Robustness Analysis
Use Cases and Function Points

Charles Wesolowski CFPS, OCUP, OCRES
Chief Architect
QinetiQ North America
Systems Engineering Group
890 Explorer Blvd
Huntsville, Al 35806

Analysis Techniques and Models

- Analysis is an activity that organizes information into a model
  - Its “primary purpose is to formulate a model of the problem domain that is independent of implementation considerations.” [UML 2.0 Infrastructure]
- Requirements Analysis Techniques include
  - Use Case Analysis
  - Robustness Analysis
  - Function Point Analysis
- UML is a language for expressing aspects of
  - Behavior: “The observable effects of an operation or event, including its results”
  - Structure: “Types, classes, relationships, attributes, and operations”

[UML 2.0 Infrastructure]

UML assists in illustrating the results of Analysis Activities

Use Case Analysis Overview

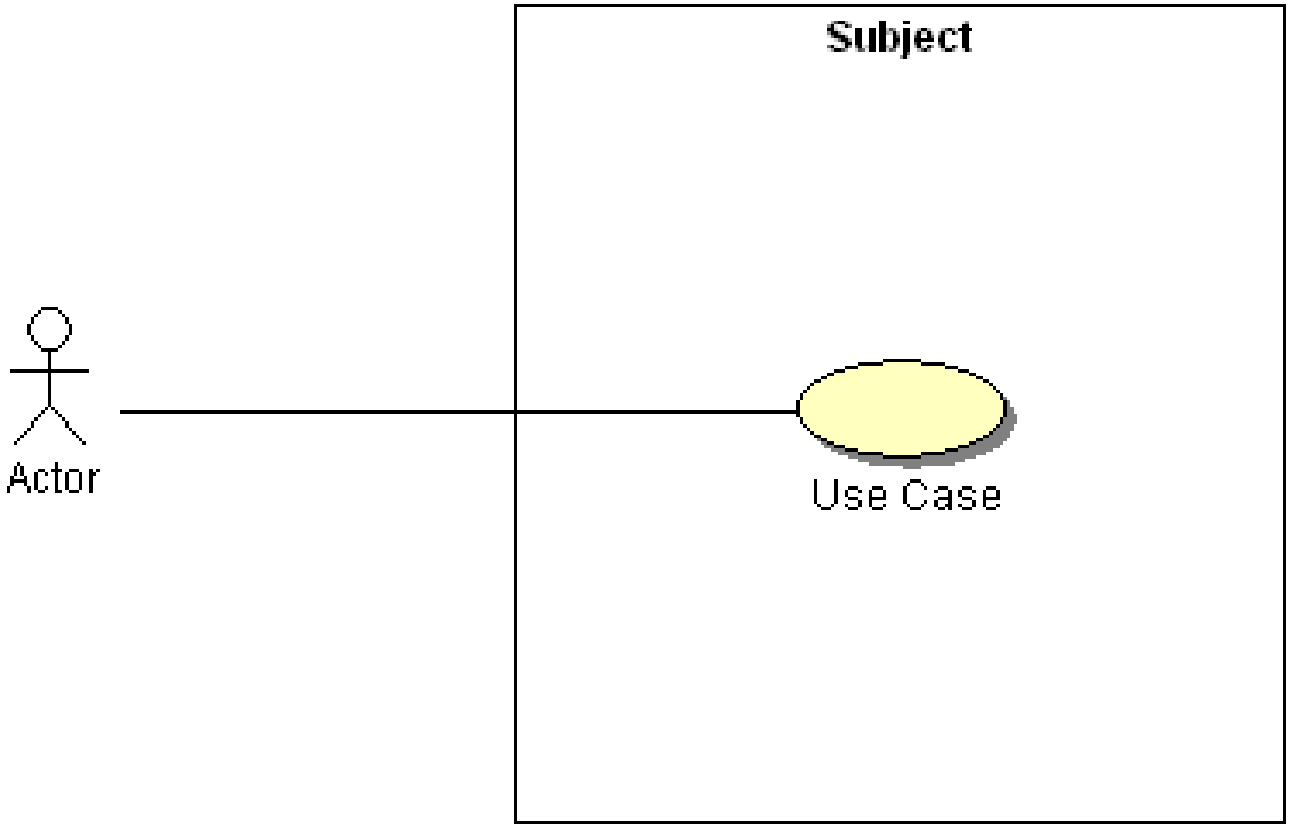
What is Use Case Analysis?

- A process that describes observable behavior (Jacobsen 1992)
  - Practiced in “A Use Case Driven Approach” to software development
  - Use case diagrams indicate observable behaviors
  - Behavioral descriptions (use case elaboration techniques) vary in form
  - Interactions, state machines, activities, or procedures
- Defines three modeling elements (UML stereotypes)
  - Subject
  - Use Case
  - Actor
- Illustrated using formal diagrams
- Supports structured and object-oriented development strategies

Use Case Definitions

- Use Case Diagram – [16.4] UML 2.0 Superstructure Specification
  - "Use case diagrams are a specialization of class diagrams such that the classifiers shown are restricted to being either Actors or Use Cases."
- Use Case – [16.3.6] UML 2.0 Superstructure Specification
  - "A use case is the specification of a set of actions performed by a system, which yields an observable result that is, typically, of value for one or more actors or other stakeholders of the system."
  - "The subject of a use case could be a system, or any other element that may have behavior, such a a component, subsystem, or class."
- Actor – [16.3.6] UML 2.0 Superstructure Specification
  - "An actor specifies a role played by a user or any other system that interacts with the subject."

What is a Use Case Diagram?



A Use Case Diagram associates behavior with an external actor.

The association indicates an unspecified form of communication.

It does not indicate data.

Use Case Elaborations

- Elaborations are descriptions that explain behavior
  - Inputs
  - Outputs
  - State
  - Procedure
  - Interactions
- May take the following forms
  - Text
  - UML Activity Diagrams
  - UML Sequence Diagrams
  - UML Communication Diagrams

Elaboration Techniques vary based upon Abstraction of Use Case Model

Elaboration Artifact Comparison

Use Case Name: Monitor Pressure Alarm

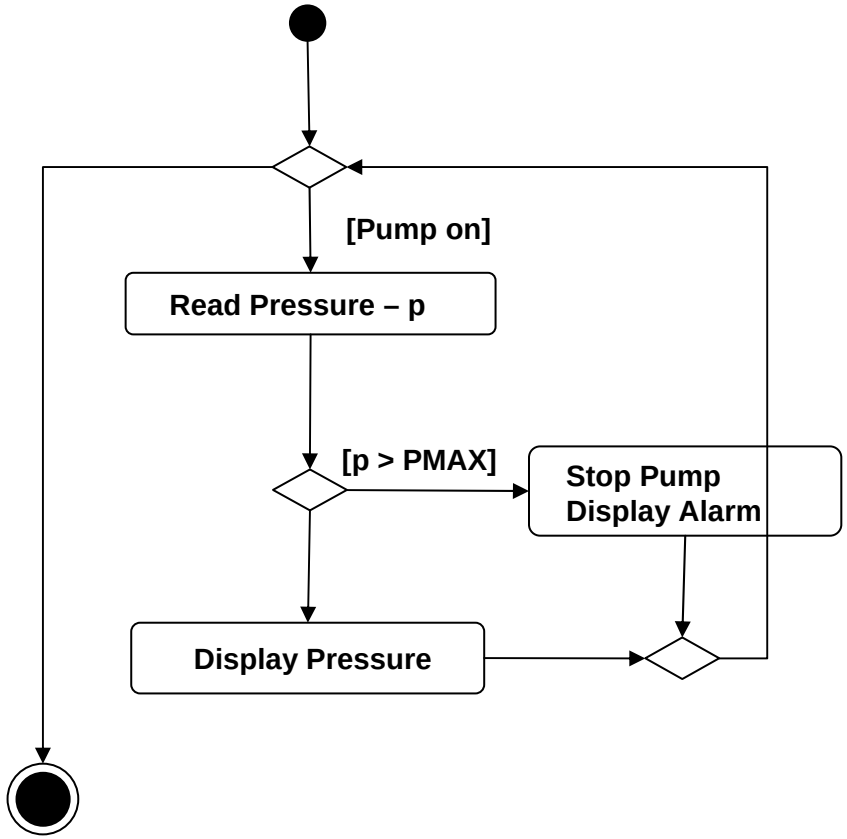
Subject: Fault Tolerance

Actors: LED, Pump, Pressure Sensor

Pre-Conditions:
Pump is on
Trigger:
Pump Enabled

Behavior:
while Pump is on
Read Pressure
if p > PMAX
Open Relief Valve
Stop Pump
Display Alarm
else
Display Pressure

Post-Conditions:
Pump Off



Use Case Elaborations Do NOT necessarily reflect Functional Size
Other complexity measures are available (e.g. McCabe)
Depending on Techniques Employed

The issue with Use Case Models is that

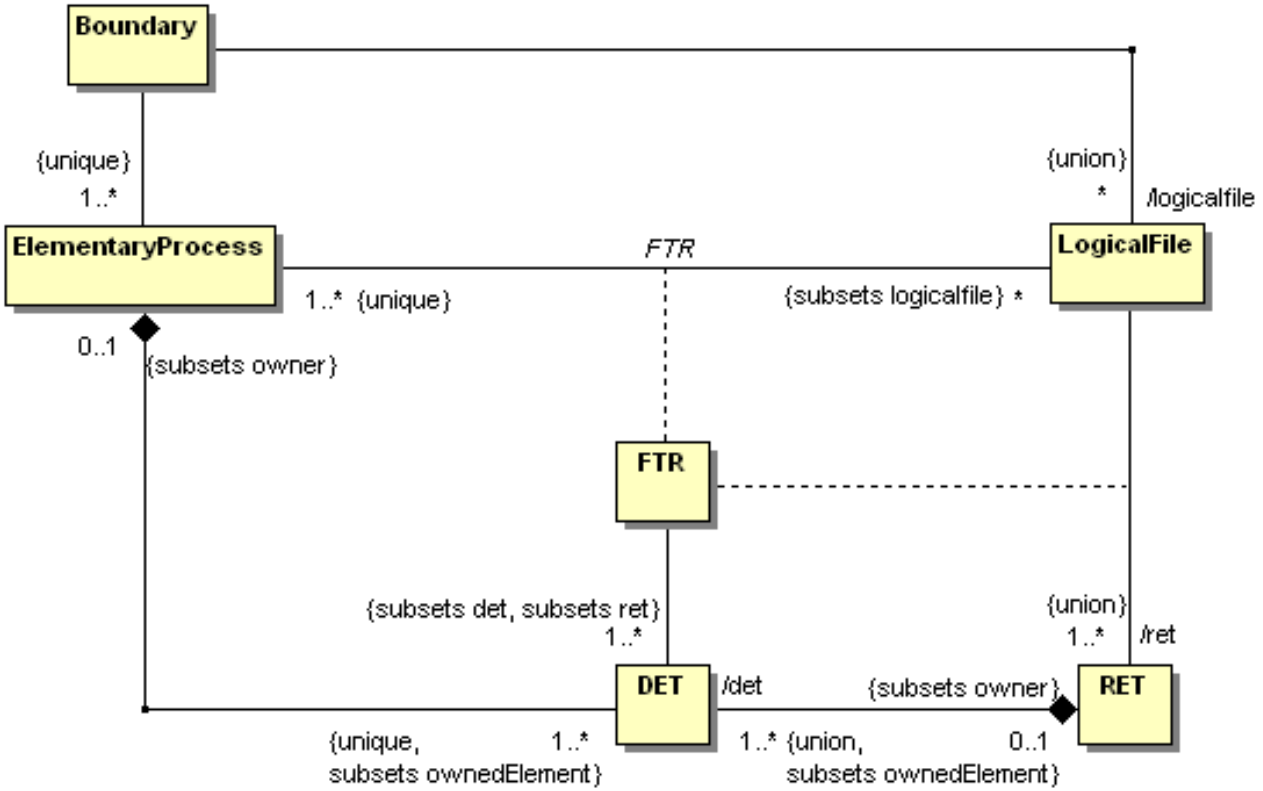
- The Subject
  - Indicates a Boundary BUT
    - May not meet the functional sizing criteria for “a software boundary”
- A Use Case
  - Indicates a Behavior BUT
    - May not meet functional sizing criteria for “an elementary process”
- A Use Case Diagram (UML 2.x)
  - Indicates a behavioral context relative to external elements (Actors) BUT
    - Does not express the Domain Object Model (Data)
- Elaboration techniques vary in form and utility
  - Behavioral descriptions are unnecessary for computing function points
    - Requires Data Model

Function Point Analysis Overview

What is Function Point Analysis?

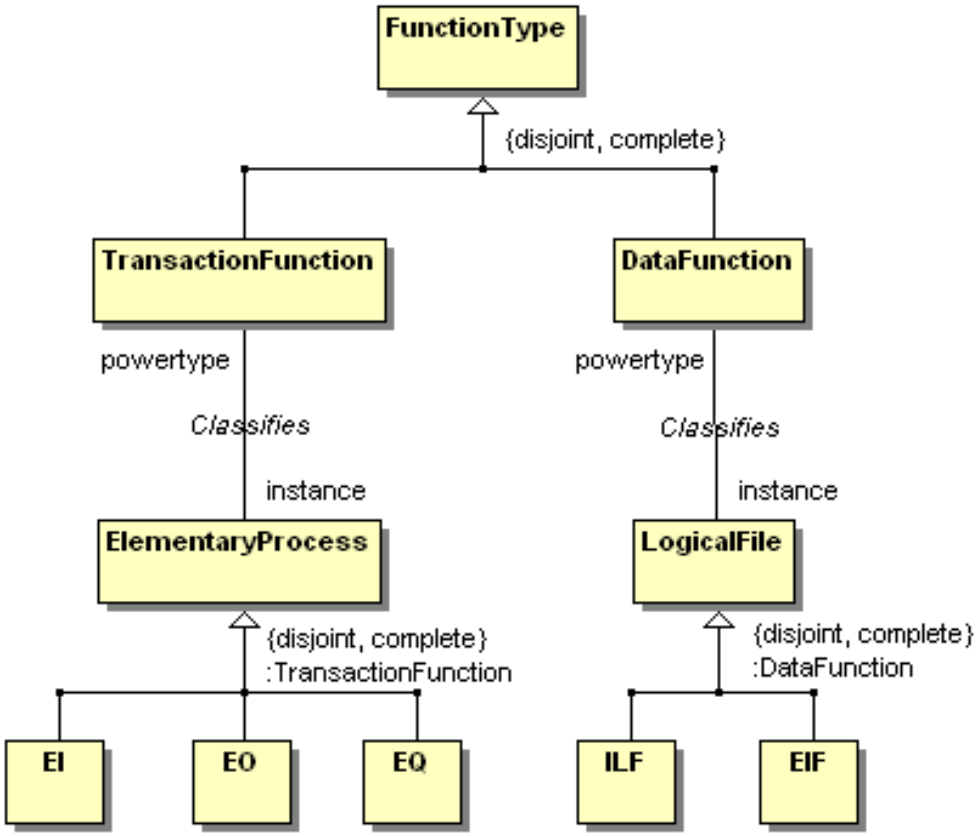
- A technique devised by Albrecht (1977)
  - Practiced by IFPUG
  - Recognized Functional Sizing Method (FSM) by ISO/IEC 20926
- Defines five key modeling elements (UML stereotypes)
  - Boundary
  - Elementary Process
  - Logical File
    - Record Element Type (RET)
  - Data Element Type (DET)
- Supports structured and object-oriented development strategies

IFPUG Modeling Elements



Function Point Analysis identifies the key behaviors and data necessary to describe functional software architecture

IFPUG Function Type Taxonomy



Function Point Analysis classifies and counts architectural elements
In order to measure the functional size of the software

Comparision of Analysis Techniques

- Use Case Analysis
  - Identifies and elaborates behavior
  - Depends on Subject
    - Systems
    - Software
    - Hardware
    - Business processes
- Function Point Analysis
  - Measures the size of software functional user requirements
  - Requires Data view in addition to Behavioral view

Robustness Analysis Overview

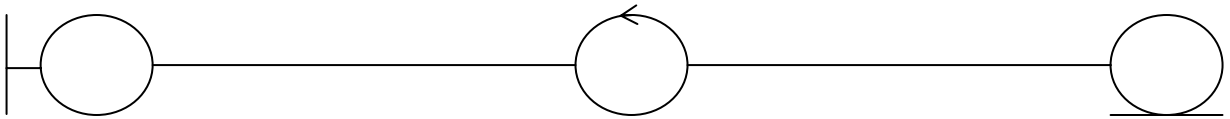
Robustness Analysis

- A techniques devised by Ivar Jacobson (1992)
  - Practiced in a Use Case Driven Approach
  - Integrates Use Case and Data Views
  - Produces an Analysis Model
- Defines three modeling elements
  - Boundary
  - Controller
  - Entity
- Illustrated using formal diagrams
- Support functional and object-oriented development strategies

Bridging Use Cases and Function Points

- Robustness Analysis
  - Reveals the scope of Use Cases
  - Accounts for behavior and data

Jacobson's Class Notation



UML Class Notation

<<Boundary>>	<<Controller>>	<<Entity>>

IFPUG's Vocabulary

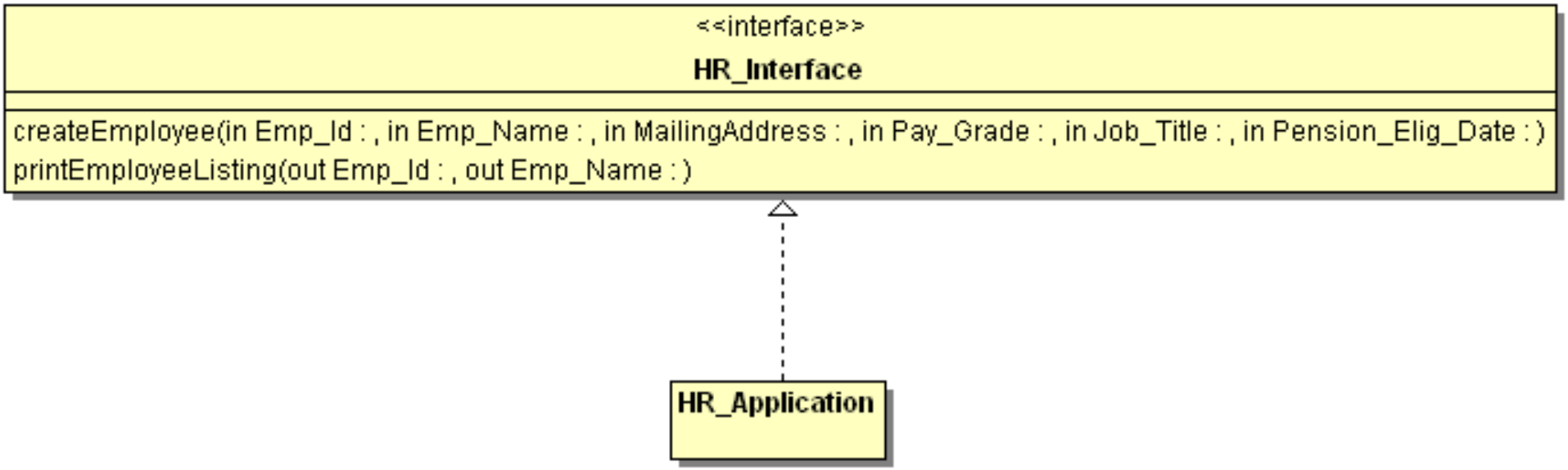
Boundary	Elementary Process	Logical File

Robustness Classes are stereotypes of UML Metaclasses that are useful in expressing Functional Architecture

Boundaries

- An IFPUG Application Boundary
  - Is influenced by the purpose of the count
  - Indicates the border between the software being measured and the user
  - Is strongly related to the UML Interface
  - Robustness Boundary classes enable modeling elementary processes as operations
- UML Interfaces
  - An interface is a kind of classifier that represents a declaration of a set of coherent public features and obligations.
  - Interfaces provide a way to partition and characterize groups of properties that realizing classifier instances must possess.
  - An interface specifies a contract; any instance of a classifier that realizes the interface must fulfill that contract.
  - An interface does not specify how it is to be implemented, but merely what needs to be supported by realizing instances.

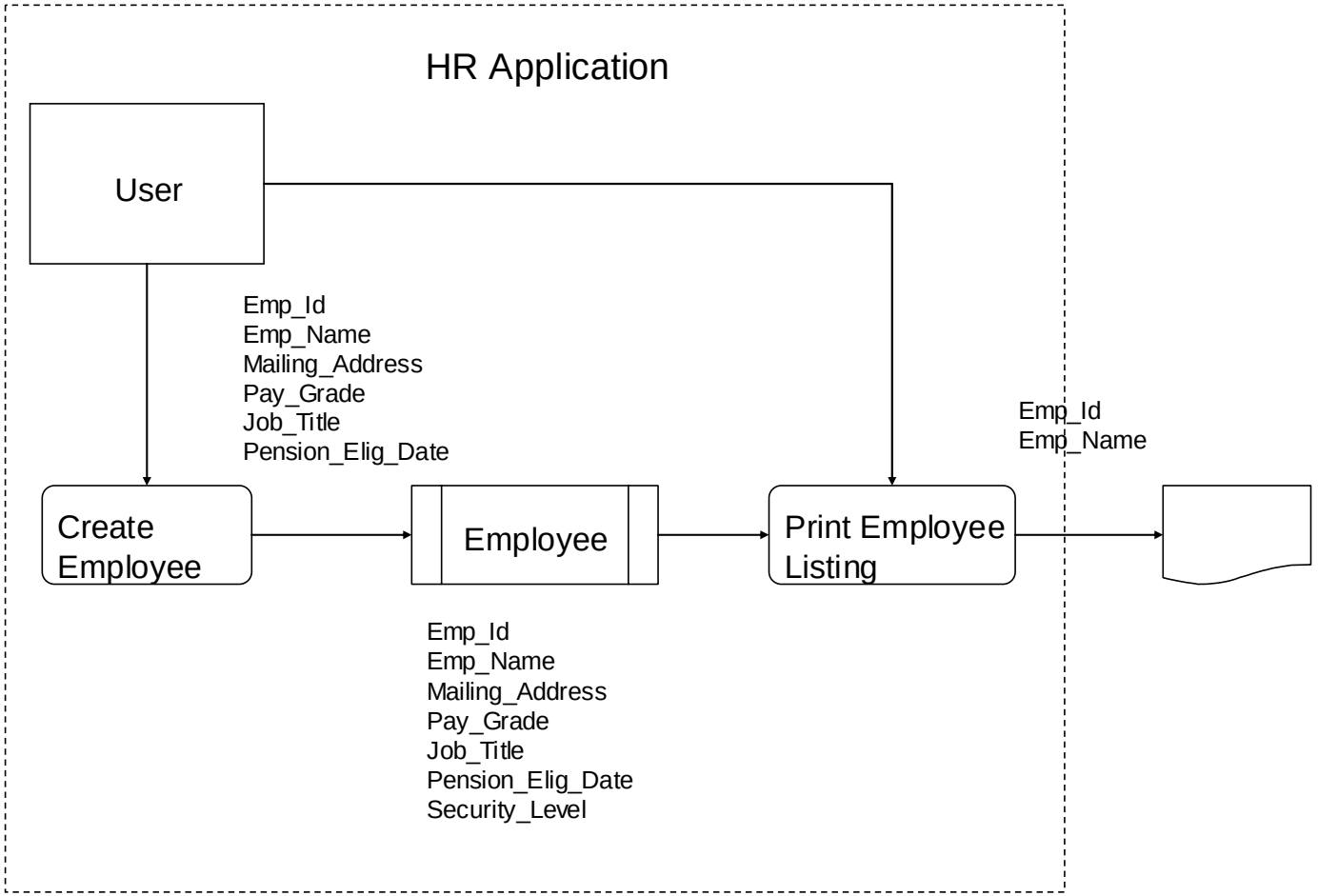
UML Interface Expression of CPM Example



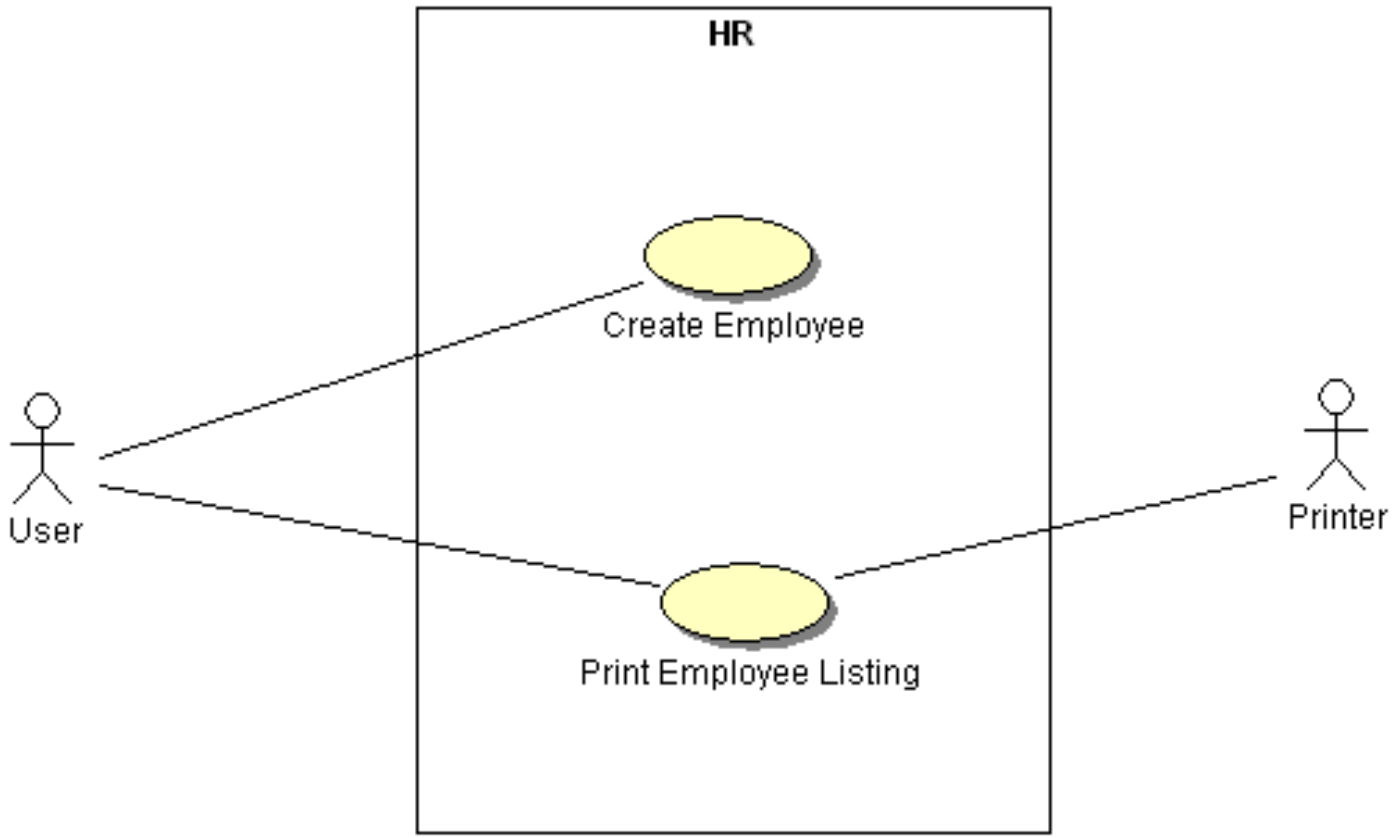
UML Interface Notation illustrates the separation between Specification and Implementation

Example from CPM

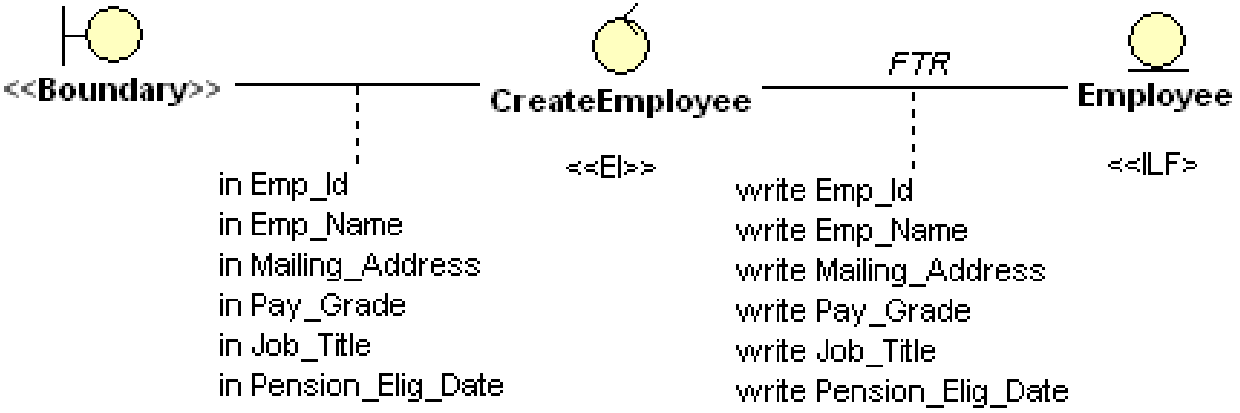
Example from CPM (4.2.1 Part 3, p.1-50)



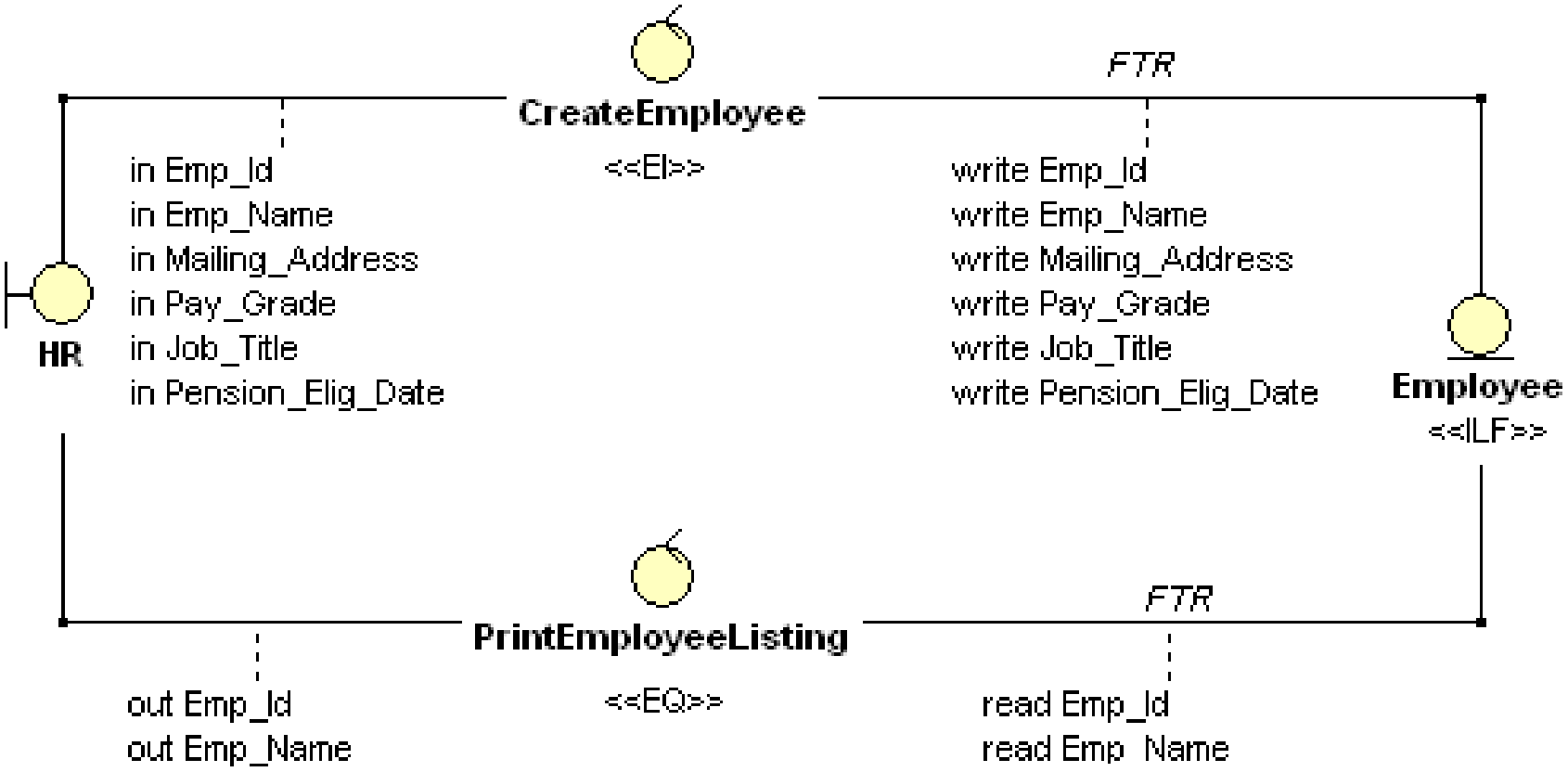
CPM Example Use Case View



Robustness Analysis of CPM Example (Abstract Boundary)



Robustness Analysis of CPM Example (Concrete Boundary)



What does Robustness Analysis Buy Me?

- Opportunity to use Function Point Analysis Process to
  - Validate Use Case Model
  - Validate Domain Object Model
  - Measure Functional Software Size
- Opportunity to meet CMMI Requirements Development criteria
  - Specific Goal 3 – Analyze and Validate Requirements
    - The requirements are analyzed and validated, and a definition of functionality is developed.
  - Specific Practice 3.2 Establish and maintain a definition of required functionality
    - Sub-practice 1 – Analyze and quantify functionality required by end users
    - Sub-practice 2 – Analyze requirements to identify logical or functional partitions
    - Sub-practice 3 – Partition requirements into groups based on established criteria
- Opportunity to illustrate the Functional Architecture Model
  - Enhance communication and understanding between stakeholders
    - Customer, Management, Engineering

Class Exercise (CPM Example Part 3, 1-23)