Software Engineering Tools and Environments: A Roadmap

William Harrison, Harold Ossher, Peri Tarr
IBM Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Introduction

- Tools and environments are good...
  - Without them, modern software engineering would be impossible
- ...but they need to be better
  - Increasingly crucial with increasing demands for and on software
  - We are pushing their limits as
    - rapidity of change increases; time-to-market decreases
    - software diversity and complexity increase; paradigms and standards shift constantly
    - new domains, with new requirements, emerge
    - new kinds of users attack
Roadmap of the Roadmap

- Brief history
- The paradox: Why isn’t every S/E using an integrated SEE?
- Context mismatch
- Permanently malleable software
- Separation and integration of concerns
- Deployment of commercial technologies
- Conclusions and soothsaying

BRIEF History

Common themes:
- Increasing need for integration
  - Data, control, process, GUI
- Increasing emphasis on openness, extensibility
- Increasing need for multiple simultaneous paradigms
  - Data, control, process; function, object, rule, ...
- Increasing need for full lifecycle support, consistency management
The Paradox

- Despite
  - Decades of R&D...
  - Plethora of tools, covering many S/E activities...
  - Much work on integration, openness, extensibility, ...
- and
  - Increasing volume and complexity of applications,
    implying increasing need for integrated SEEs
- primitive environments are still widely used
  ➔ The reality falls far short of the promise...

Context Mismatch

- SEEs and tools remain isolated and provincial
  - Novel capabilities entangled with context concerns
- This fact is at the core of
  - the inadequacy of SEEs and tools
    ➔ the problem
  - the inadequacy of software we create with them
    ➔ the meta-problem
Unpredictable Context Evolution

Software is like a square peg being forced into holes of ever-changing shapes

Pre-planned Software Evolution

Pluggable Components

Extensible Frameworks

Base Tool with Adapters for Concerns
And lots of built-in behavior for #4, #5, #6, #7, #8, #9, …
Pre-planned World Is Not Enough

- Planning for all eventualities is impossible
- Planning for too many costs too much at development time and run time
- Time-to-market pressure reduces planning

Permanently Malleable Software

Software is like clay: Mutable, then ossified

It needs to be like gold: Malleable for life

Morphogenic Software
Morphogenic Software Example

Sales Workflow System
John Doe
Worklist
1. Sales report
2. Forecast
3. Call Jill
Process

Employee Profile System
Personnel Dept
John Doe
Title
Special Abilities
Assignments

- New Requirement: Integrated e-trade system
  - Clients enter requests, assigned to brokers according to various criteria
  - Must work in distributed setting (or on new operating system, or ...)
- Challenges: Extract, adapt, integrate

Morphogenic Software

The pegs can change shape correctly to fit new holes as they arise.

- Morphogenic: The ability to evolve and adapt to uses in new or different (unanticipated) contexts, correctly and cleanly
  - Formal specification, and automatic checking
  - Extraction and integration
  - Mutability for life
    - Readily, non-invasively, correctly adaptable
    - Polymorphic and metamorphic
- Across all software artifacts, perhaps without source
Extraction and Integration

- Modules are suitable units of reuse, but
- The concerns to be extracted and reused were not separated into modules
  - Code is scattered over many modules, and tangled with code for other concerns
  - Extraction and reuse are invasive (even impossible)
- Need the right modularization at the right time

Prevented by the
Tyranny of the Dominant Decomposition

Objects: a Dominant Decomposition

<table>
<thead>
<tr>
<th>Tax Regions</th>
<th>Name/ Address</th>
<th>ID Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concern</td>
<td>Employee</td>
<td>Employee</td>
</tr>
<tr>
<td></td>
<td>Address</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>changeName</td>
<td>changeName</td>
</tr>
<tr>
<td></td>
<td>changeAddress</td>
<td>changeAddress</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Number Hours</th>
<th>Wage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>changeName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>changeAddress</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Breaking the Tyranny

- **Multi-Dimensional Separation of Concerns**
  - Identification and encapsulation of all kinds of concerns of importance
    - Identified incrementally at any time
    - Can span artifacts and lifecycle phases
  - Simultaneous decomposition into “concern modules”
    - Multiple decompositions co-exist, new ones can be added
  - Composition of concern modules
  - Handling of overlapping and interacting concerns

- **On-Demand Remodularization**

Advanced Separation of Concerns

- **Growing research area**
  - AOP, Adaptive Programming, Hyperspaces, Viewpoints, ...
  - The time is right
  - Still a long way to full Multi-Dimensional SOC

- **Key issues**
  - Identification, visualization, encapsulation and integration
  - Overlapping and interacting concerns
  - Specification and checking
  - Concerns across the lifecycle; traceability

**Tool and environment support are essential**
Morphogenic Software

- Many, multi-disciplinary challenges, such as:
  - Understanding evolutionary pressures and responses
  - Managing dependencies and interactions
  - Sophisticated, unplanned adaptation
  - Correctness and consistency management
- Some foundations
  - Advanced separation of concerns
  - Packaging mismatch
  - Component technology
    - Specification, adapter generation, ...
  - Integration technologies, especially commercial standards

Deployment of Standardized Commercial Technologies

- Data Integration -- XML
- Repository Integration -- Enterprise Java Beans
- Control Integration -- Message Brokering
Data Integration pre-XML

Data Integration -- XML --

Copyright © 2000, IBM Corporation  William Harrison, Harold Ossher and Peri Tarr  ICSE 2000 FOSE 19
Data Integration -- XML

Repository Integration -- Enterprise Java Beans
Control Integration -- Message Brokering

The Shape of Things to Come

- Advanced separation of concerns and morphogenic software
  - Advances in SOTA and standards suggest feasibility
- Non-traditional software lifecycles
- Challenging new domains

Integrated tools and environments are critical ...
... and are better enabled by

- Adoption or adaptation of commercial integration standards
  - E.g., XML, EJBs, message brokering
  - May succeed where others have failed because they do not impose models, implementations, services