October 28-30, 2009
Dallas

Tool Integration – Eclipsefying CA Gen

Mustafa ARIKAN, Arikan Productivity Group GesmbH
mustafa.arikan@arikan.at

28th October 11:30 12:30

Biography

Mustafa Arikan studied industrial engineering, mathematics and computer science in Istanbul and in Vienna and finished his education in 1986. He has meanwhile 29 years industrial experience in IT and operations research. He worked for vendors like IBM and as technology partner of Computer Associates for various large scale companies and has won many IT awards throughout his career so far. His companies serve in Austria and Turkey and in cooperation with partners in over 10 countries mainly in software modernization.

Agenda

- Software Modernization
- Goal of Legacy Transformation
- Transformation Scenarios
- Transformation state-of-the-art
- Metamodel
- Metamodel Generation
- EMF
- CA Gen Models in Eclipse
- How to read and instantiate
- CA Gen Modernization
- PL1 + COBOL Modernization
- Conclusion

Software Modernization

Legacy Transformation, or legacy modernization, refers to the rewriting or porting of a legacy system to a modern computer programming language, software libraries, protocols, or hardware platform. Sometimes referred to as software migration, legacy transformation aims to retain and extend the value of the legacy investment through migration to new platforms.

Some parts of this presentation are taken from WIKIPEDIA.
www.wikipedia.org

Software Modernization – Legacy Code

A legacy code is any application based on older technologies and hardware, such as mainframes, that continues to provide core services to an organization. Legacy applications are frequently large and difficult to modify, and scrapping or replacing them often means re-engineering an organization's business processes as well. However, more and more applications that were written in so called modern languages like java are becoming legacy. Whereas 'legacy' languages such as Cobol are top on the list for what would be considered legacy, newer languages can be just as monolithic, hard to modify, and thus, be candidates of modernization projects.

Modernization – Program Transformation

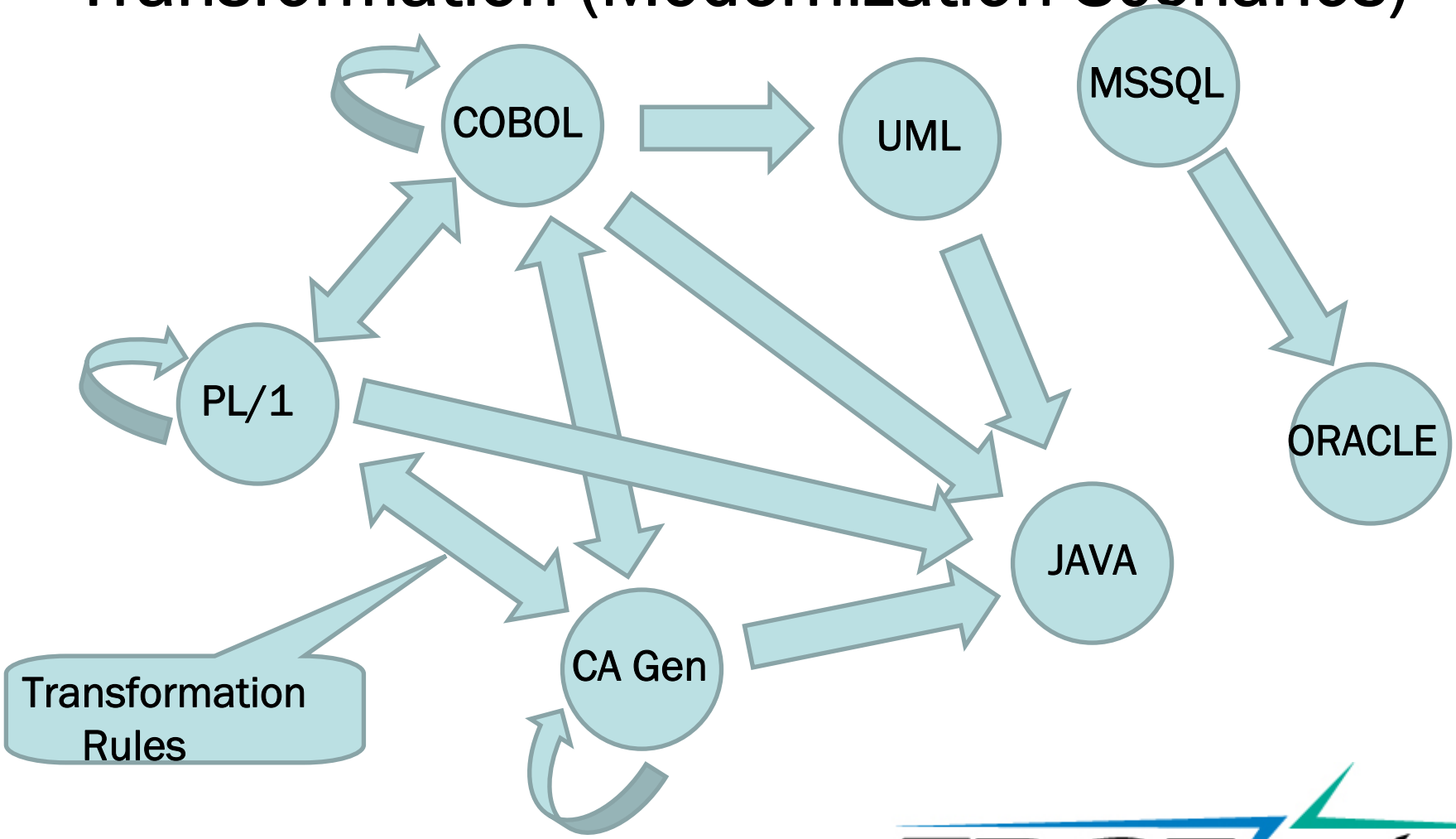
Re-implementing applications on new platforms in this way can reduce operational costs, and the additional capabilities of new technologies can provide access to functions such as web services and integrated development environments. Once transformation is complete and functional equivalence has been reached the applications can be aligned more closely to current and future business needs through the addition of new functionality to the transformed application. The recent development of new technologies such as program transformation by software modernization enterprises have made the legacy transformation process a cost-effective and accurate way to preserve legacy investments and thereby avoid the costs and business impact of migration to entirely new software.

Goal of Legacy Transformation

The goal of legacy transformation is to retain the value of the legacy asset on the new platform. In practice this transformation can take several forms. For example, it might involve translation of the source code, or some level of re-use of existing code plus a Web-to-host capability to provide the customer access required by the business. If a rewrite is necessary, then the existing business rules can be extracted to form part of the statement of requirements for a rewrite.

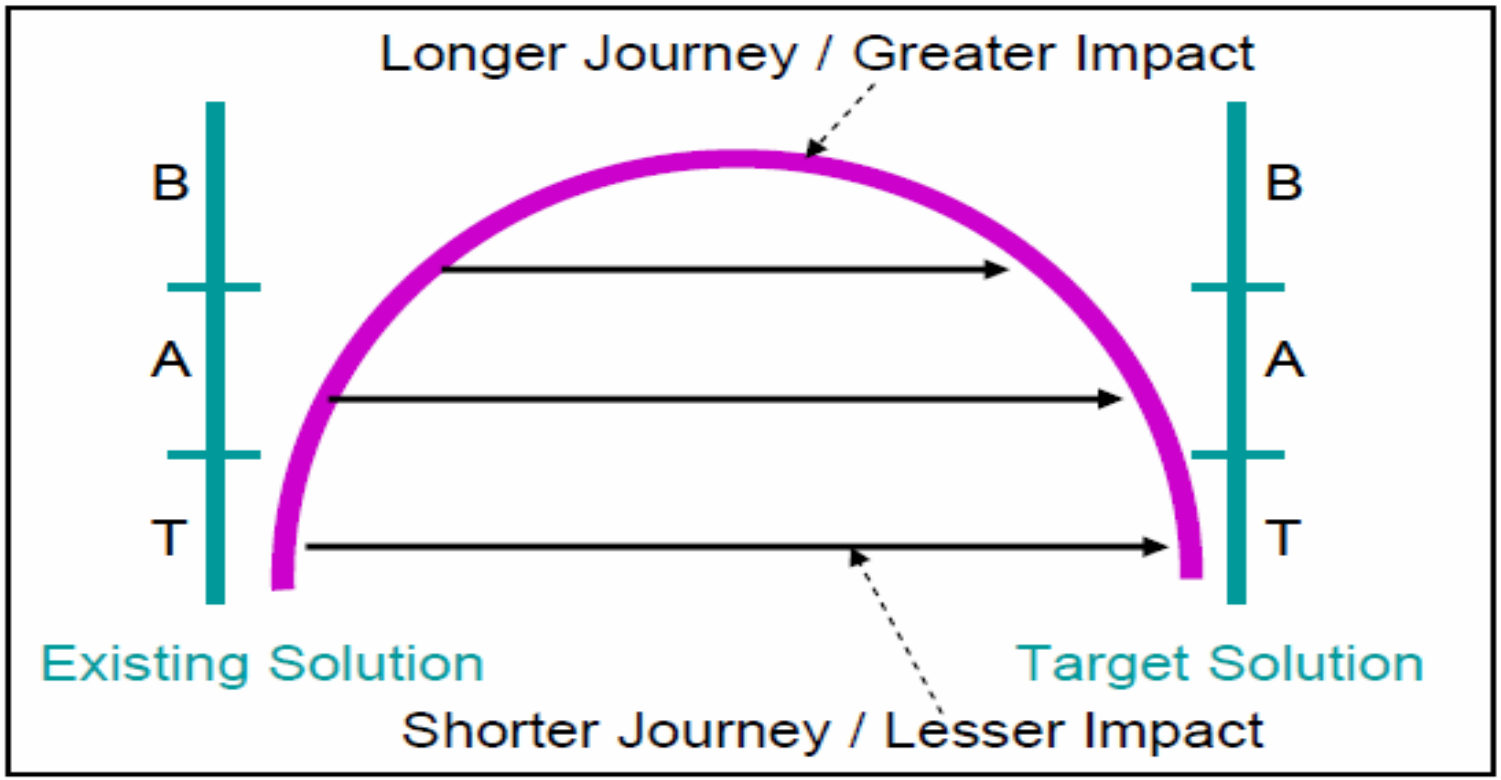
When a software migration reaches functional equivalence, the migrated application can be aligned more closely to current and future business needs through the addition of new functionality to the transformed application.

Transformation (Modernization Scenarios)



Transformation (state-of-the-art)

http://www.omg.org/docs/admtf/07-12-01.pdf



Metamodel

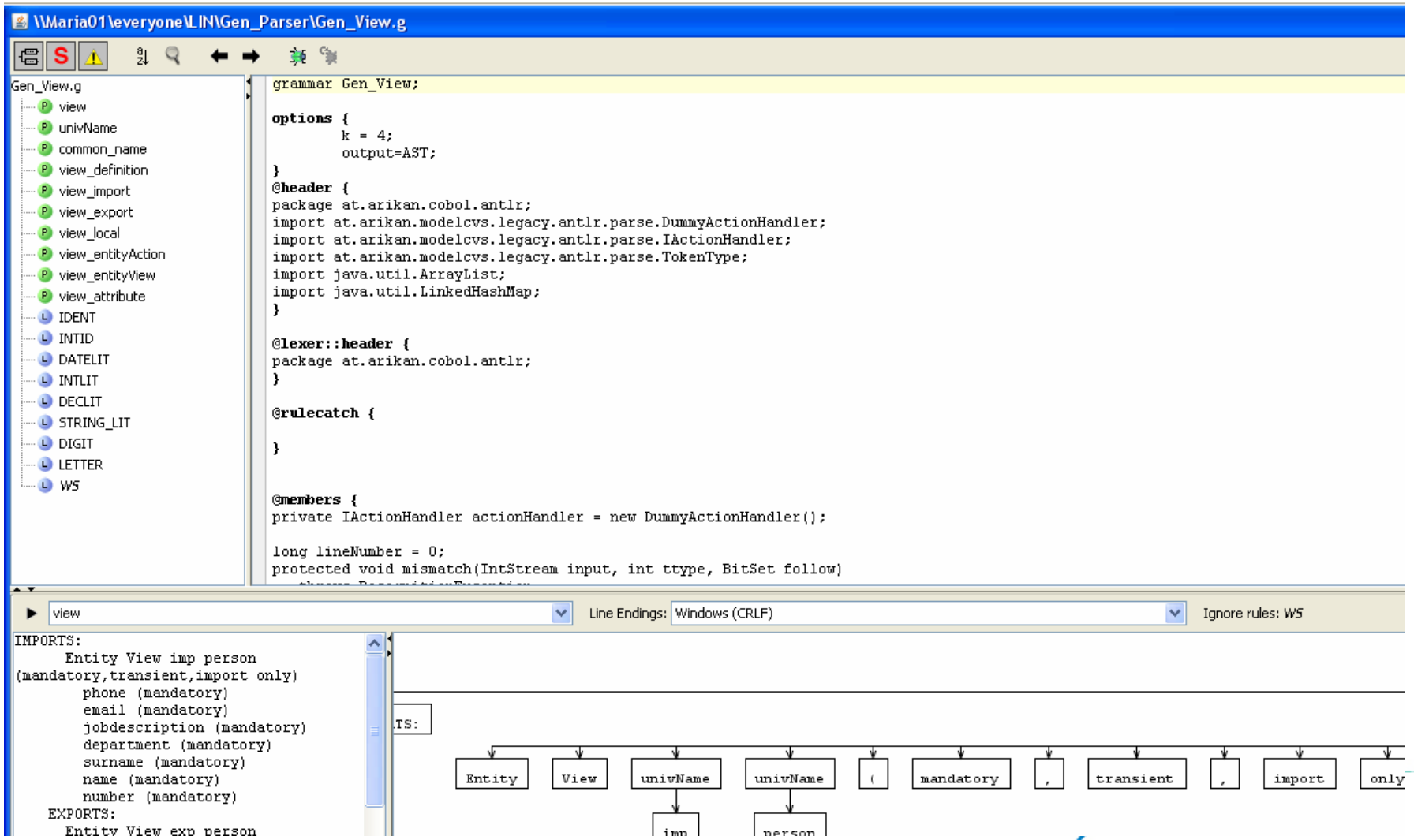
Metamodeling, or *meta-modeling* in software engineering and systems engineering among other disciplines, is the analysis, construction and development of the frames, rules, constraints, models and theories applicable and useful for modeling a predefined class of problems. As its name implies, this concept applies the notions of meta- and modeling.

Formal Grammar

A formal language is a set of *words*, i.e. finite strings of *letters, symbols, or tokens*. The set from which these letters are taken is called the *alphabet* over which the language is defined. A formal language is often defined by means of a formal grammar (also called its formation rules); accordingly, words that belong to a formal language are sometimes called *well-formed words* (or well-formed formulas).

A formal grammar (sometimes simply called a grammar) is a set of rules for forming strings in a formal language. These rules that make up the grammar describe how to form strings from the language's alphabet that are valid according to the language's syntax. A grammar does not describe the meaning of the strings—only their location and the ways that they can be manipulated

Metamodel Generation from Grammar



Eclipse Modelling Framework

Eclipse Modeling Framework Project (EMF)

The EMF project is a modeling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XMI, EMF provides tools and runtime support to produce a set of Java classes for the model, along with a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor.

EMF builds include XML Schema Definition (XSD), now a component of the Model Development Tools (MDT) project, and an EMF-based implementation of Service Data Objects (SDO). XSD provides a model and API for manipulating components of an XML Schema, with access to the underlying DOM representation of the schema document.

Other subprojects, such as Model Query, Model Transaction, and Validation Framework, are available separately.

ECORE

EMF (Core)

EMF consists of three fundamental pieces:

- **EMF** - The core EMF framework includes a meta model (Ecore) for describing models and runtime support for the models including change notification, persistence support with default XMI serialization, and a very efficient reflective API for manipulating EMF objects generically.
- **EMF.Edit** - The EMF.Edit framework includes generic reusable classes for building editors for EMF models. It provides
  - Content and label provider classes, property source support, and other convenience classes that allow EMF models to be displayed using standard desktop (JFace) viewers and property sheets.
  - A command framework, including a set of generic command implementation classes for building editors that support fully automatic undo and redo.
- **EMF.Codegen** - The EMF code generation facility is capable of generating everything needed to build a complete editor for an EMF model. It includes a GUI from which generation options can be specified, and generators can be invoked. The generation facility leverages the JDT (Java Development Tooling) component of Eclipse.

Levels of Code Generation

Three levels of code generation are supported:

- **Model** - provides Java interfaces and implementation classes for all the classes in the model, plus a factory and package (meta data) implementation class.
- **Adapters** - generates implementation classes (called ItemProviders) that adapt the model classes for editing and display.
- **Editor** - produces a properly structured editor that conforms to the recommended style for Eclipse EMF model editors and serves as a starting point from which to start customizing.

All generators support regeneration of code while preserving user modifications. The generators can be invoked either through the GUI or headless from a command line.

Making the CA Gen model available

- CA Gen model is being exported using the CA Gen APIs..
- The exported model is being converted into a CA Gen instance.
- The metamodel instance can be viewed with a CA Gen viewer

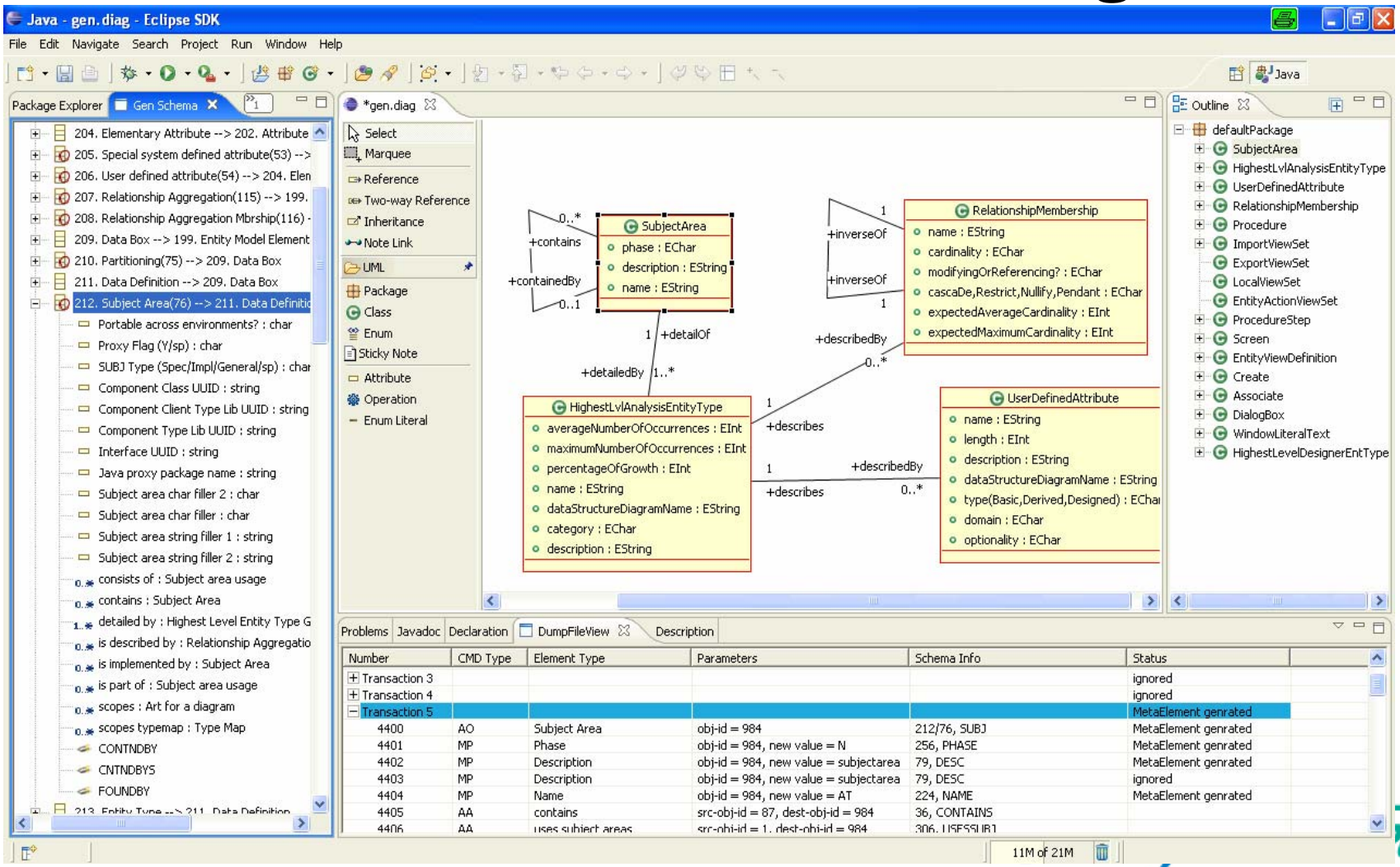
How to read and instantiate Gen MM

CA Gen has more than 800 model objects, which have to be read and persisted in the correct sequence.

CA Gen metamodel on the one hand and Ecore conforme CA Gen model on the other hand must be created.

We need here a tool which supports the creation of the MM .. This tool is ...

.. the so called CA Gen metamodeling toolkit.



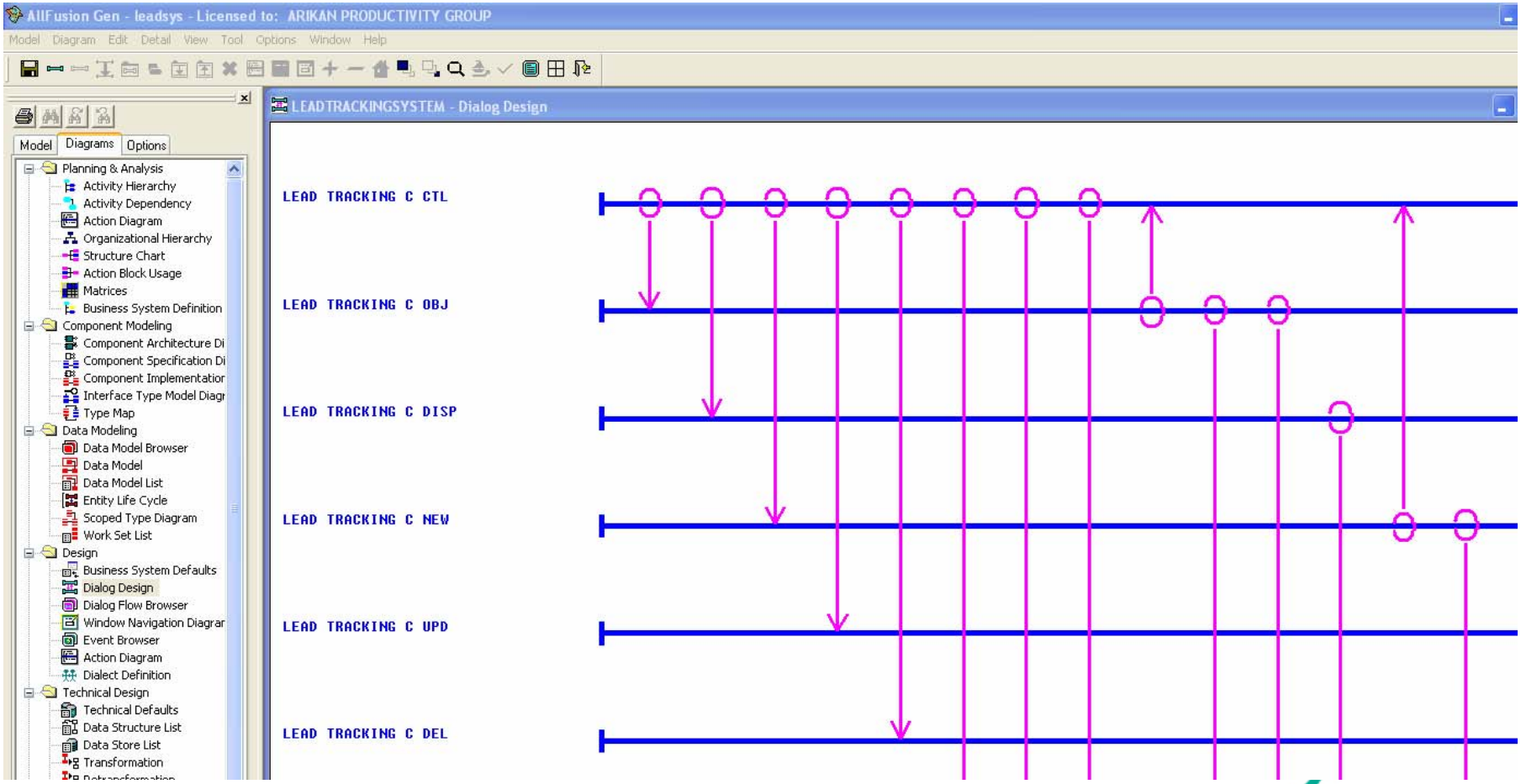
CA Gen MM Toolkit - Functionality

The CA Gen object decomposition diagram, the generic CA Gen metamodel and the API s are the starting point...

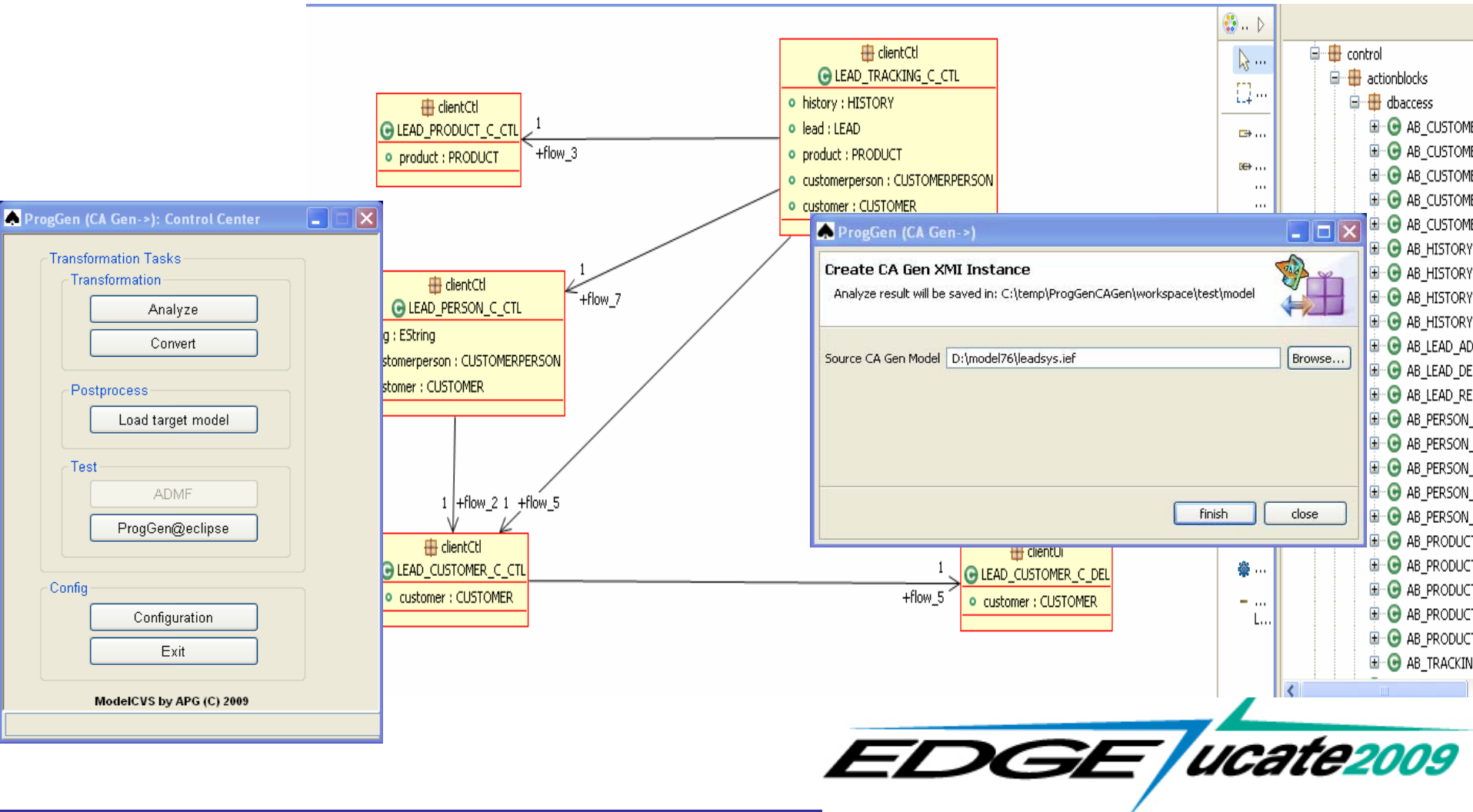
APG metamodelling Toolkit supports the generation of Ecore conform metamodel classes, which map various domains of CA Gen, like Action Diagrams or GUI ...

The metamodeling Toolkit also generates adaptors for model I/O.

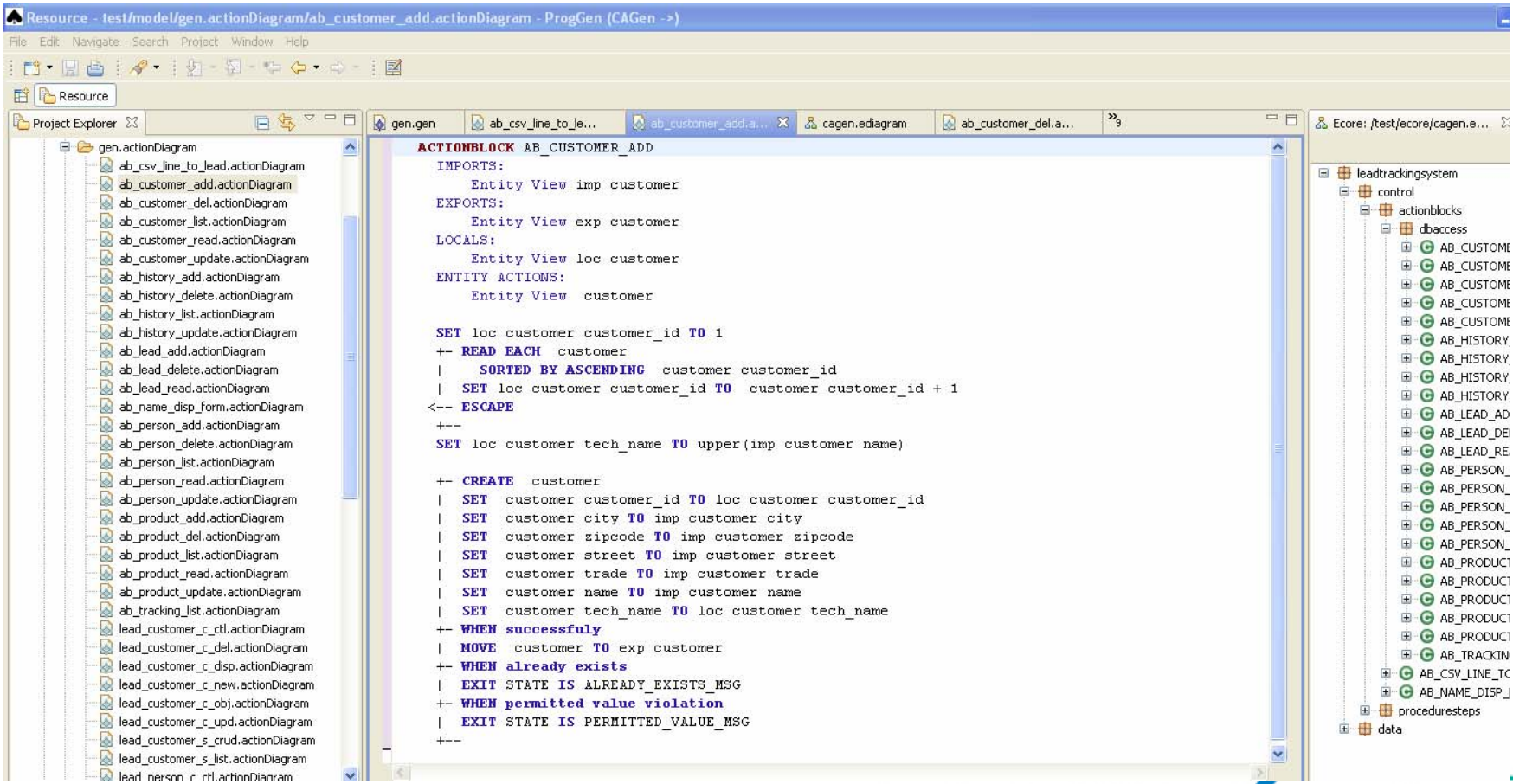
Let ‘s open a CA Gen model..



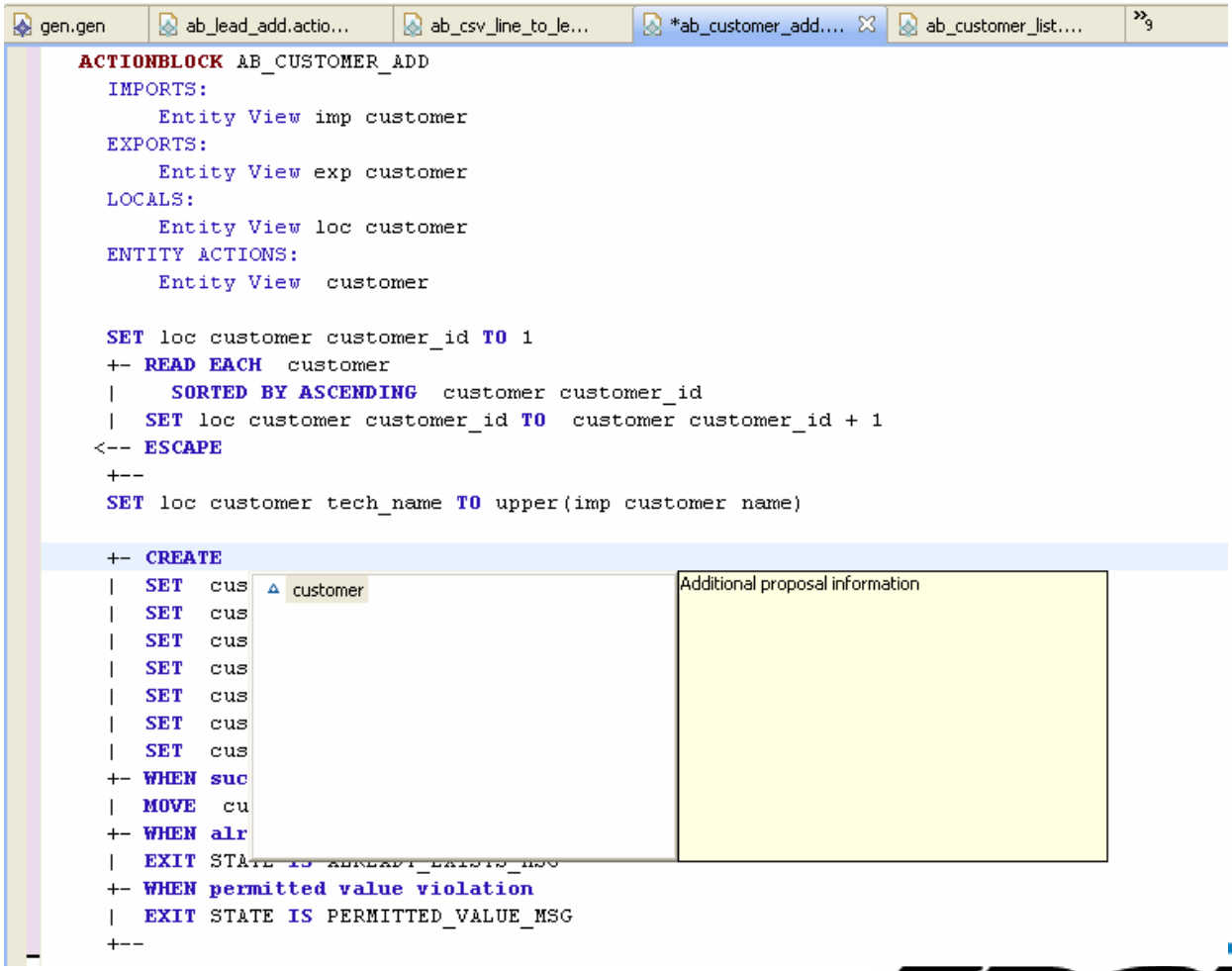
..ProgGen transfers the model into eclipse...



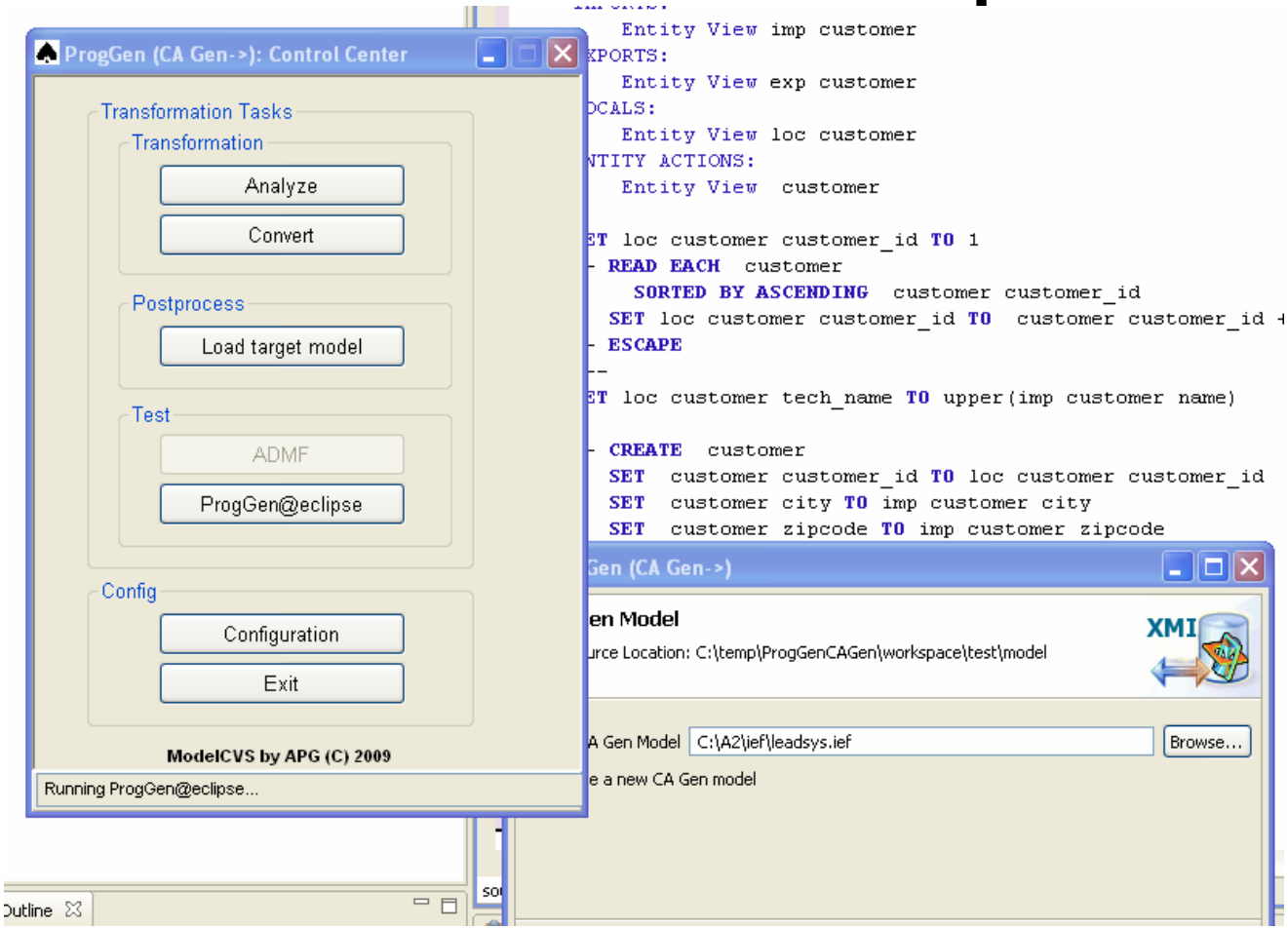
.. let us write some new action block in eclipse ..



.. using the code completion editor..



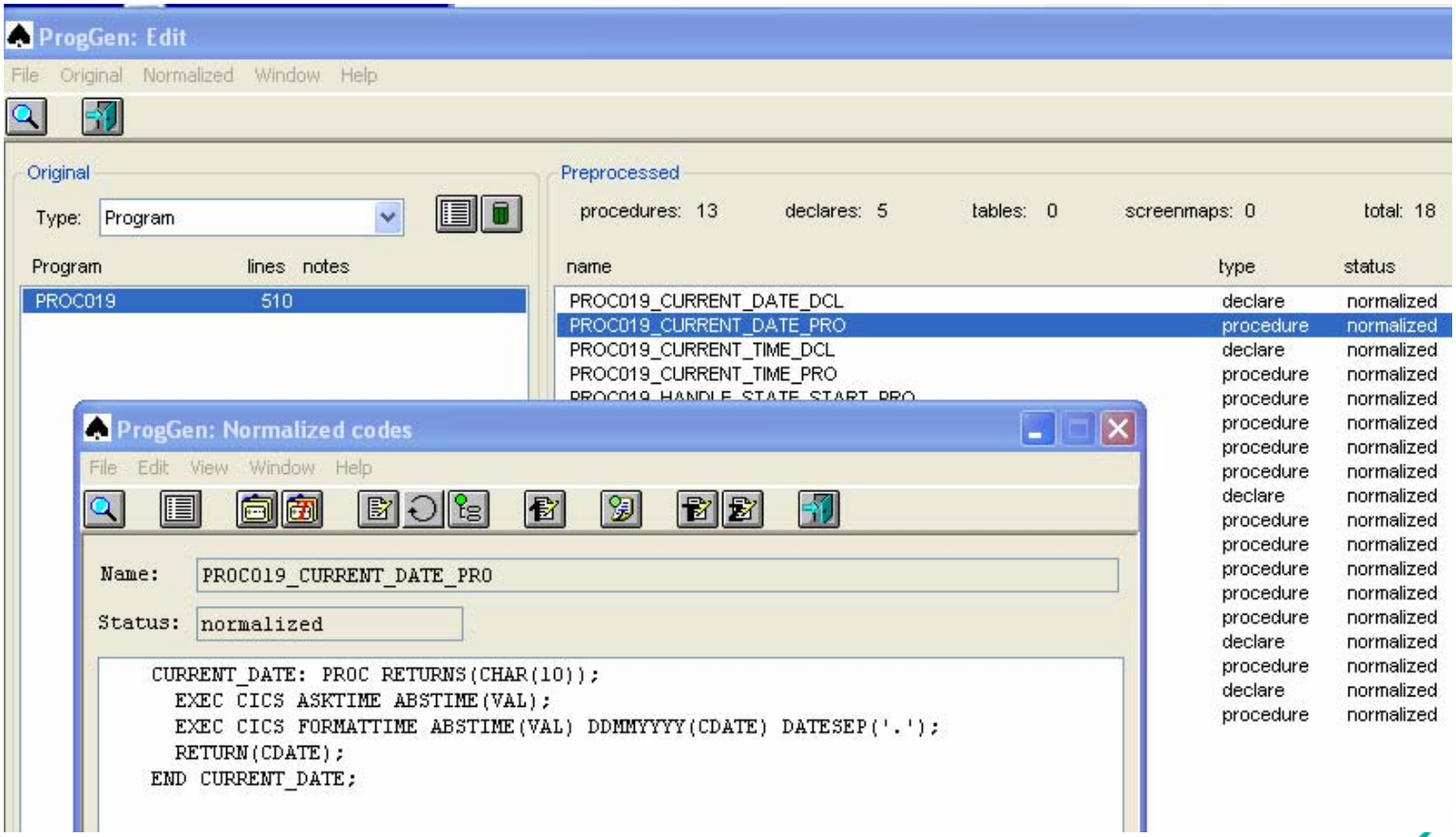
..save the model in eclipse...



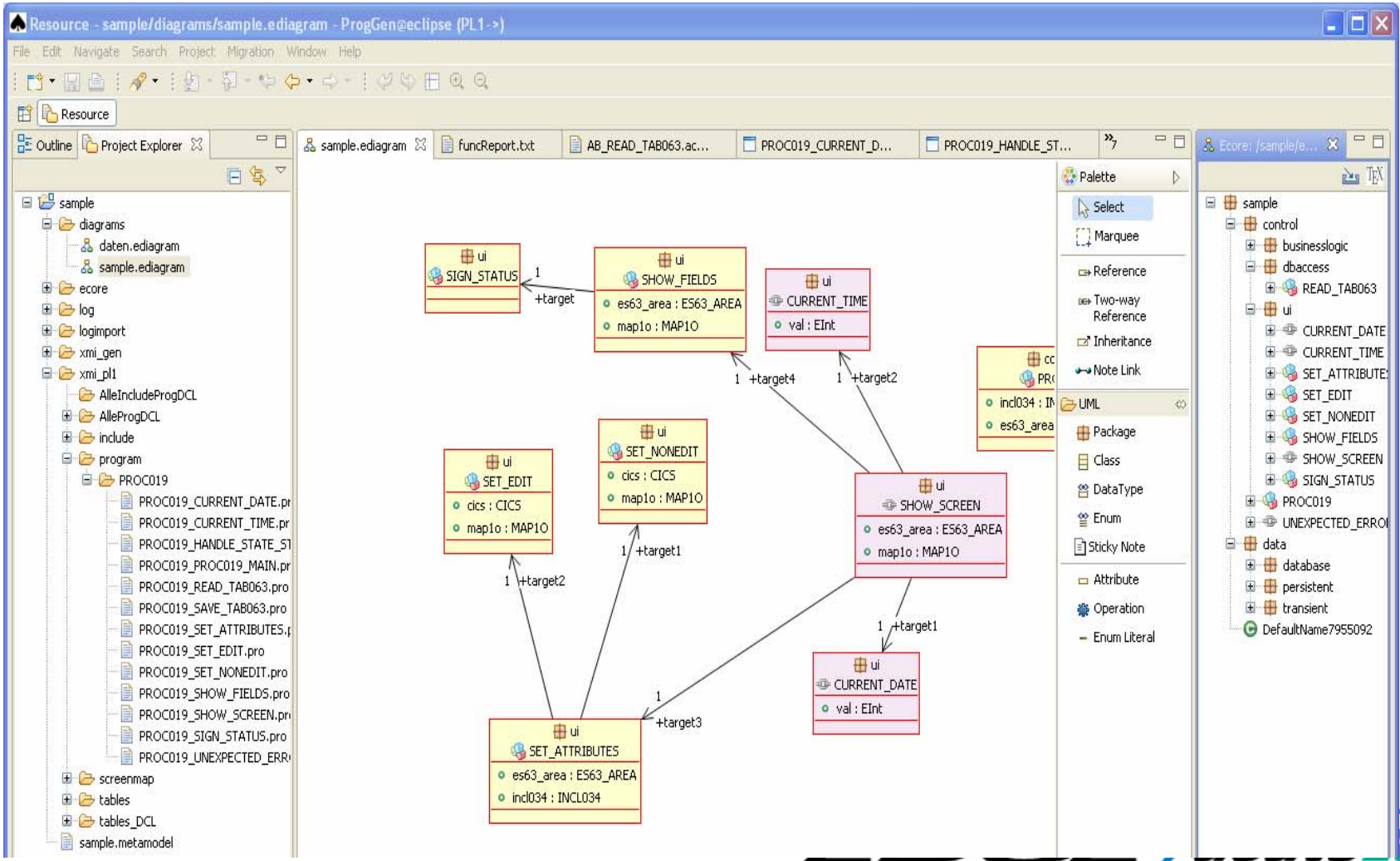
..and then transfer it back into CA Gen..

```
AB_CUSTOMER_ADD
  IMPORTS: ...
  EXPORTS: ...
  LOCALS: ...
  ENTITY ACTIONS: ...

SET loc customer customer_id TO 1
  READ EACH customer
       SORTED BY DESCENDING customer customer_id
    SET loc customer customer_id TO customer customer_id + 1
  ESCAPE

SET loc customer tech_name TO upper(imp customer name)

  CREATE customer
    SET customer_id TO loc customer customer_id
    SET city TO imp customer city
    SET zipcode TO imp customer zipcode
    SET street TO imp customer street
    SET trade TO imp customer trade
    SET name TO imp customer name
    SET tech_name TO loc customer tech_name
  WHEN successful
    MOVE customer  TO exp customer
  WHEN already exists
    EXIT STATE IS already_exists_msg
  WHEN permitted value violation
    EXIT STATE IS permitted_value_msg
```

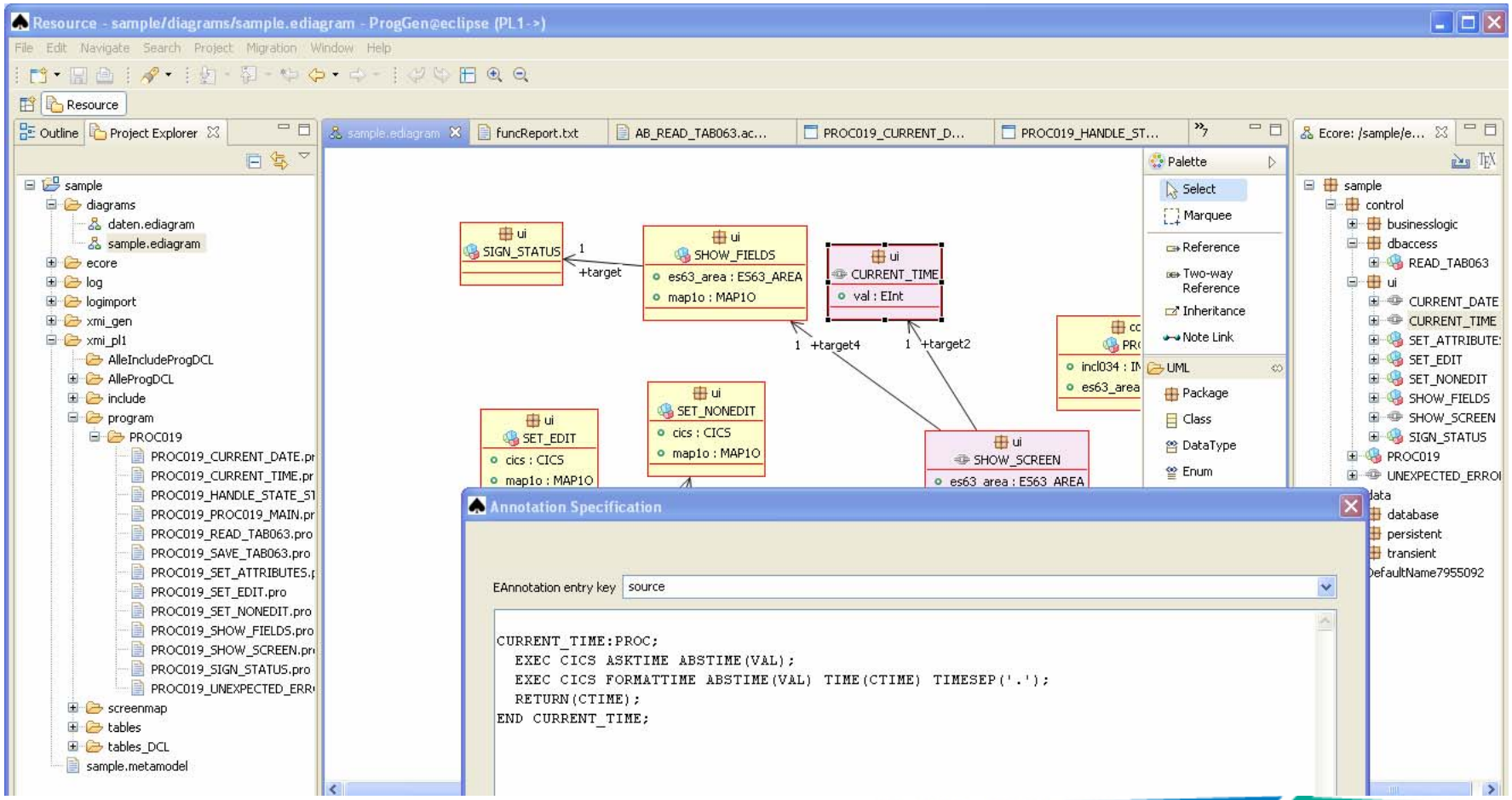
COBOL & PL1 Programs



...PL1 & COBOL both are imported into eclipse...



... they can be manipulated in eclipse ...



… what kind of manipulation …

Any kind of model restructuring is possible. Action blocks can be separated into other action blocks.

Data structures can be changed.

Data model changes can be made.

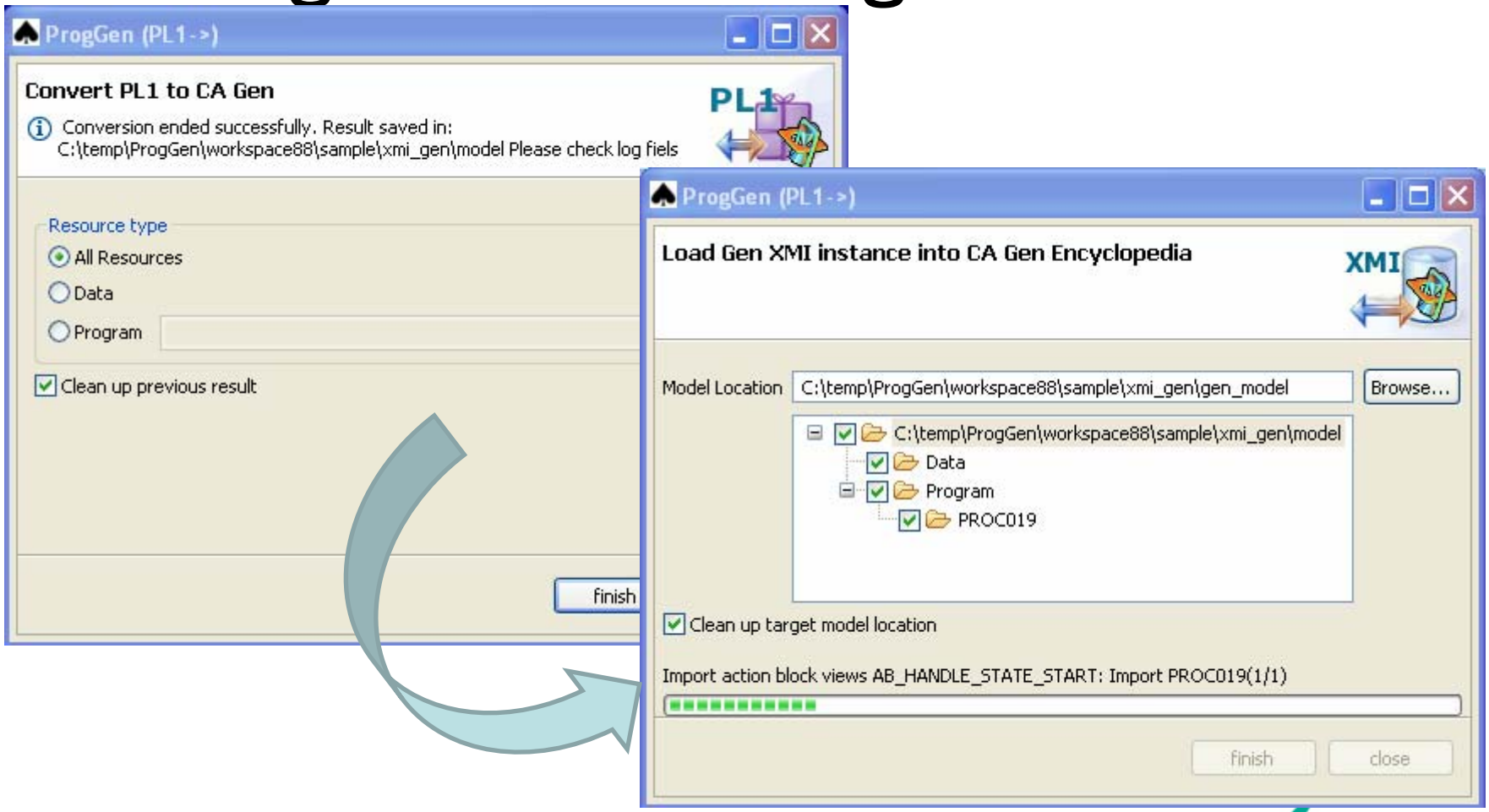
A model can be separated into smaller models.

QS check can be done.

Software can be modernized via CA Gen.

… and many others…

.. and again can be brought into CA Gen ..



Conclusion

- Eclipse Modeling Framework enhances the capabilities of CA Gen
- Ca Gen models can be brought into eclipse environment and be used fully or partly for further generations and analysis
- CA Gen models can be modernized
- PL1 or COBOL and soon ADABAS Natural models can be modernized and brought into CA Gen

For more information

www.modelcvs.com

twitter.com/modelcvsbyapg

join our news group in Xing

"SOftware MOdernization and MOdelbased SOftware DEvelopment"

xing.com/net/somomosoen

Q&A