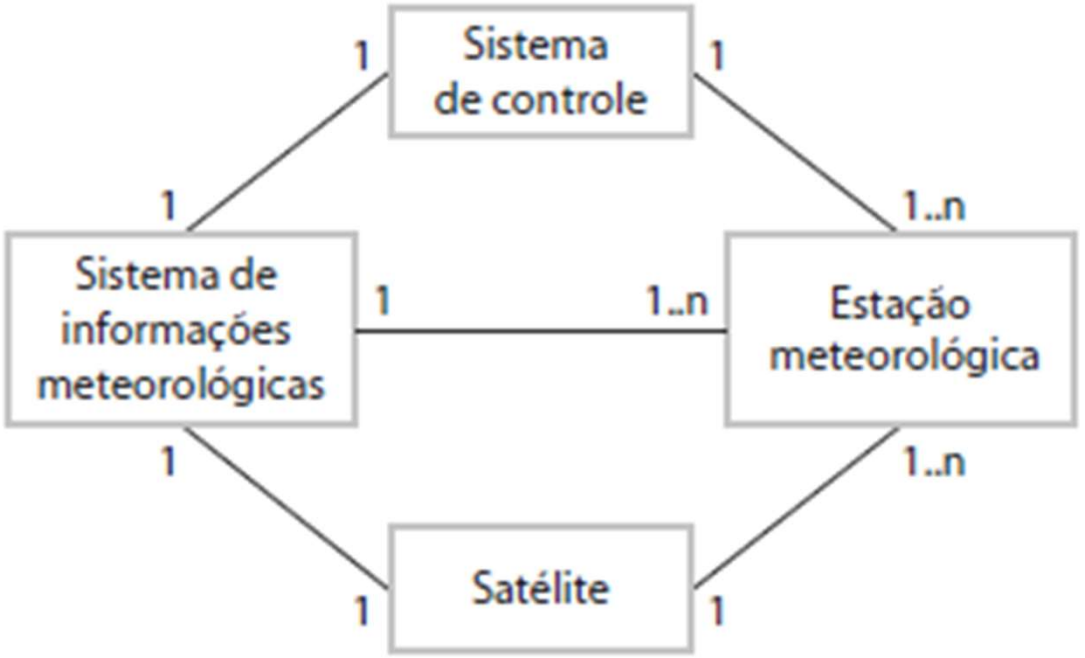
- Identificação dos componentes de software e seus relacionamentos com base nos requisitos do cliente
  - Atividade criativa
  - Intimamente ligada à implementação do software

Projeto de software

- Projeto alto nível (conceito) - > Projeto detalhado
  1. **Contexto e interações externas com o sistema**
  2. **Arquitetura do sistema**
  3. **Principais objetos do sistema**
  4. **Modelos de projeto**
  5. **Interfaces**

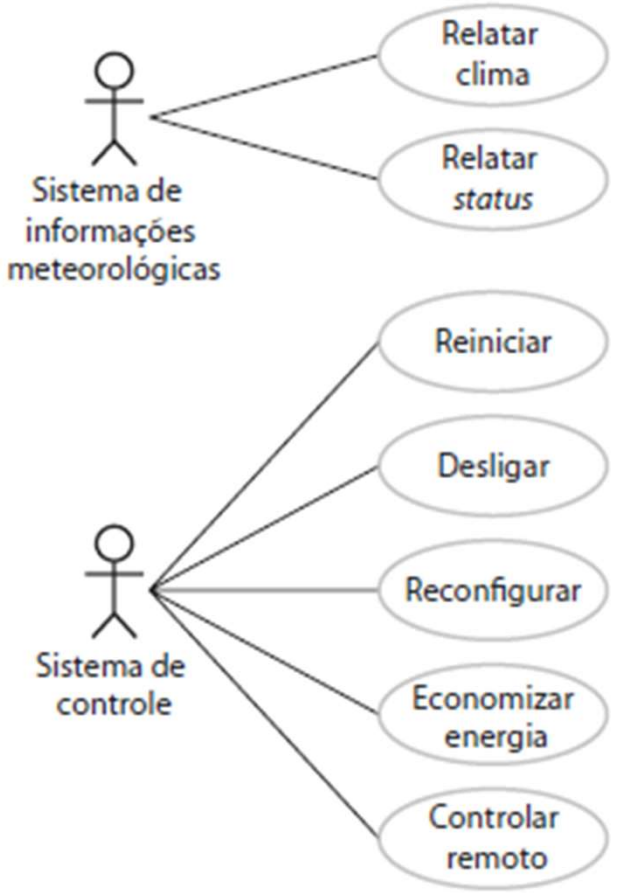
Contexto e interações do sistema

Figura 7.1 Sistema de contexto para estação meteorológica



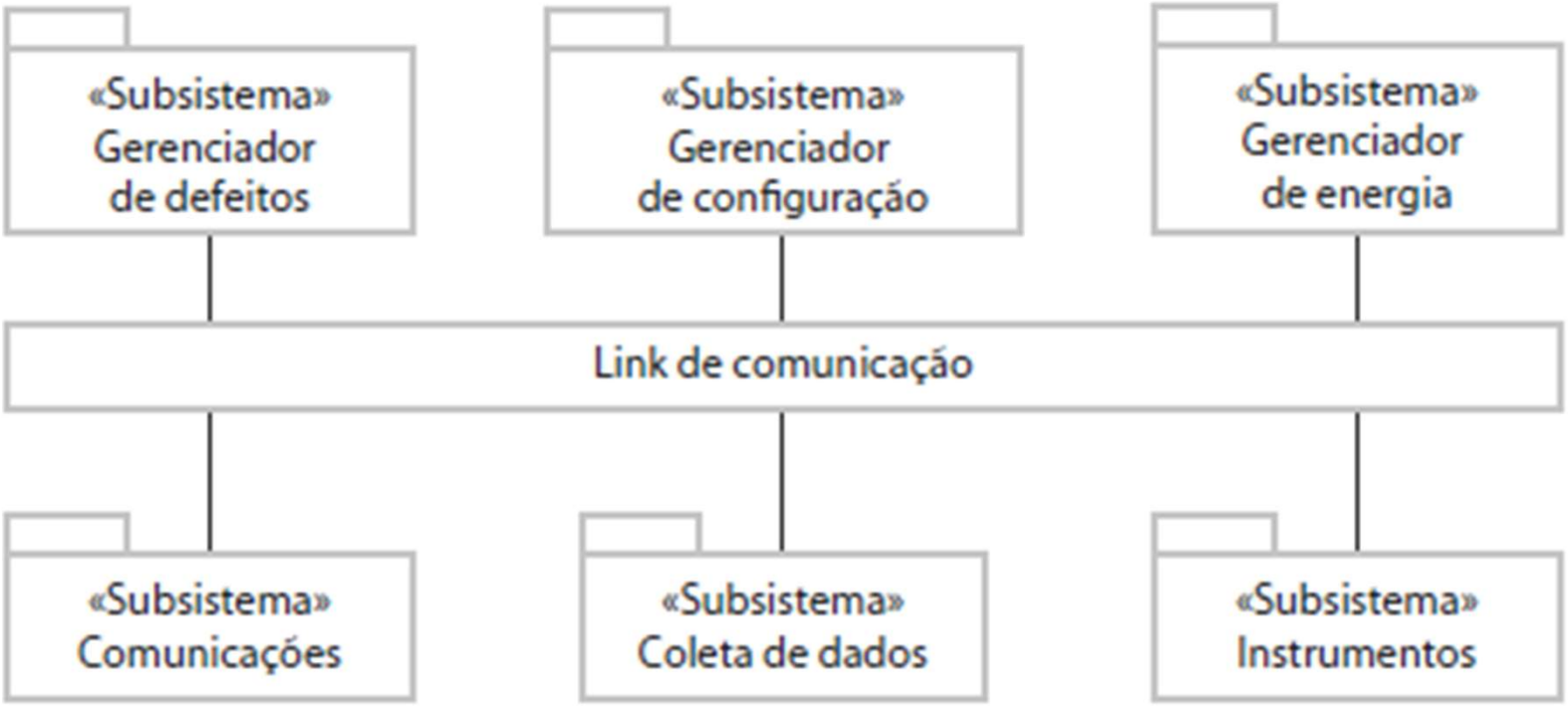
Contexto e interações do sistema

Figura 7.2 Casos de uso da estação meteorológica



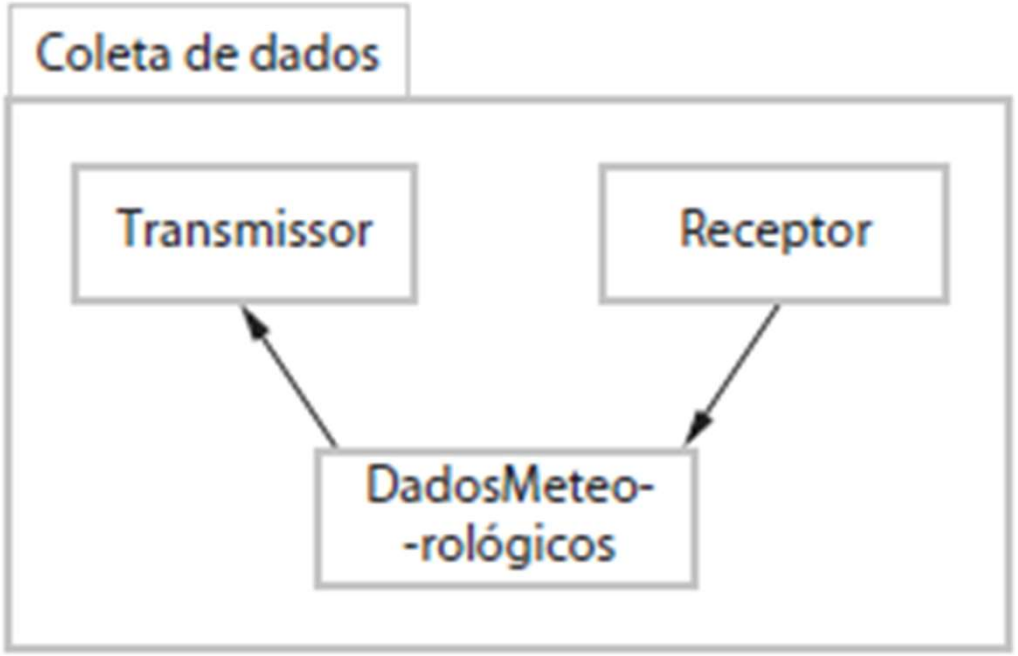
Projeto da arquitetura

Figura 7.3 Arquitetura de alto nível da estação meteorológica



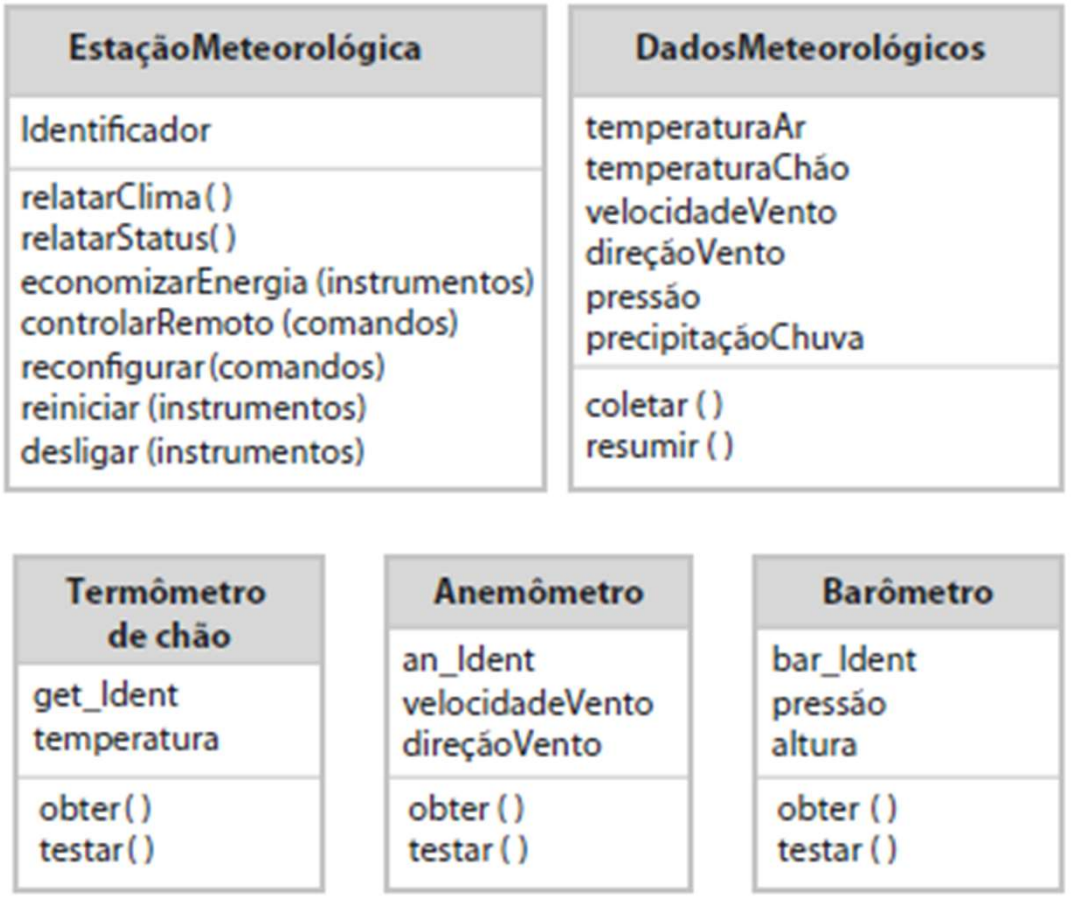
Projeto da arquitetura

Figura 7.4 Arquitetura do sistema de coleta de dados



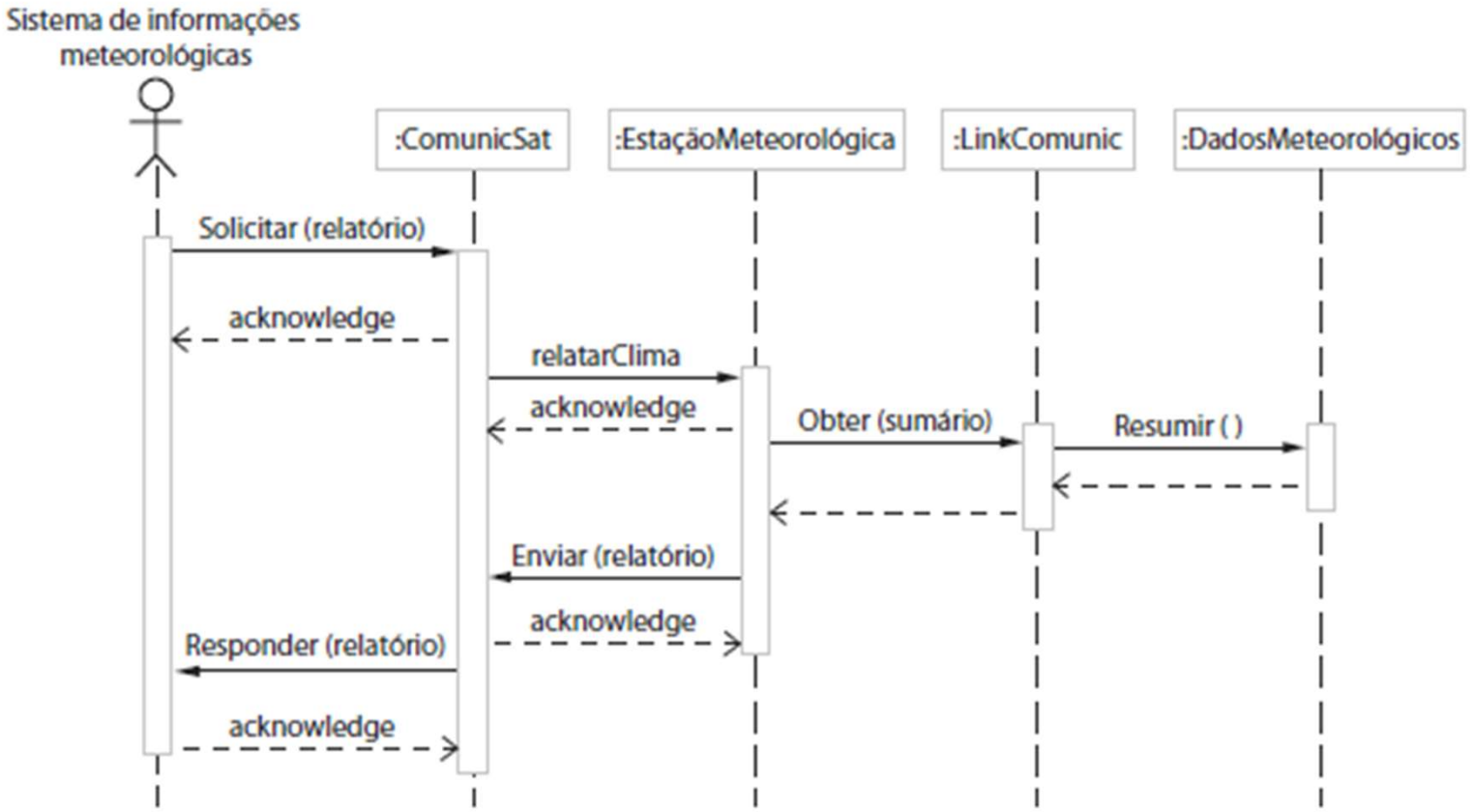
Objetos essenciais

Figura 7.5 Objetos da estação meteorológica



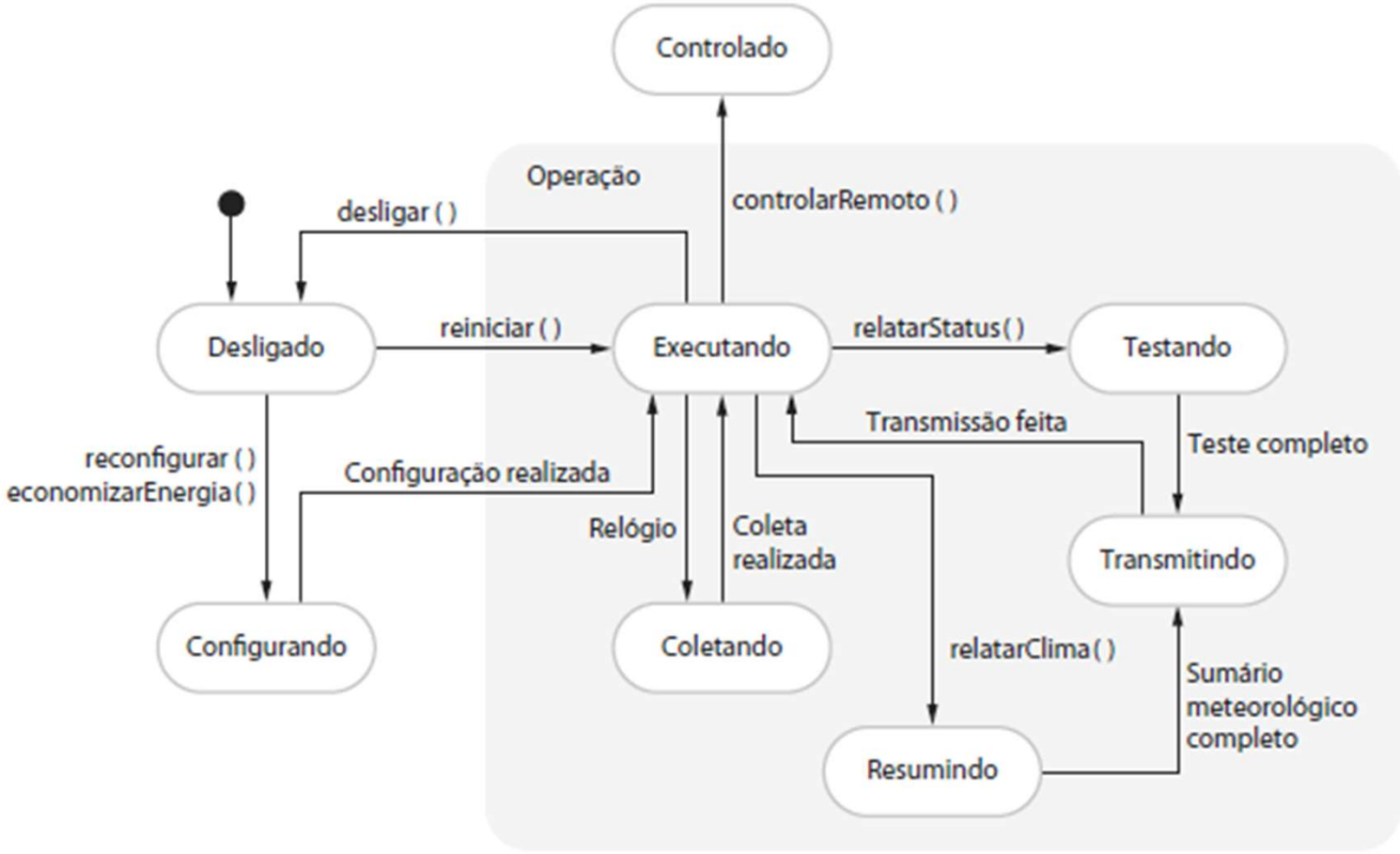
Exemplo de modelo do sistema

Figura 7.6 Diagramas de sequência descrevendo coleta de dados



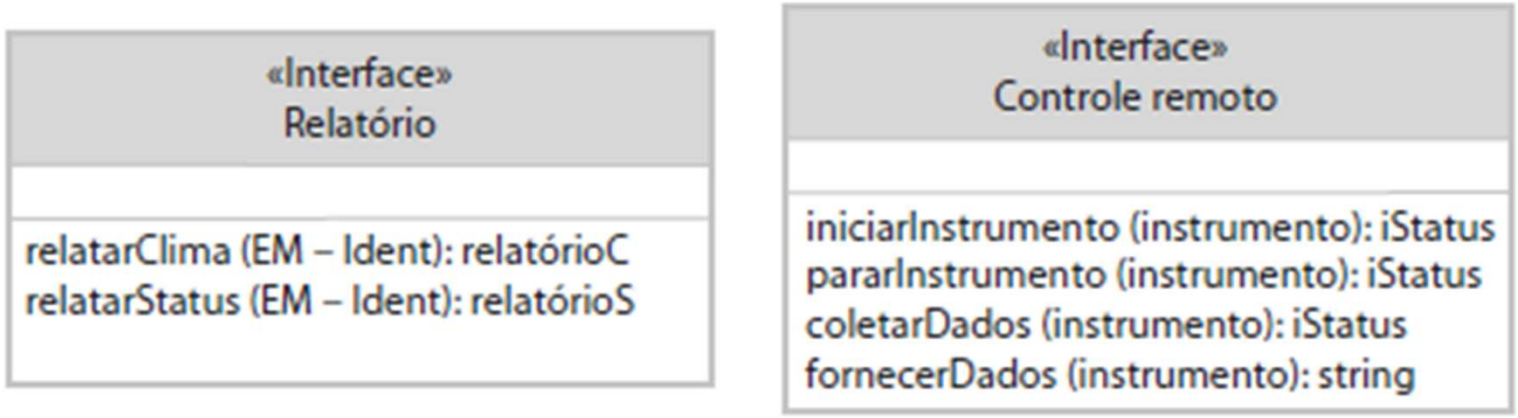
Exemplo de modelo do sistema

Figura 7.7 Diagrama de estado da estação meteorológica



Especificação de interface

Figura 7.8 Interfaces da estação meteorológica



Padrões de projeto

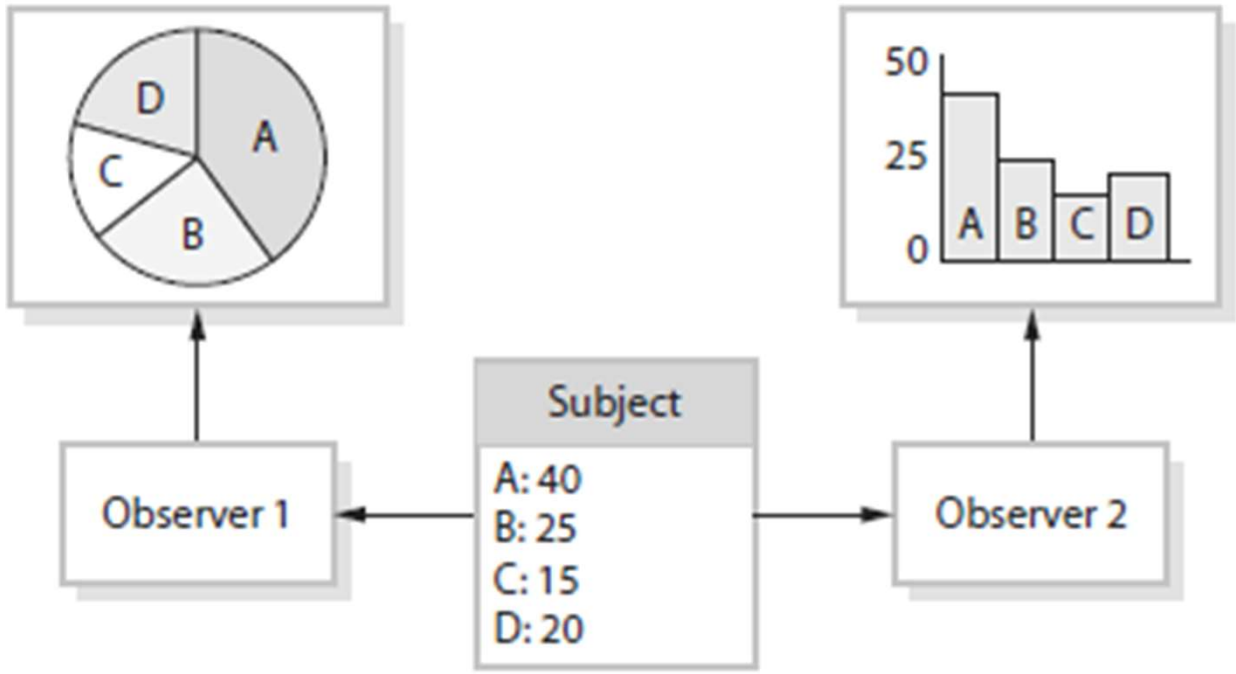
- Motivação: padrões de projeto de prédios
- Formas de descrever as melhores práticas, bons projetos e capturar a experiência de uma forma que torne possível a outros reusar essa experiência
- Soluções já testadas
- Geralmente associados a projeto orientado a objetos
  - Incluem herança e polimorfismo

Padrões de Projeto

- Elementos essenciais de um padrão de projeto:
  1. Nome significativo (referência)
  2. Descrição do problema (contexto de aplicação)
  3. Descrição da solução de projeto, seus relacionamentos e suas responsabilidades. Não é o projeto concreto em si, mas um modelo (em geral, expresso graficamente) para ele a ser instanciado de diferentes maneiras.
  4. Uma declaração das consequências (resultados e compromissos) da aplicação do padrão

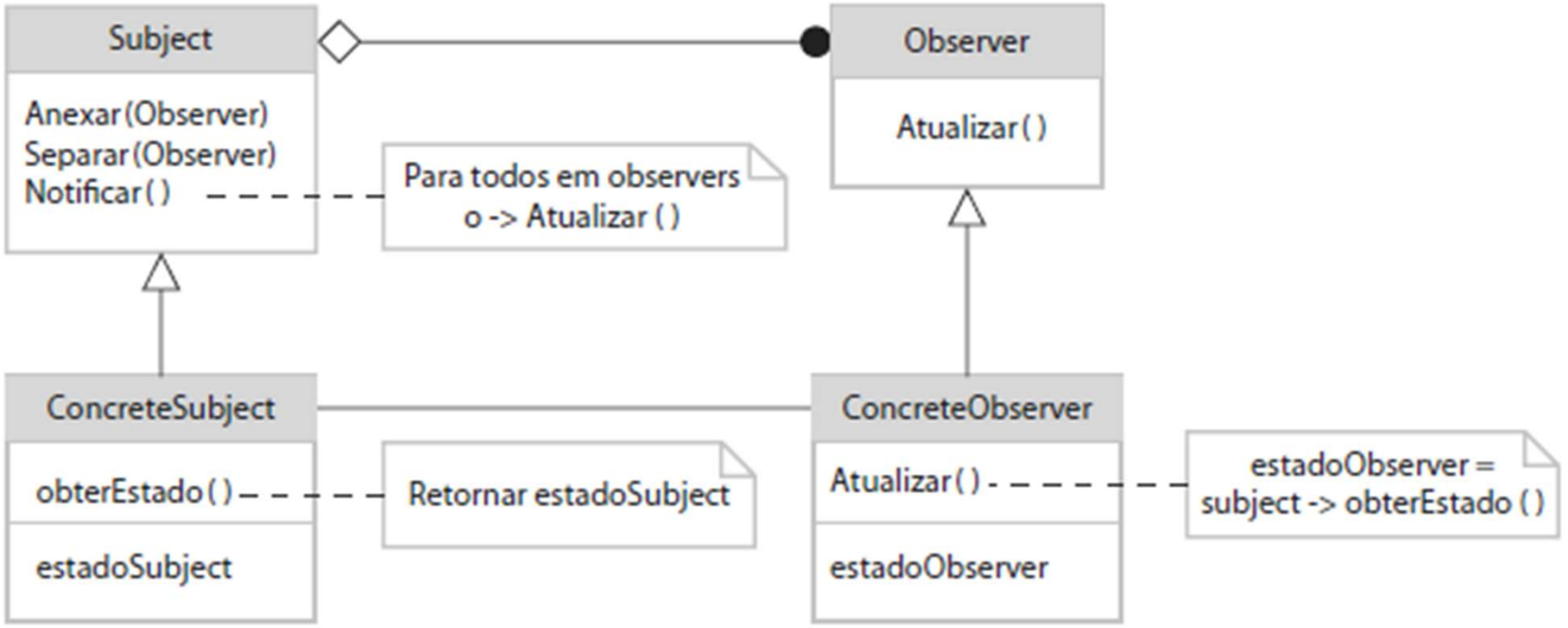
Exemplo de padrão de projeto

Figura 7.9 Múltiplos *displays*



Exemplo de padrão de projeto

Figura 7.10 Um modelo UML do padrão Observer



Exemplo de padrão de projeto

Quadro 7.2 O padrão Observer

Nome do padrão:	Observer
Descrição:	Separa o *display* do estado de um objeto a partir do objeto em si e permite que sejam fornecidos *displays* alternativos. Quando o estado do objeto muda, todos os *displays* são automaticamente notificados e atualizados para refletir a mudança.
Descrição do problema:	Em muitas situações, você precisa fornecer vários *displays* de informações do estado, como um *display* gráfico e em tabela. Nem todos eles podem ser conhecidos quando a informação é especificada. Todas as apresentações alternativas devem apoiar a interação e, quando o estado é alterado, todos os *displays* devem ser atualizados. Esse padrão pode ser usado em todas as situações em que mais de um formato de *display* de informações de estado é necessário, e em que saber sobre os formatos de *display* específicos usados não é necessário para o objeto que mantém as informações do estado.
Descrição da solução:	Trata-se de dois objetos abstratos, Subject e Observer, e dois objetos concretos, ConcreteSubject e ConcreteObject, que herdam os atributos dos objetos abstratos relacionados. Os objetos abstratos incluem as operações gerais aplicáveis em todas as situações. O estado a ser exibido é mantido no ConcreteSubject, que herda as operações de Subject permitindo adicionar ou remover Observers (cada Observer corresponde a um *display*) e emitir uma notificação quando o estado mudar. O ConcreteObserver mantém uma cópia do estado do ConcreteSubject e implementa a interface atualizar ( ) do Observer, que permite que essas cópias sejam mantidas nessa etapa. Automaticamente, o ConcreteObserver exibe o estado e reflete as mudanças sempre que o estado é atualizado. O modelo UML do padrão é mostrado na Figura 7.10.
Consequências:	O Subject só conhece o Observer abstrato e não sabe detalhes da classe concreta. Portanto, há um acoplamento mínimo entre esses objetos. Devido a essa falta de conhecimento, as otimizações que melhoram o desempenho do *display* são impraticáveis. As alterações no Subject podem causar uma série de atualizações ligadas aos Observers relacionados para serem geradas, algumas das quais podem não ser necessárias.

Leitura Específica

- [1] Vídeo "**Tutorial de Diagramas de Classes UML**". Disponível em: https://www.youtube.com/watch?v=rDidOn6KN9k
- [2] Vídeo "**Tutorial de Caso de Uso UML**". Disponível em: https://www.youtube.com/watch?v=ab6eDdwS3rA
- [2] SOMMERVILLE, Ian. **Engenharia de Software**. 10ª Ed. São Paulo: Pearson Prentice Hall, 2011. Páginas 147 até 154. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Loader/168127/pdf

Aprenda +

- PINTO, Hudson. **Atividades básicas ao processo de desenvolvimento de Software**. Disponível em: https://www.devmedia.com.br/atividades-basicas-ao-processo-dedesenvolvimento-de-software/5413
- **UML: Diagrama de Casos de Uso**. Disponível em: https://medium.com/operacionalti/uml-diagrama-de-casos-de-uso-29f4358ce4d5