Chap. 10
Specifying Operations

Object Oriented Systems Analysis and Design Using UML, (4th Edition), McGraw Hill

Topics Covered

- The Role of Operation Specifications
- Contracts
- Describing Operation Logic
- Object Constraint Language
- Creating an Operation Specification

Why We Specify Operations

- From analysis perspective:
  - Ensure users' needs are understood
- From design perspective:
  - Guide programmer to an appropriate implementation (i.e. method)
- From test perspective:
  - Verify that the method does what was originally intended

Services Among Objects

- When objects collaborate, one object typically provides a service to another

- Examples:
  - A `Client` object might ask a `Campaign` object for its details
  - The same `Client` object might then ask a boundary object to display its related `Campaign` details to the user

Contracts: an Approach to Defining Services

- A service can be defined as a contract between the participating objects
- Contracts focus on inputs and outputs
- The intervening process is seen as a black box, with irrelevant details hidden
- This emphasises service delivery, and ignores implementation

Contract-Style Operation Specification

- Intent / purpose of the operation
- Operation signature, including return type
- Description of the logic
- Other operations called
- Events transmitted to other objects
- Any attributes set
- Response to exceptions (e.g. an invalid parameter)
- Non-functional requirements

Types of Logic Specification

- Logic description is probably the most important element
- Two main categories:
  - ***Non-algorithmic*** methods focus on ***what*** the operation should achieve — black box approach
  - ***Algorithmic*** types focus on ***how*** the operation should work — white box approach

Non-Algorithmic Techniques

- Use when correct result matters more than the method used to reach it
- Or no decision made yet about best method
  - Decision table
  - Pre- and Post-Condition Pairs

Decision Table

- All work by identifying:
  - Combinations of initial conditions = 'rules'
  - Outcomes that should result depending on what conditions are true = 'actions'
- Rules and actions are displayed in tabular form

Example Decision Tree

Conditions to be tested

Conditions and actions	Rule 1	Rule 2	Rule 3
Conditions			
Is budget likely to be overspent?	N	Y	Y
Is overspend likely to exceed 2%?	-	N	Y
Actions			
No action	X		
Send letter		X	X
Set up meeting			X

Possible actions

Pre- / Post- Condition Pair

- Logically similar to decision table
- Identifies conditions that:
  - must be true for operation to execute = pre-conditions
  - must be true *after* operation has executed = post-conditions
- May be written in formal language (e.g. OCL)

Pre- / Post- Condition Pair (1)

`Advert.getCost()`

pre-conditions:

none

post-conditions:

a valid money value is returned

Pre- / Post- Condition Pair (2)

`Campaign.assignStaff(creativeStaff)`

pre-conditions:

`creativeStaff` is valid

post-conditions:

a link is created between a `Campaign` object and a `creativeStaff` object

Pre- / Post- Condition Pair (3)
`Change staff grade`

`CreativeStaff.changeGrade(grade,gradeChangeDate)`

pre-conditions:

- `grade` is valid
- `gradeChangeDate` is a valid date

post-conditions:

- a new `staffGrade` object exists
- new `staffGrade` object linked to `creativeStaff` object
- new `staffGrade` object linked to previous
- value of previous `staffGrade.gradeFinishDate` set equal to `gradeChangeDate`

Algorithmic Techniques

- Describe internal logic of a process or decision by breaking it down into small steps
- Can be constructed top-down to handle arbitrarily complex functionality
- Suitable where a decision can be made about the best method to use
- Examples:
  - Structured English
  - Activity Diagrams

Structured English

- Commonly used, easy to learn
- Three types of control structure, derived from structured programming:
  - Sequences of instructions
  - Selection of alternative instructions (or groups of instruction)
  - Iteration (repetition) of instructions (or groups)

Sequence in Structured English

- Each instruction is executed in turn, one after another:

```
get client contact name
sale cost = item cost * ( 1 - discount rate )
calculate total bonus
description = new description
```

Selection in Structured English

- One or other alternative course is followed, depending on result of a test:

```
if client contact is 'Sushila'
  set discount rate to 5%
else
  set discount rate to 2%
end if
```

Iteration in Structured English

- Instruction or block of instructions is repeated
  - Can be a set number of repeats
  - Or until some test is satisfied:

```
do while there are more staff in the list
  calculate staff bonus
  store bonus amount
end do
```

Activity Diagrams

- Part of UML notation set
- Can be used for operation logic specification, among many other uses
- Easy to learn and understand
- Some resemblance to old-fashioned flowchart technique

Example Activity Diagram: Check campaign budget