XII. Describing Business Rules

Business Rules
Structured English
Decision Tables
Decision Trees
The Object Constraint Language (OCL)

Business Rules

- Business rules are used to describe the properties of an application, e.g., the fact that an employee cannot earn more than his or her manager, or that every employee has a unique employee number.
- A business rule can be:
  - An integrity constraint on the data of the application, e.g., “each employee earns less than her manager”;
  - A derivation rule, whereby information can be derived from other information, e.g., “the price of a train ticket, in Canadian dollars, is given by the distance to be travelled in kilometers, multiplied by 0.2, multiplied by 1.5 for a first class ticket”.

Examples of Business Rules

Constraints
- (BR1) The manager of a department must belong to that department.
- (BR2) An employee cannot earn more than her manager.
- (BR3) A department of the Toronto office can only be managed by an employee who has ≥ 10 yrs experience.
- (BR4) An employee can only participate in projects associated with her department.

Derivations
- (BR5) The budget of a project is the sum of all salaries of participating employees, multiplied by 3.

Specifying Business Rules

- How do we specify business rules? We’ll be looking at several alternative notations.
  - Natural Language -- use unrestricted natural language...but such descriptions can be highly ambiguous
  - Structured English -- use a subset of a natural language (both syntactically and vocabulary-wise) to minimize ambiguities...this has been used with some success
  - Decision Tables -- use a table representation of alternative outcomes (similar to truth tables)
  - Decision Trees -- use a tree representation of alternative outcomes

Another Example

- Takes some effort to specify, not very readable, too close to an implementation

For each LOAN ACCOUNT NUMBER in the LOAN ACCOUNT FILE do the following steps:
If the AMOUNT PAST DUE is greater than $0.00 then while there are LOAN ACCOUNT NUMBERS for the CUSTOMER NAME do the following:
  - sum the OUTSTANDING LOAN BALANCES
  - sum the MINIMAL PAYMENTS
  - sum the PAST DUE AMOUNTS
report the CUSTOMER NAME, LOAN ACCOUNT on OVERDUE CUSTOMER, LOAN ANALYSIS

- Another Example

  - Takes some effort to specify, not very readable, too close to an implementation

end do
Some Rules for Structured English

- Use only nouns and terms defined in the project dictionary
- Avoid compound sentences because they can be highly ambiguous
- Avoid undefined adjectives and adverbs (such as “good”, “nice” etc.) unless clearly defined in the dictionary in terms of value ranges (e.g., “good” ↔ 65-75%)
- Avoid language that destroys the natural flow of control within the process (i.e., goto’s)
- Use a limited set of flow constructs, such as sequencing, if-then-else, while do etc.

Decision Tables

- If there are $n$ parameters (or, conditions) to a decision, each of which can take $k_1, k_2, \ldots, k_n$ values, then make up a table with $k_1 \times k_2 \times \ldots \times k_n$ columns and as many rows as there are possible actions (or, outcomes)
- For example:
  - “If the plane is more than half full and the flight costs more than $350 per seat, serve free cocktails, unless it is a domestic flight. Charge for cocktails in all domestic flights where cocktails are served, i.e., those that are more than half full.”

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Outcomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domestic</td>
<td>Serve cocktails</td>
</tr>
<tr>
<td>$\geq$ half full</td>
<td>$\geq$ $350/seat</td>
</tr>
<tr>
<td>YYNNN Y</td>
<td>YYNYNN Y</td>
</tr>
</tbody>
</table>

Some Rules for Structured English

1. Identify all conditions and all outcomes
2. Create the decision table, with one column for each possible combination of condition values and one row for every possible outcome
3. Fill in the table
4. Eliminate ambiguities, uncover cases, contradictions, redundancy

How to Construct Decision Tables

Completion and Simplification of a Decision Table

Going to a Place

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Y Y Y Y Y N N</td>
<td>Y Y Y N N N N N</td>
<td>Y N N N N N N</td>
<td>Y Y Y N N N</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Another Example

<table>
<thead>
<tr>
<th>Is budget likely to be overspent?</th>
<th>Is overspent likely to be over 2%?</th>
<th>No action!</th>
<th>Write letter!</th>
<th>Set up meeting!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y Y</td>
<td>N N</td>
<td>X X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Nodes of a decision tree represent partial outcomes, successors of a node represent mutually exclusive alternatives, leaves of a decision tree represent outcomes.

Getting Home?
- Short Trip?
  - Have Car?
    - Take Car
    - Walk
  - No Car?
    - Taxi
- In-Town Trip?
  - Have Car?
    - Take Car
    - TTC
  - Fly
- Out-of-Town Trip?
  - Have Car?
    - Take Car
  - Taxi

Clarifications
- Question: Is there a difference between freight shipping and handling?
  - Answer: No, all rates include freight and handling.
- Question: The description mentions “up to 20lbs” and “over 20lbs”. Which rate applies for exactly 20lbs?
  - Answer: It’s generally understood that “up to 20lbs” means “up to and including 20lbs”. We can’t spell out every little thing, you know?
- Question: The fourth sentence could be read in two ways: “both outside the local area and also over 20lbs”, or, alternatively, “outside the local area and, in addition, either over 20lbs or express not required”, Which is correct?
  - Answer: The second one. The first meaning couldn’t be right because you would end up charging the local express rate when express delivery was not required. I see your point though, it is a bit confusing...

The Freight Decision Tree
- Area
  - East of Miss.
  - West of Miss.
- Weight
  - ≤ 20lbs
  - > 20lbs
- Service
  - Express
  - Normal
- Method
  - Surface
  - Local area
  - Outside local area

Summary
- Decision trees are best used with applications involving up to 15-20 outcomes.
- Decision tables are more appropriate for problems involving complex combinations of up to 5-6 conditions (but can handle much larger number of outcomes).
- Structured English (and state-oriented models) are most appropriate for problems involving sequential considerations of alternative steps.

Pre- and Post-Conditions
- What conditions must be true before an operation can take place?
- What are the conditions that will be true after an operation is completed?
  - For example, consider
    - Campaign.assignStaff(creativeStaff.id)
      - Pre-condition: creativeStaff.object.id is not null;
      - Post-condition: a link is created between campaignObject and creativeStaff.object.
The Object Constraint Language

- Some constraints can be adequately expressed in the graphical language (e.g., multiplicity of an association).
- Some can not. For example, constraints within operation specifications (pre- and post-conditions).
- The Object Constraint Language (OCL) provides a formal language for specifying constraints which can supplement the models created in terms of UML diagrams.
- The language has a precise syntax that enables the construction of unambiguous statements.
- Each expression has an associated context, which is usually the class to which the expression is attached.

OCL Examples

Pre- and Post-Conditions in OCL

```oclasses
CreativeStaff::changeGrade(id:String, grade.id:String, gradeChangeDate:Date)
pre:
  self.id->notEmpty
  self.grade.id->notEmpty
  self.gradeChangeDate >= today (assumes no retroactive changes)
post:
  staffGrade[grade.id]->exists
  self.staffGrade->notEmpty
  self.staffGrade[grade.id].previousGrade->notEmpty
  self.staffGrade.gradeFinishDate = gradeChangeDate
```

What Does the Post-Condition Mean?

- CreativeStaff
- StaffGrade
- Grade

Additional Readings