

# Modelarea sistemelor informatice de e-Commerce



# De ce modelăm?

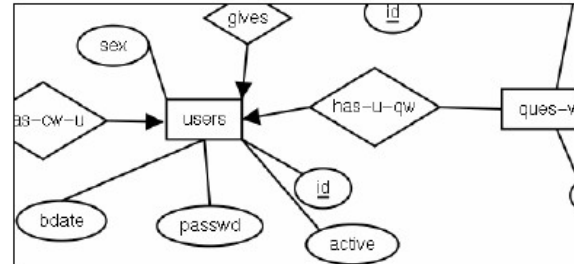
- Aplicațiile software pot fi complexe; cum le putem gestiona eficient?  
Posibilă soluție: folosim o bună reprezentare a sistemului
- **Model**: reprezintă anumite proprietăți ale unui obiect într-un anumit context
  - Reducerea complexității prin ascunderea detaliilor ce nu sunt necesare (abstractizare)
  - Anumite proprietăți ale sistemului pot deveni mai vizibile (claritate)
  - Facilitează aplicarea unei **metodologii**
  - În general; modelele oferă o mai bună **utilizabilitate**
- De obicei se folosesc mai multe modele în paralel; modele diferite pentru diverse scopuri

# Modele

- **Modele de date**

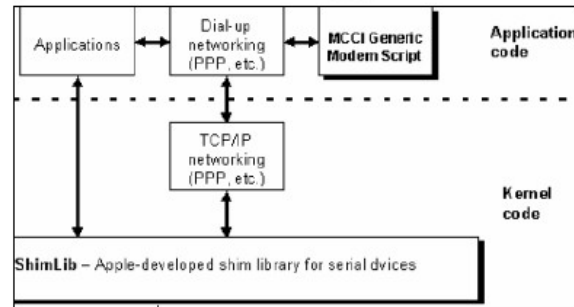
Tipuri de date și relațiile între acestea

E.g. Diagrame ER, diagrame de clase



- **Modele arhitecturale**

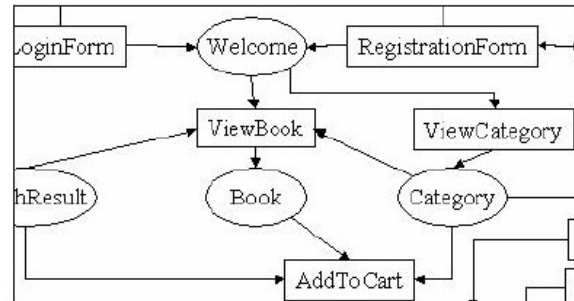
Componente ale unui sistem și relațiile între acestea



- **Modele pentru interfețe utilizator**

Structura UI (navigare, interacțiuni, ...)

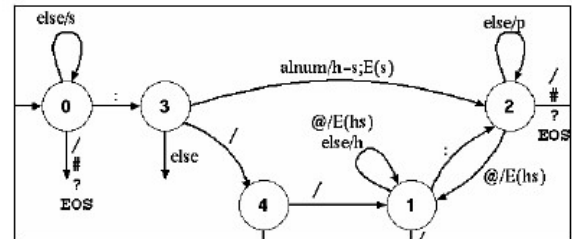
E.g. formcharts, diagrame ecran



# Mai multe modele

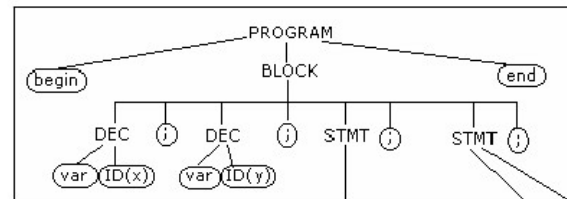
- Modele stare tranziție**

Stări ale sistemului și tranziții între acestea  
E.g. Mașini cu stare pentru modelarea jocurilor

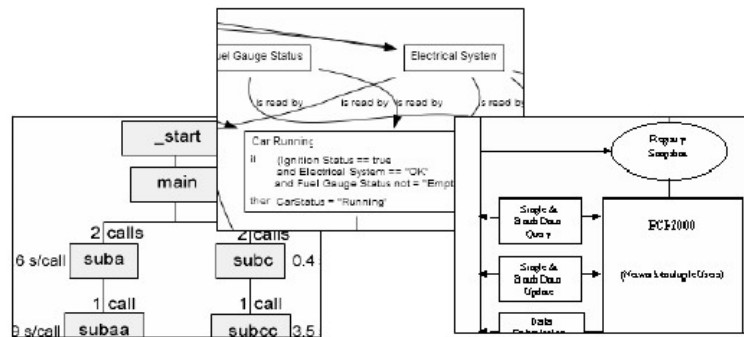


- Modele ale codului sursă**

Structura codului programului  
E.g. Arbori de sintaxă abstracți (AST)



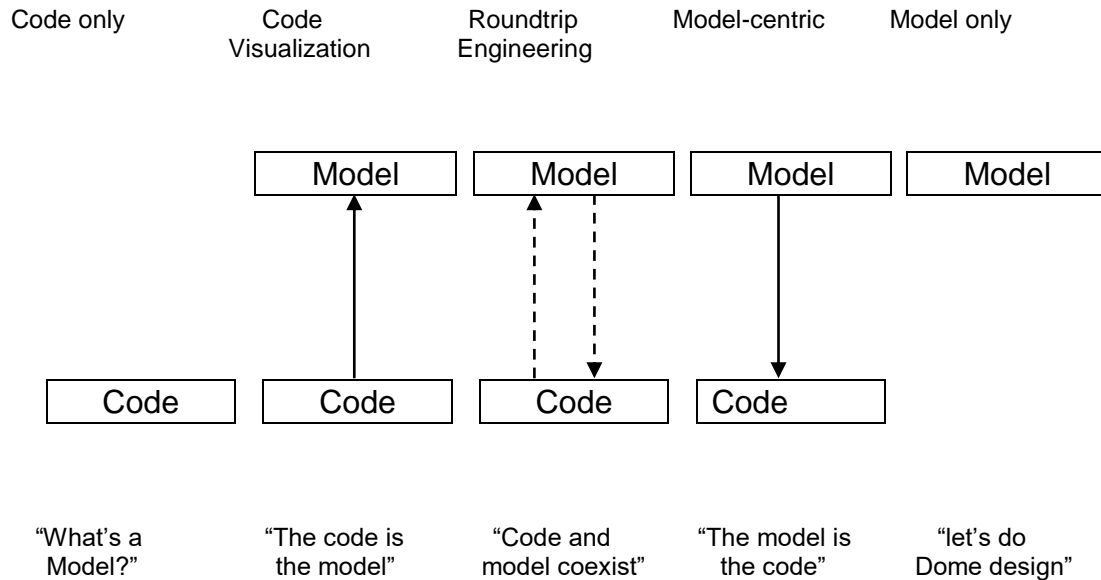
Grafuri de apeluri, grafuri de dependențe, diagrame de fluxuri de date & multe altele...



# Modelare orientată obiect

- Punct de pornire: descrierea informală a cerințelor
- Analiza OO:
  - Modelează invarianti specifici domeniului unui sistem
  - Părți stabile ce descriu concepte stabile
  - Ex: într-o universitate sunt întotdeauna studenți, cursuri, profesori...
- Proiectarea OO:
  - Rafinează modelul de analiză cu părți dependente de implementare
  - Ex: persistență, distribuție, folosirea anumitor tehnologii sau componente

# Moduri de sincronizare modele-cod sursă aplicate



# Evoluția sistemelor (1)

- **Abordare code-only**
  - sistemele sunt scrise direct într-un limbaj de programare
- **Abordarea code visualization**
  - după analiza problemei, codul e reprezentat grafic și modificat
- **Dezvoltarea round-trip engineering – RTE**
  - Separarea modelelor de codul sursă
- **Abordarea centrată pe modele (model centric approach)**
  - modelele sistemului informatic sunt suficient de detaliate pentru a permite implementarea completă a acestuia
- **Abordarea model-only**
  - modelele sunt folosite strict la înțelegerea și reprezentarea domeniului de studiat, a proceselor de afaceri, a analizei arhitecturii sistemului, etc.



## Evoluția sistemelor (2)

- Abordarea MDA

- cea mai recentă abordare
- asemănată mai mult cu o abordare în care codul este parțial sau complet generat pe baza mai multor modele, obținute prin aplicarea de diferite limbaje standard de modelare
- acesta este modul de dezvoltare de sisteme informatice care poate răspunde cu acuratețe cerințelor beneficiarilor, oferind mai multă flexibilitate impusă de evoluția acestora în timp
- abordarea MDA permite dezvoltarea unui sistem IT pe baza standardelor existente deja, furnizând un cadru de interoperabilitate și interconectare a sistemelor software diferite.

# Practica modelării sistemelor informatice

- **Modelele** reprezintă o abstractizare aproximativă a elementelor reale ce urmează a fi realizate.
- Mai puțin la început pentru că aplicațiile software erau mai simple, ușor de realizat, dar mai ales pentru că puteau fi modificate fără costuri suplimentare semnificative
  - exista tiparul **imaginează-construiește-modifică**
- Astăzi sistemele informatice sunt din ce în ce mai complexe
  - se impune integrarea cu aplicații software deja existente
  - sunt utilizate permanent
  - se impune adaptarea lor la noile condiții reale.

# Ce modele sunt importante?

- Business Model
  - Vizualizarea proceselor de business
- System Architecture Model
  - Vizualizarea cerințelor de sistem, structurii și comportamentului
- Use Case Model
  - Vizualizarea cerințelor funcționale
- Analysis Model
  - “Ce” trebuie să realizeze sistemul pentru a realiza cerințele funcționale impuse
- User Experience Model
  - Vizualizarea interacțiunii utilizatorului cu sistemul
- Design Model
  - “Cum” realizează sistemul cerințele funcționale
- Data Model
  - Vizualizarea stocării persistente
- Implementation Model
  - Vizualizarea codului

# Abordarea IBM

- Beneficiarii produselor software așteaptă livrarea la timp și de calitate a unui produs care să răspundă cerințelor lor.
- Specialiștii firmei IBM au formulat **patru caracteristici** pe care trebuie să le îndeplinească în procesul de dezvoltare software, și anume:
  - dezvoltare iterativă;
  - concentrare pe arhitectura sistemului informatic;
  - asigurarea continuă a calității impuse de beneficiari și gestionarea schimbărilor și rezultatelor obținute, cât și a complexității sistemului IT;
  - înțelegerea procesului de proiectare și a riscurilor asociate.

# Abordarea IBM

- Prin modelarea sistemelor informatice, dezvoltatorii software pot:
- crea modelele sistemului înainte angajării de resurse adiționale;
- proiecta aplicații software pornind de la cerințe, dând siguranță și încredere în calitatea sistemului obținut;
- aplica dezvoltarea iterativă în care modelele sau alte artefacte obținute au un nivel ridicat de abstractizare, permițând modificări rapide și oricât de frecvente sunt necesare, funcție de modificările mediului de lucru.

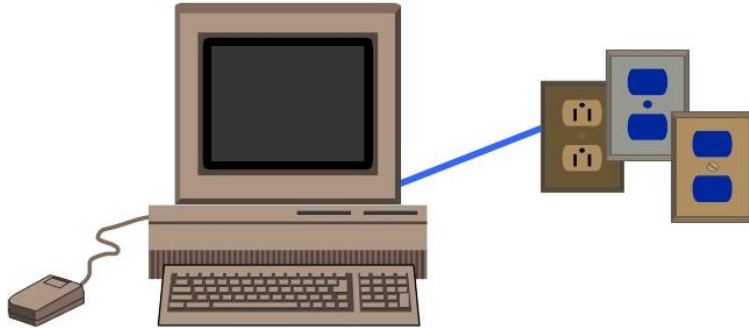
# Reticența la modelare

- Nu toți dezvoltatorii software înțeleg necesitatea modelării software
  - Adesea sistemele informatice sunt simple, ușor de înțeles și abia apoi devin din ce în ce mai complexe în mod natural.
  - În unele cazuri ei nu apelează la modelare pentru că pur și simplu nu percep necesitatea acesteia decât mult mai târziu.
- Unii specialiști argumentează rezistența la modelarea software ca fiind o trăsătură de cultură, sau, pentru că aceasta presupune instrumente suplimentare, pregătire suplimentară, anumite costuri adiționale, o durată mai mare de timp necesar de alocat și eforturi suplimentare.

## Avantajele modelării

- înțelegerea mai bună a problemei de rezolvat
  - realizarea unui sistem informatic de calitate
  - proiectarea și construirea arhitecturii sistemului
  - crearea unei vizualizări a codului sursă sau a altor forme de implementare a acestuia.
- 
- Totuși modelarea nu reprezintă ”totul sau nimic”, ea reprezintă doar o parte din procesul de dezvoltare software.

# Viziunea OMG



The Global Information Appliance



# Consensul modelării

- Eterogenitatea este pretutindeni
  - Nu există consens asupra platformelor hardware
  - Nu există consens asupra sistemelor de operare
  - Nu există consens asupra protocoalelor de rețea
  - Nu există consens asupra limbajelor de programare
- Totuși trebuie să existe concens asupra modelelor, interfețelor și interoperabilității!



# Model Driven Architecture

- Inițiativa Model Driven Architecture (MDA™) a grupului OMG urmărește **integrarea** bunurilor deja existente în software-ul dezvoltat
- MDA ajută la integrarea mixului de soluții deja existente și oferă o arhitectură pentru a suporta orice schimbări neașteptate viitoare
- Focusată pe integrarea aplicațiilor de tip “legacy”
- Asigură integrarea facilă a aplicațiilor COTS
- Modelele sunt testabile și simulabile

# Ce este MDA?

- Procesul de dezvoltare software nu mai seamănă nici pe departe cu procesul clasic “waterfall”, ci constă într-o permanență extindere și rafinare a unei soluții parțiale deja existente, căreia, după un număr de iterații, i se adaugă plus-valoare din punct de vedere al beneficiarului
- Def: **O modalitatea de a specifica și construi sisteme**
  - Bazată pe modelarea folosind UML
  - Suportă între ciclul de dezvoltare: analiză, proiectarea, implementare, deployment, mentenanță, evoluție & integrarea cu sistemele viitoare
  - Interoperabilitate și Portabilitate
  - Folosește standarde deschise
  - Costuri inițiale scăzute și maximizarea ROI
- Se aplică direct în mixul pe care astăzi îl avem în față:
  - Programming language
  - Network
  - Operating system
  - Middleware

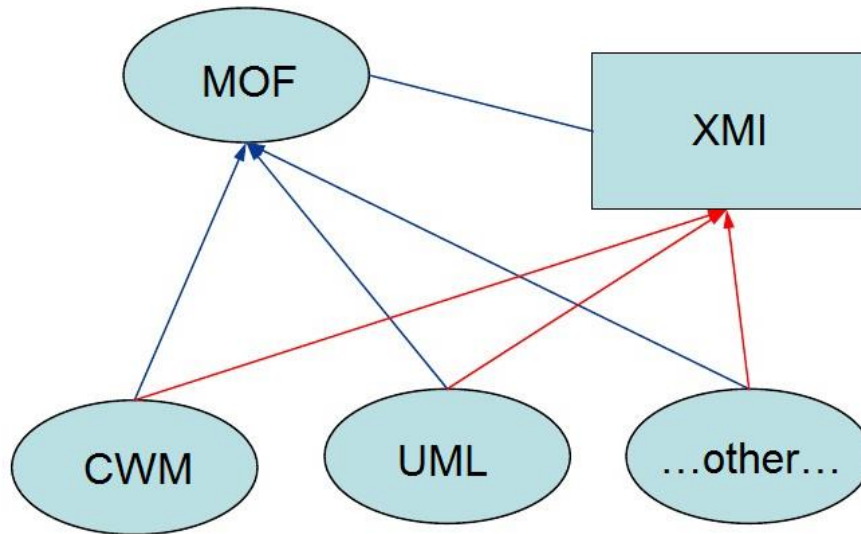
## Categorii de instrumente MDA

- Instrumentele MDA se împart în trei categorii:
- comerciale sau open source;
- parțial sau complet de implementare;
- generatoare de cod din modele sau executarea modelelor.

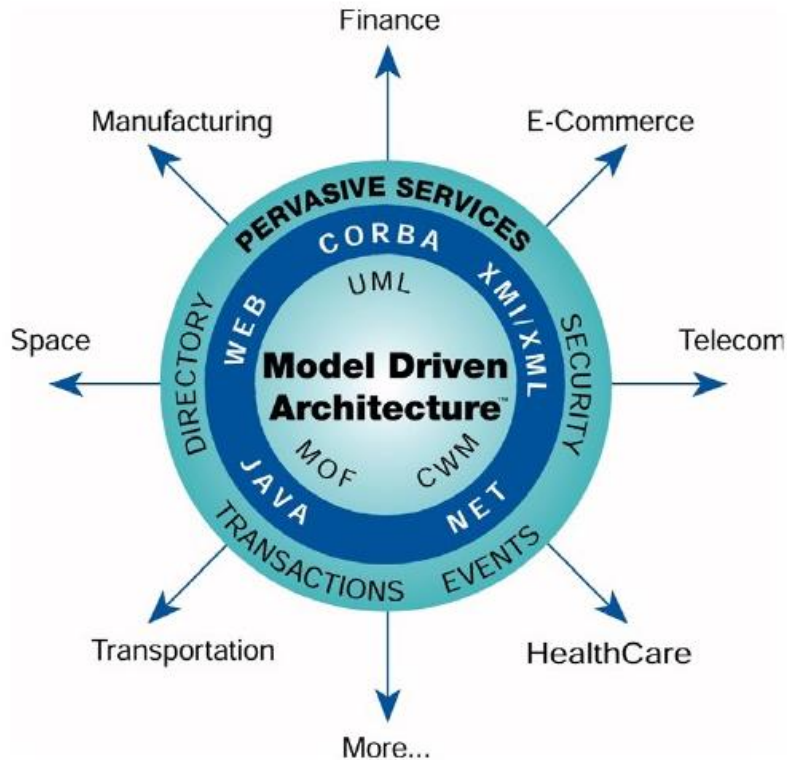
# Suita de modelare a OMG

- **Unified Modeling Language**
  - UML<sub>TM</sub> rămâne singurul limbaj standardizat de modelare orientat obiect, cea mai bine cunoscută parte a standardului
- **Common Warehouse Metamodel**
  - CWM<sub>TM</sub>, integrarea de inițiative data warehousing
- **Meta-Object Facility**
  - MOF<sub>TM</sub>, standard de integrare a metadatelor
  - Definește metadatele și serviciile de metadata
- **XML Metadata Interchange**
  - XMI<sub>TM</sub>, standardul XML-UML
  - Interoperabilitate a instrumentelor UML
  - Colecție de reguli de mapare XML/MOF

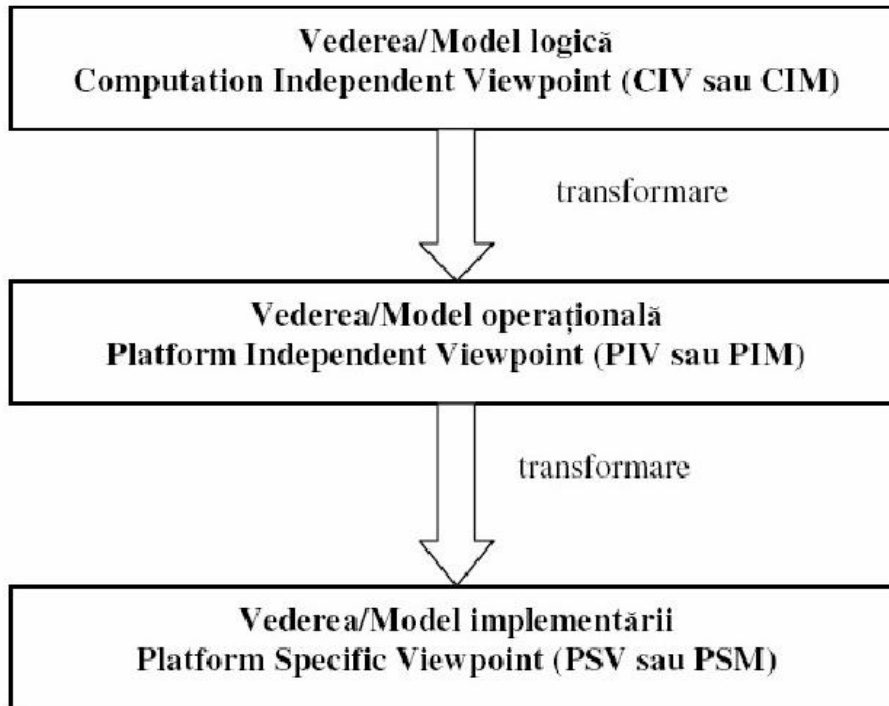
# OMG Modeling Suite



# Model Driven Architecture



### 3 vederi ale arhitecturii MDA





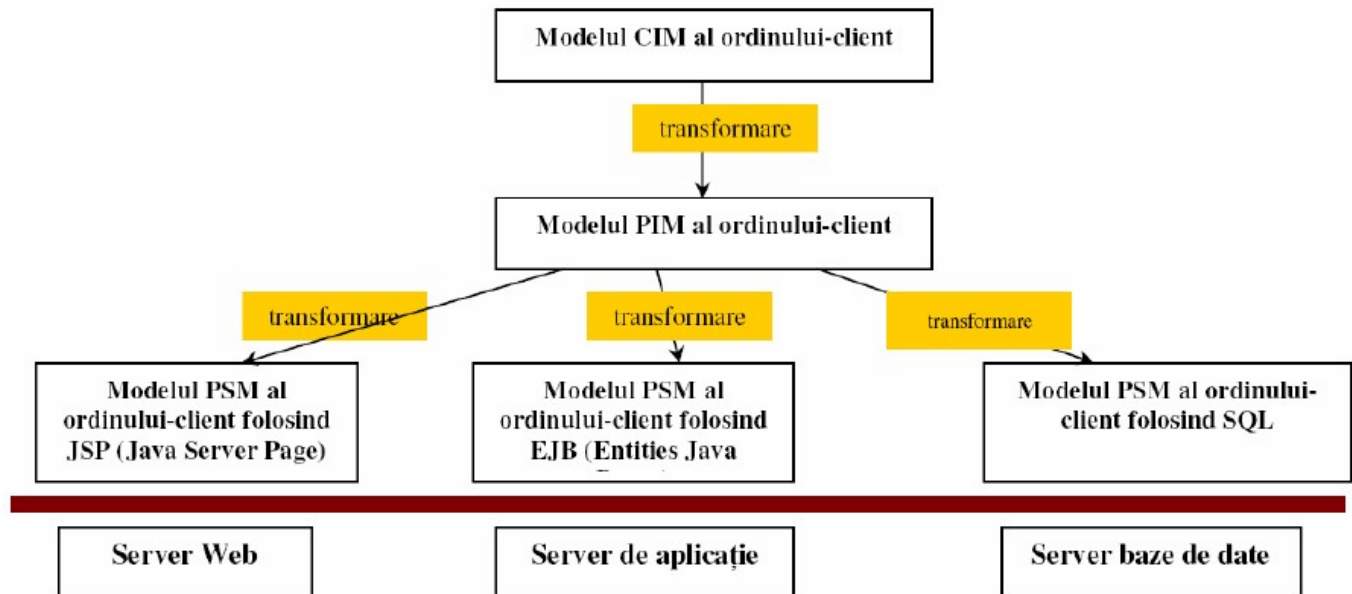
## Vederi MDA

- Computation Independent Viewpoint – CIV
  - separă modelarea logică a sistemului informatic de specificațiile de implementare;
- Platform Independent Viewpoint – PIV
  - se focalizează pe modelarea operațională a sistemului, dar fără detalierea specificațiilor de implementare;
- Platform Specific Viewpoint – PSV
  - conține specificațiile de implementare (platforma hardware, platforma software, produse middleware, tehnologii IT folosite).

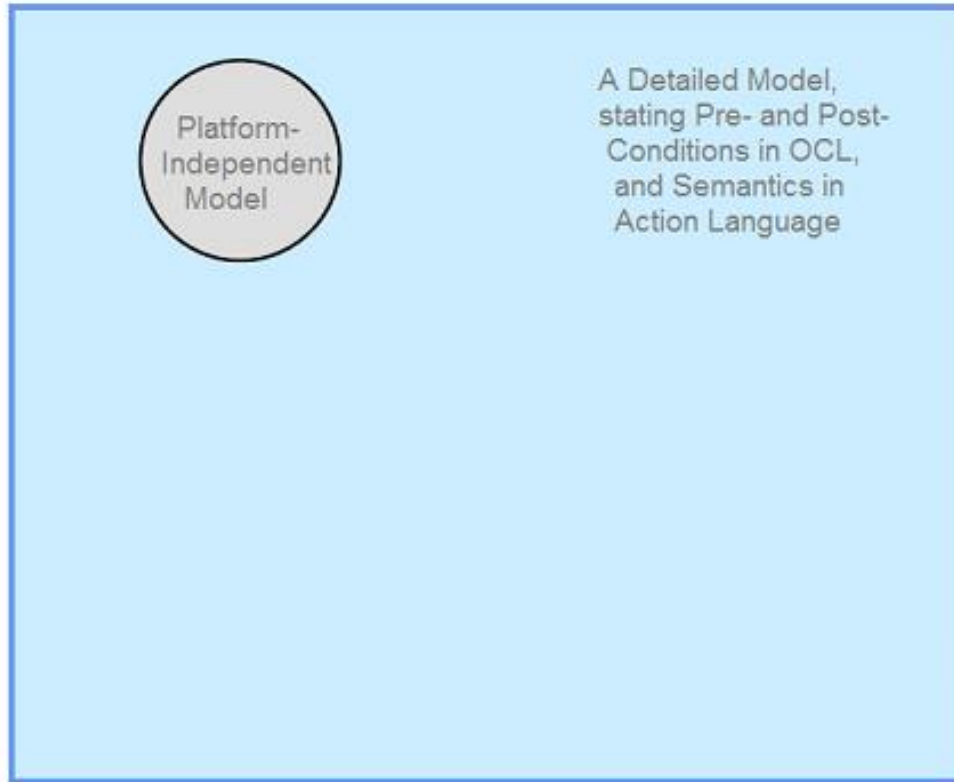
## Exemplu (1)

- integrarea în sistemul informatic al unei firme a ordinelor de la clienți transmise prin Interne
  - vederea CIM va consta din diagrame UML la nivel conceptual prin care se arată ce va face sistemul informatic
  - vederea PIM descrie funcțiile și structura acestuia prin diagrame UML detaliate
  - mai multe vederi PSV

## Exemplu (2)

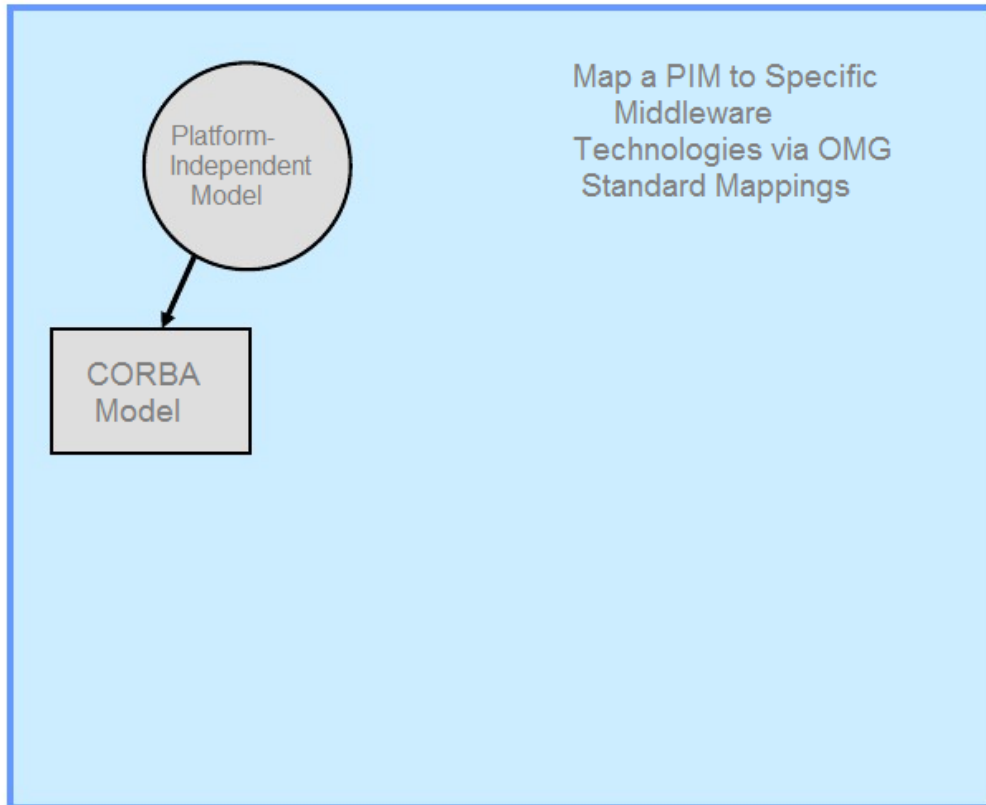


# Construirea unei aplicații MDA



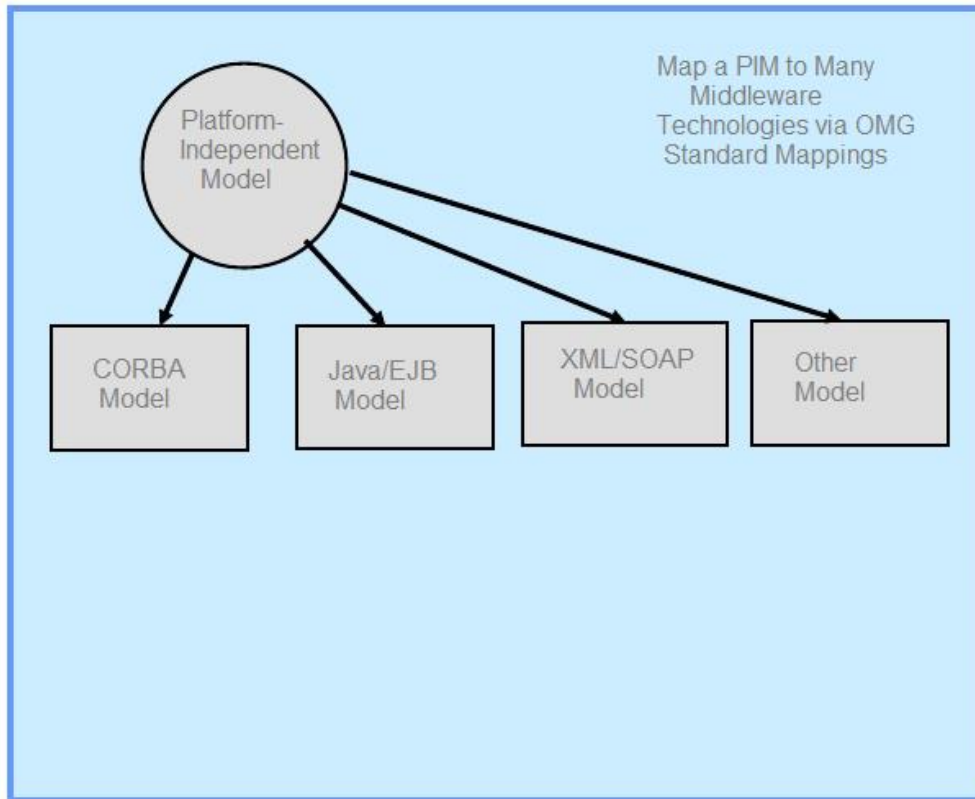
Start with a Platform-Independent Model (PIM) representing business functionality and behavior, undistorted by technology details.

# Generarea modelului specific platformei



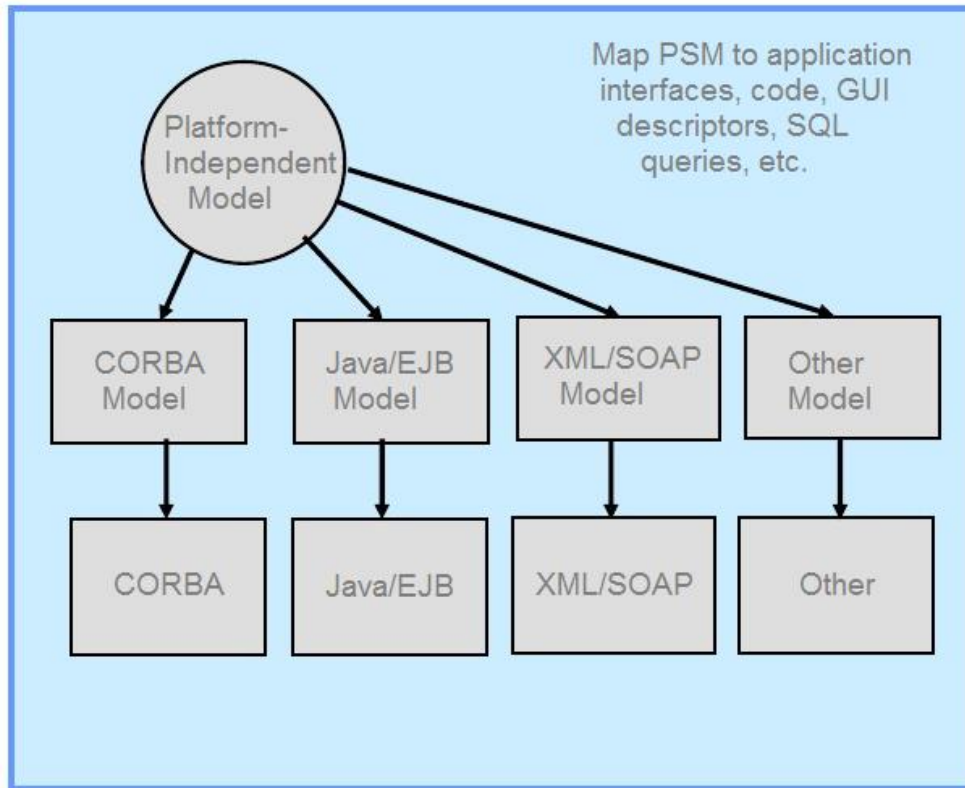
MDA tool applies a standard mapping to generate Platform-Specific Model (PSM) from the PIM. Code is partially automatic, partially hand-written.

# Maparea pe multiple tehnologii de deployment



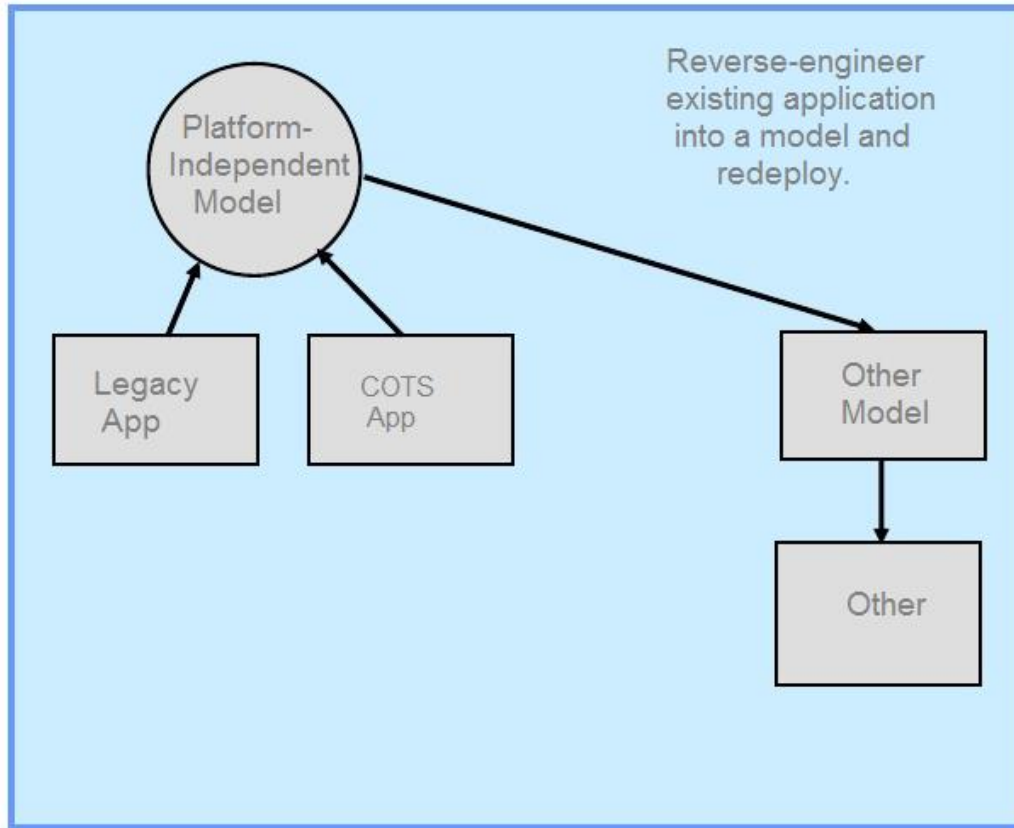
MDA tool applies an standard mapping to generate Platform-Specific Model (PSM) from the PIM. Code is partially automatic, partially hand-written.

# Generarea implementărilor



MDA Tool generates all or most of the implementation code for deployment technology selected by the developer.

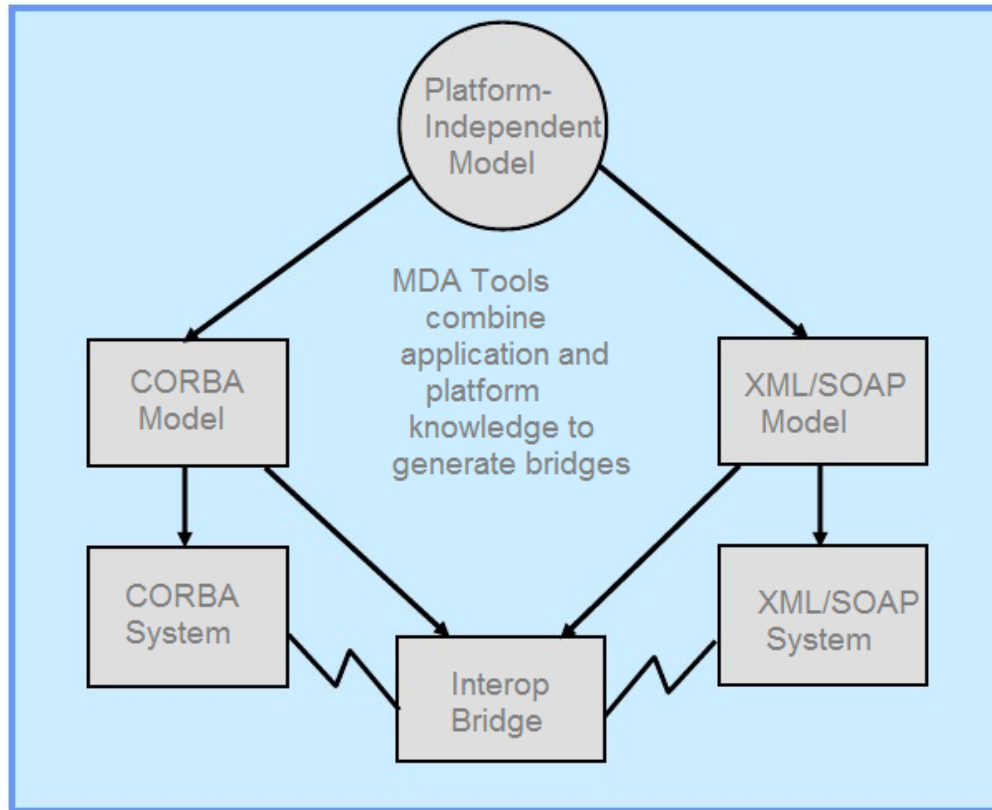
# Integrarea Legacy & COTS



MDA Tools for reverse engineering automate discovery of models for re-integration on new platforms.



# Automating Bridges



Bridge generation is simplified by common application models, simplifying creation of integrated applications both within and across enterprises.

# MDA în Standarde Industriale

- MDA promovează standarde ce sunt funcționale independent de tehnologie
  - Aplicabil pentru deploymenturi mari & mici, noi aplicații, legacy și COTS
  - Aplicabil pentru CORBA, DCOM, .Net, etc.
- MDA a fost adoptat de grupurile de standardizare OMG
- Grupuri din marketing văd valoarea adusă de abordarea MDA:
  - Legacy Transformation
  - Financial Services
  - Healthcare
  - Life Sciences Research
  - Manufacturing
  - Space & Ground Systems
  - Telecommunications

# MDA în Practică

- Several excellent proofs-of-concept:
  - Wells Fargo (an architecture that has already been resilient through a decade of change)
  - Lockheed Martin Aeronautics
  - GCPR in US government
  - Deutsche Bank Bauspar
  - Defense Information Systems
  - Merrill Lynch
  - Österreichische Bundesbahn
  - Thales Training & Simulation
  - Zuercher Kantonal Bank
  - CGI
  - Chubb and Son

## Mai multe informații

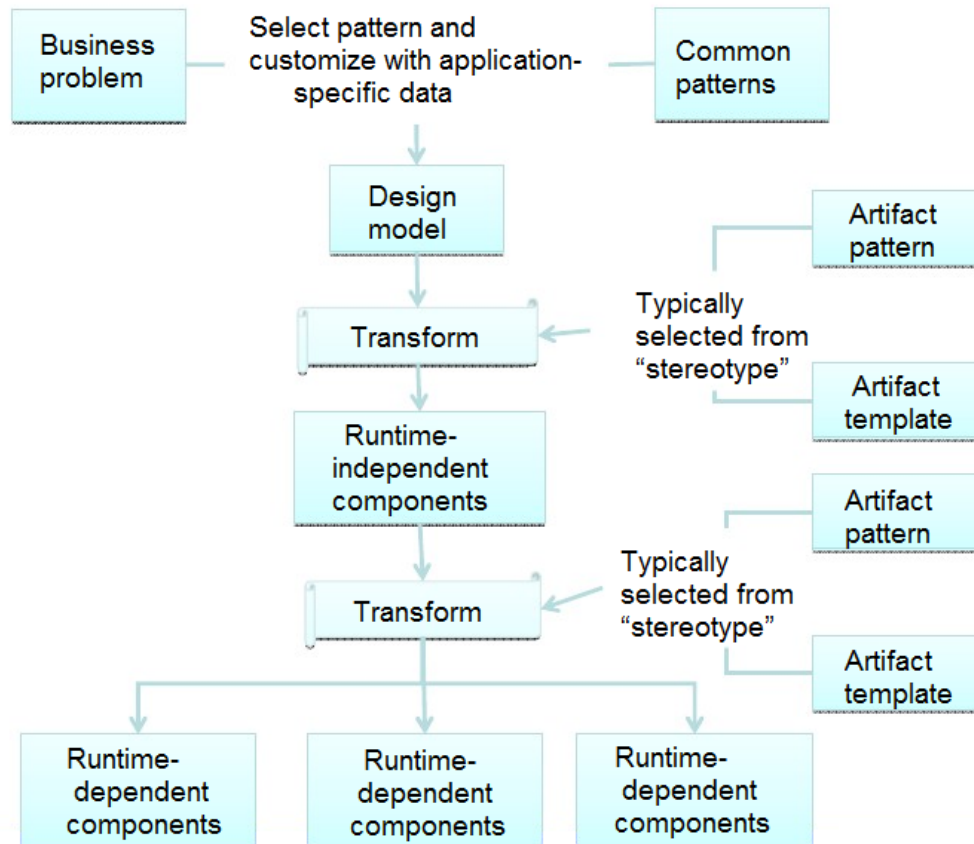
- MDA Information Page
  - <http://www.omg.org/mda/>
- OMG General Information
  - <http://www.omg.org/>

<http://www.omg.org/~soley/oois03.ppt>

# MDD

- **Model Driven Development** (MDD) = dezvoltare dirijată prin modele
  - metodă de proiectare a sistemelor informatice aplicabilă în cadrul metodologiei MDA.
- Se construiește un ansamblu de modele ale sistemului de analizat cât și ale noului sistem, pe baza cărora se generează alte modele sau codul sursă al sistemului.
- Totul se centrează pe transformarea modelelor sistemului de realizat și generare de cod sursă.
- Această metodă necesită un mediu integrat de dezvoltare (IDE) care să suporte:
  - limbajul UML, șabloane, transformarea modelelor UML și generare de cod sursă.
- **Rational Software Architect** (RSA) este un astfel de instrument integrat de proiectare și dezvoltare de sisteme informatice.

# Transformarea problemă reală – soluție IT aplicând metoda MDD



# Șabloane

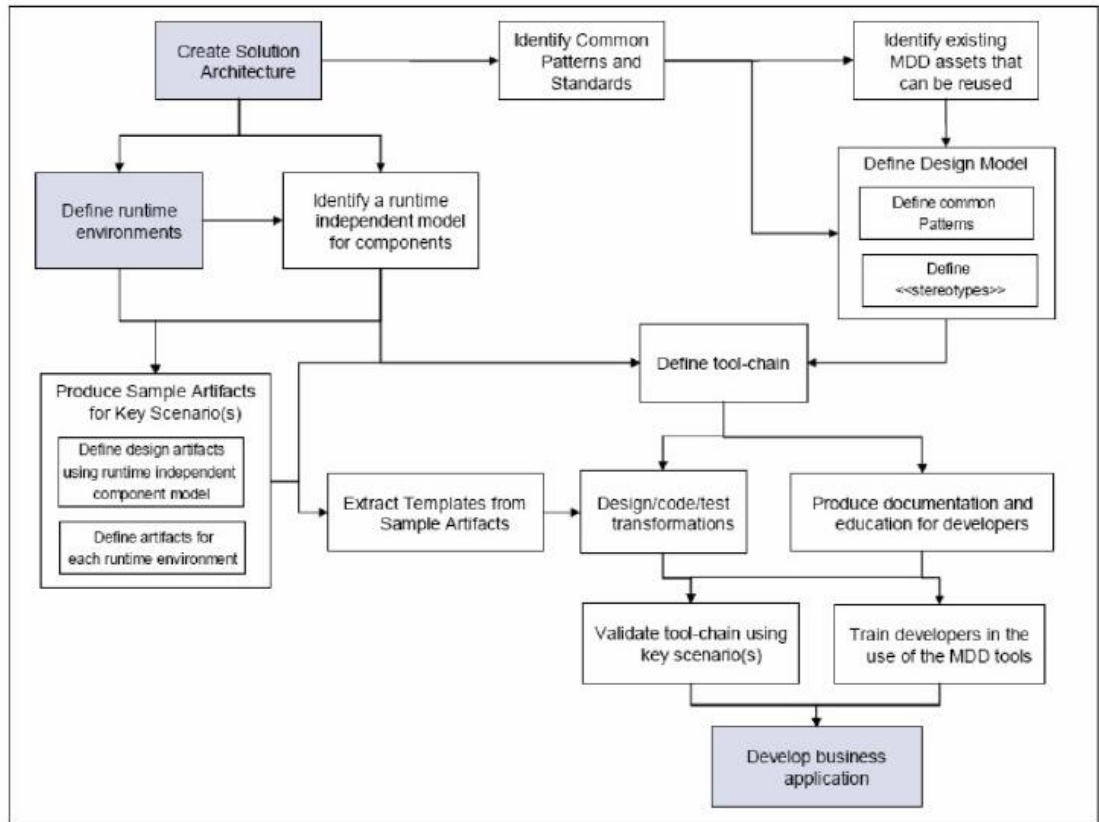
- Șabloanele sunt modalități de descriere a motivațiilor, a ceea ce se va întâmpla și de ce.
- Motivele studierii și aplicării acestora sunt:
  - reutilizarea,
  - stabilirea unei terminologii comune ce îmbunătățește comunicarea echipei software,
  - trecerea la o perspectivă de nivel înalt asupra problemei de rezolvat,
  - îmbunătățirea posibilității de modificare a codului sursă,
  - facilitarea adoptării unor alternative de proiectare mai bune,
  - ofera alternative la ierarhiile laborioase de clase pentru probleme complexe cât și posibilitatea unei proiectări mai bune și nu numai a unei funcționale.

# Șabloane

- Integrarea șabloane-MDD
  - crește productivitatea procesului de dezvoltare de aplicații soft
  - se îmbunătățește calitatea sistemului livrat și întreținerea acestuia,
  - reutilizarea componentelor sistemului informatic,
  - reducerea costurilor și a riscurilor unui proiect de investiții,
  - crește flexibilitatea sistemului dezvoltat,
  - se îmbunătățește comunicarea în cadrul echipei de dezvoltare cât și dintre aceasta și beneficiarii sistemului informatic.
- Metoda MDD se fundamentează pe abstractizare
  - permite realizarea vederii conceptuale a sistemului informatic, centrarea pe funcțiunile sistemului fără specificații de implementare.
- Este recunoscut faptul că este mai rapidă crearea modelelor unui sistem informatic decât scrierea de cod-sursă, motiv pentru care se agreează această metodă care permite lucrul la un nivel ridicat de abstractizare, permite includerea specificațiilor de proiectare în modele, urmat de generarea codului sursă prin aplicarea de transformări automate între modele.



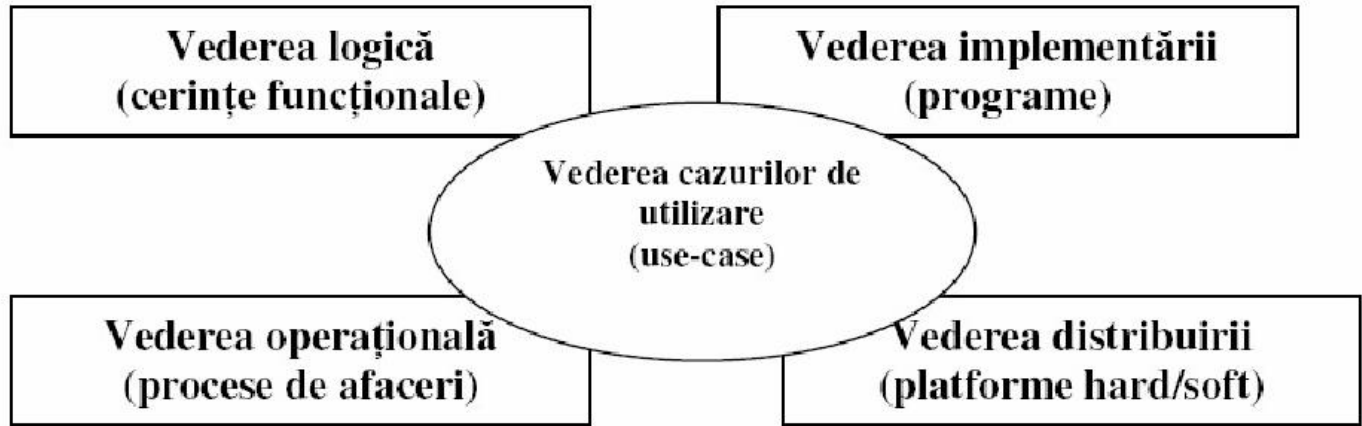
# Etapele de realizare a sistemelor informatice în cadrul metodei MDD



# Etape MDD

- Primele etape sunt de creare a arhitecturii sistemului și de definire a mediului de implementare (platforme soft/hard, tehnologii IT).
- Pe baza arhitecturii sistemului informatic sunt identificate șabloanele de analiză ce se pot aplica, cât și a modulelor de aplicații dezvoltate în alte proiecte MDD.
- Pe baza acestora se creează modelul de proiectare al sistemului și modelul de implementare pe componente, pe baza căruia se produc un număr de artefacte inițiale ce vor reprezenta scheme pentru următoarele transformări MDD.
- Se definesc instrumentele ce vor fi utilizate și o planificare a întregului proiect de dezvoltare a sistemului informatic.
- Sunt generate modele de programe (template-uri) pe baza artefactelor obținute anterior, care sunt testate, evaluate, sunt create module funcționale pentru care se furnizează și documentația aferentă.
- În acest moment se poate trece la dezvoltarea aplicațiilor, precedată de instruirea în prealabil a echipei de dezvoltare.

# Perspective arhitecturale MDD

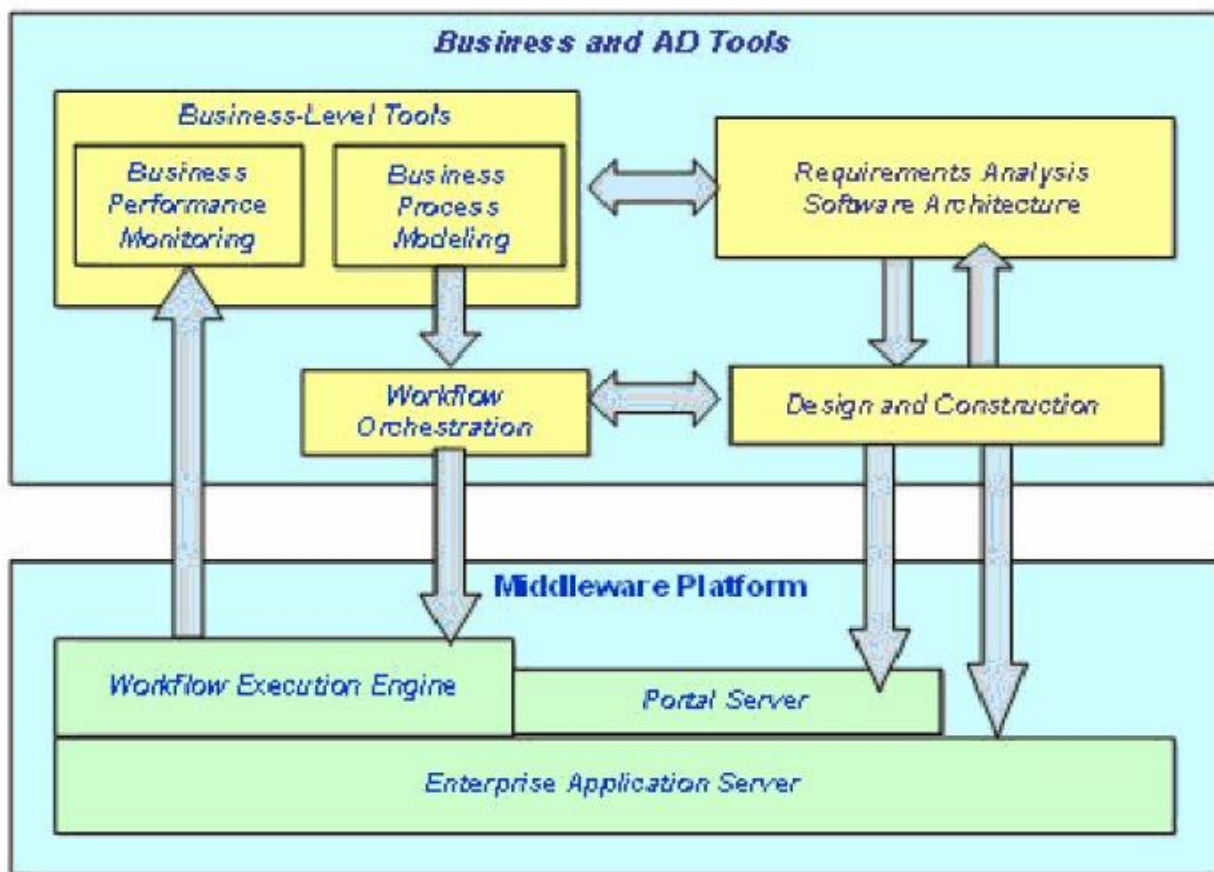


- vederea implementării - corespunde structurării programelor ce formează sistemul în componente
- vederea logică - descrie cerințele funcționale ale sistemului
- vederea distribuiri - definește aspectul spațial al sistemului (echipamente hardware, noduri de rețea)-
- vederea operațională sau a proceselor, ce corespunde structurii de exploatare a programelor și componentelor executabile.

# Analiza comparativă a procesului tradițional de dezvoltare a sistemelor informatice și cel bazat pe metoda MDD

Dezvoltarea tradițională a sistemelor informatice	Dezvoltarea sistemelor bazată pe metoda MDD
Centrat pe codificare și testare	Centrat pe analiză și proiectare, modelare
Codificarea manuală a programelor	Generarea automată a codului
Generarea manuală a documentației	Generarea automată a documentației
Testarea software-ului continuu	Validarea automată
Specificațiile sunt bazate pe hârtie	Prototipizare interactivă rapidă

## Noua viziune privind instrumentele de proiectare și realizare de soluții IT



# Componente în noua viziune

- **Platforma middleware** – reprezintă o platformă de execuție (runtime) formată din aplicații server robuste capabile să administreze soluții IT dezvoltate pentru platforme software eterogene sau soluții IT noi bazate pe standarde open;
- **Model de execuție** – interfețele platformei middleware definesc un model de programare ce trebuie înțeles de arhitectii IT. Orice utilizator al platformei middleware trebuie să aibă deja modelul conceptual al soluției IT și proiectarea fizică pe componente executabile;
- **Fluxurile de lucru** – reprezintă procesele de afaceri cât și relațiile dintre acestea și posibilități de configurare;
- **Instrumente de modelare a proceselor de afaceri** – includ instrumente de modelare a acestor procese cât și posibilitatea monitorizării acestora prin simularea impactului asupra evenimentelor declanșate în platforma middleware;
- **Arhitectura sistemului prin componente** – include instrumente care să permită gestionarea de aplicații moștenite, aplicații noi dezvoltate și asamblarea lor.

# Rational Software Architect

- octombrie 2004 - firma IBM a lansat platforma **IBM Software Development Platform** introducând o nouă generație de instrumente ce permit aplicarea metodei MDD.
  - Un loc central îl ocupă instrumentul integrat **Rational Software Architect** (RSA) ce suportă această metodă bazată pe limbajul UML.
- Instrument CASE ce se bazează pe metodologia MDA și oferă suport în dezvoltarea aplicațiilor Web statice sau dinamice ce pot rula pe platforma J2EE, include capabilități JSF, de utilizare a șabloanelor (depozitul RAS – Reusable Asset Specification).

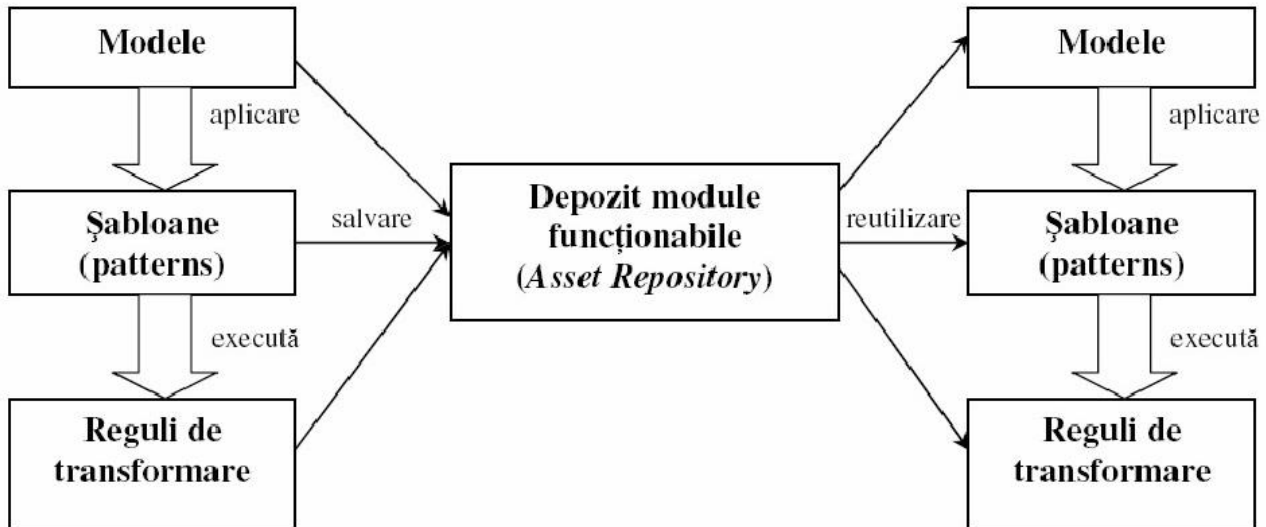
# Obiectivele RSA

- separarea proiectării de implementare
- reducerea timpului și costului de proiectare și dezvoltare a sistemelor informatice
- simplificarea și îmbunătățirea procesului de testare
- integrarea activităților de proiectare și dezvoltare
- standardizarea acestui proces
- creșterea calității sistemului realizat
- realizarea automată a unei documentații de calitate
- reutilizarea modulelor aplicațiilor și a documentației
- specificarea corectă și completă a cerințelor sistemului
- Simplitate
- flexibilitate.



# Componentele mediului RSA

- La arhitectura mediului integrat RSA se adaugă componente specifice aplicării șabloanelor



# IBM Rational Software Architect

An integrated platform for innovation and collaboration

Best of breed, comprehensive modeling tools that facilitate communication and collaboration

With the power of abstraction, automation and simplification

DoDAF

## UML Profile-based Integrated Architecture

## Profile for Software Services

## Leveraging Jazz platform integrated with Rational Team Concert

Exploit the latest in modeling language technology  
and leverage an open and extensible modeling  
platform

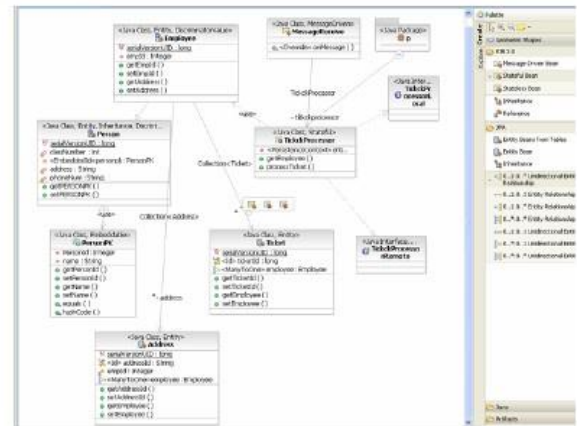
Simplify and unify Java and C++ design and development by integrating with other facets of the lifecycle such as:

Rational Data Architect, Rational Requisite Pro

Rational Asset Manager, Rational Team Concert

Rational ClearCase, Rational ClearQuest

## Telelogic Synergy and Change



## What's New: Rational Software Architect

Custom modeling environment for your business

Modeling with Domain Specific Language (DSL)

Work with reduced subset of UML

Rational Deployment Architecture Platform

Rich tools for deployment architecture definition

Verification tools for deployment architecture

Enhanced transformations and visualizations

Extensive Java and C++ support

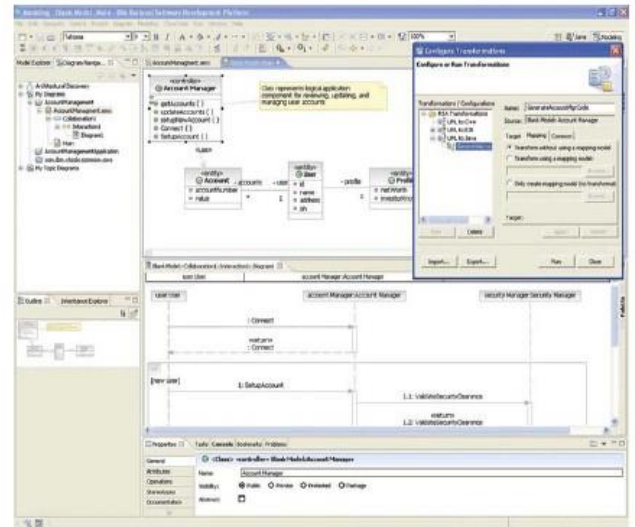
Increased visibility into existing source code

Integrations

Rational Asset Manager

Rational Team Concert on Jazz

Telelogic Change and Synergy



# Getting Started: User Assistance

User Assistance model to enable users of all skill levels

Leverages Product Tours to assist with the discoverability of capabilities

Tutorial Gallery leverages tutorials as learning aids

“Watch and Learn”

“Play and Learn”

“Do and Learn”

Samples gallery provides completed assets for reference purposes

Showcase

Application

Technology

All user assistance can be launched from a Welcome perspective



## Modeling software with UML



### Architectural specification using UML

Learn more about Model Driven Development using the Unified Modeling Language.

## Developing the application code



### Application development

Developing Java, C++, J2EE Web, Web services, XML and Data applications.



### Pattern and anti-pattern detection

You can manually explore application architecture using browse diagrams and also perform automated discovery of patterns in the application architecture using rules.

## Best practices and process



### Iterative development process

Learn about the Rational Unified Process configuration for software architects.



## Key Feature: C/C++ Development Environment

The screenshot displays the Eclipse IDE interface for C/C++ development. The top menu bar includes File, Edit, Navigate, Search, Project, Diagram, Run, Window, and Help. The main workspace is divided into several panes:

- Left Pane (Project Explorer):** Shows a hierarchical tree view of the C/C++ project structure, including folders like 'HelloWorld' and 'HelloWorld.cpp'.
- Top Center Pane (Code Editor):** Displays the source code for 'HelloWorld.h' and 'HelloWorld.cpp'. The code is syntax-highlighted and includes comments like 'Welcome to the Eclipse CDT'.
- Right Pane (Outline):** Provides a summary of the code elements, including classes and functions.
- Bottom Left Pane (Diagram Navigator):** Shows a UML class diagram visualization of the C/C++ classes and structs. A context menu is open over the diagram, offering actions like 'Add Name', 'Add UML', 'Add Data', 'Navigate', 'Open', 'Open With...', 'File', 'Edit', 'Find/Replace...', 'Filter', 'Format', 'Update Data', 'Generate', 'Harvest', 'Properties...', and 'Show Properties View'.
- Bottom Right Pane (Properties):** Displays the properties of the selected element in the diagram.

Callouts highlight the following features:

- Perspective for C/C++ Development:** Points to the overall IDE layout.
- C/C++ project hierarchical tree view:** Points to the Project Explorer on the left.
- C/C++ editor with syntax highlighting, code completion, and advanced search:** Points to the Code Editor at the top center.
- UML class diagram visualization of C/C++ classes and structs:** Points to the Diagram Navigator at the bottom left.

## Key Feature: C/C++ Development Environment

### Editing and Navigation

C/C++ Syntax Highlighting, Outline View

C++ Class Browser (Hierarchy View)

C/C++ Search

C/C++ Content Assist

### Project Import

Automated assistance in setting up CDT for search and content assist.

### UML C/C++ Code Editor

### Debug

GDB Integrated

Extensible Debug Interface

### Build

Standard Make for projects with existing build infrastructure

### Managed Build

Automatic makefile generation

GNU tools supported out of box

Managed build is extensible, build tools can be plugged-in and build tools options selectable

Meets Internationalization and Accessibility requirements

### Extensibility

Provides extension points for managed build, debuggers, ...



## Key Feature: Modeling assistance

Simplify the capture of UML models during Analysis and Design

Make modeling more accessible to a broader audience

New custom views improve the editing experience

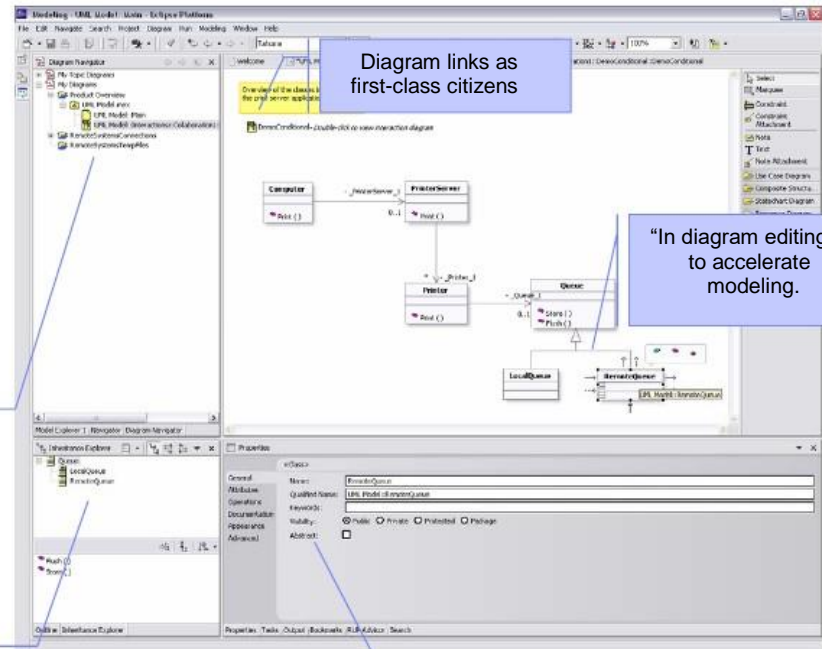
New "Diagram Navigator" view provides a diagram filtered view of the models and workspace

Inheritance view

Diagram links as first-class citizens

"In diagram editing" to accelerate modeling.

New properties view



## Key Feature: Patterns

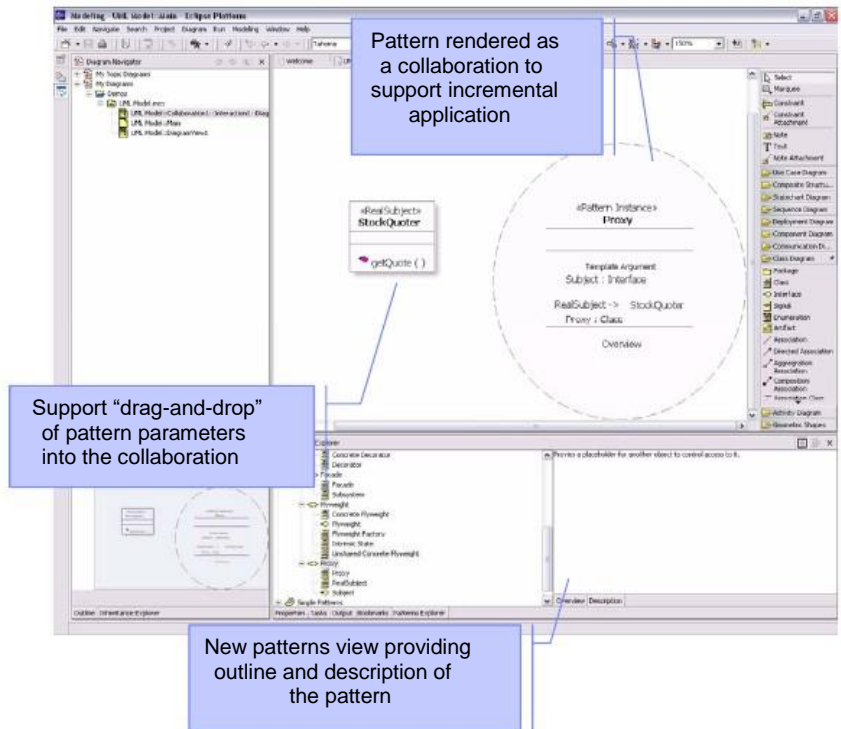
Applying Patterns is very simple

Evolution of pattern experience  
based on lessons learned

Pattern-authoring provides greater  
flexibility using Open API

All Gang of Four design patterns  
provided

Additional patterns provided via  
RAS repository on IBM  
developerWorks





## Key Feature: Transformations

Transformations are optimal for “batch” style computationally intensive operations

Model-to-model

Model-to-code

Code-to-model

Out-of-the box code transforms

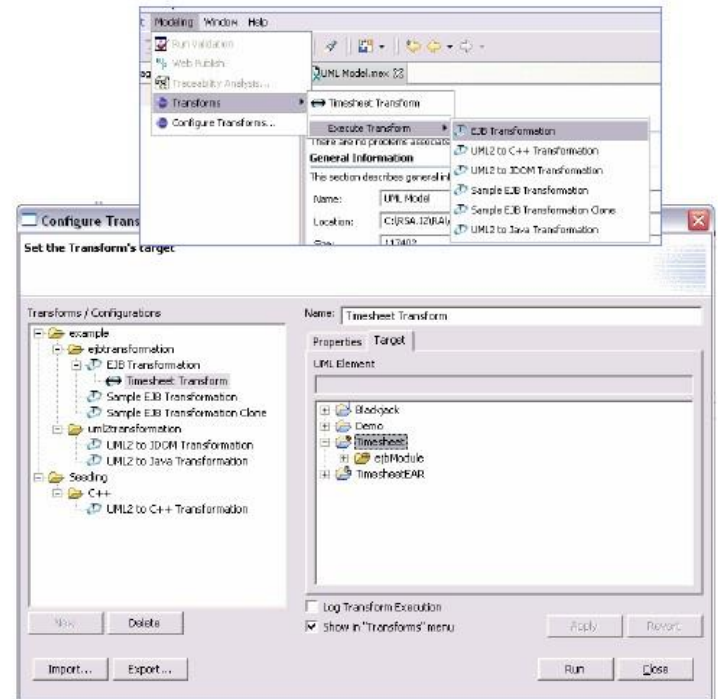
UML-to-Java/JSE

UML-to-C++

UML-to-CORBA IDL

Plus sample model-to-model transforms

Transformations may be updated via RAS repository hosted on IBM developerWorks





# Key Feature: Visualize method bodies

Facilitates understanding and application's behavior by providing visualization of detailed code

Diagrams can be integrated in Javadoc reports

The screenshot displays the IBM Rational software interface. On the left, a 'Package Explorer' shows a project structure with packages like 'com.ibm.rational' and 'com.ibm.rational.java'. The main editor shows Java code for a class named 'com.ibm.rational.java'. The code includes comments and method definitions. A callout box points to the 'Package Explorer' with the text: 'Integrated with the Java Package view'. Another callout box points to the 'UML' tab, which shows a sequence diagram for the 'com.ibm.rational.java' class. The diagram includes lifelines for 'self' and 'com.ibm.rational.java', and messages between them. A callout box points to the sequence diagram with the text: 'Leverages UML 2.0 sequence diagram constructs for loops, conditionals, etc...'. A third callout box points to the 'UML' tab with the text: 'Alternate abstract view of method behavior'. A fourth callout box points to the 'UML' tab with the text: 'Select method to be visualized using UML'. A fifth callout box points to the 'UML' tab with the text: 'Topic diagram for method is automatically updated/refreshed when method is updated'.

Integrated with the Java Package view

Leverages UML 2.0 sequence diagram constructs for loops, conditionals, etc...

Alternate abstract view of method behavior

Select method to be visualized using UML

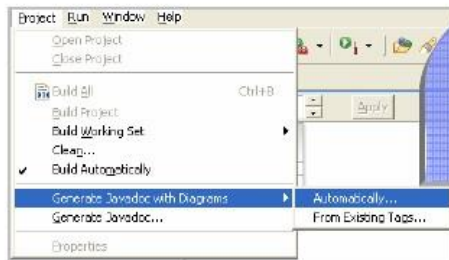
"Topic" diagram for method is automatically updated/refreshed when method is updated

# UML Enhancements: JavaDoc with Embedded UML Diagrams

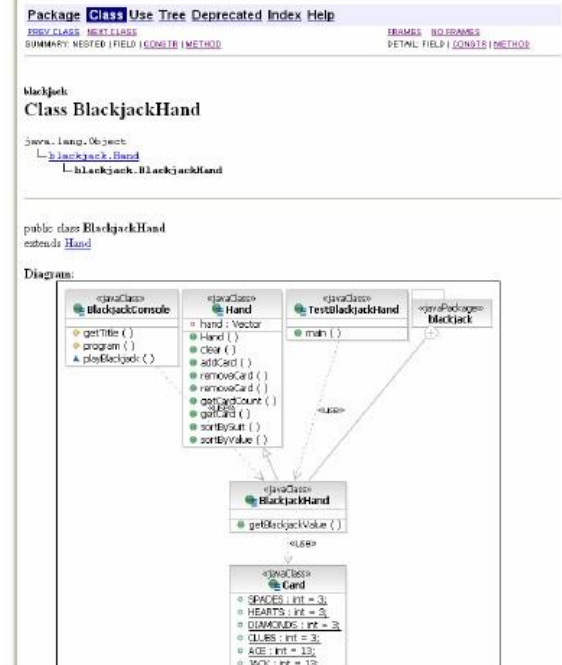
Produce enriched JavaDoc

UML diagrams right on the pages

Completely integrated with hyperlinks



All Classes  
[BlackjackConsole](#)  
[BlackjackHand](#)  
[Card](#)  
[ConsoleApplet](#)  
[ConsoleCanvas](#)  
[ConsolePanel](#)  
[Deck](#)  
[Hand](#)  
[TestBlackjackHand](#)  
[TestUI](#)



# UML Enhancements: Interaction modeling

Interactions are expressed more effectively using UML 2.0 constructs

Support specification of test scenarios

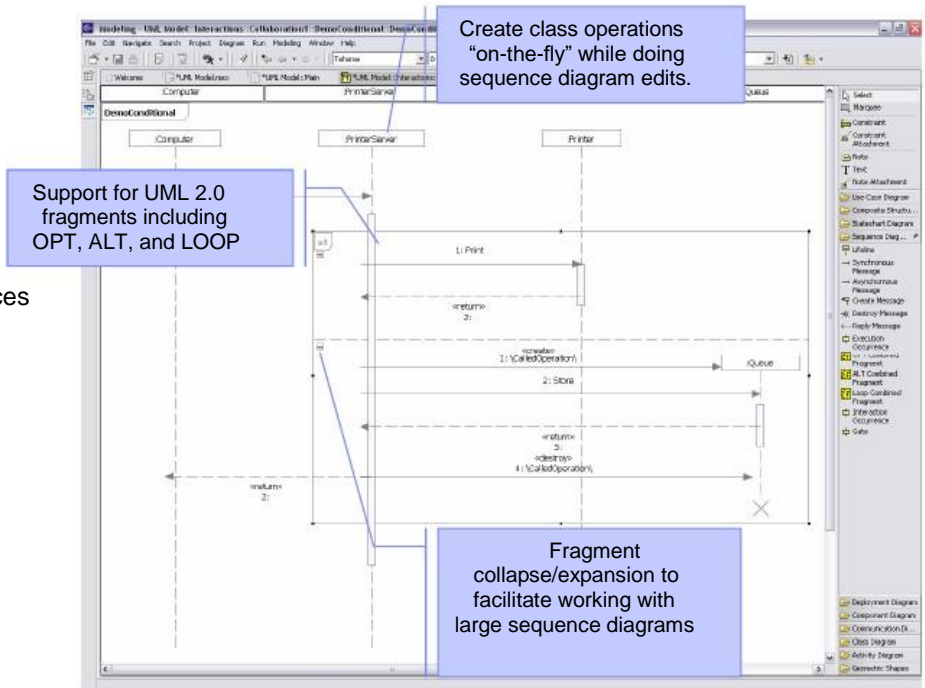
Loop, alt, opt

Interaction fragment references

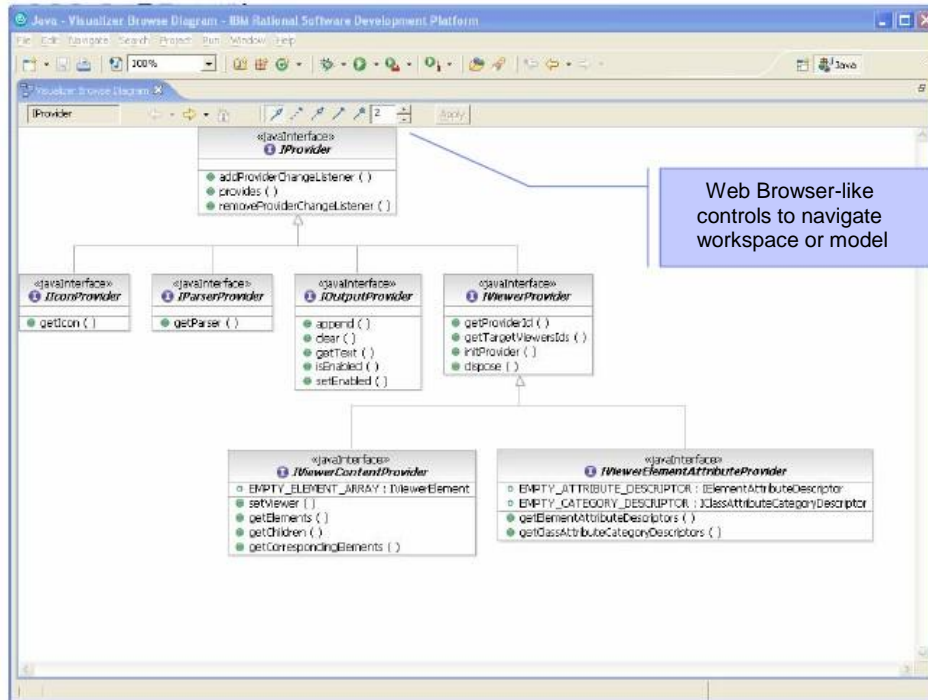
Interactions can be rendered as either sequence or communication diagrams

Sequence diagram editing improvements

Ordering and reordering



## UML Enhancements: Browse Diagrams



Enables users to understand and discover models and applications without having to create or maintain diagrams



## Team: RequisitePro integration

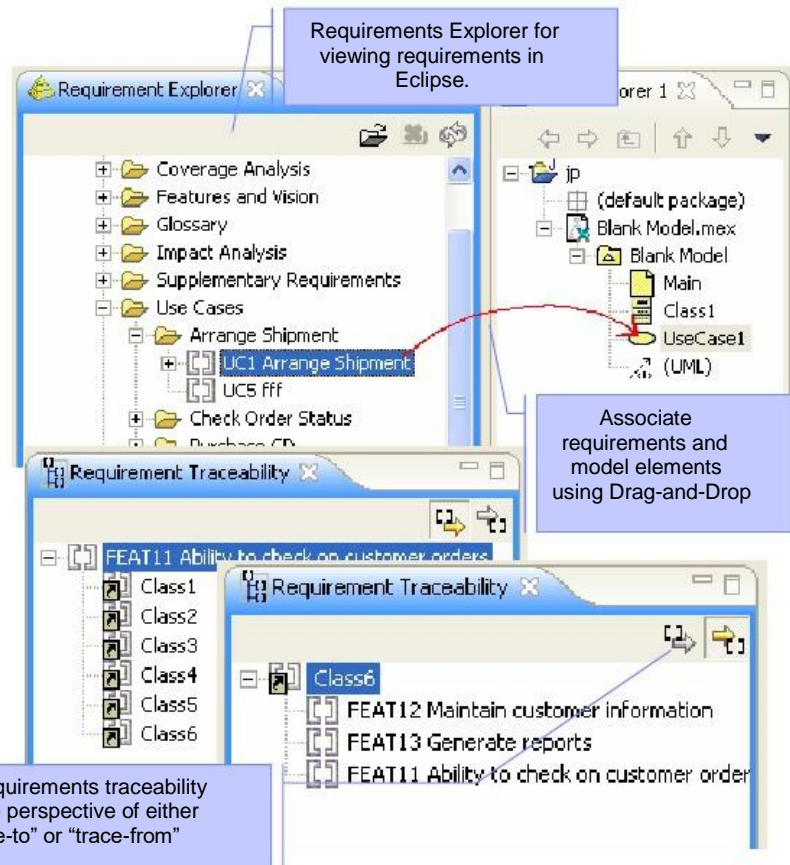
Open and browse multiple RequisitePro projects

See requirements, packages, and views

Associate requirements with model elements via drag and drop

Create model elements from requirements

Customizable synchronization



# Team: Process Guidance



Ease of Use

Integration with Rational Unified Process

Tool Mentors provide guidance for activities

User customizable views with user defined content

Improved navigation of RUP

The screenshot displays the RUP Navigator and RUP Advisor tool windows. The RUP Navigator window shows a tree view of the Rational Unified Process (RUP) phases and artifacts, with 'System Analyst' selected. The RUP Advisor window provides context-sensitive guidance for the 'System Analyst' role, showing a diagram of the role's responsibilities and a list of artifacts and activities. The Search window is also visible, showing a search query for 'use case actor' and a list of results.

**Role: System Analyst**

The System Analyst role leads and coordinates requirements elicitation and use-case modeling by outlining the system's functionality and identifying the system, for example, identifying what actors exist and what use cases they will require when interacting with the system.

**Topics**

- Description
- Related Information
- Staffing
- Further Reading

**RUP Advisor provides context sensitive guidance**

**Search**

Search query: use case actor

☐ Case sensitive

☒ Advanced

**Topics:**

- ☒ Tool Mentors
- ☒ Artifacts
- ☒ Activities
- ☒ Roles
- ☒ Workflow Details
- ☒ General Content

**Pages to include:**

- ☒ Main description
- ☐ Concept pages
- ☐ Checkpoints
- ☐ Guidelines
- ☐ Templates
- ☐ Examples

☒ Match whole

**RUP Advisor**

Tool Mentors (2)

- Tool Mentor: Working with Rational Architect
- Tool Mentor: Use Case Diagrams

Artifacts (2)

- Artifact: Actor
- Artifact: Use-Case Model

Checkpoints: Use Case

Guidelines: Use Case

Guidelines: Activity Diagram in the Use-Case Model

Report: Use Case -<use-case name>

Use Case Specification: <Use-Case Name>

Activities (2)

- Activity: Review Requirements
- Activity: Detail the Software Requirements

**Search is integrated with Eclipse search**



# Team: ClearQuest integration

The screenshot displays the IBM ClearQuest Rational Software Development Platform interface. The main window shows a hierarchical view of query results for 'ClearQuest Query Results (dev2003.06.000CLSC)'. The results are organized into a tree structure with columns for ID, Heading, Severity, Owner, and State. A context menu is open over the selected record, showing options like 'Change State', 'Assign...', 'Modify...', 'Utilities', 'Queue...', 'Postpone...', 'Details', 'Expand Selected', 'Refresh', and 'Properties'.

On the left, a 'ClearQuest Navigator' pane shows a tree of queries and reports, including 'Personal Queries', 'Public Queries', and 'Reports'. A callout points to this pane with the text: 'Easy access to queries, charts, & reports'.

At the bottom left, a 'Console' window shows the SQL query and its execution status. A callout points to this window with the text: 'Console, SQL Query & Properties views'.

At the bottom right, a 'ClearQuest Record Details' window shows a form for a specific record, including fields for ID, Description, Subj Project, UCN Project, Priority, Severity, and a large text area for the description. A callout points to this window with the text: 'View record forms, charts and reports'.

On the right side, a callout points to the hierarchical view of the query results with the text: 'Hierarchical result set view shows parent-child relationships'.

## Team: ClearCase integration

The screenshot displays the ClearCase - Eclipse Platform interface. The ClearCase Navigator on the left shows a project structure with folders like 'stef\_Rel3\_Int', 'My Activities', 'Classes', 'html', 'InvStat', 'jcd2.1', 'lost+found', 'processes', 'Proj', 'RobotIDatabase', 'Pose', 'Source', 'com', 'rational', 'cdshop', 'admin', 'business', 'servers', 'services', 'util', 'WINDNA', 'word', and 'XML'. The ClearCase Details view on the right shows a table of file elements with columns: Name, Size, Kind, Modified Time, State, Version, and Rule. The ClearCase View Configuration view at the bottom shows a table of version history with columns: Date, User, Name, Version, and Event Kind.

**ClearCase Details view shows selected version information**

Name	Size	Kind	Modified Time	State	Version	Rule
AdminFrame.java	2051	File Element Ver...	Oct 19, 2004 3:39:56 PM	Loaded	/main/1	Rel3
CDAdmin.java	2313	File Element Ver...	Oct 19, 2004 3:39:57 PM	Loaded	/main/Rel3_Int...	
CDMainFrame.java	17189	File Element Ver...	Oct 19, 2004 3:39:57 PM	Loaded	/main/1	
CustomizeMainFrame.java	17452	File Element Ver...	Oct 19, 2004 3:39:58 PM	Loaded	/main/1	
Fonts.java	3612	File Element Ver...	Oct 19, 2004 3:39:59 PM	Loaded	/main/Rel3_Int...	
Logon.java	4739	File Element	Oct 19, 2004 3:40:00 PM	Checked...	/main/Rel3_Int...	
TaxMainFrame.java	12131	File Element Ver...	Oct 19, 2004 3:40:00 PM	Loaded	/main/1	

**ClearCase Navigator view with integrated UCM activities**

**Display version history, view & update config spec, display search results**

Date	User	Name	Version	Event Kind
Oct 19, 2004 4:04:30 PM	demo	/Classics/Source/com/r...	/main/Rel3_Integration/stef_Rel3/CHECKEDOUT	checkout
Oct 19, 2004 4:04:30 PM	demo	/Classics/Source/com/r...	/main/Rel3_Integration/stef_Rel3/0	create
Oct 19, 2004 4:04:30 PM	demo	/Classics/Source/com/r...	/main/Rel3_Integration/stef_Rel3	create
Oct 14, 2002 3:50:34 PM	demo	/Classics/Source/com/r...	/main/Rel3_web_Int/2	create
Oct 14, 2002 3:50:34 PM	demo	/Classics/Source/com/r...	/main/Rel3_web_Int/1	create
Oct 14, 2002 3:50:04 PM	demo	/Classics/Source/com/r...	/main/Rel3_web_Int/0	create
Oct 14, 2002 3:50:04 PM	demo	/Classics/Source/com/r...	/main/Rel3_web_Int	create
Oct 14, 2002 3:48:41 PM	demo	/Classics/Source/com/r...	/main/Rel3_Integration/2	create
Oct 14, 2002 3:47:43 PM	demo	/Classics/Source/com/r...	/main/Rel3_Integration/1	create
Oct 14, 2002 3:47:42 PM	demo	/Classics/Source/com/r...	/main/Rel3_Integration/0	create
Oct 14, 2002 3:47:42 PM	demo	/Classics/Source/com/r...	/main/Rel3_Integration	create
Oct 14, 2002 3:36:03 PM	demo	/Classics/Source/com/r...	/main/rel1_bugfix/1	create
Oct 14, 2002 3:36:01 PM	demo	/Classics/Source/com/r...	/main/rel1_bugfix/0	create
Oct 14, 2002 3:36:01 PM	demo	/Classics/Source/com/r...	/main/rel1_bugfix	create
Oct 14, 2002 3:35:56 PM	demo	/Classics/Source/com/r...	/main/2	create
Oct 14, 2002 3:35:54 PM	demo	/Classics/Source/com/r...	/main/1	create
Oct 14, 2002 3:35:48 PM	demo	/Classics/Source/com/r...	/main/0	create
Oct 14, 2002 3:35:48 PM	demo	/Classics/Source/com/r...	/main	create
Oct 14, 2002 3:35:48 PM	demo	/Classics/Source/com/r...	/main	create

## Team: Model compare & merge

Model differences & conflicts

Description of selected difference or conflict

Diagram view of selected difference or conflict for contributor

Choose view type

## Summary: Key Features Rational Software Architect SE

### Architecture Support

- Java, J2SE, C++
- UML2 Modeling
- Architecture Discovery via Application Analysis
- Patterns and Transformations

### Team Environment

- Enhanced Compare / Merge
- Integrated RequisitePro Views
- Process Advisor
- CC and CQ fully integrated

### Open Platform

- Based on Eclipse 3.4 Shell
- Testing and Team tools work together

