Principles of integrated software development environments

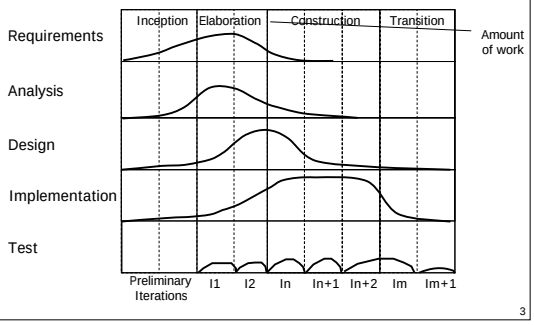
Wolfgang Emmerich
Professor of Distributed Computing
University College London
http://sse.cs.ucl.ac.uk

Learning Objectives

- Be able to define the notions of “tool” and “environment”
- Appreciate why tools and environments are critical in large-scale software development efforts
- Understand the conceptual building blocks of a modern development environment

2

Context: Software Process (e.g. USDP or RUP)



3

Software Development Artifacts

- While conducting an activity in a software process, developers produce, modify, review or consume software development artifacts
- Artifacts document "views" of the software system, e.g.
  - What functionality needs to be provided?
  - What quality requirements need to be met?
  - How is the software system structured into components and classes?
  - How are these classes realised?
  - How is the software system build from these classes?
  - Do they function correctly?
  - Does their integration address the functional and quality requirements?
- Large-scale development projects produce 1000s of artifacts

4

Software Development Artifacts (cont'd)

- Artifacts are (mostly) written in formal languages, e.g.
  - Functional requirements in SysML or UML Use case diagrams
  - Non-functional requirements in SLAng
  - Design in UML
  - Realization in Java, C#, Python or Ruby
  - Database schemas and queries in SQL
  - Ant or make build definitions
  - JUnit tests
  - Domain specific languages in Excel for acceptance testing

5

Formal languages demand tool support

- Well studied subject you should all be familiar with.
- Just to remind ourselves
- Formal languages
  - Have a context-free syntax
  - Have a static semantics
  - Might have usage conventions
  - Have a dynamic semantics
- Developers need tool support to write artifacts in formal languages (editors / compilers / checkers).

6

Software Development Tools

- A software development tool is a software system that assists a software developer in creating, reviewing, analyzing, transforming or executing one or several software artifacts.
- Tools are language sensitive / syntax-directed
- Examples:
  - UML tool
  - Eclipse
  - X86 Assembler
  - Ant
  - C++/Java/C# Compiler
  - Java Virtual Machine / Common Language Runtime
  - Lint

7

The need for tool integration

- In addition to static semantic constraints, artifacts often have inter-document consistency constraints
- For example:
  - A use case is elaborated in an interaction diagram.
  - A UML class and a Java class need to have the same names
  - A JavaBean class is refined in a table of a SQL schema
  - A JUnit test references the Java class under test
  - An ant build file refers to a number of Java classes by name
- Maintaining these consistency constraints manually in large-scale projects is prohibitively expensive.
- Requires tool integration

8

Integrated Software Development Environments

- An integrated software development environment (IDE) contains a number of software development tools. These tools are integrated and may have a common UI look-and-feel, work jointly on artifacts and enable team collaboration.
- Examples:
  - Eclipse
  - Microsoft Visual Studio
  - Netbeans
  - Rational Rose

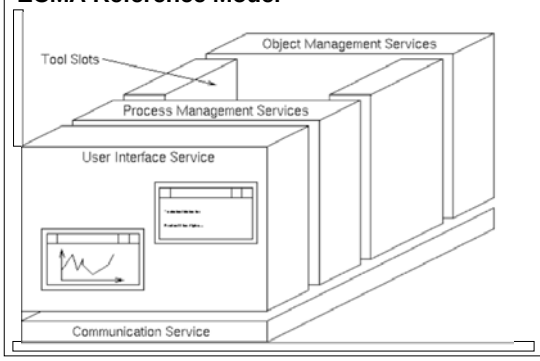
9

Reference Models/Architectures for IDEs

- In this course we will review the principles of how IDEs are assembled
- This of course has been done before
- Reference models & architectures for software engineering environments, IPSEs and IDEs
  - ECMA TR/55 (The “toaster” model)
  - IPSEN
  - Eclipse

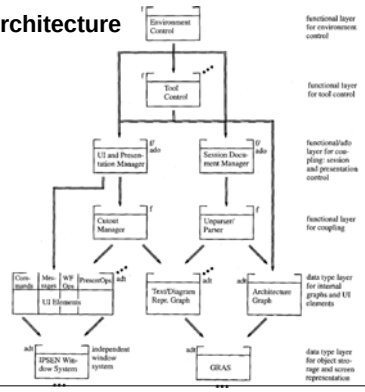
10

ECMA Reference Model



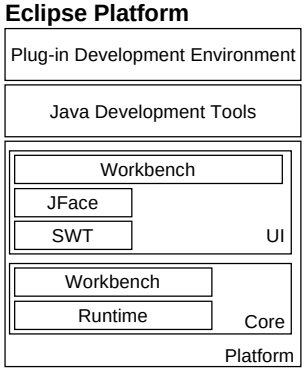
11

IPSEN Architecture



12

Eclipse Platform



13

Key points

- Developers need tool support to cope with complex development projects
- Tools are frequently syntax-directed
- Tool integration creates development environments

14

References

- M. Nagl (ed): Building Tightly integrated development environments. LNCS 1170. Springer Verlag. pp 32-44. http://dx.doi.org/10.1007/BFb0035684. 1996
- Reference Model for Frameworks of Software Engineering Environments. Edition 3 of ECMA TR/55. http://hissa.nist.gov/sp.500-211.ps
- E. Gamma and K. Beck. Contributing to Eclipse. Pearson. 2004 pp.1-7.

15