

Refinements

- All design involves refinement--moving from a high-level understanding to an implementation
 - Multiple layers may be involved (not just two) depending on the complexity of the problem
- To execute a multilayer design properly, three properties must obtain:
 1. The top layer must faithfully represent the requirements
 2. Each layer must be internally consistent
 3. Each layer must faithfully represent the layer above it

Valid Refinements

- Designer needs to demonstrate that the implementation satisfies the specification
- This demonstration can be expressed using a set of verification conditions, sometimes called *proof obligations*
- The demonstration itself can be formal or informal depending on the process used

Exercise

- Assumptions
 - The requirements are expressed as a UML analysis model
 - We are concerned with the specification of a single class (\bar{A})
 - We are only concerned with functional requirements
- Question 1
 - What should our specification of this class include?

Exercise

- Assumptions
 - The requirements are expressed as a UML analysis model
 - We are concerned with the specification of a single class (A)
 - We are only concerned with functional requirements
- Question 1
 - What should our specification of this class include?
 - Attributes (types, sizes, properties, etc.)
 - Operations (signatures, pre/post conditions)
 - Invariants over the attributes

Exercise - 2

- Property 1 states that our specification for class \bar{A} must match the requirements
 - Question: How would we normally accomplish this?

Exercise - 2

- Property 1 states that our specification for class \bar{A} must match the requirements
 - Question: How would we normally accomplish this?
 - This property is usually checked via some form of review, either individual or meeting

Exercise - 3

- Property 2 states that each level must be internally consistent
- Question: How can we demonstrate the consistency of class A ?

Exercise - 3

- Property 2 states that each level must be internally consistent
- Question: How can we demonstrate the consistency of class A ?
 - Make sure that each operation maintains any invariants

Notation

- P_1, P_2, \dots, P_n are operations in the abstract specification
- S is a set of abstract states
- s is an element of S (combination of attribute values)
- s' is the abstract state s after an operation has completed
- inv is any invariant; $invA$ is any invariant of the abstract spec; and $invC$ is any concrete invariant
- $Pre-P_i(s, args)$ are the precondition for operation P_i when executed in state s with arguments $args$
- $Post-P_i(s, args, s', res)$ are the postcondition for operation P_i after execution beginning in state s with arguments $args$ resulting in state s' with results res

1. *Valid* Operations

- Within any given specification (either abstract or concrete), operations must preserve invariants
 - That is, if the invariant is true before execution and the operation terminates then the invariant is true after execution

$$\begin{aligned} & (\text{inv}(s) \wedge \text{Pre-}P_i(s, \text{args}) \wedge \\ & \text{Post-}P_i(s, \text{args}, s', \text{res})) \\ & \quad \Leftrightarrow \\ & \text{inv}(s') \end{aligned}$$

Exercise - 4

- Property 3 states that each layer must faithfully represent the layer above
- This implies checking the mapping between the abstract layer and the concrete refinement
- Question: What checks do we have to make concerning this mapping?

Exercise - 4

- Property 3 states that each layer must faithfully represent the layer above
- This implies checking the mapping between the abstract layer and the concrete refinement
- Question: What checks do we have to make concerning this mapping?
 - Each abstract state must be refined into at least one concrete state. That is, the representation is *adequate*
 - The concrete layer can't get into a state that doesn't correspond to an intended abstract state. That is, the representation is *total*
 - Each abstract operation must be faithfully *modeled* by a concrete one

More Notation

- Q_1, Q_2, \dots, Q_n are concrete operations corresponding to P_1, P_2, \dots, P_n
- t is a concrete state (a state over concrete attributes) within state set T
- retr is a function that maps from concrete states to the corresponding abstract states. That is, $\text{retr}(t) = s$, where s is the (single) abstract state that corresponds to concrete state t

2. Adequate Representation

- A refinement must be adequate
 - That is, the implementation must be rich enough that each abstract state is represented by a concrete state
 - In other words, for each abstract invariant, if the invariant is true in that state, then there must exist a corresponding state in the implementation in which the invariant is also true

$$\forall s \in S: \text{inv}(s)$$
$$\Rightarrow$$
$$(\exists t \in T: \text{invC}(t) \wedge s = \text{retr}(t))$$

3. *Total* Representation

- We must make sure that refinements don't produce states that are meaningless with respect to the abstract specification
- There may be many concrete states corresponding to a given abstract one, but each concrete state must somehow be meaningful with respect to the abstract specification
- That is, the retrieve function must be total

$$\forall t \in T: \text{invC}(t)$$
$$\Rightarrow$$
$$((\exists s \in S: s = \text{retr}(t)) \wedge \text{invA}(\text{retr}(t)))$$

Operations

- To refine operations, we must assure ourselves of two things
 - The implementation must be able to handle all inputs described in the specification
 - The output produced by the operation, along with any changes made to class attributes must satisfy the operation specification

4. Operation *Modeling*

- Circumstances acceptable to the abstract specification must be acceptable to the concrete implementation
 - That is, refinements can weaken the pre conditions (run under wider circumstances)

$$\forall t \in T: \text{invC}(t) \wedge \text{pre-}P_i(\text{retr}(t), \text{args}) \Rightarrow \text{Pre-}Q_i(t, \text{args})$$

5. Models - 2

- Execution of the implementation must leave the system in a valid state that corresponds to what the abstract specification dictates
 - That is, concrete post conditions must imply abstract post conditions

$$\begin{aligned} \forall t \in T: \text{invC}(t) \wedge \text{Pre-}Q_i(t, \text{args}) \wedge \\ \text{Post-}Q_i(t, \text{args}, t', \text{res}) \\ \Rightarrow \\ \text{Post-}P_i(\text{retr}(t), \text{args}, \text{retr}(t'), \text{res}) \end{aligned}$$

Summary

- The designer is responsible for checking five things in addition to checking that a specification matches its requirements document
 1. Operations at each level must preserve invariants
 2. A refinement must be adequate
 3. A refinement must be total
 4. Operation preconditions must be weak enough
 5. Operation postconditions must be strong enough