Innovator 2008

Features

The Integrated Modeling Platform for Business Process and Software Engineering

MID
the modeling company

Contents

Overview Innovator Editions ... 1
- Features ... 1
- Innovator – Integrated Modeling Platform for Future-Proof Software ... 3
  - Editions ... 3
  - Integration ... 3
  - General Functionalities ... 3
  - Standards ... 3
  - Customized Software Production Environments ... 3
- Architecture ... 4
  - Modular and Open ... 4
  - Client-Server Architecture ... 4
  - Administration ... 4
  - Licensing ... 5
  - License Distribution ... 5
- Hardware and Software Basis ... 6
  - Supported Platforms ... 6
  - Memory Requirements ... 6
  - Hard Disk Space Requirements ... 6
  - Network Operation ... 6
  - Graphical User Interfaces ... 6

User Interface ... 7
- Operation and Menus ... 7
  - Look & Feel ... 7
  - Windows Standard ... 7
  - Multi-Language Capabilities ... 7
- User Interfaces ... 7
  - Log-In ... 7
  - Model Browser ... 8
  - Graphical Editors ... 8
  - Text Editor ... 8
  - Instructions and Labels Area ... 9
  - Templates ... 9
  - Configurable Help Menu ... 9
  - Settings ... 9

General Functionalities of Innovator ... 10
- Functionality ... 10
  - Profiles ... 10
  - Model Management ... 10
  - Namespaces ... 11
  - Labels ... 11
  - Specifications ... 12
  - Generations ... 12
  - External Objects ... 12
  - Search ... 13
- Consistency and Quality Assurance ... 13
  - Data Consistency ... 13
  - Context Sensitivity ... 13
  - Methodological Verification ... 13
  - Configuration ... 14
  - Documentation ... 14
- API ... 14
  - Metamodel Description ... 14
  - Tcl API ... 14
  - Java API ... 15
  - .NET API ... 15

Add-Ins ... 15
Automatic Command Sequences ... 15
Integration of the Editions ... 16
Functionalities of the Innovator eXcellence Editions ... 17
Project-Specific Model Configuration ... 17
Configuration Editor ... 17
Profiles ... 17
Stereotypes ... 17
Evaluations and Engineering Actions ... 18
Verification Routines ... 18
Menus ... 18
Representation ... 19
Creation of Elements ... 19
Search for Profile Elements ... 19
Browser Configuration ... 19
Documentating Profiles ... 19
Editing Model Elements ... 20
Non-Modal Dialog ... 20
Traceability Using the Traceability Wizard ... 20
Data and Version Management ... 21
Powerful Online Repository ... 21
Open Metamodel ... 21
Hardware Independence ... 21
Repository Directory ... 21
Administration ... 21
Data Security ... 21
Team Support ... 22
Consolidation ... 22
References ... 22
Multi-User Mode and User Concept ... 22
Multi-User Capability ... 22
User Administration ... 22
Model Administrator ... 23
Password Protection ... 23
Access Rights ... 23
Privileges ... 23
Re-Use ... 23
Integrated Message System ... 24
Version Management ... 24
Version Management Repository ... 24
Version Management Browser ... 24
Displaying Version Differences ... 25
Version Files for any External Version Systems ... 25
Migration ... 26
Generation of Documentation ... 27
Customizable Templates ... 27
Structure Definition ... 27
Title Pages ... 27
Headers and Footers ... 27
User Chapters ... 27
Table of Contents and Index ... 27
Templates ... 28
Formatting ... 28
Output Formats ... 28
Preview Window ... 28
Microsoft Word ... 28
XML ... 28
Postscript ... 28
ASCII ... 29
Export of Graphics ... 29

Generation ... 29
Single Click Generation ... 29
Quick Reports ... 29
Printing ... 29
Direct Printing/Print File ... 29
V-Modell® XT Support ... 30
Innovator Object eXcellence ... 31
Features ... 31
Object-Oriented Modeling with UML 2.1 ... 33
UML as Standard ... 33
Modeling ... 33
Packages and Package Diagrams ... 33
Class Diagrams ... 33
Object Diagrams ... 33
Use Case Diagrams ... 33
Component Diagrams ... 34
Deployment Diagrams ... 34
Activity Diagrams ... 34
State Machines ... 34
Sequence Diagrams ... 34
Composite Structure Diagram ... 34
Artifacts ... 34
Languages ... 35
Code Generation ... 35
BPEL Export ... 35
UML Profiles ... 35
Integration and MDA Transformations ... 36
XMI Interface ... 36
Innovator Object classiX ... 37
Features ... 37
Object-Oriented Modeling with UML 1.4 ... 39
UML as Standard ... 39
Modeling ... 39
Packages and Package Diagrams ... 39
Class Diagrams ... 39
State Diagrams ... 39
Use Case Diagrams ... 39
Sequence Diagrams ... 40
Collaboration Diagrams ... 40
Activity Diagrams ... 40
Object Diagrams ... 40
Component Diagrams ... 40
Target Languages ... 40
Languages ... 40
Round-Trip Engineering ... 40
Development Environments ... 41
UML Profiles ... 41
Integration and MDA Transformations ... 41
XMI Interface ... 41
Innovator Business classiX ... 42
Features ... 42
Business Process Modeling with UML 1.4 ... 44
UML as Standard ... 44
Configuration Model ... 44
Class Model ... 44
Privileges ... 44
Customizability ... 45
UML 1.4 Extensions ... 45

Activity Types ... 45
Activity Definition ... 45
Conditions ... 46
Analysis ... 46
Workflow ... 46
Process Models ... 46
Modeling and Metamodeling ... 47
Model Management ... 47
Configuration Model ... 47
Business Process Model ... 47
Analysis ... 48
Workflow Support ... 48
Process Model Support ... 48
Special Profiles ... 48
Integration and MDA Transformations ... 48
XMI/XML Interfaces ... 49
XMI Export ... 49
XML Export for Microsoft Project ... 49
Innovator Data eXcellence ... 50
Features ... 50
Data Modeling with ERM and SERM ... 52
Modeling ... 52
Conceptual Schema ... 52
Database Independency ... 53
Database Schema ... 53
Modeling in the Database Schema ... 53
Generation and Reverse Engineering ... 54
Supported Databases ... 54
Target Language Types ... 54
Integration and MDA Transformations ... 55
Innovator Data classiX ... 56
Features ... 56
Data Modeling with ERM and SERM ... 58
Modeling ... 58
Conceptual Schema ... 58
Database Independency ... 59
Database Schema ... 59
External Schema ... 60
Generation and Reverse Engineering ... 60
Supported Databases ... 61
Target Language Types ... 61
Integration and MDA Transformations ... 61
XMI Interfaces ... 61
Innovator Function classiX ... 62
Features ... 62
Function-Oriented Modeling with SA/RT/SD ... 64
Structured Analysis ... 64
Real-Time Extension ... 64
Structured Design ... 64
Modular Design ... 64
From Analysis to Design ... 65
From Design to Implementation ... 65
Reverse Engineering ... 65
Modeling ... 65
Model Management ... 65
Structured Analysis ... 65
Real-Time Extension ... 65
Structured Design ... 66
Implementation ... 66

Target Languages and Interfaces ... 66
Implementation Languages ... 66
Generation ... 66
Reverse Engineering Structured Design for C ... 66
MATLAB and Simulink ... 66
Integration and MDA Transformations ... 67
Innovator Programming classiX ... 68
Methods ... 68
Nassi-Shneiderman ... 68
Generation of Source Code ... 68
Reverse Engineering ... 68
Procedure ... 68
Integration ... 68
Compilers ... 68
Search/Replace ... 68
Undo ... 69
Macros ... 69
Structuring ... 69
Correct Program Structure ... 69
Usability ... 69
Navigation ... 69
Folding Environments ... 69
Usability ... 69
Syntax Highlighting ... 70
Source Code Generation ... 70
Customizability ... 70
Comments ... 70
Scan Markers ... 70
Target Languages ... 70
Syntax Checking ... 70
Innovator Report classiX ... 71
Features ... 71
V-Modell® XT Support ... 72
Company-Specific Process Models ... 72
Conformity in the Documentation Process ... 72
Functionality ... 72
Integration and MDA Transformations ... 74
Vertical Integration for Innovator classiX and eXcellence ... 74
Business-CASE Transformation ... 74
Horizontal Integration for Innovator classiX ... 74
Object-Object Transformation ... 74
Object-Data Transformation ... 75
Data-Function Transformation ... 75
User-Defined Transformations ... 75
Analysis-Design Transitions for Innovator classiX ... 76
Object ... 76
Function ... 76
Innovator and Integrated Development Environments ... 77
Model Integration ... 77
EJB Applications for Innovator classiX ... 77
Data Modeling ... 77
GUI Integration ... 77
Innovator View ... 77
Eclipse Plug-In ... 78

.NET Support ... 78
Code Generation for C# ... 78
.NET Programming Interface ... 78
Integration in Microsoft Visual Studio ... 78
Other Interfaces ... 79
SCC Interface ... 79
Integration with of Innovator classiX with PVCS-VM and PVCS-Dimensions ... 79
XMI Interface ... 79
XML Export for Microsoft Project from Innovator Business ... 80
BPEL Export from Innovator Object eXcellence ... 80
Interface to ARIS Design Platform ... 80
Innovator Web ... 81
Web Access to Innovator Models ... 81
Navigation ... 81
User Interface ... 81
Product Documentation ... 82
Online Help ... 82
Tutorial ... 82
User Manual ... 82
Administrator Manual ... 83
MID Modeling Methodology M³ ... 83
Configuration Manual for Innovator eXcellence ... 83
Migration Manual ... 83
API Help ... 83

Overview Innovator Editions

Features	Innovator					
	eXcellence		classiX			
	Object	Data	Object	Business	Function	Data
Methodology						
Unified Modeling Language (UML 1.4)			✓	✓		
Unified Modeling Language (UML 2.1)	✓					
Structured Analysis and Structured Design (SA/SD) with real-time extension (RT)					✓	
Entity Relationship and Structured Entity Relationship (ER/SER)		✓				✓
Editors and Representation Types						
Administration program for licenses and repositories	✓	✓	✓	✓	✓	✓
Model browser with model tree view, list of model contents and search engine	✓	✓	✓	✓	✓	✓
Configuration editor	✓	✓				
Diagram editors	✓	✓	✓	✓	✓	✓
Table editors		✓		✓	✓	✓
Nassi-Shneiderman diagram editor					✓	
Specification editor	✓	✓	✓	✓	✓	✓
Data Management, User Administration and Licensing						
Multi-user online repository	✓	✓	✓	✓	✓	✓
Semantic and syntactic quality assurance	✓	✓	✓	✓	✓	✓
Integrated version management	✓	✓	✓	✓	✓	✓
Comparison of model data	✓	✓	✓	✓	✓	✓
Model-wide user and user group administration	✓	✓	✓	✓	✓	✓
Repository transformer for migration of model data	✓	✓	✓	✓	✓	✓
Floating licensing	✓	✓	✓	✓	✓	✓
Web access (read)	✓	✓	✓	✓	✓	✓
Import/Export Interfaces						
Integration of external objects			✓	✓	✓	✓
Artifacts (model, script, code, data bank etc.)	✓	✓				
Import of files DDL, XMI		SQL/DDL				SQL/DDL
XMI export based on	UML 2.1		UML 1.4	UML 1.4		CWM
XML export				Microsoft-Project		
Integration of implementation tools	✓		✓		✓	
Integration of ARIS Design Platform	✓					
Integration of MATLAB and Simulink					✓	

Features	Innovator					
	eXcellence		classiX			
	Object	Data	Object	Business	Function	Data
Tcl API reading and modifying			✓	✓	✓	✓
Java API reading and modifying	✓	✓	✓[1]	✓[1]	✓[1]	✓[1]
.NET API reading and modifying	✓	✓	✓[1]	✓[1]	✓[1]	✓[1]
Documentation						
Integrated generation of documentation	✓	✓	✓	✓	✓	✓
Documentation with Innovator Report (V-Modell®XT-compliant or company-specific process-compliant)	✓	✓	✓	✓	✓	✓
Customizable templates tailored to your corporate identity	✓	✓	✓	✓	✓	✓
Export of graphics (EMF, EPS, PNG, SVG)	✓	✓	✓	✓	✓	✓
Configuration, Model Management and Metamodeling						
Hierarchical package structures based on profiles and model templates	✓	✓	✓	✓	✓	✓
Configuration model based on a UML 1.4 class model				✓		
Programming Languages and Target Systems						
Java, C++, CORBA IDL, Visual Basic			✓			
Java, C++, C#	✓					
C, COBOL					✓	✓
DB2 (all platforms), ORACLE, Informix, MS SQL-Server, other databases possible		✓				✓
BPEL	✓					
Engineering Techniques (Extensible)						
Architecture/model-driven, incremental forward engineering	✓	✓	✓	✓	✓	✓
Round-trip engineering			✓			
Reverse engineering		✓[2]	✓		✓	✓[2]
Transformations and Implementation Support						
Generative and associative transformations	✓	✓	✓	✓	✓	✓
Workflow support				✓		
System Platforms for Client and Server						
Windows (2000, Server 2003, XP, Vista)	✓	✓	✓	✓	✓	✓
Linux	✓	✓	✓	✓	✓	✓
Solaris	✓	✓	✓	✓	✓	✓

[1] Read only

[2] Not COBOL

Innovator – Integrated Modeling Platform for Future-Proof Software

Editions

The Innovator editions are parts of an integrated solution that covers the area of business process modeling and all important modern software engineering methods. The following editions are available:

Innovator eXcellence

- Innovator Object eXcellence — Object-oriented modeling UML 2.1
- Innovator Data eXcellence — Data modeling

Innovator classiX

- Innovator Object classiX — Object-oriented modeling UML 1.4
- Innovator Business classiX — Business process modeling
- Innovator Data classiX — Data modeling
- Innovator Function classiX — Function-oriented modeling
- Innovator Report — Project documentation compliant with V-Modell®XT

Integration

The individual editions are integrated by means of methodical MDA transformations (model-to-model and/or model-to-code) or import/export interfaces. This way, modeling results can be reused and traced in other models (e.g. from business process analysis to an implemented IT system that supports the process in the appropriate software engineering models).

General Functionalities

The individual editions are complemented by common, general functionalities such as the generation of model documentation or the management of current and historic modeling results as well as the integration of the results of a great variety of external tools.

Standards

In modeling, the individual Innovator editions support accepted standards such as the Unified Modeling Language (UML) of the Object Management Group (OMG) for business process and object modeling, Entity Relationship modeling according to Chen/Sinz, Structured Analysis (SA) according to DeMarco with real-time extension (RT) according to Hatley/Pirbhai and Structured Design (SD) according to Yourdon.

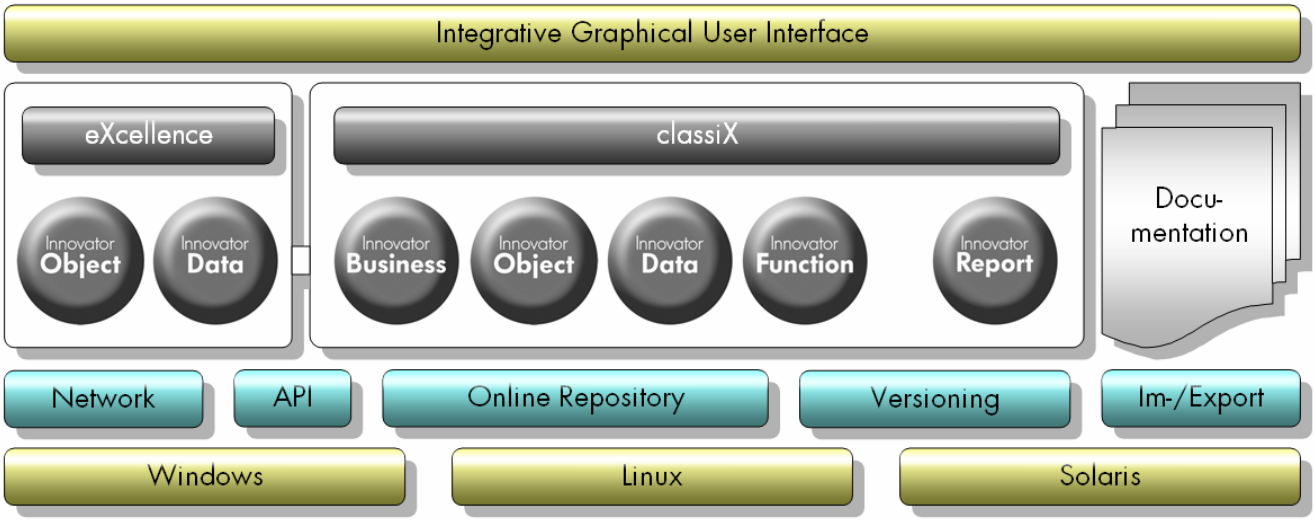
Customized Software Production Environments

The various editions which, as a full ensemble, cover the entire software lifecycle can be combined to form a customized software production environment.

Architecture

Modular and Open

Innovator is based on a four-layer model:



The supported operating systems form the basis of the four-layer model. The online and version repositories as well as the network access constitute the second layer. Various editions with their methods such as business process modeling or object-oriented modeling represent the third layer. The top layer consists of the standard user interfaces for the various operating systems.

Client-Server Architecture

The client-server architecture of Innovator enables multi-user operation both in heterogeneous as well as in homogeneous networks. All licenses required to work with Innovator are dynamically managed by a License Server program. Innovator comes with Installation and Administration programs.

The repository data is managed by a Repository Server program. This program manages the access rights and coordinates user access to the model information. Client programs are used to create, modify and delete model information. These programs display the model information in a browser or, depending on the type of information, in the form of diagrams, tables or text. All programs can run anywhere in the network.

Administration

An Administration Program is provided for all administrative activities concerning licenses and repositories (it replaces the License Browser and Repository Browser of previous INNOVATOR versions)

Such administrative activities include activation and splitting of licenses, administration of repositories (e.g. saving, quitting, definition of backup times for repositories), creation and administration of models as well as user management.

The administration user interface consists of a window with three areas. The License Server Area (left) displays all configured license servers, the Repository Area shows the repositories and models of the license server selected in the left area. Accessed information (including information about environment variables, licenses, repositories) is displayed in tabs in the bottom area of the window.

Licensing

Innovator supports the Floating License concept. Licensing depends exclusively on the number of users who work simultaneously with a given Innovator edition. The number is independent of the operating system used.

Platform licenses are available for the various operating systems supported. These licenses allow Innovator to be used on the corresponding operating system platform. If you change to a different platform, it is sufficient to re-acquire this license.

Various domain-specific profiles which support industry-specific reference models or implementation environments are licensed via profile categories. The appropriate profile category must be licensed for each user working with models based on such profiles.

If you want to view Innovator models using standard Web browsers, the Web platform and the required number of Web clients must be licensed.

License Distribution

Innovator permits a differentiation between a main license server and any number of project license servers. Repositories of the main license server are available on a cross-corporate basis, while repositories of project license servers are only available to the corresponding projects. The licenses from the license server of an organization can be assigned from the main license server to the various projects.

In order to enable simultaneous work with earlier INNOVATOR versions, it is possible to assign licenses from the main license server to a project license server for previous releases.

Single seat licenses (so-called Laptop Licenses) may be checked out from the main license server for users working without license server connection. These licenses are generated for one specific machine and are available on that machine by transfer of the license repository. These licenses can be scheduled for periods precise to the day. After expiry of the period, they are again available to the main license server.

Hardware and Software Basis

Supported Platforms

Innovator 2008 (Version 9) runs with the following hardware/operating system configurations:

- PC min. 1 GHz with Microsoft Windows 2000/Server 2003/XP
- PC min.1 GHz with SuSE Linux 10.3(or higher) or Red Hat Linux 3.4 or higher
- SUN SPARC Station with Solaris 8 or (higher)
- SUN x86 with Solaris 8 (or higher)

Please inquire for the availability on other systems.

The data can be used under all operating systems.

Memory Requirements

The recommended available memory for Innovator is at least 512–1024 MB on a server and 256 MB on a client.

Hard Disk Space Requirements

Innovator requires a hard disk space of 1 GB.

Network Operation

The following network software and protocols are required to run the Innovator editions in a network:

- Microsoft Windows 2000/Server 2003/XP: TCP/IP
- Solaris/Linux: All network operating systems with the appropriate TCP/IP implementation of the hardware manufacturer

Graphical User Interfaces

Microsoft Windows user interfaces are supported in the versions Windows 2000, Windows Server 2003 and Windows XP.

Motif 2, version 2 or higher, is used as the basis for the graphical user interfaces under the operating systems UNIX and Linux.

A resolution of at least 1280 x 1024 pixels is recommended for ease of use.

User Interface

Operation and Menus

Look & Feel

The operation, look & feel and functionality of the Innovator editions are identical for the supported operating systems.

Windows Standard

The user interfaces follow the Microsoft Windows standard. The menus comply with the Windows standard. User-specific interface preferences can be configured.

Innovator provides configurable toolbars and popup menus for frequently used commands. Intelligent Defaults Functions support the user. The menu commands are context-sensitive, i.e. they depend on the current situation. Comprehensive, context-sensitive online help systems are available for all graphical Innovator user interfaces.

The user can work on several models and in several windows at the same time to display different diagrams and views so that the required information is always easily accessible even in large systems.

Multi-Language Capabilities

The user can set the Innovator user interface language as required. Innovator supports German (de_de) and English (en_us) for all visible user interface elements such as menu items, dialog boxes, messages, etc. The online help is also available in German and English.

User Interfaces

Log-In

A log-in dialog is provided for access to the models. The following types of log-in are available:

Administrator	An administrator has all rights in the model.
Guest	A Guest has read-only rights.
User	Log-in for normal users with the access rights assigned to this role (user group).

Users must additionally log-in using a role specification in Innovator eXcellence. This prevents the undesired propagation of access rights when new model elements for users with several roles are created.

Role specification for log-in is not provided in Innovator classiX.

Model Browser

Innovator provides a Model Browser that displays all the model information. The Model Browser allows you to create and edit model elements. In order to provide concise information even in the case of comprehensive models, the Model Browser displays information on the model structure, the model's contents and detailed information on individual model elements in various areas. The Results Area allows you to display a user-specific collection of model elements (e.g. search results).

You can display the diagram and table editors as well as the specifications editor for the respective model elements from the Model Browser.

Graphical Editors

Innovator provides graphical editors for all model elements which allow you to work on diagrams or tables. In addition to a WYSIWYG representation, the diagram editors offer Zoom functions which allow users to keep track of even complex models. Fonts and page layout are easy to configure. The Undo function undoes any number of steps. The status line displays important information on the selected element or on the entire diagram or table.
A special command allows the user to display comprehensive, structured information on model elements.

Different colors and fonts may be defined to obtain a clearly structured representation of the diagrams and the properties of the elements they contain.

Modeling information and notes may either be placed anywhere in the diagram or associated with a diagram element.

The editors provide alignment commands and various standard layout functions to facilitate the arrangement of the diagram elements.

A navigation frame can be shown/hidden in the left area of the window to facilitate navigation in the diagram. The navigation frame displays the diagram elements in a hierarchical structure.

Model elements can be selected both in the navigation frame and in the diagram itself. The selected model element is then displayed in the opposite screen area.

The navigation frame can also display model elements that are not shown in the diagram (such as hidden attributes or methods of classes).

In the Innovator eXcellence editions, it is also possible to rename elements in the navigation frame.

Text Editor

An integrated text editor allows you to enter specifications. In addition to import/export interfaces, the text editor features a link concept to access existing descriptions in ASCII format or MS Office documents or external graphics. It is possible to verify the specifications according to the graphical definitions. It is also possible to integrate an external editor.

Tabs in the text editor workspace enable quick access to the specifications created in the editor.

Instructions and Labels Area

In addition to the actual specifications, the Instructions Area displays history information as well as methodical contents of the model which can be used to create the text. A further area shows the labels configured for the model element and the current values.

Templates

Guidelines for descriptions and text defaults can be defined as templates and automatically included in the specifications.

Several named specifications may be configured for the model elements. This way, it is easy, for example, to include internal suggestions in the model.

Configurable Help Menu

You can configure entries for all interface help menus using special configuration files. You can use common URLs for internet addresses and documents in Microsoft Windows. In Unix, the jump target can be directly passed to the configurable browser i.e. use of internet addresses is limited.

Settings

All settings made in the Innovator user interfaces are automatically saved when you exit the application. Therefore, explicit saving of settings is not required.

General Functionalities of Innovator

Functionality

Profiles

A profile allows you to define defaults for modeling results for special application areas or special process models. By defining such a model template, you specify the appropriate adaptations and extensions.

For example, profiles may comprise:

- General basic model defaults
- Configured property values (such as stereotypes or tagged values)
- Create and display templates for elements
- Packages and package structures
- Create defaults for new elements
- Possible assignments of model elements to packages
- Labels and specifications
- External objects
- Search and verification routines
- Complete configuration models models
- Any model element

By using a profile when creating a new model, the user can initialize the model with the corresponding properties. Profiles can also be reloaded in existing models as add-ons. The model administrator can create such profiles from a model and save them for re-use. This enables the propagation of company-specific modeling methodologies and styles across a complete organization.

Model Management

All Innovator editions use the concept of packages to organize and structure models. According to the UML specification, a package is a mechanism for arranging elements in groups and hierarchies. The package hierarchy is displayed in the left tree view of the Model Browser and in a number of dialogs.

Model elements (including packages) can be assigned to other packages. There are two different assignment types:

- Parent package (owning package)
  This assignment specifies the package that owns the model element. By deleting a parent package, you delete all the elements it contains. There can be exactly one assignment of this type per element. The assignment to a parent package is also supported for packages.

- Reference package (classiX editions)
  This assignment specifies a package in which the model element is referenced. There can be any number of assignments of this type per model element. Reference package assignments are not supported for packages.

The status column of the list of model elements in the Model Browser indicates the type of assignment. An 'R' and a special color designate a reference assignment of an element to a package.

Package diagrams are used to represent dependencies between packages. They may also be used to specify create defaults (parent package for new element).

The package hierarchy and these assignments structure the model. The type of assignment as well as the permitted element types are specified by the model administrator and monitored by the system.

The permitted model elements (as specified in the model configuration) can be chosen from configured menus in the Model Browser and the editors. When model elements are created, this is monitored according to the configuration.

Connections can be created by means of drag&drop from the source to the target element (e.g. relationships or data flows).

Namespaces

The UML-based Innovator editions Business, Object and Data eXcellence support namespaces. The names of model elements do not have to be unique in the complete model, but only in the assigned parent package. In order to uniquely identify an element, the name and the complete path of all parent packages is
required.

In order to support re-use and prevent unwanted duplicate names, the system can check for duplicate element names in the model when the name is entered.

Labels

The Innovator repository allows the user to define and store labels (additional information) for an element. This way, the modelers can include their own information on model elements in Innovator. Searches for labels and their values enable, for example, the rapid creation of project progress reports.

The number and naming of the labels may differ from model element type to model element type. It is possible to define general labels that are valid for all model elements. The defined labels can be displayed in the various views such as diagram, table or tree. Colors can be assigned to the label values which may be used in the editors to represent the model element names in colors.

Innovator provides convenient search criteria for model elements with special labels or label values. The hits can be easily selected in the Results Area of the Model Browser and the appropriate representations displayed.

The right to modify defined label values in Innovator multi-user mode can be easily and quickly assigned to specific users.

You can specify for each individual label whether it is to be displayed on screen and in the generated documentation.

The valid value range for a label can be specified in terms of a fixed set of default values, numerical ranges or character strings. In addition, you can specify a default value for the label value.

Specifications

Any number of text descriptions (specifications) can be defined for a model element and stored in the Innovator repository. This way, users can include their own text types in Innovator. The number and naming of the text types may differ from model element type to model element type. It is possible to define general text types that are valid for all model elements.

You can specify for each individual text type whether its contents are to be included in the generated documentation.

The model administrator may configure an external text editor instead of the internal editor for working with the specifications. However, in such a case, the maintenance functionality of the referenced model contents is not available.

Generations

Any number of versions (generations) of a model element can be stored in the Innovator repository for online access. The system allows you to create, delete and check in these generations as well as track the differences between them.

When creating generations, you may specify an optional short description. The Model Browser displays the number of generations that exist of the model element in question. In order to freeze different generations of different elements, these can be subjected to version management.

External Objects

The artifacts of external tools such as MS Office applications as well as visualization or process control data can be stored in the models in the Innovator classiX editions. Innovator provides the following functionality for such external objects:

- Any type of external result can be configured.
- A tool for processing the external result can be configured for each external result independent of the operating system. OLE, DDE or a command interface can be used to integrate the tool.
- External results can be renamed and deleted. Innovator supports the multi-user concept for external results (access rights, locking, unlocking).
- External results can be exported; external results created outside of Innovator can be imported.
- Graphics files created outside of Innovator can be imported as print results and assigned to external results. These print results are used in the documentation instead of the external result.
- The general search mechanisms in the Model Browser can also be applied to external results.
- In the specifications of model elements, external results can be referenced and included.
- Templates can be configured for MS Word and MS Excel which are used when new objects are created.

Search

All Innovator editions provide a powerful search engine.

You can search:

- For names (with matchcode search)
- For label values
- For element properties
- For element types (e.g. diagram types or table types)
- For elements which violate a verification routine
- By history (creation date, modification date, user)
- For user-definable properties by means of Tcl scripts
- In specifications, diagram notes and implementations
- Several searches can be combined into a single search by means of "AND" and "OR". Searches can be saved as separate commands to be used just like all other menu commands (optionally with an icon on one of the toolbars).

Consistency and Quality Assurance

Data Consistency

Innovator circumvents the consistency problem in multi-user mode by means of a data management concept with locking mechanisms in the online repository that prevents redundancies. This automatically ensures the consistency of the model data at any time. It is not necessary to consolidate the work results of different project members.

In addition, Innovator provides numerous verification routines to ensure the methodological correctness of the model.

Context Sensitivity

Context sensitivity, supported by Intelligent Defaults functions, prevents major inconsistencies when the elements are created.

Methodological Verification

In addition to entry verification, Innovator provides numerous methodological verifications the user can trigger within the context of the current work situation.

These verifications can be applied at diagram level or to submodels or, in the Model Browser, to the entire model.

Configuration

The rules applied in a verification run can be selected from a given set of verification options and configured. The Innovator classiX editions allow users to create and apply their own verification rules via integrable Tcl scripts.

Verification routines can be created for any aspect of model verification. Such verification routines are a collection of specific, available verification options for the model elements.

The verification routines are saved under a name and can be used like a menu command (optionally with an icon on the left toolbar).

A separate privilege is provided for creating, modifying and deleting verification routines.

The verification routines ensure that all project members apply the same verification mechanisms. The verification routines are available in the Model Browser as well as in the diagrams and tables in the same way and yield identical results.

Documentation

The verification results as well as the configuration can be included in the generated documentation. Besides you can work on the verification messages in the text editor.

API

Metamodel Description

The metamodel of the Innovator repository is documented in a separate API online help. Read and write access to all data is possible by means of programming languages, Tcl (for Innovator classiX) and Java or C# (for Innovator eXcellence) via the API. This enables the implementation of Innovator extensions or interfaces to other tools via the API.

Tcl API

The Public Domain Language Tool Command Language (Tcl) by John K. Ousterhout allows you to access all information in the repository and process the data as required in the work environment in Innovator classiX.

Innovator classiX enables any type of repository data report by means of Tcl. One of the many examples of the Tcl interface shipped with Innovator Data classiX shows the generation of SQL/DDL scripts from the database schema of Innovator Data classiX.

Tcl is an easy-to-learn interpreted script language that allows the user to get information on the current contents of the repository or to execute user-defined transformations. Just like all other Innovator editions, the Tcl commands access the specified repository in online mode.

The created Tcl scripts are almost fully portable between the operating systems supported by Innovator.

Java API

The object-oriented, platform-independent language Java (Sun Microsystems) allows you to access all information in the repository and process the data as required in the work environment in Innovator eXcellence. The Java API requires Java 5.

Innovator eXcellence enables any type of repository data report by means of Java.

Java is a widely used programming language that allows the user to get information on the current contents of the repository or to execute user-defined transformations. Just like all other Innovator editions, the Java commands access the specified repository in online mode.

The created Java scripts are almost fully portable between the operating systems supported by Innovator.

Reading access is also possible for Innovator classiX models.

.NET API

As with Java API and virtually all identical functions, Innovator eXcellence enables access to all information from the repository using a .NET language (e.g. C#, C++, VB.NET). The .NET API only requires the .NET framework from version 2.0 for this.

This means that any number of evaluations can be created for Innovator eXcellence in a .NET language of your choice. The .NET API acts as an independent client, which can be executed completely independently from the installation of an Innovator.

Reading access is also possible for Innovator classiX models.

Add-Ins

The user-defined scripts can be integrated as Add-Ins via the menu and used just like all other menu commands (optionally with an icon on the toolbar).

Automatic Command Sequences

With InoAutoCommand.jar a sequence of menu commands can be implemented automatically in model trees of any Innovator model by a configurable control file. InoAutoCommand can be started directly by a JavaVM. Therefore the obligatory Innovator paths and libraries (inojapi.jar etc.) must be defined. The control file contains in sections the description of model and login data, the definition of environment variables and the actual command sequences.

Even though InoAutoCommand.jar is a Java application, it can be used with eXcellence and classiX models.

For the fully automatic execution the used menu commands must not open dialog boxes, thus require no interactive operation by the user.

Using time-steered mechanisms of the operating system (Unix: 'cron jobs', Windows: 'at' or 'schtask.exe'commands) with InoAutoCommand, you can start time-consuming actions, e.g. the generation of documentations or the execution of generators, to a fixed time.

Integration of the Editions

The diverse modeling methods are integrated by means of associative or generative transformations (mappings) between models.

Innovator classiX supports the following types of transformations:

- Business → CASE (Object, Function, Data)
- Object ↔ Object
- Object ↔ Data
- Data → Function
- Data ↔ Object eXcellence

Innovator eXcellence supports the following types of transformations:

- Object ← Business classiX
- Object ↔ Data

Functionalities of the Innovator eXcellence Editions

Project-Specific Model Configuration

Configuration Editor

Innovator eXcellence provides a Configuration Editor for all information on the configuration of a model and its profiles. In addition to the mere UML 2.1 profile definition, this includes, for example, properties such as colors and fonts, menus with Create Templates, Engineering Actions and the configuration of verification routines.

The Configuration Editor allows you to create and edit profiles. Elements can be conveniently assigned to the configuration structure of a profile by means of drag&drop.

Design and handling of the Configuration Editor ensure the transparent and efficient definition of model templates and add-ons.

Profiles

Innovator distinguishes between the definition of a software development process and the application of such a process in an actual software development project.

In the process definition, you create UML profiles. A UML profile groups UML extensions that are logically related. Profiles can build on each other, i.e. one profile imports another profile. Such a profile not only defines stereotypes and stereotype properties, it is also used for extensive tool configurations to ensure optimum user support and governance in a given project. For example, you can specify that a model with the stereotype «analysisModel» may only contain packages with the stereotype «analysisPackage». A package with the stereotype «analysisPackage» may only contain class diagrams with the stereotype «analysisClassDiagram» or classes with the stereotype «entity», «control» or «boundary». Language-specific design profiles (e.g. Java) and technology-specific profiles (such as J2EE, Web Services, ...) can be defined in the same way as the analysis profile mentioned above.

In a concrete project, one or several of these profiles are imported in Innovator eXcellence. The cross-corporate application of such profiles promotes a uniform modeling style. For example, a profile ensures that all J2EE projects deliver the same modeling results. Among other things, this is a prerequisite for the reuse of generators in multiple projects and facilitates the integration of new team members in a project.

A profile can contain one model language. All stereotypes created in the profile automatically relate to this language. When an element is created, it is always assigned to a profile.

Profiles can import other profiles to include the information of the imported profiles. These profiles, in turn, can be imported in the actual models.

Profiles are subject to the user concept, i.e. access rights and locking.

Stereotypes

In Innovator, each element that can be a stereotype has a predefined original stereotype. In a profile, such stereotypes can be further specialized. A stereotype can inherit from several other stereotypes. All new stereotypes have at least one generalization.

Stereotype Properties

Any type of properties can be created for stereotypes. Specialized stereotypes inherit the properties of their parents. As opposed to previous INNOVATOR versions, it is now possible to add special properties to a stereotype.

Labels and Specification Texts

Configuration and inheritance of labels and specification texts are identical and are handled in the same way as stereotype properties. In addition, there is a general element type that can contain labels and configurations for specification texts for all elements.

Create Templates

Create Templates can be defined for stereotypes. In the Configuration Editor, these templates can be changed via the same dialogs as in the Model Browser.

Permitted Relationships

A stereotype's relationships (e.g. owning relationships (contents)) that are permitted in modeling can be defined.

Naming Conventions

The validity range and the naming conventions can be specified for any nameable element per stereotype. These presets are verified during modeling.

Evaluations and Engineering Actions

User-defined evaluations and Engineering Actions which are implemented via the Java API can be configured as actions in the menu. It is possible to assign privileges for such actions to user groups (roles). In such a case, the configured actions are only available to the appropriate user groups (roles).

Verification Routines

Predefined methodological verification routines can be used in any profile. The assigned element types and stereotypes define the possible contents of such a verification routine.

Menus

Menus are created on the basis of the configured contents. The menus for the Model Browser are defined below the model stereotype. In the case of diagrams, the permitted contents for the respective stereotypes is evaluated and provided as a menu. The defined Create Templates are available as menu items.

In addition, it is possible to create menus for Engineering Actions and verification routines. All menu items can be configured in various languages.

Representation

The display settings for displaying elements in diagrams are centrally set in the configuration. You can locally overwrite the display options in a model if this is permissible by the configuration.

An icon and the color as well as the font and font color can be defined for each stereotype of a model element.

It is possible to define a property for each stereotype which displays a bitmap preceding the name of the corresponding element.

The settings for the representation of stereotypes, packages, symbols and other properties are made in the model for a diagram for all elements of a type.

A logical element is displayed in various ways in the diagram using various existences. Each of these possible displays can be created separately.

As an additional display option, stereotype properties can be displayed in compartments of classifiers in diagrams.

Behavior call actions, interaction references or sub state machines can be opened and the referenced contents graphically displayed.

Creation of Elements

The submenu items and Create Templates grouped in the configuration are shown as selection buttons at the left side of the window.

An element whose size is important when it is created (e.g. representation of continuations or combined fragments in sequence diagrams) can be created with the required size by means of drawing a frame that defines the size of the element to be created.

Search for Profile Elements

The configuration editor has a search machine. The search results can be collected in a list and be jumped to from there in the tree. The search result can be filtered after types.

Browser Configuration

How the contents is displayed in the model browser’s detail window can be set by the user using special templates.

Documentating Profiles

Documentation generation can also be applied to profiles so that a document is automatically created with configuration specification.

Editing Model Elements

Non-Modal Dialog

The Innovator eXcellence functionality for editing model elements has been completely re-engineered.

All changes to the properties of a model element (such as renaming, setting labels and property values or editing specifications) are made in the Edit/Specifications dialog which can be displayed by means of a double-click on the model element. Property pages replace the tabs that are used in the Innovator classiX editions.

In the property pages, the structure follows the model contents that can be reached from the selected model element in a hierarchical way. This enables you to modify several dependent model elements in the same dialog.

The property pages in the non-modal dialog are refreshed along with the selection of model elements in the diagram.

The most frequently used dialog boxes of the eXcellence edition (e.g. Edit/Properties) can be changed in their size. Dialog contents are adapted accordingly.

Traceability Using the Traceability Wizard

Dependency relationships are used with automatic model transitions or with linking requirements and the model elements they realize. A dependency editor enables this dependency relationship to be visulaized and maintained between model elements. Traceability of model transitions and model elements to the requirements is transparent.

Data and Version Management

Powerful Online Repository

Open Metamodel

Innovator uses a uniform, binary-compatible online repository for all supported operating systems and computer architectures. The repository is based on an open metamodel. The metamodel is described along with the API online documentation in the form of a Structured Entity-Relationship Model (SERM).

Hardware Independence

The consistent online repository is a prerequisite for a network and computer-independent modeling and development environment. The model information can be used under all operating systems supported by Innovator; each team member in the network can directly access this data. Transformations of the data are not required. Innovator repositories may contain models of all editions.

Repository Directory

A directory with the name of the repository contains all files and directory that belong to an Innovator repository.

This structure simplifies administrative tasks such as creating backups or duplicates.

Administration

An Innovator repository is managed by a repository administrator. This role comprises the following responsibilities: creation, renaming, deletion of models, backup of repositories, starting and shutting down of repository servers and specification of times for automatic backups. It is possible to prevent users from logging on to the models contained in a repository and to give the appropriate reason, e.g. repository shutdown for backup purposes. The role of the repository administrator is password-protected.

Data Security

In the case of a repository failure (computer failure, unexpected program termination, power failure, etc.), a recovery procedure can recover the most recent status of the repository and all the model information it contains except for the last modification.

This way, all model information is re-available in the same state as shortly before the interruption. Work can immediately be resumed on the basis of this state.

Team Support

The central data management in the Innovator online repository makes all modeling results immediately available to all project members. This ensures that all information is up to date at any time and can be reused. Via the integrated client/server concept, Innovator checks the rights of the users to access model elements or procedures.

Consolidation

No consolidation of the results of the different team members is required since all developers work on the basis of the same data. Modifications made by one project member become immediately visible to the rest of the team. This means that no time, effort or money has to be invested into merging the results produced by the various project members.

A consolidation is only necessary if projects are split into partial projects and partial projects merged into a single project. However, this is usually only required if there is no direct connection between the team members via a Local Area Network (LAN) or a Wide Area Network (WAN). If a project is split into partial projects, such sub-projects are consistent in themselves due to the online repository technology.

References

Each Innovator edition provides a Jump menu command. This command shows all referenced elements in this and other models and allows you to quickly jump to them.

For maximum working speed, a default reference is defined for almost any element that can be activated by means of a double-click with the Shift key pressed. If the jump target is not unique, Innovator displays a list allowing you to select the desired target.

Multi-User Mode and User Concept

Multi-User Capability

All Innovator editions are multi-user-capable in the network. A user concept allows for the creation and deletion of users and user groups as well as the assignment of access rights to individual elements or privileges to procedures. The locking mechanisms and access rights are automatically controlled by the system. This ensures that users cannot inadvertently overwrite the work of other team members.

User Administration

Users are defined by means of names which are also used as login names. Users are grouped in user groups which are themselves defined by means of names. A given user can belong to any number of user groups.

Users and user groups are specified by the model administrator.

The model administrator can log off individual users from a model.

Special groups with predefined passwords may be defined for searching model elements via the Web access. The model administrator may explicitly set or revoke the permission to log users or user groups into a model.

Users can be imported, selected from listing services using Lightweight Directory Access Protocol (LDAP), and be put on automatically as Innovator users in the model. This functionality is available for the operating systems Microsoft Windows and Sun Sparc Solaris.

Model Administrator

The model administrator is a special, password-protected user. Any project member can switch to model administrator mode provided they know the password. This way, the role of the model administrator is not necessarily bound to a specific person.

The model administrator can display all users logged-in to a model and the number of elements locked by these users. In addition, the system indicates the number of Web users who have logged in to the model via a Web login.

Password Protection

All users can specify their own passwords. The model administrator can identify the users who have not defined a password. A password can be deleted by the user owning the password or by the model administrator.

Access Rights

The right to change model elements is defined via user groups. It is possible to specify individual model elements. Users who do not belong to a user group do not have the right to change elements. Users can be members of several groups. This way, one and the same person can assume different roles in terms of the model and have the right to access various parts of the model. In addition, it is possible to change roles with all windows remaining open.

Privileges

In addition to the modification rights, special command privileges may be assigned to individual user groups (versioning of model elements, assignment of labels, definition of verification configurations).

The Innovator status line provides information on the status of the element displayed in all view modes. The Model Browser gives a complete overview of which elements are currently locked by which user.

In Innovator Business a special right controls whether the members of a group may view the configuration model.

Re-Use

Users and user groups with the corresponding privileges may be saved just like other model configurations and used as templates or be reloaded in other models.

Integrated Message System

All Innovator users can communicate across model and repository boundaries via an integrated message service.

In addition, it is possible to configure automatic e-mail messages that are triggered by specific events.

Version Management

Version Management Repository

All elements of the Innovator online repository can be versioned. The users may access these version objects at any time. With the version objects, Innovator also provides a convenient way to reuse defined information as templates in new projects.

In developing new software or maintaining existing systems, you frequently need different versions of elements or groups of elements of a project or a model, e.g. for prototyping purposes. Depending on whether a new version turns out to meet the requirements, it will either be kept or rejected in favor of one of its predecessors. In some cases, it may also be necessary to develop different versions of an element in parallel, e.g. a class with its implementation.

The Innovator version management repository supports this type of task. Groups of elements or individual elements from a project or a model as well as external objects may be checked into the Innovator version repository in different versions. The only limitation that applies to external objects is that such objects must have ASCII format. When checking in an element, the user may create a note that can, for example, include the reason for the element being checked in.

In addition to checking in and out, the following functionalities are available for version objects:

- Assigning keywords
- Deleting version objects and groups
- Releasing and locking version objects in order to enable the re-use of version objects
- Displaying, searching and comparing version objects

When version objects are checked out, released or deleted, you can search by keywords or by persons who checked in the version object.

Version Management Browser

The Version Management Browser shows the contents of a version repository, similar to a file system. In the version management server, the concept of a folder in a file system is represented by the management node which serves to group version objects. Management nodes can be structured hierarchically.

Version objects are assigned to a management node. If a version object is checked in below a management node that already contains a predecessor version of this object, a new version of the object is created. Only the delta information differing from the previous version is stored.

An object may also be stored below different management nodes. Since objects are not versioned globally but only below a management node, it is possible to keep and implement different instances of an object in parallel in the version repository.

In addition to the Innovator Version Management Browser, you may also use the Innovator project or model tree in order to move individual objects and complete object groups from the model repository to the version repository and vice versa. This way, older versions can be edited in the model repository and written back to the version repository.

It is also possible to integrate Innovator with external version and configuration management systems (refer to the chapter "Other Interfaces").

Displaying Version Differences

In a difference comparison tool, the logical differences between models or parts of models can be compared with each other or with the current online model.

You can use Innovator elements from the following sources for making comparisons:

- Active model repository
- Active version repository
- Version files
- Defined search

You can specify which structures should be compared within the input amount using a structure definition.

As well as ASCII object files, you can also specify the differences between archived object files of *.tob type.

The results of such comparisons are displayed in a user interface, similar to the Model Browser.

- The top left area (Package Area) shows the tree structure of the compared elements.
- The top center area (Element Area) shows a list of the Innovator elements assigned to the package selected in the Package Area.
- The top right area (Element Detail Area) shows the sub elements contained in the element selected in the center Element Area. The information is displayed in the form of a tree structure and varies, depending on the type of element selected.
- The bottom area (Properties Area) displays the properties of the element selected in one of the top areas in the form of a list.

The four possible states (new, old, different and identical) of an element or a property are indicated by means of icons in the four areas. In addition, you can configure different colors for the states.

Version Files for any External Version Systems

Partial models can be combined with a version object and transferred to any external version systems.

In the model browser, you can combine any selection of model elements in a single version file. Using the adjustable command lines for preparation and reworking, this version file can be transferred to any external version system or retrieved in the model from there.

Configuration for export or import in the respective dialog cannot be saved. For the export you can e.g. include a search for selection specification. The configuration is saved.

In the eXcellence model's configuration editor, analog profiles can be exported and imported as version files.

Migration

To migrate existing projects in the current Innovator version, use the repository transformer, which automatically customizes model data to the altered data structures and creates repositories for the current Innovator version.

Before this transformation, arrange a project license server for the previous Innovator version for existing projects in the parallel reinstallation.

Detailed information about migration is documented in the respective migration manual (German). If necessary, use this manual to gather information about preliminary work and re-workings to eliminate data loss.

Generation of Documentation

Customizable Templates

The documentation generator is an integrated component of the Innovator tools. It allows you to generate documentation from the contents of models, based on any number of definable documentation templates. The creation of templates is integrated into the configuration editor for eXcellence editions; the available configuration can be directly accessed. This means that e.g. sub chapters which can never deliver contents according to the configuration are not primarily offered for the structure.

Structure Definition

The structure of the documentation is defined in such a way that it can be used for all projects. It can easily be customized to comply with existing user-specific or organization-specific documentation guidelines. The structure definition comprises, among other things, title pages, headers and footers, user chapters, table of contents and index.

Chapter list with sequence, numbering and headings. Page breaks can be specified. It is possible to insert new chapters into this structure.

The creation of documentation structures is supported by means of help texts which explain the contents of the next structural level to be shown.

Title Pages

The system supports title pages including the import of external graphics.

Headers and Footers

The contents and layout for headers and footers can be specified. The headers and footers may consist of several lines; it is also possible to import logos and other graphics. Field functions are available for the page number, the date, the time and other information.

User Chapters

User-specific or method-independent chapters which are not directly generated by the Innovator editions, such as maintenance conditions or QA directives, can be inserted. These chapters may be entered via the integrated Innovator text editor or integrated from external files by means of a reference mechanism. The user can choose the most suitable format, depending on the final output format of the documentation, from ASCII files, MS Word files or PNG or EPS graphics files.

Table of Contents and Index

The table of contents and the index are the base components of the chapter structure.

Cross references are automatically created by Innovator (with hyperlinks in HTML files and book marks in Word documents).

Templates

In order to ensure that the generated documentation always meets the requirements of the organization's documentation guidelines, it is possible to save the structure, formatting information and output format as documentation commands or templates and made available as menu items. Users can only use these templates to generate documents.

Formatting

Paragraph formats such as left or right aligned, centered or justified are available for all types of document sections, both method-dependent and user-specific. It is also possible to specify page breaks and to use bulleted lists (two levels).

Output Formats

Innovator uses the structure definitions and the current project contents in order to create a documented snapshot of the model. Contents and sequence of the desired model information are specified individually. Search results and selection filters can be used for the comfortable determination of model contents.

The information concerning the model contents is then converted depending on the configured output type. The available outputs comprise the Preview Window, Word, XML, Postscript, ASCII and direct printout.

Preview Window

The Preview Window shows the entire documentation in a WYSIWYG format. In the Preview Window, the following formatting parameters can be changed: fonts, underline spacing, page layout such as size and margins. In addition, it is possible to export the document to a Postscript or graphics format.

Microsoft Word

Microsoft Word documents can be created with graphics in the following formats: PNG, SVG, EMF (Windows platforms only) or EPS.

The templates in Innovator make it easy to adapt to the organization's standards for Word documents.

XML

In addition to the output formats Preview Window and MS Word, Innovator can generate the documentation as an XML document which may be converted to HTML by means of an XSL transformation. A sample XSL stylesheet ships with Innovator.

Postscript

The documentation can be optionally generated in Postscript format.

ASCII

ASCII is another possible output format for the documentation. By means of filters or Tcl scripts, this output can be converted to other common formats such as RTF, HTML and FrameMaker MIF.

Export of Graphics

Graphics can be exported for re-use in a variety of formats. Innovator supports the following formats:

- EMF
- EPS
- PNG
- SVG

Under Microsoft Windows, the diagrams can be copied to the Clipboard via the EMF format and directly pasted into other Windows applications for further processing.

Generation

Single Click Generation

Documentation commands and templates can be configured as menu commands in the Model Browser. After selection in the Model Browser of the model contents to be documented, a single click is sufficient to generate the documentation.

It is also possible to include a search in the documentation command or template so that the scope of the documentation is preset. This ensures uniform documentation.

Quick Reports

If you can do without complete references and verification messages in the generated documents, you can accelerate the generation procedure of documentation for parts of models.

Printing

A Page Preview function with adjustable page sizes and cross-page layout and positioning functions is available for convenient print layouting.

The diagrams and tables can be printed directly to a printer to be selected or exported as graphics. In addition, it is possible to batch several diagrams or tables from the Model Browser at the same time.

Direct Printing/Print File

It is also possible to generate a print file in the format of the configured printer or to print directly.

V-Modell® XT Support

The V-Modell® XT is the successor of the V-Modell® 97. The model defines procedures, the corresponding deliverables and roles for planning and implementation of IT projects.

Innovator consistently supports the planning and development process with several analysis and design methods. It is the ideal tool to create and document the deliverables in compliance with the specifications.

Innovator supports the implementation of IT projects according to the V-Modell® XT by providing corresponding documentation templates for the models for all relevant V-Modell® XT deliverables. External documents can be integrated in the documentation.

Innovator Object eXcellence

Features	Innovator Object eXcellence
Methodology	
Unified Modeling Language (UML 2.1)	✓
Editors and Representation Types	
Administration program for licenses and repositories	✓
Model browser with model tree view, list of model elements and search engine	✓
Configuration Editor	✓
Package diagrams	✓
Class diagrams	✓
Object diagrams	✓
Use Case diagrams	✓
Component diagrams	✓
Deployment diagrams	✓
Activity diagrams	✓
State diagrams	✓
Sequence diagrams	✓
Composite structure diagram	✓
Specification editor	✓
Data Management, User Administration and Licensing	
Multi-user online repository	✓
Integrated version management	✓
Semantic and syntactic quality assurance	✓
Model-wide user and user group administration	✓
Floating licensing	✓
Web access (read)	✓
Import/Export Interfaces	
Artifacts (model, script, code file, binary file, data bank table etc.)	✓
Export based on XMI (UML 2.1)	Object eXcellence
Integration of implementation tools (JBuilder, oAW, Eclipse)	✓
.NET support (C# code generation, .NET API, integration in Visual Studio)	✓
Integration of ARIS Design Platform	✓
Generation of Documentation	
Integrated generation of documentation	✓
Documentation compliant with V-Modell®XT and particular process models with Innovator Report	✓
Customizable documentation templates tailored to your corporate identity	✓
Export of graphics (EMF, EPS, PNG, SVG)	✓

Features	Innovator Object eXcellence
Configuration, Model Management, Metamodeling	
UML profile support	✓
Hierarchical packages structures based on profiles	✓
Programming Languages	
Java, C++, C#, C	✓
BPEL	✓
Engineering Techniques	
Architecture/model-driven incremental forward engineering	✓
Model-driven transformations and implementation support	
Generative and associative transformations	✓
System Platforms for Client and Server	
Windows (2000, Server 2003, XP)	✓
Linux	✓
Solaris	✓

Object-Oriented Modeling with UML 2.1

Innovator Object eXcellence builds on the UML 2.1-compliant metamodel of the online repository. The metamodel also serves as the basis of the general functionalities described for Innovator eXcellence. Innovator Object eXcellence supports the major UML 2.1 diagram types and elements. Modeling of constraints, including syntactical verification, is supported in behavior diagrams using OCL (Object Constraint Language).

UML as Standard

The Object Management Group (OMG) standardized the Unified Modeling Language (UML) for the modeling of object-oriented systems.

The Innovator edition Object eXcellence provides support along all stages in the development of software systems on the basis of UML 2.1.

Modeling

The Innovator edition Object eXcellence enables object-oriented modeling using the following UML 2.1 modeling techniques:

Packages and Package Diagrams

According to the UML specification, a package is a mechanism used to group elements. Each package can contain elements (including packages). Package diagrams are used to show the dependencies between packages.

Packages can be used as namespaces, i.e. elements of a given type must only have a unique name within the package. In order to uniquely identify an element, the name and the complete path of all packages are required.

Class Diagrams

The class diagram is a central component of UML. It describes the static structure of the classes and interfaces of a system as well as the relationships between them (associations and dependencies).

Object Diagrams

The object diagram shows instances of classifiers (classes, interfaces, components etc.) at a particular point in time. It does not model the complete instance but a relevant section. Associations on classifier level are modeled by links between the instances.

Use Case Diagrams

A use case diagram shows the relationships between stakeholders and a set of use cases. Elements in use case diagrams can be linked by means of relationships (associations, interactions, generalizations, Include or Extend dependencies).

Component Diagrams

Components are a key aspect in modeling a system. A component is a replaceable part of a system that corresponds to a set of interface specifications and realizes them.

Deployment Diagrams

The deployment diagram shows the allocation of artifacts to hardware units. Hardware units are linked by communication connections. The installation, configuration, supply and execution of artifacts in the hardware environment can be represented.

Activity Diagrams

An activity diagram describes the activities required in executing a use case. Activity diagrams use the semantics of Petri nets and implement data flows by means of tokens. So-called swim lanes are supported to enable the modeling of responsibilities. Connectors support the clarity.

State Machines

A state machine diagram describes the dynamic behavior of the classes defined in the class model. A state machine diagram is assigned to a class and describes the behavior of its instances.

Sequence Diagrams

A sequence diagram describes the interaction of various objects. The interchange of messages between objects is depicted chronologically. The chronological sequence of the events (messages) is expressed by the top-down order in the diagram. Timing constraints and length of timing constraints can be used.

Composite Structure Diagram

The composite structure diagram shows the shows the internal structure of a classifier and its interaction with its environment.

Artifacts

Artifacts represent physical information units. This can be e.g. a model, a source coding file, a script, a binary file or a table of a relational data base. Artifacts can possess a correspondence on the file system. In Innovator the packages of such artifacts are regarded as listings of the file system. Therefore a file assigned by an artifact can be opened and edited from Innovator.

Languages

Code Generation

Innovator Object eXcellence supports model-driven code generation according to MDSD (Model-Driven Software Development) as architecture/model-driven, incremental forward-engineering.

The Innovator eXcellence metamodel is based on UML 2.1 and serves as the basis for code generation. Source code is automatically generated from domain-specific Innovator models using the open, standard and template-based language XPand by openArchitectureWare (http://architecturware.sourceforge.net).

This directly links the generator metamodel with Innovator and, in doing so, ensures high-performance, dynamic access to the Innovator models during template evaluation.

Templates for the programming languages Java, C++ or C# are included within the scope of delivery of Innovator; C code can also be created.

BPEL Export

With the "Business Process Execution Language for Web Services" (BPEL4WS), IBM and Microsoft developed a description language for Web Service processes.
The OASIS organization further develops and standardizes this language (http://www.oasis-open.org/).

Innovator now provides the possibility to model the collaboration of web services and to transform the model to executable BPEL code. After defining the basic web services you want to build upon, you can define more complex web services by orchestrating the basic ones. This is done using UML 2.1 activities. Innovator then transforms the BPEL-specific design model to several XML files to be passed to a BPEL engine which conforms to standards, with which you can execute the orchestration processes.

UML Profiles

A profile allows the modeler to extend the UML and to adapt it to the requirements of special tasks and special process models. For such purposes, UML provides, among other extension mechanisms, stereotypes and stereotype properties. A profile contains the UML extensions. Profiles are a highly flexible and powerful mechanism and allow almost any type of adaptation and extension. Examples include:

- Tool configurations for process models
  such as Unified Process, MID Modeling Methodology M³
- Modeling processes
  such as OMG's Model Driven Architecture (MDA) (separation of platform-independent models and platform-specific models (PSM))
- Extensions for specific types of implementation
  such as EJB or Web Applications

Integration and MDA Transformations

Transformations (generative or associative) to/from Innovator Object eXcellence are supported for the following Innovator editions:

- From Innovator Business classiX
- From/to Innovator Data classiX, Data eXcellence

Please refer to the chapter "Integration and MDA Transformations" for detailed information on transformations.

XMI Interface

XML Metadata Interchange (XMI) is a standard from Object Management Group (OMG) and is used as an interchange format between software development tools. The XMI export in accordance with UML 2.1 is supported for data exchange with the subsequent applications in the modeling toolchain.

Innovator Object classiX

Features	Innovator Object classiX
Methodology	
Unified Modeling Language (UML 1.4)	✓
Editors and Representation Types	
Administration program for licenses and repositories	✓
Model browser with model tree view, list of model elements and search engine	✓
Package diagrams	✓
Class diagrams	✓
Use Case diagrams	✓
Activity diagrams	✓
State diagrams	✓
Object diagrams	✓
Sequence diagrams	✓
Collaboration diagrams	✓
Component diagrams	✓
Specification editor	✓
Data Management, User Administration and Licensing	
Multi-user online repository	✓
Integrated version management	✓
Semantic and syntactic quality assurance	✓
Model-wide user and user group administration	✓
Floating licensing	✓
Web access (read)	✓
Import/Export Interfaces	
Integration of external objects	✓
XMI export based on Document Type Definition (DTD)	UML 1.4
Integration of implementation tools (WSAD, JBuilder)	✓
Generation of Documentation	
Integrated generation of documentation	✓
Documentation compliant with V-Modell®XT and particular process models with Innovator Report	✓
Customizable documentation templates tailored to your corporate identity	✓
Export of graphics (EMF, EPS, PNG, SVG)	✓
Configuration, Model Management and Metamodeling	
UML profile support	✓
Hierarchical packages structures based on profiles	✓
Programming Languages	
Java, C++, CORBA IDL	✓

Features	Innovator Object classiX
Engineering Techniques	
Architecture/model-driven incremental forward engineering	✓
Round-trip engineering for supported programming languages	✓
Reverse engineering for supported programming languages	✓
Model-Driven Transformations and Implementation Support	
Generative and associative transformations	✓
System Platforms for Client and Server	
Windows (2000, Server 2003, XP)	✓
Linux	✓
Solaris	✓

Object-Oriented Modeling with UML 1.4

UML as Standard

The Object Management Group (OMG) standardized the Unified Modeling Language (UML) for the modeling of object-oriented systems.

The Innovator edition Object classiX provides support along all stages in the development of software systems on the basis of UML 1.4.

Modeling

The Innovator edition Object classiX enables object-oriented modeling using the following UML 1.4 modeling techniques:

Packages and Package Diagrams

According to the UML specification, a package is a mechanism used to group elements. Packages are displayed in the overview tree. Each package can contain elements (including packages). Package diagrams are used to show the dependencies between packages.

Packages can be used as namespaces, i.e. elements of a given type must only have a unique name within the package. In order to uniquely identify an element, the name and the complete path of all packages are required.

Class Diagrams

A class diagram shows classes, interfaces and their relationships. Innovator Object classiX provides numerous functions for class diagrams such as display filters, UML and OOP declarations, etc.

State Diagrams

State diagrams describe the dynamic behavior of the classes defined in the class model. A state diagram is assigned to a class and describes the behavior of its instances.

Use Case Diagrams

A use case diagram shows the relationships between stakeholders and a set of use cases in a closed system. Use cases can be further specified by means of sequence, collaboration and activity diagrams.

Elements in use case diagrams can be linked by means of relationships (interactions, generalizations, Include or Extend dependencies).

Sequence Diagrams

A sequence diagram describes the interaction of various objects for a scenario of a use case. It specifies the internal view of a usage scenario, i.e. the way the scenario can be executed within the system. The interchange of messages between objects is depicted chronologically. The chronological sequence of the events (messages) is expressed by the top-down order in the diagram.

Collaboration Diagrams

A collaboration diagram expresses the structural organization of the objects that send and receive messages.

Activity Diagrams

An activity diagram describes the activities required in performing a use case. So-called swim lanes are supported to enable the modeling of responsibilities.

Object Diagrams

The static relationships between concrete objects are modeled in an object diagram.

Component Diagrams

Components are a key aspect in modeling the physical aspects of a system. A component is a physical and replaceable part of a system that corresponds to a set of interface specifications and realizes them.

Target Languages

Languages

The following languages are available for implementation:

- Java
- C++
- CORBA IDL

The namespace semantics are supported in a language-specific way.

Round-Trip Engineering

Round-trip engineering supports the alternating, repeated usage of Innovator Object and a development environment by ensuring the consistency of the data in case of a modification in one environment and an appropriate adaptation of the data in the other environment to restore consistency. Code is generated by means of Tcl scripts, i.e. the code generation functionality can be adapted to specific requirements.

Development Environments

Various development environments may be used to implement the results of the modeling process. All source code-oriented development environments such as Borland JBuilder are supported. Special add-ins for Microsoft VisualStudio and Eclipse exist.

UML Profiles

A profile allows the modeler to extend the UML and to adapt it to the requirements of special tasks and special process models. For such purposes, UML provides, among other extension mechanisms, Stereotypes and Tagged Values. A profile contains the UML extensions. Profiles are a highly flexible and powerful mechanism and allow almost any type of adaptation and extension. Examples include:

- Tool configurations for process models
  such as Unified Process
- Modeling processes
  such as OMG's Model Driven Architecture (MDA) (separation of platform-independent models and platform-specific models (PSM))
- Extensions for specific types of implementation
  such as EJB or Web Applications

Integration and MDA Transformations

Transformations (generative or associative) are supported for the following Innovator editions:

- From Innovator Business classiX
- To/From Innovator Data classiX
- To/From Innovator Object classiX

Please refer to the chapter "Integration and MDA Transformations" for detailed information on transformations.

XMI Interface

XML Metadata Interchange (XMI) is a text-based interchange format for metadata and data. The XMI export generates one or several XMI files with the corresponding model information from Innovator Object classiX (UML 1.4) according to the Document Type Definition (DTD) generated from the corresponding UML 1.4 specification.

Innovator Business classiX

Features	Innovator Business classiX
Methodology	
Unified Modeling Language (UML 1.4)	✓
Editors and Representation Types	
Administration program for licenses and repositories	✓
Repository browser for models and versioning	✓
Model browser with model tree view, list of model contents and search engine	✓
Package diagrams	✓
Activity diagrams (customizable)	✓
Activity definition diagrams	✓
Object diagrams (customizable and enhanced by activities)	✓
Business Use Case diagrams	✓
Sequence diagrams	✓
Collaboration diagrams (customizable)	✓
Analytical and Simulative Evaluation	
Process costs with probability of reachability	✓
Running and waiting times with resource loads and other process ratios	✓
Data Management, User Administration and Licensing	
Multi-user online repository	✓
Integrated version management	✓
Semantic and syntactic quality assurance	✓
Model-wide user and user group administration	✓
Floating licensing	✓
Web access (read)	✓
TCL-API reading and modifying	✓
Import/Export Interfaces	
Integration of external objects	✓
XMI export based on Document Type Definition (DTD)	UML 1.4
XML export for Microsoft Project from Innovator Business	✓
Generation of Documentation	
Integrated generation of documentation	✓
Documentation compliant with V-Modell®XT and particular process models with Innovator Report	✓
Customizable documentation templates tailored to your corporate identity	✓
Export of graphics (EMF, EPS, PNG, SVG)	✓
Configuration and Model Management	
Configuration model based on a UML 1.4 class model	✓
Hierarchical package structures based on profiles	✓

Features	Innovator Business classiX
Process Model Support	
Project type with execution conditions	✓
Direct and indirect tailoring operations	✓
Generation of project manuals	✓
Engineering Techniques	
Forward engineering	✓
Implementation Support	
Transformation of business process model into structured and object-oriented software models	✓
Workflow support	✓
System Platforms for Client and Server	
Windows (2000, Server 2003, XP)	✓
Linux	✓
Solaris	✓

Business Process Modeling with UML 1.4

UML as Standard

Globalization and the requirements of international markets force organizations and companies in almost all industries to subject their business processes, services and communication channels to stringent and systematic analyses. The results of such analyses are intended to uncover weak spots and demonstrate optimization potential. This enables improved customer orientation as well as more efficient and economical processes.

A great number of studies, essays and methodologies have researched approaches to perform such a re-organization in the most comprehensive, effective and transparent way.

The UML (Unified Modeling Language) - a description language for models from the area of object-oriented software engineering - provides a systematic approach as well as notation and representation features that lend themselves to solve the requirements involved in business process analysis and optimization.

Innovator Business supports UML along all stages, from strategic planning, operative requirements and company organization all the way to the integration of an IT system.

As opposed to object-oriented modeling which uses classes as the main modeling elements, a business process model focuses on the dynamic aspects of processes and sequences (activities). Therefore, activity diagrams and the hierarchical grouping of activities constitute the central aspects in modeling the processes in an organization.

In addition, business use cases as well as object diagrams are used to model static relationships (such as company structures, organizations, etc.).

UML sequence diagrams and collaboration diagrams in various instances are available to model the communication between business processes or scenarios of a use case.

Configuration Model

Class Model

The activity, object and event types in the business process model are configured in a UML class model with class diagrams, classes, their stereotypes and property values. The possible static relationships with cardinalities as well as the attributes of the types which can be used in modeling the business processes are also specified here.

New instances of diagram types (based on UML activity diagrams, object diagrams, sequence diagrams or collaboration diagrams) can be defined via the configuration model and its package structure (Metamodeling Light).

Privileges

This configuration model serves as the basis for modeling business processes. It can only be viewed and modified by user groups with the appropriate privileges.

Customizability

Configuration models may be customized and extended by users with the corresponding privileges and loaded to serve as the basis for new models.

When business processes and organization structures are modeled, the classes defined in the configuration model are instantiated and used in activity diagrams, object diagrams and/or other UML diagram types.

UML 1.4 Extensions

Activity Types

It is possible to define activity types in a class model as a class by means of a special property value. This way, they have the same properties as objects (attributes, relationships, inheritance, etc.). When you model processes, activities are instances of their classes, just like objects. Concrete values can be assigned to the attributes of activities and objects and displayed in diagrams.

This way, activities can be used in object diagrams (e.g. activity breakdown), but also have relationships with other objects in activity diagrams (e.g. allocation of resources). A UML swim lane is a special type of such a relationship with a special notation. However, a complex business process model usually requires a more precise specification of the relationship between activities and objects. Therefore, the notation possibilities in activity diagrams were extended in such a way that relationships are possible between objects and activities.

An additional extension allows you to interchange messages between activities.

Activity Definition

The activity definition diagram is an optional diagram type that does not exist in UML. It is a special type of activity diagram which focuses on the activity to be defined.

The following properties are specified for the activity to be defined with the representation functions of the activity diagram:

- Initial conditions
- Messages received and sent
- Relationships with objects (resources, etc.)
- Results
- Product flow

These specified properties are then available when you model the processes. It is also possible to generate or update activity definition diagrams from previously modeled processes.

Conditions

Conditions are circumstances that may influence processes, but that are not results determined by performing an activity. In your models, you can branch to results of an activity or you can branch depending on such conditions in the process. Conditions are distinct model elements with specifications and labels.

Analysis

The objective of business process analysis is the determination of performance indicators (such as throughput times, probability values, costs, resource consumption, etc.) which are relevant for the optimization of the business processes. This enables the identification of weak spots in the modeled business processes.

Business process analysis uses two different methods:

- Analytical calculation
  In the analytical calculation, the performance indicators are determined on the basis of complex mathematical formulas. In this case, an individual pass through the activity diagram is analyzed.
- Simulation
  Simulation is used to analyze the collaboration of several activities in the model (which may result in concurring access to resources). The performance indicators are determined by means of statistical evaluation.

Diagram analysis tables and environment analysis tables are available to support business process analysis.

Analysis tables are used for input and output of the analysis or the simulation. Such analysis tables may contain the data relating to a single activity diagram (diagram analysis table) or the complete process (environment analysis table).

The data can be exported to MS Excel for further processing of the analysis/simulation results.

Workflow

The processes modeled with Innovator may be used for flow control by the workflow engines in the runtime environment. Innovator supports the modeling of components (UML component diagrams) to generate these sequences.

The XMI generation of the process information is the interface to all workflow engines which can import UML as the basis of their workflow model.

Process Models

A process model is a special business process that controls the execution of projects.

The V-Modell®XT or the Unified Process (OMG) are typical examples of such a process model.

The objective of a process model is to create a description of all artifacts of a project to be executed and of the processes used in creating these artifacts. The more diverse the requirements are in terms of the artifacts described in the process model, the more variants of the process can exist.

However, in addition to the artifacts that influence the process, conditions such as the scope or the complexity of the service to be provided affect the process. Adapting the model, i.e. selecting the process variants, is referred to as "tailoring".

Comprehensive projects may comprise such a large number of conditions and corresponding variants that additional support is required in tailoring the process model to the special requirements without endangering the consistency of the model. This means that predefined tailoring operations must be available. Modeling a process that can be adapted to different conditions must therefore contain tailoring operations that ensure model consistency. In addition, it is necessary to specify the conditions under which a given adaptation may be made.

Innovator not only allows the user to model individual tailoring operations, but also to combine numerous tailoring operations that are usually performed together into larger units (project types that are parameterized via execution conditions).

When a project is started, the project manager is responsible for tailoring the process model. This means that the project manager must check all possible prerequisites for tailoring the model and decide whether a possible adaptation is to be made. Upon completion of the tailoring activities, Innovator removes all the variants of the process model that are not required and produces a project manual, i.e. a process description that is tailored to the project.

Modeling and Metamodeling

Innovator Business supports business process modeling with the following modeling techniques:

Model Management

- Packages and package diagrams with namespaces

Configuration Model

- Class diagrams

Business Process Model

- Use case diagrams
- Activity diagrams with swim lanes, extended by conditions and activity-object relationships
- Sequence diagrams
- Collaboration diagrams
- Object diagrams (extended by activities)
- Activity definition diagrams
- Any other instance of diagram type based on object diagrams, activity diagrams, sequence diagrams and collaboration diagrams such as organization charts, value chains, infrastructure charts, task structure trees, etc.

Analysis

- Diagram analysis tables
- Environment analysis tables

Workflow Support

- Component diagrams
- XMI generation

Process Model Support

- Project types
- Execution conditions
- Direct tailoring operations
- Indirect tailoring operations

Special Profiles

Innovator Business ships with an industry-neutral basic configuration model. A configuration which you can use to access the MID Modeling Methodology M³ in the modeling at a business process level is also included.

In order to support the modeling of special processes, additional configuration models (profiles) are available (extra charge).

- SmartISO
  Modeling of processes for ISO certification
- SmartOffice
  Extended modeling of office processes
- SmartProduction
  Modeling of sequences for production processes

Integration and MDA Transformations

Transformations (generative or associative) are supported for the following Innovator Editions:

- From Innovator Business classiX
- To Innovator Object classiX
- To Innovator Data classiX
- To Innovator Function classiX

The methodical basis for these transformation is the MID Modeling Methodology M³.

Please refer to the chapter "Integration and MDA Transformations" for detailed information on transformations.

XMI/XML Interfaces

XMI Export

XML Metadata Interchange (XMI) is a text-based interchange format for metadata and data. The XMI export generates one or several XMI files with the corresponding model information from Innovator Business classiX according to the Document Type Definition (DTD) generated from the corresponding CWM specification.

XML Export for Microsoft Project

The XML export from Innovator Business for Microsoft Project allows you to base project management in Microsoft Project on quality-assured Innovator process models. Innovator can export any process hierarchy of a business model as an XML file. This file is compliant with Microsoft's XML scheme for Microsoft Project.

The export considers activities as tasks or summary tasks, the attribute for the duration of the activity as estimated task duration. It also considers the required resources required as well as the created objects.

Innovator Data eXcellence

Features	Innovator Data eXcellence
Methodology	
Entity Relationship and Structured Entity Relationship (ER/SER)	✓
Editors and Representation Types	
Administration program for licenses and repositories	✓
Model browser with model tree view, list of model elements and search engine	✓
Configuration editor	
Package diagrams	✓
Entity Relationship Diagram in accordance with Chen, Martin, data structure analysis (DAS) or UML 2	✓
Structured entity relationship diagrams	✓
Database diagrams	✓
Specification editor	✓
Data management, User Administration and Licensing	
Multi-user online repository	✓
Integrated version management	✓
Semantic and syntactic quality assurance	✓
Model-wide user and user group administration	✓
Floating licensing	✓
Web access (read)	✓
Import/Export Interfaces	
Artifacts (model, script, database table etc.)	✓
Import of files, format	SQL/DDL
RDBMS access via JDCB	CWM
Java and C# API	
Generation of Documentation	
Integrated generation of documentation	✓
Documentation compliant with V-Modell®XT and particular process models with Innovator Report	✓
Customizable documentation templates tailored to your corporate identity	✓
Export of graphics (EMF, EPS, PNG, SVG)	✓
Configuration, Model Management, Metamodeling	
Hierarchical package structures based on profiles	✓
Programming Languages and Target Systems	
DB2 (all platforms), ORACLE, Informix, MS SQL-Server, other databases possible	✓
Engineering Techniques	
Forward engineering for supported target systems	✓
Reverse engineering for supported target systems	✓
Model-Driven Transformations and Implementation Support	
Generative and associative transformations	✓

Features	Innovator Data eXcellence
System Platforms for Client and Server	
Windows (2000, Server 2003, XP)	✓
Linux	✓
Solaris	✓

Data Modeling with ERM and SERM

Entity-Relationship Modeling (ERM) according to Chen is the standard method for semantic data modeling. Modeling of IT systems is one of the key applications of this method.

Modeling

One modeling concept is provided along with the appropriate diagrams for the conceptional schema and database system respectively.

The models are independent from one another, uncoupled and can be converted into each other through automated model trasitions.

The target system for technical attribute types or the desried databases can be configured and can be used model-specifically.

Conceptual Schema

A conceptual schema describes the requirements of the users in an implementation-independent way. It supports modeling with entities, attributes, relationships, keys and semantic data types.

Methods

Entities are displayed with adjustable sizes and compartments e.g. for attributes and foreign keys. Compartments can be blended out and can be automatically maintained by the system when contents are created.

Notations from Chen, James Martin, data structure analysis (DSA), UML and the structured entity relationship model (SERM) are supported.

ER/SER

In the classical ER model, the entities can be positioned anywhere in the diagram. In the structured extension, the SER model, the left-right arrangement of entities represents existence dependencies.

SER modeling offers a number of key advantages over classical ER modeling:

- Modeling of cyclical existence dependencies is prevented
- Transformation of the conceptual data model into a relational database system is facilitated
- Existence dependencies are visualized
- Complex data models are expressed in a clearly structured way

The objective of conceptual data modeling is to create a normalized data model without redundancies. Redundancies must be removed to ensure that data is modified only at a single point in the system and to maintain data integrity.

Database Independency

In order to support the independence of the conceptual schema from a special database, semantic data types are modeled. They describe the information to be stored, e.g. ZIP code instead of an n-digit number. A semantic data type instead of a fixed database type can be assigned to each attribute:

- A semantic data type can be assigned to several attributes
- Database-specific types are assigned to a semantic data type

Changing a semantic data type modifies all attributes and tables that are based on this data type.

Database Schema

The conceptual schema is mapped to a concrete database system at the level of the database schema. Tables, columns, views, primary and foreign keys, indices, triggers, stored procedures and access authorizations are supported in database schema. As well as semantic data types, direct data types are also available from the target systems for typings.

Database tables and views can be displayed using adjustable sizes and compartments, e.g. for table columns and foreign keys. Compartments can be hidden and automatically maintained.

The notation IDEF1X is supported.

Modeling in the Database Schema

Depending on the use case, one or more database schemas are possible for one conceptional schema.

This means that each target system (e.g. Oracle, DB2) can have its own database schema but camn also use a common database schema for various target systems. The conversin of datatypes is carried out according to the rules set in the target system's configuration.

The model transition from conceptional to database schema can be fully automated. Dependency relationships are created for the conceptional schema's model elements or a new alignment is updated for this; this ensures traceability.

Dependency relationships can be manually verified and maintained. The dependency editor supports the user through this function.

Functions, such as the combining and splitting of database tables (split columns, split rows) also support modeling in the database schema.

Database authorization concepts, such as authorization and grants, user, groups, user roles and privileges can be stored and evaluated in the model.

Configuration of Type Systems

New databases can be easily included in Innovator.

The implementation-specific information relates, among other things, to the following properties:

- Data types
- Mapping rules for data typing other type systems
- Database options

Database Tables

In addition, the model can be extended by technical attributes and database tables in the database schema.

Denormalization

Denormalization is also possible, for example by combining several tables or splitting tables into columns or rows.

Database Views

Database views are linked to database tables and/or database views using FROM clauses. The FROM clauses are displayed in the database diagram as directed edges

Generation and Reverse Engineering

- SQL/DDL generation
- DDL generation for create table and alter table
- Generation of SQL-Create-View
- Reverse engineering of SQL/DDL
- Reverse engineering of SQL/Views

Supported Databases

- DB2, DB2/UDB
- Oracle (including reading direct access)
- Informix
- MS SQL-Server
- Support for other relational databases can be easily implemented

Target Language Types

- Java
- extensible

Integration and MDA Transformations

A transformation is provided for supporting object-oriented application with Innovator Object eXcellence. It creates or adjusts the class model’s elements to the elements in the data model. Please refer to the chapter "Integration and MDA Transformations" for detailed information on transformations.

Innovator Data classiX

Features	Innovator Data classiX
Methodology	
Entity Relationship and Structured Entity Relationship (ER/SER)	✓
Editors and Representation Types	
Administration program for licenses and repositories	✓
Model browser with model tree view, list of model elements and search engine	✓
Package diagrams	✓
Entity Relationship and Structured Entity Relationship diagrams	✓
Entity tables	✓
Database tables and views	✓
Specification editor	✓
Data management, User Administration and Licensing	
Multi-user online repository	✓
Integrated version management	✓
Semantic and syntactic quality assurance	✓
Model-wide user and user group administration	✓
Floating licensing	✓
Web access (read)	✓
Import/Export Interfaces	
Integration of external objects	✓
Import of files, format	SQL/DDL
XMI export based on Document Type Definition (DTD)	CWM
Generation of Documentation	
Integrated generation of documentation	✓
Documentation compliant with V-Modell®XT and particular process models with Innovator Report	✓
Customizable documentation templates tailored to your corporate identity	✓
Export of graphics (EMF, EPS, PNG, SVG)	✓
Configuration, Model Management, Metamodeling	
Hierarchical package structures based on profiles	✓
Programming Languages and Target Systems	
C and COBOL	✓
DB2 (all platforms), ORACLE, Informix, MS SQL-Server, other databases possible	✓
Engineering Techniques	
Forward engineering for supported programming languages and target systems	✓
Reverse engineering for supported target systems	✓
Model-Driven Transformations and Implementation Support	
Generative and associative transformations	✓

Features	Innovator Data classiX
System Platforms for Client and Server	
Windows (2000, Server 2003, XP)	✓
Linux	✓
Solaris	✓

Data Modeling with ERM and SERM

Entity-Relationship Modeling (ERM) according to Chen is the standard method for semantic data modeling. Modeling of IT systems is one of the key applications of this method.

Modeling

Innovator Data provides support in all tasks from creating normalized database schemas, external schemas, physical database schemas and database views all the way to generating the results for the database.

Conceptual Schema

A conceptual schema describes the requirements of the users in an implementation-independent way.

Methods

The schema is expressed in Innovator either as an Entity-Relationship Model (ER model) according to Chen or as a Structured Entity-Relationship Model (SER model) according to Sinz.

ER/SER

In the classical ER model, the entities can be positioned anywhere in the diagram. In the structured extension, the SER model, the left-right arrangement of entities represents existence dependencies.

SER modeling offers a number of key advantages over classical ER modeling:

- Modeling of cyclical existence dependencies is prevented
- Transformation of the conceptual data model into a relational database system is facilitated
- Existence dependencies are visualized
- Complex data models are expressed in a clearly structured way

Innovator supports two diagram types which can be transformed into each other so that both types of representation can be used.

The objective of conceptual data modeling is to create a normalized data model without redundancies. Redundancies must be removed to ensure that data is modified only at a single point in the system and to maintain data integrity.

Notations

Innovator Data supports the following configurable notations for the diagrams of the conceptual schema:

- Chen
- Data Structure Analysis (DSA)
- James Martin
- Sinz (SERM)

Database Independency

In order to support the independence of the conceptual schema from databases, Innovator is used to model semantic data types. These data types, referred to as data elements, describe the information to be stored, e.g. ZIP code instead of an n-digit number. A data element instead of a fixed database type can be assigned to each attribute:

- A data element can be assigned to several attributes
- Database-specific types are assigned to a data element

Changing a data element modifies all attributes and tables that are based on this data element.

Data elements can be combined, represented and edited in data element tables.

Database Schema

The conceptual schema is mapped to a concrete database system at the level of the database schema.

Database System

Implementation-specific information is added to account for the target database system.

New databases can be easily included in Innovator.

The implementation-specific information relates, among other things, to the following properties:

- Attribute properties: special database types and default values
- Table properties: table sizes, etc.
- Index properties: attribute order, sorting and uniqueness

Database Tables

In addition, the model can be extended by technical attributes and database tables in the database schema.

Denormalization

Denormalization - for example by combining several simple tables of the conceptual schema into a single database table - can also be performed at this level.

Database Views

Database views are an essential means of aggregating, grouping, sorting and evaluating information from a relational database system. Database views are defined as database queries.

Database views have the following functions:

- Simplification of the access to database systems
- Flexibility and independence of the database access
- Protection of the database system

Database views are virtual tables. They contain columns that are similar to the attributes of database tables. However, as opposed to a database table, the columns are defined by means of derivation from or adoption of attributes of database tables or columns of other database views. These columns may themselves be used as the source for other database views.

The View columns and the FROM clauses as well as the subsequent WHERE clause are displayed and edited in the Innovator table editor. Innovator provides assistance in creating the SQL View statement by letting you select database tables, individual attributes or all attributes of a table for the database view. References to elements of the physical data model are maintained by Innovator in the database views.

Existing CREATE-VIEW-SQL statements are syntax checked (SQL92-Parser). If the syntax checker detects an error, the statement can still be saved as an unchecked database view. This way, Innovator can handle all views.

If the syntax check is successful, simple database views (e.g. no nested SELECT or UNION clauses) can be graphically displayed by Innovator. Syntactically incorrect database views are shown as plain text.

External Schema

The functions of database applications frequently operate on overlapping extracts of the database schema.

Extracts

Such overlapping extracts are referred to as external schemas or, in Innovator, external views.

Application View

An external view describes the specific extracts of the conceptual schema important for specific applications or projects. Innovator lets you model such external views on the basis of the conceptual schema.

Data Types for SA

The external views created this way can be used in the data dictionary with a functional model for which a Structured Analysis (SA) and a Structured Design (SD) are created. This ensures the consistency of the data structures between the data model and the functions.

In addition, this approach decouples the application from the data model.

Generation and Reverse Engineering

- SQL/DDL and SQL/DML generation
- Generation of SQL-Create-Views
- Reverse engineering of SQL/DDL
- Reverse engineering of SQL/Views
- Generation of type definitions for C
- Generation of COBOL copybooks

Supported Databases

- DB2, DB2/UDB
- Oracle
- Informix
- MS SQL-Server
- Support for other relational databases can be easily implemented

Target Language Types

- C
- COBOL
- Java
- extensible

Integration and MDA Transformations

Transformations (generative or associative) are supported for the following Innovator classiX editions:

- From Innovator Business classiX
- From/To Innovator Object classiX
- From/To Innovator Object eXcellence
- From/To Innovator Data classiX
- To Innovator Function classiX

Please refer to the chapter "Integration and MDA Transformations" for detailed information on transformations.

XMI Interfaces

XML Metadata Interchange (XMI) is a text-based interchange format for metadata and data. The XMI export generates one or several XMI files with the corresponding model information from Innovator Data classiX according to the Document Type Definition (DTD) generated from the corresponding CWM specification.

Innovator Function classiX

Features	Innovator Function classiX
Methodology	
Structured analysis and design (SA/SD) with real-time extension (RT)	✓
Editors and Representation Types	
Administration program for licenses and repositories	✓
Model browser with model tree view, list of model elements and search engine	✓
Package diagrams	✓
Data and control flow diagrams	✓
State diagrams	✓
Operation diagrams	✓
Module diagrams	✓
Data Dictionary	✓
Decision tables	✓
Process activation tables	✓
Nassi-Shneiderman diagram editor	✓
Specification editor	✓
Data Management, User Administration and Licensing	
Multi-user online repository	✓
Integrated version management	✓
Semantic and syntactic quality assurance	✓
Model-wide user and user group administration	✓
Floating licensing	✓
Web access (read)	✓
Import/Export Interfaces	
Integration of external objects	✓
Integration of implementation tools	✓
Integration of MATLAB and Simulink	✓
Generation of Documentation	
Integrated generation of documentation	✓
Documentation compliant with V-Modell®XT with Innovator Report	✓
Customizable documentation templates tailored to your corporate identity	✓
Export of graphics (EMF, EPS, PNG, SVG)	✓
Configuration, Model Management, Metamodeling	
Hierarchical package structures based on profiles	✓
Programming Languages	
C and COBOL	✓

Features	Innovator Function classiX
Engineering Techniques	
Architecture/model-driven incremental forward engineering	✓
Reverse engineering for C	✓
Model-Driven Transformations and Implementation Support	
Generative and associative transformations	✓
System Platforms for Client and Server	
Windows (2000, Server 2003, XP)	✓
Linux	✓
Solaris	✓

Function-Oriented Modeling with SA/RT/SD

Structured Analysis

Innovator Function supports the Structured Analysis (SA) method introduced by DeMarco. This method has been established as a worldwide standard for function-oriented system analysis.

System analysis comprises all tasks involved in fully analyzing user requirements and modeling the corresponding system. The result of system analysis is a functional specification of the system to be implemented. This functional specification contains all requirements which are independent of the implementation technology to be used. These technology-independent requirements are described with the Structured Analysis.

The approach uses the principle of hierarchical refinement of processes together with a data dictionary for the definition of information flows. Elementary processes can be described by means of specifications or decision tables.

Structured Design can be used to transform the result of system analysis into a system design.

Real-Time Extension

In specifying the requirements, Structured Analysis focuses on the processing of data. Due to a lack of appropriate expression means, flow control and specification of events that require immediate response are difficult to describe. Real-time extension to Structured Analysis solves this problem. The extension provides, for example, control flows which control the system behavior, as well as decision and process activation tables and state transition diagrams.

Runtime requirements tables are also supported to specify a system's response behavior.

Innovator Function implements all extensions according to Hatley/Pirbhai. In addition to technical automation systems, the Real-Time Method (RT) is increasingly used in commercial systems to model time or event-oriented sequences.

Structured Design

While system analysis is an implementation-independent and software-neutral activity, decisions concerning the actual implementation of the system are made in the design phase. The system design describes the architecture of the software system to be developed. Functions, logical components and modules are defined and the semantics of the flow of information (function calls, input parameters, return values) specified. It is possible to distinguish internal functions (functions still to be implemented in the project) and external functions (functions already implemented outside of the project for re-use).

Modular Design

Innovator Function supports users in the design of their system on the basis of the Structured Design method. In order to support the design of complex software systems, Innovator supports the modular design level in addition to the functional call hierarchy. Function groups (modules) and their intermodular dependencies are described graphically at this level.

Modules and operations can be grouped according to any type of criterion via a package concept. Package dependencies can be shown graphically. Package diagrams can be used to easily describe the task/library structure of a system, for instance.

From Analysis to Design

If the system design was preceded by a system analysis, the results can be used without transformation. System design and system analysis are always consistent.

From Design to Implementation

Innovator supports the direct transition from the design to the implementation phase in one of the target languages C or COBOL.

Reverse Engineering

Existing C systems can be analyzed and converted into a Structured Design by means of reverse engineering.

Modeling

Innovator Function supports structured modeling with the following modeling techniques:

Model Management

- Packages and package diagrams

Structured Analysis

- Data flow charts
- Data dictionary
- Decision tables

Real-Time Extension

- Control processes
- Control flow charts
- Decision tables
- State transition diagrams
- Process activation tables
- Runtime requirements tables

Structured Design

- Modules, operations and logical components
- Calls, formal and actual parameters
- Operation diagrams
- Module diagrams

Implementation

- Nassi-Shneiderman diagrams
- Any editor can be integrated for implementation

Target Languages and Interfaces

Implementation Languages

- C
- COBOL

Generation

- Generation of Includes for C
- Generation of forward declarations for C
- Generation of COBOL copybooks
- Generation of the code frames for modules, operations and data

Reverse Engineering Structured Design for C

- Analysis of C programs and generation of a Structured Design

MATLAB and Simulink

- Linking of model elements to the simulation and code generation tools MATLAB and Simulink (MathWorks Inc); direct processing and management of the results in the Innovator model

Integration and MDA Transformations

Transformations (generative or associative) are supported for the following Innovator classiX editions:

- From Innovator Business classiX
- From Innovator Data classiX
- From/To Innovator Function classiX

Please refer to the chapter "Integration and MDA Transformations“ for detailed information on transformations.

Innovator Programming classiX

Methods

Nassi-Shneiderman

Innovator Programming enables the user to create and modify Nassi-Shneiderman diagrams (structure charts). This technique can be used to describe the sequence of algorithms.

Innovator Programming can be used to implement the results of Innovator Function. Analysis and design results are seamlessly integrated into the implementation process. These results can be used during the implementation of the system.

Generation of Source Code

The algorithms described graphically are automatically converted to C or COBOL source code, depending on the selected programming language.

Reverse Engineering

Existing C source code modules can be graphically processed or documented with the reverse engineering functionality.

Procedure

Integration

Innovator Programming enables the seamless integration of the analysis and design results into the implementation process. The programmer works with the automatically generated modules with the analysis/design specifications. Templates can be used to control the contents of the modules. The programmer completes the algorithms and then tests the modules or programs.

Compilers

Innovator Programming enables the easy integration of external programs (such as compilers or source code analysis tools), so that the user does not have to quit the program for testing. The errors are shown in a list and the program enables direct navigation to the targets. It is possible to integrate several different compilers.

Search/Replace

A Search/Replace function for regular expressions and structure elements is available for working with algorithms. The Search/Replace patterns are saved when the application is terminated so that they can be re-used.

Undo

An Undo/Redo function (10 actions) is provided for convenient corrections

Macros

Macros allow you to combine complex actions and activate them via a single menu command.

Structuring

Correct Program Structure

All control flow structures such as procedures, loops, decisions, etc. can be created via menu commands.

Due to the context-sensitive menu logic, it is impossible to create syntactically incorrect program structures. Depending on the selected language, only the available constructs are offered. This ensures a smooth learning curve.

Zoom functions are available for all structure blocks. It is possible to define named markers which serve as targets for convenient jump commands. The markers are kept when the file is saved. In order to ensure a clear structure of deeply nested IF-THEN-ELSE structures, an IF-THEN-ELSE-IF construct is available for C. Preprocessor constructs are available for C/C++.

Usability

Navigation

A navigation bar that can be hidden or shown as required can be used to visualize the contents of the structure chart by means of configurable properties which can be employed to jump to the appropriate sections in the structure chart.

Folding Environments

Folding environments and hierarchical Hide/Show functions allow you to structure complex issues and to keep the program documentation clear and legible.

Usability

Excellent navigability is ensured by convenient mouse, scroll bar and keyboard functions as well as jump commands.

A tab lets you manage open files. A Favorites menu is provided for the most frequently used files.

Syntax Highlighting

Different fonts and font colors for code, comments and keywords as well as construct-specific syntax coloring ensure excellent code readability. In addition, you can select whether you want the program to display comments only, code only or comments plus code in the structure chart. This is an important issue in the documentation.

Source Code Generation

Customizability

By means of parameter files, the generated source code can be adapted to a considerable degree to conform to corporate standards.

Comments

When working on existing algorithms, you can comment on parts of them. These parts are not compiled.

Scan Markers

Parameters control whether you work with structure chart files or if just scan markers are used in the source file. It is also possible to generate mere source code without scan markers.

Target Languages

Innovator Programming is available for the following target languages:

- ANSI-Standard C
- ANSI-Standard C++
- COBOL

Due to the parameterization possibilities, you can also use the Innovator Programming for creating shell procedures and similar artifacts.

Syntax Checking

You can check the code for syntactical correctness by calling external programs (such as compilers). In addition, it is verified whether the analysis and design specifications are adhered to in the module or the program.

Innovator Report classiX

Features	Innovator Report classiX
Methodology	
Import and visualization of the defined proceedings, results and roles for planning and execution of IT projects from the V-Modell®XT	✓
Detailing of V-Modell®XT products with project results	✓
Management of project results	✓
Editors and Representation Types	
Administration program for licenses and repositories	✓
Model browser with model tree view, list of model elements and search engine	✓
Data Management, User Administration and Licensing	
Multi-user online repository	✓
Integrated version management	✓
Model-wide user and user group administration	✓
Floating licensing	✓
Web access (read-only)	✓
Generation of Documentation	
Integrated generation of documentation	✓
Document structures and document templates	✓
Documentation compliant with V-Modell®XT	✓
V-Modell®XT-independent documentation	✓
User-defined topic structure with hierarchical structure	✓
Process Models	
V-Modell®XT	✓
Organization-specific process models	✓
System Platforms for Client and Server	
Windows (2000, Server 2003, XP)	✓
Linux	✓
Solaris	✓

V-Modell®XT Support

V-Modell®XT is a process model for planning and carrying out projects. By specifying specific, standardized procedures, related results and roles, the V-Modell increases project transparency, improves project management and lastingly increases the probability of success.

Innovator supports planning and development process with several integrated analysis and design methods and is therefore the ideal tool to provide and document the results in accordance with the requirements.

The realization of IT projects compliant with V-Modell®XT will be supported by corresponding documentation templates for all relevant products of V-Modell®XT.

Company-Specific Process Models

To support product documentation independently from V-Modell®XT, you can define various products from existing Word documents, split them into topics and thereby use the existing contents.

Conformity in the Documentation Process

A document repository is shipped with Innovator Report, which contains prepared document structures for user requirement and functional specifications for product types, system/software architecture, system/software specifications and database design. The document structure defines the structure of a Word document generated for a topic. Templates ensure conformity in the documentation process.

Functionality

- Import of the results from the V-Modell®XT editor and project assistant
- Assignment possibility for pre-structured Word documents to product
- Visualization and navigation through the V-Modell®XT or the organization-specific process model
- Allocation from project-participants to roles with import of their rights
- Flexible topic allocation with Innovator model types and models in all structure levels
- Automatic creation of initial products during the project initialization
- Additional instantiation of non-initial products to project need in the context of the V-Modell®XT requirements
- Direct editing of topics from the model browser
- Automated filling of modeled topics with model results from Innovator models
- Model-spreading jumping between topics and models (traceability)
- Access protection on products and topics in accordance with V-Modell®XT roles
- Organizational reports (e.g. degree of achievement of decision points)

- Automatic generation process for completed V-Modell®XT products
- Export of completed V-Modell®XT products for further use
- List of figures for illustrations generated by Innovator

Integration and MDA Transformations

Vertical Integration for Innovator classiX and eXcellence

Business-CASE Transformation

Innovator Business describes business processes at an organizational level that is independent of an IT system. The models created with Innovator Business represent the existing or planned process reality. In contrast, the methods of software engineering are used to model systems which are to support a business process, i.e. a business process constitutes the context for a software system supporting this process.

By accounting for the interdependencies of the two worlds and integrating the development of business processes and software systems, Innovator ensures maximum convergence and lets you create systems with increased total effectiveness.

Innovator supports this objective by means of a transformation mechanism used to map the models on the business process side and models on the software engineering side. In this context, what is meant by software engineering methods are the models created with Innovator Function, Innovator Data and Innovator Object.

Placeholder elements, so-called model-external references, which are visualized in the Model Browsers of the Innovator editions, show the relationships between the elements of the business process model and the elements of the other models. Mapping of elements, i.e. the creation of model references, can be performed manually by means of associative mapping or automatically by means of generative mapping.

Cross-model relationships (via placeholder elements) enable navigation. This way, you can always navigate between related elements across models and repositories and display the source or target element of a transformation.

Horizontal Integration for Innovator classiX

Object-Object Transformation

The Innovator object-object transformation is used to map a class model to a different class model.

The mapping process distinguishes between master and slave models. The master model is the model to serve as the reference model, the slave model is the model to be adapted.

The objective of an MDA transformation is to modify a definable set of elements of the slave model in such a way that it corresponds to the master model. However, other parts of the slave model are not to be influenced.

In addition, the transformation does not modify the parts of the master model to be mapped.

In many cases, it is not desirable to map the complete master model to the slave model. It is possible to define partial models in the master model which must contain the elements to be mapped. These partial models are defined via the system packages to be mapped. In addition, complex filter policies allow you to specify the elements of the partial model to be mapped to the slave model.

Re-use of class libraries of an object model in another model is a typical example of such a mapping process.

The model-external references enable bidirectional switching between the models at any time.

Object-Data Transformation

Object-oriented modeling has become a strategic technique in application development. However, corporate data is usually not managed in object-oriented database systems, but in relational database management systems (RDBMS). The structure of the data does not focus on application-specific requirements, but represents an organization-wide data model that contains all the data relevant to the organization and the relationships between this data. Applications must be based on this data model and/or extend it in a way compliant with the model.

Innovator provides an MDA transformation mechanism which resolves the dichotomy of application perspective and data management perspective by mapping the structures of a data model and an object model. The relationships between the elements of the data model and the object model are stored in model references. The user can define the direction of the transformation and the assignment method (automatically or interactively).

The stored references enable bidirectional navigation between the models at any time.

Data-Function Transformation

When creating or maintaining function-oriented applications, you must account for the specifications of a company-wide data model if this model is to be accessed. Data structures in Innovator Data classiX are made available for the corresponding application in Innovator Function by means of external views.

These data structures (external views) are transformed to the data dictionary of the appropriate Innovator Function model by means of generative mapping. The transformation can be bidirectional.

In order to ensure consistency with the data model, the generated data dictionary entries in the function model cannot be modified. However, they can be used just like normal data dictionary entries for modeling.

The stored references enable bidirectional navigation between the models at any time.

User-Defined Transformations

In addition to the methodical transformation mechanisms, Innovator supports any type of link of model elements from any model. Copying and a special Paste command in the Model Browser enable the assignment.

The stored external references enable bidirectional navigation between the models at any time.

Analysis-Design Transitions for Innovator classiX

Object

The analysis-design transition enables automatic generation of design classes from analysis classes. This process generates identical copies and creates the declarations for the programming language selected. In addition, an existing design model is aligned and enriched with the new information. The following functionalities are provided:

- Alignment of relationships
- Creation of a copy of class diagrams
- Rules for the implementation names of classes, attributes and methods
- Generation of the package hierarchy as in the analysis

Function

The analysis-design transition enables the automatic generation of an operation design from a hierarchy of data flow diagrams. An operation is created for each process and a call hierarchy derived from the process hierarchy. Several options are available for the design of terminators and containers (internal or external operations or logical components). In addition, an operation diagram is generated for each data flow diagram. The following functionalities are available:

- Design of containers as logical components, internal or external operations
- Design of terminators as internal or external operations
- Rules for the implementation names of operations and parameters
- Generation of a separate operation for the context diagram

Innovator and Integrated Development Environments

Due to the unique coverage of Innovator and the modeling functionalities from the business process model to a structured and object-oriented software model all the way to the conceptual and physical data model, an unprecedented integration at model and process levels can be reached together with the development environments provided by companies such as Microsoft, Eclipse and Borland.

Innovator integrates all areas of modeling from the business process to the software model to the maximum degree possible to achieve maximum efficiency and convergence.

Model Integration

EJB Applications for Innovator classiX

Enterprise JavaBean applications usually use business data contained in databases. Data integrity is key, redundancies must be prevented.

By using Innovator Data with a mapping process between a relational and an object-oriented model (Object-Data transformation), you can create a non-redundant data model for the persistent bean objects from the very outset.

While the data model serves as the basis for persistency, the bean class model focuses on the distribution of functionalities and their re-usability in a component architecture.

After the bean class model and the data model are created with Innovator, this information is extracted and exported to the correct directory structure in the form of Java classes, embedded in package structures, together with the default EJB deployment descriptor, the Borland-specific deployment descriptors and various XML files for mapping the beans to the database. The information is then available for further processing.

Data Modeling

The results of the data modeling are included in the deployment descriptors of the EJB models and can also be directly made available to the database.

GUI Integration

Integration of the Innovator Editions Organizations uses various tool families in the development process. While business and organization experts create the business and software models with the Innovator editions, the programmers implement the system with tools such as development environments.

In order to ensure maximum efficiency in the development process, the maximum amount of information from the business and software models must be transferred to the implementation environment.

Innovator View

A GUI integration is provided to support users who work with Innovator and the Microsoft, Eclipse or Borland tool families.

The implementation tool is extended by the "Innovator Model Browser" view which enables context-sensitive jumps to the appropriate elements in Innovator. This way, you can, for example, navigate from a bean class directly to the corresponding class diagram that contains the class.

Eclipse Plug-In

The Innovator plug-in for Eclipse provides a view that enables access to the elements of an Innovator model. This view shows the model structure and the model elements in the form of a tree. It is possible to activate the appropriate Innovator edition such as diagram or text editors via the context menu and switch to the Innovator environment.

The information required for implementation (classes) is available in the Eclipse development environment after an export from Innovator.

The plug-in supports Eclipse 2.1 and 3.

.NET Support

Code Generation for C#

C# code generation is template-supported and takes place using oAW templates.

.NET Programming Interface

A .NET programming interface (API) is available for access to Microsoft Visual Studio in the Innovator model.

Integration in Microsoft Visual Studio

An Innovator add-in provides the Innovator model view as an integrated component of the .NET development environment. Innovator can be accessed with a standard connection by starting Visual Studio or by activating the command.

The Innovator add-in contains a repository browser for selecting the Innovator models. You can log-in via the Innovator Object eXcellence models. A complete model tree is displayed in its own window for each model. The elements which can be generated from code (classes, interfaces, signals, lists, data types, properties and methods) are displayed.

For alignment with the source code, the elements can be marked in the Visual Studio in the following way:

- whether code was generated from it into a particular directory
- whether they were loaded into an open .NET project

The Innovator add-in enables jump operations, such as changing from the model element in the add-in to respective elements in Innovator or to code elements in the Visual Studio. The elements therefore access the respective code element in the Visual Studio via the model element in the add-in of the element in Innovator. This also works the other way round.

Other Interfaces

SCC Interface

Source Code Control (SCC) is a defined interface for the integration of version control systems via the Interface Common Source Code Control defined by Microsoft.

The integration is only available under Microsoft Windows since the interface is only defined for Windows. The following systems are integrated with Innovator via the SCC interface:

- ClearCase
- PVCS Version Manager

Other version control systems that support this standard interface can be integrated.

Integration with of Innovator classiX with PVCS-VM and PVCS-Dimensions

Instead of the integrated version management of Innovator, you may also use SERENA's PVCS Version Manager or PVCS-Dimensions to manage the Innovator artifacts.

This integration enables access to any version both via the Innovator GUI as well as the PVCS Version Manager or PVCS-Dimensions GUIs. The integration provides the following functionalities:

- Checking in/out of all lockable objects, object groups or complete projects
- Checking in/out via convenient dialog interfaces or in batch mode
- Representation of the Innovator objects directly in PVCS via the integration of the appropriate Innovator editors
- Support for all call options of the PVCS commands put and get
- Transparent control of the Innovator integration via engineering actions

XMI Interface

XML Metadata Interchange (XMI) is a text-based interchange format for metadata and data. The XMI export generates one or several XMI files with the corresponding model information from Innovator Object classiX (UML 1.4), Business classiX or Data classiX (CWM) according to the Document Type Definition (DTD) generated from the corresponding specification (UML 1.4 or CWM).

In the Innovator Object eXcellence edition, the export is based on XMI (UML 2.1).

XML Export for Microsoft Project from Innovator Business

The XML export from Innovator Business for Microsoft Project allows you to base project management in Microsoft Project on quality-assured Innovator process models. Innovator can export any process hierarchy of a business model as an XML file. This file is compliant with Microsoft's XML scheme for Microsoft Project.

The export considers activities as tasks or summary tasks, the attribute for the duration of the activity as estimated task duration. It also considers the required resources required as well as the created objects.

BPEL Export from Innovator Object eXcellence

With the "Business Process Execution Language for Web Services” (BPEL4WS), IBM and Microsoft developed a description language for Web Service processes. The OASIS organization further develops and standardizes this language (http://www.oasis-open.org/).

Innovator now provides the possibility to model the collaboration of web services and to transform the model to executable BPEL code. After defining the basic web services you want to build upon, you can define more complex web services by orchestrating the basic ones. This is done using UML 2.1 activities. Innovator then transforms the BPEL-specific design model to several XML-files to be passed to a standards-conform BPEL engine, with which you can execute the orchestration processes.

Interface to ARIS Design Platform

Interoperability of Innovator is, among other things, converted with ARIS Mapping for data exchange between ARIS Design Platform and Innovator Object eXcellence. In this way, ARIS models can be mapped into UML 2.1 models in Innovator. “ARIS Mapping Solution” also makes the coverage of the Innovator modeling platform accessible for development teams who use the IDS Scheer AG ARIS Design Platform for technical modeling instead of Innovator Business.

Elements necessary for the eXcellence model are identified and adopted using an efficient mapping algorithm. Mapping searches for methodical samples of ARIS construction and maps the components of a sample which is found onto a corresponding UML model element.

Both initial mapping and delta mapping can be carried out for adopting modifications from the ARIS model. This enables both of the tools to work parallel to each other and ensures the synchronization of technical specification modifications with the IT model.

Innovator Web

Web Access to Innovator Models

Innovator Web allows for reading Web access to all Innovator models of all editions by means of standard Web browsers such as Microsoft Internet Explorer or Mozilla Firefox.

Text descriptions, graphics, diagrams and tables can be viewed and printed.

Requirements for Innovator Web:

Servlet Container	Servlet 2.3 and JSP V1.2
Java	JDK 1.4 or higher

In addition, platform and client licenses for Innovator Web are required.

Navigation

Hyperlinks are used for navigation in the model. In the graphics, navigation in model elements is possible by selecting elements.

Bookmarks are supported (setting, jumping).

User Interface

The edition responds to the corresponding browser configuration (English or German menus, etc.).

The Innovator Web GUI is customizable (e.g. to the corporate design).

Product Documentation

Online Help

Menu items and dialogs of all Innovator editions are completely described in a comprehensive, clearly structured, context-sensitive online help.

When you select a menu and move the mouse pointer over the menu items, the status line displays the function of the corresponding item. If additional information is available on a menu, the status line text is preceded by a symbol. To display this description in a pop-up information field, it is sufficient to press the F1 key.

Dialogs provide similar help functions for the individual dialog elements. To display the pop-up information field, it is sufficient to move the mouse point over the desired element and press F1.

The standard access to the help functions via the Help (?) menu is also available on Microsoft platforms.

If additional information is available on an element, the pop-up information field contains a "more" link that opens a browser with the complete text as an HTML page.

Depending on the language settings defined in the setup, the online help is available in English or German language.

Tutorial

The tutorial should help you to get a general idea of the purpose and function range of Innovator 2008 and to learn at the same time the fundamentals of the handling of Innovator.

The tutorial (.PDF file) is available in English and German language.

User Manual

The user manual shows you how you proceed when working with Innovator. Contents or index help you to find desired information. The user manual classiX parts A to C describe the method-spreading functionalities of Innovator 2008 classiX (set up models, work with Innovator, model documentation). Parts D and G describe in each case one of the classiX editions of Innovator.

The eXcellence manuals are in progress. You can find out information about configuring eXcellence models in the configuration manual and the solutions paper about the MID Modeling Methodology.

Other manuals provide information about special topics, such as difference comparison results of models.

The manuals (.PDF file) are currently solely available in German language.

Administrator Manual

The administrator manual is written for the Innovator administrator.

It describes the architecture, the administration, the installation and the start-up of Innovator. Thus, as administrator, you are put into the position to create the necessary and desired constellation with the installation and license administration from Innovator. In addition, the administrator manual informs you about the handling of repositories and models as well as about using own symbols in Innovator.

The administrator manual (.PDF file) is available in English and German language.

MID Modeling Methodology M³

Innovator contains documentation about the MID Modeling Methodology M³. It is available as PDF in the help menu of the model browser. A corresponding demo repository shows the development of an EJB3 banking application with M³, which uses Innovator Business, Object, Data und Report. The documentation and the demo repository are only available in German language.

Configuration Manual for Innovator eXcellence

The configuration manual describes the configuration editor in Innovator's eXcellence edition and the principal configuration of eXcellence models via UML 2.1 profile.

Migration Manual

The migration manual contains a description of the work steps necessary for migrating models from Innovator versions 9.x to Innovator 2008. It is written for customers and users of Innovator (version 9.0) or Innovator 2007 (version 9.1) who wish to upgrade or are considering an upgrade to Innovator 2008.

The document provides you with information about transforming and editing models in Innovator 2008.

The migration manual is solely available in German language.

API Help

The API functionality is documented by an API help which explains the relevant parts of the Innovator-internal metamodel as well as the appropriate read and write functionalities.

This help is solely available in English.

More Information: www.mid.de

Headquarters

MID GmbH
Eibacher Hauptstrasse 141
90451 Nuremberg
Germany

Tel.: +49 (0)911 96836-0
Fax: +49 (0)911 96836-10

E-Mail: info@mid.de

Cologne Branch

Ettore-Bugatti-Strasse 6-14
51149 Cologne
Germany

Tel.: +49 (0)2203 8901048
Fax: +49 (0)2203 8901401

Stuttgart Branch

Silberburgstrasse 187
70178 Stuttgart
Germany

Tel.: +49 (0)711 633859-0
Fax: +49 (0)711 633859-10

Munich Branch

Keltenring 7
82041 Oberhaching
Germany

Tel.: +49 (0)89 95476831-0
Fax: +49 (0)89 95476831-9