Application-Oriented System Design

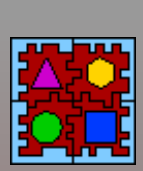
LISHA/UFSC

Prof. Dr. Antônio Augusto Fröhlich
guto@lisha.ufsc.br
http://www.lisha.ufsc.br/~guto

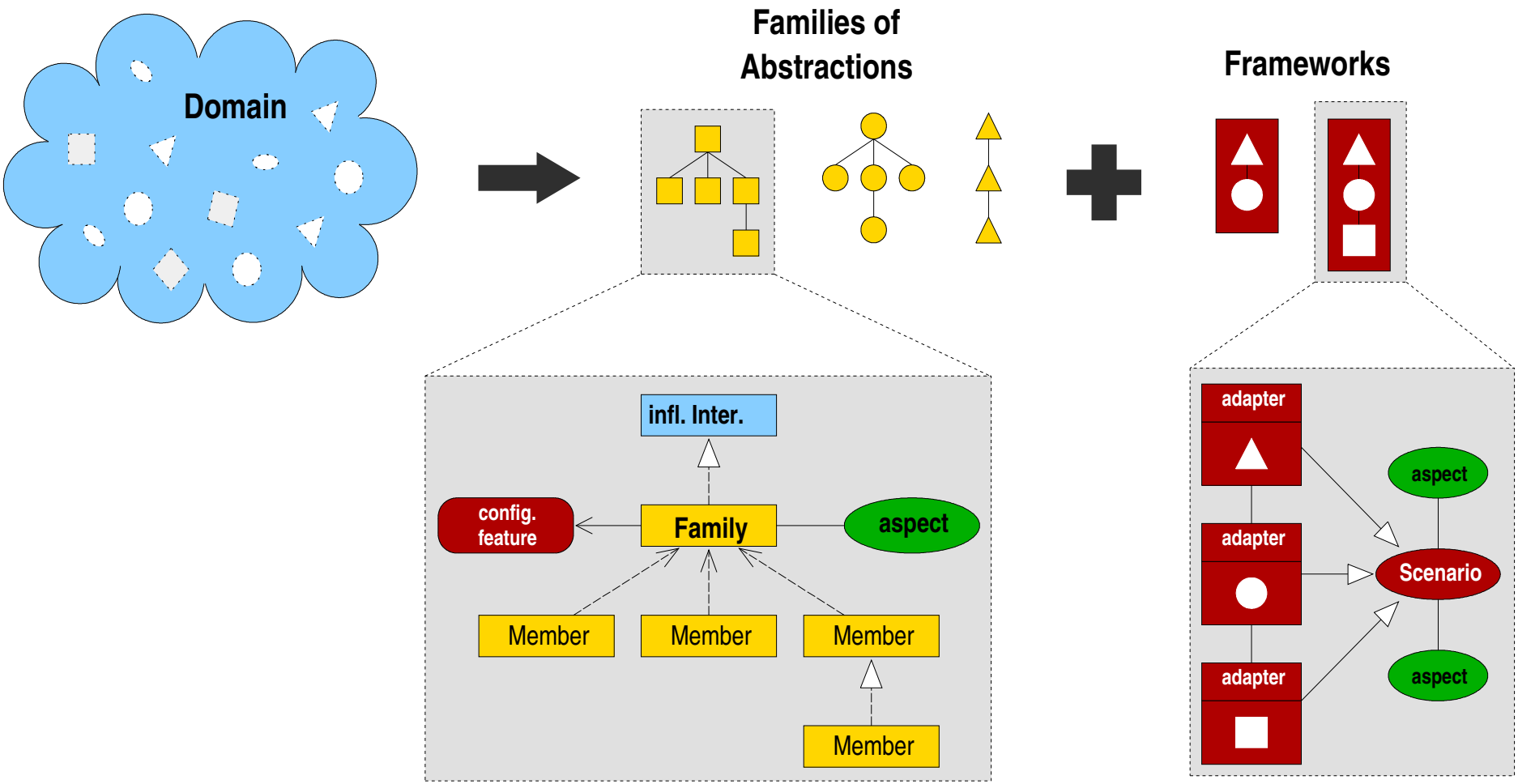
March 2004

Application-Oriented Operating Systems

"An application-oriented operating system is only defined with regard to the corresponding application(s), for which it implements the necessary run-time support that is delivered as requested."



Application-Oriented System Design

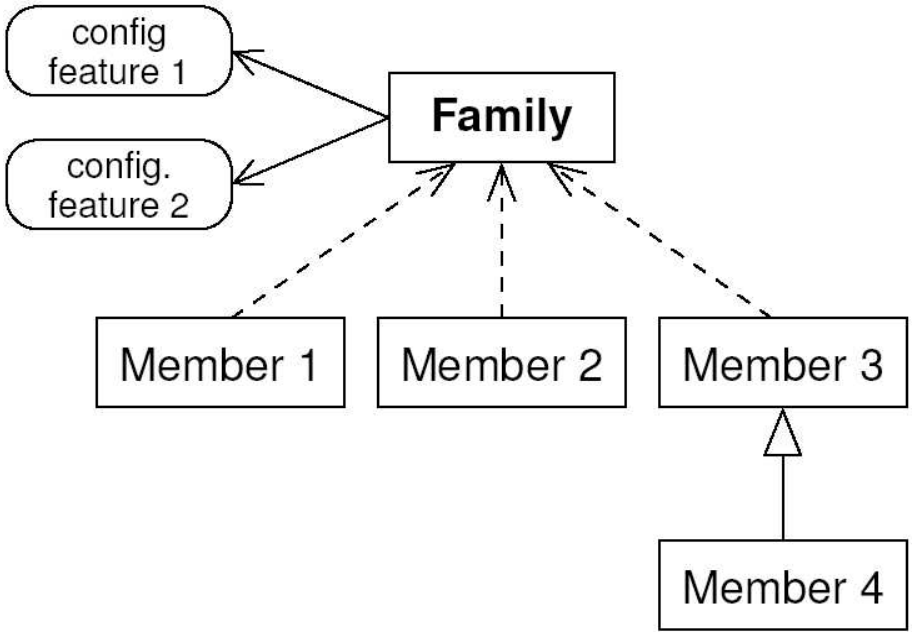


Application-Oriented Domain Decomposition

- Abstractions model domain entities
- Commonality analysis
  - Build families of abstractions
- Variability analysis
  - Shape family members (subclassing or not)
  - Separate scenario aspects
- Factorization
  - Configurable features
- Inter-family relationships
  - System-wide properties
  - Reusable architectures

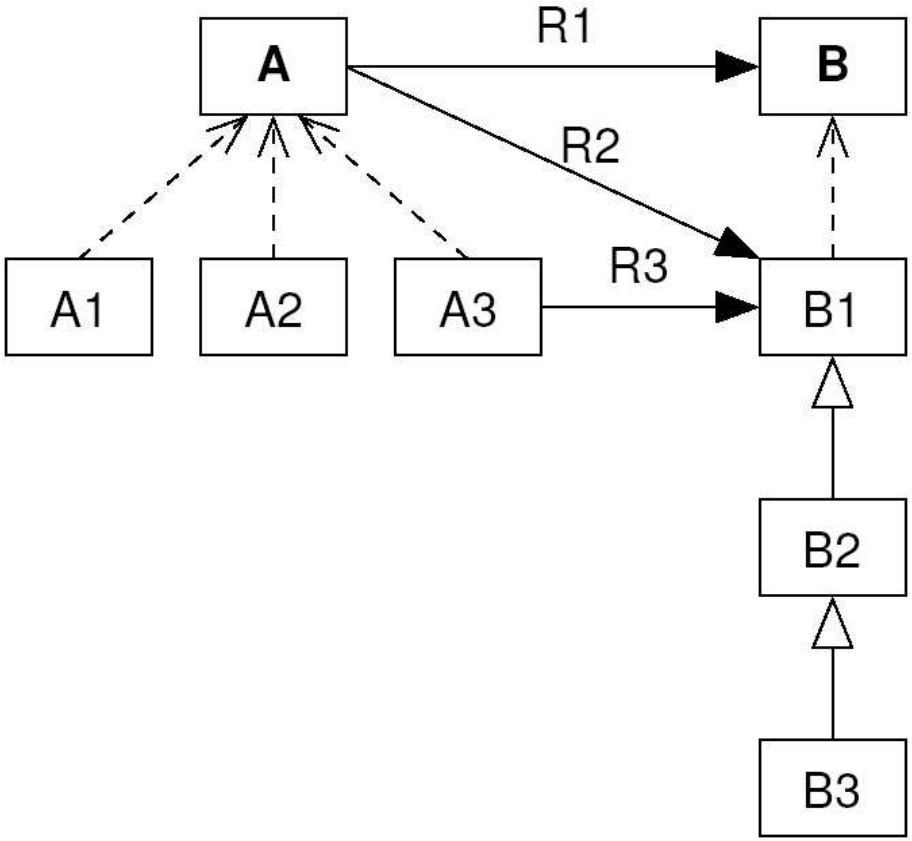
Scenario-Independent Abstractions

- Can be reused in a variety of scenarios
- Yield software components
  - Application-ready ADTs
  - Correspondence with domain entities
- Families
  - Class hierarchy
  - Cooperating classes
  - Common package
    - Base class or utility classes
    - Configurable features



Inter-Family Relationships

- Shape framework composition rules
- Avoid
  - Restrictive rules
  - Loose rules
  - Relations for the sake of reuse
    - Factorization

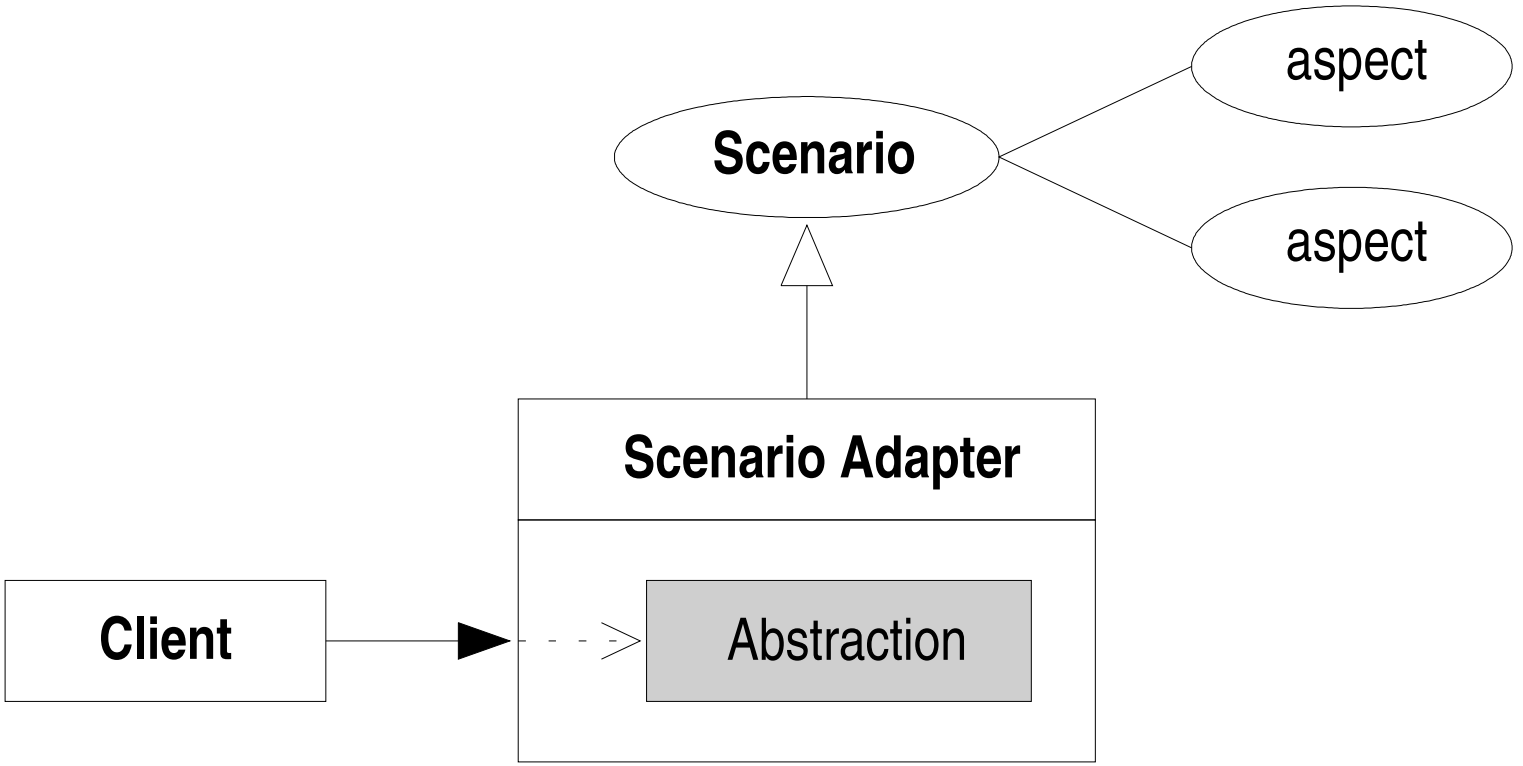


Scenario Aspects

- Properties that transcend the scope of abstractions
  - Scenario dependencies
  - Non-functional properties
- Can also be organized as families
- Application to abstractions
  - AOP Weaver
  - Scenario adapters

Scenario Adapters

- Scenario adapters
  - Adapt an abstraction to match the semantics dictated by a scenario



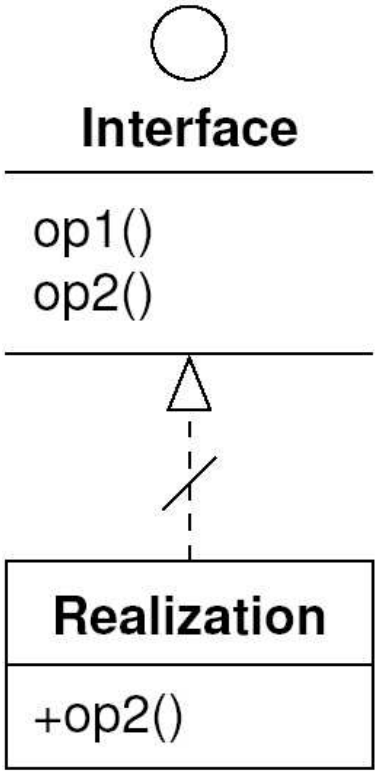
Configurable Features

- Configurable features differ from aspects in that
  - They are specific to a single family of abstractions (do not crosscut families)
  - They are not transparent to abstractions
    - but encapsulate generic programming implementations of algorithms and data structures associated to the feature that can be reused by abstractions when the feature is turned on

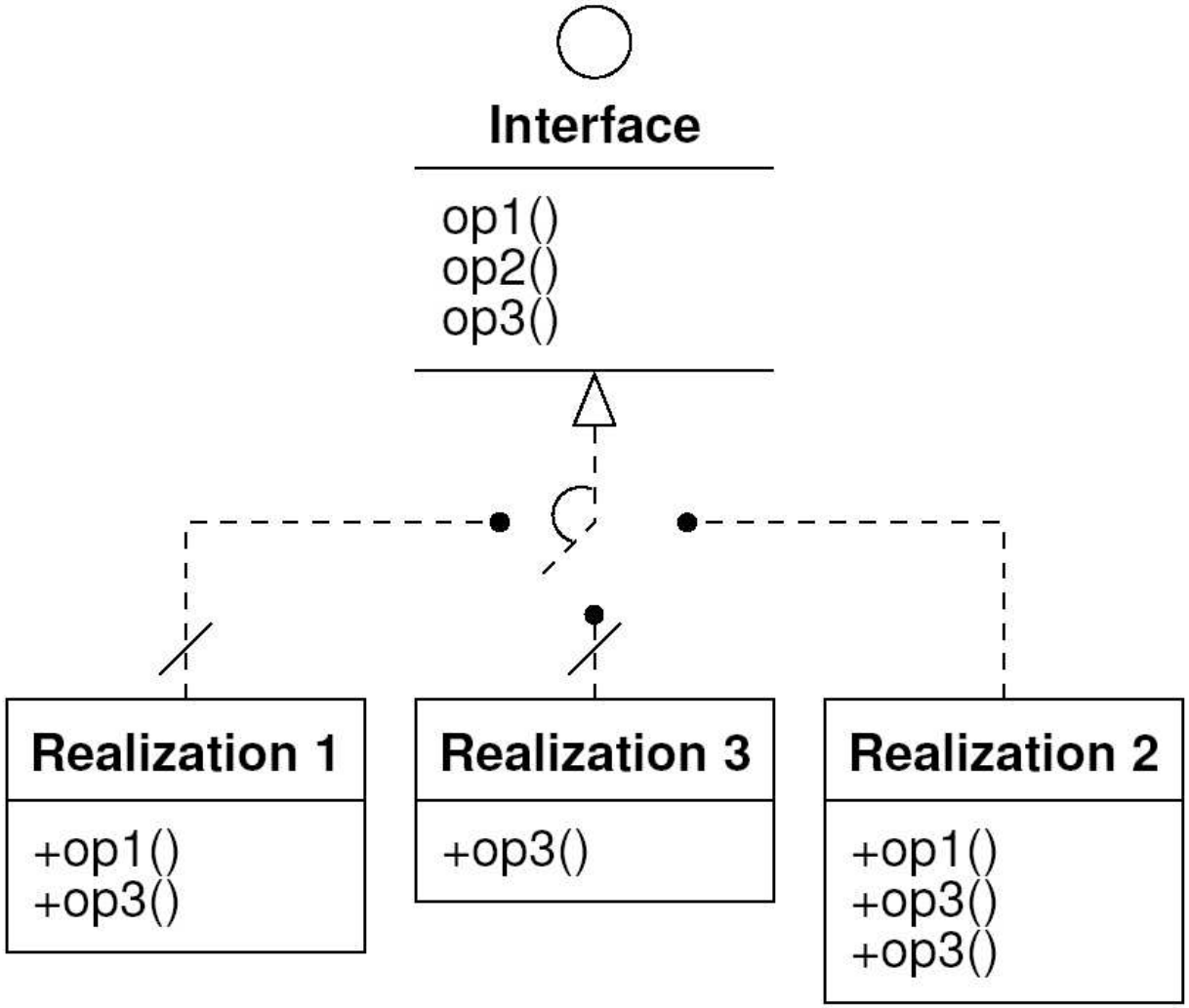
Inflated interfaces

- Export families of abstractions to applications as if they were a single abstraction
  - Well-known to application programmers
  - Comprehensive
  - Promote requirement analysis

Partial and Selective Realization

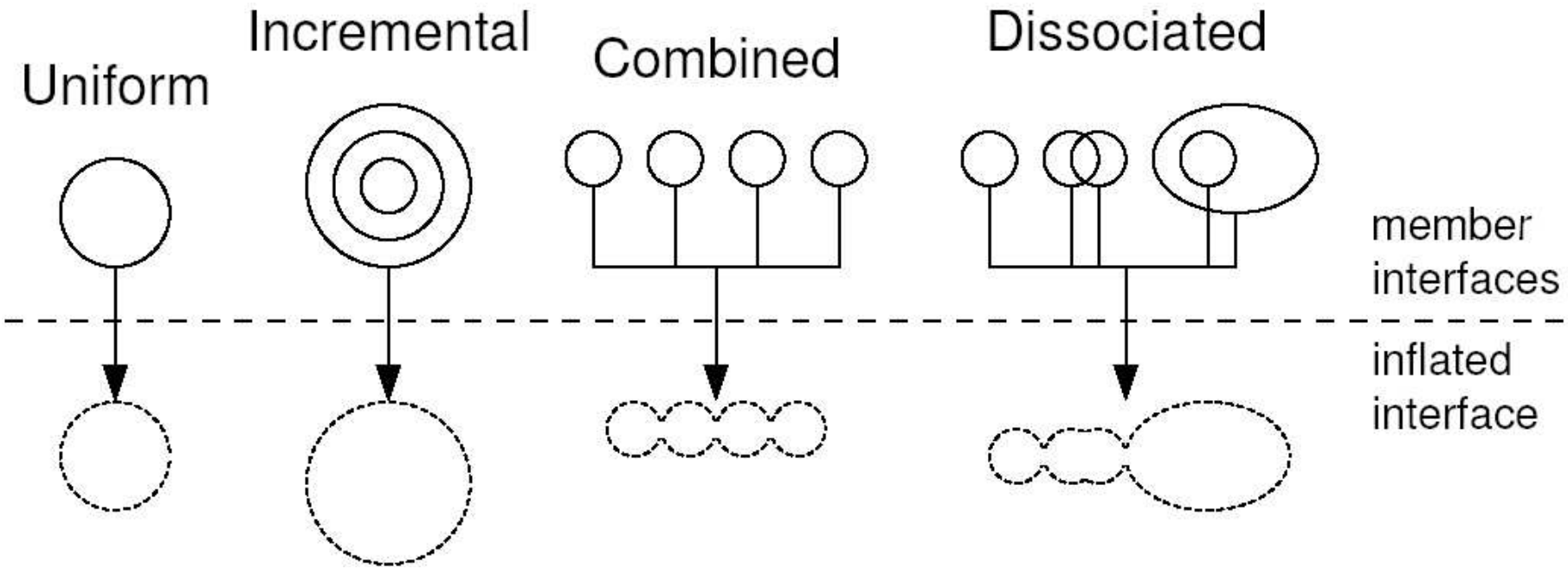


(a)

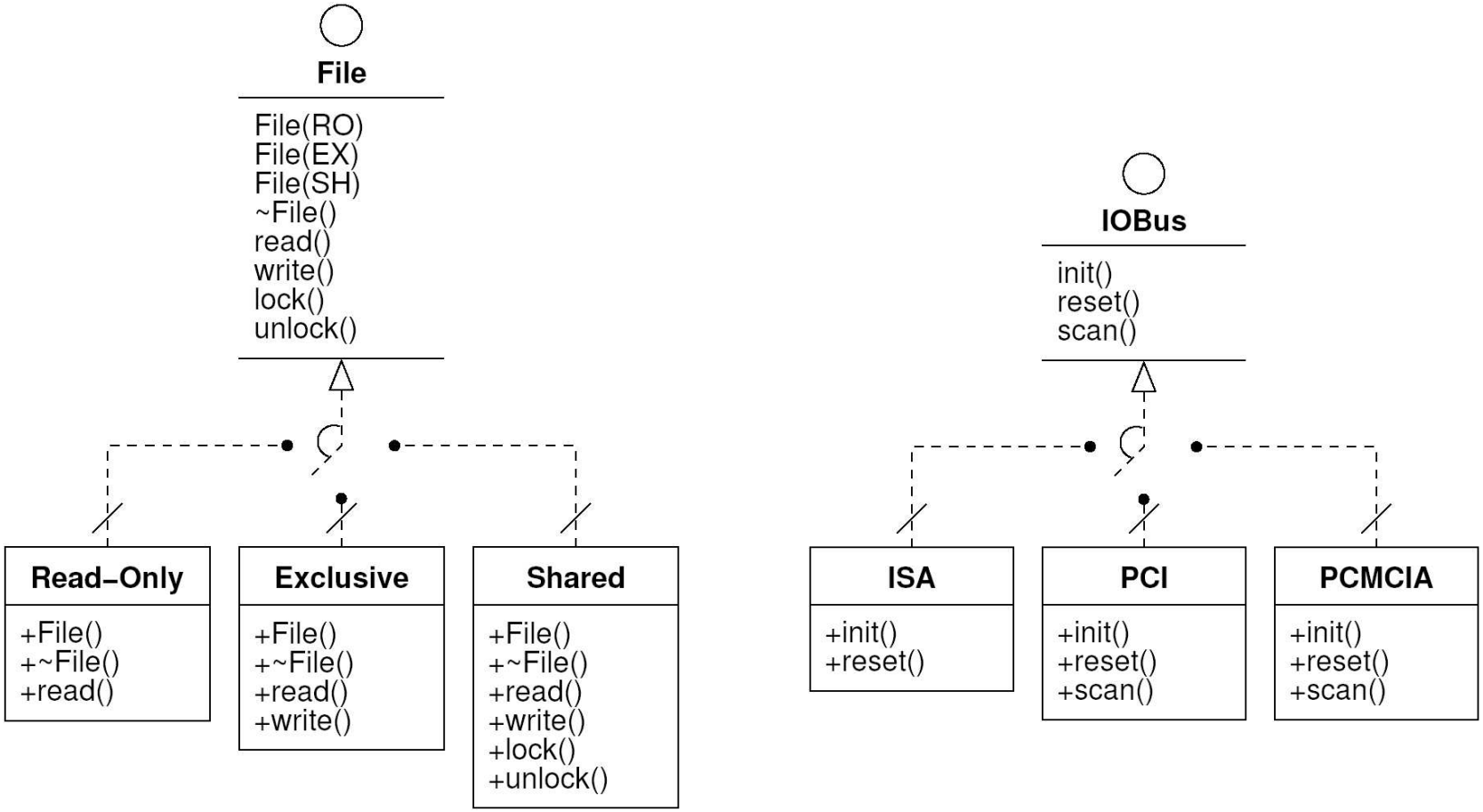


(b)

Inflated Interface Types



Inflated Interfaces of Dissociated Families of Abstractions



Component Frameworks

- Also known as “black-box frameworks”
  - Based on the idea of software components and defined interfaces (in opposition to inheritance and overriding used in white-box frameworks)
  - The reuse of a component does not imply on reusing the whole framework along with it
- Defined as compositions of scenario adapters (place holders for components) and a configuration knowledge base that specifies components' requirements and dependencies

Application-Oriented OS