OMG

MDA Overview

by Sparx Systems

http://www.sparxsystems.com

Trademarks

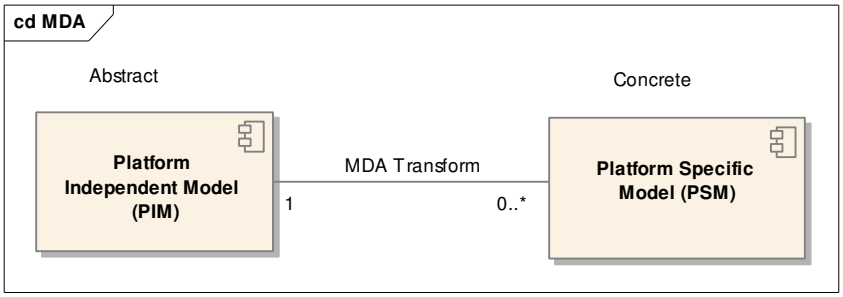
Table of Contents

Introduction - Model Driven Architecture

The OMG's Model Driven Architecture initiative is aimed at increasing productivity and re-use through separation of concern and abstraction. A Platform Independent Model (PIM) is an abstract model which contains enough information to drive one or more Platform Specific Models (PSM). Possible PSM artifacts may include source code, DDL, configuration files, XML and other output specific to the target platform.

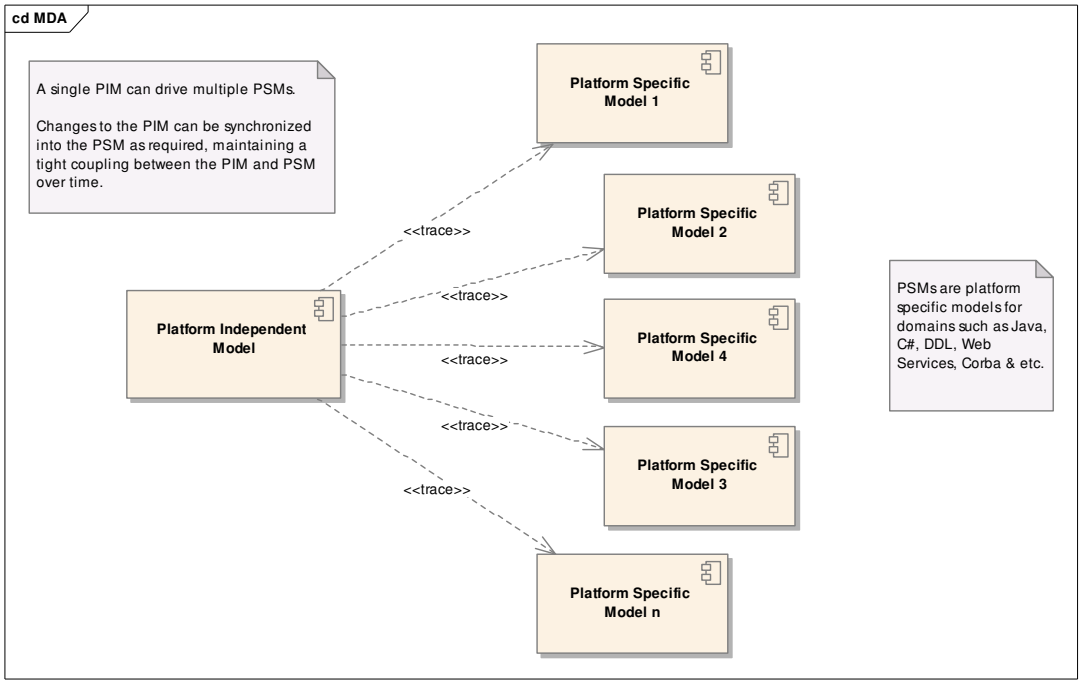
MDA aims to enhance portability by way of separating system (abstract) architecture from platform (concrete) architecture. Platform Independent Models describe the structure and function of a system, but not the specific implementation.

MDA has the capability to define transformations that map from PIMs to PSMs. This facilitates the development of a system in abstraction, and simplifies implementation of that system across a variety of target platforms.



For example, an MDA Transform from a PIM to a DDL will create DDL table elements from a class, whereas the same class transformed to an EJB Entity Bean will result in a package containing the class and interface elements required by EJB.

Enterprise Architect (EA) helps you manage such transformations and even write your own transformation rules for any language. It will also aid you in keeping as many Platform Specific Models as you need synchronized to a single Platform Independent Model. Enterprise Architect has built-in support for MDA transforms, to C#, DDL, EJB, Java and XSD.



MDA Commitment and Current Toolset in Enterprise Architect

Sparx Systems is committed to developing and evolving MDA support within their UML 2.1 based modeling tool, Enterprise Architect. Enterprise Architect offers a number of tools targeted at MDA driven development, including a comprehensive PIM to PSM Transformation Engine, allowing modelers to target multiple PSMs from a single PIM – and to synchronize PIM changes into the PSM on demand.

Coupled with code generation templates, UML Profiles, UML Patterns, Frameworks, extensive reverse engineering capabilities and a complete automation API for scripting and extending, Enterprise Architect provides a solid and extensible MDA platform.

The core MDA component in EA is a template based transformation engine that generates PSM model elements from a PIM source. The templates allow for generation of highly specific PSM artifacts from the PIM. The MDA Transformation templates are similar to, and have the same syntax as, the code generation templates within EA.

Summary of MDA and Model Driven Generation Support in Enterprise Architect

- Built-in Transforms for DDL, EJB, Java, C# and XSD
- Transform templates for defining user transforms
- Languages supported in base version: C++, Java, C#, VB.Net, Visual Basic, PHP, Delphi
- Plug-ins for CORBA, CIM and Python available
- Support for pluggable technologies through
    - UML Patterns
    - UML Profiles
    - Code generation templates
    - Code parsing grammars
    - PIM to PSM Transformation Templates

Details of Built-In Transforms:

- **DDL:** Transforms platform-independent class elements to platform-specific table elements.
- **EJB Entity:** Transforms platform-independent class elements to packages containing the class and interface elements that comprise an EJB Entity Bean.
- **EJB Session:** Transforms platform-independent class elements to packages containing the class and interface elements that comprise an EJB Session Bean.
- **Java:** Transforms platform-independent elements to Java language elements.
- **JUnit:** Converts a Java model to a model where test methods are created for each public method of any original class.
- **C#:** Converts a PIM to a standard C# implementation set.
- **NUnit:** Converts a .Net language specific language specific model where test methods are created for each public method of any original class.
- **WSDL:** Converts a simple representation of a WSDL interface into the elements required to generate that interface.
- **XSD:** Transforms platform-independent elements to XSD elements.

Database Modeling

- Extensive support for modeling database systems and generating DDL.
- Support for reverse engineering a range of popular database systems via ODBC, including Oracle, MS Access, SQL Server, PostgreSQL, Progress OpenEdge, and more.

Access to Model Information Outside EA

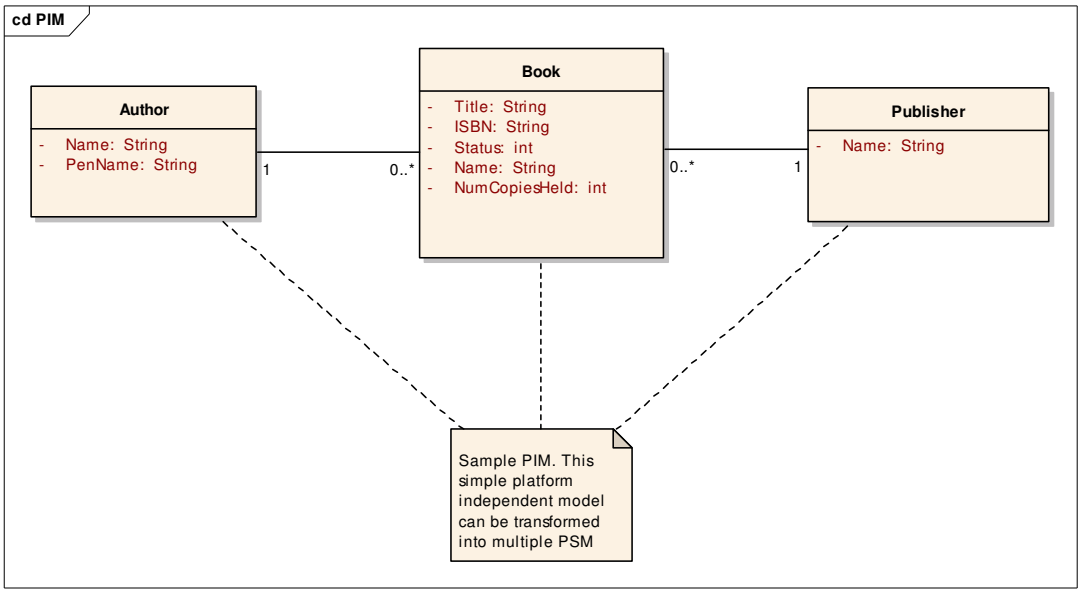
- Support for XMI import/export to other tools (modeling tools and/or third party tools such as AndroMDA).
- Extensive automation API for accessing model elements and diagrams.
- CSV import/export feature for simple data.
- Standard relational database repository structure.

A Quick Ooverview of the MDA Transformation Process in Enterprise Architect

The following pages show some details of a PIM to PSM transform and the result. The opportunity exists within EA to provide an end-to-end modeling environment that targets multiple PSMs from a single PIM and synchronizes changes from the PIM forward as model changes are made.

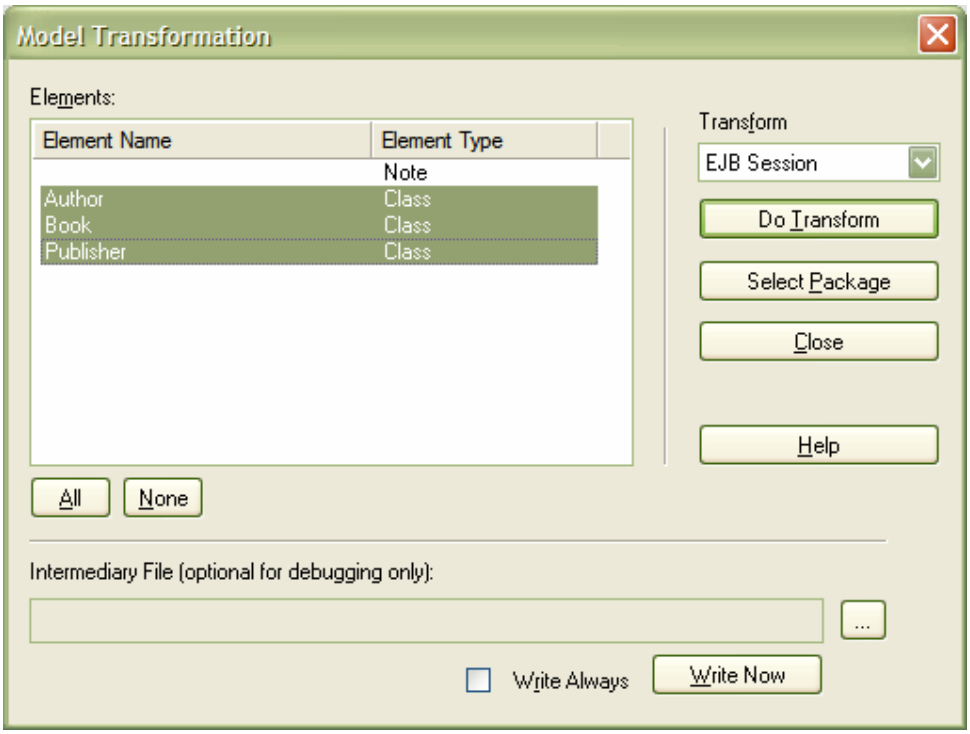
The Platform Independent Model

This diagram shows a simple PIM with three classes, showing the abstract properties and relationships between an Author, their Books and their Publisher. This PIM has no platform specific properties, but is truly an abstract representation.

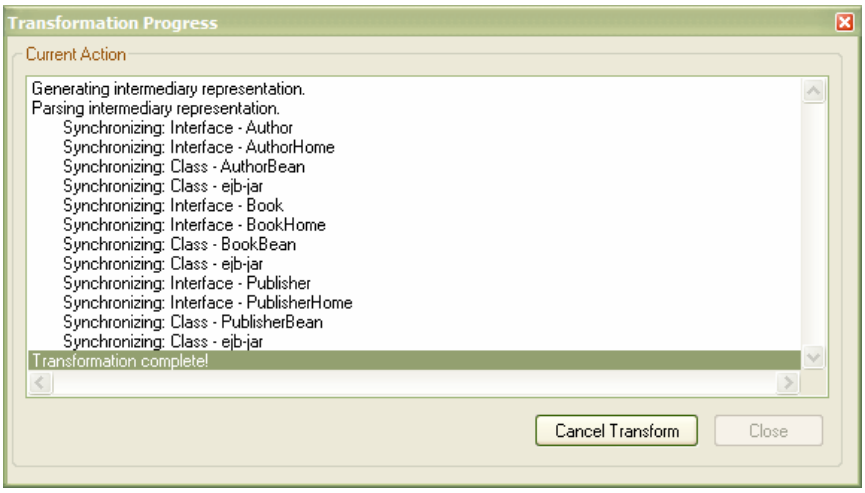


Select and Start the Transform

Selecting the three classes in EA and choosing "Transform Elements" from the context menu provides the following dialog. Here you may choose the PIM elements to transform, the transformation to be invoked, and a target package, if necessary. You can select from the inbuilt transformations, create your own transformations, or use those supplied by third parties.

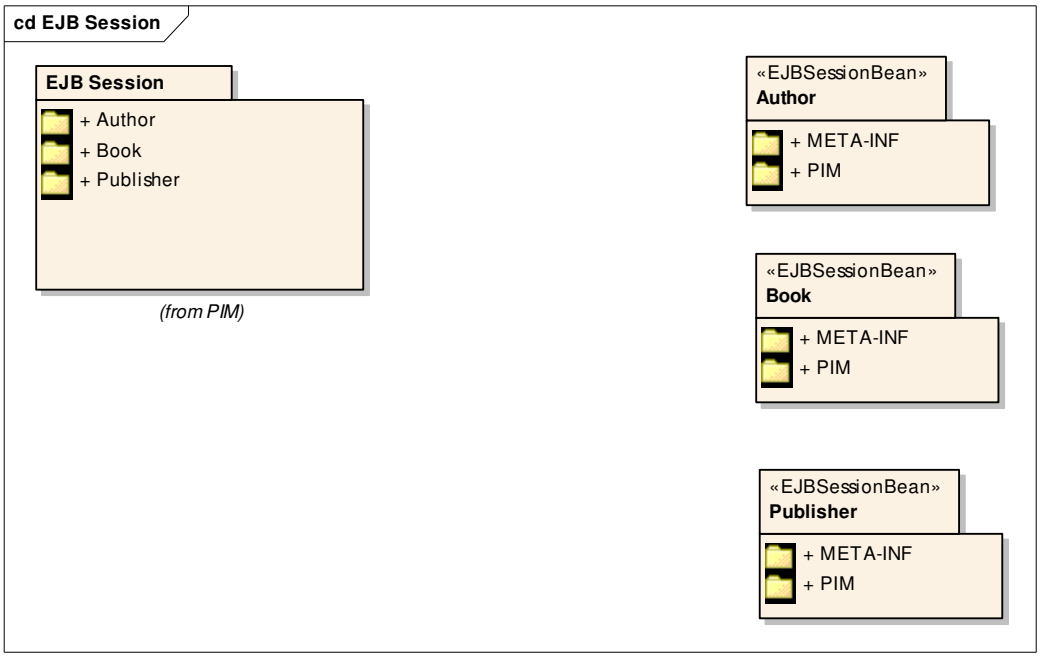


The Transform in Progress



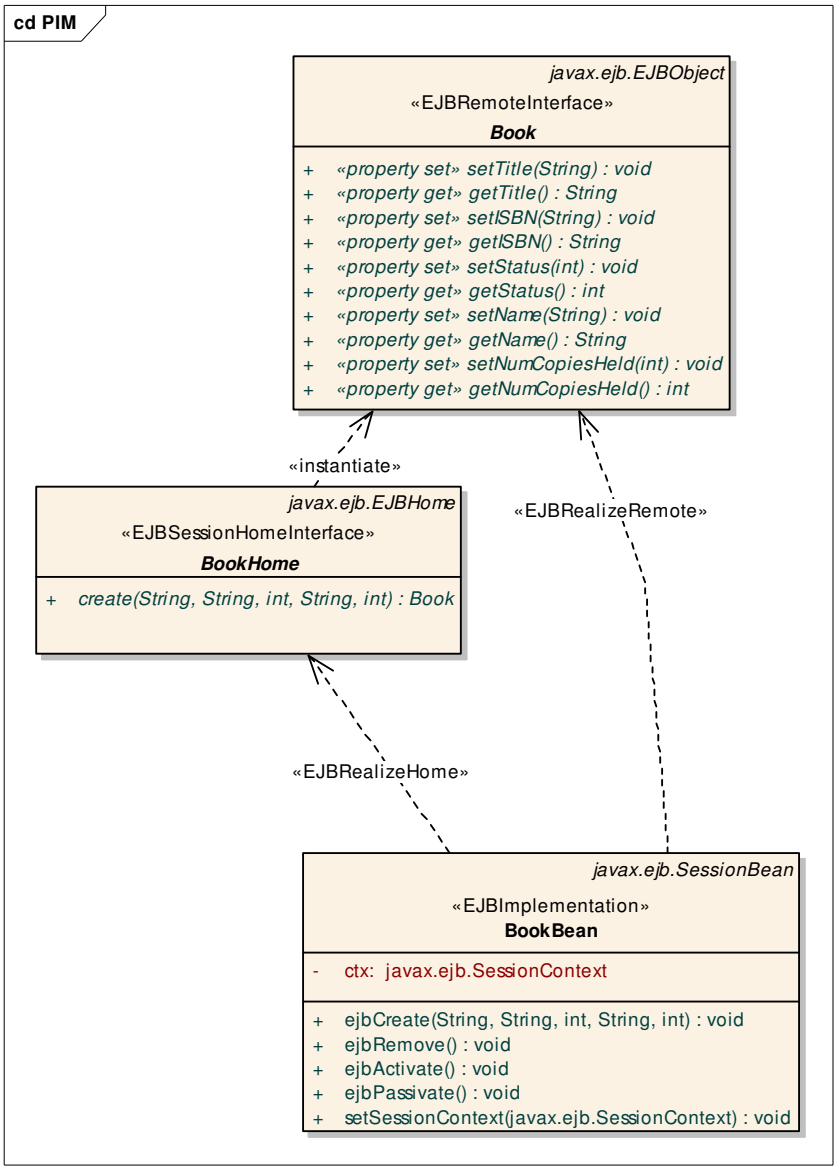
Packages Created During Example Transform

For the EJB Session transform, this diagram shows the PSM packages created – one for each EJB Session bean created from the base PIM classes.



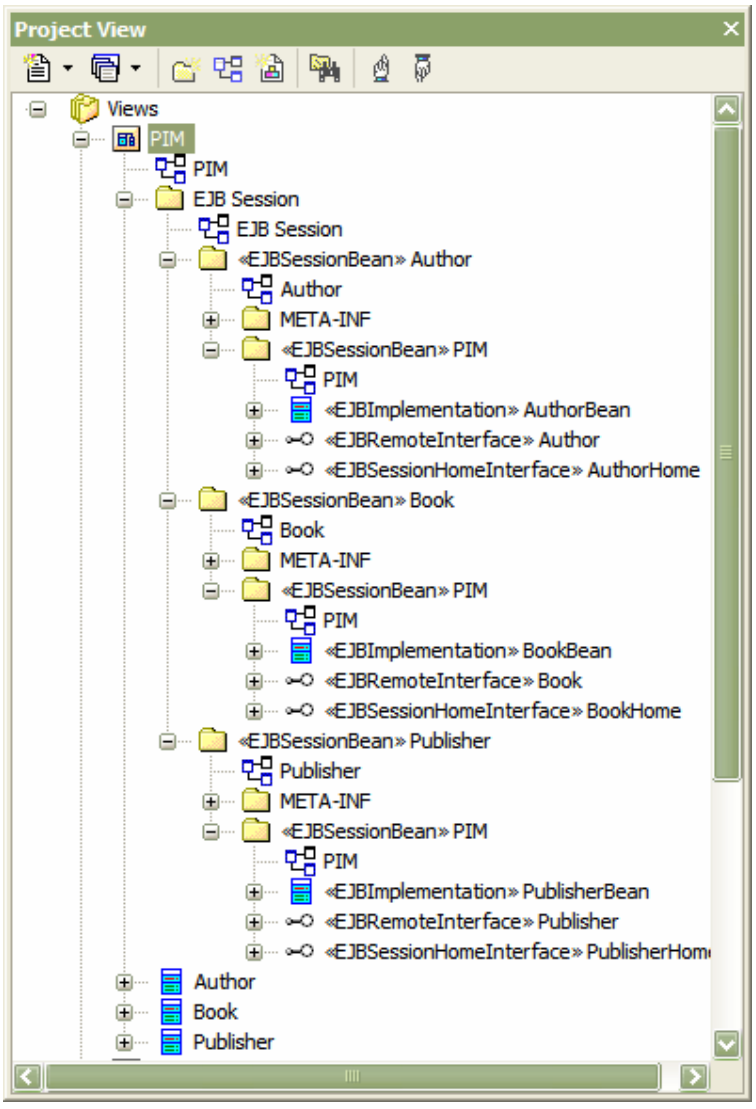
Example Bean Created from MDA Transformation

This diagram shows an example of the EJB Session Bean PSM. As a PSM is tightly bound to the target domain, Java and EJB specific properties, code and settings are included in the model. These are logically derived during the transform according to the rules of the transform template.



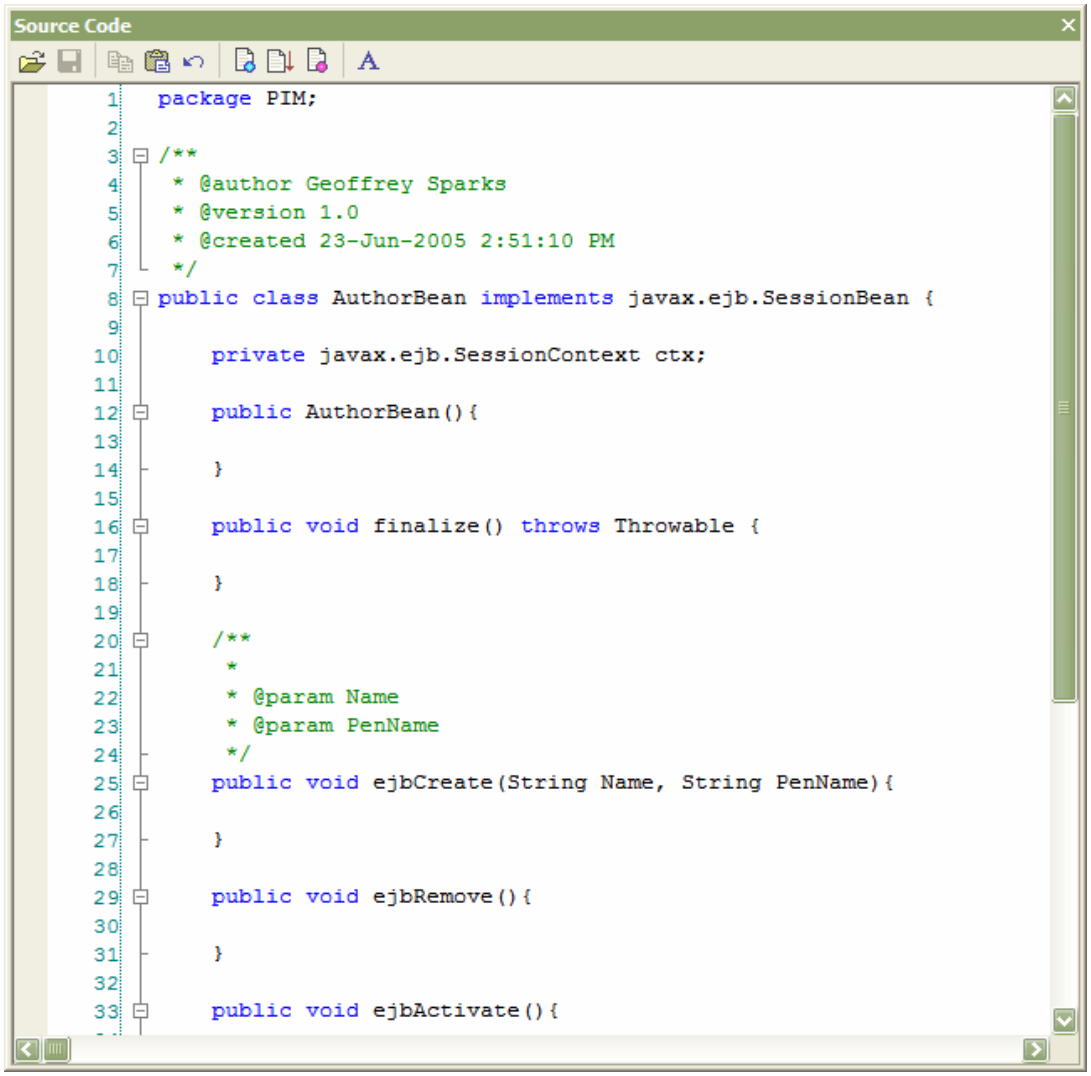
Project View of Artifacts Generated from Transform

This view shows the set of artifacts generated into the model during the transform. Note that changes to the PIM can be forward synchronized at any time, resulting in changes to all the affected PSMs during the next Transform process. This allows the PIM to drive development of the PSM, and for the PIM and PSM to be synchronized at all times.



Source Code Generated from PSM Using Automatic Code Generation

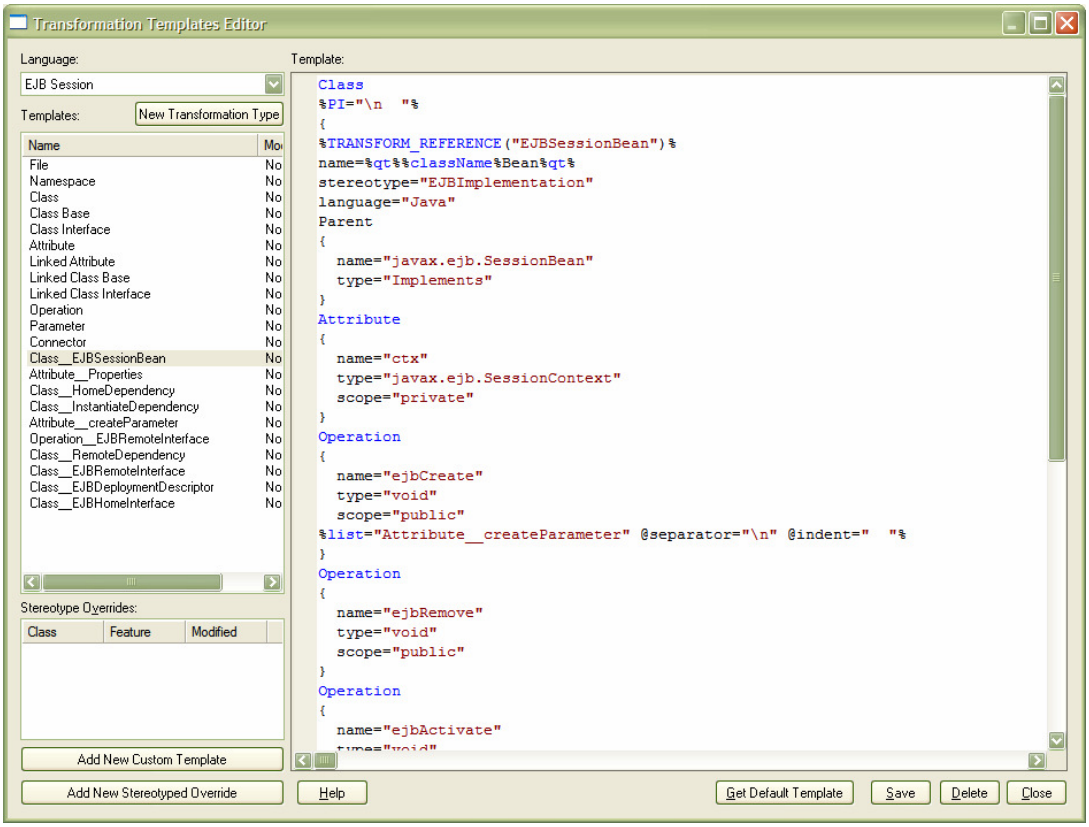
As the PSM contains specific platform properties, the source code generated can be tailored to the target platform quite tightly. This results in high quality code requiring less manual work and output that is in line with your company's coding standards.



Creating and Modifying Transforms: The MDA Template Editor

Modifying the transform templates is no more difficult than modifying the code generation templates in EA. The template language is quite expressive and allows for detailed transformations from the PIM to PSM context.

Investing some time in writing detailed transforms for your target platforms can provide a high ROI when it comes to re-using code and generating the basic structure of your application from the model, rather than by hand.



Web References

For a summary of MDA resources and MDA style transforms in EA see:

http://sparxsystems.com.au/resources/mda/index.html

For an overview for writing transformations see:

http://sparxsystems.com.au/resources/mda/writing_transformations.html

Information on Extending Enterprise Architect for MDA and Software Factories

EA features a number of ways whereby users and third party developers can plug-in additional behavior to enhance the existing functionality. Sparx Systems has used this mechanism to build interfaces from EA to Visual Studio.NET™ and Eclipse™, as well as CORBA, CIM and Python add-ins.

The following table details major extensibility mechanisms. Further information is contained within the EA Help file and by email from support@sparxsystems.com.au

Mechanism	Description
UML Patterns	Patterns are snapshots of interacting elements which may be pasted into a model – or applied to existing model elements to extend their functionality. EA supports saving and loading pattern libraries. Stored in XML format and imported into EA.
UML Profiles	Profiles are sets of stereotyped elements (including attributes, operations, links and constraints). Stereotypes typically have custom tagged values. When a stereotype is applied to a model element, the tagged values, constraints and any alternate image renderings are also copied over. EA supports building Profiles according to the UML 2.0 recommendation. Stored in XML format and imported into EA.
Code Generation Templates	EA has an extensive Code generation template language. Code templates may be written to replace the existing ones within EA or to provide code generation for additional languages.
Model Transformation Templates	EA supports model transformation templates. These allow MDA PIMs to be transformed into PSMs and forward synchronized on demand. The template language is much the same as the code generation templates.
Configurable reverse engineering grammars (coming soon)	EA has a powerful grammar driven reverse engineering capability. Currently this is in use, but details and tools for writing grammars are not available. In the near future we hope to distribute suitable tools and information for building new Grammars for custom

Mechanism	Description
	languages
Frameworks	Like most modeling tools, EA supports loading large frameworks into models to leverage re-use assets (eg. Java SDK, .NET SDK and others). Frameworks are stored in XMI format.
Tagged Values/Stereotypes and other Reference Data	EA supports saving and loading of reference data and basic types in XML format and loading into other models.
MDG Technology	MDG Technology files wrap up Profiles, Patterns, Templates, Reference Data and more into a convenient single file for distribution to users and clients. A single Technology file might support a language such as Python for example – or a technology such as Web Services.
Automation API	The Automation API supports any ActiveX client and allows access to the internal model elements. In addition to element access, the API supports a number of convenience functions such as XMI import/export and loading Technology Files.
Plug-in Architecture	The plug-in architecture extends the Automation API and offers the ability to present menus to the user and respond to user selections. It also lets a plug-in take over a main window "diagram tab" for any purpose – eg. reporting, metrics, or code generation, etc.
MDG Link	MDG Link is a plug-in extension for associating a model package hierarchy with a specific plug-in tool (eg. MDG Link for VS.Net) and presenting a common set of menu options that a tool can implement (eg. merge, build, run, etc.)
XMI I/O	Standard XMI export and import functions provide the ability to export model segments and process in a third party tool. The resultant target may be generated code, data models, XMI for other tools or one of many other possibilities. EA supports saving and loading XMI from the automation API, allowing automated processing of models by XMI based tools.

Recommended Reading

Raistrick, Chris; Colin Carter, Paul Francis; Ian Wilkie; John Wright. *Model Driven Architecture with Executable UML*. Cambridge University Press, 2004. ISBN: 0-521-53771-1