Ontwerp van SoftwareSystemen

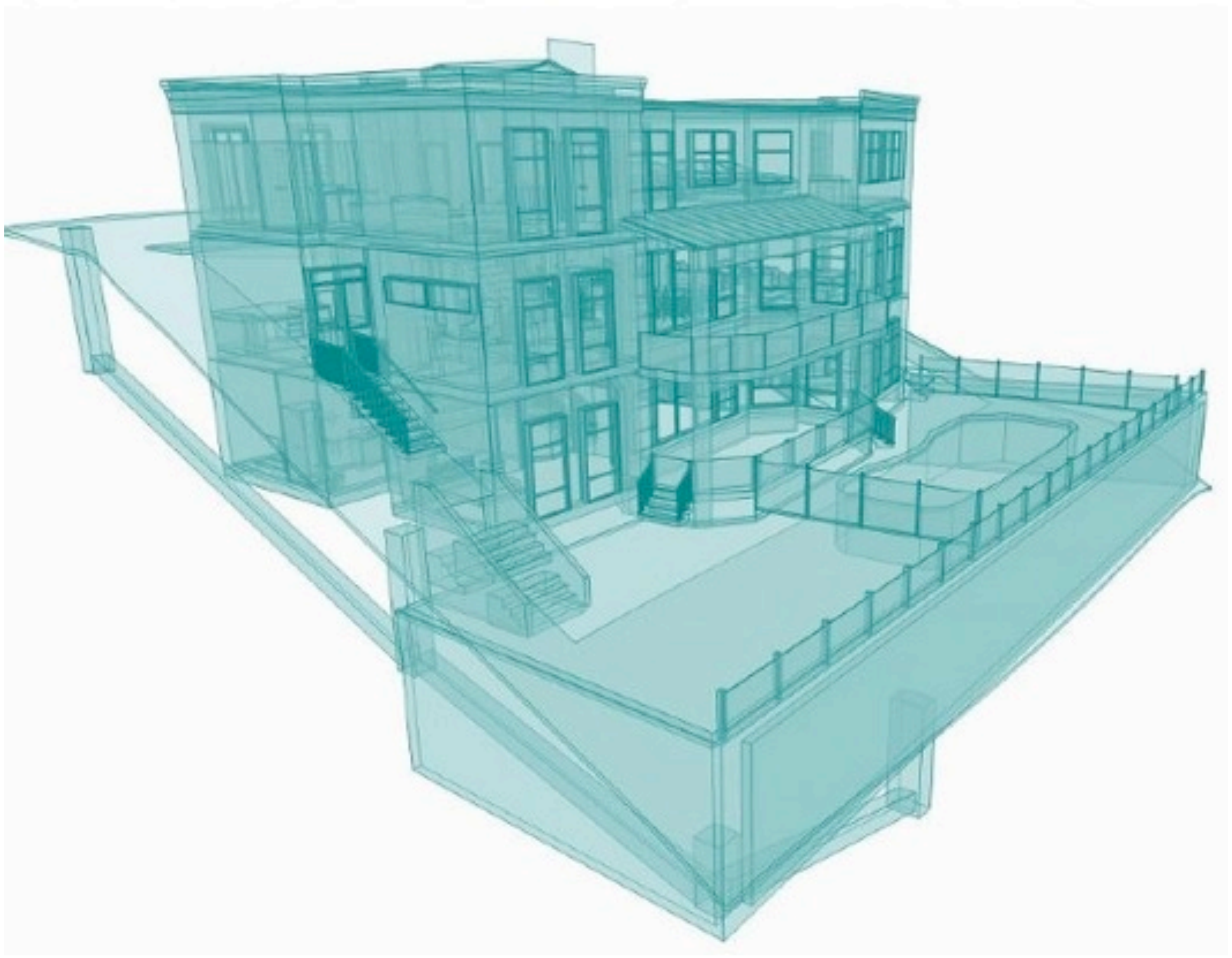
7 On Multi-user Development Tools,Versioning and Packaging of code, their Relations, the Universe and Everything.

Roel Wuyts
OSS 2013-2014

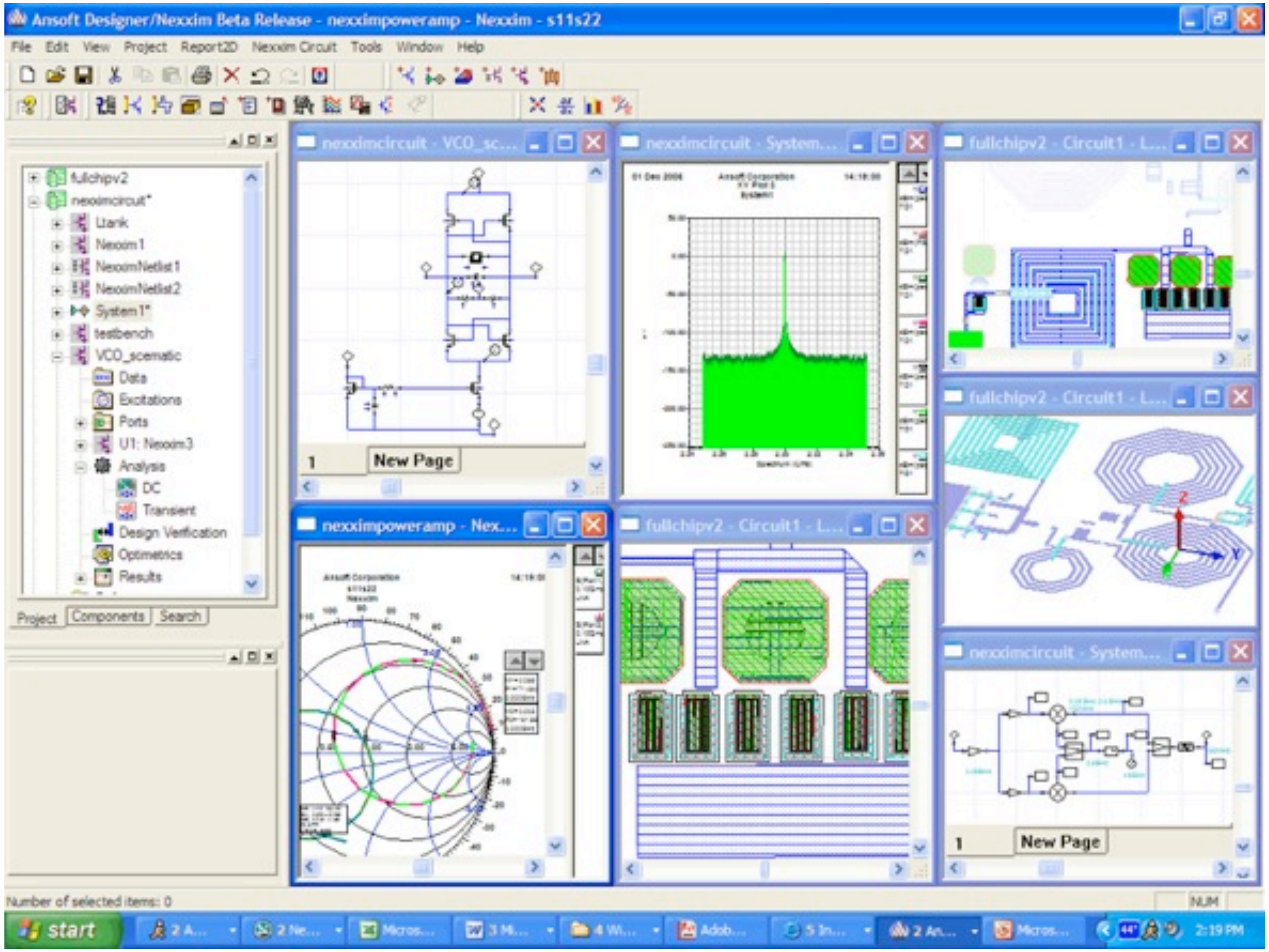
Developing Complex Systems

- How do scientific disciplines construct complex systems ?

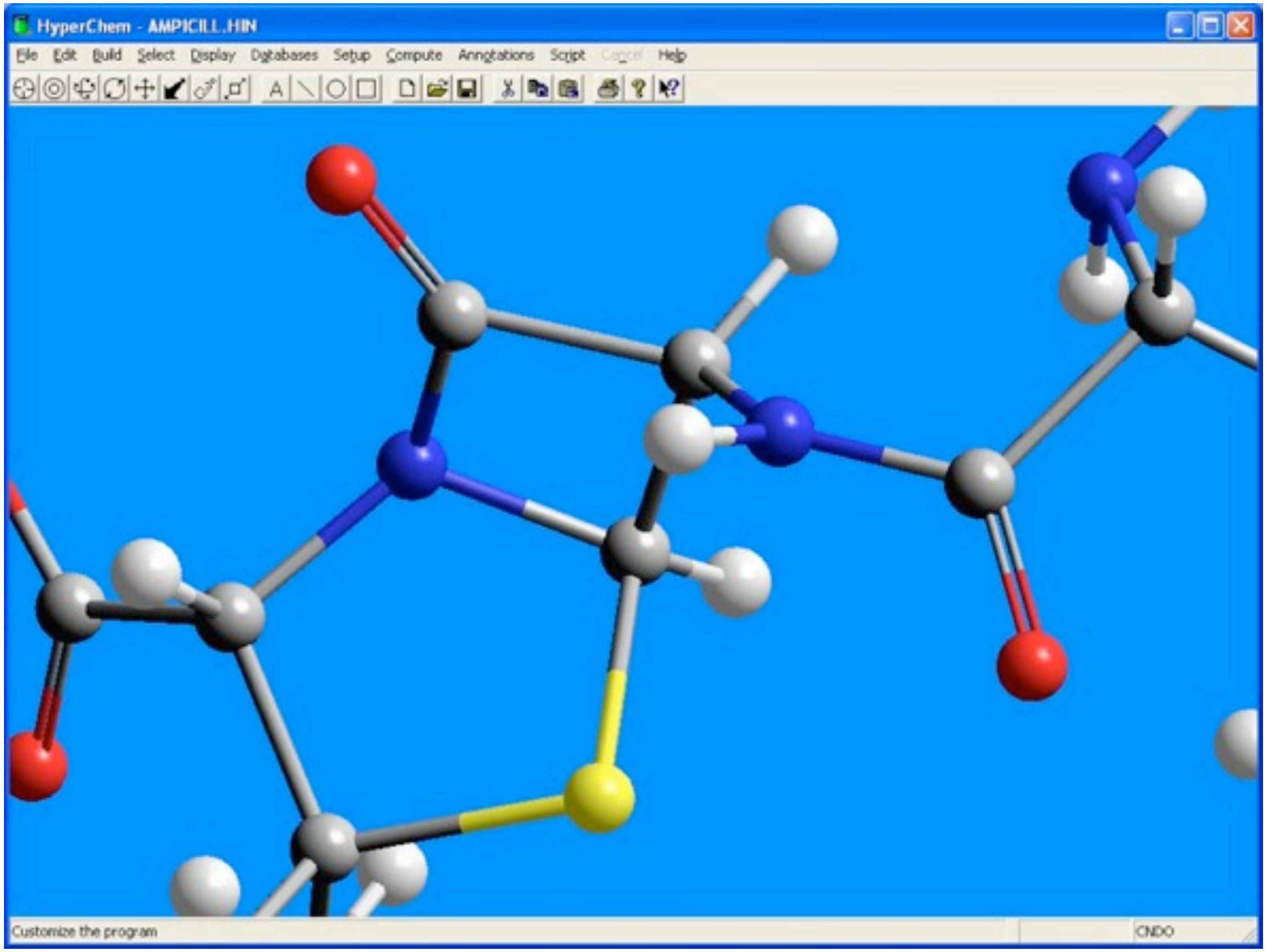
Architectural Software



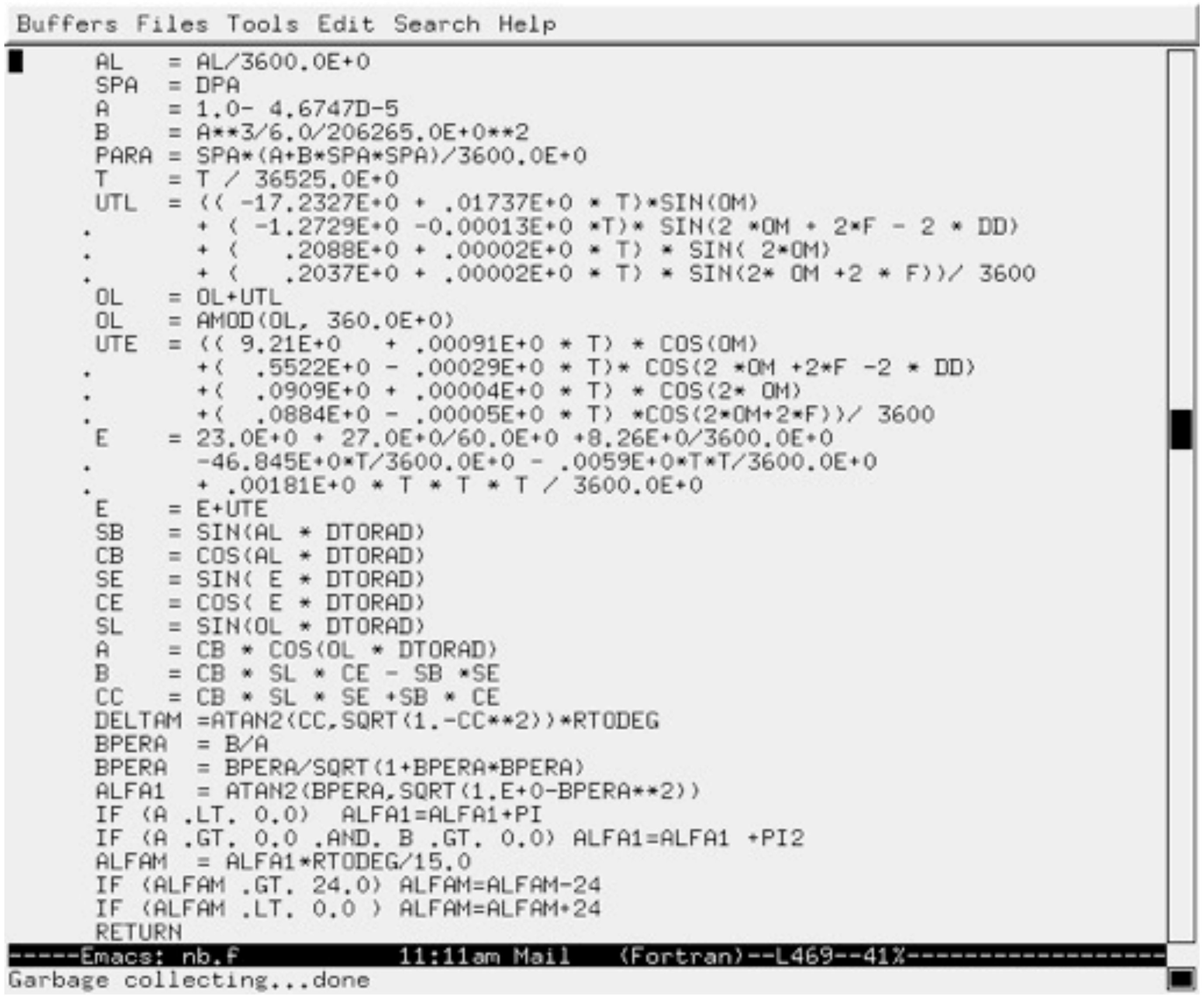
RF/mW Design & Analog/RFIC Verification



Visualization & Manipulation of molecules


HyperChem - AMPICILL.HIN
File Edit Build Select Display Databases Setup Compute Annotations Script Cancel Help
Customize the program
CNDO

Computer Science

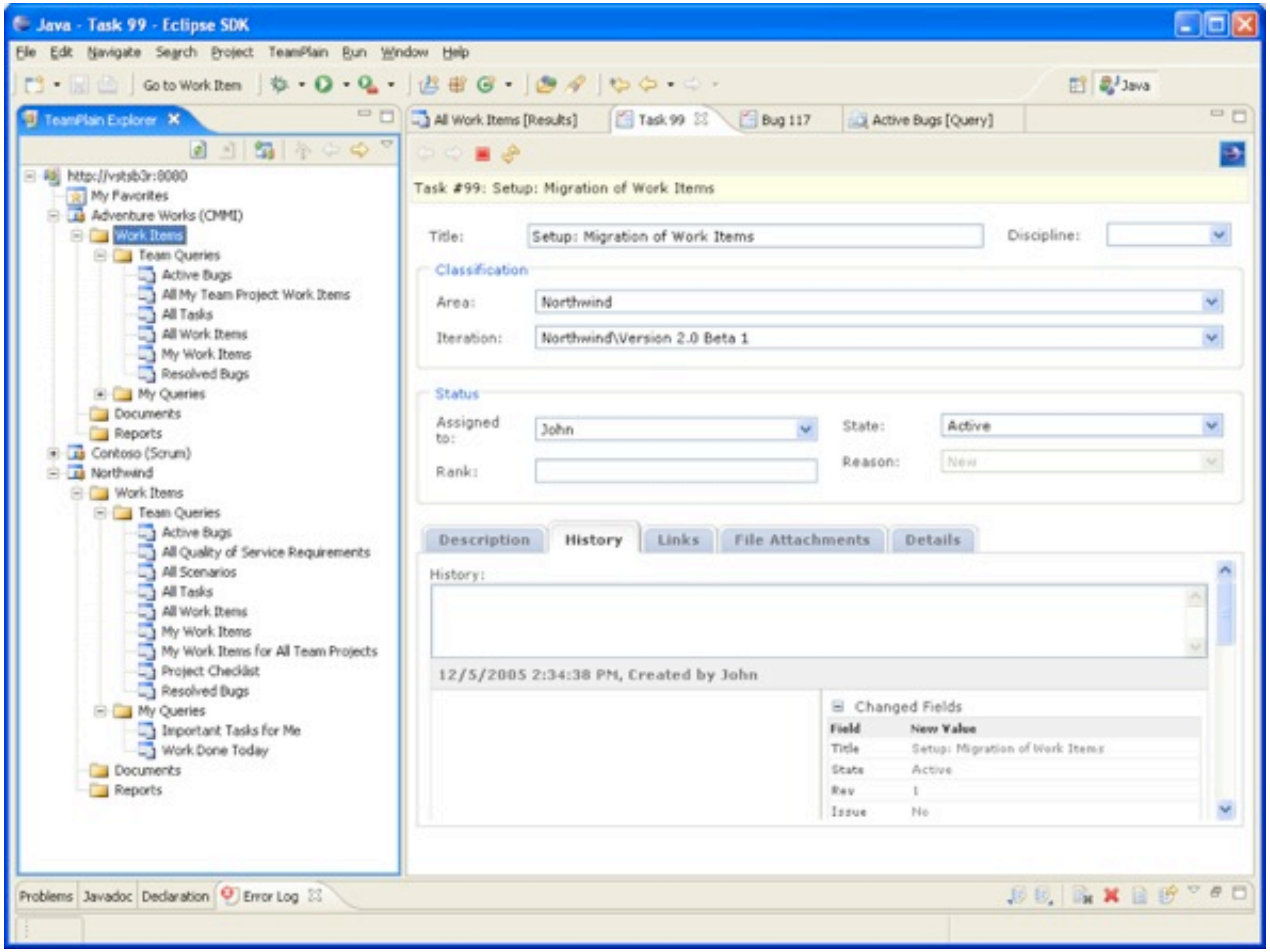


```
      AL    = AL/3600.0E+0
      SPA   = DPA
      A     = 1.0- 4.6747D-5
      B     = A**3/6.0/206265.0E+0**2
      PARA = SPA*(A+B*SPA*SPA)/3600.0E+0
      T     = T / 36525.0E+0
      UTL   = (( -17.2327E+0 + .01737E+0 * T)*SIN(OM)
     .         + ( -1.2729E+0 -0.00013E+0 *T)* SIN(2 *OM + 2*F - 2 * DD)
     .         + (   .2088E+0 + .00002E+0 * T) * SIN( 2*OM)
     .         + (   .2037E+0 + .00002E+0 * T) * SIN(2* OM +2 * F))/ 3600
      OL    = OL+UTL
      OL    = AMOD(OL, 360.0E+0)
      UTE   = (( 9.21E+0    + .00091E+0 * T) * COS(OM)
     .         +(  .5522E+0 - .00029E+0 * T)* COS(2 *OM +2*F -2 * DD)
     .         +(  .0909E+0 + .00004E+0 * T) * COS(2* OM)
     .         +(  .0884E+0 - .00005E+0 * T) *COS(2*OM+2*F))/ 3600
      E     = 23.0E+0 + 27.0E+0/60.0E+0 +8.26E+0/3600.0E+0
     .         -46.845E+0*T/3600.0E+0 - .0059E+0*T*T/3600.0E+0
     .         + .00181E+0 * T * T * T / 3600.0E+0
      E     = E+UTE
      SB    = SIN(AL * DTORAD)
      CB    = COS(AL * DTORAD)
      SE    = SIN( E * DTORAD)
      CE    = COS( E * DTORAD)
      SL    = SIN(OL * DTORAD)
      A     = CB * COS(OL * DTORAD)
      B     = CB * SL * CE - SB *SE
      CC    = CB * SL * SE +SB * CE
      DELTAM =ATAN2(CC,SQRT(1.-CC**2))*RTODEG
      BPERA  = B/A
      BPERA  = BPERA/SQRT(1+BPERA*BPERA)
      ALFA1  = ATAN2(BPERA,SQRT(1.E+0-BPERA**2))
      IF (A .LT. 0.0)  ALFA1=ALFA1+PI
      IF (A .GT. 0.0 .AND. B .GT. 0.0) ALFA1=ALFA1 +PI2
      ALFAM  = ALFA1*RTODEG/15.0
      IF (ALFAM .GT. 24.0) ALFAM=ALFAM-24
      IF (ALFAM .LT. 0.0 ) ALFAM=ALFAM+24
      RETURN
-----Emacs: nb.f          11:11am Mail    (Fortran)--L469--41%----------------
Garbage collecting...done
```

Corollary

- We need to construct systems that are typically more complex than in other disciplines
  - for several reasons
- We have tangible elements to manipulate
  - Buildings, circuits and molecules *need* a representation that is different than their physical one
- Yet lots of developers still seem to prefer basic tools
  - yes, emacs is a basic tool...

Eclipse ?



Task #99: Setup: Migration of Work Items

Title: Setup: Migration of Work Items
Discipline:

Classification

Area: Northwind
Iteration: Northwind\Version 2.0 Beta 1

Status

Assigned to: John
State: Active
Rank:
Reason: New

History:

12/5/2005 2:34:38 PM, Created by John

Changed Fields

Field	New Value
Title	Setup: Migration of Work Items
State	Active
Rev	1
Issue	No

Eclipse...

- Eclipse is a decent integrated development environment
  - integrates navigation, editing, unit tests, refactoring, ...
  - was developed by a lot of former Smalltalk people :-)
- But at its core it is file-based
  - So ? Why don’t I like this ?

Files versus Objects

- Non computer science disciplines:
  - Architects work with construction materials&buildings
    - So do their tools
  - Molecular biologists work with modules
    - Environment manipulates molecules
  - ...
- We work with objects
  - Most tools deal with files ?!

Smalltalk image approach

- The Smalltalk image is a live environment
  - consists entirely of objects
  - objects are manipulated
- Files are one way of *storing* objects
  - code too, since code are objects
  - Databases are another mechanism, or network sockets or …

Sidenote on Environments

- Good developers tailor their environment
  - So they need to be easily extensible
    - emacs: easy
    - Smalltalk environments: easy
    - Eclipse: possible
    - Most environments: hard or not possible
- Always favor an extensible one
  - control your tools!

Multi-user Development

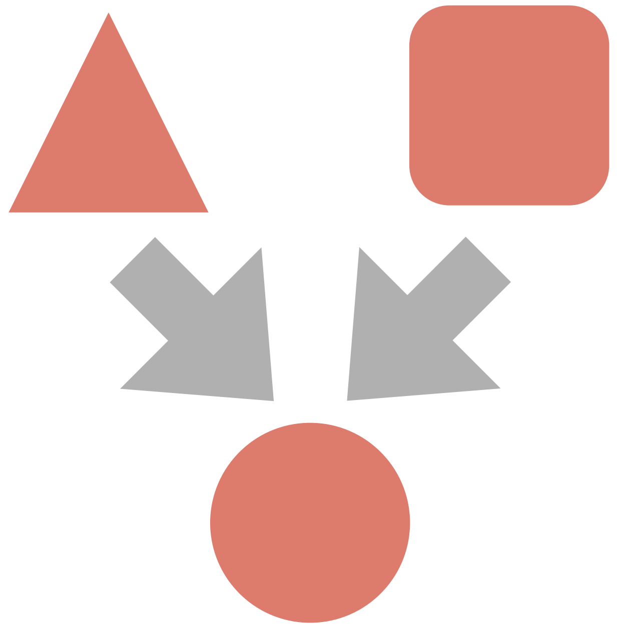
- Needed
  - a code repository that allows multiple users
  - integrated versioning
  - configuration management
- The language also has packaging mechanisms
  - with or without namespaces
- These concepts cross-cut

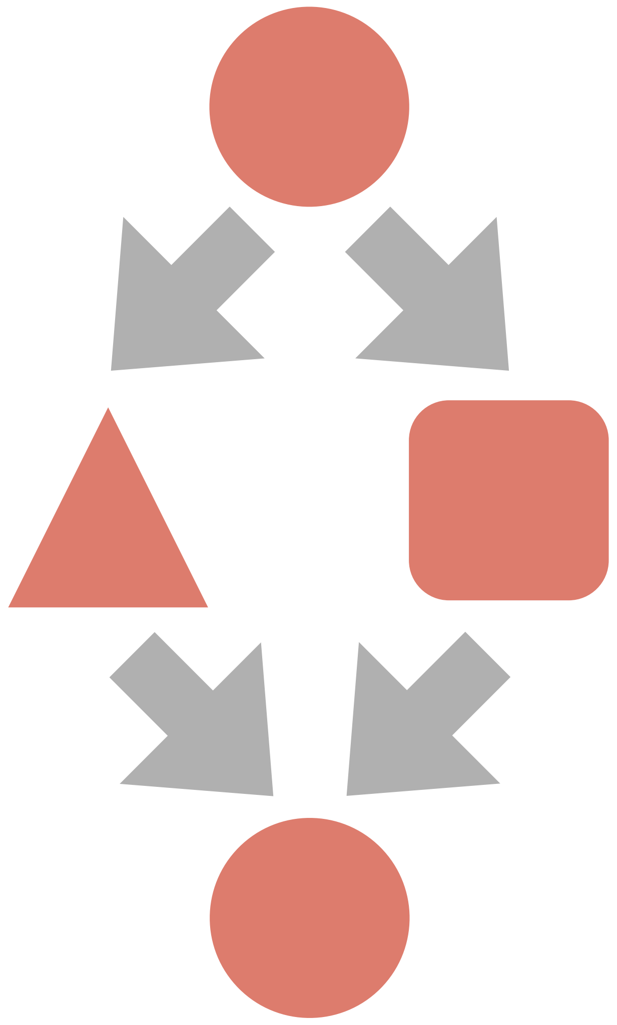
Code repositories and multiple users

- Need to store code (obviously)
  - but preferable also binairies, documentation, tests, ...
- Locking vs. concurrent
  - Lock: one user has (part of) code, unlocks when done
  - Concurrent (lazy locking): several users can work simultaneously on the same system
- Support for merging

Merges

- Two-way merge





- Three-way merge

Example

- One framework,
- instantiated for two different clients,
- each with their own customizations,
- Where there is a stable version,
- and two development branches
  - a new version and a brand new one
  - one dependent on the customization of the framework for one particular client

Common Concepts

- Repository: holds data
- Local copy/working copy
- Change or Delta: a modification to the data

- Load or Check out: create local copy from repository
- Commit or Publish: copy changes to repository

Version Control Concepts

- Edition: copy of data of the repository
- Version: frozen edition, identified by name/number

Let’s view two systems

- cvs
- Envy

(Many more exist, but cvs is archetypical for most popular tools, and Envy is a nice contrast)

cvs : Concurrent Versioning System

- Granularity: file
- Users work detached:
  - Load local copy of files from cvs server (*repository*)
  - Work on local copy (*working directory*)
  - Commit changed files back to server
- Loading local copy can be done from the network
  - using secure shell or not

Conflicts

- Multiple users can work on same file
  - each in its own working directory
- When committing, versions in working directory are checked with versions in repository
  - triggers merge when there are differences

Semantics

- cvs stores text
  - has no semantics about what it stores
  - works with latex files, C++ files, ...
- Therefore it cannot use semantics
  - e.g. renaming a method, changing a latex label, ...

Envy

- Granularity: Method
- Users work connected to the repository
- Works with methods, classes, ...
  - e.g. have all versions for a particular method
- They load code in their environment, and version it when done
  - Everybody can see and use all versions

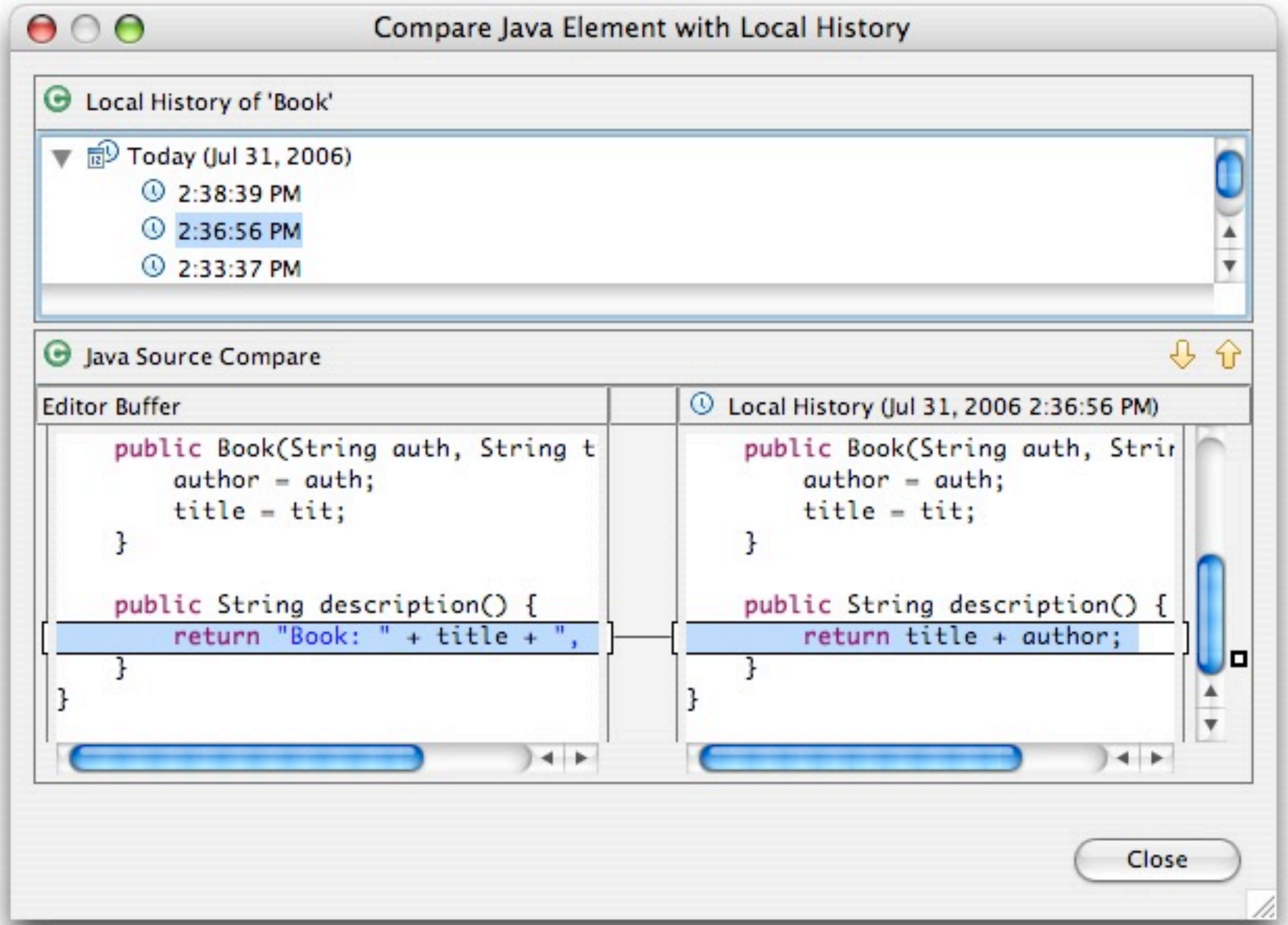
Envy: Configuration Management features

- Editions are made into versions
- Applications group classes and methods
  - can have editions and versions themselves
  - have prerequisite versions (!)
- Configurations group applications
  - (e.g. Manifests)
- Support for conditional loading and prerequisites
  - Platform-specific code, for example

On granularity...

- With cvs, you have a history of the *files* you've checked in
- With Envy, you have a history of the *development* you did
- This is fundamentally different

What is Envy doing in Eclipse ?!



```
    public Book(String auth, String t
        author = auth;
        title = tit;
    }

    public String description() {
        return "Book: " + title + ",
    }
}
```

```
    public Book(String auth, Strir
        author = auth;
        title = tit;
    }

    public String description() {
        return title + author;
    }
}
```

More recent approaches

- svn
  - better cvs
- distributed version control systems
  - examples: git, mercurial
  - much better support for branching, versioning, integration

Concepts in Code Repositories

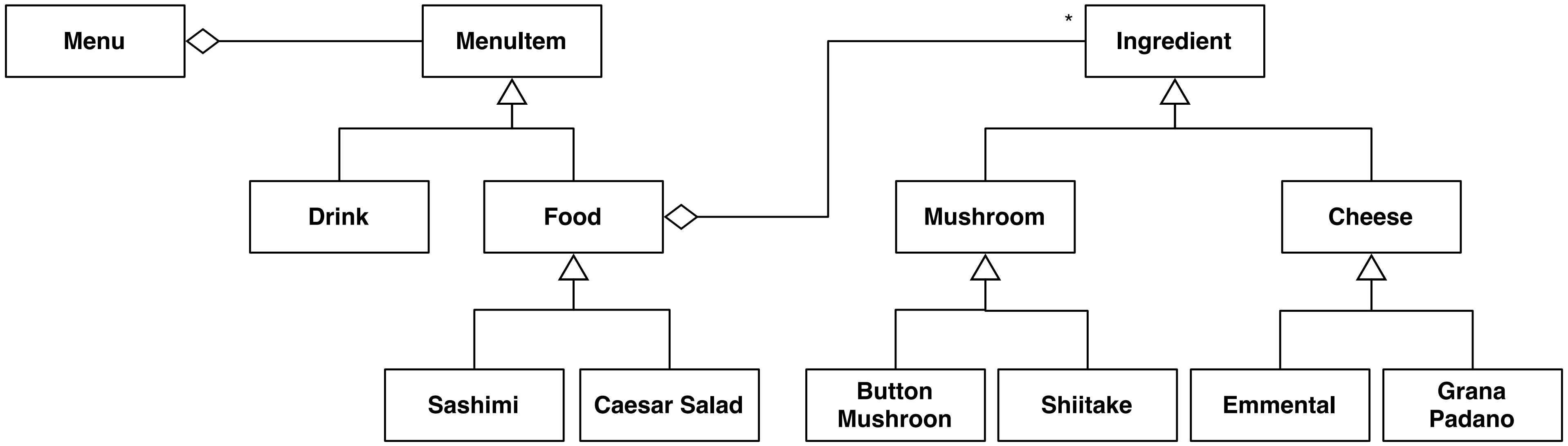
- Code
- Package
- Configuration

- Packages and Namespaces should be orthogonal
  - package contains definitions
  - namespaces is a visibility mechanism

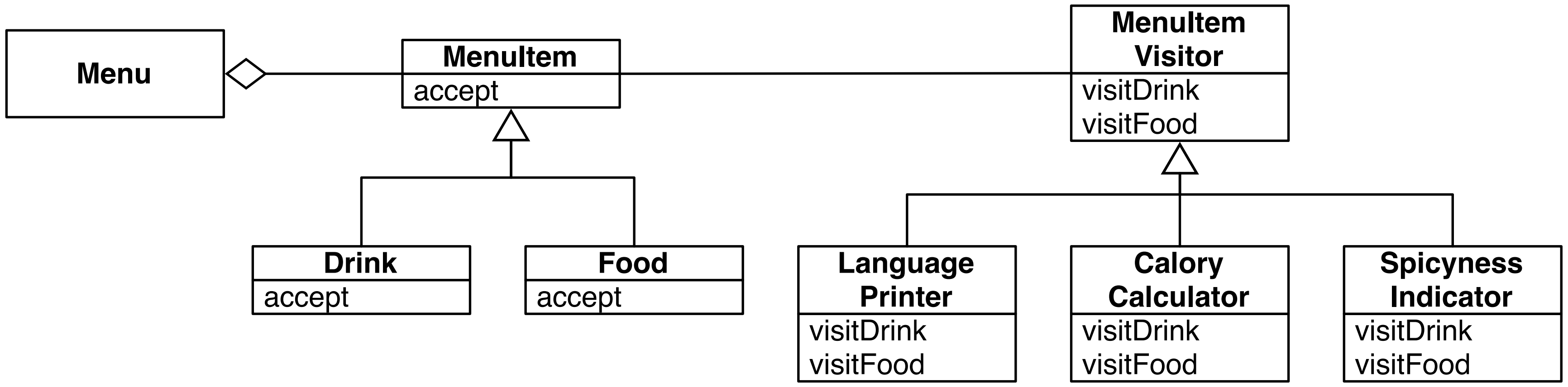
Versioning

- All the elements need to be versionable
- Decisions, decisions:
  - granularity of version
    - line of code, method, class+methods, package, ...
  - forms of version numbers
    - single number, composed number, alphanumeric
  - version numbers versus release numbers
    - and their relationships

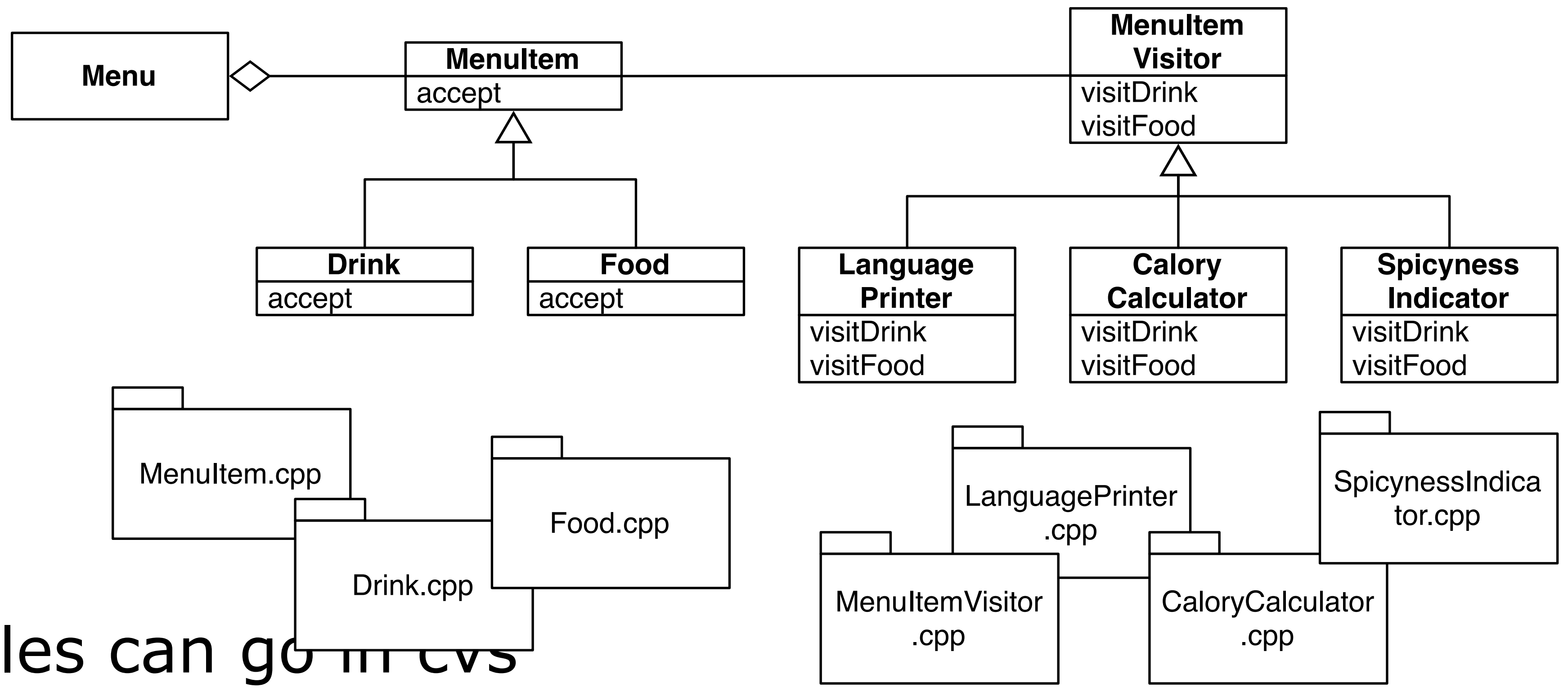
Concrete example : Menu Framework



Menu Framework with Visitor

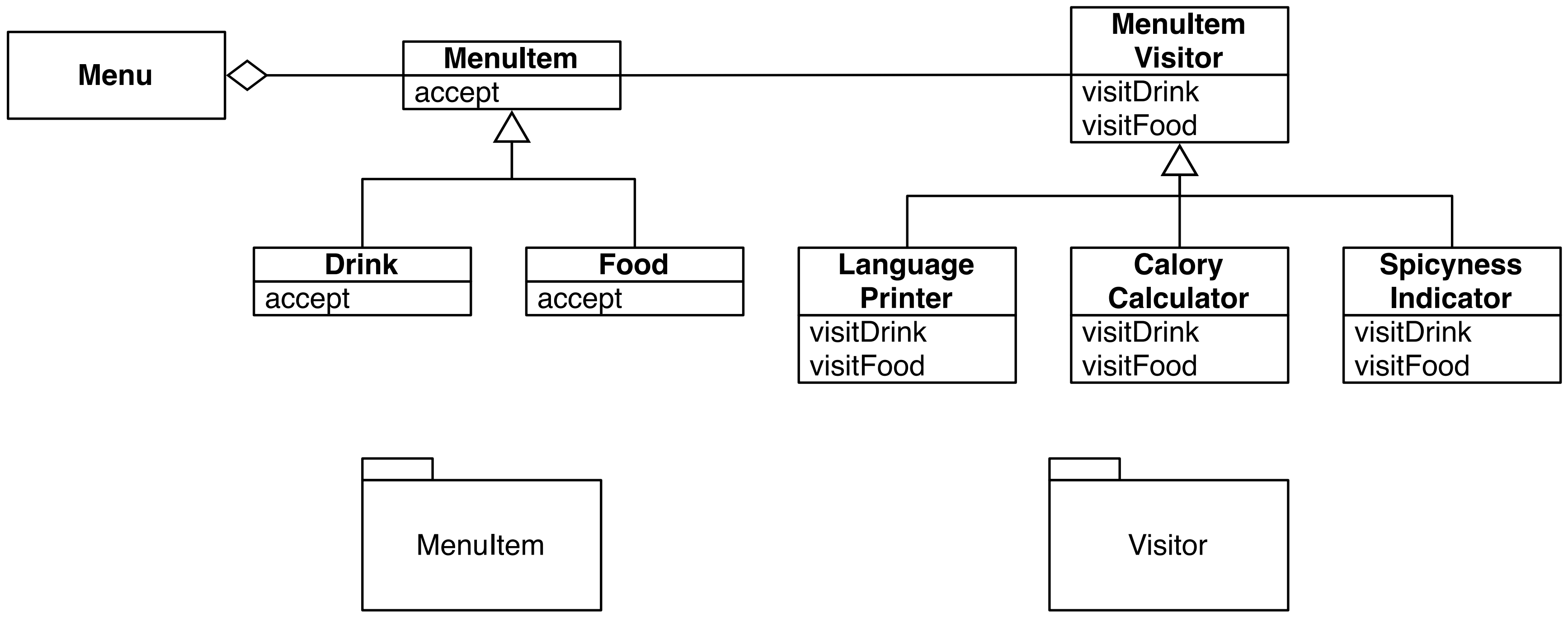


C++ Files



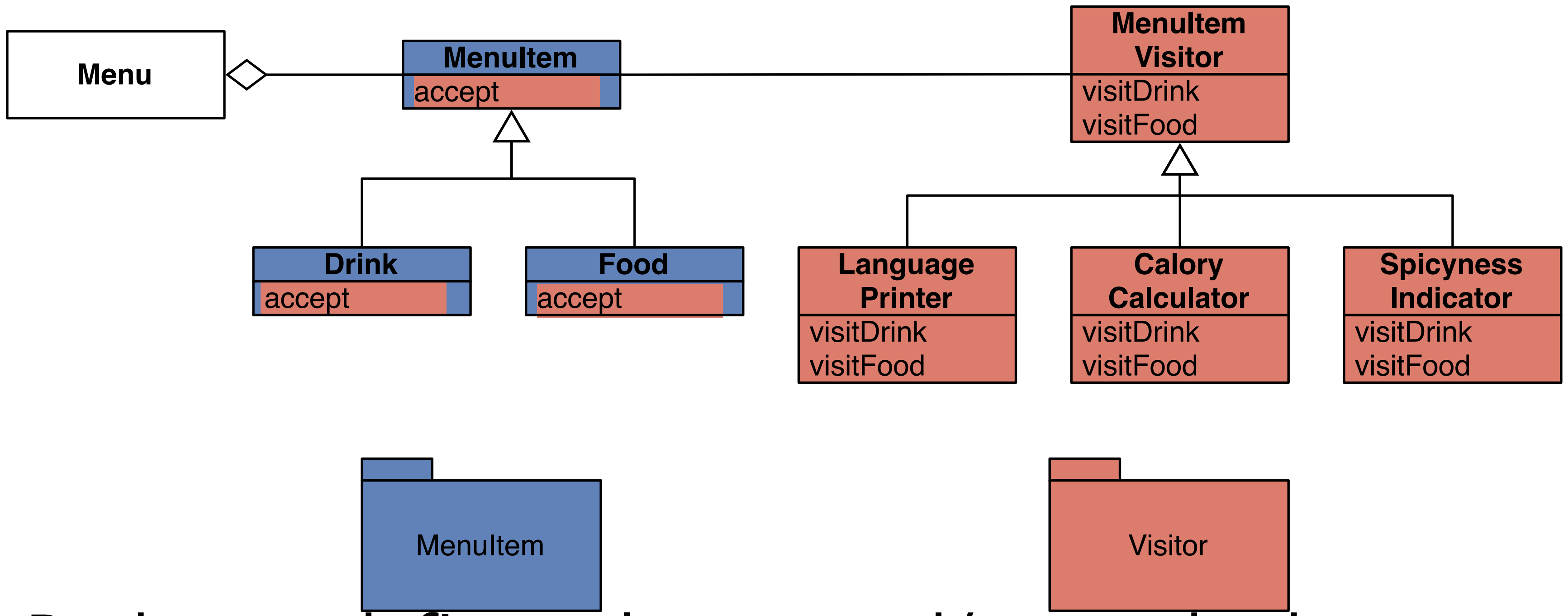
- Files can go in cvs
  - But decomposition is not the right one
  - What if the visitor traversal needs to be changed?

Java Packages



- Packages to regroup classes, storage still in files
- Decomposition still not the right one
  - What would be the right decomposition?

Smalltalk class extensions



- Packages defines classes and/or methods
  - Can be different versions, under control of different people/project/companies

Note: declarative packages

- The Class Extensions scheme can be done with files
  - See Smalltalk file-outs
- Declarative system is needed
  - Class definition
  - Method definition, not nested within class
- Java Packages are different
  - Packages contain classes, classes contain methods
  - Watch out for a new Java package system :-)

Software-engineering wise

- Important to be able to separate development into logical, manageable pieces
  - e.g. Visitor design pattern
- Each piece should have:
  - owners & responsibles
  - versions
  - dependencies
  - post-load and pre-unload statements

Corollary

- Good packages support evolution
  - Company can sell parsetree
  - Other company can sell visitor for parsetree

- Code repositories and packages should support flexible forms of packaging code

- Code repositories, packaging & storage are linked

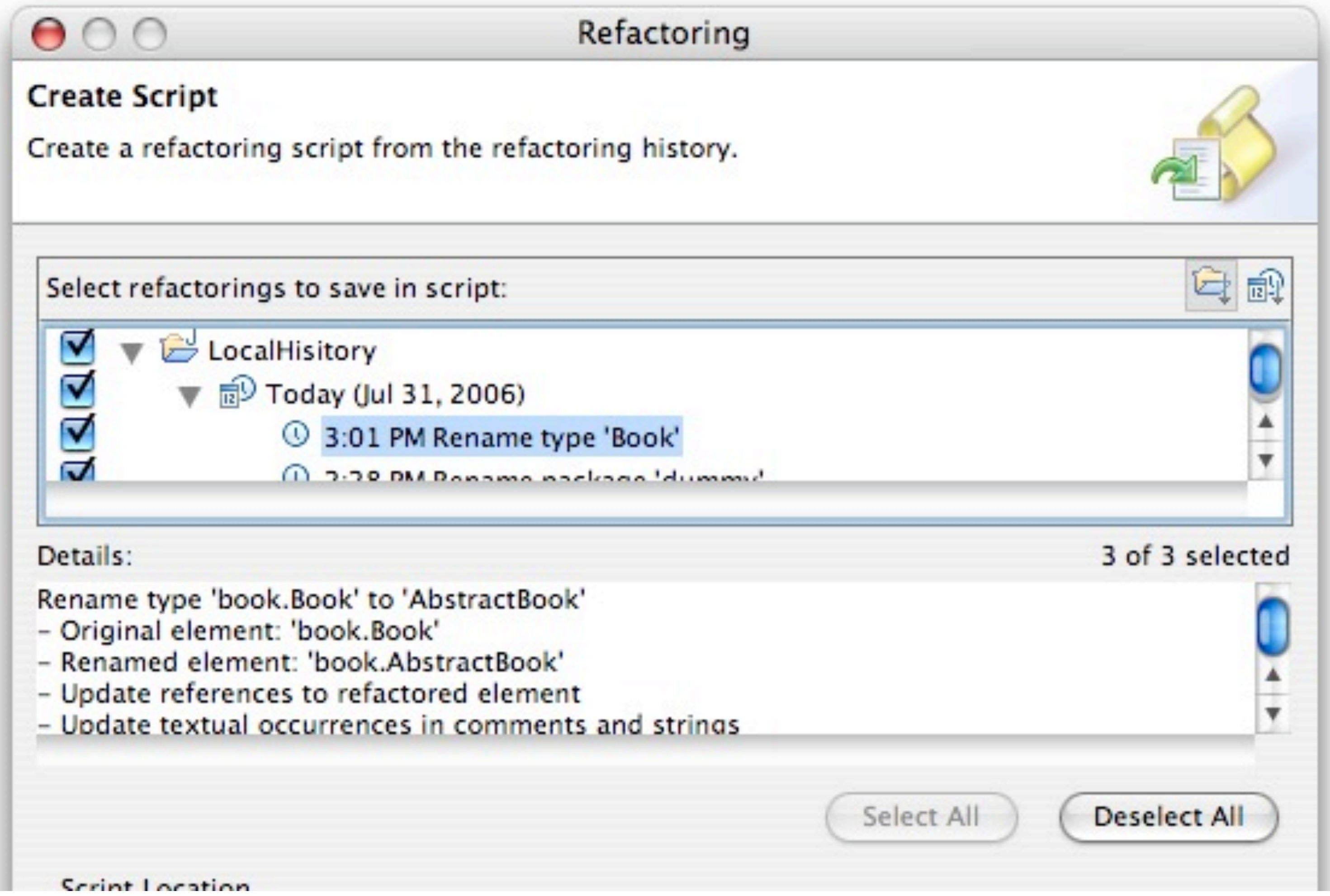
Sidenote

- Design question:
  - why is the plug-in mechanism in Eclipse so difficult?

Last but not least

- We discussed granularity
  - want to see the development you really did, not the changes you made
- Nice example: Refactoring Scripts in Eclipse
  - Record and replay the refactorings you did
- Why is this practical ?

Saving & Replaying Refactoring Scripts



Refactoring

Create Script

Create a refactoring script from the refactoring history.

Select refactorings to save in script:

☑ LocalHisitory
☑ Today (Jul 31, 2006)
☑ 3:01 PM Rename type 'Book'
☑ 2:28 PM Rename package 'dummy'

Details: 3 of 3 selected

Rename type 'book.Book' to 'AbstractBook'
- Original element: 'book.Book'
- Renamed element: 'book.AbstractBook'
- Update references to refactored element
- Update textual occurrences in comments and strings

Conclusion

- Multi-user development needs to be supported
  - code repositories with concurrent access
  - version support
  - (automatic) merge support
  - configuration management
- Current systems are quite weak
  - cvs & files
  - watch out for newer offerings