Purpose

- Present progress on my 17-939 major project:
  - A study of the technology maturation and research strategies for Integrated Development Environments (or tools for software engineering)
- Present my “plan of attack” to solicit feedback and advice:
  - Technology maturation
  - Research strategies

Outline

- Status
- Plan of attack: IDE maturity
- Plan of attack: research strategies
- Discussion…

(Useful) Interaction encouraged, within our allotted time, please focus on approach – not details

Status

- Initial literature review: SE tools/environments
  - Clarified topic
    - Focus-in on Integrated Development Environments
  - Enough work to put a “plan of attack” together
  - Focus is helping find relevant research literature
- Reality more complex than my notional model
- Way-Ahead:
  - Execute plan of attack
  - Draft of paper next critical step (mid-Nov)
Plan of attach: IDE maturity

- Problems
- Timeline
- Functional maturity
- IDE maturity (Redwine/Riddle85)
- Way ahead

Problem: definitions
(What counts as an IDE?)

- What qualifies as a programmer Tool?
  - Line across card decks
  - Post-it notes
  - Focus: automated tools (editors, debuggers, etc.)
- What qualified as an Integrated Environment?
  - The computer room & building with operators
  - PL-I, ISPF/cc, vi, make / Turbo Pascal 1.0
  - Focus: a group of tools that synergistically work together

Problem: context
(Shared knowledge & changing OS)

- Older papers exist in a context difficult to map into over a decade later
  - Shared knowledge has changed significantly
  - The OS “environment” is taken as static for IDE work – OS more fluid
  - Less of a problem for me after 1987
- Example: Dolotta76 before Bianchi76

Timeline: OS environment

- A gross oversimplification but useful to understand IDE maturity
**IDE timeline: readings**

1945 Eniac

1960 PDP-1

1973 Alto

1975 Altair

This approach let me map research and commercial practice on the same chart

**IDE functional maturation**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen / GUI editor</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>Pen</td>
</tr>
<tr>
<td>Syntax aware editor</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>ガンダリ</td>
<td>YES</td>
</tr>
<tr>
<td>Integrated compiler</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>EPDE3</td>
<td>YES</td>
</tr>
<tr>
<td>Integrated interpreter</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>ARCADIA</td>
<td>YES</td>
</tr>
<tr>
<td>Integrated Debugger</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td></td>
<td>YES</td>
</tr>
<tr>
<td>Integrated Profiler</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td>YES</td>
</tr>
<tr>
<td>Integrated source code control</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>Integrated documentation tools</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>

**IDE maturation: readings**

**IDE maturity: way ahead**

- Continue to dig into the topic as discussed in my previous slides
- Focus research search and industry search towards building a solid maturity timeline
  - Build a clear picture of technology maturity
  - Avoid sidetracks (formal methods, design tools)
    - Harder to do than to say
    - Less mature, but some investigation may be unavoidable
- Draft final paper
Plan of attachment: research strategies

- Overview
- Pro-forma/strategy (readings)
- Problems
- Research strategy evolution (readings)
- Way ahead

Research strategies overview

- 2nd part of my project is to evaluate the evolution of IDE research strategies
- Approaches:
  - Newman(95) Pro-forma abstracts
    - Doesn’t break out components of research strategy
    - Maps out some of the “valid” space in the Shaw model
    - Pro-forma helpful to map paper into Shaw model
  - Shaw(00-evolving) research strategy model
    - Better, I think, for this purpose
    - I take this model as the foundation of my approach

Pro-forma/strategy: Bianchi76

- Pro-Forma: Radical Solution
  - A radical solution to the problem of <software development> is described, based on <using Unix on a separate non-target computer to integrate a set of practical programming tools (Programmer’s Workbench)>.
  - In comparison with <flowchart, coding form, keypunch, deck to the computer center>; it offers <(1) increased productivity, (2) better code style/readability, (3) assistance with documentation of code changes, (4) metric support>, all of which have been demonstrated in preliminary tests. It has a number of deficiencies <(1) poor environment documentation, (2) file system management, and (3) not on-line on target machine>. Strategies are suggested for addressing these deficiencies.
- Question: Method/Means
- Result: Tool
- Validation: Experience (persuasion)

Pro-forma/strategy: Teitelbaum81

- Pro-Forma: Enhanced Tool
  - The effectiveness of <automated tools> in supporting the <implementation of software> has been demonstrated. An enhanced tool, <the Cornell Program Synthesizer>, is described for the <implementation of software> based on <an integrated set of automated tools, including (1) syntax directed editing, (2) full-screen editing, (3) integrated debugging, and (4) program interpretation>.
  - Examples are provided confirming the effectiveness of its support for <software development>.
  - The Cornell Program Synthesizer takes the point of view that programs are not text; they are hierarchical compositions of computational structures and should be edited, executed, and debugged in an environment that consistently acknowledges and reinforces this viewpoint.
- Question: Method/Means
- Result: Tool
- Validation: Experience (narrative)
Pro-forma/strategy: Teitelman84

* Pro-Forma: Enhanced Tool
  - The effectiveness of integrated tools in supporting the implementation of software has been demonstrated. An enhanced tool, Cedar, is described for the implementation of software based on an integrated set of automated tools, including (1) an integrated graphical environment, (2) full-screen editing, (3) integrated debugging, (4) program interpretation and compilation, and (5) extensive documentation support. Examples are provided confirming the effectiveness of its support for software development. The Cedar project is one of the first interactive, experimental programming environments targeted at a compiled language: the Cedar language (based upon Mesa).

  * Question: Method/Means
  * Result: Tool
  * Validation: Experience (narrative)

Pro-forma/strategy: Osterweil87

* Pro-Forma: Radical Solution
  - A radical solution to the problem of software engineering is described, based on viewing processes as computer programs. In comparison with traditional views of business process documentation, it offers (1) a vehicle for the materialization of software development and maintenance processes, (2) reuse of these processes, (3) syntactic checking of documented processes, and (4) a basis for integrating software tools with human process activities. It has the deficiency that no standard “process” language exists. Strategies are suggested for addressing this deficiency. Software process description, via “process programs”, should become a key focus of software engineering.

  * Question: Feasibility
  * Result: Technique
  * Validation: Persuasion

Problem: how many dimensions?

* Shaw model has three dimensions (Q,R,V):
* To evaluate strategy evolution we need to add time:
  - Now 4 dimensions
* To spot dead-ends we need today’s “impact”
  - Now 5 dimensions
  - Impact vs. immaturity?
  - Practice vs. research?

New dimensions: time

* Time
  - Calendar time seems to be most appropriate
  - Easy to determine
    - Paper publication (my choice)
    - When paper says “work” occurred (another choice)
**New dimensions: impact**

* Impact (on area of interest: IDEs)
  - Impact must be evaluated from the perspective of “our-time” not the perspective of “research-time”
  - Impact can come in two major forms:
    • Impact on ongoing research today
    • Impact on practice today
    - Question: Is this really a measure of how well the research problem mapped to the real-world problem?
  - An unknown impact (dropped research & not in practice) indicates a dead-end or a lack of maturity
  • I am unwilling to propose a decision criteria

**Impact categories**

* Unknown (dead-end or immature)
  - Implies little/no research and little/no practice
* Research (ongoing today):
  - Incremental contribution
  - Seminal
* Practice (today):
  - Minority application
  - Majority application
  - Ubiquitous (95%+)

Here are my impact categories
-Note that an area have impact in research and practice at the same time

**Research strategy evolution: readings**

<table>
<thead>
<tr>
<th>Research Strategy</th>
<th>Time</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Questions</td>
<td>Results</td>
<td>Validation</td>
</tr>
<tr>
<td>Osterweill87</td>
<td>Feasibility</td>
<td>Technique</td>
</tr>
<tr>
<td>Teleman84</td>
<td>Method/ Means</td>
<td>Tool</td>
</tr>
<tr>
<td>Telebaun81</td>
<td>Method/ Means</td>
<td>Tool</td>
</tr>
<tr>
<td>Bianchi76</td>
<td>Method/ Means</td>
<td>Tool</td>
</tr>
</tbody>
</table>

**Research strategy evolution: analysis**

* Data can be looked at (some ideas):
  - Research strategy by maturity phase
  - Research strategy by age or phase
  - Research strategies can be examined by impact
  • Age/Within a maturity phase/Across multiple phases
* Need to come up with charts that show data clearly
  - Excel not very helpful…so far
Research strategies: way ahead

* Continue to dig into the topic as discussed in my previous slides
* Evaluate research papers used to build maturity timeline
  – Map into Shaw model with time and impact added
  – Analyze results
    • What can we say about the evolution of IDE research strategies across the maturity timeline
* Draft final paper

Discussion

* Questions or Comments?