Software Tools, Methodology, Process and Environments

Topic 9 Environments 1

ICS 121

- Tools facilitate getting work done
  - analytical tools
  - software tools
- Methodology guides the proper use of tools
- Process helps to enact a methodology
- Environments are a synergistic collection of tools with process support

Analytical Tools

Topic 9 Environments 2

ICS 121

- Problem solving techniques that help in software development
  - cost-benefit analysis
    - » compare expected benefits against estimated costs
  - stepwise refinement
    - » divide and conquer
  - abstraction
    - » focus on some important property and ignore (for the time being) irrelevant details
- Analytical tools underly many software development methods

Software Tools and Methodology

Topic 9 Environments 3

ICS 121

- Software tool is an automated implement for performing work
- Tools facilitate work because they are
  - fast
  - immune to boredom
  - immune to "cheating"
- Methodology is indispensable as the guide to proper technique application
- Need a tool-supported methodology

Neither methodology nor tools alone are enough

Useful & Usable Software Tools

Topic 9 Environments 4

ICS 121

- U&U software tools have the following characteristics:
  - Powerful
  - Comfortable
  - Convenient
  - Natural
  - Reliable
  - Robust
- Most software development tools are not, but these are:
  - Compilers
  - Editors
  - Loaders
  - All have been polished, refined, debugged through long-term use
  - All have been made comfortable, effective through extended periods of exposure and maintenance
  - Work styles and methodologies have been molded by such tools

Tool Use Obstacles

Topic 9 Environments 5

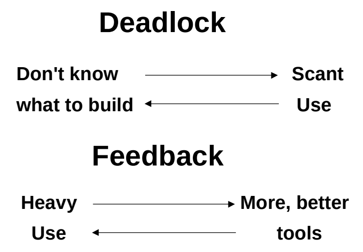
ICS 121

- Too many developers are like *hammer and screwdriver* carpenters
- Tools are not powerful
  - many tools are *wanna be* tools
- Tools are unreliable and/or not robust
  - comes with *time* and *money*
- Tools are uncomfortable, inconvenient, unnatural
  - comes from *successful use*

Tool Building

Topic 9 Environments 6

ICS 121



Tool Building Obstacles

- Short history of large-scale software development
- Limited success in
  - developing software well
  - exploiting "tools" successfully
- Few software techniques have had time and use required to achieve *tool* status
- No "tools" at all in some key areas (e.g., real-time analysis)

Underlying Problem

Not enough time and experience for feedback

––> No stability on effective lifecycle process and tool support

Software Engineering Environments

- An integrated collection of tools
  - Comprehensive
  - Open and Extensible
  - Flexible
  - Efficient
- Synergistic integration is the essence of an environment
  - Internal integration around internal objects
  - External integration around external tool usage and processes
- An experimental test bed for ideas, prototypes

CAD/CAM for software development

Environment Characteristics

- Broad in scope
  - Covers all lifecycle-related activities
    - » requirements, specification, design, implementation, testing, debugging, maintenance, configuration management, etc.
  - Supports all personnel
    - » technical (developers, reviewers, testers)
    - » managers (visibility
    - » clerical
  - Supports office automation
    - » mail
    - » document preparation

Environment Characteristics

- User Friendly
  - Highly interactive
    - » user has control
  - Graphics oriented
    - » uses windows, icons
    - » creates visual models
    - » animates development
- Flexible/Extensible
  - Addition of new tools
  - Addition of new object types
  - Addition of new functionality
  - Addition of new methodologies
  - Addition of new processes

Environment Focus

- Function-Centered
  - how, not what or why
  - * Interlisp, Unix, Mentor
- Object-Centered / Relation-Centered
  - what, not why or coordination
  - * STP, Rational
- Process-Centered
  - flexible, extensible, programmable
  - roles of personnel delineated
  - * Arcadia, Marvel
- (Domain-Oriented) Knowledge-Based
  - why, problem-domain programming
  - * gIBIS, DODE, KIDS

Environment Capabilities

- Description
  - software artifacts
  - relations between artifacts
  - graphical as well as textual
  - interactive manipulation
  - versions of artifacts
- Analysis
  - quality
  - pre-implementation: completeness, consistency, predicted performance
  - post-implementation: syntactic and semantic checking, verification and validation, performance monitoring
  - modification analysis / change propagation

Environment Capabilities

- Project Management
  - progress monitoring
  - resource allocation
  - standards and practices
  - access and modification control
  - documentation
- Technical Assistance
  - knowledge of software development
  - monitor modifications, discover inconsistencies
  - tutor user and guide process
  - adapt to individual style

Computer-Aided Software Engineering

- Commercially available environments that assist in most steps of development
  - carry out drudge work that typically bores developers and then gets slighted
    - » document organization
    - » diagram drawing and maintenance
    - » change management
    - » version control and configuration management
    - » metrics
- A CASE tool assists in one aspect of development
  - *upper CASE* assists in early life-cycle phases
  - *lower CASE* assists in later phases
  - generator technology supports rapid prototyping
- A CASE environment is a collection of tools that together support one or more phases of software development
  - not much of a process focus

[Process-Centered] Environment Infrastructure

- Process Specification and Interpretation
  - procedural, non-procedural
  - pro-active, re-active
- Object Management
  - types, relations, triggers, tool invocation
- Interoperability Mechanisms
  - event-based integration
- User Interface Management
  - direct manipulation
- Measurement and Evaluation
  - low overhead, flexible collection
  - feedback and improvement

[Knowledge-Based] Environment Infrastructure

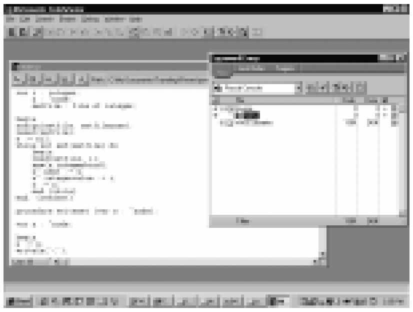
- Construction
  - building a system is supported by a palette of components
- Specification
  - textual, formal/semi-formal descriptions which parallel construction
  - construction and specification co-evolve
- Argumentation
  - alternatives clarify reasoning about problem, requirements, and design
- Simulation
  - verifies non-static properties
- Catalog
  - previously completed projects support current projects
- Explanation
  - catalog items must be explained, esp. their non-visual properties

ICS 121 Tool Environment

Discussion

- What tools would be useful for you in the following lifecycle phases?
  - requirements
  - design
  - formal specification
  - implementation
  - integration
  - test

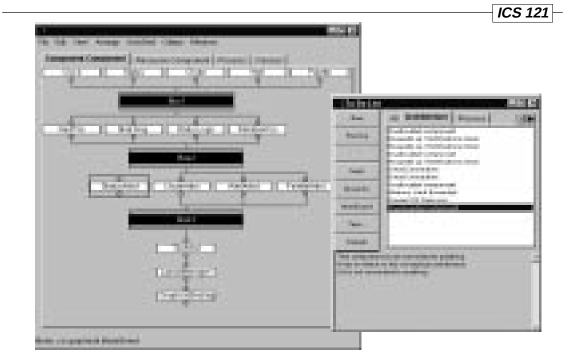
Integrated Development Environments



Topic 9 Environments 19

Software Architecture

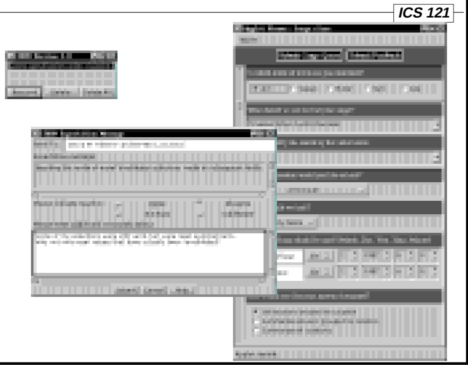
ICS 121



Topic 9 Environments 20

Usage / Event Monitoring

ICS 121



Topic 9 Environments 21

Design Rationale / Group Awareness

ICS 121