DeepTelos for ConceptBase: A contribution to the MULTI process challenge

Manfred Jeusfeld
University of Skövde
manfred.jeusfeld@his.se

Outline

1)Telos and DeepTelos

2)Implementation fo selected requirements

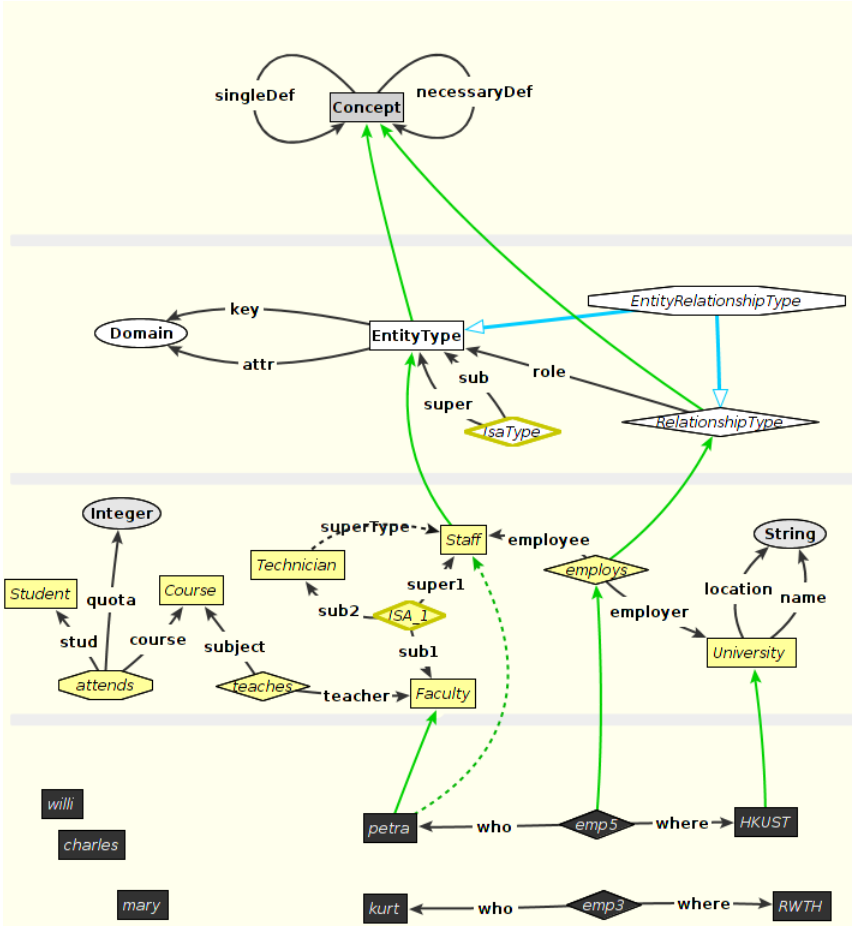
3)Discussion

4)Lessons learned

Telos and DeepTelos

- Telos (Mylopoulos et al 1990) is based on the requirements modeling language RML (Stanley 1986).
- Served a second purpose in the EU DAIDA project in thge late 1980s: to represent artifacts from all stages of software development, in particular database applications
- Uses a single ”proposition” predicate P(id,x,m,y) to represent any explicit information
- Instantiation, specialization and attribution/relationships are defined by logical axioms
- ConceptBase implements O-Telos via a Datalog-neg engine. some extra-Datalog features were added: arithmetic, recursive functions, active rules
- O-Telos is defined by 30 axioms, see http://conceptbase.sourceforge.net/userManual81/cbm011.html

Traditional use of Telos: Metamodeling



(M3)

(M2)

(M1)

(M0)

Telos supports an unlimited instantiation hierarchy.

In addition each object (=proposition) is instance of the *omega class* Proposition.

Note that all levels have instances here that are user-defined.

No axiom forbids relationships across these levels. Levels in (O-)Telos are derived from the instantiation hierarchy.

http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/d3622851/erdcomplete.gel

DeepTelos2

Original version: see http://conceptbase.sourceforge.net/deeptelos/

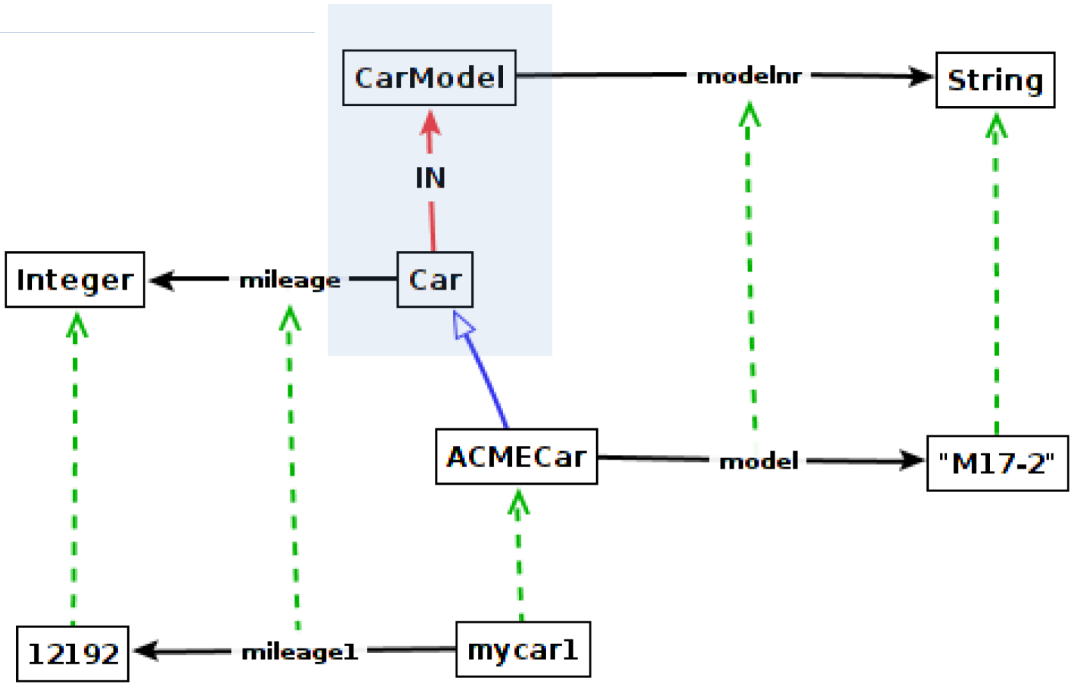
We use here DeepTelos, Revision 2, which provides a better interplay between the Telos specialization and DeepTelos. In particular, one can define explicit sub-class hierarchies under the ”most-general instances”.

Definition of DeepTelos2 in ConceptBase

```
DeepTelosRules in Class with
   rule
    mrule1 : $ forall m,x,c/Proposition
       (x in c) and (m IN c) and not (x isA m) ==> (x ISA m) $;
    mrule2 : $ forall x,c,d/Proposition (c ISA d) and (x in c)  ==> (x in d) $;
    mrule3 : $ forall c,d,m,n/Proposition
       (m IN c) and (n IN d) and (c ISA d)  ==> (m ISA n) $;
    mrule4 : $ forall m,x,c/Proposition
       (m IN c) and (x isA m) and not (x in QueryClass) ==> (x in c) $;
    mrule5 : $ forall m,mx,x,c/Proposition
       (m IN c) and :(x isA mx): and (mx ISA m) and not (x in QueryClass) ==> (x in c) $
  constraint
    mconstr1 : $ forall x,m,c/Proposition (m IN c) and (x in c) ==> not (x in m) $
end
```

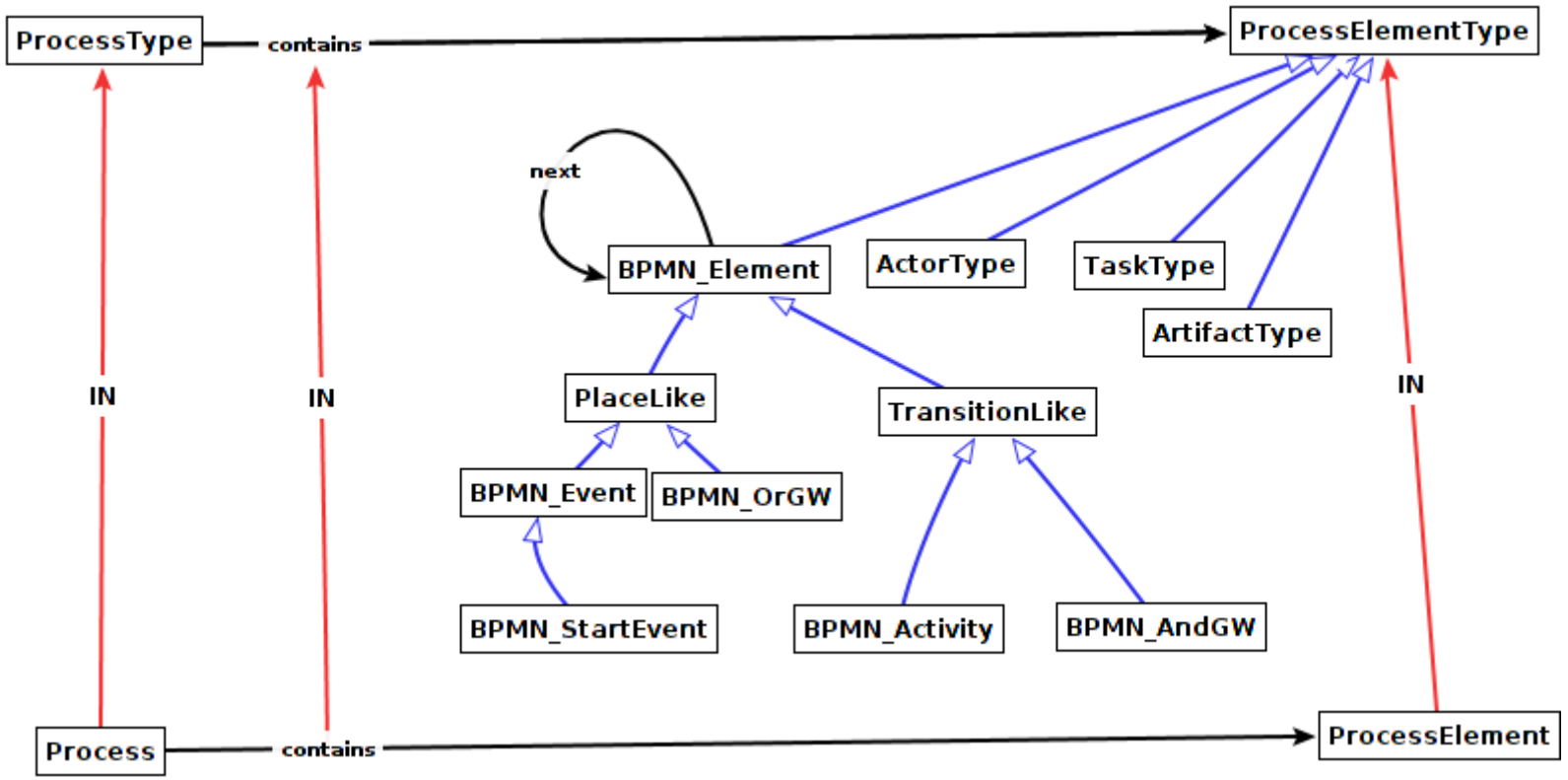
The basic idea of DeepTelos

Regard this as a pair: ”Car” is the proxy of ”CarModel” at one abstraction level lower: ”most-general instance”



http://conceptbase.sourceforge.net/multi2019challenge/carexample.gel

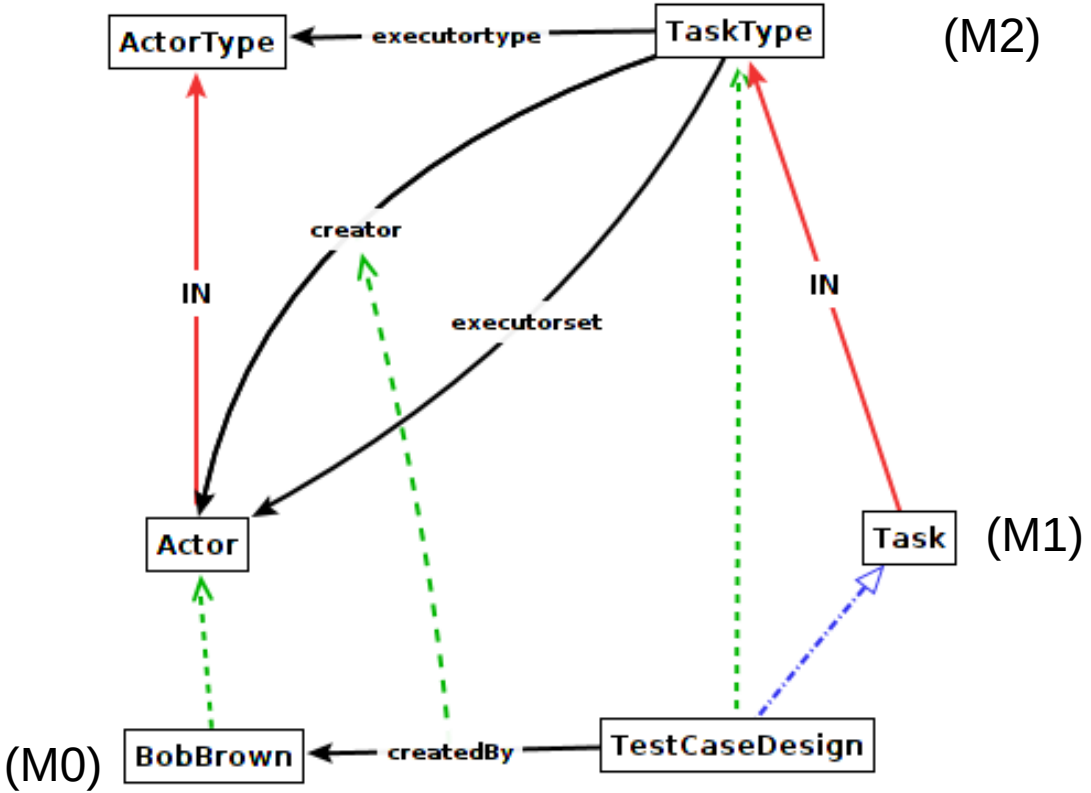
Use for process modeling



We re-use an existing Telos definition of BPMN-core

http://conceptbase.sourceforge.net/multi2019challenge/p1-p3.gel

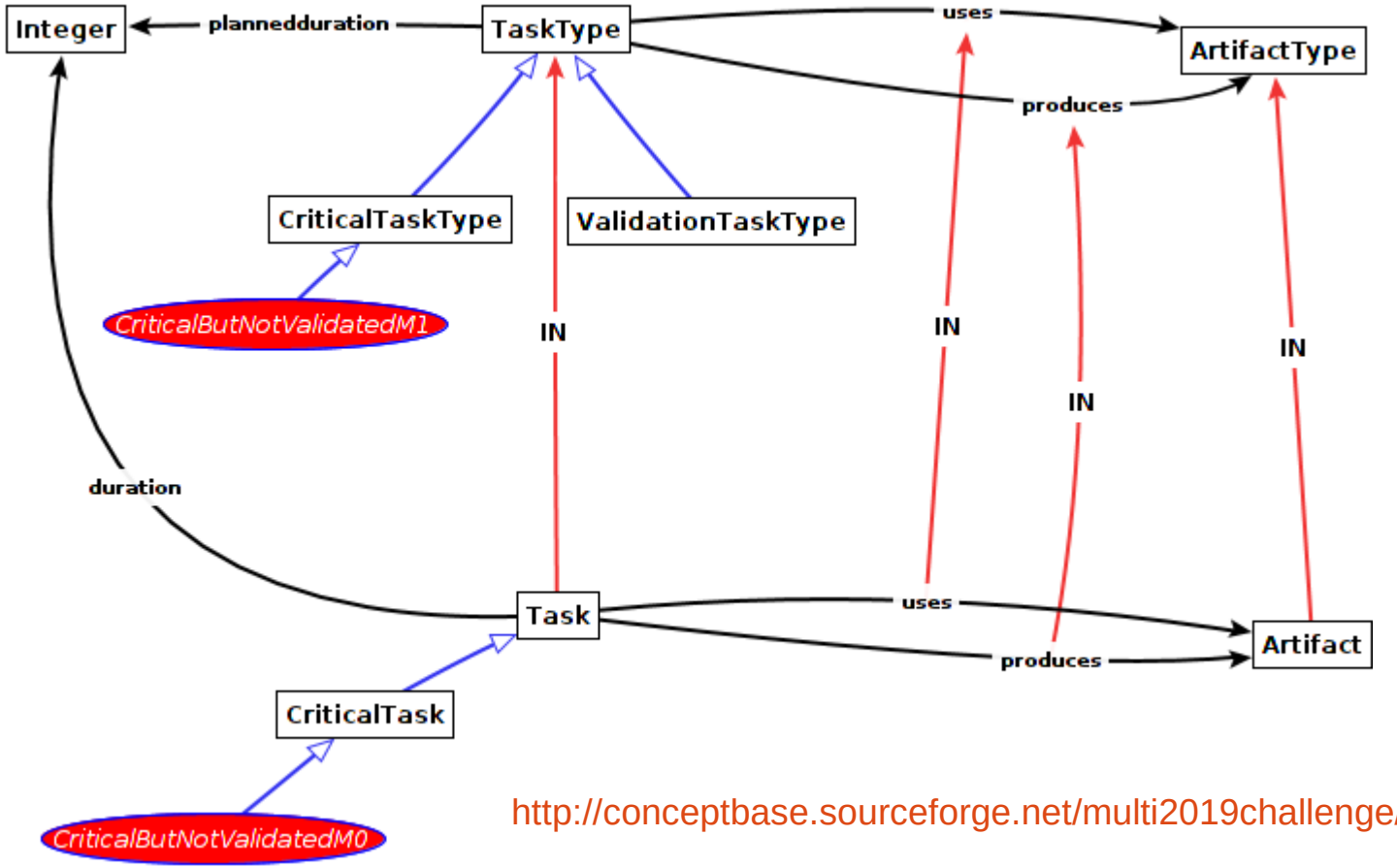
Executors, Cross-level links



The relations 'creator' and 'executorset' cross the UML-ish levels; so do their instances such as 'createdBy'

http://conceptbase.sourceforge.net/multi2019challenge/p4-p6.gel

Input/Output Artifacts



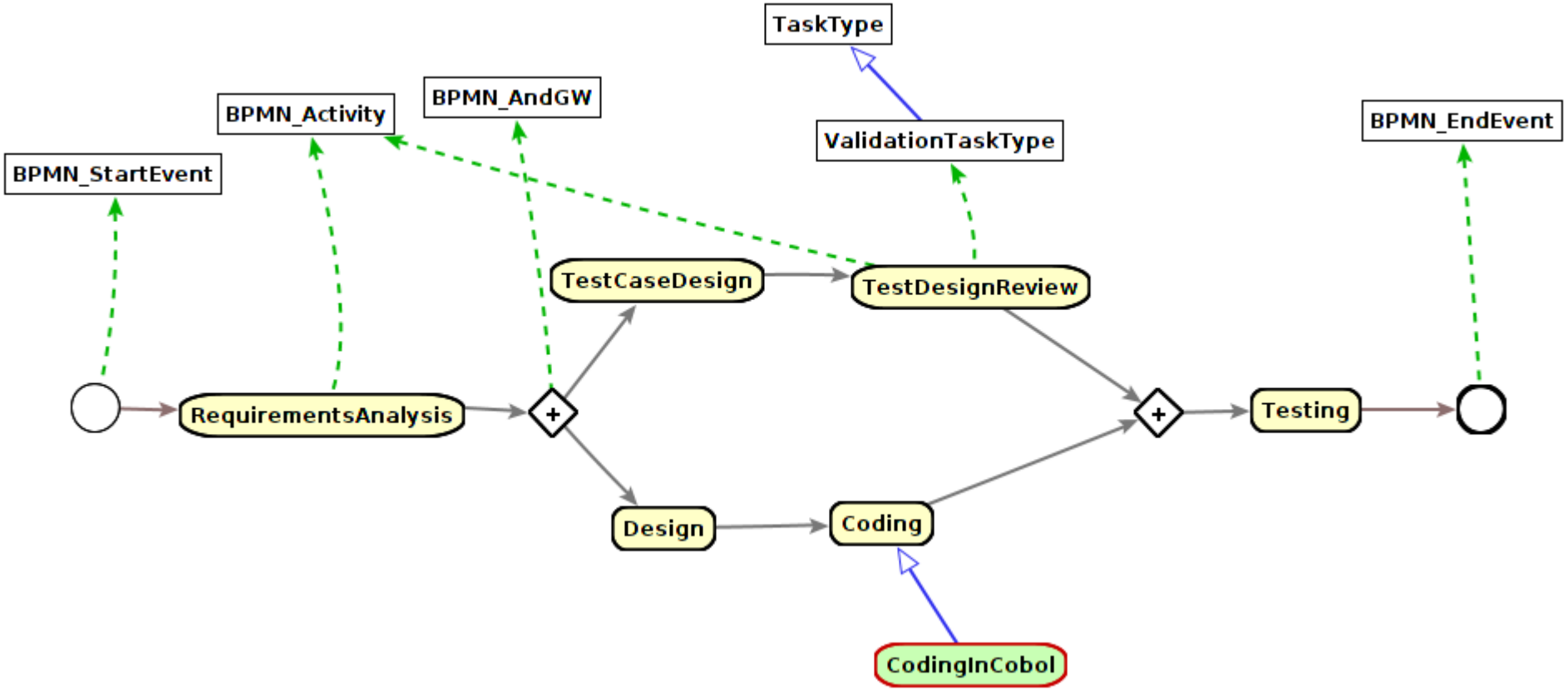
http://conceptbase.sourceforge.net/multi2019challenge/p7-p9.gel

Queries

```
UnmatchedTask in QueryClass isA ProperTask with
   computed_attribute
    T : ProperTaskType;
    PT : ProperProcessType;
    P : ProperProcess
  constraint
    cm : $
      (P contains this) and
      :(this in T): and (P in PT) and
      (PT <> Process) and (T <> Task) and not (PT contains T) $
end
```

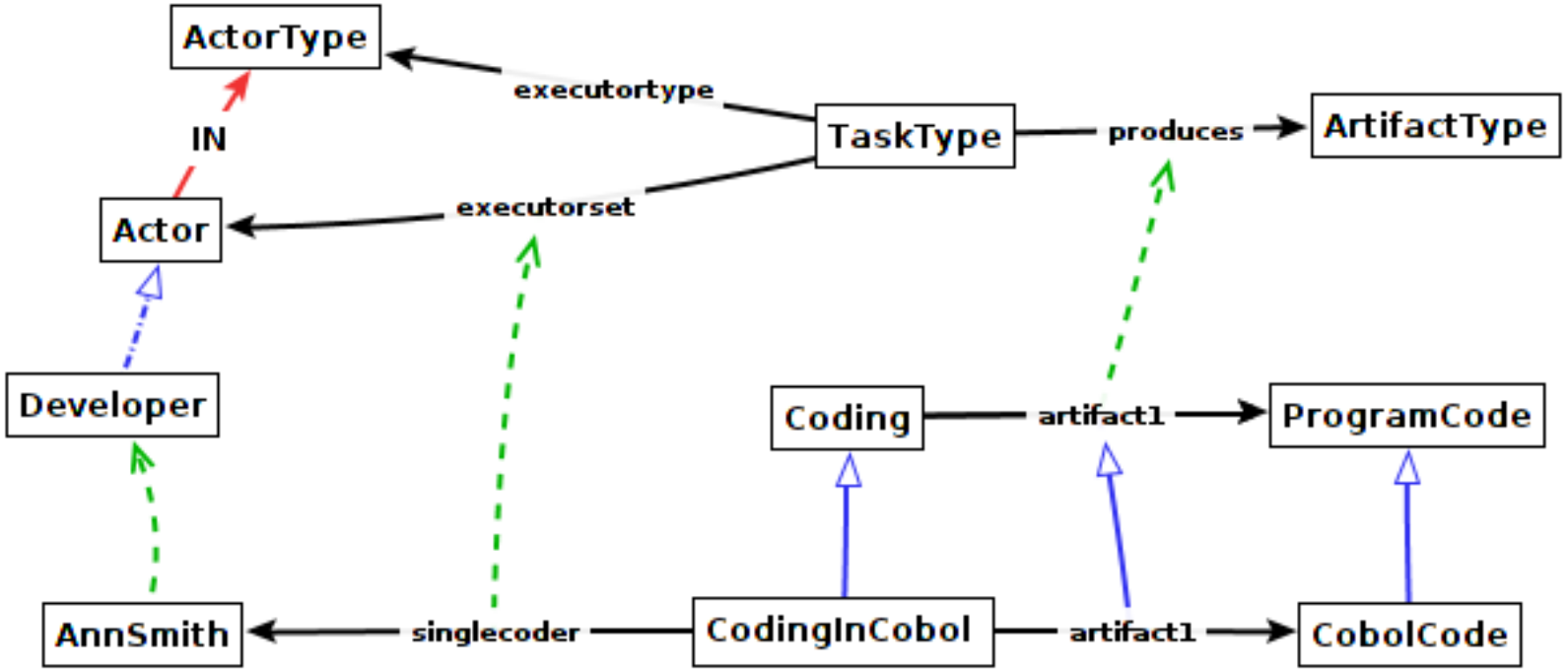
The processes shall only uses the tasks defined in their ProcessType

The ACME Process Example



http://conceptbase.sourceforge.net/multi2019challenge/acme-process.gel

CodingInCobol as subclass



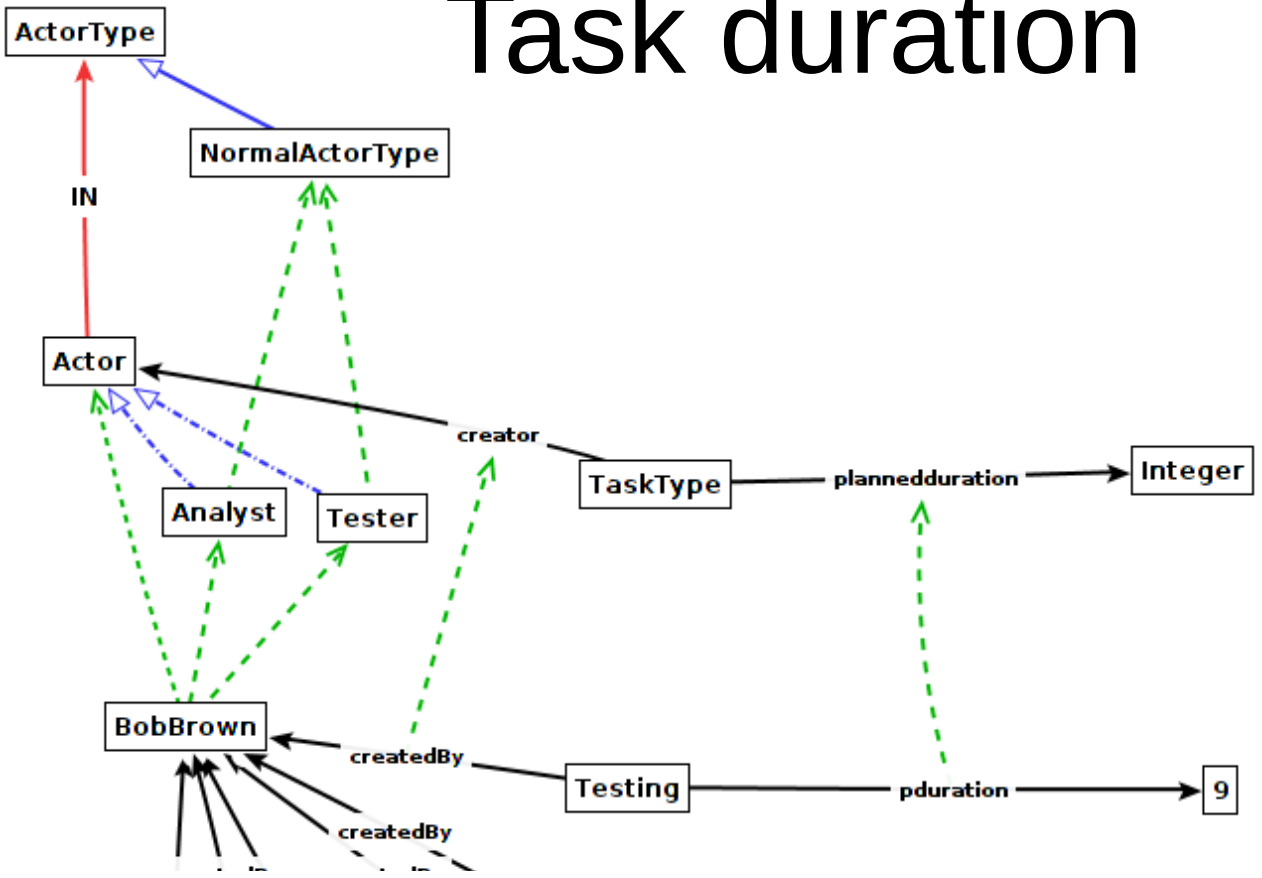
http://conceptbase.sourceforge.net/multi2019challenge/s5-s7.gel

UseCobol constraint

```
CodingInCobol in Class,BPMN_Activity isA Coding with
   constraint
    useCobol :
      $ forall cic/CodingInCobol (cic useslanguage Cobol) $
end
```

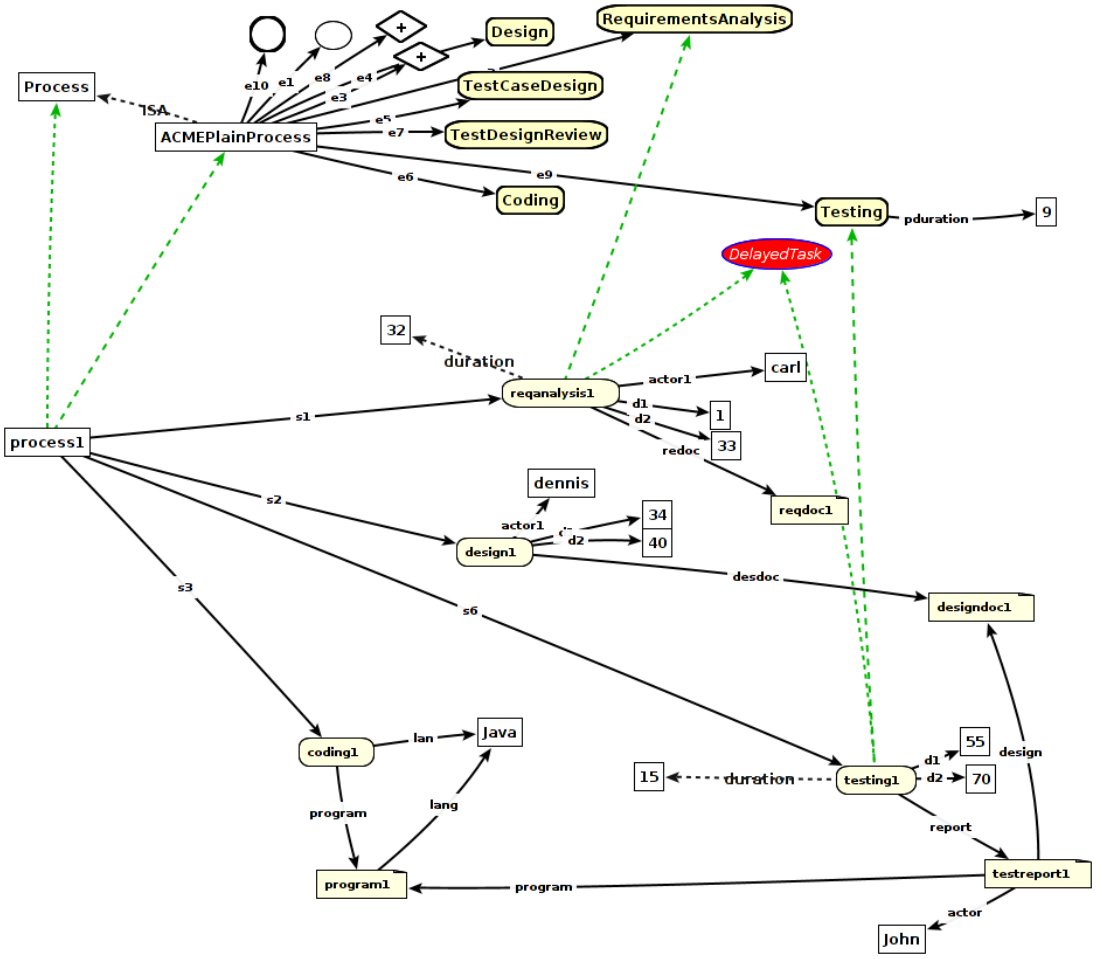
Example on how to use the constraint language in ConceptBase

Task duration



http://conceptbase.sourceforge.net/multi2019challenge/s11-s12.png

ACME Plain Process



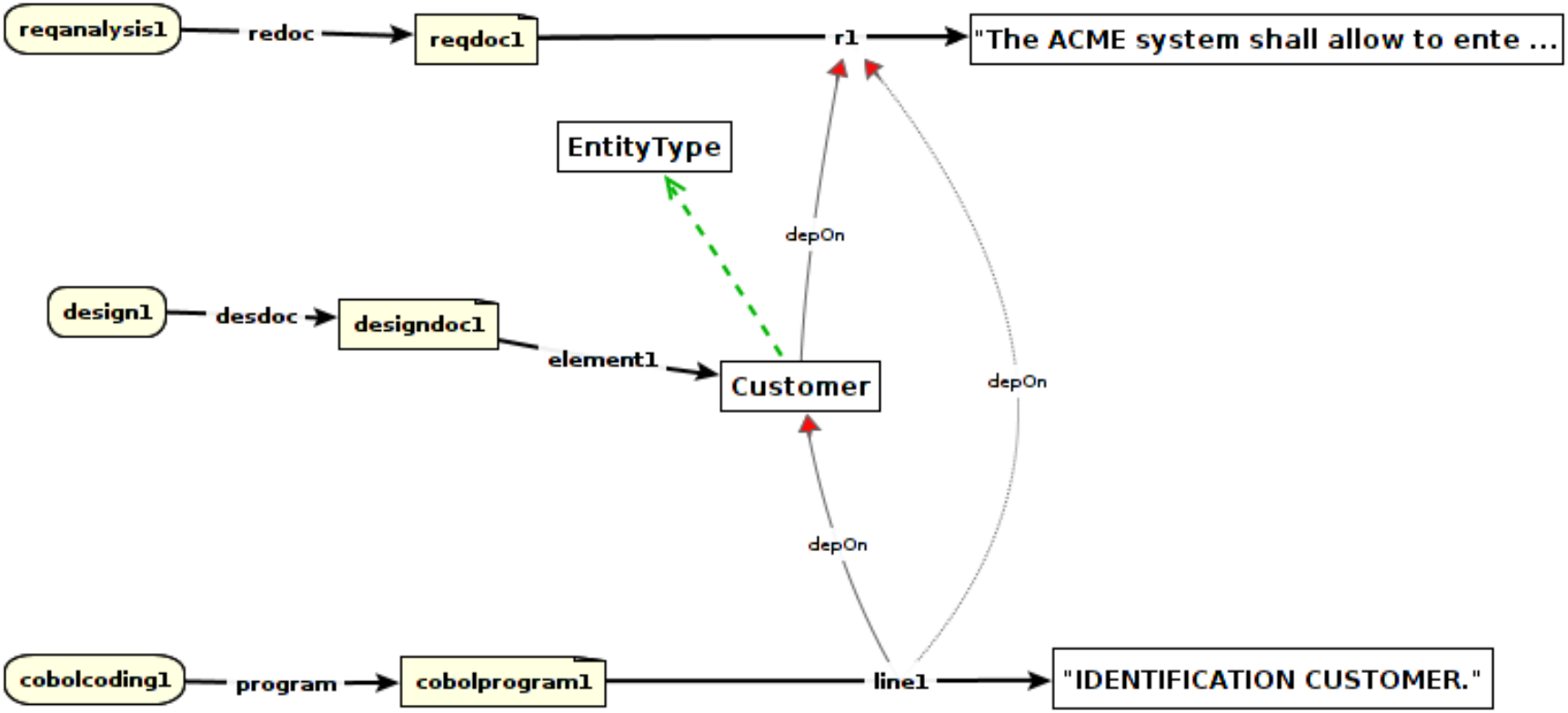
http://conceptbase.sourceforge.net/multi2019challenge/process1.gel

DelayedTask

```
Task with
   rule
    durrule : $ forall t/Task d/Integer
       (d = taskDuration(t)) ==> (t duration d) $
end

DelayedTask in QueryClass isA Task with
   constraint
    isDelayed : $ exists T/TaskType pd,d/Integer
                  (this in T) and (T plannedduration pd) and
                  (this duration d) and (d > pd) $
end
```

Tracing dependencies



http://conceptbase.sourceforge.net/multi2019challenge/process3.png

Formalization of dependencies

```
Proposition with
  attribute,transitive
    dependsOn: Proposition
end

ArtifactType in Class with
  rule
    deprule1: $ forall a1,a2/ArtifactType tt/TaskType
                    (tt uses a1) and (tt produces a2) and (tt <> Task) ==> (a2
dependsOn a1) $
end

Artifact in Class with
  rule
    deprule2: $ forall a1,a2/Artifact t/Task
                    (t uses a1) and (t produces a2) ==> (a2 dependsOn a1) $
end
```

Discussion

- Two basic modeling constructs IN+ISA defined with six axioms
- Re-use the Telos axioms on instantiation, specialization and attribution/relationships
- Datalog-neg semantics (closed world assumption) for the constructs
- Re-use BPMN-core
- Crucial was the extensibility of the omega level (Proposition)
- Added features: represent M0 level (process traces), define delayed tasks
- No explicit levels but level could be computed by queries based in the lattice of IN-relations
- All requirements were implemented; the ”authorization” part is a bit weak though

Summary

- DeepTelos could fully realize the MULTI 2019 process challenge
- During the course, some limitations of the multi-level formula compiler of ConceptBase were detected: order of definitions play a greater role than anticipated)
- We also found that query classes must be excluded from the DeepTelos axioms (otherwise, stratification errors can occur or even seemingly endless loops)
- The module concept of ConceptBase helped greatly in indentifying re-usable model parts
- Re-use (of meta models) may be the greater benefit of multi-level modeling