SemCheck - Script for the Case Studies

Manfred A. Jeusfeld
University of Skövde, Sweden
manfred.jeusfeld@acm.org
February 2017
last change: 2017-06-04

Contents

This work was supported in part by the Erasmus+ project Open Model Initiative (OMI).
Project 2014-1-AT01-KA203-000942 (OeAD Austria)

1. Introduction and context

The Erasmus+ OMI initiative aims to facilitate academic education in enterprise modeling by create a shared and open modeling environment, in which various domain-specific modeling languages (DSML) are being integrated. The DSML form modeling perspectives, e.g. for data modeling, process modeling, goal modeling, and so forth.

This document describes a tool that is designed to support the open modeling environment by providing semantic integrity checking services (SemCheck [1]), which can be used to analyze interrelated models formulated to find errors and to extract information from the models.

Homepage of SemCheck: http://austria.omilab.org/psm/content/semcheck/info

We describe first the software that is needed to use the SemCheck services. Then, a first case study shows how the constructs of the ArchiMate [5] enterprise architecture language can be supported, in particular to implement traceability services for ArchiMate implementations. The next chapter presents services for the 4EM [6] suite of enterprise modeling languages. The focus here is to support integrity and query services for cross-notational links.

The last chapter reports on the current state of integration of SemCheck into the open ADOxx modeling toolkit.

The examples source files referred to by this document come with the own Creative Commons licenses. They allow in any case the non-commercial use, in particular the use in academic context.

2. Software installation

You need a computer with one of the following operation systems:
- Linux, e.g. Ubuntu 16.04 (needed in particular for the ADOxx case study)
- Windows XP or higher (Windows 7 and 10 are fine)
- Apple MacOS

Further you need a Java SE Runtime Environment (JRE) 7 (recommended) or higher. If not already installed on your computer, then download and install it following the instructions at http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html

Note that Java comes with its own license conditions.

2.1 Download and install ConceptBase.cc

Download from http://conceptbase.sourceforge.net/CB-Download.html

ConceptBase (short for ConceptBase.cc) comes under a free and open software license allowing private, academic and commercial use: http://conceptbase.sourceforge.net/CB-FreeBSD-License.txt

Latest stable version for Linux, Windows, and Mac OS-X

- Read the ConceptBase.cc Copyright. Only proceed to the download step if you agree.
- Download CBinstaller.jar from Sourceforge. It contains the latest ConceptBase binaries plus the installation program.
- Open a command window (terminal), change to the directory where you downloaded CBinstaller.jar.
- Execute in the command window: `java -jar CBinstaller.jar`. Under Windows, you can also directly execute CBinstaller.jar from the web browser.
- The CBinstaller window shall pop up. If the button "Install" is activated (green color), then just press it to install ConceptBase. You can also adapt the installation directory before pressing "Install" but we recommend to use the default.

If the "Install" button is not activated, then Download/Select buttons are activated to download and select an installation archive. After download, you can continue with "Install".

If the above download links fail, then download CBinstaller.jar from the CB-Forum.

Some behavioral anti-virus tools could block CBinstaller from installing ConceptBase on your computer. You need to temporarily disable these tools or install ConceptBase manually as described in the installation guide.

The sample source files referred by this document are either downloadable via the URLs provided in this document or as tool download “Modelling Toolkit - SemCheck-Demo” at
http://austria.omilab.org/psm/content/semcheck/download?view=download
that has all material in a ZIP archive.

Starting ConceptBase

Linux, Mac OS-X: Add the ConceptBase installation directory to your search path, e.g.

```
export CB_HOME=$HOME/conceptbase
export PATH=$CB_HOME:$PATH
```

These two lines should also be added to your init file of your Unix shell, for example `.bashrc` if you are using bash. Start the ConceptBase user interface CBIva via the command

```
cbiva
```

in a command window. Then start the ConceptBase.cc server via the menu item File/Start CBserver. You can also start the CBserver in another command window via the command 'cbserver [options]'. Consult the user manual for instructions on how to interact with the system. The tutorial on metamodeling is a good starting exercise as well.

Windows: You can start the ConceptBase user interface CBIva by double-clicking the batch file

```
cbiva.bat
```

in the ConceptBase installation directory (usually c:\conceptbase). You can also use a Windows command window or PowerShell to start it. It will automatically connect the user interface to a public ConceptBase server. **Add the path ' c:\conceptbase' to your Windows environment variable PATH**. You can find instructions for example in https://javatutorial.net/set-java-home-windows-10

Associate CBGraph to graph files in your Web browser

A graph file (a.k.a. GEL file) stores a graph displaying some Telos model. The Telos models are included in the graph file as well and will be passed to the ConceptBase server (CBserver) when you open the GEL file with CBGraph.

- Use your Web browser (e.g. Firefox) to click on the link telos.gel. The browser shall ask you whether to download the file or associate an application. Choose to associate an application. Then, browse to the installation directory of ConceptBase.cc. Type for Windows "cbgraph.bat" and for other platforms "cbgraph" in the filename field and press OK. Afterwards, a click on a link to a graph file shall directly start CBGraph. Our web servers of ConceptBase use as Mime type for graph files application/cbgraphfile or application/x-cbgraphfile. The latter is preferred.

Windows 10: Follow instructions at http://conceptbase.sourceforge.net/CB-WinLinux.html to enable starting a local CBserver under Windows 10.

Older Windows, Mac OS-X: We currently do not ship binaries of the CBserver for older Windows and Mac OS-X. However, ConceptBase.cc is fully functional on these platforms by connecting to a public CBserver hosted at University of Skövde. This public CBserver is pre-configured to be used for Windows and Mac OS-X installations. You can also configure your ConceptBase installation to use another public CBserver, e.g. installed and running on a Linux server in your local network.

3. Case 1: ArchiMate

ArchiMate is an open enterprise architecture modeling language published as a standard from The Open Group. The current version is V2.1 but we primarily focus here on Version 1.0.

https://www2.opengroup.org/ogsys/catalog/c091

Goal of the case study: Provide traceability services for ArchiMate. The services shall allow to trace elements from the business layer down to the technology layer and vice versa.

Approach: Represent the ArchiMate constructs in a meta model (here: directly in ConceptBase) and defines deductive rules that realize the traceability services.

Acknowledgements: This case study is based on the group work by Sander van Arendonk, Niels Colijn, Dirk Janssen, Jeffrey Kramer in the Method Engineering course 2008 at Tilburg University. Sample code used with permission under a CC BY-NC 3.0 license.

3.1 The Meta Model of ArchiMate in ConceptBase

The main goal of this case study is to support traceability between any ArchiMate concepts at any of the three ArchiMate layers (business, application, technology). Hence the meta model both has to represent the ArchiMate constructs plus the provisions for traceability.

The complete specification is at

http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/d3827407/Pt1-NDL-and-NL.sml.txt

Consider the following parts of the file for understanding the specification:

```
Layer end
BusinessLayer in Layer end
ApplicationLayer in Layer end
TechnologyLayer in Layer end

Aspect end
InformationAspect in Aspect end
BehaviourAspect in Aspect end
StructureAspect in Aspect end
```

The three layer meta classes BusinessLayer, ApplicationLayer, and TechnologyLayer form a matrix with the three aspect meta classes InformationAspect, BehaviourAspect, and StructureAspect. ArchiMate model concept are typically classified into both one of the layer and one of the aspects.

Next, consider the generic meta class AnyNode:

```
AnyNode with
  transitive
    dependsOn: AnyNode
end
```

It defines a relation 'dependsOn' to be transitive. Transitivity is implemented by a deductive rule defined in the same meta model:

```
forall x,z,y,M/VAR AC/Proposition!transitive C/Proposition
        P(AC,C,M,C) and (x in C) and (y in C) and (z in C) and
        (x M y) and (y M z) ==> (x M z)
```

The ArchiMate concepts are then defined in ConceptBase based on the ArchiMate meta model:

```
AM_Product in AM_GenericNode, InformationAspect, BusinessLayer isA AnyNode with
  aggregation
    aggregatesBService: AM_BusService;
    aggregatesAService: AM_AppService;
    aggregatesIService: AM_InfService;
    aggregatesContract: AM_Contract
end
```

Note that the constructs are instantiated both into an aspect (here: InformationAspect) and into a layer (here: BusinessLayer). Further, they are defined as subclasses of AnyNode. This makes their instances (here: any ArchiMate Product) also an instance of AnyNode. They are then subject to have dependencies to other ArchiMate constructs. The links between ArchiMate model concepts induce depencies. For example, if a product aggregates a BusinessService, then it dependes on that service. We model this by stating that any such link is also a dependency:

```
AM_Product!aggregatesBService isA AnyNode!dependsOn end
```

ArchiMate has some dedicated link types to relate concepts from different layers. One of them is the 'realises' link. We use a deductive rule to map such links to dependencies:

```
forall b/AM_Behaviour s/AM_BusService (b realises s) ==> (s dependsOn b)
forall i/AM_BusInteraction s/AM_BusService (i realises s) ==> (s dependsOn i)
```

The ConceptBase representation has a relative large number of such rules.

The derived dependency links can be used in queries to extract information from an enterprise architecture model. For example consider the following query:

```
AM_Service3 in GenericQueryClass isA AM_AppService with
  constraint
    c1: $ exists os1,os2/AM_OperatingSystem (~this dependsOn os1) and
           (~this dependsOn os2) and (os1 \= os2) $
end
```

The query returns all application services that depend on two different operating systems. This query relies on the transitivity of 'dependsOn': The two operating systems os1 and os2 are part of the infrastructure layer. Some server nodes may depend on them. Further, system software like a DBMS may depend on the server node. Finally, the returned application services (application layer) indirectly depend on the DBMS.

More example analysis queries can be found at

http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/d3827411/Pt2-Analysis.sml.txt

3.2 Textual analysis of the ArchiMate model

It is assumed that you downloaded and installed ConceptBase.cc. Also make sure that the installation directory of ConceptBase is in the search path (see section 2: Starting ConceptBase).

Step 1: Download the the file ARCHIMATE.zip from http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/3822537 and extract it into a directory 'ARCHIMATE' on your local computer. It contails the following files:

```
Pt1-NDL-and-NL.sml.txt (ArchiMate notation)
Pt2-Analysis.sml.txt   (example queries)
Pt3-Example.sml.txt    (ArchiSurance example enterprise model)
Pt4-GraphTypes.sml.txt (graphical types for graphical analysis)
runall.cbs             (CBShell script for textual analysis)
archisurance.gel       (graph file for graphical analysis)
```

The files *.sml.txt are the Telos source files for the ArchiMate case study. The CBShell file runall.cbs shall pass these source files to a ConceptBase server and run some sample analysis queries on them. The file 'archisurance.gel' shall be used later for a graphical analysis.

Step 2: Open a command window. Under Linux, this is a plain terminal window running 'bash' as shell. Under Windows, you should use PowerShell. Then enter the following commands:

```
cd ARCHIMATE
cbshell runall.cbs
```

The text file `runall`.cbs is a so-called CBShell script. It contains commands that interact with a ConceptBase server. In this case it connects to a ConceptBase server, defines the ArchiMate language in ConceptBase, defines some analysis queries, and an example ArchiMate enterprise model ('ArchiSurance'). Then, four example queries are executed to extract dependency information from the enterprise model. The output of the script looks like following:

```
cbshell runall.cbs
Connecting to a ConceptBase server
Defining the ArchiMate language (part) in ConceptBase ...

Finds all infrastructure Nodes that are used by business behaviour that is performed by a
given actor
=====================================================================
DellServer1 in AM_Service1 with
  actor
    COMPUTED_actor_id_2653: ArchiSurance;
    COMPUTED_actor_id_2664: ValuationExpert;
    COMPUTED_actor_id_2668: Client
end

...

WindowsServer2008 in AM_Service1 with
  actor
    COMPUTED_actor_id_2653: ArchiSurance;
    COMPUTED_actor_id_2664: ValuationExpert;
    COMPUTED_actor_id_2668: Client
end



Finds all business services that rely on WindowsServer2008
=====================================================================
ClaimPaymentService

Finds all business services that rely on DebianLinux
=====================================================================
ClaimPaymentService,ClaimValuationService,ClaimAcceptService

Finds all application services that rely on more than one different operating system
=====================================================================
PaymentService
```

The first query 'AM_Service1' returns all node elements (part of the network infrastructure) elements such that a behaviour element 'b' depends on them and an actor depends on the behaviour element b. All such actors are returned together with the respective node element. So, essentially it shows which actors depend on an infrastructure node via some behaviour element.

The second query 'AM_Service2' has a parameter 'WindowsServer2008'. It returns all business servives that depend on this system software. The third query does the same for the system software 'DebianLinux'. Finally, the last query returns all application services that depend on two or more different operating systems.

All queries heavily depend on the transitivity of the 'dependsOn' relation. It realizes the traceability of all model elements in an ArchiMate enterprise model.

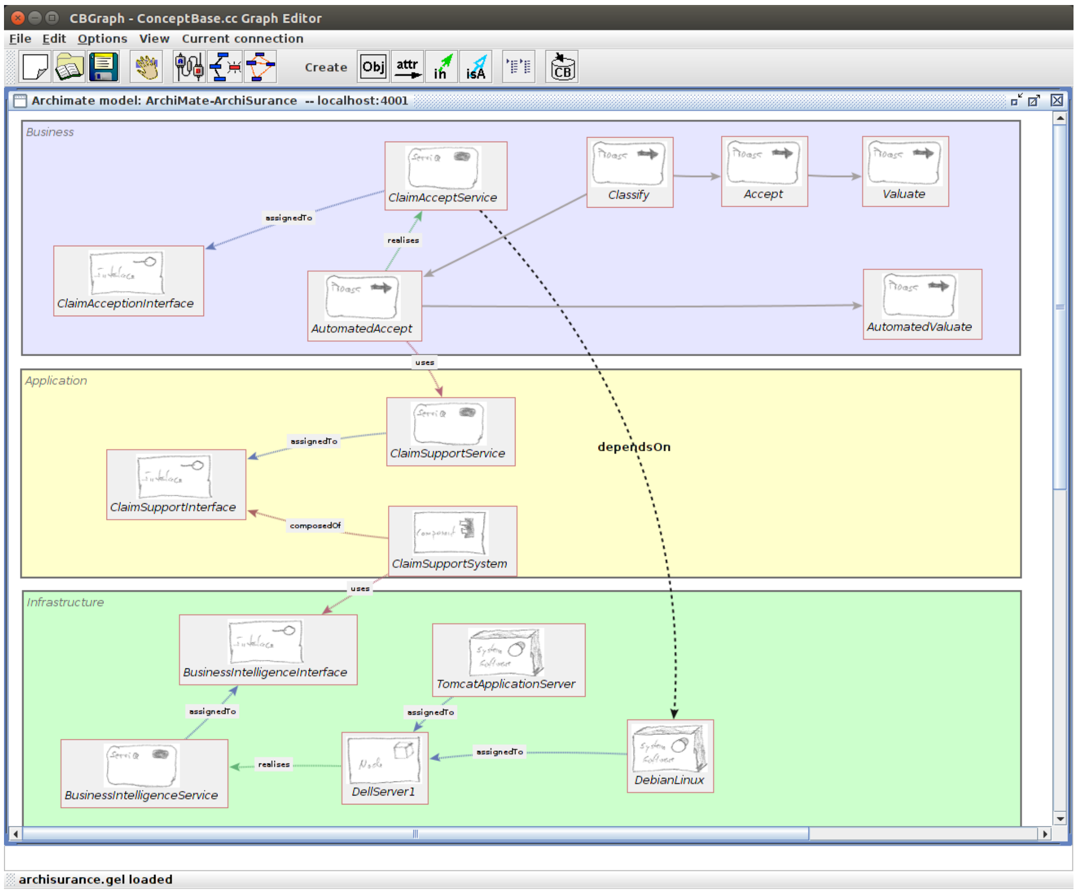
3.3 Graphical analysis of the ArchiMate model

ConceptBase has a graphical browser that allows to navigate and display attributes and relations between objects. The ArchiMate [5] case study includes excerpts of the ArchiSurance example enterprise model. The previous section showed how to create textual analysis reports. In this section, we show how to graphically navigate the transitive 'dependsOn' relation that was superimposed on ArchiMate via the definition of the class 'AnyNode'.

Step 1: To start the demo, open a terminal/command window and enter

```
      cd ARCHIMATE
      cbgraph archisurance.gel
```

The file 'archisurance.gel' contains all Telos sources of the ArhciMate case study plus a graphical view on it. By starting it with cbgraph, the Telos sources will be sent to a ConceptBase server and the graphical view will be displayed in a window.

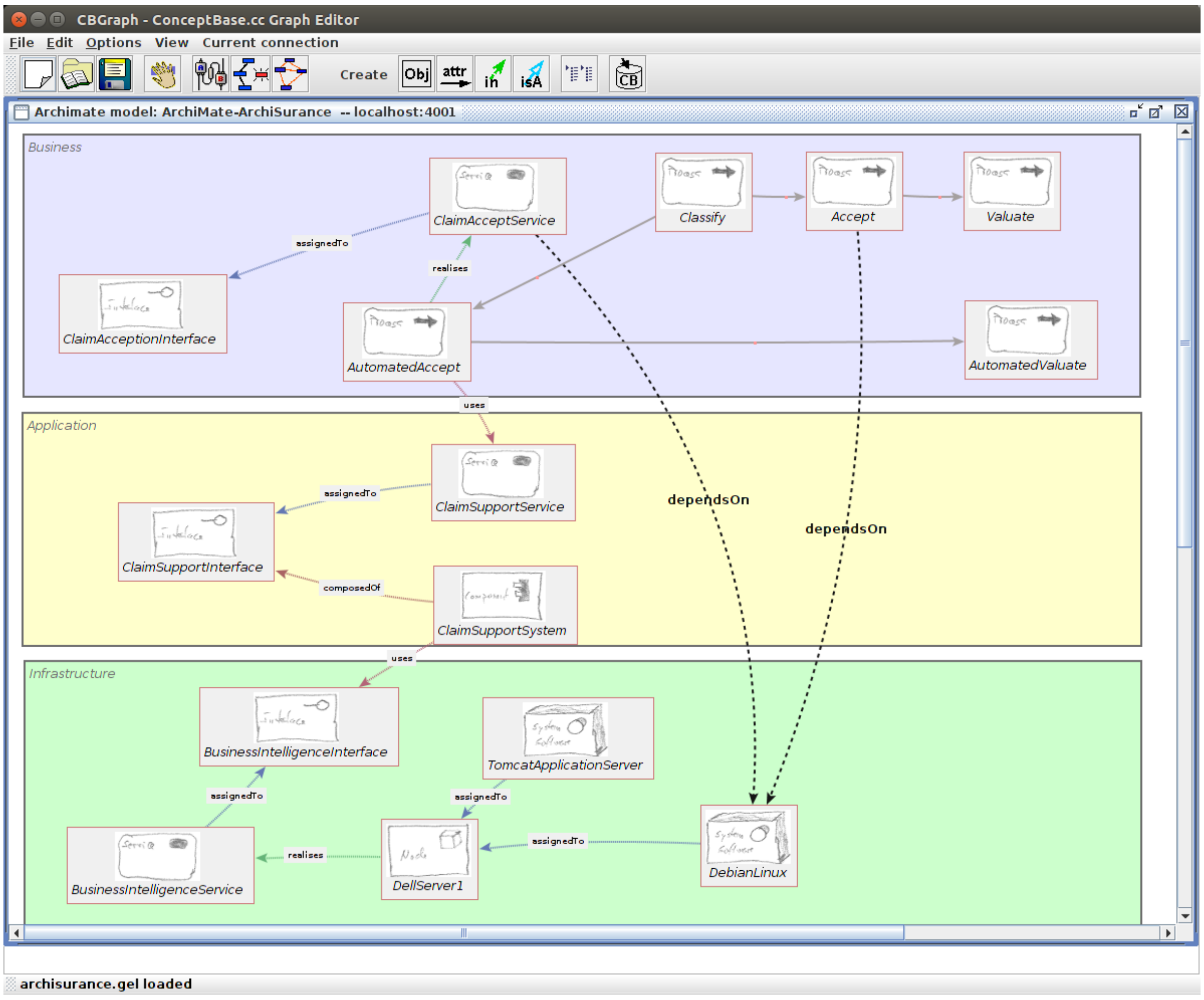


The graphical view displays some elements of the ArchiSurance enterprise model. They are grouped into the layers 'Busness', 'Application', and 'Infrastructure' (=technology). The graphical notation is mimicking the ArchiMate standard but is not quite the same due to limitations of the graphical browser of ConceptBase. The elements are linked by relations like 'assignedTo' and 'realises'. These are standard relations used in ArchiMate. The grey links, e.g. between 'Classify' and 'Accept' are precedence links between processes (called triggers in ArchiMate). They also constitute dependencies since a process depends on its preceding processes. The graphical view also shows a dependency link between 'ClaimAcceptService' (business layer) and 'DebianLinux' (infrastructure layer. The dotted line indicates that this relation was derived from the rules defined in ConceptBase, in particular the transitivity rules. The chain of explicit links that lead to this dependency ios also shown in the graphical view:

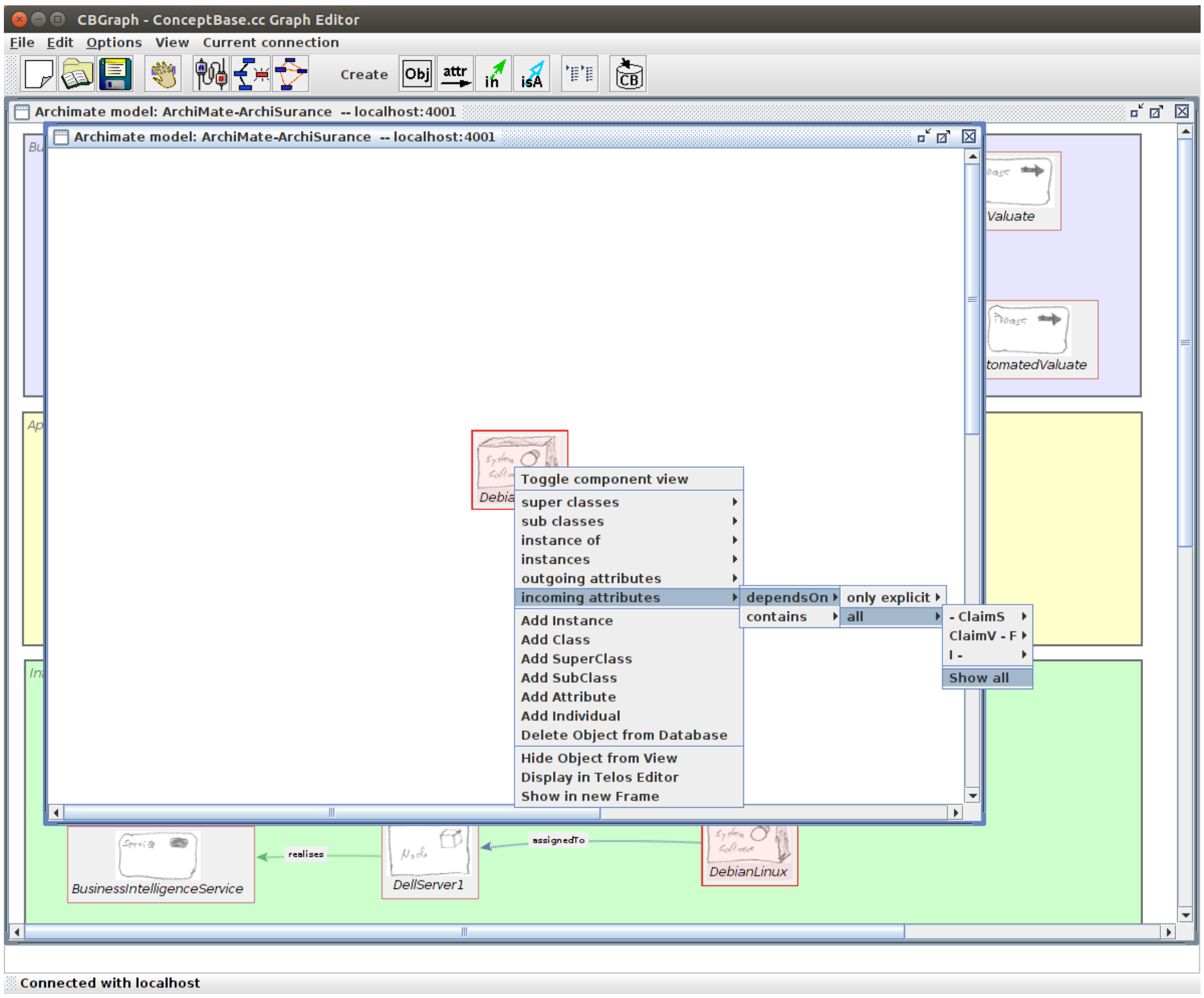
```
ClaimAcceptService, AutomatedAccept, ClaimSupportService,ClaimSupportInterface,
ClaimSupportSystem, BusinessIntelligenceInterface, BusinessIntelligenceService,
DellServer1, DebianLinux.
```

Both directions of 'dependsOn' can be followed, i.e. to elements that a given element is depending on (e.g. ClaimAcceptService to DebianLinux) and vice versa.

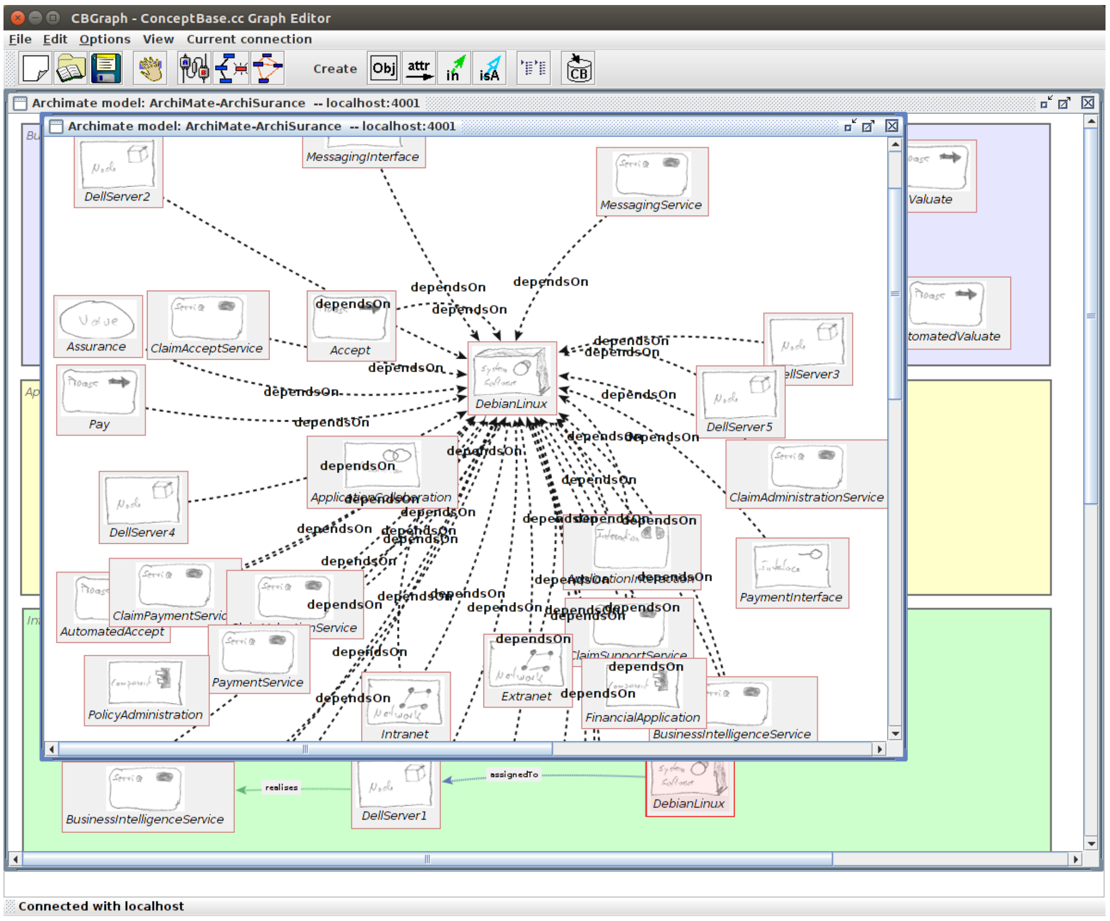
Step 2: To show the reverse direction, right-click on 'DebianLinux', follow menu to 'incoming attributes' (=reverse direction). Then follow 'dependsOn' and 'all'. It will onfold a sub-menu from where you can select specific elements. Take the first one and then 'Accept' as example. The result is shown below:



Step 3: Next, we like to display all elements that depend on DebianLinux. To display them, a new graphical is appropriate to avoid clutter of links. Right-click on 'DebianLinux' and select 'Show in New Frame'. Enlarge the frame and move the 'DebianLinux' node with the left mouse button to its center.



Select the menu path incoming attributes/dependsOn/all/Show all. It will display all elements of the ArchiSurance case that directly or indirectly depend on 'DebianLinux'. This is quite a number. You can re-arrange the nodes to display them. The result may look like below.



This finalizes the ArchiMate case study. Check out the queries defined in `Pt2-Analysis.sml.txt` and use them as starting point to define your own analysis queries. You may also want to use the functions of the ConceptBase answer formatting to produce reports in the (textual) format that suits your needs. Details are in the ConceptBase user manual at
http://conceptbase.sourceforge.net/userManual80/cbm004.html

4. Case 2: 4EM

4EM is a set of interrelated modeling perspectives for enterprise modeling [6]. This case study is about (partially) representing the meta models of the perspectives in ConceptBase and analyze their interrelationships. This shows how ConceptBase can be used to query also meta models, not just models. Such a meta model-based query could for example return all *interface concepts* that are bridging between two modeling perspectives.

Goal of the case study: Analyze the completeness of the meta model of 4EM, in particular by which interface constructs they are interrelated.

Approach: Represent the 4EM constructs in a specialized meta² model and realize the analysis functions by queries on the meta² model.

Acknowledgements: Janis Stirna provided an early draft of the 4EM meta models. They were the bsis for the 4EM meta models coded in ConceptBase.

4EM was derived from the EKD enterprise modeling method. Certain terms in the ConceptBase representation thus use the acronym EKD due to this heritage. We do however always refer to 4EM [6].

4.1 The Meta Model of 4EM in ConceptBase

4EM comprises the following six modeling perspectives: the goal perspective, the process perspective, the actors and resources perspective, the concepts perspective (a.k.a. the data model), the technical components and requirements perspective, and the business rule perspective.

All six perspectives have their own concepts but are also strongly interrelated to each other. To represent the perspectives in ConceptBase, we first look at the meta² model (the meta meta model) that is used to facilitate their representation. The source code of the definitions can be found at http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/d3827471/00-ekd-ndl.sml.txt

The definition provide some generic constructs for multiplicities ('necessary', 'single') of attributes and relations. Further, relations can be defines to be transitive and to be inverse to each other.

The first new addistion is a meta² class 'Notation'. It lists the constructs of a modeling notation. A rule derives that all links inside a notation (between two contructs 'x' and 'y' of the same notation) are also part of the notation.

```
Notation in Class isA Node with
  attribute
    includes: NodeOrLink
  rule
    addLinks: $ forall N/Notation x,y/NodeOrLink link/NodeOrLink!connectedTo
                    (N includes x) and (N includes y) and From(link,x) and To(link,y)
                ==> (N includes link) $
end
```

The next new contructs are for 'Model' and 'ModelElement':

```
Model in Node with
   attribute
      contains: ModelElement
end
```

Since we want to analyze the perspectives, we introduce a construct 'Model'. It is just a container of model elements. Model elements are defined by a rule:

```
ModelElement in NodeOrLink,Class with
  rule
    me1 : $ forall x/VAR (x [in] NodeOrLink)  ==> (x in ModelElement) $
end
```

The predicate (x [in] mc) expresses that 'x' is an instance of some class 'c', which itself is an instance of the meta class 'mc'. Here, 'NodeOrLink' is a meta^2 class. Its instances are the constructs of 4EM, e.g. 'GM_Goal' (the construct for goals). Hence, any instance of a class like 'GM_Goal' is also an instance of 'ModelElement'.

The full translation of 4EM to ConceptBase is available from the archive
http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/d4167283/4EM.zip

4.1 Analysis queries for meta models

Comprehensive meta models of enterprise modeling methods can have many constructs. It may then be a challenge to find out whether all constructs are well-related to each other, in particular when the modeling perspectives are developed by different teams.

We therefore provide some analysis queries that target this situation. Their definitions are in the file

http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/d3827503/20-ekd-analyze.sml.txt

To facilitate the analysis queries, we first map all relations between modeling constructs to 'links':

```
NodeOrLink in Class with
  attribute
    link: NodeOrLink
  symmetric
    link_sym: NodeOrLink
  transitive
    linkTo: NodeOrLink
  rule
    conn2link: $ forall n1,n2/NodeOrLink (n1 connectedTo n2) ==> (n1 link n2) $;
    isa2link: $ forall n1,n2/NodeOrLink :(n1 isA n2): ==> (n1 link n2) $;
    link2sym: $ forall n1,n2/NodeOrLink (n1 link n2) ==> (n1 link_sym n2) $;
    sym2tr: $ forall n1,n2/NodeOrLink (n1 link_sym n2) ==> (n1 linkTo n2) $
end
```

Each connection between node or links (meta^2 class!) is a 'link'. The relation 'link_sym' is the symmetric closure of 'link' (i.e. if n1 is linked to n2, then n2 also to n1 in the symmetric closure), and 'linkTo' is the transitive closure of 'link_sym', hence, 'linkTo' can be used to analyze connections regardless of their direction and regardless whether they are direct or indirect.

The first analysis query returns all constructs

```
ImportedConstruct in GenericQueryClass isA NodeOrLink with
  computed_attribute,parameter
    notation : Notation
  constraint
    isOutside: $ exists n/NodeOrLink
                    (notation includes n) and (n link_sym this)
                    and not (notation includes this) $
end
```

Hence, an imported construct is any construct that is connect to some modeling construct 'n' of the given notation (provided as parameter of the query). Note that 'link_sym' is symmetric but not transitive. Further note that the query makes no reference at all to 4EM. It is generic for any modeling notation using the facilities of the meta^2 model discussed in this chapter.

The next analysis query returns those constructs of an 'exporting' notation (given as parameter) that is an improted construct for the 'importing' notation (also given as parameter):

```
InterfaceConstruct in GenericQueryClass isA NodeOrLink with
  computed_attribute,parameter
    importing: Notation;
    exporting: Notation
  constraint
    isInBetween: $ (this in ImportedConstruct[importing/notation])
                     and (exporting includes this) $
end
```

Hence, the analyst provides the names of the two notations and gets as result the interface between them. The final query is to detect constructs that are not assigned to any notation bur well used:

```
UnAssignedConstruct in QueryClass isA Node with
  constraint
    noMT : $ exists nota1/Notation (this in ImportedConstruct[nota1])
             and not exists nota2/Notation (nota2 includes this) $
end
```

Thus, a notation 'nota1' uses a construct 'this' (answer variable) but no notation 'nota2' defines it.

4.2 Using the analysis queries for the 4EM meta model

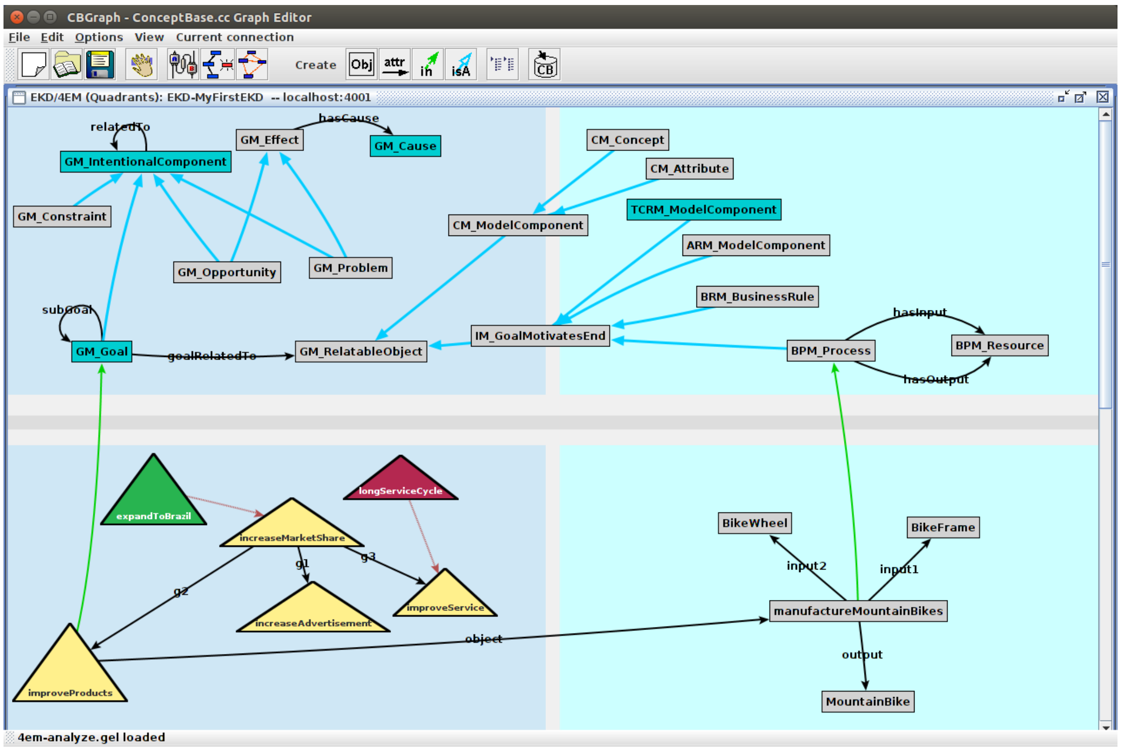
The following script shows how to analyze the 4EM meta model with the three analysis queries discussed in the previous section.

Step 1: Download the the file 4EM.zip from http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/3821866
It contains the ConceptBase sources for representing 4EM and the analysis queries discussed in the previous section. Further it contains the file '4em-analyze.gel'. This file includes all sources for 4EM meta models plus some example 4EM model.

Step 2: Open a command window. Under Linux, this is a plain terminal window running 'bash' as shell. Under Windows, you should use PowerShell. Then enter the following commands:

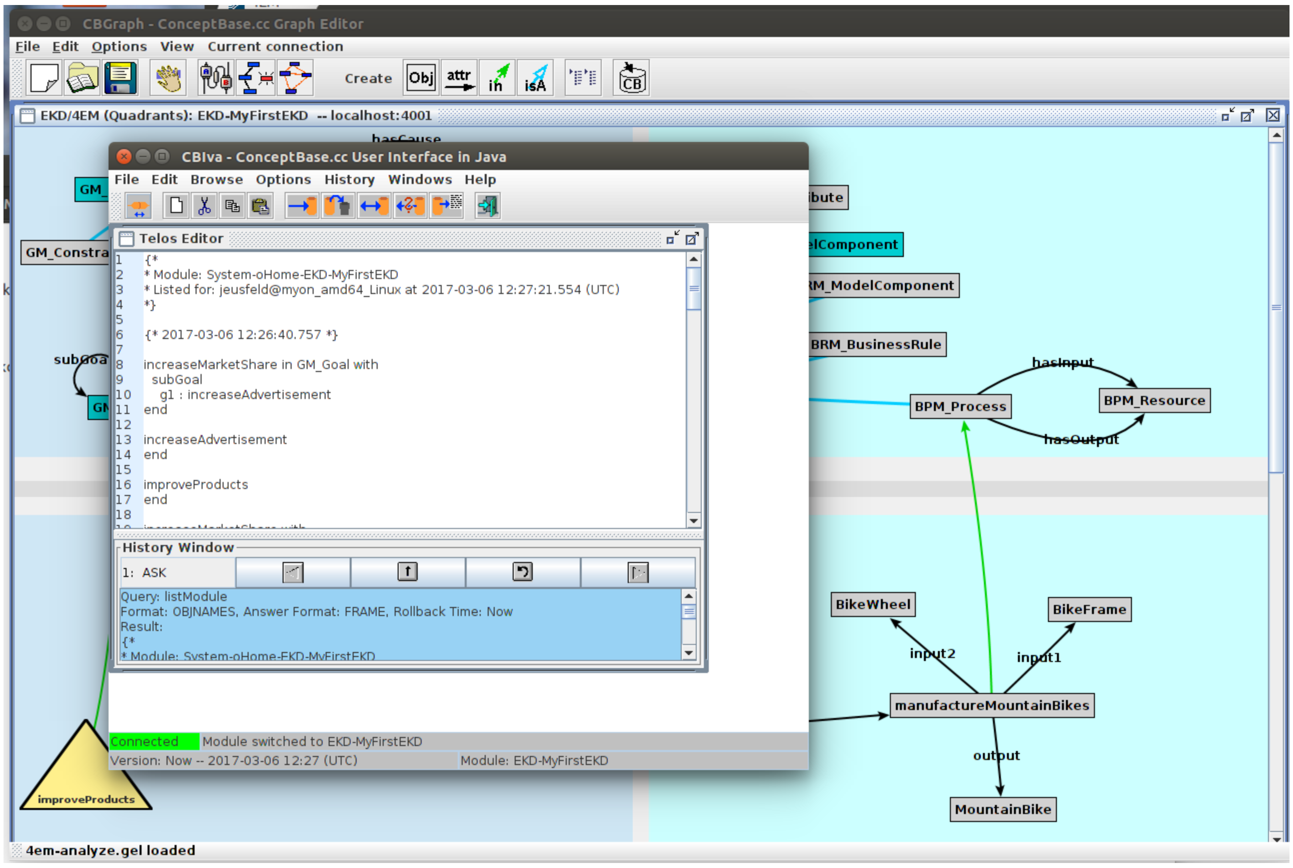
```
cd 4EM
cbgraph 4em-analyze.gel
```

The graph editor CBGraph opens up and displays a view on a 4EM model:



The lower part is some example 4EM model displaying goals and an excerpt of a process model. Note that the graphical symbols are not the standard 4EM symbols but just illustrative. The upper part is the visualization of a part of the 4EM meta model showing some goal modeling constructs and their interface to other 4EM perspectives.

Step 3: Select the menu option 'File / Start CBIva Workbench'. It starts the textual user interface and shows the content of the example 4EM model 'MyFirstEKD'.


CBGraph - ConceptBase.cc Graph Editor
File Edit Options View Current connection
Create Obj attr in isA CB
EKD/4EM (Quadrants): EKD-MyFirstEKD -- localhost:4001
CBIva - ConceptBase.cc User Interface in Java
File Edit Browse Options History Windows Help
Telos Editor
History Window
hasCause
GM_
GM_Constra
subGoa
GN
ibute
lComponent
RM_ModelComponent
BRM_BusinessRule
hasInput
BPM_Process
BPM_Resource
hasOutput
BikeWheel
BikeFrame
input2
input1
manufactureMountainBikes
output
MountainBike
improveProducts

```
{*
* Module: System-oHome-EKD-MyFirstEKD
* Listed for: jeusfeld@myon_amd64_Linux at 2017-03-06 12:27:21.554 (UTC)
*}

{* 2017-03-06 12:26:40.757 *}

increaseMarketShare in GM_Goal with
  subGoal
    g1 : increaseAdvertisement
end

increaseAdvertisement
end

improveProducts
end
```

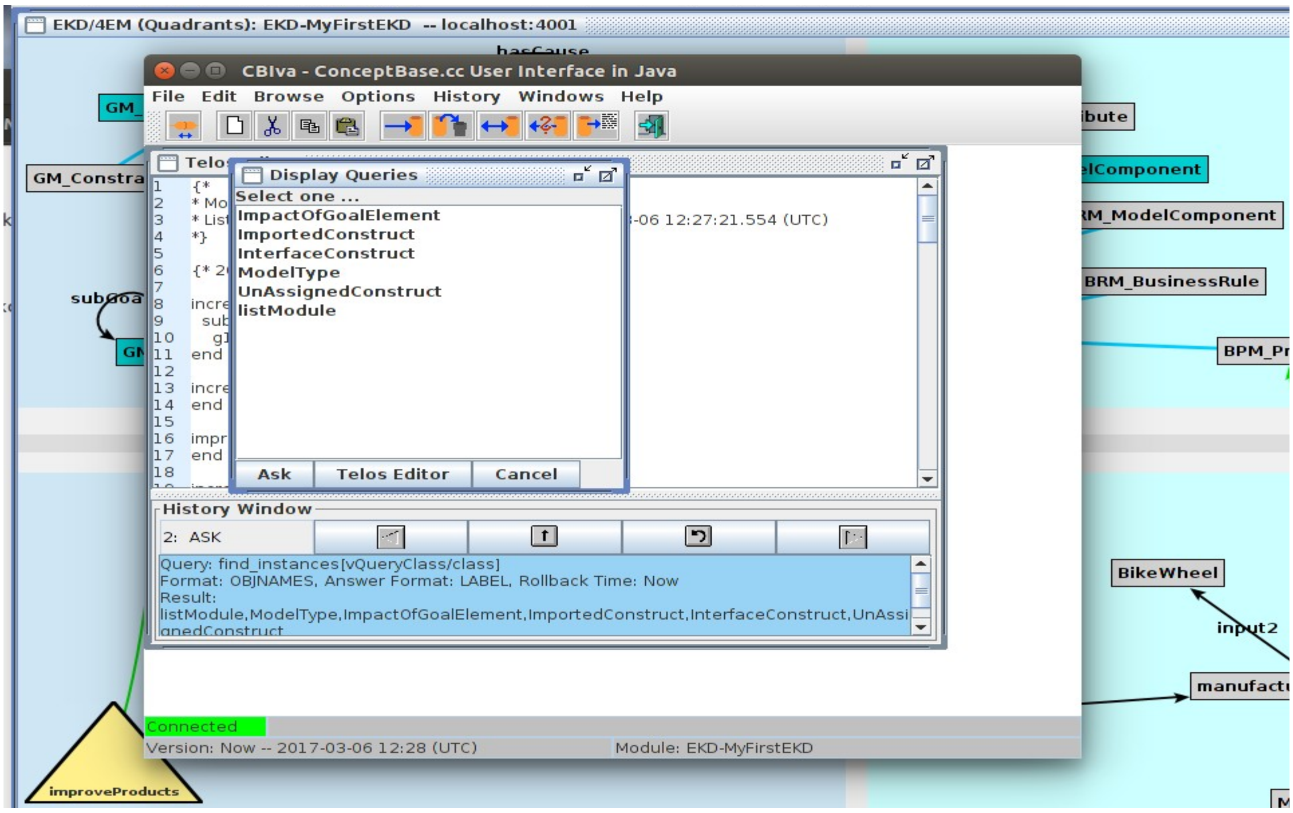
```
1:  ASK
Query: listModule
Format: OBJNAMES, Answer Format: FRAME, Rollback Time: Now
Result:
{*
* Module: System-oHome-EKD-MyFirstEKD
```

Connected Module switched to EKD-MyFirstEKD

Version: Now -- 2017-03-06 12:27 (UTC) Module: EKD-MyFirstEKD

4em-analyze.gel loaded

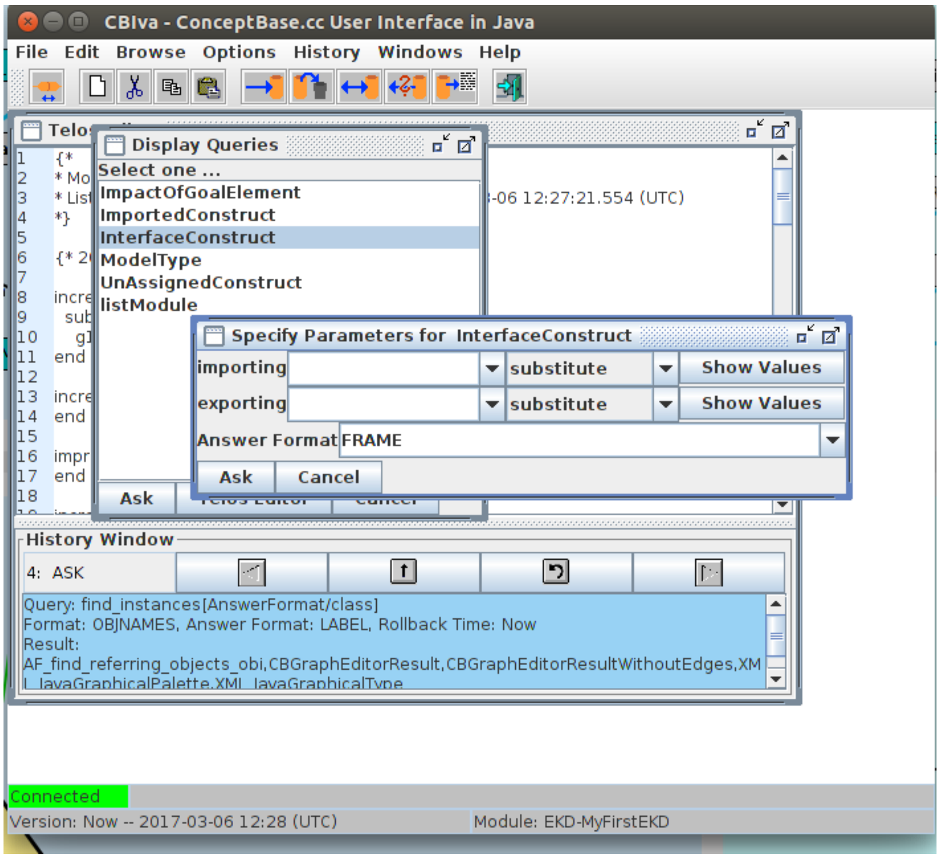
Step 4: Select the menu option Browse / Display Queries in the CBIva Workbench. It displays the currebtly defined analysis queries.



Query: find_instances[vQueryClass/class]
Format: OBJNAMES, Answer Format: LABEL, Rollback Time: Now
Result:
listModule,ModelType,ImpactOfGoalElement,ImportedConstruct,InterfaceConstruct,UnAssignedConstruct

Connected
Version: Now -- 2017-03-06 12:28 (UTC)
Module: EKD-MyFirstEKD

Select the query 'InterfaceConstruct':



importing | substitute
exporting | substitute
Answer Format FRAME

Query: find_instances[AnswerFormat/class]
Format: OBJNAMES, Answer Format: LABEL, Rollback Time: Now
Result:
AF_find_referring_objects_obi,CBGraphEditorResult,CBGraphEditorResultWithoutEdges,XML_JavaGraphicalPalette,XML_JavaGraphicalType

Connected
Version: Now -- 2017-03-06 12:28 (UTC)
Module: EKD-MyFirstEKD

The query window asks for the parameters 'importing' and 'exporting'. You may select the corresponding notations such as 'GoalModel' for these parameters, or leave them open. In this case, ConceptBase shall fill all possible values to these two parameters and give a comprehensive report about interface constructs between any two 4EM modeling perspectives:

```
GM_Goal in InterfaceConstruct with
  exporting
    COMPUTED_exporting_id_1730 : GoalModel
  importing
    COMPUTED_importing_id_1862 : BusinessProcessModel;
    COMPUTED_importing_id_2091 : BusinessRuleModel
end
...
```

The answer shows all interface constructs. For example, the construct 'GM_Goal' is exported by the 'GoalModel' perspective and imported into two other perspectives, 'BusinessProcessModel' and 'BusinessRuleModel'.

Try the other analysis queries to get a more complete picture of the 4EM modeling perspectives. Try for example 'UnAssignedConstruct' to see that some 4EM constructs were not yet assigned to any of the 6 perspectives.

5. ADOxx coupling

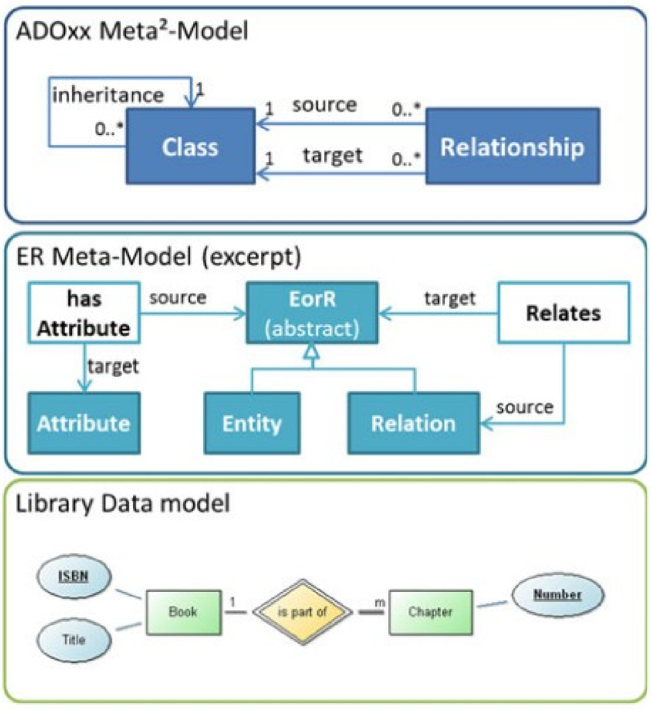
ADOxx is a suite for meta modeling and modeling. It provides two tools. First, the ADOxx development toolkit is used to define domain-specific modeling language (DSML)s by meta models of their constructs and attaching modeling procedures and semantic grounding (e.g. simulation engines for process models). The ADOxx modeling toolkit is configured by the meta models created by the development toolkit and provides a customized modeling environment for the DSML at hand.

The purpose of the ADOxx-to-ConceptBase coupling is to provide semantic integrity checking for both the ADOxx development toolkit and the ADOxx modeling toolkit. The principal idea is discussed in the SemCheck chapter [1].

This chapter only discusses how ADOxx can generate the ConceptBase commands to realize the integrity checking services. The principal coupling was tested by a prototype in spring 2016. The code is however not yet stable and about to be integrated in future releases of ADOxx.

5.1 ADOxx meta² model in ConceptBase

ADOxx uses a dedicated meta² model for defining DSMLs. The meta² model is presented in [7]. Below is the represesenttion of the meta model and its use for the Entity-Relationship modeling language (ER).



ADOxx meta² model and its uses for ER (from [7])

The mapping into ConceptBase requires first a manual translation of the ADOxx meta² model into ConceptBase. The meta² model is quite simple, but it does feature a number of semantic constructs. First, there is an 'inheritance' construct. We interpret this as a sort of sub-classing and will provide some semantics in form of deductive rules and integrity constraints for it. Second, there is are various cardinality constraints. We shall use the 'single' and 'necessary' categories in ConceptBase to model them. Thirdly, there is an '(abstract)' stereotype. We interpret that as an abstract class, i.e. its instances are limited to instances of its subclasses.

Subsequently, you see part of the definitions used to embed the ADOxx meta² model in ConceptBase:

```
Proposition with
  attribute
    reflexive:  Proposition;      {* any object is related to itself       *}
    transitive: Proposition;      {* relation is closed under transitivity *}
    symmetric: Proposition;       {* if x rel y then also y rel x          *}
    antisymmetric: Proposition;   {* if x rel y and (y rel x) then x=y *}
    asymmetric: Proposition       {* if x rel y then not y rel x     *}
end

RelationSemantics in Class with
  constraint
   asym_IC: $ forall AC/Proposition!asymmetric C/Proposition x,y/VAR M/VAR
                     P(AC,C,M,C) and (x in C) and (y in C) and
                     (x M y)  ==> not (y M x) $;
   antis_IC: $ forall AC/Proposition!antisymmetric C/Proposition x,y/VAR M/VAR
                     P(AC,C,M,C) and (x in C) and (y in C) and
                     (x M y) and (y M x)  ==> (x = y) $
  rule
   trans_R: $ forall x,z,y,M/VAR
                     AC/Proposition!transitive C/Proposition
                     P(AC,C,M,C) and (x in C) and (y in C) and (z in C) and
                     (x M y) and (y M z) ==> (x M z) $;
   ...
end


...

ABSTRACT in Class with
   constraint
     no_explicit_instance : $ forall c/ABSTRACT x/Proposition
             (x in c) ==> not In_s(x,c) $
end
```

The full source code is provided at
http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/d3892244/meta2-er.cbs.txt
and in the SemCheck-Demo folder.

Note the definition of 'ABSTRACT'. It forbids explicit instances ('In_s').

Next, we define the ADOxx meta² model in ConceptBase:

```
CLASS with
  attribute,single,transitive,reflexive,antisymmetric
    inheritance: CLASS
end

CLASS with
   attribute
     relationship: CLASS
end


InheritanceRule in Class with
  rule
     r1: $ forall C,D/CLASS x/Proposition  (C inheritance D) and
              (x in C) ==> (x in D) $
end
```

Note that the 'inheritance' link is defined by the categories 'single' (a class can have at most one superclass), 'transitive' (the superclass of a superclass of c is also a superclass of c), 'reflexive' (each class is a superclass of itself) , and 'antisymmetric' (if two classes are superclasses of each other in both directions, then they must be the same). The original ADOxx meta² model has the cardinality '1' on the target side of 'inheritance'. We map that here as '0..1' (=single) rather than '1..1' (=single plus necessary). The reason is that each superclass hierarchy must have a top-most class that has no superclass itself.

The 'relationship' attribute of CLASS represents the RELATIONSHIP concept of the ADOxx meta² model. An attribute in ConceptBase always has a exactly one source (where it starts from) and exactly one target (where it ends in). Hence, the cardinality constraints imposed from the ADOxx meta² model are automatically satisfied.

The 'InheritanceRule' provides the semantics of the 'inheritance' contruct. It is modelled here as class membership inheritance. Each instance 'x' of a subclass 'C' is also an instance of the superclass 'D'.

The class 'Proposition' is predefined in ConceptBase. It stands for any model element (=object) represented in ConceptBase. Note that ConceptBase represents instances, their classes, their metaclasses etc. all as objects/propositions. The same holds for any attribute or link.

5.2 ADOxx ER model in ConceptBase

The next part is the definition of the ER meta model:

```
EoR in ABSTRACT,CLASS with
  relationship
    hasAttribute: ATTRIBUTE
end


ATTRIBUTE in CLASS end

ENTITY in CLASS with
  inheritance
     super: EoR
end

RELATION in CLASS with
  inheritance
      super: EoR
  relationship
      relates: EoR
end
```

The ConceptBase representation closely follows the way how the meta model is represented in ADOxx. Note that ConceptBase uses so-called categories to represent that one link is an instance of another one. For example, the 'super' link of ENTITY is an instance of the 'inheritance' link of CLASS:

5.3 ADOxx example model in ConceptBase

The source code in http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/d3892244/meta2-er.cbs.txt (see SemCheck-Demo folder) contains the representation of an example ER model:

```
ISBN in ATTRIBUTE end
Title in ATTRIBUTE end
Number in ATTRIBUTE end

Book in ENTITY with
  hasAttribute
    attr1: ISBN;
    attr2: Title
end

Chapter in ENTITY with
  hasAttribute
    attr3: Number
end

isPartOf in RELATION with
  relates
    relates1: Book;
    relates2: Chapter
end
```

Consult the above source code for the complete mapping.

5.4 Testing the constraint services

ConceptBase knows no predefined abstraction levels such as meta² model, meta model, model etc. Still, it makes sense to discuss the use of the constraint services for different level pairs:

Meta² model vs. Meta Model:

The representation of the ADOxx meta² model provides a number of constraints that each DSML (i.e. ADOxx meta model) must fulfill. As an example, consider the single-valuedness of the 'source' and 'target' links.

An example is to add a more strict cardinality constraint for the 'inheritance' construct:

```
CLASS with
  attribute,single,necessary,transitive,reflexive,antisymmetric
    inheritance: CLASS
end
```

This will prevent to formulate any subclass hierarchy. The 'inheritance' relation would be forced to be empty.

Secondly, consider the constraint an ABSTRACT:

```
ABSTRACT in Class with
   constraint
     no_explicit_instance : $ forall c/ABSTRACT x/Proposition
             (x in c) ==> not In_s(x,c) $
end
```

It refers to an instance c of 'ABSTRACT'. This instance 'c' (e.g. the meta class 'EoR') may not have an explicit instance (which would belong to the model level).
The constraint can be tested by attempting to declare such as instance:

```
dummy in EoR end
```

The meta² model of ADOxx is fixed. Hence, the semantic constraints are also fixed and can be hard-coded.

Meta model vs. Model:

Such constraints are specific to the modeling language. For example, one may demand that each instance of 'ENTITY' has at least one attribute.

The constraint would be

```
ENTITY_Rule1 in Class with
  constraint
     atleastoneAttribute: $ forall e/EoR (e in ENTITY) ==>
                                exists a/ATTRIBUTE (e hasAttribute a) $
end
```

One can violate the constraint by just adding a solitary entity type:

```
Student in ENTITY
end
```

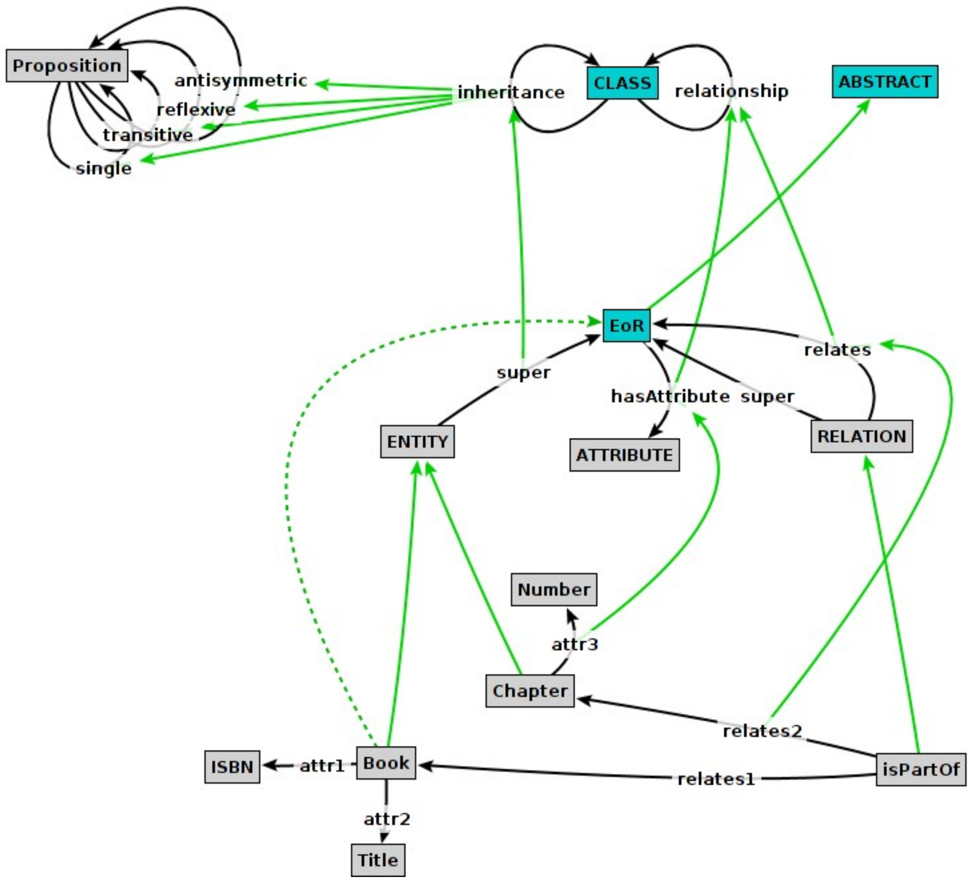
A more advance method for such constraints is to define them as test queries:

```
ENTITYwithoutATTRIBUTE in QueryClass isA EoR with
  constraint
     atleastoneAttribute: $ (this in ENTITY) and
                            not exists a/ATTRIBUTE (this hasAttribute a) $
end
```

The query can be called at any time and shall return the entity types that have currently not attribute. Entity types without attributes then formally do not violate the constraint. However, the 'violators' can be retrieved by the query at any time, e.g. when the ER model is about to be released.

The definition of the meta model constraints is part of the definition of the DSML. Unlike as for the constraints of the meta^2 model, the meta model constraints are depending on the DSML to be defined. For a smooth integration with ADOxx, it would be good to allow the definition of the constraints via ADOxx (e.g. by attaching them as 'constraint' text to the meta classes and then passing them to ConceptBase.

The ConceptBase representation of the ADOxx meta model for ER can also be graphically analyzed, see http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/d4167576/BEEUP-ERMM-CB.gel



ConceptBase representation of the ADOxx meta² model and its use for ERD

Model vs. Instance:

Such constraints are domain-specific constraints, e.g. about how many chapters a book may have. We give no specific example here but refer to the ConceptBase Forum for examples. http://conceptbase.sourceforge.net/CB-Forum.html

5.4 Technical coupling

The ADOxx tools are stand-alone programs. The coupling between the ADOxx tools and ConceptBase should thus been loose and provide an optional constraint checking support. The easiest way of coupling is to use the ConceptBase CBShell interface. As such, CBShell is a command line client that supports a number of command to interact with a ConceptBase server (CBserver).

We suggest to start a CBserver on a Linux host in your network. You can install ConceptBase on the Linux computer just like under Windows (see section 2.1). The difference to Windows (and MacOS) is that is allows to start a CBserver. Take care that the firewall settings do not prevent the CBserver from being accessed from within your network. The default portnumber of the CBserver is 4001. It is not necessary to make the CBserver visible to computers outside your local network. The only purpose of the CBserver shall be to support the ADOxx tools. If your ADOxx tools run outside the netwrokd in which the CBserver is running, you need to make it accessible to the outside world however.

To start the CBserver, open a terminal on the Linux computer and enter

```
cbserver -r 2 -a jonny -g public -ia 0.5 -u nonpersistent -d MDB &> log.txt &
```

The option -r instructs the CBserver to restart after 2 seconds if it is shutdown by a client (e.g. ADOxx). This restart presents a fresh CBserver to ADOxx tools. The option `-a jonny` declares 'jonny' as administrator user of the CBserver. This is the user who can permanently shutdown the CBserver, i.e. bypassing the restart function. The user name is a login name of a user on the client side. The option '-g public' does two things. First, it puts the CBserver into the so-called slave mode. This means that it will shut-down when its last clients logs off. This will then trigger the fresh restart for the next clients. Second, it configures the CBserver database module 'oHome' to be a so-called auto-home module. That means that every user of the CBserver gets his/her own database module assigned. For example, the user 'marie' would get the home module 'oHome-marie'. The option -ia 0.5 specifies after how many hours of inactivity a client is regarded to be permanently inactive. This option lets the CBserver restart even if there are some inactive clients still logged in, e.g. when the client fails to exit the CBserver. You can disable tyhis function by choosing a negative value, e.g. '-ia -1'. The option -u nonpersistent' will regard all updates to the database as volatile, i.e. they are lost after a restart. This is in the case of ADOxx tools the recommended behavior since the models and meta models are maintained on the ADOxx side. The CBserver only temporarily holds portions of the models. Finally, the option '-d MDB' loads the ConceptBase database MDB. This database can be preconfigured with the ConceptBase representation of the ADOxx meta^2 model (CLASS and its links). Those definitions should be stored in the database module oHome of the ConceptBase server. As an alternative, the ADOxx tools can define the meta^2 model at run time using the 'tell' command (see below). The tail of the command is about storing the trace messages into a log file.

While the CBserver discussed in this section is intended for use with ADOxx, you can also directly access it via the ConceptBase user interface tools 'cbiva' (editor and query interface), 'cbgraph' (graphical interface), and 'cbshell' (command line interface). See also http://conceptbase.sourceforge.net/userManual80/cbm007.html#sec102 in the ConceptBase User Manual.

Subsequently, we assume that you started the CBserver on a host 'cbserver.acme.com' using the default port number 4001. Further we assume that you have some Java program or class, say

Adoxx2CB.java, which foirms the interface to ConceptBase on the ADOxx side. The ADOxx2CB class can use the following API to interact with ConceptBase:

```
LocalCBclient cbClient = new LocalCBclient();

String answer;
```

This constructor provides the API object `cbClient` to be used for the subsequent method calls. The variable answer is used to read the response for the subsequent API commands.

```
answer = cbClient.connect(sHost, iPort, sTool, sUser);
```

This method connects your Java program to the CBserver specified by sHost (the domain name of the computer on which your CBserver runs on) and iPort (the port number as an integer; usually 4001). The string sTool shall be a self-selected name of your tool, e.g. "ModelerXY" and sUser is a string containing a user name (typically your user name). If you use `null` as username, then the cbClient will use the login name of the computer user that started the Java program. The return value is "yes" if successful and "no" else.

```
answer = cbClient.disconnect();
```

This method disconnects connects your Java program from the CBserver. The return value is "yes" if successful and "no" else. If you started the CBserver in non-persistent mode (recommended for ADOxx), then the disconnection shall also delete all frames defined previously.

```
answer = cbClient.pwd();
```

This method outputs your current working module path, e.g. "System-oHome".

```
answer = cbClient.mkdir(newModule);
```

This method created a new submodude (e.g. "MyMod") in the current module.

```
answer = cbClient.cd(newModule);
```

This method changed the current module to newModule, e.g. "MyMod".

```
answer = cbClient.tells(sFrames);
```

This method tells (=defines) the objects given by the string sFrames to the current module of the CBserver. The return value is "yes" if successful and a string containg user-readable error messages else. The string sFrames may contain many frames. They are stored in ConceptBase in one transaction. Hence, take care that the frames are correctly formed.

`answer = cbClient.untells(sFrames);`

This method untells (=removes) the objects given by the string sFrames from the current module of the CBserver. The return value is "yes" if successful and a string containing user-readable error messages else.

`answer = cbClient.asks(sQuery, sFormat);`

This method asks the query call `sQuery` (given by a String) to the CBserver The return value is a string containing the answer to the query. The return value is "no" in case of errors, e.g. when the query is not defined. The query call can have arguments, e.g. `"get_object[Class/objname]"`. The CBserver shall use the answer format `sFormat` for the result. Thus, if you want to define your own answer format, then use the facilities described at http://conceptbase.sourceforge.net/userManual80/cbm004.html. The query shall be answered in the context of the current module and the current time ("Now").

`answer = cbClient.asks(sQuery);`

Same as `cbClient.asks(sQuery,"default")`, i.e. the CBserver determines the applicable answer format (in most cases: `"FRAME"`).

Example code for a TinyClient class:

```
import i5.cb.api.*;

public class TinyClient {
  private static LocalCBclient cbClient = null;
  public static void main(String argv[])  {
    String answer;
    cbClient=new LocalCBclient();
    answer = cbClient.connect("cbserver.acme.org",4001,"TinyClient",null);
    answer = cbClient.tells("Employee in Class end");
    if (answer.equals("yes")
      answer = cbClient.asks("get_object[Employee/objname]");
    System.out.println(answer);
    answer = cbClient.disconnect();
  }
}
```

You need to compile the Java class with the cb.jar library. This contains the API to be used. The compilation under Windows can be done in the directory where the Java class is defined:

```
 javac -classpath c:\conceptbase\lib\classes\cb.jar TinyClient.java
```

Then start the CBserver on your Linux machine (e.g., `cbserver.acme.org`). The port number is by default 4001:

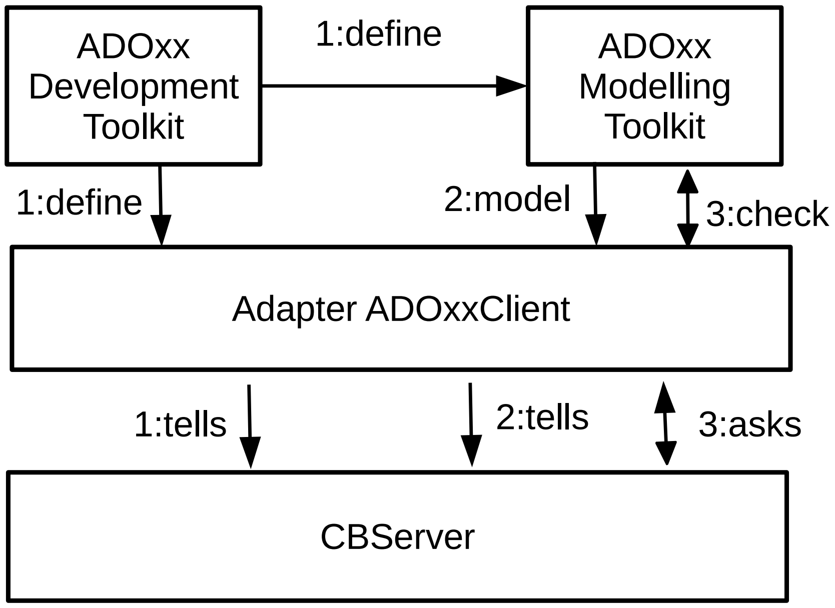
```
cbserver -r 2 -a jonny -g public -ia 0.5 -u nonpersistent -d MDB &> log.txt &
```

Finally run your Java program that uses the API:

```
 java -classpath c:\conceptbase\lib\classes\cb.jar;. TinyClient
```

5.5 The ADOxxClient Scenario

The ADOxx case study is completed by code to mimic the way how ADOxx can directly interact to a ConceptBase server via the simple-String-based interface. We use the following architecture for this coupling:



The adpater in the middle is Java code that extracts models and meta models from ADOxx (Development toolkit or Modelling toolkit) and then tells them to the ConceptBase server.

The first step is to create the initial database for the ConceptBase server, preloaded with definitions about the ADOxx meta² model. We assume that you have access to a Linux server, on which you have installed ConceptBase (see chapter 2). Make sure that you have set the environment variable CB_HOME, ideally in the shell profile like .bashrc. Also make sure that $CB_HOME is in your Linux search path.

Step 1: Copy the file createADOCB.cbs from your directory SemCheck-Demo/ADOxx-CONCEPTBASE-ER to the Linux computer. Then execute the command

```
cbshell createADOCB.cbs
```

The command will create a database ADOCB on your Linux computer. It overwrites any existing database ADOCB.

Step 2: Start a re-starting CBserver on the Linux computer (one line)

```
cbserver -sm slave -r 1 -port 4002 -a yourusername -t low -d ADOCB -u
nonpersistent &> log-4002.txt &
```

The slave mode makes sure that the CBserver is stopped when the last client disconnects. The -r command makes sure that it is freshly re-started afterwards. We use here the port number 4002. You may also use another port number. The -a option sets the user who has administrator rights for the CBserver, i.e. who can also stop it without re-starting. Use your user name on your client computer (e.g. Windows) that you use work with ADOxx or any other modeling environment to be connect to the CBserver. The update mode is set to non-persistent, i.e. after a re-start the database is set to its initial state.

Step 3: Return to your client computer (e.g. Windows) and open a command window (e.g. PowerShell). Switch to the directory SemCheck-Demo/ADOxx-CONCEPTBASE-ER. Then, edit the file ADOxxClient.java. Adapt the string `sCBserver` to the domain name of your Linux computer that runs the CBserver. You need to have ConceptBase installed on the client (Windows) computer as well (see chapter 2). Then execute

```
      javac -classpath c:\conceptbase\lib\classes\cb.jar ADOxxClient.java
```

It will compile the example ADOxxClient.

Step 3: Run the ADOxx client by

```
      java -classpath c:\conceptbase\lib\classes\cb.jar;. ADOxxClient
```

The ADOxxClient will connect to the CBserver, define the ER meta model (see section 5.2), then define an example ER model LIT_MODEL and perform some integrity checks.

You can use this Java program as a role model for integrating the services of ConceptBase into ADOxx or any other modelling environment.

To stop the CBserver, switch to the Linux computer and executes

```
      cbshell
      connect localhost 4002
      stopServer
      exit
```

You can also connect to the CBserver from your Windows machine via CBIva and stop the CBserver via the menu option File/StopServer. Note that you have previously started the CBserver with your Windows username as administrator.

You can also have the CBserver running on your Linux computer for a virtually unlimited time. The re-start mechanism makes sure that it always waits for connections with a fresh ADOCB database. The CBserver can also cater for multiple parallel clients. However, it would have to be set up in a slighlty different way using dedicated home modules for each client.

References

[1] Manfred A. Jeusfeld: SemCheck - Checking Constraints for Multi-perspective Modeling Languages. In Dimitris Karagiannis, Heinrich C. Mayr, John Mylopoulos (eds.): Domain-Specific Conceptual Modeling - Concepts, Methods and Tools. Springer, 2016, pp. 31-53. DOI http://dx.doi.org/10.1007/978-3-319-39417-6_2.

> This bookchapter introduces into constraint checking for multi-perspective modeling languages. It discusses as examples ArchiMate, 4EM and also the way how ADOxx can be coupled to ConceptBase.

[2] M. Jarke, R. Gallersdörfer, M.A. Jeusfeld, M. Staudt, S. Eherer (1995): ConceptBase – A Deductive Object Base for Meta Data Management. J. Intell. Inf. Syst 4(2):167-192, http://dx.doi.org/10.1007/BF00961873.

> The standard journal paper on ConceptBase.

[3] M.A. Jeusfeld (ed.): ConceptBase.cc User Manual Version 8.0. http://conceptbase.sourceforge.net/userManual80/

> The online version of the ConceptBase V8.0 User Manual. It defines the underlying Telos language (variant used in ConceptBase is called O-Telos). It also defines the logical sub-languages for deductive rules, integrity constraints and queries. The user manual is integrated with the ConceptBase Forum, a repository with many examples on using ConceptBase for various tasks.

[4] M.A. Jeusfeld (2009): Metamodeling and method engineering with ConceptBase. In Jeusfeld, M.A., Jarke, M., Mylopoulos, J. (eds): Metamodeling for Method Engineering, pp. 89-168, The MIT Press, 2009; Pre-print: http://conceptbase.sourceforge.net/mjf/Metamodeling-and-method-engineering-with-ConceptBase--preprint.pdf

> A book chapter focusing on the use of ConceptBase for metamodeling. ConceptBase was one of the first metamodeling tools, starting from 1988. Its flavor of metamodeling is strictly based on deductive rules (Datalog-neg) defined over predicates for instantiation, specialization, and relations/attributes. The chapter discusses the Yourdan method (a.k.a. Modern Strcutured Analysis) to show how integrity constraints can cross the boundarries of a single modeling language.

[5] M. Lankhorst et al. (2013): Enterprise Architecture at Work. Third Edition. Springer, Berlin, ISBN 978-3-642-29650-5.

> Standard reference for ArchiMate. See also the case study in chapter 3.

[6] K. Sandkuhl, J. Stirna, A. Persson, M. Wißotzki (2014): Enterprise Modeling - Tackling Business Challenges with the 4EM Method. The Enterprise Engineering Series, Springer 2014, ISBN 978-3-662-43724-7.

> Standard reference for 4EM. See also chapter 4.

[7] Dimitris Karagiannis, Robert A. Buchmann, Patrik Burzynski, Ulrich Reimer, Michael Walch: Fundamental Conceptual Modeling Languages in OMiLAB. In Dimitris Karagiannis, Heinrich C. Mayr, John Mylopoulos (eds.): Domain-Specific Conceptual Modeling - Concepts, Methods and Tools. Springer, 2016, pp. 3-30.