Program Editors

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

Learning Objectives

- Understand the principle requirements for program editors:
  - Language-sensitive editing
  - Static semantic constraints
  - Automated completions
  - Browsing support
  - Documentation aid
  - Refactoring support
- Appreciate how program editors in common IDEs meet these requirements

Usability, usability, usability

Overarching requirement:
- Increase programmer productivity
- Fundamental difficulty faced by every programmer - express yourself in a formal language
- Awareness of the syntax and semantics of the programming language(s) in use
- Support, but don't get in the way
- Highlight errors - but don't prevent them

None of this is sufficiently achieved by vi or emacs!!

Language-sensitive editing

- Syntax-directed editing, but turned out to be too restrictive - some programmers still like it for verbose constructs, e.g.
  - "surround with try/catch clause"
- Incremental parsing on-the-fly
- Temporal tolerance of syntax errors
- Highlighting of syntax errors
- Detailed error reports
- Pretty printing and automatic indentation
- Commenting / uncommenting

4

Supporting static-semantic correctness

- Visualising static-semantic errors, e.g.
  - Scoping errors
  - Typing errors
  - Uncaught exceptions
  - Unused declarations
  - Uninitialized variables
  - Unreachable statements
- Consistency constraints between different artifacts
  - Within the same language (e.g. import statements in Java)
  - Across languages e.g.
    - Java statements in HTML code of Java Server Pages
    - Compliance between XML documents and their schemas

5

Auto-completion

- Most modern programming languages have complex scoping rules
- Pro-active editing support in the presence of static semantic constraints
- Suggestions of possible completions
  - Methods to call
  - Types to use
  - Variable references
- Automatic generation of imports

6

Browsing support

Aim: Support navigation in complex source code

- Locate declarations and references of
  - Classes
  - Variables
  - Methods
- Bookmark important source code locations
- Outline class overview and inheritance hierarchy
- Keep track of traversal history and allow going backward and forward (required when trying to understand complex interactions between more than one classes)

7

Refactoring Support

- Code eventually deteriorates
- Refactoring is required for example to
  - Rename declarations
  - Reorganise inheritance hierarchies
  - Relocate methods or fields into other classes
  - Change the visibility of fields
- Program editors can aide these significant changes and perform them fully automatically

8

Workflow management

- To-Do-Lists
  - May be compiled from in-lined comments
  - In-lined comments may be generated by the editor intself
- Unresolved errors and warnings
- Tests that have not yet passed

- Integrated with browsing capabilities of the editor

9

Documentation Support

- Prevaling documentation style: literate programming
- Generation of documentation headers for artifacts and individual program fragments
  - Extraction of parameters
  - Extraction of return types
- Integration with templating and documentation engines (e.g. Java Doc)
- API Documentation preview

10

Integration Requirements

Editing programs is not done in isolation

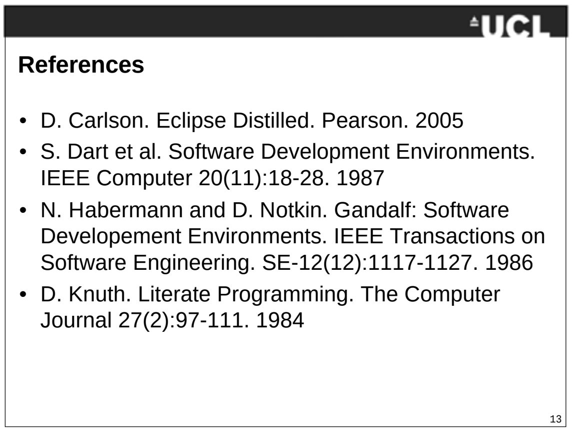
- Testing tools (e.g. Unit testers, coverage analyzers)
- Debuggers
- Metrics tools
- Version and configuration management tools
- Build tools
- Database connectors
- Application servers
- Browsers

11

Key Points

- Modern program editors substantially increase productivity of programmers
- Achieved through
  - Language-sensitivity
  - Automation of mundane tasks
  - Integration with other development tools

12



References

- D. Carlson. Eclipse Distilled. Pearson. 2005
- S. Dart et al. Software Development Environments. IEEE Computer 20(11):18-28. 1987
- N. Habermann and D. Notkin. Gandalf: Software Developement Environments. IEEE Transactions on Software Engineering. SE-12(12):1117-1127. 1986
- D. Knuth. Literate Programming. The Computer Journal 27(2):97-111. 1984

13