OCL Workshop, September 2012

Experiences using OCL for Business Rules on Financial Data

Author: David Garry

Contact: tricia.balfe@nomos-software.com
david.garry@nomos-software.com
Introduction

• Have built own OCL implementation, for executing OCL over XML
  – Generate OCL to Java

• Execute rules on XML messaging
  – focus on finance

• Have encountered three common requirements
Three Problems

• Identifying exact error locations

• Supporting additional data types with operations

• Checking against external data sources
Identifying Error Locations

• Commercial usage:
  – detect issues in messaging, and provide troubleshooting report to originators of the messaging
  – report must be comprehensive; originators have no debugging capability
  – we replace a process whereby message originators must ensure they comply with paper-based messaging guidelines

• Problem:
  – Need to report exact error location / cause of error
  – Depending on the OCL expression, this can be difficult
  – We do not write the OCL and can’t control expressions used
When a failure occurs, what caused the failure?

```
context OriginalGroupInformation20

StsRsnInf->forAll(a |
  (a.Orgtr.Nm->size() = 1 or
   (a.Orgtr.Id->size() = 1 and a.Orgtr.Id.OrgId.BICOOrBEI->size() = 1))
  and a.Orgtr.PstlAdr->size() = 0
  and a.Orgtr.CtryOfRes->size() = 0)
```
Some Options

• Rewrite the rule
  – Split into a number of rules;
    • Often a good option, but not in all cases. In the example, the rule is a discrete rule from a standards specification, so don’t want to split
  – Follow a more suitable OCL pattern; See IEEE paper “Testing-oriented improvements of OCL specification patterns”
    • Likely to make the OCL authoring more difficult. May not solve every case

• Our solution
  – Trigger OCL queries on failed contexts
  – Author of the OCL constraint also authors the queries, and decides what is the most useful information to return
Query Rules

OCL Constraint

\[\text{context } \text{OriginalGroupInformation20}\]
\[\text{StsRsnInf->forAll}(a \mid (a.\text{Orgtr}.Nm->\text{size()} = 1 \text{ or } (a.\text{Orgtr}.Id->\text{size()} = 1 \text{ and } a.\text{Orgtr}.Id.OrgId.BICOrBEI->\text{size()} = 1))\]
\[\text{and } a.\text{Orgtr}.PstlAdr->\text{size()} = 0 \text{ and } a.\text{Orgtr}.CtryOfRes->\text{size()} = 0)\]

Returns any postal address and country of residence information included
- Should also include queries to return names and ids

Triggered OCL Queries

\[\text{Context } \text{OriginalGroupInformation20::getPstlAdr()} : \text{Set(PostalAddress6)}\]
\[\text{body: } \text{self.Orgtr.PstlAdr->asSet()}\]

\[\text{Context } \text{OriginalGroupInformation20::getCtryOfRes()} : \text{Set(CountryCode)}\]
\[\text{body: } \text{self.Orgtr.CtryOfRes->asSet()}\]
Additional Data Types and Operations

- **Commercial usage**
  - OCL authors work with domain-specific models e.g. FpML, or ISO20022 payments
  - Need to be able to write commonly occurring rules for these domains easily

- **Examples:**
  - For FpML (messaging for derivatives trading) need rules on dates (and timezones)
    - Need support for date types, with operations like after, before etc
  - For ISO20022, an IBAN type is included (unique bank id)
    - Need support for checking structure of the IBAN (checksums etc)

- **Extensions to OCL for domain-specific models must be ‘easy’/'lightweight’**
  - Date / time operations can be made available to all
  - But operations such as checking IBAN should only be available for the ISO20022 models
Additional Data Types and Operations

• We do code generation…

• Rather than defining operations using OCL expressions

• We make the operations available on the types
  – And ensure there are implementations in our code generation target (java)
context EarlyTerminationEvent

adjustedExerciseDate.before(adjustedEarlyTermDate)
Example of isValidIBAN Operation

```
context CashAccount16
self.Id.IBAN.isValidIBAN()
```

```xml
<xs:simpleType name="IBAN2007Identifier">
  <xs:restriction base="xs:string">
    <xs:pattern value="[A-Z]{2,2}[0-9]{2,2}[a-zA-Z0-9]{1,30}"/>
  </xs:restriction>
</xs:simpleType>
```
Referencing External Data Sources

- Commercial Usage
  - Frequently need to check data against external data sources
    - Codelists managed by external organisations
    - Data in customer databases

- Need mechanism to allow customers to write OCL expressions that include these types of checks

- Needs to be easily configurable
  - Customer needs to be able to define which types of checks they want to make available
  - Depending on the runtime environment, the external data source may change
    - e.g. one database instance in a test environment, another in the production environment
Referencing External Data Sources

• We provide a mechanism to allow customers to configure functions on the types we support

• And they provide the runtime implementation in java
Example of Operation to Check Codelists

```
context OriginalGroupInformation20
self.OrgnlMsgId.isExternalFinInstIdCde()
```
Conclusion

• We have outlined some requirements that we have needed to address

• We try to find practical ways to solve the problems
  – Taking account of user needs
  – And implement quickly, simply and flexibly

• Purpose on presenting here:
  – generate discussion on how to provide a common approach to solving these problems for OCL users
## Nomos at a Glance

### Mission: Fast Deployment of Business Rules

### Company

- Private company, software vendor, founded 2007, Ireland
- Tricia Balfe, founder and CEO
- David Garry, founder and CTO
- Chris Horn, non-executive director, formerly of Iona Technologies (CORBA)

### Partnerships

- NoMagic

### Memberships

- OMG
- tmforum

### Investors

- Kernel Capital
- Enterprise Ireland