OCL Tutorial

Feb 16, 2007
Prepared by Mazeiar Salehie
Resource

- This tutorial is based on the Arena management system discussed in the following book:
- The example is in Chapter 9 – Object design: specifying interfaces
- Some modifications have been applied to OCL expressions in the book to comply with OCL 2.0 specification.
- The examples are about constraints on classes, but the same expressions are valid for components except for some slight modifications for provided and required interfaces.
First Example

<table>
<thead>
<tr>
<th>Tournament</th>
</tr>
</thead>
<tbody>
<tr>
<td>- maxNumPlayers: int</td>
</tr>
<tr>
<td>+ getMaxNumPlayers(): int</td>
</tr>
<tr>
<td>+ getPlayers(): List</td>
</tr>
<tr>
<td>+ acceptPlayer(p:Player)</td>
</tr>
<tr>
<td>+ removePlayer(p:Player)</td>
</tr>
<tr>
<td>+ isPlayerAccepted(p:Player): boolean</td>
</tr>
</tbody>
</table>
public class Tournament {

/** The maximum number of players
 * is positive at all times.
 * @invariant maxNumPlayers > 0
 */
private int maxNumPlayers;

/** The players List contains
 * references to Players who
 * are registered with the
 * Tournament. */
private List players;

/** Returns the current number of
 * players in the tournament. */
public int getNumPlayers() {...}

/** Returns the maximum number of
 * players in the tournament. */
public int getMaxNumPlayers() {...}
OCL Expressions 1

• **context** Tournament::acceptPlayer(p)
  **pre**: not isPlayerAccepted(p)

• **context** Tournament::acceptPlayer(p)
  **pre**: getNumPlayers() < getMaxNumPlayers()

• **context** Tournament::acceptPlayer(p)
  **post**: isPlayerAccepted(p)

• **context** Tournament::acceptPlayer(p)
  **post**: getNumPlayers() = getNumPlayers()@pre + 1
OCL Expressions 2

• **context** Tournament::removePlayer(p)
  **pre**: isPlayerAccepted(p)

• **context** Tournament::removePlayer(p)
  **post**: not isPlayerAccepted(p)

• **context** Tournament::removePlayer(p)
  **post**: getNumPlayers() = getNumPlayers()@pre - 1
Second Example

League
- +start:Date
- +end:Date
- +getActivePlayers()

{ordered}

Tournaments

Tournament
- +start:Date
- +end:Date
- +acceptPlayer(p:Player)

Players

Player
- +name:String
- +email:String
Informal Constraints

• A Tournament’s planned duration must be under one week.
• Players can be accepted in a Tournament only if they are already registered with the corresponding League.
• The number of active Players in a League are those that have taken part in at least one Tournament of the League.
Three Basic Types of Navigations

1. Local attribute

2. Directly related class

3. Indirectly related class

- Any OCL constraints can be built using a combination of these types
An Instance Problem Space

2 Leagues, 2 Tournaments and 5 Players

Winter 2007
OCL Expressions

- **Local attribute navigation**
  context Tournament
  inv: end - start <= Calendar.WEEK

- **Directly related class navigation**
  context Tournament::acceptPlayer(p)
  pre: league.players->includes(p)

- **Indirectly related class navigation**
  context League::getActivePlayers
  post: result =
    tournaments.players->asSet()
Third Example

TournamentControl

+applyForTournament()

TournamentForm

+selectSponsors(advertisers):List
+advertizeTournament()
+acceptPlayer(p)
+announceTournament()
+isPlayerOverbooked():boolean

Tournament

+name: String
+start: Date
+end: Date
+acceptPlayer(p)
+removePlayer(p)
+schedule()

Player

+matches

Match

+start: Date
+status: MatchStatus
+playMove(p, m)
+getScore(): Map

Advertiser

+sponsors

matches

Player

matches
OCL Expressions 1

• **OCL forall quantifier**
  
  /* All Matches in a Tournament occur within the Tournament’s time frame */

```ocm
context Tournament
  inv:
  matches->forAll(m:Match | m.start.after(t.start) and m.end.before(t.end))
```

Winter 2007
OCL Expressions 2

- **OCL exists quantifier**

  /* Each Tournament conducts at least one Match on the first day of the Tournament */

  ```
  context Tournament
  inv:
    matches->exists(m:Match | m.start.equals(start))
  ```
OCL Expressions 3

• **context**
  TournamentControl::selectSponsors(advertisers)
  **pre**: interestedSponsors->notEmpty and tournament.sponsors->isEmpty

• **context**
  TournamentControl::selectSponsors(advertisers)
  **post**: tournament.sponsors.equals(advertisers)
OCL Expressions 4

• **context** TournamentControl::advertiseTournament()
  **pre**: tournament.sponsors->isEmpty and not tournament.advertised

• **context** TournamentControl::advertiseTournament()
  **post**: tournament.advertised

• **context** TournamentControl::acceptPlayer(p)
  **pre**: tournament.advertised and interestedPlayers->includes(p) and not isPlayerOverbooked(p)

• **context** TournamentControl::acceptPlayer(p)
  **post**: tournament.players->includes(p)
OCL Expressions 5

• All Matches of in a Tournament must occur within the time frame of the Tournament

context Tournament
inv: matches --> forAll(m | m.start.after(start) and m.start.before(end))
Forth Example

- Dealing with a loop in class diagram
OCL Expressions 1

• No Player can take part in two or more Tournaments that overlap

context TournamentControl
inv: tournament.players->forAll(p|
   p.tournaments->forAll(t|
      t <> tournament implies
      not t.overlap(tournament))))
OCL Expressions 2

• A match can only involve players who are accepted in the tournament

```
context Match
inv: players->forAll(p|
    p.tournaments->exists(t|
        t.matches->includes(self)))
```

• Can we simplify the above expression as:

```
context Match
inv: players.tournaments.matches->includes(self)
```

• It seems so. But, the answer is no! Can you tell why?