Chapter 3

Software Engineering Tools

Topics

- Project management tools
- System modeling tools
- Integrated development environments
- Change and configuration management tools
The theme

- A tool can be used
  - for personal productivity and/or
  - for teamwork (workgroup, collaborative) productivity → Computer-Supported Collaborative Work (CSCW) or workgroup computing

- Without adequate level of maturity, the use of a tool can hurt more than help

- A generic term for software engineering tools is Computer Assisted Software Engineering (CASE)

- Tools that comprehensively assist programmers are termed Integrated Development Environments (IDEs)

- Change and configuration management tools are a natural extension of project management tools

Project management tools

- Project management is about distribution and control of budgets, time and people
- “if you can’t plan it, you can’t do it”
- “unless you know your past, you cannot properly plan your future”
- Project management requires tools for effective scheduling and controlling project activities, for estimating project costs, for collecting metrics, etc.
- Contemporary tools are frequently included in integrated management support environments that tie together typical project management with strategic planning, business modeling, portfolio management, document management, workflows, etc.
- Many of traditional project management tools are available in web-based versions
### Categories of project management tools

- Project scheduling and controlling
- Aligning project and performance management with strategic objectives
- Unifying project management with web-based collaboration and content management
- Unifying project management with web-based portfolio management
- Integrating project management with metrics
- Integrating project management with risk management

### Project scheduling and controlling

- **Activity network graphs** identify events that must occur before an activity can begin, specify the duration of each activity and its due date, assign human and other resources to all activities, etc.
  - CPM (Critical Path Method) networks
  - Gantt charts
- **Project management tools** are capable of generating activity graphs once necessary project information is provided
- **Project management tools** adjust and rebuild schedules whenever changes occur to an activity or to a resource
Aligning project and performance management with strategic objectives

- Project management is subservient to demands of effectiveness, efficiency, and performance improvement

- Some project management tools are included in larger environments to support a wide variety of managerial activities, such as managing:
  - goals
  - development plans
  - meetings
  - e-mail
  - documents
  - performance reviews including annual reviews
Many software projects are distributed – they span multiple people and cross organizational boundaries. Project management becomes a “component” in a web-based collaboration of knowledge workers. Collaboration management includes support for project stakeholders to lodge issues, define and assign actions, document outcomes. Content management for easy access and sharing of documents and other informational content.
Unifying project management with web-based portfolio management

- **Portfolio management** – process of enabling the enterprise to align resources and investments with strategic objectives
- **Tools in this category** foster a web-based work environment for
  - project discussions, tracking and work assignment
  - team calendars and timesheets
  - electronic meetings to reach consensus and take consultative decisions
  - sharing project documents and maintaining versioning of them
  - workflow automation and tracking of decisions
  - conventional planning and scheduling with Gantt and CPM document templates and patterns to encourage best practices
  - aligning projects with strategic plans and business models
  - managing individual and shared resources
  - allocating skills to human resources and allocating resources to projects
  - generation of cross project summaries and status indicators
Integrating project management with metrics

- **Metrics** is the discipline of measuring software development processes and products so that collected information can aid future project planning.

- **Architectural design** must be backed by metrics to ensure system supportability, i.e. its understandability, maintainability and scalability.

- **Metrics collection tools** must be easy to use and not obstructive.

- **What-If** analysis.
Small Worlds (what-if analysis)

Small Worlds (class internals analysis)
Integrating project management with risk management

- **Risks** – “potentially adverse circumstances that may impair the development process and the quality of products” (Ghezzi et al.)
- **Risk management** is a decision-making activity that assesses the impact of risks (uncertainties) on decisions
  - it weighs distributions of possible project outcomes versus the probabilities of arriving at these outcomes
  - given the acceptable level of tolerance on risk, it estimates the likelihood with which a desired outcome will happen
- **Tools** that integrate project management with risk management are frequently implemented as add-ins to conventional project management tools (such as Microsoft Project), add-ins to spreadsheets (such as Microsoft Excel), or as database applications (e.g. on top of Microsoft Access).

Risk Radar™ (Risk data input)

<table>
<thead>
<tr>
<th>Impact:</th>
<th>Cost: Low</th>
<th>Schedule: 0</th>
<th>Technical: 2</th>
<th>Other: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk Exposure: 0.2</td>
<td>Risk Level: L</td>
<td>Level Trend: ⇒</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Probability</th>
<th>High</th>
<th>Medium</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status:</th>
<th>Watch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk Category:</td>
<td>Technical</td>
</tr>
</tbody>
</table>

|----------------------------------|-----------------------------------|---------------------|-----------------------------------------------|

Project funding allocations may be affected, or the project may not meet performance requirements.
System modeling tools

- **Characteristics:**
  - visual
  - present models at various level of abstraction
  - traditionally known as CASE
  - support teamwork (CSCW – computer supported cooperative work)

- **Can be classified into tools for:**
  - managing requirements
  - visual UML modeling
  - report generation
  - database modeling
Managing requirements

- **Requirements** are text statements within a requirements document
  - normally, one requirements document per use case
  - defined at multiple levels of abstraction
  - there is hierarchical arrangement between requirements

- **Tool** must facilitate team collaboration in:
  - writing requirements document,
  - identifying various categories of requirements in the documents,
  - managing changes to requirements, and
  - making requirements accessible to all developers.

---

**IBM Rational Requisite Pro (requirements document)**

Use Case Specification: Arrange Shipment

1. **UC7 Arrange Shipment**
   1.1 Brief Description
   This use case is an extension of the Checkout use case. A customer in the warehouse system places an order by phone. The warehouse system is considered an actor in this use case scenario. The information given by the warehouse system to the store is all of the necessary information needed to place the order and ship the product(s) to the club member.

2. Flow of Events
   2.1 Basic Flow
   UC7: Upon completion of the Checkout use case complete order information will be sent to the warehouse system
**Visual UML modeling**

- Tools in this category range from simple graphics-only tools to repository-based tools supporting teamwork and allowing customization of graphical representations.

- The most sophisticated tools
  - have a repository set up on a commercial database,
  - provide complete integration of models,
  - interface to other software engineering tools,
  - support multiple versions of design artifacts and models,
  - generate code for a variety of programming languages and databases, etc.
Report generation

- So that visual models and related documentation can be shared among all project stakeholders, not just developers
- Reports can be generated according to various templates and various file formats
  - templates determine the structure of documents
  - users can specify required content
  - file formats usually supported by modeling tools include HTML, Microsoft Word, and/or Acrobat's pdf
  - tool generating HTML files should give an option to generate HTML reports to one screen, thus allowing for convenient printing

MagicDraw™ and IBM Rational Suite
Database modeling

- **Database modeling** requires unique set of capabilities
- Although high-level database models can be developed using UML class diagrams, specific logical/physical models require referencing tables, not classes
- Database specific concepts, not known to UML, include also
  - referential integrity,
  - indexes,
  - stored procedures,
  - triggers.
- Differences between database management systems must also be supported

Sybase PowerDesigner

```sql
create table "Department" (  
  "department_code" CHAR(3),  
  "department_name" VARCHAR2(10)  
)  
create unique index "dpt_name уникали" on "Department"  
create unique index "department_name уникали" on "Department"
```
Integrated Development Environments (IDE)

- Software programming environments
- Standard capabilities of IDEs include
  - code completion,
  - support for programming standard features (code auditing),
  - integration with the build tools such as Ant or make, etc.
- IDEs can be distinguished with regard to:
  - integration with software modeling
  - support for the scope and nature of application development and required technologies
  - integration with distributed development environments and with change and configuration

Categories of IDE-s

- Routine programming tasks
  - writing the program
  - executing the program
  - debugging the program
- Integration with software modeling
- Enterprise application development
- Integration with business components
- Integration with change and configuration management
Routine programming tasks (Sun ONE Studio)

Writing the program (Sun ONE Studio)
Writing the program (Sun ONE Studio)

```java
/**
 * HelloWorld.java
 * public void println(java.lang.Object)
 * 
 * Print an Object and then terminate the line. This method behaves as
 * though it invokes println(String) and then println().
 *
 * Parameters:
 * @param x - The object to be printed.
 */

public class HelloWorld {
    private static int id;
    private int myId;
    private int size;

    /** Creates a new instance of HelloWorld */
    public HelloWorld() {
        myID = ++id;
        // init...
    }

    public void hello() {
        System.out.println("Hi my ID is "+myId);
    }

    public void expandTheWorld(){
        size += (myID * Math.random());
    }

    public static void main(String[] args) {
        // main
    }
}
```

1. What methods in current object?
2. Description of current method

Executing the program (Sun ONE Studio)

**Program's output**

```
Hi my ID is 1
HelloWorld[1, 0]
```

**Is the program running or already dead?**
Debugging the program (Sun ONE Studio)

Setting breakpoints or watch expressions

Integration with software modeling (Together ControlCenter)
Enterprise application development

- In Java world, an enterprise IDE implies the support for the design, development, and deployment of J2EE applications
- Enterprise-strength IDE is expected to support
  - various templates to get job done quickly, including templates for JSP, EJB, Java files, web services, etc.
  - multiple GUI types
  - immediate visualization in the code of changes in the GUI
  - JDBC-compliant database connectivity
  - reverse-engineering of preexisting database tables into entity beans ready to be deployed on an application server

Sun ONE Studio for J2EE development
Integration with Business Components

- Business components constitute the middle-tier of a multi-tier system architecture
- Business components are best deployed on a separate server – an application server
- Development of a database back-end application is usually performed in the following sequence:
  - reverse engineer preexisting database schema/tables
  - generate a set of GUI “views” that reflect the database tables
  - customize the views to apply business logic

BC4J application (Oracle JDeveloper)
Integration with change and configuration management

- **Teamwork** capabilities are frequently achieved by linking to independent change and configuration management tools.
- **Configuration management tool** differs from a version control system (VCS) by being able to manage all modeling artifacts, not just programming code.
- Version control systems provide graphical “diff” tools to perform comparisons between various versions of the code.
- Configuration management tool allows developer to **check-out** artifact objects to his/her workstation (private workspace) and **check-in** back to the project repository on completion of the task.

Sun ONE Studio (version control integration)

Profile

Version Control System Profile

- < Select a Profile >
- CVS (Windows NT/2000)
- ClearCase (Windows)
- WoE Empty (Windows)
- PVCS (Windows NT/2000)
- Perforce
- VSS (Windows NT/2000)
**Change and configuration management tools**

- **Change management tool** records changes that have occurred during the software lifecycle.
- **Version control** is the domain of configuration management tools.
- **Steps** in change and configuration management:
  - private workspace ↔ public workspace
  - configuration (stream) = set of versions
  - to-do-list = required changes
  - check-out of "change set" → check-in → marking change set as delivered (visible to the team)
  - creating a new baseline (build)
  - promoting a baseline to "recommended baseline" → "rebasing" the development configurations of developers.
Support for changes

Changes to or within a project must be managed, for otherwise the changes will “manage” the project. Project management is a form of change management.

Changes can be due to:
- external or internal project conditions (budget, resources, competition, etc.)
- changes in user requirements
- defects in software
- necessary software enhancements (future features)

The change process should be defined in a workflow:
- change management tool can integrate with a workflow management tool.

Support for versions

Support for system building

Support for reengineering
IBM Rational Clear Quest (managing defects)

Possible actions (state changes)

IBM Rational Clear Quest (defect chart)
Support for versions

- A configuration management system has an internal facility to maintain various version derivation graphs.
- The graphs allow:
  - tracking changes applied to an artifact
  - composing versions of various objects to create a higher-level configuration of versions
- compare-and-merge capabilities – to compare the differences between the current version of the artifact and the baselined version.

Microsoft SourceSafe (merging versions)
“System building is the process of compiling and linking software components into a program which executes on a particular target configuration” (Sommerville)

The ultimate aim of system building is to create a distributable – an executable version of software that can be distributed to its users.

The main challenge in system building is to:
- adhere to all dependencies between software components and their various versions
- resolve all references to data files

Favorite Java-based build tool is Apache Ant (Another Neat Tool)
- Ant configuration files (build files) are written in XML and can be extended using Java classes

Sun ONE Studio and Apache Ant

- Ant configuration files (build files) are written in XML and can be extended using Java classes
Software reengineering is a branch of knowledge concerned with methods, techniques, and tools for examining an altering legacy systems to recover their designs and re-implement them in a new form.

Reengineering consists of a range of technologies:
- reverse engineering
- re-documentation from source code
- restructuring of program logic
- re-targeting the system to a modern platform
- source code translation to another language
- data re-engineering (as opposed to process re-engineering)
- forward engineering
Summary

- Software engineering tools can be classified into four groups:
  - project management tools,
  - system modeling tools (traditionally known as CASE tools)
  - Integrated Development Environments (IDEs), and
  - change and configuration management tools.
- Project management tools are concerned with project scheduling and controlling
  - related functions are in the areas of performance management, web-based collaboration, content management, portfolio management, metrics, and risk management.
- System modeling tools embrace all tools that assist software engineers in development tasks – from analysis via design to implementation.
- IDEs are sophisticated programming workbenches that provide a friendly programming environment to assist teams of programmers in all typical programming tasks.
- Change and configuration management tools are really two sides of the same coin
  - change management tool records changes that have occurred during the software lifecycle
  - version control is the domain of configuration management tools.