Instrumente pentru Dezvoltarea Programelor

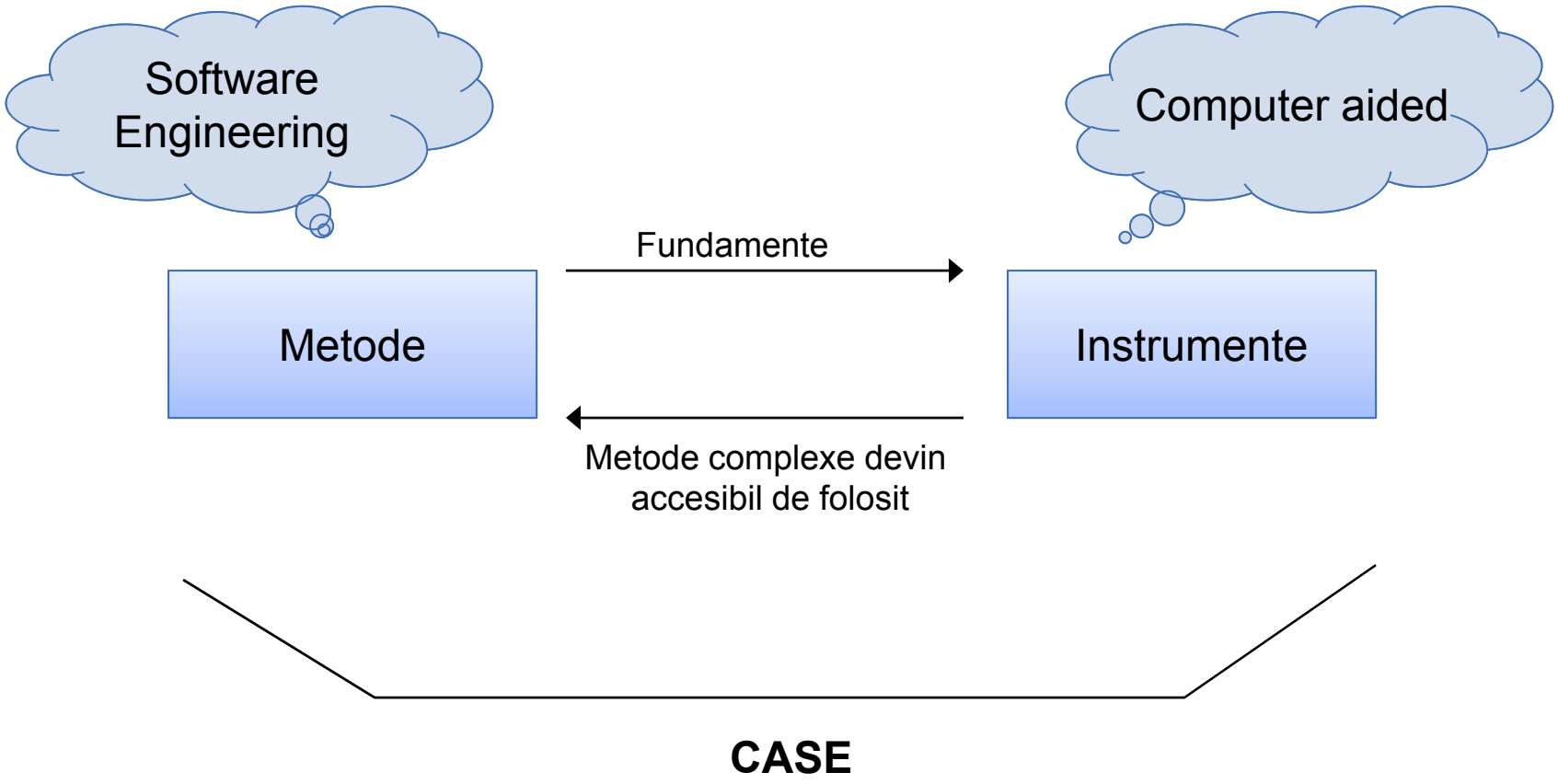
Instrumente CASE.

Computer-Aided Software Engineering

- Ingineria asistată de calculator (CASE) - instrumente de asistenţă a ciclurilor de dezvoltare şi evoluţie software

- *Prin instrumente CASE înţelegem aplicaţiile software care-i sprijină/ajută pe analişti, proiectanţi, programatori, inclusiv personalul de testare şi întreţinere, să analizeze, să proiecteze, să implementeze (cel puţin parţial), să modifice (extindă), respectiv să construiască teste pentru sistemele informatice.*

Perspectiva

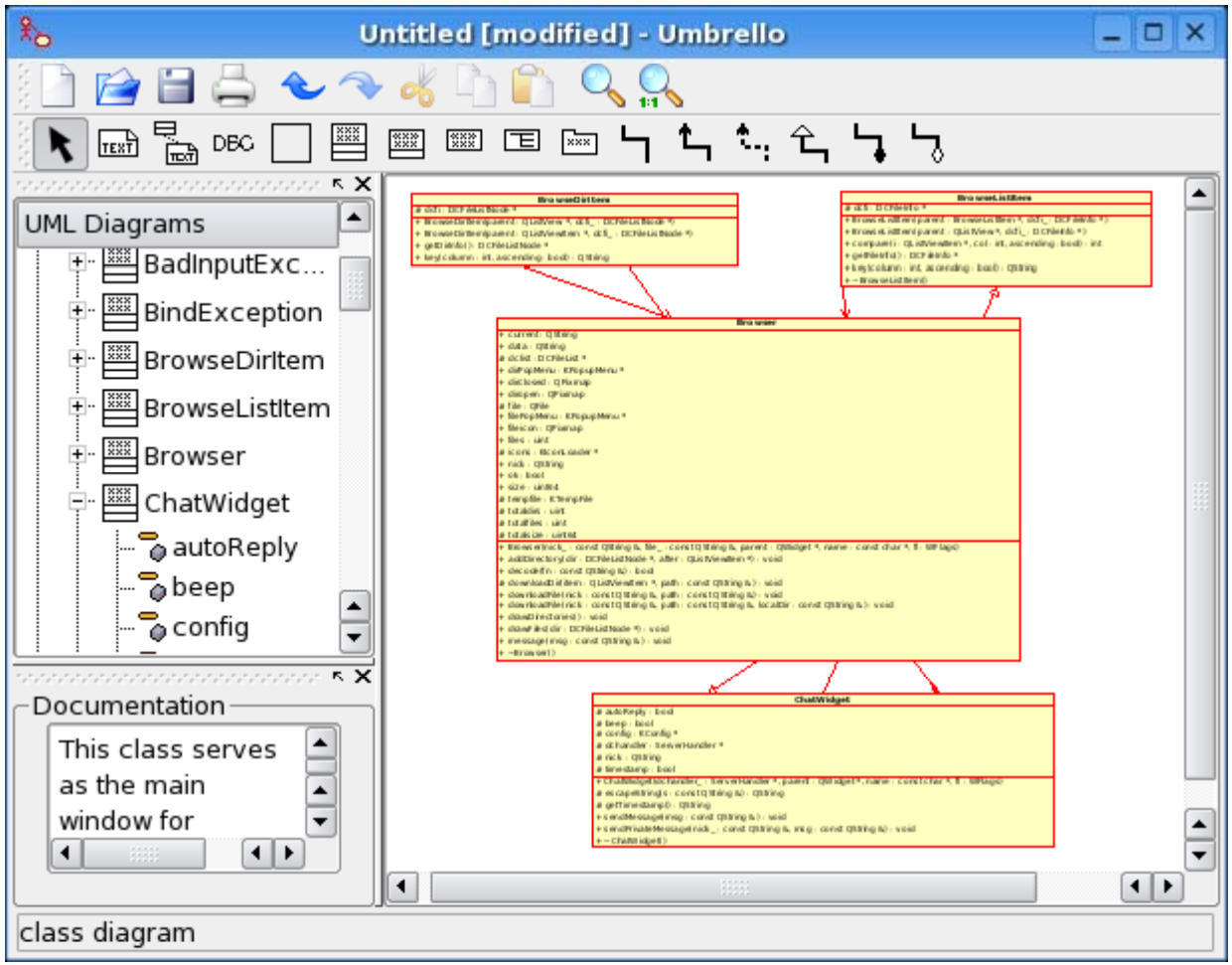


Instrumente CASE

- Programatorii au nevoie de:
  - Versiuni la zi ale documentelor unui proiect
  - Sistem de ajutor online pentru limbajul de programare, editor, etc.
  - Diverse manuale online
  - Editoare cu funcţii de automatizare (auto-completion, syntax checker, etc)

- Automatizarea activităţilor
  - IDEs (ex. Eclipse, Visual Studio)
  - Instrumente de automatizare a construcţiei (ex. ant, make)
  - Instrumente de construcţie grafică (GUI) pentru interfeţe utilizator (ex. VE pentru Eclipse, VS Designer)
  - Version control systems (Subversion, CVS, etc.)
  - Instrumente de modelare (Together, ArgoUML, Rational Rose)
  - Generatoare (JET, ANTLR) de rapoarte, ecrane, etc.
  - Reverse engineering tools
  - …

Exemplu de instrument CASE

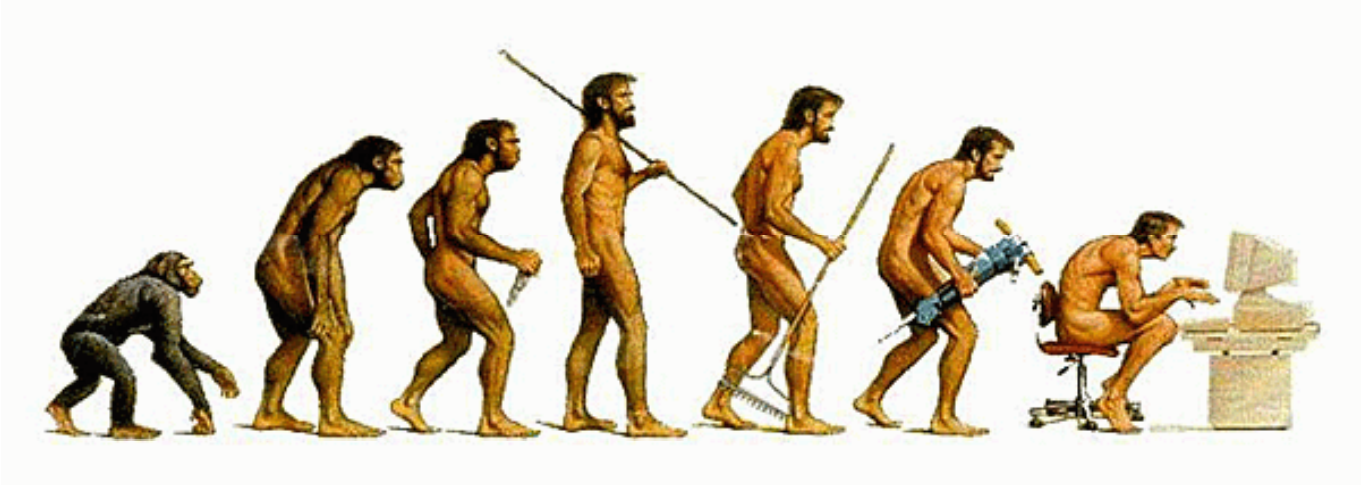


Istoric

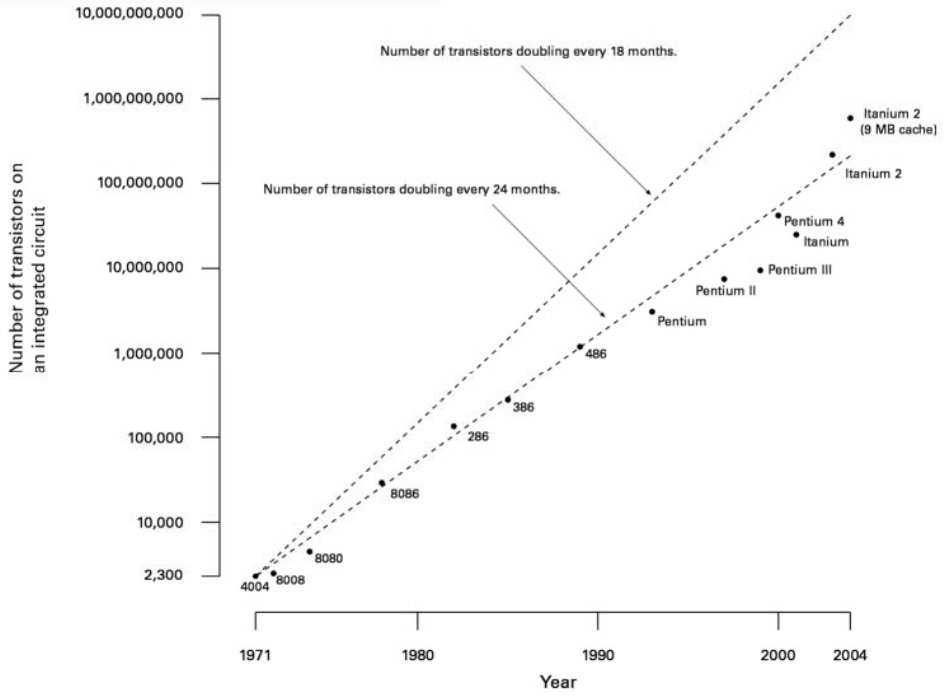
- Evoluţia instrumentelor CASE:
  - Programarea considerata o “forma de arta” – anii 60
  - Metode structurate – 1965
  - Tehnici de modelare a datelor – 1970
  - Limbaje de generaţia a patra – 1975
  - Instrumente de proiectare a specificaţiilor software – 1980
  - Instrumente pentru prototipizarea interfeţei utilizator – 1985
  - Instrumente pentru generarea automată a codului – 1990
  - CASE-uri integrate, CASE-uri orientate obiect – 1995
  - Component Software (Java) 1996

Mai multe pe http://en.wikipedia.org/wiki/Computer-aided_software_engineering#History_of_CASE

Instrumente CASE



Moore's Law



Evolutie?

Software tools?

Ei ar putea lucra fara instrumente?

Dar voi?

Trei generaţii de instrumente CASE

- *Prima generaţie CASE*
- Un instrument pentru o anumita etapa a procesului software
  - planificarea strategică (la nivelul sistemelor complexe)
  - etapa de analiză
  - etapa de proiectare
  - generare de cod
- Caracteristici:
  - oferă interfaţă grafică pentru utilizator
  - instrumente de dimensiuni mici, volum mare de date
  - generare de cod se referă la definirea datelor (ecrane, rapoarte, definiţii, fragmente de cod).

Trei generaţii de instrumente CASE

- *A doua generaţie de produse CASE*
  - Aceleaşi facilităţi ca şi cele din prima generaţie şi în general aceleaşi tipuri de instrumente +
  - Generarea de cod să se realizeze pe main-frame-uri pe care să existe un generator central de cod şi care să stocheze codul generat într-un depozit.
- Permit lucrul în echipă pentru elaborarea de proiecte de obicei complexe
- Asigură facilităţi de management de proiect.
- Acestei generaţii îi aparţin CASE-uri care oferă suport pentru întreg ciclul de viaţă (integrated-CASE sau I-CASE)
  - oferă suport pentru realizarea proiectelor folosind mai multe metode de analiză şi proiectare.

Trei generaţii de instrumente CASE

- ***Generaţia a treia*** de produse CASE
  - cuprinde CASE-urile ultimei perioade, numite şi medii sau Workbench (colecţie de instrumente CASE şi de alte componente integrate care asigură suportul pentru majoritatea tipurilor de interacţiuni între componentele mediului şi între utilizator şi mediu)
- Generaţia a treia de CASE presupune utilizarea acestora în organizaţii elaboratoare de software şi este generaţia actuală care oferă:
  - facilităţi individuale pe PC
  - facilităţi la nivel de proiect pe LAN
  - facilităţi la nivel de organizaţie pe mainframe

Clasificarea CASE

1. Instrumente CASE de nivel superior
   - utilizate în fazele de analiză şi de proiectare ale procesului de dezvoltare a sistemelor (asigură realizarea diagramelor, generarea formularelor şi a rapoartelor etc.);
2. Instrumente CASE de nivel inferior
   - permit proiectarea şi realizarea sistemului vizat (CASE pentru implementare, verificare, stabilire a configuraţiei etc.).

1. UpperCASE (front-end tool)
2. LowerCASE (back-end tool)

Clasificarea CASE

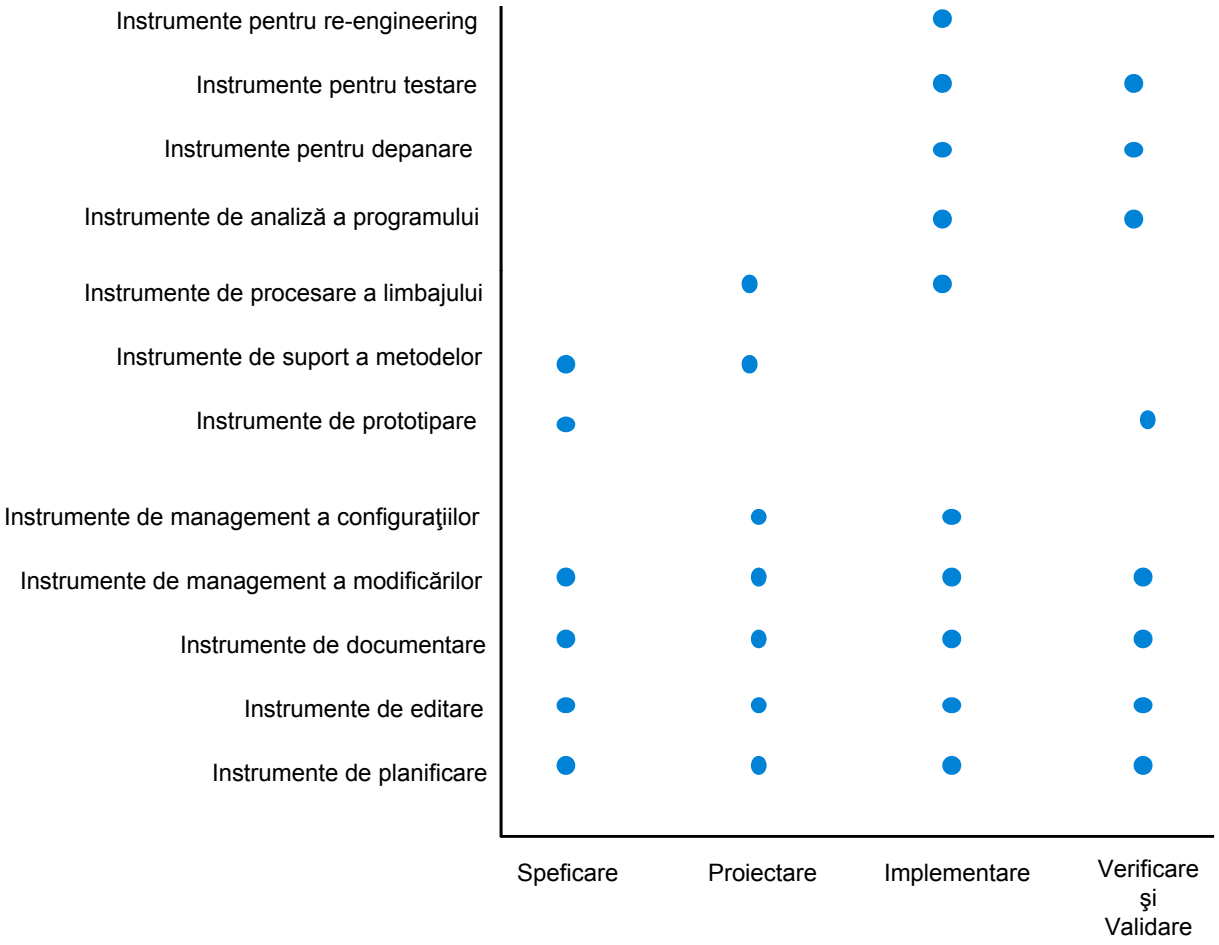
Ian Sommerville clasifică instrumentele din 3 perspective:

1. Perspectiva funcţională
   - pe baza funcţiilor specifice furnizate
2. Perspectiva proceselor
   - în funcţie de activităţile de proces suportate
3. Perspectiva integrării
   - în funcţie de modul în care sunt organizate în unităţi integrate ce furnizează suport pentru unul sau mai multe activităţi ale procesului software

Clasificare funcţională

Tip de instrument CASE	Exemple
Instrumente de planificare	Instrumente PERT, instrumente de estimare, spreadsheets
Instrumente de editare	Editoare text, editoare de diagrame, procesoare word
Instrumente de gestiune a modificărilor	Instrumente de gestiune a cerinţelor, sisteme de control a modificărilor
Instrumente de management a configurărilor	Sisteme de management a versiunilor, instrumente de construcţie a sistemelor
Instrumente de prototipare	Limbaje de nivel foarte înalt, generatoare de interfeţe utilizator
Instrumente de suport a metodelor	Editoare de proiect, dicţionare de date, generatoare de cod
Instrumente de procesare a limbajului	Compilatoare, interpretoare
Instrumente de analiză a programului	Generatoare de referinţe încrucişate, analizoare statice, analizoare dinamice
Instrumente de testare	Generatoare de date de test, comparatoare de fişiere
Instrumente de depanare	Sisteme de depanare interactivă
Instrumente de documentare	Programe de formatare a paginilor, editoare de imagini
Instrumente de re-eingineering	Sisteme de referinţe încrucişate, sisteme de restructurare a programelor

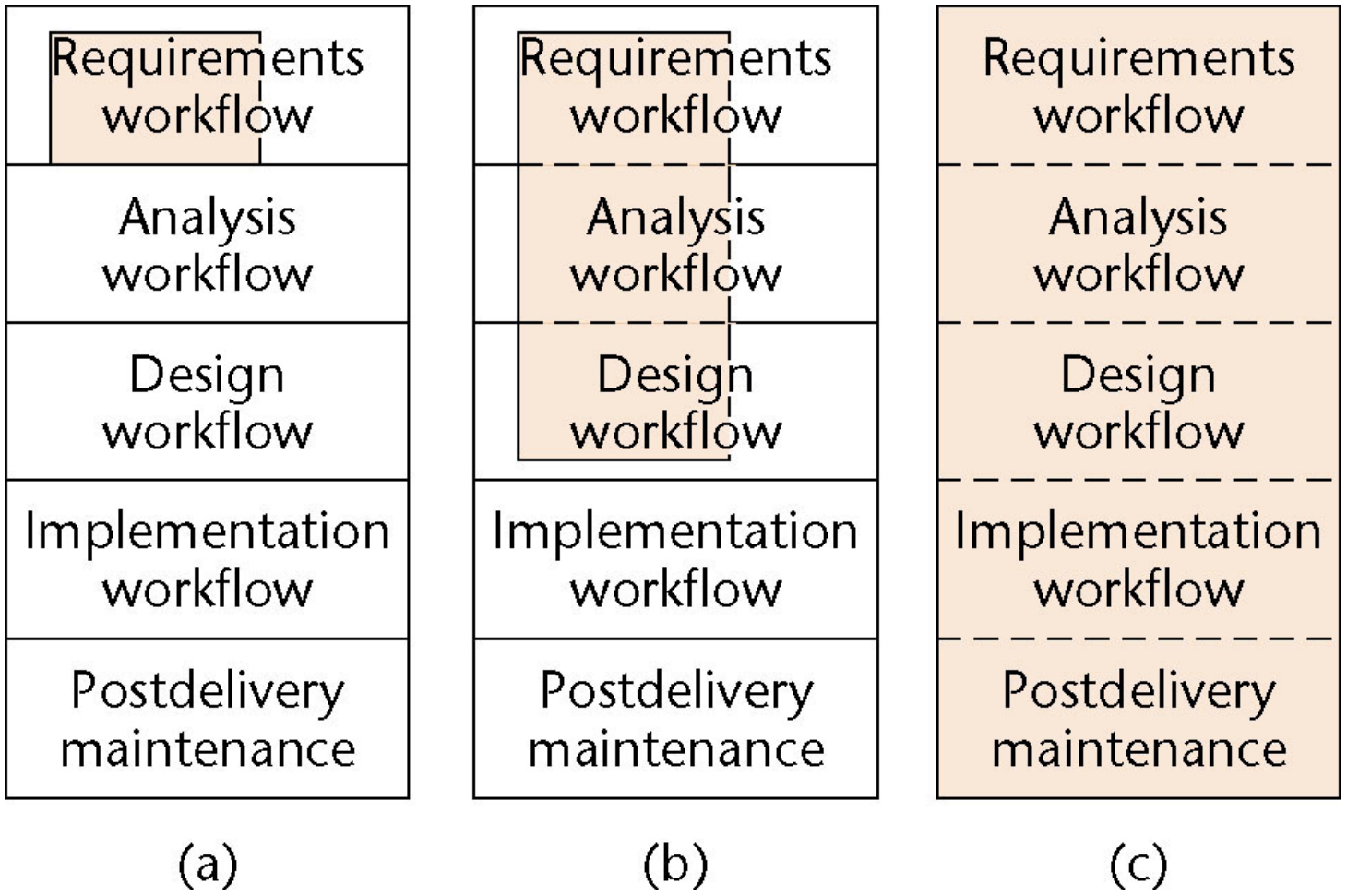
Clasificare bazată pe activităţi



Suportul oferit procesului software

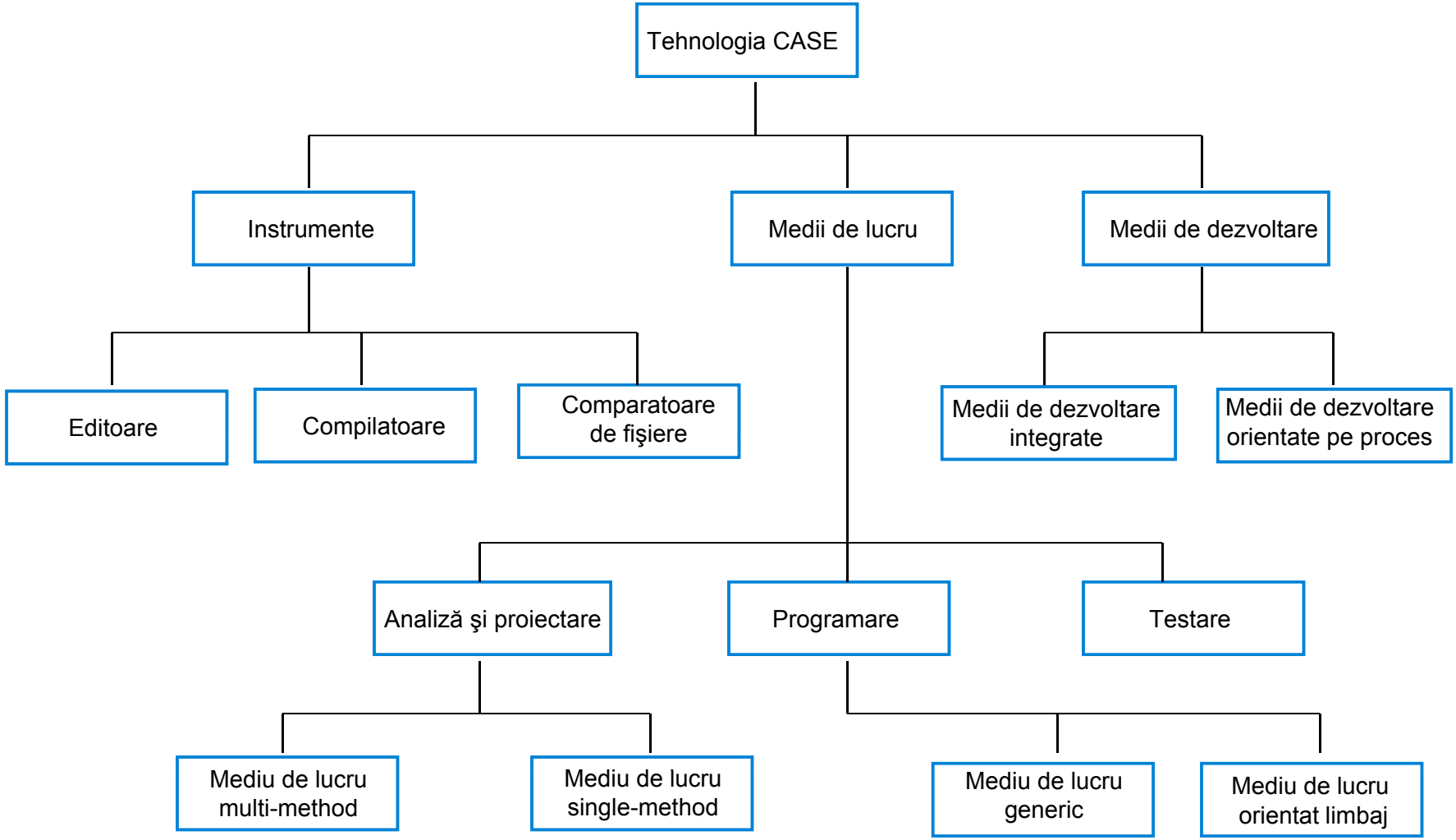
- Instrumente (ex. make, ant, javadoc)
  - Suportă task-uri individuale precum compilare, editare, etc.
- Medii de lucru/workbenches (=IDE/Integrated Development Environments) (ex. Eclipse, Visual Studio)
  - Suportă etape distincte ale procesului software (specificaţia, proiectarea, etc.)
  - Includ adesea un număr de instrumente integrate
- Medii de dezvoltare/environments (ex. suita IBM WebSphere)
  - Suportă toate sau un set substanţial al activităţilor procesului software
  - Includ de obicei mai multe medii de lucru

Instrument vs. Mediu de lucru vs. mediu de dezvoltare

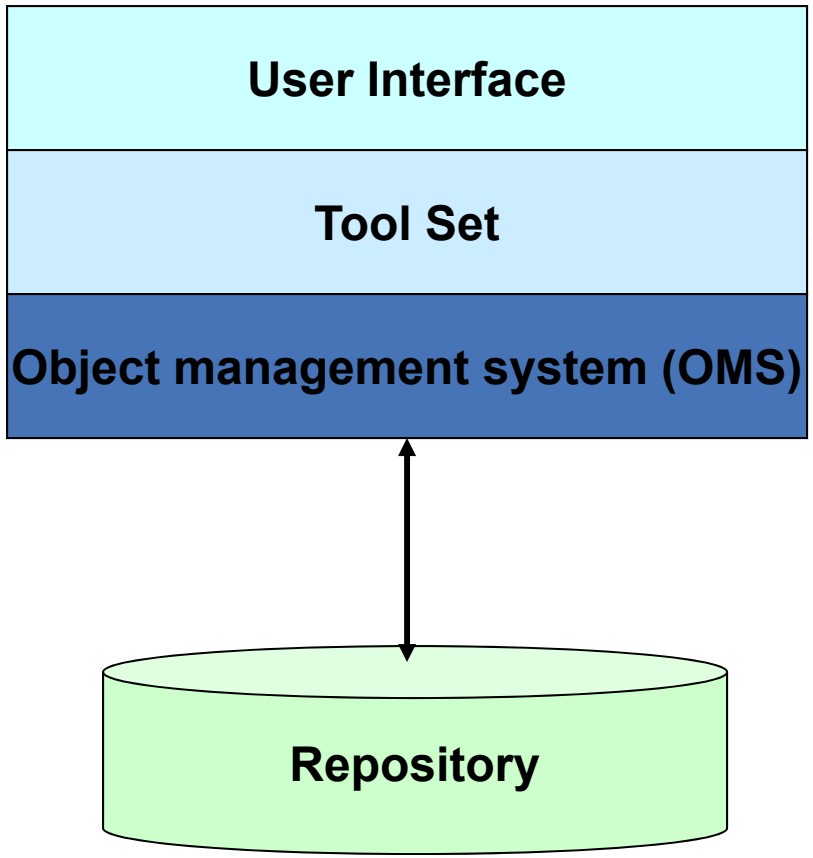


(a) Tool vs. (b) workbench vs. (c) environment

Instrumente, medii de lucru, medii de dezvoltare



Arhitectura unui mediu CASE



Componente de bază ale unui sistem CASE

- Depozitul de date (data repository)
  - acumulează şi stochează, în mod organizat, toate informaţiile introduse de diferite persoane, la momente diferite de timp, care vor servi în etapele de analiză, proiectare şi creare a codului
  - se pot delimita depozitul de informaţii (Information Repository) – conţinând informaţiile despre afacerea organizaţiei şi despre portofoliul său de aplicaţii - şi dicţionarul de date (Data Dictionary), care specifică numele (identificatorii) şi descrierea datelor, gestionează controlul *accesului* la depozitul de informaţii, conţine descrierile resurselor necesare prelucrărilor datelor;
- Editorul de diagrame
  - componentă ce facilitează realizarea şi modificarea diagramelor specifice metodologiei pentru a fost creat instrumentul CASE respectiv;
- Analizorul de structură
  - găseşte şi elimină erorile dificil de localizat după culegerea informaţiilor, efectuând comparaţii între datele nou introduse şi cele deja existente în baza de informaţii;

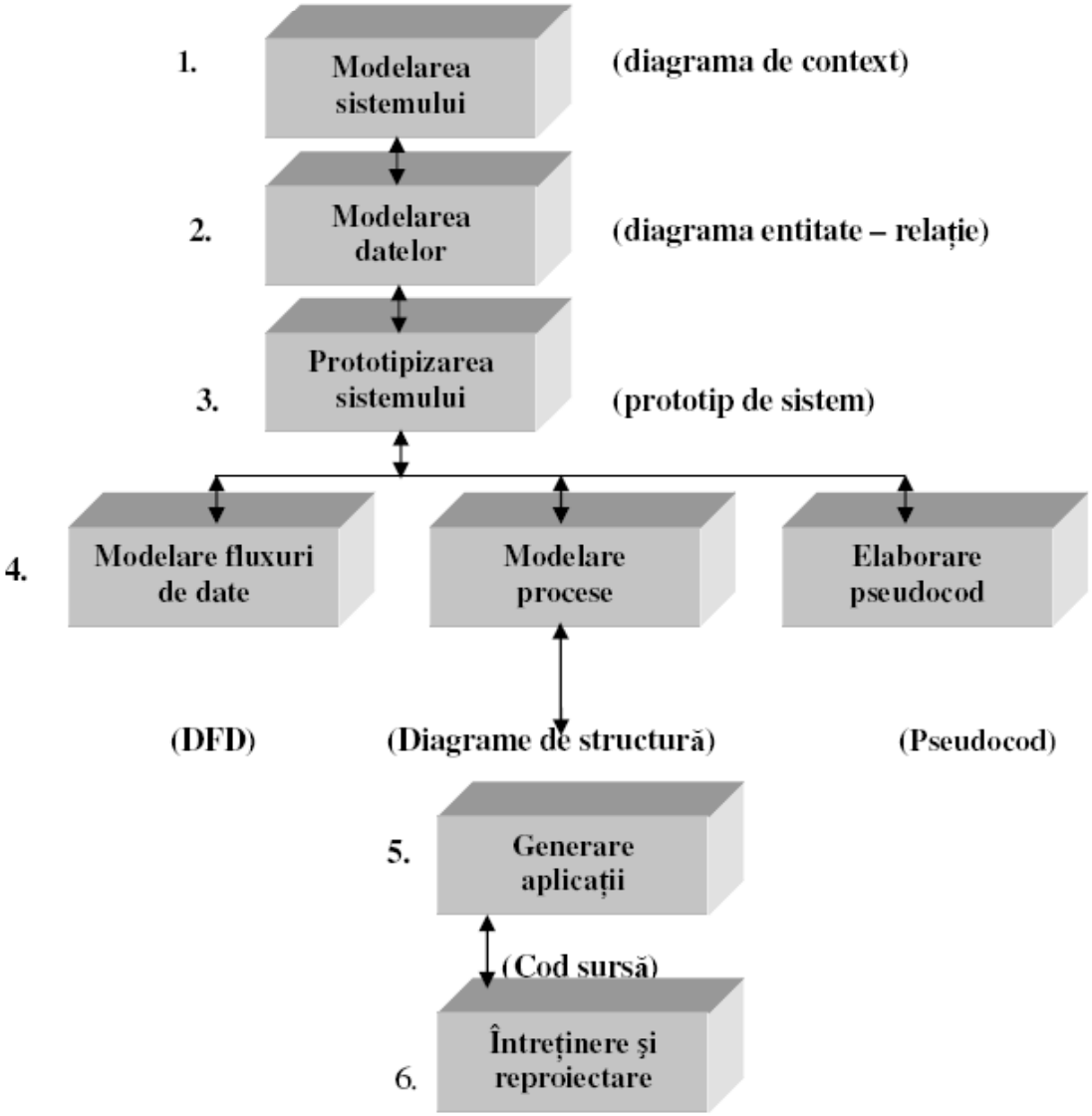
Componente de bază ale unui sistem CASE

- Instrumente pentru *reverse engineering*
  - asigură actualizarea diagramelor conform schimbărilor realizate în codul sursă;
- Generatorul de cod
  - poate converti în cod diagramele realizate în faza de proiectare;
- Navigatorul specializat
  - instrument pentru vizualizarea informaţiilor unui ansamblu de entităţi care au o structură complexă, între care există un mare număr de relaţii;
- Generatorul de documentaţie
  - include modele de documente, oferind utilizatorilor posibilitatea de a-şi concepe propriile documente într-o manieră flexibilă;

Componente de bază ale unui sistem CASE

- Generatorul de formulare şi de rapoarte
  - conceperea interfeţelor (interactivitatea) produsului cu utilizatorii;
- Componente de transformare
  - permit trecerea de la un model sau o diagramă la alt model, respectiv la altă diagramă;
- Instrumentele pentru managementul de proiect
  - oferă facilităţi destinate gestiunii configuraţiei fiecărui proiect (proiectul de aplicaţie, codul şi documentaţia unui sistem dezvoltat);
- Instrumentele de verificare automată a aplicaţiei

Instrumente CASE şi procesul software



Avantaje ale folosirii CASE

- Realizarea automată a documentaţiei sistemelor
  - Documentaţia şi specificaţiile de proiectare reprezintă piesele de bază ale unui proiect de dezvoltare
  - Din lipsă de timp sau din neglijenţă, documentaţia este ultimul lucru la care se gândesc cei din echipa de realizare
  - După implementarea sistemului şi mai ales în timpul exploatării şi întreţinerii lui se consumă un timp foarte mare din lipsa informaţiilor privind detaliile de proiectare a sistemului
  - Prin instrumentele CASE, documentaţia şi specificaţiile sistemului se pot obţine automat pe baza depozitului datelor, oferind astfel posibilitatea ca echipa de specialişti să nu mai fie “sufocaţi” şi de această responsabilitate

Avantaje ale folosirii CASE

- Automatizarea parţială sau totală a fazelor de analiză şi proiectare a sistemelor
  - Determină scurtarea ciclului de viaţă al sistemelor, creşterea calităţii lor şi eliminarea erorilor de proiectare
  - Instrumentele CASE dispun de module de verificare şi validare, de tehnici de normalizare a datelor şi chiar de prototipizare a sistemelor

Avantaje ale folosirii CASE

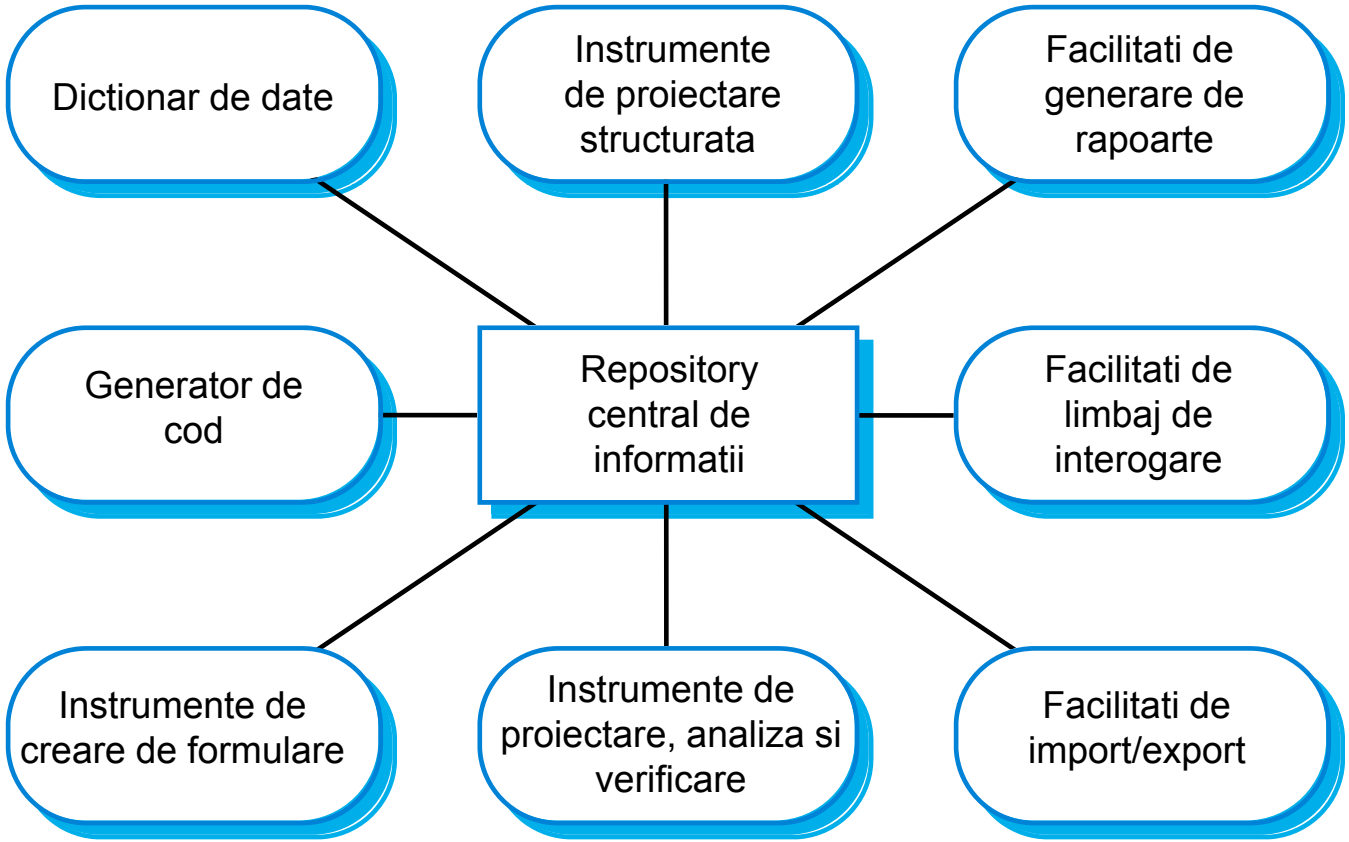
- Coordonarea/managementul proiectelor de dezvoltare a sistemelor
  - Sunt puse la dispoziţia membrilor echipei de proiectare informaţii cu privire la activităţile desfăşurate şi rezultatele obţinute, astfel încât există posibilitatea ca în orice moment să se cunoască cu exactitate stadiul de dezvoltare al sistemului, timpul şi resursele consumate

- Generarea automată a codului-sursă al aplicaţiilor
  - Una din promisiunile CASE-ului o constituie susţinerea eforturilor din faza de formulare a cerinţelor şi până în faza de implementare şi întreţinere
  - Dezvoltatorul – componenta de creativitate (gândiţi-vă la RPC, CORBA, etc.)

Studiu de caz: instrumente CASE pentru analiză şi proiectare

- În procesul de analiză sunt adesea folosite modele
  - înţelegerea şi dezvoltarea sistemului
  - omise o serie de detalii (abstractizare a sistemului studiat şi nu o reprezentare alternativă a sistemului)
- Exemple de tipuri de modele de sistem posibil create de-a lungul fazei de analiză includ:
  - *Modelul flux de date*. Arată modul în care sunt procesate datele în diversele etape ale funcţionării sistemului.
  - *Modelul compozit* = model *agregare.* Arată modul în care entităţile sistemului sunt compuse din alte entităţi.
  - *Modelul arhitectural*. Prezintă principalele subsisteme ce compun sistemul.
  - *Modelul clasificare*. Clasele de obiecte/diagramele de moştenire prezintă modul în care entităţile sistemului prezintă caracteristici comune.
  - *Modelul stimul-răspuns* = *diagramă stare tranziţie*. Arată modul în care sistemul reacţionează la evenimente interne şi externe.

Studiu de caz: instrumente CASE pentru analiză şi proiectare



Studiu de caz: instrumente CASE pentru analiză şi proiectare

- *Editoarele de diagrame*
  - folosite pentru crearea de modele ale obiectelor, modele ale datelor, modele comportamentale, etc.
  - captează diverse informaţii legate de entităţi şi pot salva informaţii în repository-ul central.
- *Instrumentele de analiză şi verificare a proiectului*
  - ajută la procesarea proiectului şi la raportarea unor erori şi anomalii.
  - pot fi integrate cu sistemul de editare astfel încât erorile utilizatorului să fie regăsite într-un stagiu timpuriu al procesului software
- *Limbaje de interogare a repository-ului*
  - permit proiectantului regăsirea erorirlor şi a informaţiilor asociate de proiectare în repository-ul central.
- *Dicţionarul de date*
  - menţine informaţii relative la entităţile folosite într-un proiect al sistemului.

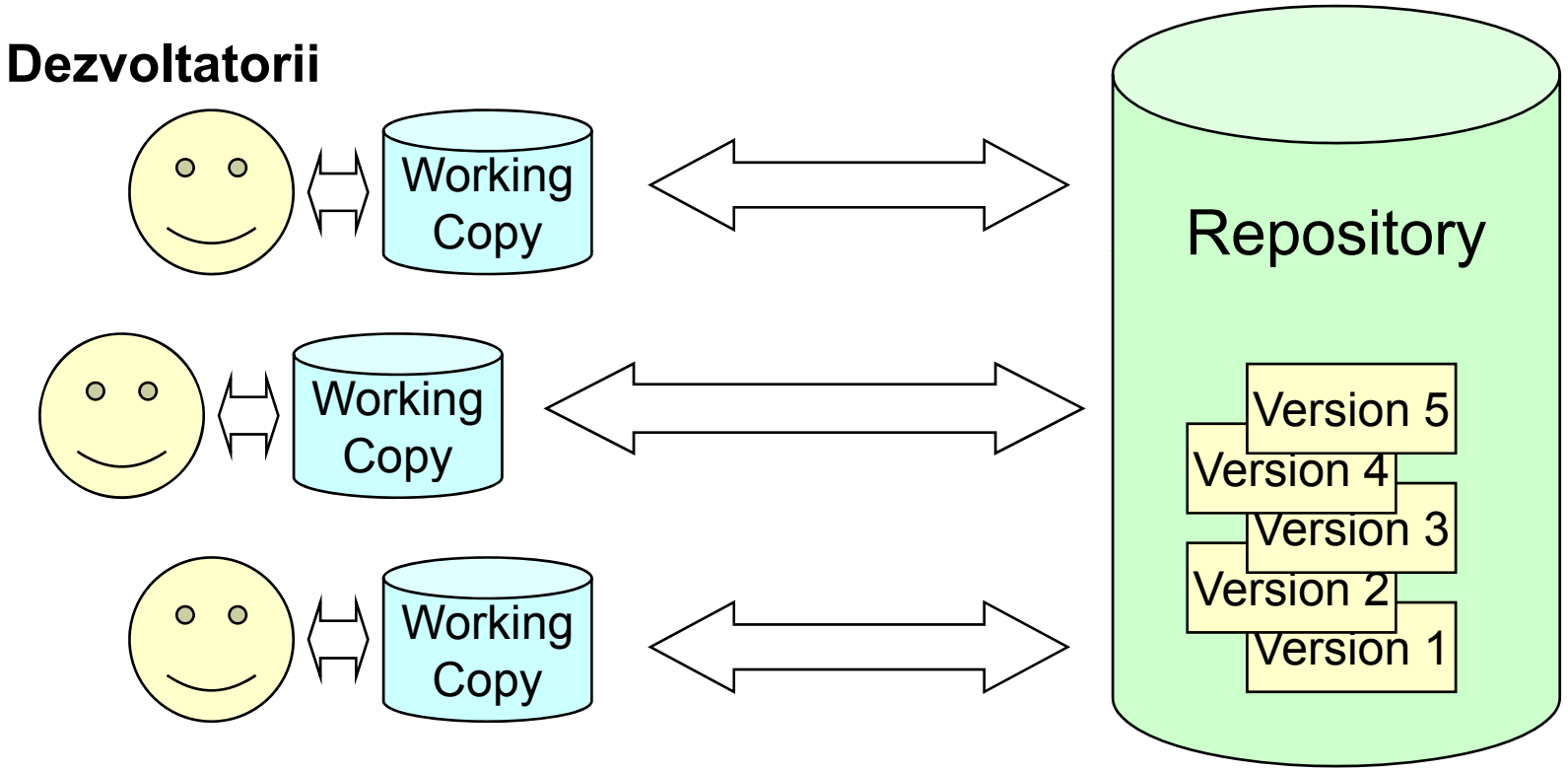
Studiu de caz: instrumente CASE pentru analiză şi proiectare

- *Instrumentele de definire şi generare a rapoartelor*
  - preiau informaţia din repository şi generează documentaţie privind sistemul.
- *Instrumente de definire a formularelor*
  - permit specificarea de formate pentru ecrane şi documente.
- *Facilităţile de import/export*
  - permit schimbul de informaţii din repository-ul central cu diverse ale instrumente de dezvoltare.
- *Generatoarele de cod*
  - Generarea de cod sau schelete de cod automat pornind de la proiectul sistemului definit şi păstrat în repository-ul central.

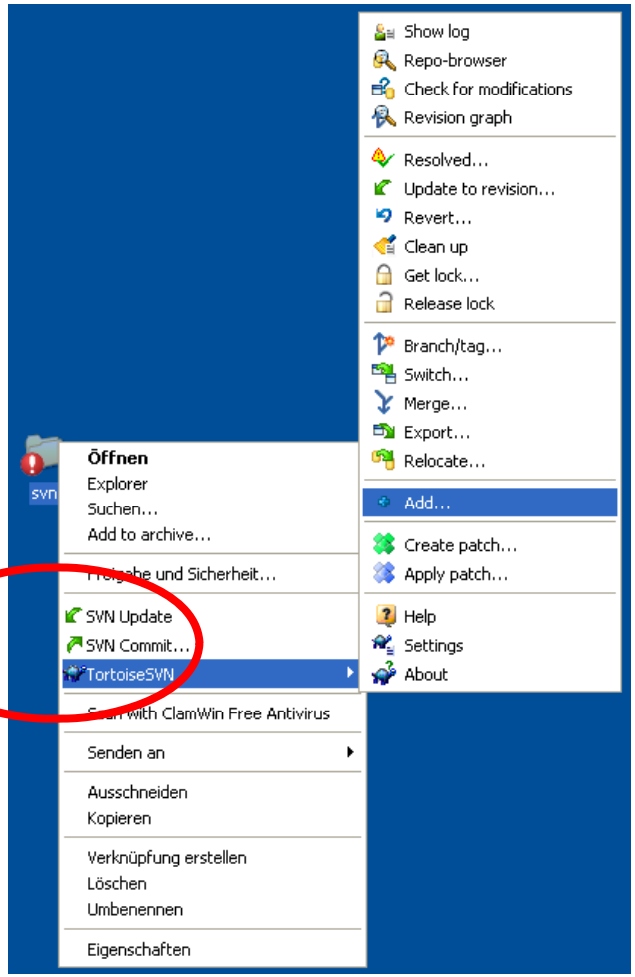
Version Control Systems

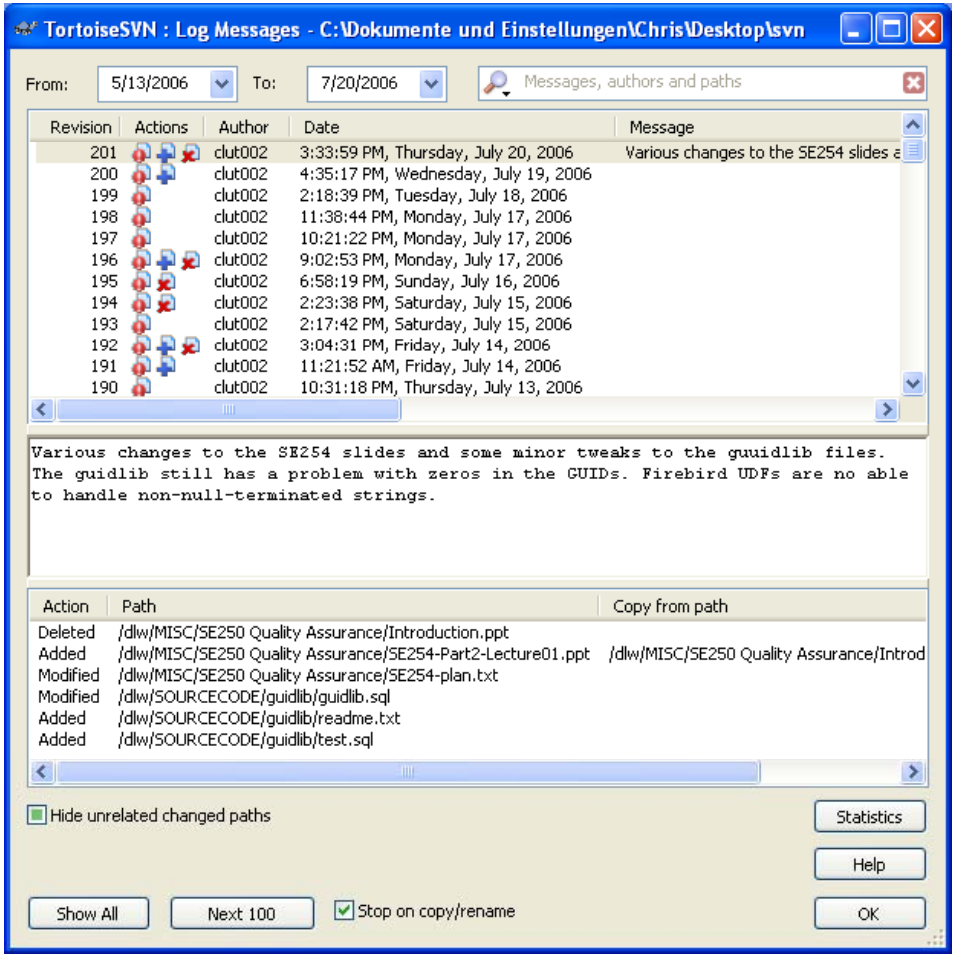
- Tehnologie pentru gestionarea schimbărilor pe care mai mulţi dezvoltatori le fac în comun asupra unui *repository* partajat
- Schimbarea duce la crearea de noi versiuni ale fişierelor modificate
- Versiunile vechi sunt întotdeauna accesibile
- Permite utilizatorilor
  - partajarea proiectelor
  - asigurarea că schimbările efectuate de un dezvoltator nu sunt accidental şterse sau alterate de un alt dezvoltator prin schimbările efectuate de acesta

Version Control Systems



Exemplu: Subversion





SVN

- SVN sau Subversion
  - Instrument CASE folosit astăzi de mulţi dezvoltatori pentru menţinerea modificărilor efectuate asupra codului sursă (dezvoltare colaborativă)
- SVN este popular în rândul comunităţilor open-source
- Folosit în multe proiecte open source, precum Apache Software Foundation, KDE, GNOME, Python, and etc.
- SVN rulează pe sisteme de operare precum UNIX, Win32 sau MacOS X

Istoria SVN

- În 2000 CollabNet Inc. a încercat scrierea unui înlocuitor al popularului (la vremea aceea) CVS din cauza limitărilor acestuia.
- Bad News! CVS la ora respectivă devenise standardul de facto în rândul comunităţilor open source! De ce? Pentru că oamenii nu aveau altă soluţie decât să folosească CVS!
- CollabNet s-a hotărât să scrie o noua versiune de version control system de la zero! DAR bazat pe CVS şi folosind CVS
- Obiectivele urmărite:
  - Menţinerea unei metodologii de control a versiunilor
  - Păstrarea funcţionalităţilor oferite de CVS
  - Produsul trebuia să fie similar CVS a.î. Orice utilizator putea uşor adopta noul produs fără mare efort de înţelegere a conceptelor
- În August 2001 Subversion a devenit disponibil gratuit şi descărcabil dintr-un repository CVS!!!

Câteva dintre funcţionalităţile oferite

- Directory versioning
  - SVN implementează un sistem de fişiere virtual ce menţine consistenţa modificărilor la nivelul întregului arbore de directoare de-a lungul timpului
- True version history
  - Se pot adăuga, şterge, copia şi redenumi ATÂT fişiere cât şi directoare.
- Atomic commits
  - Previne apariţia unor probleme ca urmare a comiterii numai a unui set parţial de modificări.
  - Numerele de revizie se acordă per commit şi nu pentru fiecare fişier comis
  - Fiecărei revizii îi sunt ataşate mesaje de log (şi nu sunt stocate redundant ca în cazul CVS)
- Versioned metadata
  - Metadatele stocate împreună cu fişierele şi directoare pot fi de asemenea versionate.
- Choice of network layers
  - Este permisă folosirea mai multor protocoale de acces la repository: HTTP şi HTTPS (Apache server cu protocolul WebDAV/DeltaV)

Funcţionalităţi oferite (cont.)

- Consistent data handling
  - Folosind un algoritm de diferenţiere binară ce funcţionează asupra atât a fişierelor text, cât şi a celor binare.
  - Diferenţele între fişiere sunt transmise în ambele direcţii în reţea (client <-> repository server)
- Efficient branching and tagging
  - Sunt create ramuri şi taguri noi prin copierea proiectelor (similar modalităţii hard-link) – reducerea timpului
- Hackability
  - Este implementat ca o colecţie de biblioteci C partajate.
  - Uşor de menţinut şi folosit de către alte aplicaţii şi limbaje de dezvoltare.

SVN vs. CVS

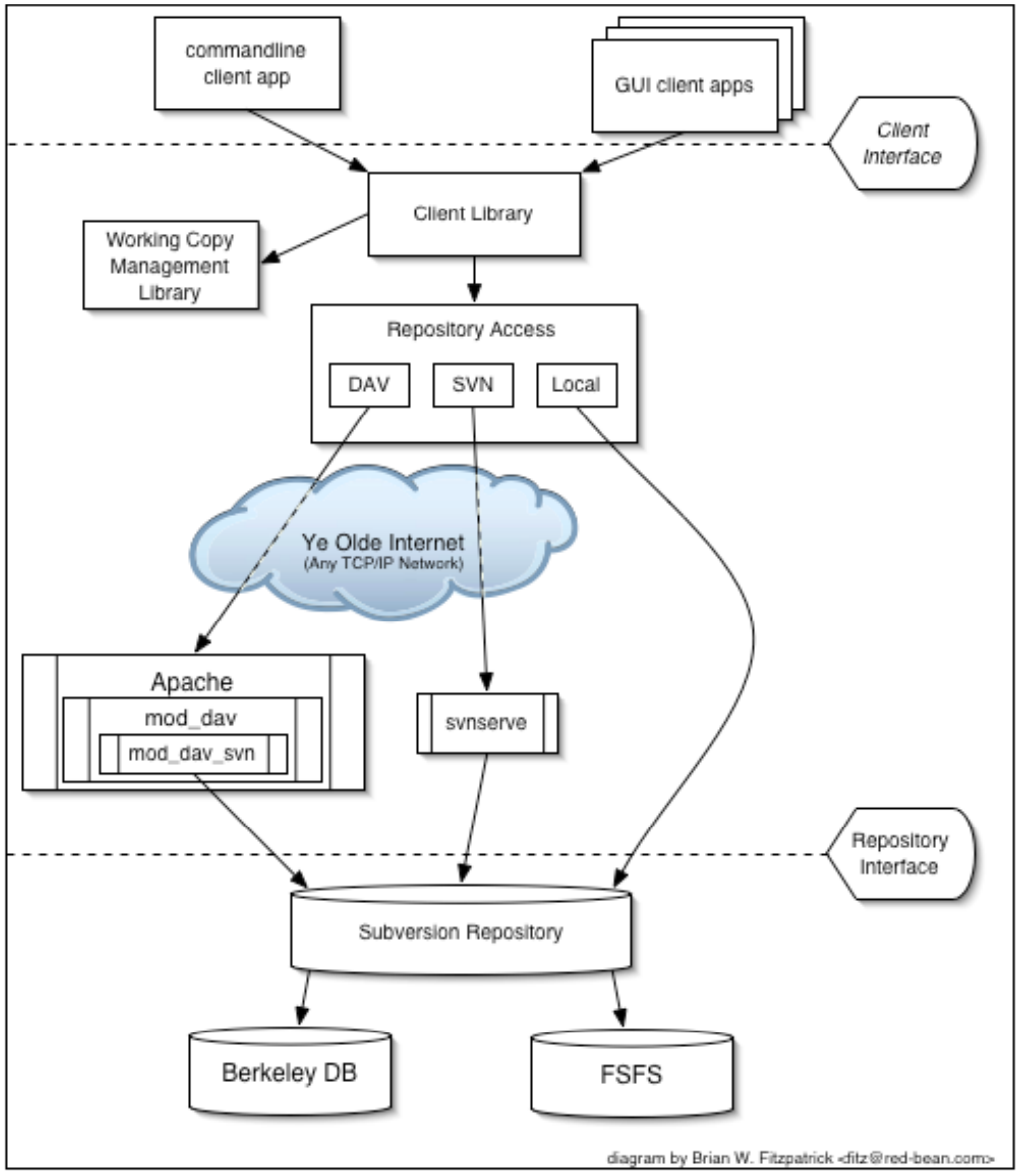
CVS

- Pros
  - folosit pe scară largă, bine documentat
  - suportat pretutindeni
- Cons
  - permite comiterea numai a fişierelor
  - Încet
  - adecvat pentru stocarea datelor text, necesită informaţii speciale pentru stocarea altor formate de fişiere

SVN

- Pros
  - mutarea atât a fişierelor, cât şi a directoarelor
  - overall revision number: versionare şi testare regresivă mai uşoară
  - previne comiterea accidentală a fişierelor aflate în conflict
  - suport pentru comanda diff
- Cons
  - necesită de două ori mai mult spaţiu decât CVS
  - no rollback of commit: e nevoie de stocarea unor stări bune a proiectului pentru a suprascrie comituri greşite

Arhitectura Subversion

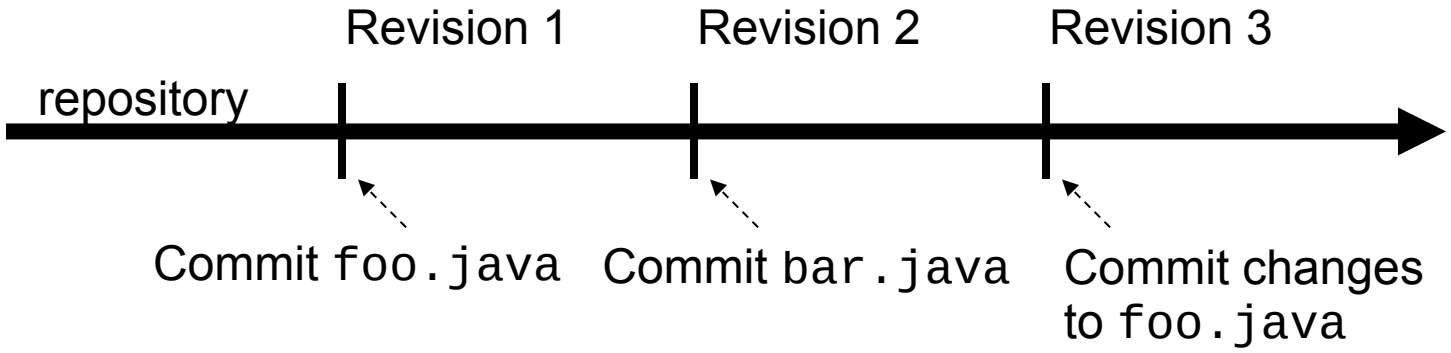


Elementele Controlului Versiunii

- Repository
  - Locul în care sunt păstrate şi menţinute proiectele versionate
  - NU SE LUCREAZĂ NICIODATĂ direct asupra fişierelor din repository
- Working Directory
  - Stochează o cope locală a proiectului
  - Interacţionează cu Repository prin
    - Fişierele pot fi **checked out** din repository şi se *lucrează asupra fişierelor locale* situate în *directorul de lucru*
    - Se pot **comite** modificările înapoi în repository atunci când modificările au fost efectuate complet

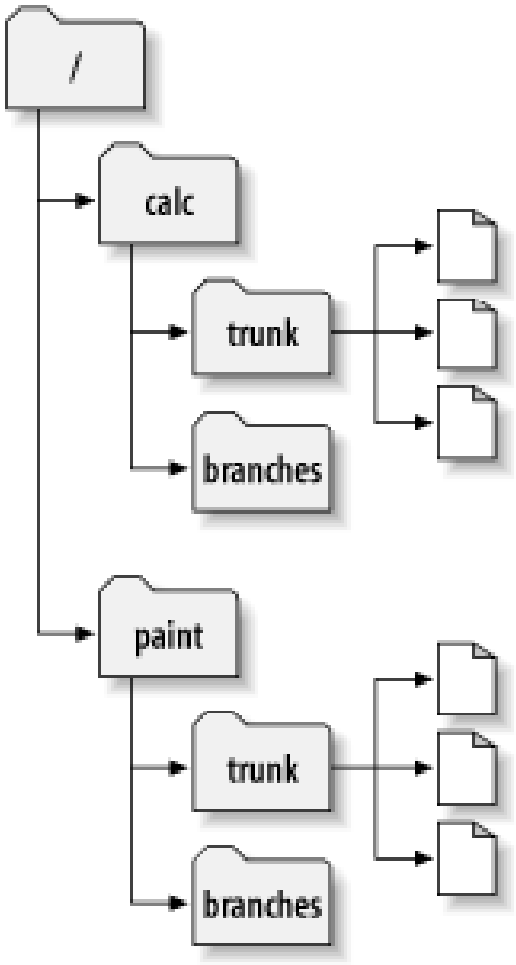
Elementele Controlului Versiunii

- Revizii
  - Sistemul de control al versiunii (VCS) stochează o istorie a modificărilor într-o **revizie** (**revision)** odată cu comiterea setului de modificări
  - Unele VCS-uri, ca şi Subversion, lucrează cu revizii ce sunt globale în întreg repository
  - Fiecare revizie este identificată printr-un număr unic de revizie



Proiectele

- /trunk
  - Păstrează “linia principală” a dezvoltării
- /tags
  - “Snapshot” al unui proiect în timp
- /branches
  - Conţine copii de tip branch
  - Copii de tip “ramură” al trunk
- Exemplu: client branch



Clientul SVN de tip Command Line

```
$ svn import /tmp/myproject
  file:///path/to/repos/myproject -m "initial import"
Adding /tmp/myproject/branches
Adding /tmp/myproject/tags
Adding /tmp/myproject/trunk
Adding /tmp/myproject/trunk/foo.c
Adding /tmp/myproject/trunk/bar.c
Adding /tmp/myproject/trunk/Makefile
…
Committed revision 1.
$
```

Setarea unui repository

- Se crează un nou repository
```
$ svnadmin create --fs-type fsfs /uac/gds/cprj/my_repository
```

- Se crează o structură de director pentru import
```
$ mkdir repos
$ mkdir repos/trunk
$ mkdir repos/branches
$ mkdir repos/tags
$ touch repos/trunk/hello.c
```

- Se importă directorul în repository
```
$ svn import --message "Initial import" repos file:///uac/gds/cprj/my_repository
```

- Se şterg fişierele originale
```
$ \rm –rf repos
```

- Repository-ul este accesat folosind următorul URL:
  - `file:///uac/gds/cprj/my_repository`
  - *protocol name:* `file://`
  - *UNIX path name*: `/uac/gds/cprj/my_repository`

- Pentru un repository în Windows:
  - `file:///c:/path/to/repository`

Repository Structure

- my_repository
  - trunk
  - branches
  - tags

Regăsirea fişierelor din Repository

- Se obţine o copie a celei mai recente revizii şi aceasta se stochează local într-un director de lucru local numit `my_repos_trunk`

```
$svn checkout
 file:///uac/gds/cprj/my_repository/trunk
 my_repos_trunk
```

```
$ svn <command> [<options>] [<targets>]
```

Cmd Option	Description
--message	Attach log to revision
--revision (-r)	Specify a specific revision
--username	Specify a username

Ciclul de bază al procesului de lucru

1. Actualizarea copiei locale

```
$ svn update
```

2. Modificări

```
$ svn add hello.c
$ svn delete hello.c
$ svn copy hello.c bye.c
$ svn move hello.c bye.c
```

Don't make changes using commands provided by your OS, use Subversion commands!
Otherwise, Subversion will not be aware of the change

3. Examinarea schimbărilor curente

```
$ svn status
$ svn diff hello.c bye.c
$ svn revert hello.c
```

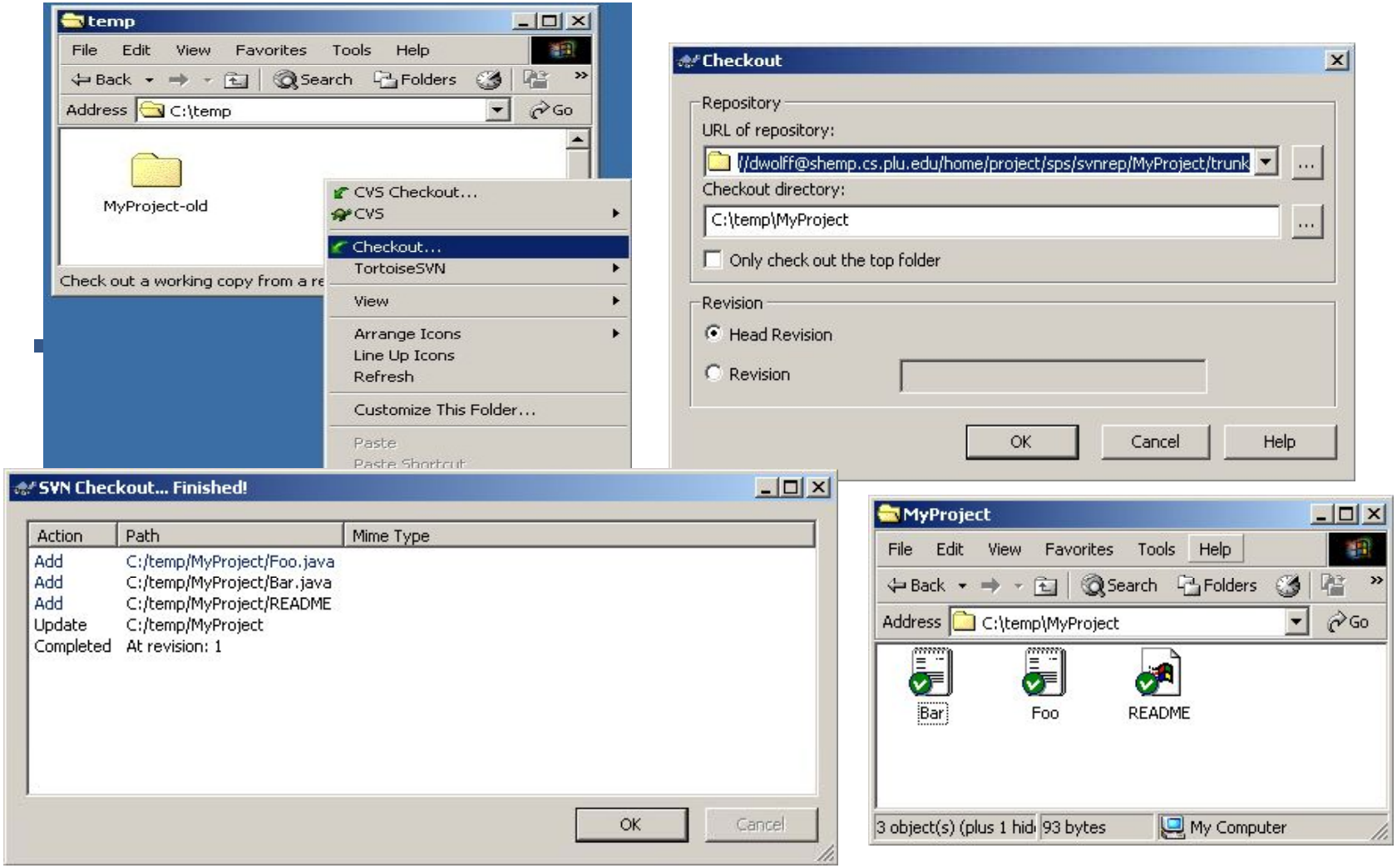
4. Merge între modificările efectuate de alţii şi copia locală

```
$ svn update
$ svn resolved hello.c
```

5. Comiterea propriilor modificări

```
$ svn commit
$ svn ci
```

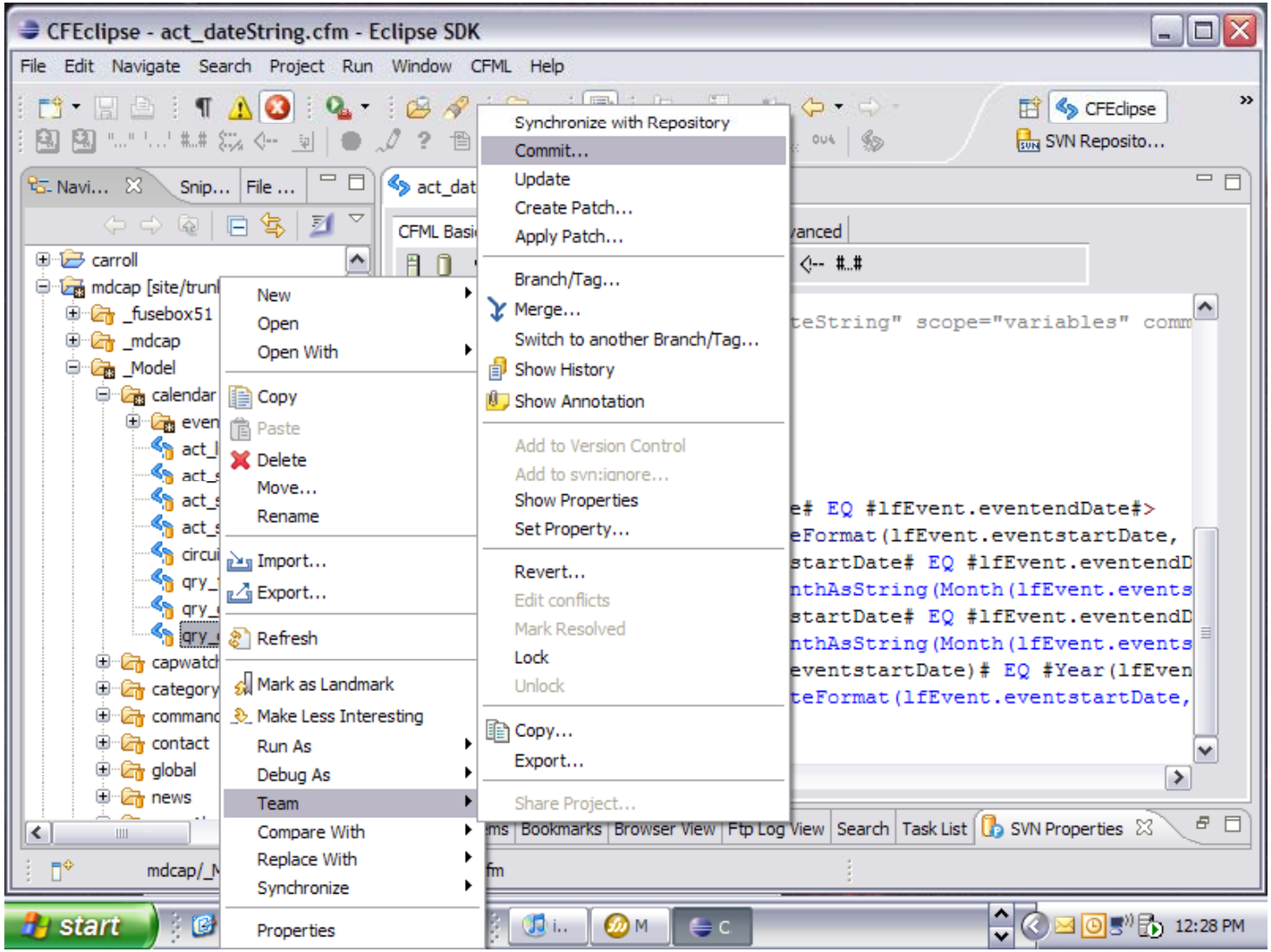
TortoiseSVN



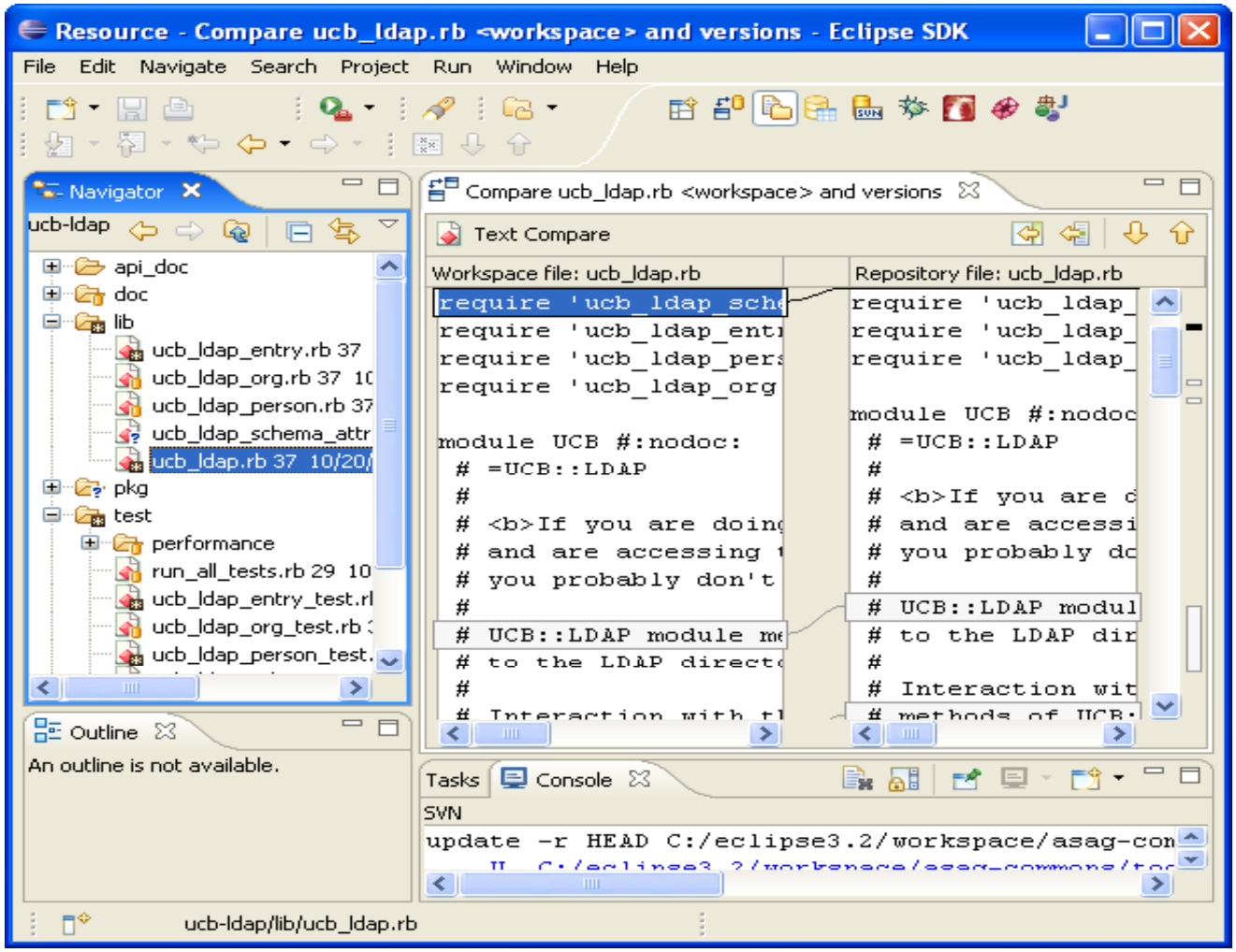
Subclipse

- Plug-in Eclipse
- Proiect open source

http://subclipse.tigris.org/

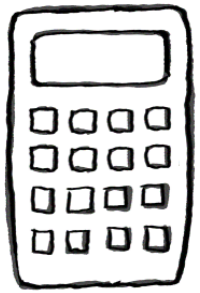


Subclipse

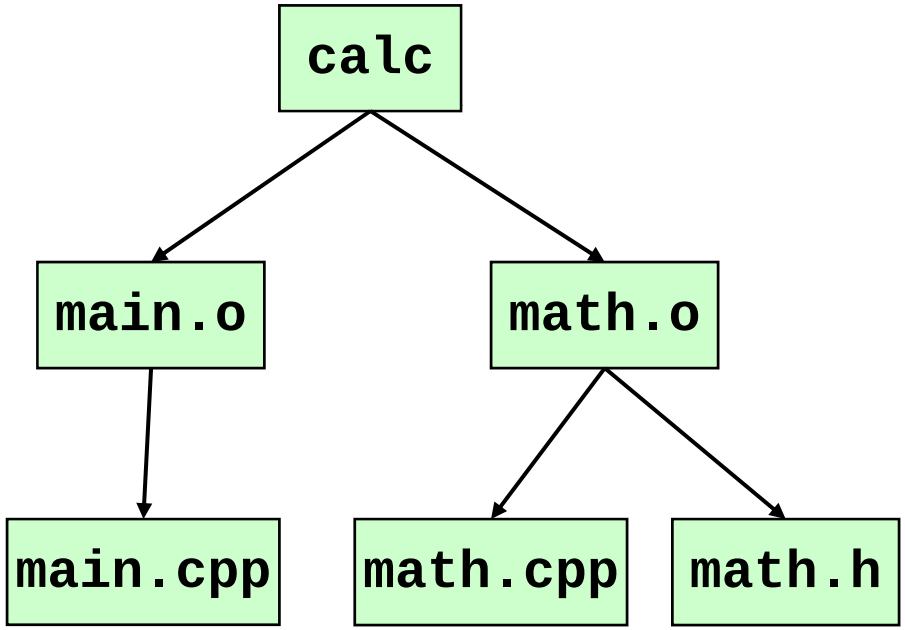

Resource - Compare ucb_ldap.rb <workspace> and versions - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Navigator
ucb-ldap
api_doc
doc
lib
ucb_ldap_entry.rb 37
ucb_ldap_org.rb 37 1(
ucb_ldap_person.rb 37
ucb_ldap_schema_attr
ucb_ldap.rb 37 10/20/
pkg
test
performance
run_all_tests.rb 29 10
ucb_ldap_entry_test.rl
ucb_ldap_org_test.rb
ucb_ldap_person_test.
Outline
An outline is not available.
Compare ucb_ldap.rb <workspace> and versions
Text Compare
Workspace file: ucb_ldap.rb
Repository file: ucb_ldap.rb
require 'ucb_ldap_sche
require 'ucb_ldap_enti
require 'ucb_ldap_pers
require 'ucb_ldap_org
module UCB #:nodoc:
=UCB::LDAP
#
<b>If you are doing
and are accessing t
you probably don't
#
UCB::LDAP module me
to the LDAP directo
#
Interaction with tl
require 'ucb_ldap_
require 'ucb_ldap_
require 'ucb_ldap_
module UCB #:nodoc
=UCB::LDAP
#
<b>If you are d
and are accessi
you probably do
#
UCB::LDAP modul
to the LDAP dir
#
Interaction wit
methods of UCB:
Tasks
Console
SVN
update -r HEAD C:/eclipse3.2/workspace/asag-con
ucb-ldap/lib/ucb_ldap.rb

Build Tools. Make

```
all: calc

calc: main.o math.o
   g++ main.o math.o -o calc

main.o: main.cpp
   g++ -c main.cpp

math.o: math.cpp math.h
   g++ -c math.cpp

clean:
   rm *.o
```

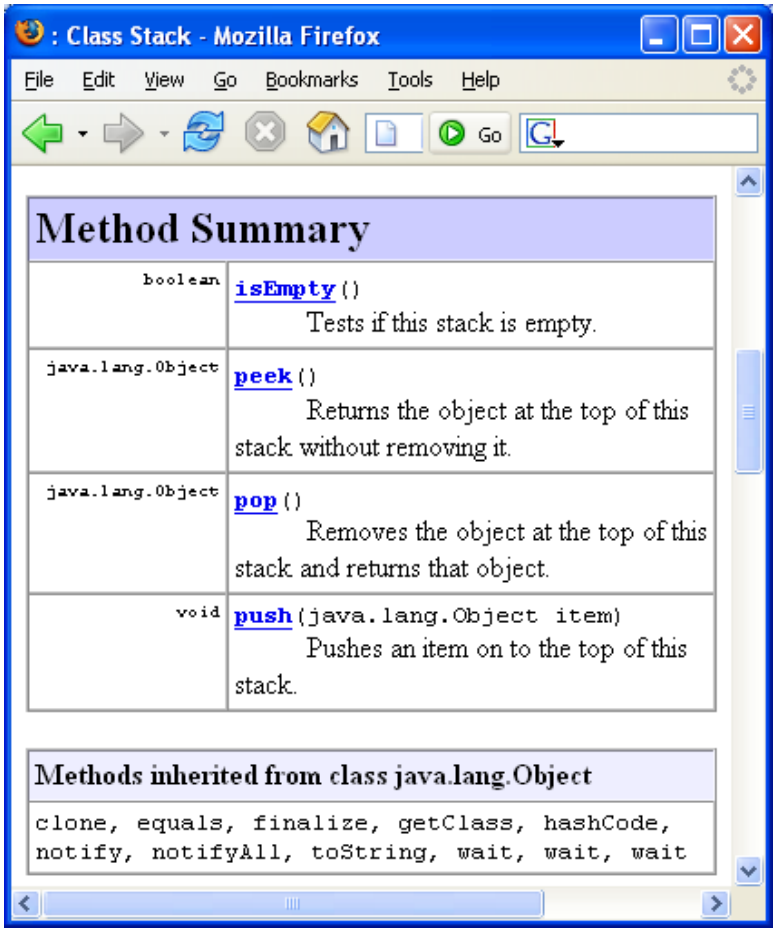


Graf de depedenţă

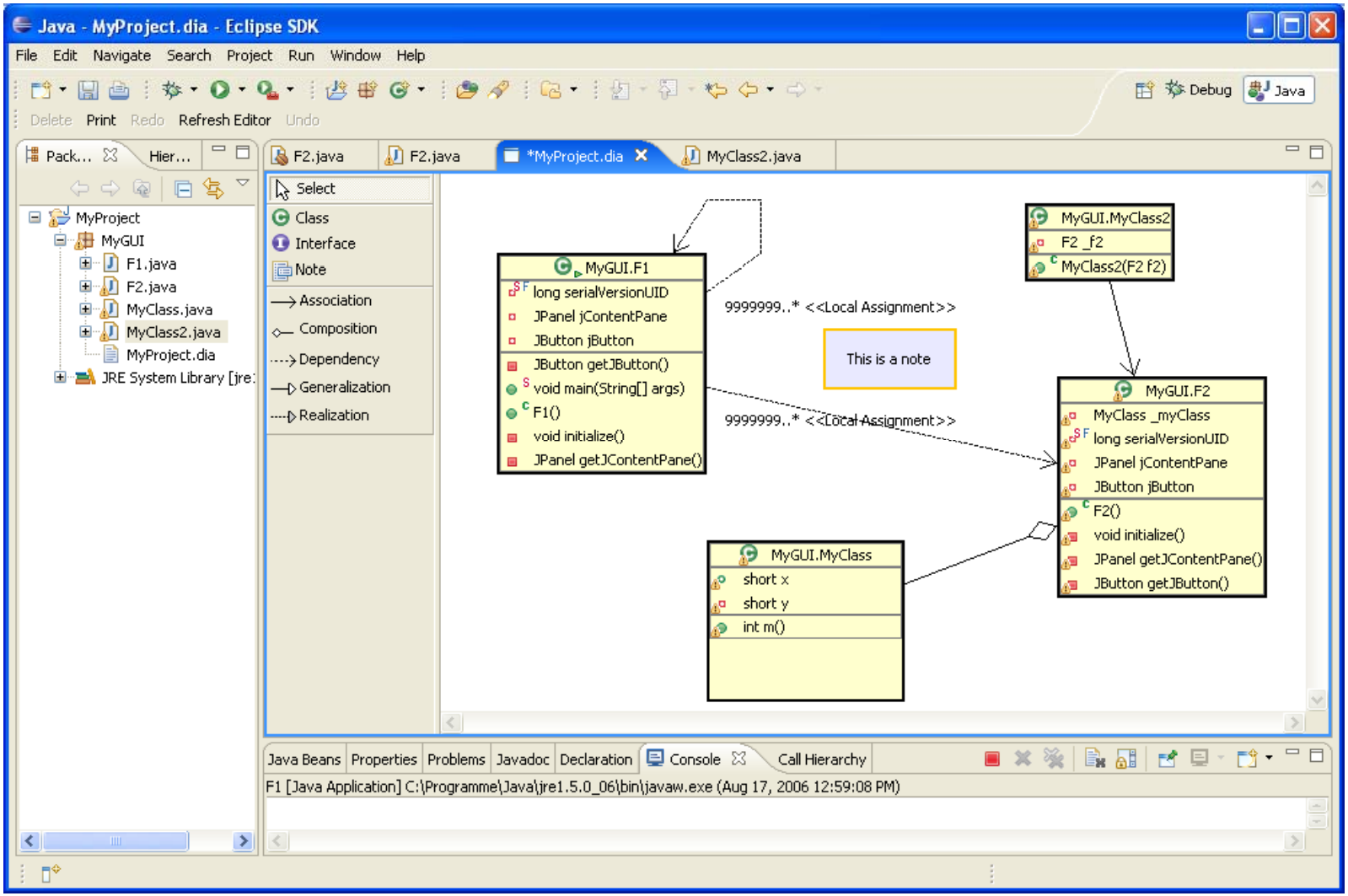


Instrumente de documentare. Javadoc

```
public class Stack {
  /**
   * Pushes an item on to
   * the top of this stack.
   * @param item the item to be pushed.
   */
  public void push(Objec item){
        this.elements.add(item);}

  /**
   * Removes the object at the top
   * of this stack and returns that
   * object.
   * @return The object at the top
   *         of this stack.
   * @exception NoSuchElementException
   *         if this stack is empty.
   */
  public Object pop()
  throws NoSuchElementException {
    // ...
} }
```

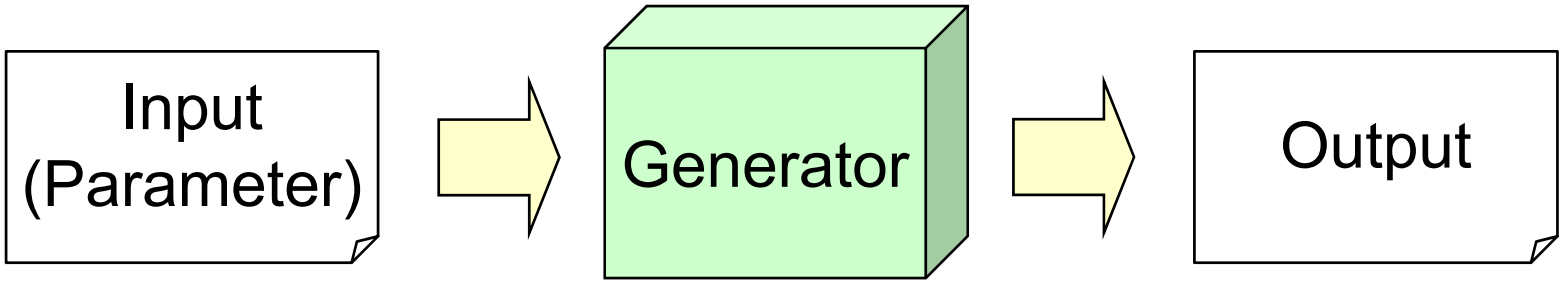


Instrumente de modelare



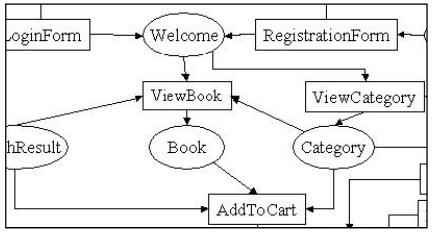
Generatoare (1)

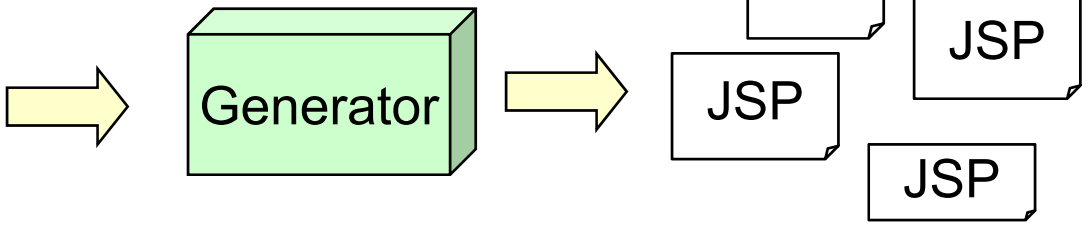
- Există taskuri de dezvoltare de “rutină”:
  - interfeţe (DB, GUI), unele pattern-uri de proiectare şi functionalităţi standard, etc.
- Idee: task-uri de automatizare folosind generatoare parametrizate
- Implică transformarea unor construcţii de nivel înalt în construcţii de nivel redus
- Avantaje:
  - reducerea timpului de dezvoltare
  - evitarea erorilor prin reducerea gradului de implicare a oamenilor acolo unde acest lucru nu este neapărat o necesitate



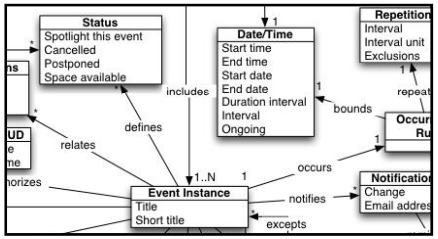
Generatoare (2)

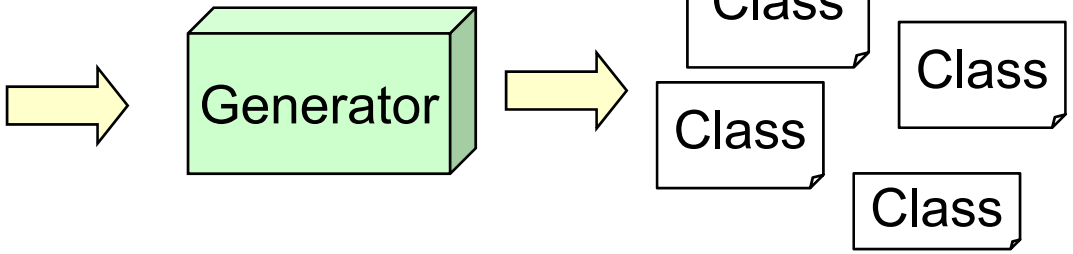
Generarea unei interfeţe web



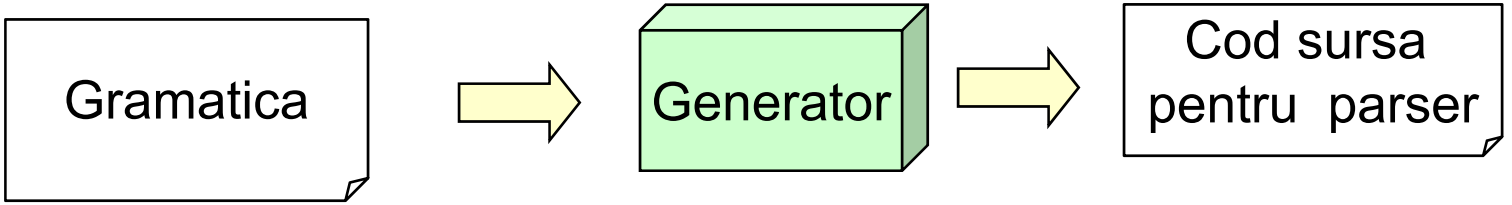


Generarea de clase Java



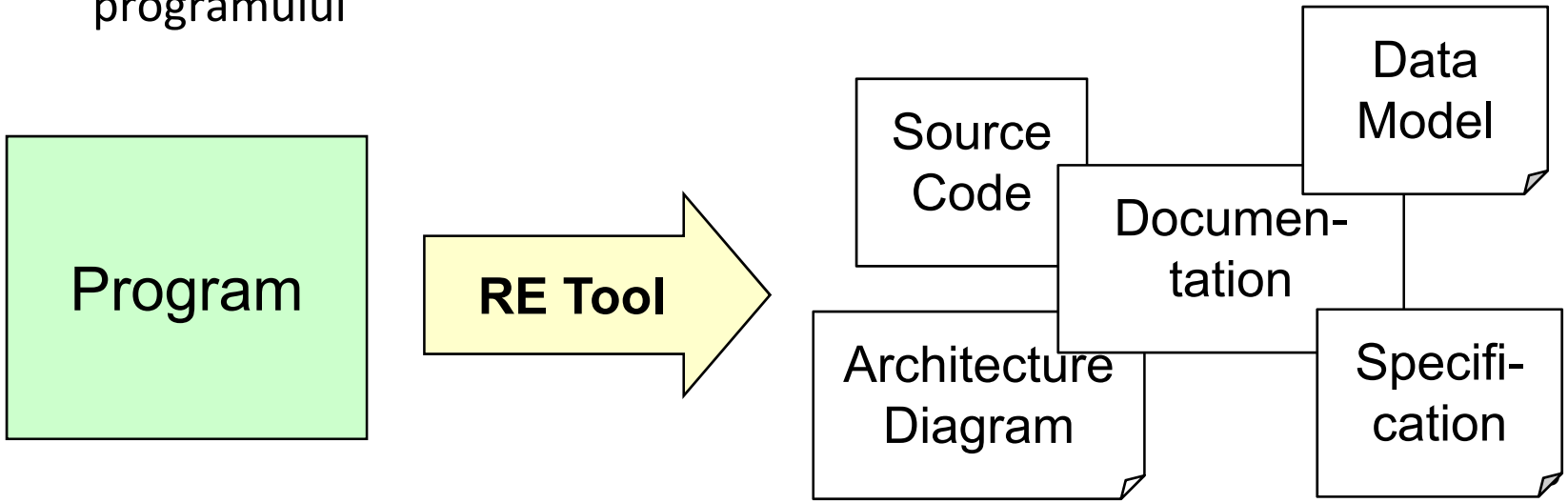


Generarea de parsere de limbaj



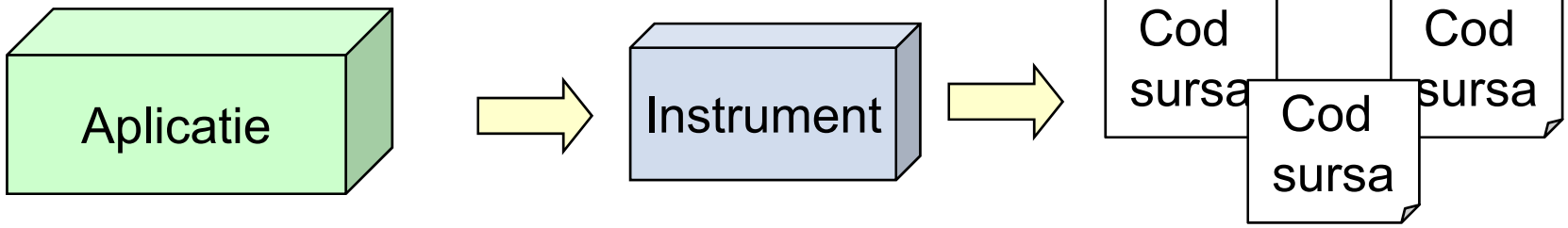
Reverse Engineering Tools (1)

- Regăsirea informaţiei plecând de la un sistem finalizat
- Metode:
  - Analiza statică: examinarea codului
  - Analiza dinamică: observarea comportamentului programului la runtime
  - Analiza black-box: observarea ieşirilor programului
  - Analiza white-box: black-box + observarea comportării interne a programului

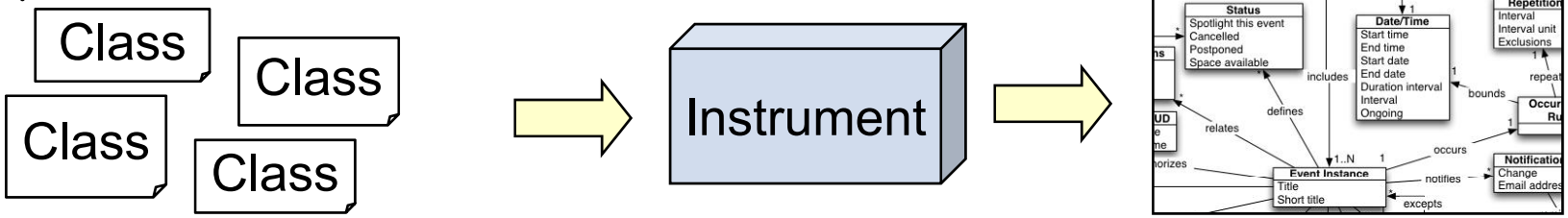


Reverse Engineering Tools (2)

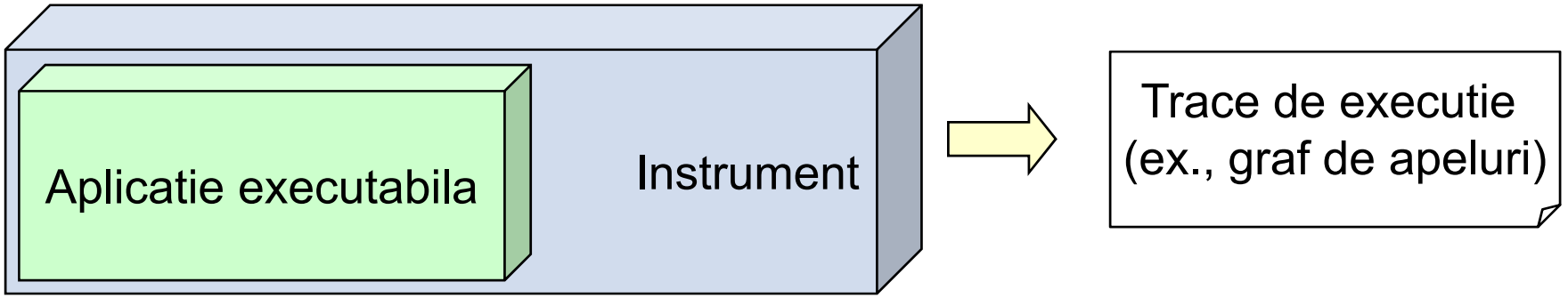
Recuperarea codului sursă



Recuperarea modelului de date



Explorarea comportamentului de runtime



Problemele CASE

- Tehnologia CASE a dus la îmbunătăţiri semnificative ale procesului software
- Dar nu de ordinul de mărime prezis:
  - Dezvoltarea software necesită creativitate – greu de automatizat
  - Ingineria dezvoltării este o activitate de grup – implică interacţiune în cazul unor proiecte de mare amploare
- Tehnologiile CASE suportă greu astfel de probleme

Productivitatea folosind CASE (1)

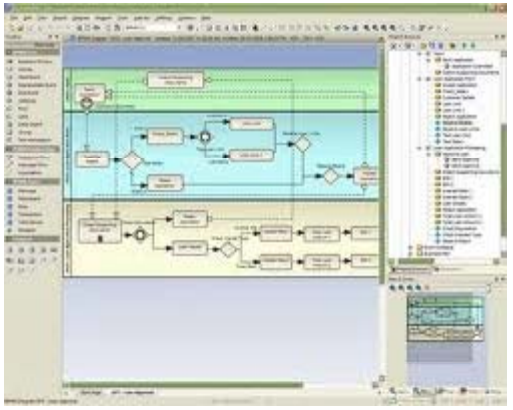
- Studiu realizat în 45 de companii (1995)
  - ½ sisteme informatice
  - ¼ software scientific
  - ¼ software real-time (pentru control aerian)
- Rezultatele:
  - Doar 10% câştig anual în productivitate
  - Costuri: 125.000$ per person/an

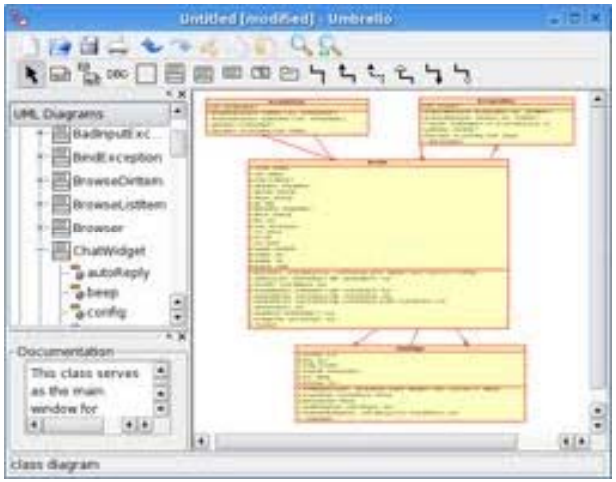
Productivitatea folosind CASE (2)

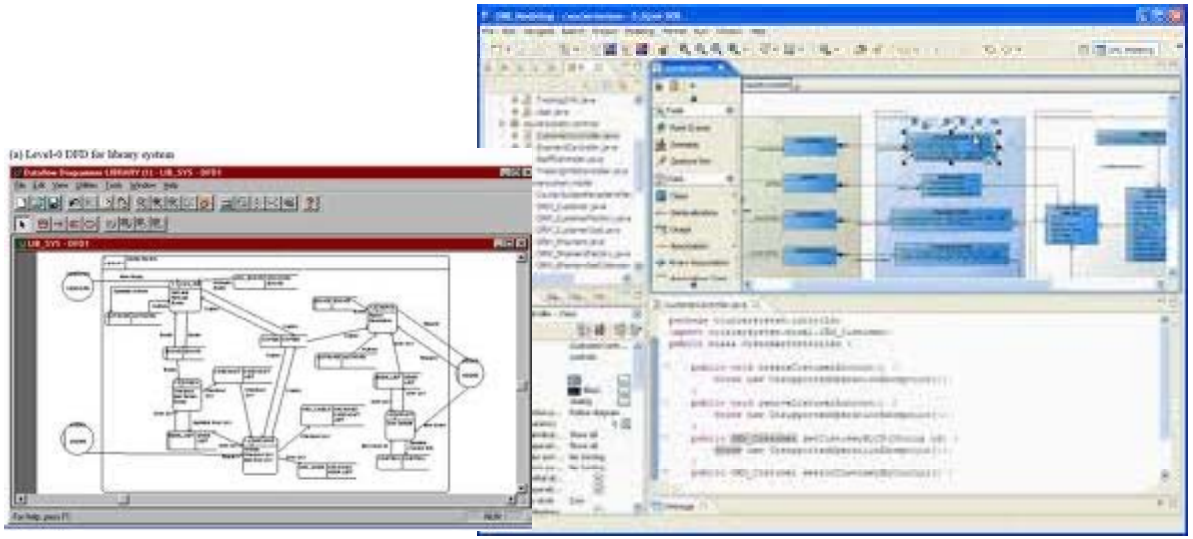
- Studiu condus pe 15 companii din Fortune 500 (1997)
- Este vital ca într-o companie să existe:
  - Training
  - Un proces software bine conturat
- Mediile CASE ar trebui folosite numai dacă există un nivel de maturitate în dezvoltarea procesului software suficient de matur

- "*A fool with a tool is still a fool*"

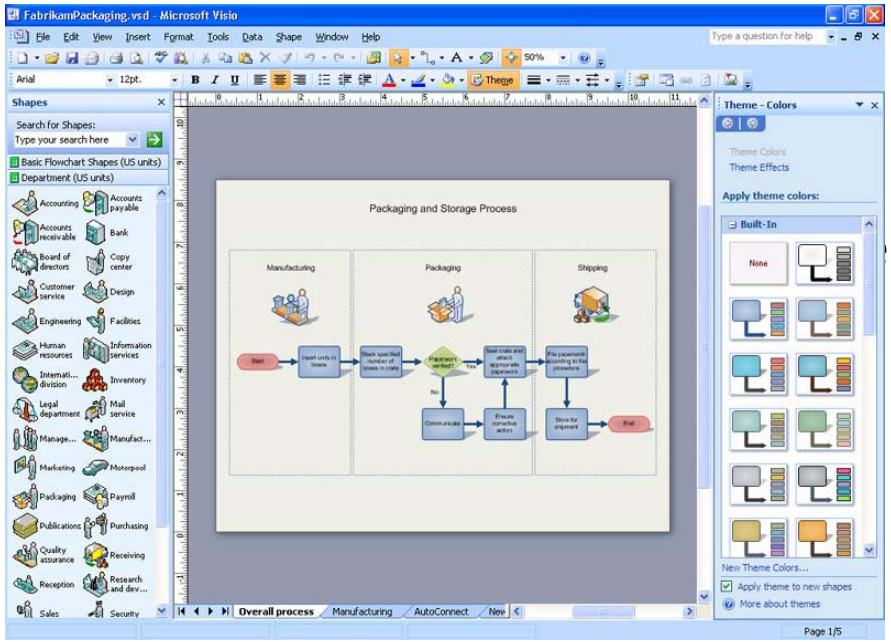
Exemple de produse CASE

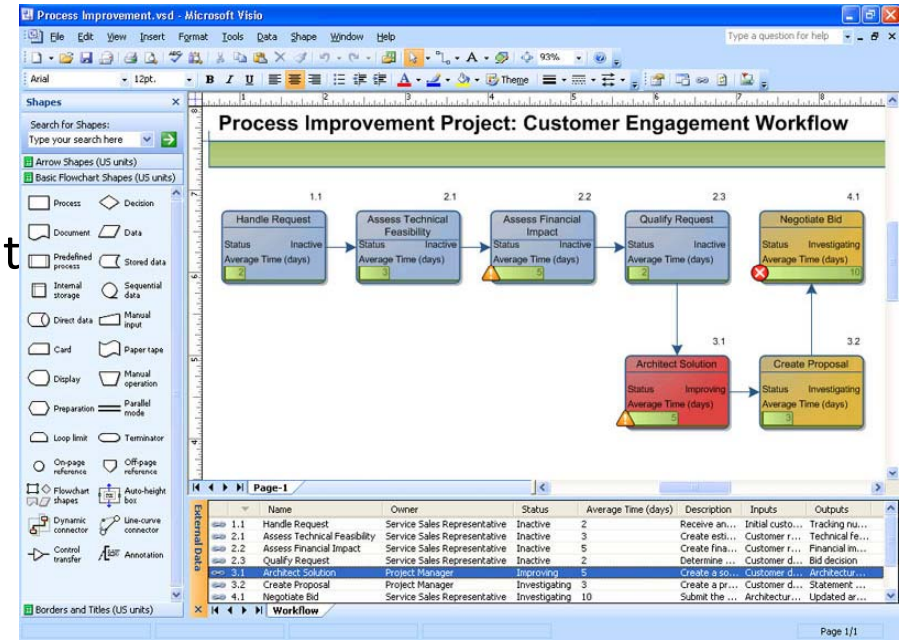






Microsoft Visio Professional





IBM Rational Unified Process (RUP)

- Concept de dezvoltare software (Rational Software Corporation) integrat cu unelte de dezvoltare software în suitele IBM Rational
- Este compus din:
  - Best practices - RUP include o bibliotecă de “best practices” pentru software engineering, acoperind de la managementul proiectului până la testarea detaliată a produselor.
  - Process delivery tools - RUP este livrat folosind tehnologii web ce permit integrarea cu alte instrumente de dezvoltare software.
  - Configuration tools - RUP este alcătuit din componente şi plug-inuri ce pot fi selectate şi configurare în funcţie de necesităţile fiecărui proiect.
  - Process authoring tools – O organizaţie poate extinde sau modifica instrumentele RUP prin crearea propriilor plug-inuri folosind produsele din suita Rational Process Workbench.
  - Community/Marketplace - Rational Developer Network (RDN) furnizează un loc pentru partajarea diverselor extensii de procese software.

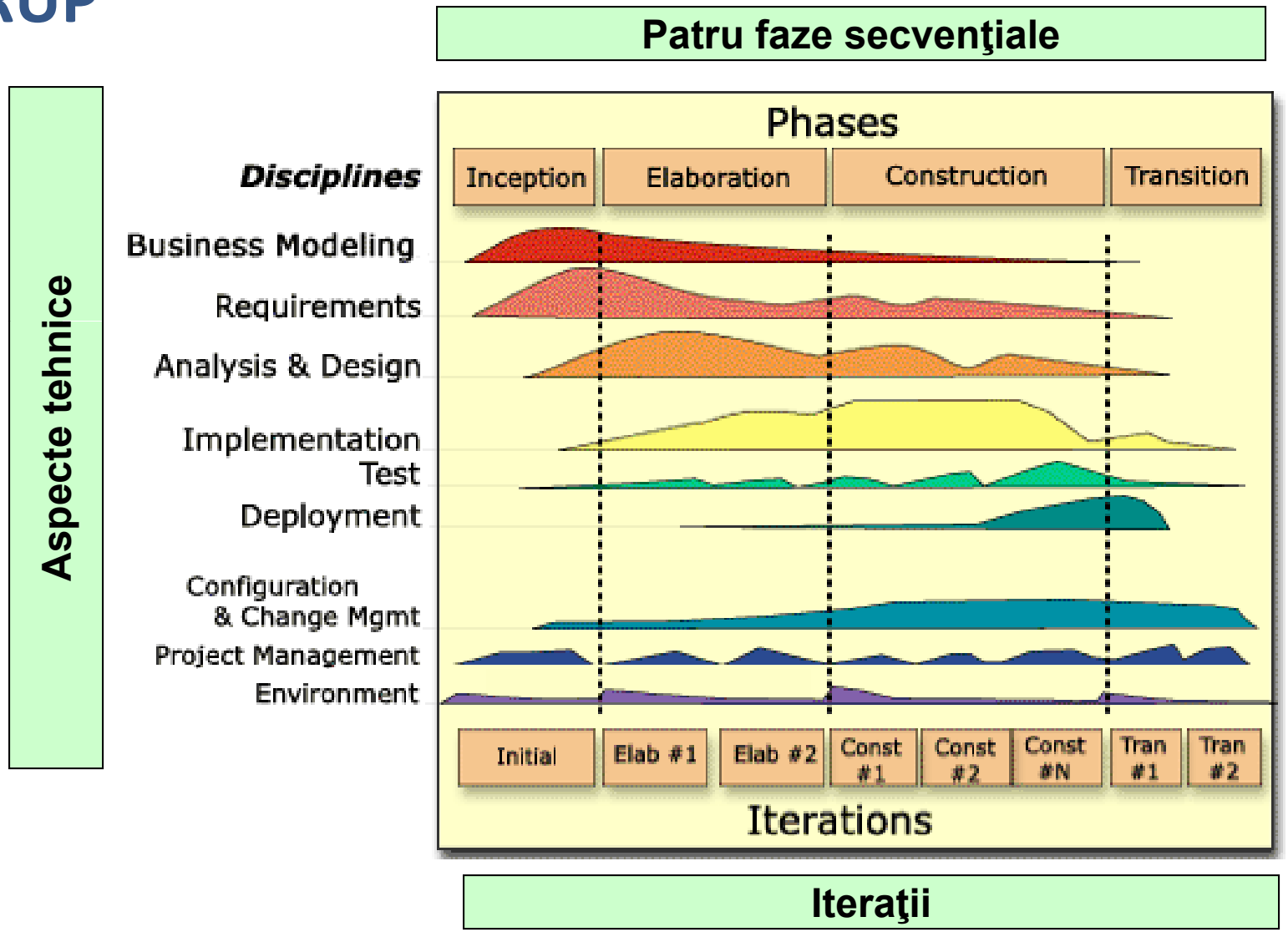
Framework


Configuration Tools
Process Delivery Tools
Rational Unified Process®
Process Authoring Tools
Community/ Marketplace

Ciclul de dezvoltare RUP

- Alcătuit din patru faze secvenţiale ce modelează aspecte financiare, strategice, comerciale şi umane din derularea proiectului software.
- Nouă activităţi ce modelează aspectele tehnice ale dezvoltării proiectului: modelarea proceselor de business, implementare, testare, etc.
- O fază a unui proiect RUP este împărţită în iteraţii
  - cuprind activităţi de dezvoltare ce produc *releases* a software-ului final executabil.

RUP


Patru faze secvenţiale
Aspecte tehnice
Disciplines
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration
& Change Mgmt
Project Management
Environment
Phases
Inception
Elaboration
Construction
Transition
Initial
Elab #1
Elab #2
Const #1
Const #2
Const #N
Tran #1
Tran #2
Iterations
Iteraţii

Componente ale activităților RUP

Un *rol* presupune:

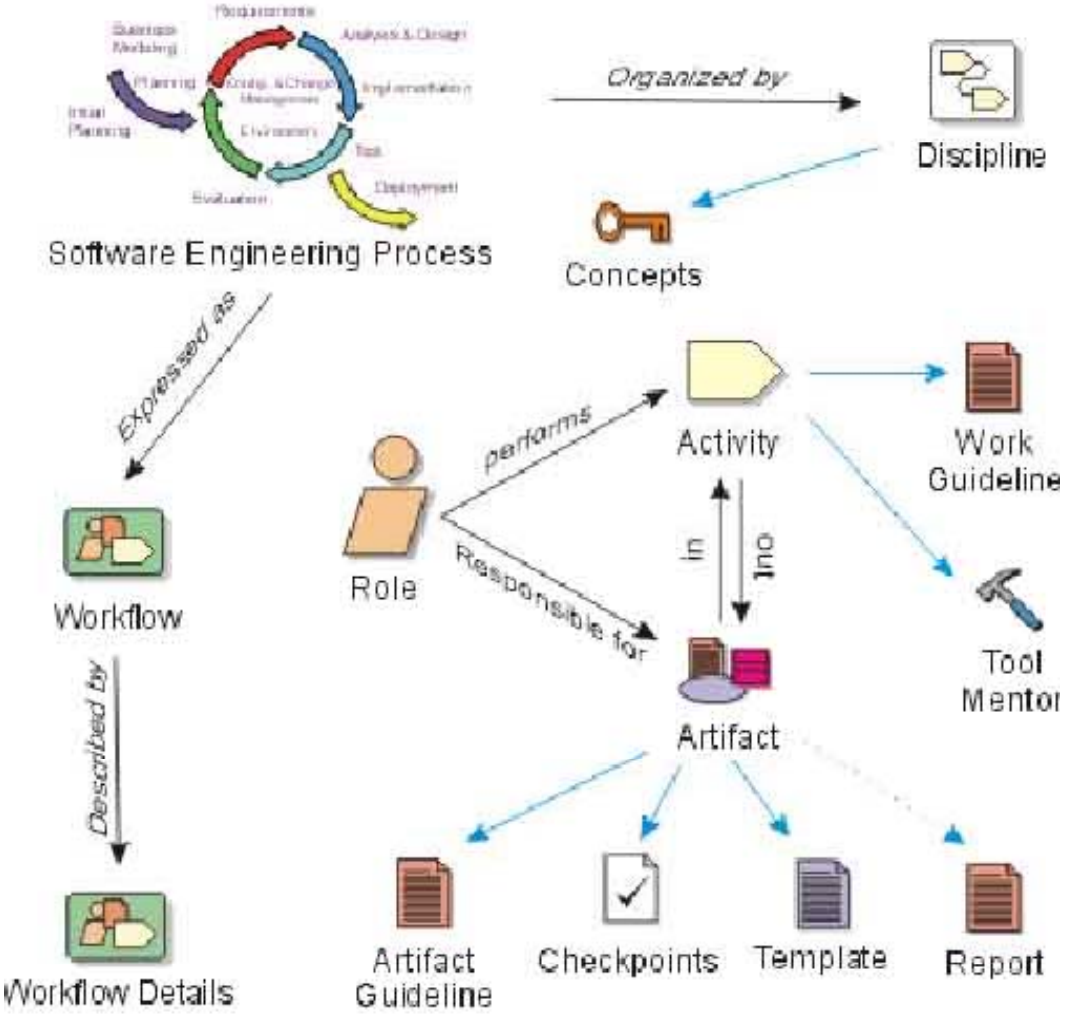
-Set de activităţi

-Set de produse livrate (artifacts)

Artifact guidelines arată cum sunt dezvoltate, evaluate şi folosite produsele

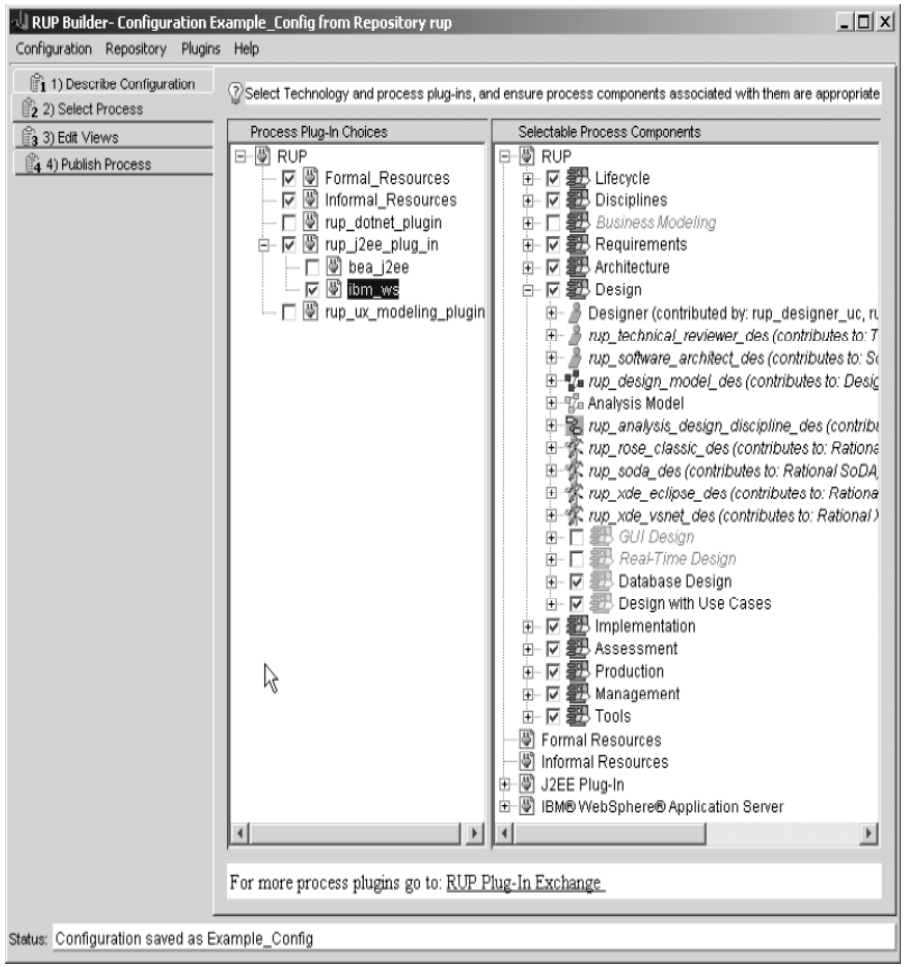
Work guidelines explică cum trebuie implementată o activitate

Tool mentor descrie cum pot ajuta instrumentele software fiecare dintre actvităţi



Selectarea Componentelor şi Plug-ins Constructor RUP (RUP Builder)

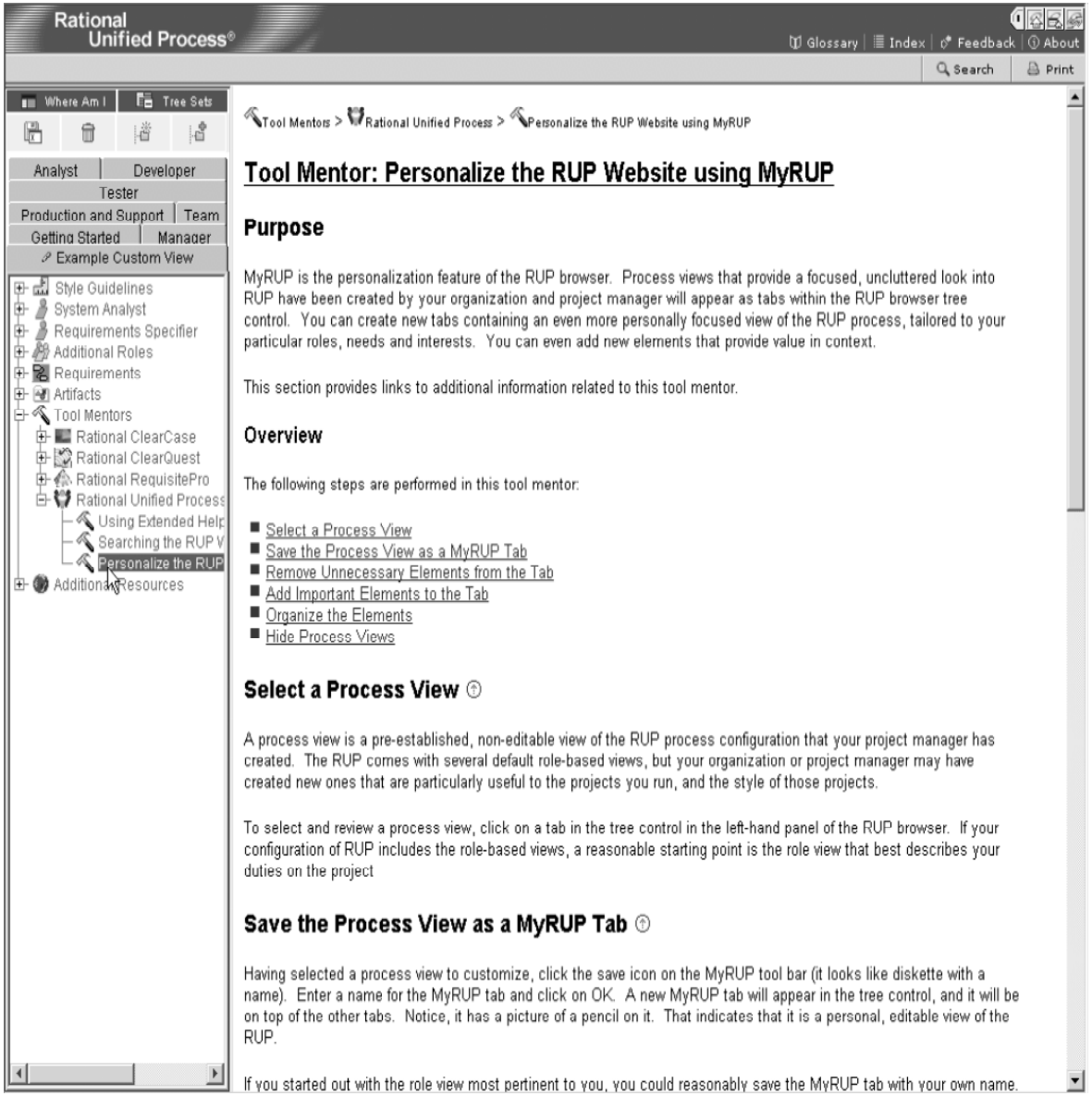
- Patru etape
- Ajută la:
  - selectarea unei configuraţii de bază,
  - folosirea plug-in-urilor în alte componente ale procesului,
  - configurarea viziunilor procesului,
  - publicarea RUP-ului personalizat



Definirea Vederilor de Proces

- Process View reprezintă un control bazat pe roluri personalizat ce conţine legături către elemente din RUP Process Configuration, precum şi legături către fişiere sau alte URL-uri externe configuraţiei.
- Process Views sunt create în RUP Builder.
- MyRUP permite fiecărui individ să-şi contureze propria sa viziune asupra proiectului
- Asistenţi de instrument (*Tools Mentors*) - oferă asistenţă pentru utilizarea instrumentelor Rational în scopul îndeplinirii sarcinilor RUP.

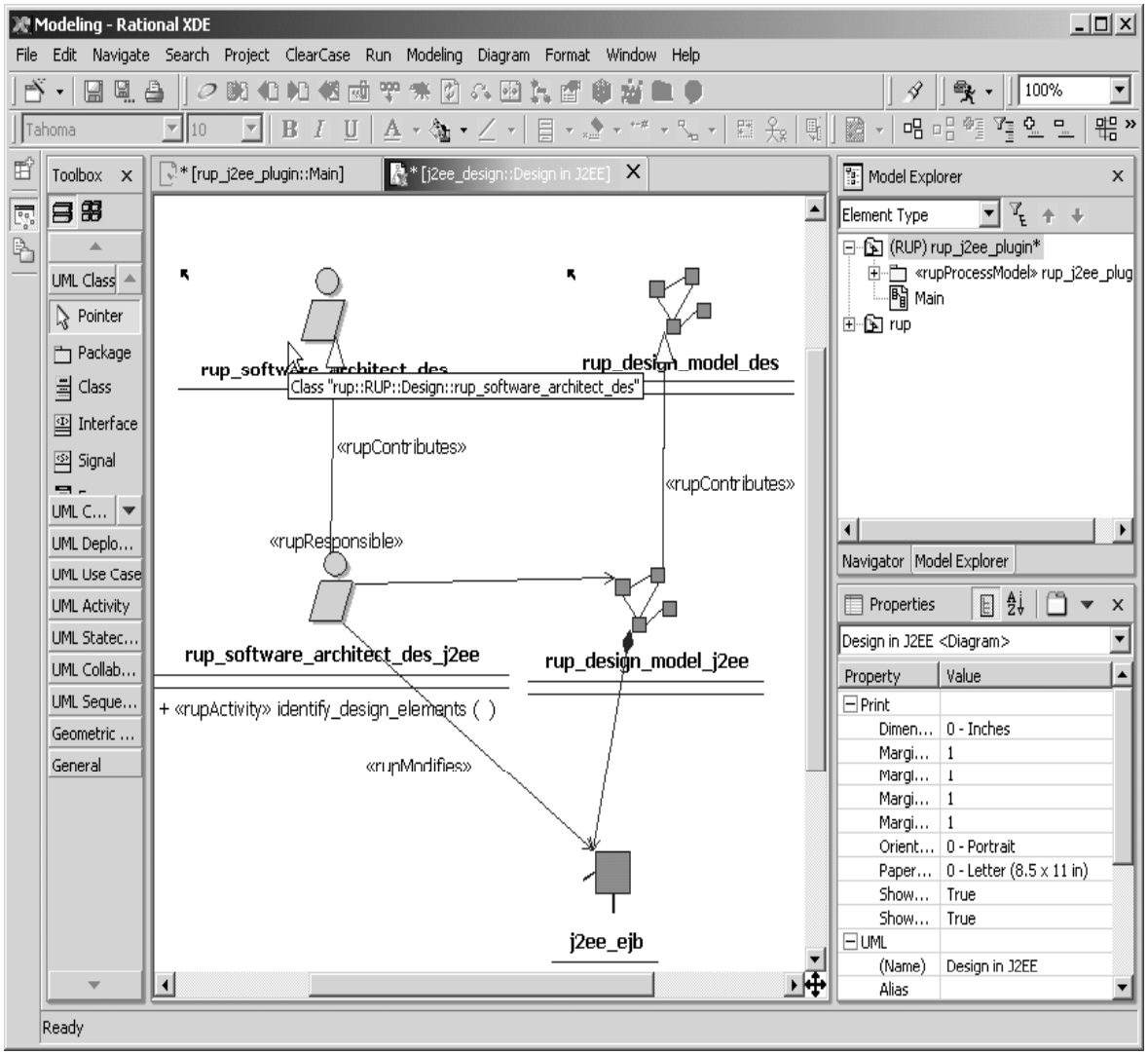
Process view



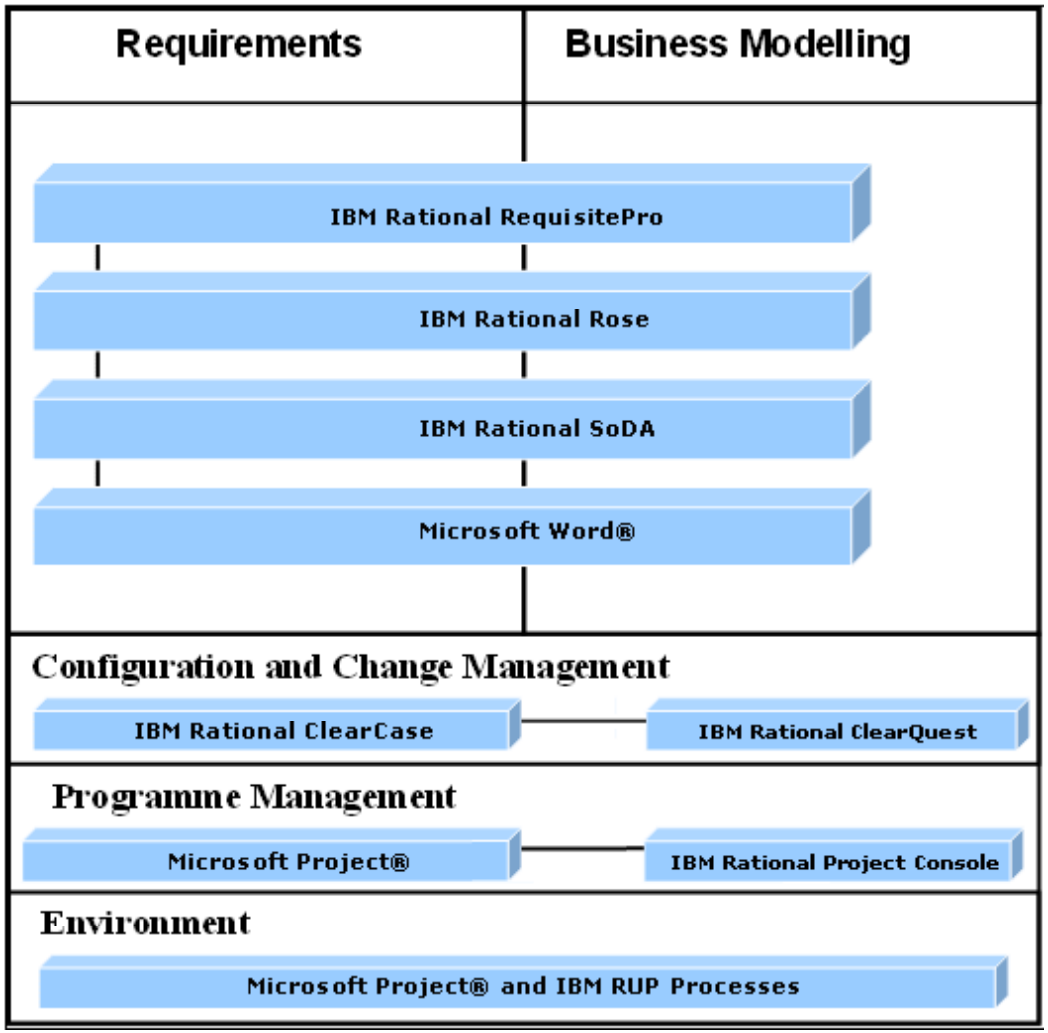
Tool Mentor

RUP Modeler

- RUP Modeler instrument de modelare ce permite modificarea proceselor RUP



Instrumente din suita IBM Rational



Instrumente din suita IBM Rational

- **IBM Rational® Requisite Pro®**
- Furnizează o bază de date centralizată a tuturor cerinţelor programelor dezvoltate.
- Este integrat cu Microsoft Word, aspect ce permite importarea tuturor cerinţelor descrise în documentul cerinţelor direct în baza de date
- Beneficii :
  - Ajută la vizualizarea de către echipă a proceselor de business.
  - Îmbunătăţesc comunicarea între acţionari şi clarificarea cerinţelor de business.
  - Ajută la gestionarea complexităţii.
  - Ajută la capturarea fenomenelor de business vitale pentru produsul final şi înţelegerea lor de către echipele de dezvoltare.

- **Familia de produse IBM Rational Rose®**
- Permite analiştilor de business modelarea proceselor de business, folosind Unified Modelling Language (UML)
- Beneficii:
  - Reducerea timpului şi efortului necesar în producerea documentaţiei aferente dezvoltării programului.
  - Producerea de rapoarte la termen, actualizate şi consistente privind datele activităţilor implicate de procesul de dezvoltare.
  - Oferă capabilităţi de publicare online (Web) a documentelor şi rapoartelor.

Instrumente din suita IBM Rational

- **IBM Rational ClearCase®**
- Gestionează modificările apărute în cadrul produselor dezvoltare şi furnizează un control automatizat al diverselor versiuni software.
- Beneficii:
  - Gestiunea automată a modificărilor.
  - Protejarea integrităţii produselor dezvoltate.

- **IBM Rational ClearQuest®**
- Automatizează procesele aferente managementului problemelor, riscurilor şi modificărilor.
- Beneficii:
  - Salvarea timpului şi efortului şi îmbunătăţirea preciziei managementului prin automatizarea proceselor de modificare manuală a diverselor documente aferente activităţilor de dezvoltare a produsului software.
  - Furnizează rapoarte asupra managementului problemelor, riscurilor şi modificărilor apărute.

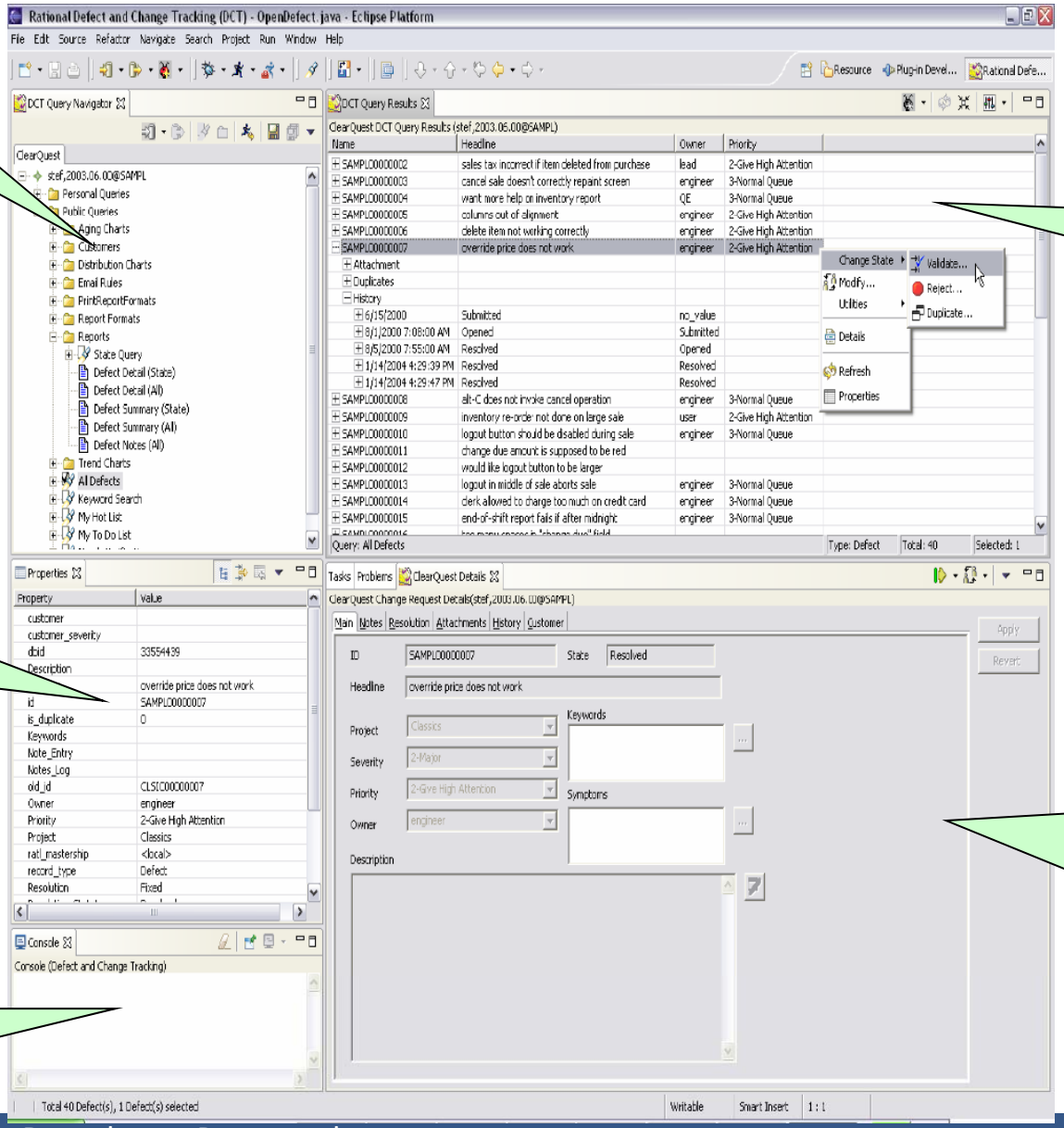
- **IBM Rational Project Console®**
- Automatizează procesul de cercetare şi raportare a stării curente a dezvoltării produsului software. Include metrici asupra progresului în format Web şi poate fi adaptat pentru a captura diverse metrici privind beneficiile curente ale dezvoltării.
- Beneficii:
  - Salvarea timpului necesar creării, construirii şi menţinerii unui site Web privind progresul curent al procesului de dezvoltare software.
  - Salvarea timpului şi efortului presupus de colectarea manuală a stării diverselor activităţi de dezvoltare.
  - Furnizează un punct unic de depozit a informaţiilor actualizate privind starea proiectului ce poate fi folosit de către toţi membrii echipei de dezvoltare.

Rational ClearQuest Client for WebSphereStudio

Workspace view

Properties view

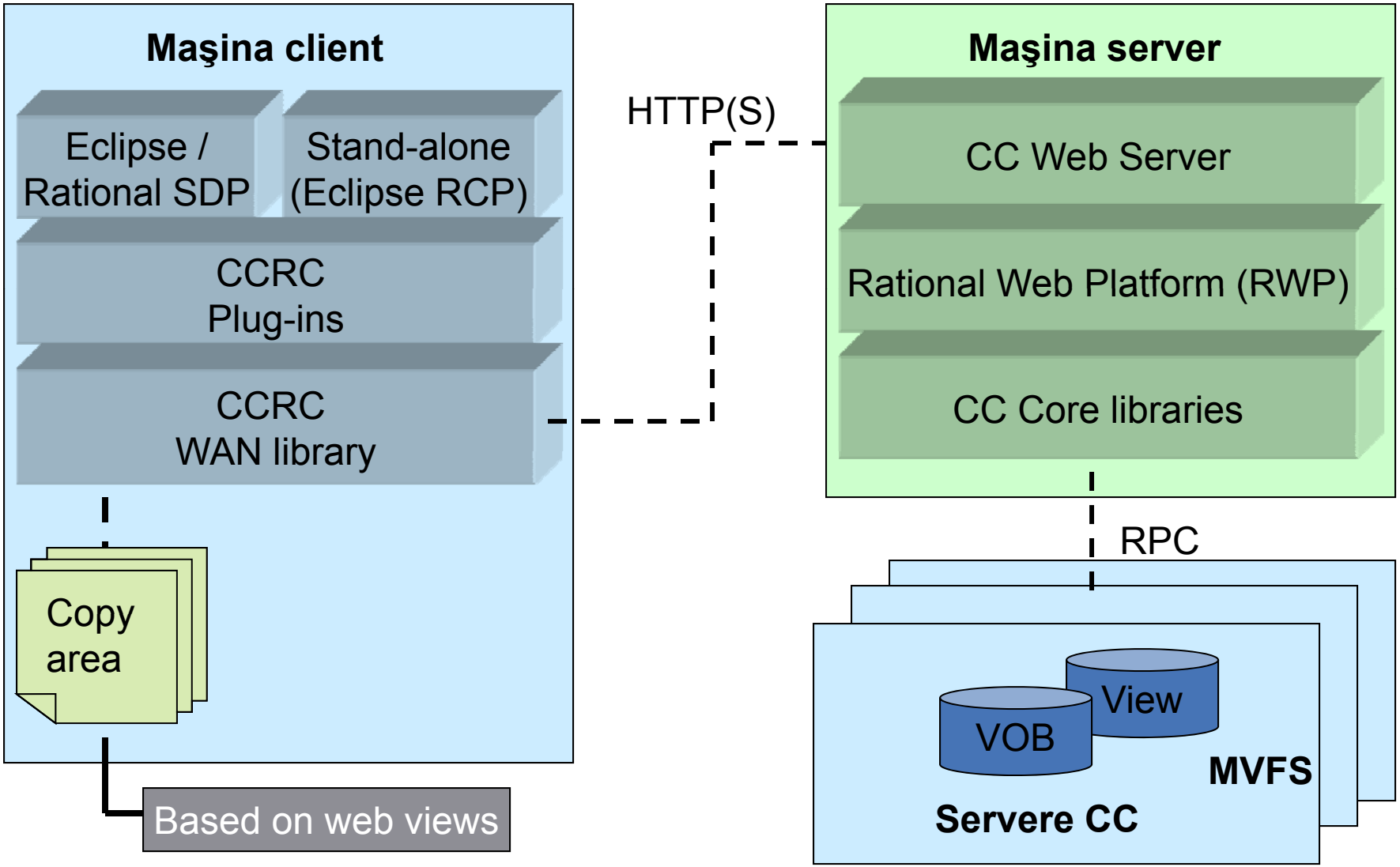
Console view



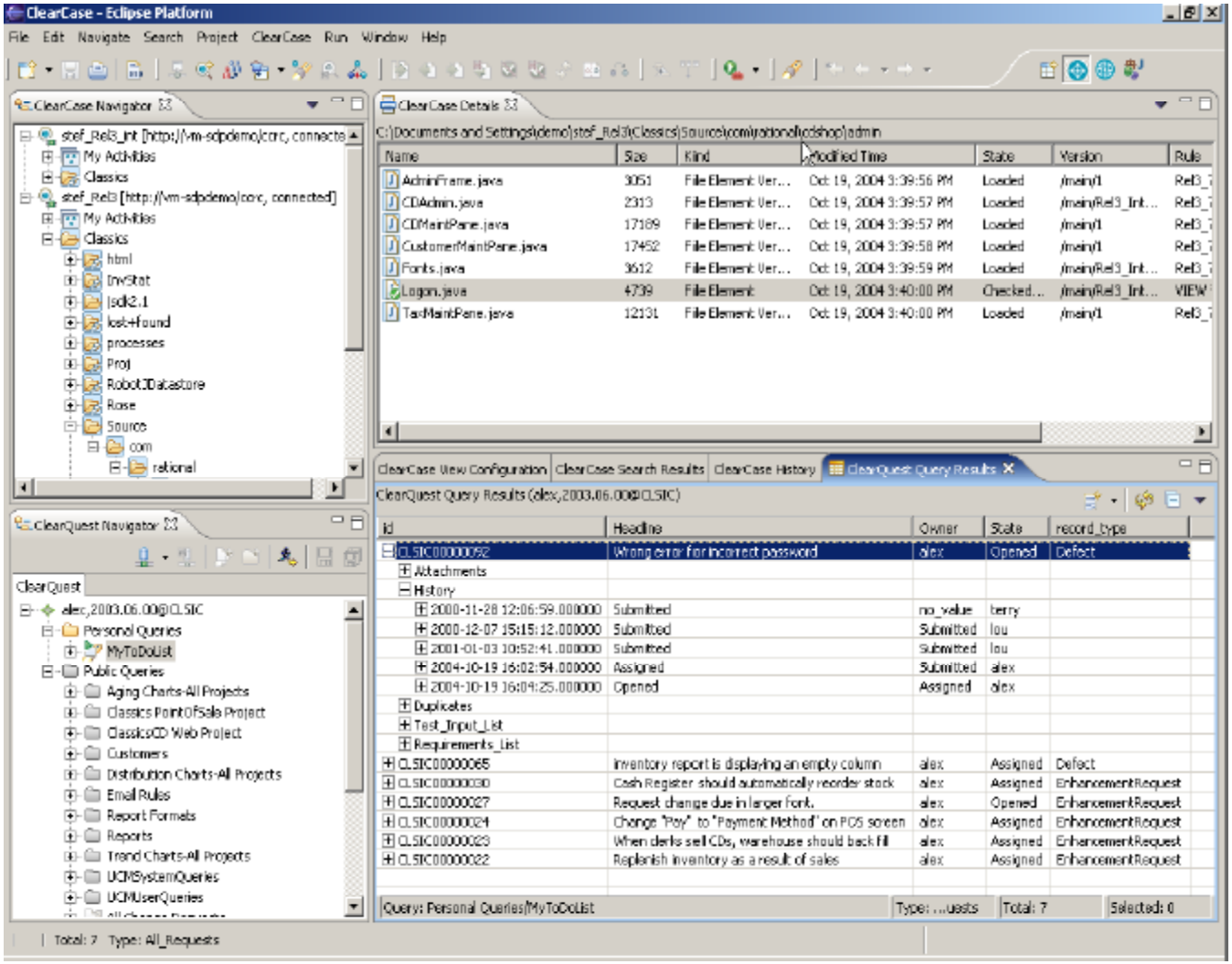
Result Set view

Record Details, Chart and Report views

Arhitectura ClearCase Remote Client

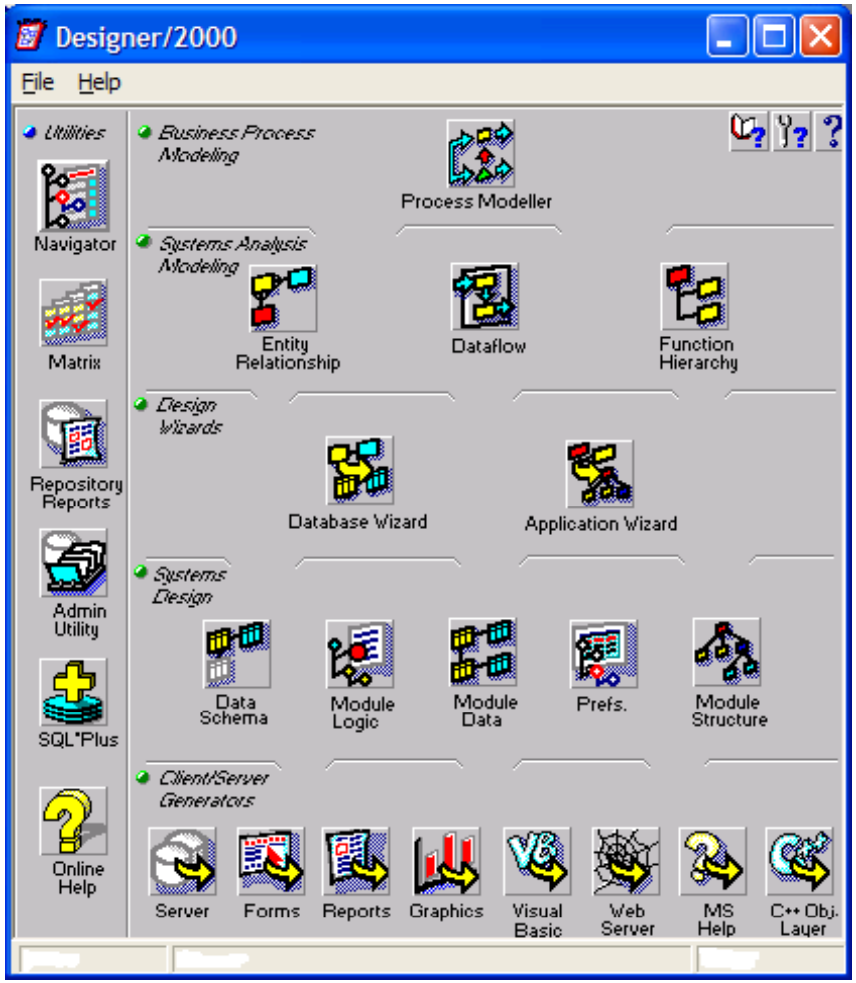


ClearCase Remote Client for Eclipse



Designer 2000

- Produs al firmei Oracle



Sumar

- Instrumente CASE: definiţie, proprietăţi
- Clasificarea instrumentelor CASE
- Arhitectura generică a unui mediu CASE
- Studiu de caz: instrumente CASE pentru etapele de analiză şi proiectare
- Exemple de instrumente CASE: Version Control Systems (SVN)
- Exemple de produse CASE: Microsoft Visio Professional, IBM Rational Unified Process