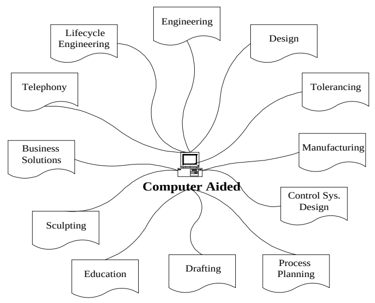
CASE: Past, Present, and Future

David Notkin
Dept. of Computer Science & Engineering
University of Washington
May 1997
www.cs.washington.edu/homes/notkin

UW CSE
CASE (ICSE 97)

Computer **A**ided **S**oftware **E**ngineering

Using computers to help people engineer software



What is CASE?

- Broadly construed, CASE is any software system that helps in any software engineering task
  - Configuration management tools
  - Test tools
  - grep, make, emacs …
  - …
- But this is too broad to be useful
  - Roughly, CASE comprises environments that support software engineering

Environments vs. tools

- Integrated
  - User interacts with a single environment
  - Environment is responsible for managing consistency
- Sharing of representation

- Standalone
  - User interacts with each tool separately
  - User must apply tools appropriate to ensure consistency
- Independent representations

Why environments?

- "During the past decade there has been a growing realization [that software tools] have by and large failed to reduce cost and improve quality. … [T]he essence of an environment is that it attempts to redress the failures of individual software tools through synergistic integration."
  – Osterweil, 1981

Why environments?

- "Current software development environments often help programmers solve their programming problems by supplying tools such as editors, compilers, and linkers, but rarely do these environments help projects solve their system composition or management problems."
  – Notkin & Habermann, 1979

What are environments?

- "A software development environment consists of a set of techniques to assist the developers of software systems, supported by some (possibly automated) tools, along with an organizational structure to manage the process of software development. Historically, these facilities have been poorly integrated."
  – Wasserman, 1981

Where might the computer help?

- Interaction and rich user interfaces
- Translation of high-level descriptions
- Maintaining consistency among large and complex representations
- Encoding knowledge about an activity, organization or process
- Broader availability of pertinent information
- Communication medium

CASE is ...

- "The CASE technology is a combination of software tools and methodologies. … Spanning all phases of the software lifecycle, not just on implementation, CASE is the most complete software technology yet."
  – McClure, 1989

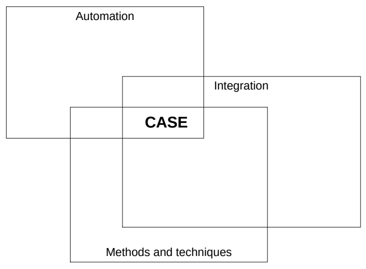
CASE is ...

- "To be truly successful, a CASE product must literally support application development from womb to tomb, from the initial conception of an application through its long-term maintenance."
  – Towner, 1989

CASE is

- "CASE provides the rigorous automated support required to build flexible, high-quality information systems quickly."
  – Mylls, 1994

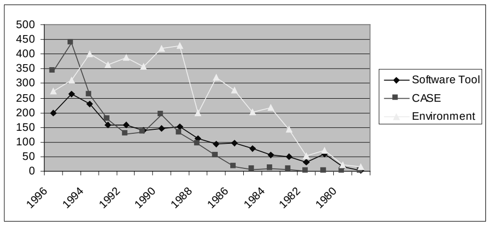
CASE is … (Stone, 1993)



CASE confusion

- Environments in academia, CASE in industry
- CASE in information systems & information technology
  - The MIS world of CIOs
  - Application Development (AD)
  - Information Engineering (IE)
  - Methodology plays a key role
- Upper CASE vs. lower CASE

Appearances in INSPEC



- Programming *environment*, program development *environment*, software development *environment*, software engineering *environment*, integrated program support *environment*

ICSE 2 (1976)

- "Environment"
  - None
- "Tool"
  - 2 technical papers
- "CASE"
  - None

ICSE 9 (1987)

- "Environment"
  - 3 technical papers, 2 panels
- "Tool"
  - 1 technical paper
- "CASE"
  - None

ICSE 10 (1988)

- "Environment"
  - 6 technical papers, 1 panel
- "Tool"
  - 7 technical papers
- "CASE"
  - None

ICSE 97

- "Environment"
  - 1 technical paper, 5 demos, 1 panel
- "Tool"
  - 1 technical paper, 3 demos, 1 experience report
- "CASE"
  - just this presentation

An environment taxonomy

- Language-centered environments
- Structure-oriented environments
- Toolkit environments
- Method-based environments

  – Dart, Ellison, Feiler, Habermann, 1987

(A few) classic environments

- Interlisp
- Smalltalk-80
- Unix
- Cedar

Interlisp (Xerox PARC)

- Teitelman & Masinter, 1981
- Language-centered environment
- Very fast turnaround for code changes
- Monolithic address space
  - Environment, tools, application code commingled
- Code and data share common representation

Smalltalk-80 (Xerox PARC)

- Goldberg, 1984
- Language-centered environment (OO)
  - Classes as first-class objects, inheritance, etc.
- Environment structured around language features (class browsers, protocols, etc.)
- Rich libraries (data structures, UI, etc.)

Unix (Bell Labs)

- Toolkit-based environment
- Simple integration mechanism
  - Convenient user-level syntax for composition
- Standard shared representation
- Language-independent (although biased)
- Efficient for systems' programming

Cedar (Xerox PARC)

- Teitelman, 1984
- Intended to mix best features of Interlisp, Smalltalk-80, and Mesa
- Primarily was an improvement on Mesa
  - Language-centered environment
  - Abstract data type language
    - » Strong language and environment support for interfaces
  - Key addition: garbage collection

(Some) key research environments

- Toolpack
- Gandalf
- Mentor
- Cornell Program Synthesizer/Generator
- Arcadia
- Pecan

Toolpack (Osterweil, 1983)

- Consider breadth of tools needed for software development (static analysis and testing tools, documentation, etc.)
- "Tool fragments" to support tight integration of tools into an environment
- Centralized tree-structured file system for sharing data
- Focus on mathematical software

Gandalf (Habermann, Notkin, et al.)

- Structure-based environments
  - Direct-manipulation of program objects
- Environment generation
  - Integration through implicit invocation by active abstract syntax trees
  - Shared database structured on ASTs
- Higher-level, closer to task
  - User focuses on "what" not "how"

Mentor

- Donzeau-Gouge, Huet, Kahn, Lang, Levy, 1984
- Structure-based environment
- Users could define dynamic views
  - General-purpose tree manipulation language
- Formal basis for semantic definition
  - Led to Centaur generation system
  - Used natural semantics

Cornell

- Program Synthesizer [Teitelbaum & Reps, 1981]
  - Syntax-based editing environment
  - Text at expression-level
- Synthesizer Generator [Reps & Teitelbaum, 1984]
  - Generation of syntax-based editors
  - Based on definition of attribute grammars
  - Incremental attribute grammar update algorithm [Reps 1983]

Arcadia (R. Kadia, 1992)

- Process-based environment
  - Process definition and execution
- Analysis and testing
- Measurement and evaluation
- UI development and management
- Event-based integration
- Typed object-base

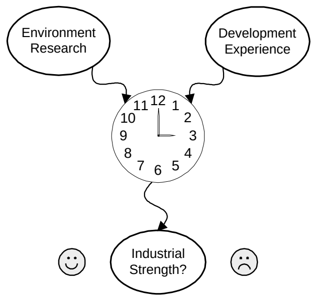
Pecan (Reiss, 1984)

- Graphically-based environment
- Multiple, concurrent views
  - Data structures
  - Symbol table
  - Flowchart
  - Nassi-Shneiderman diagrams
- Many views read-only

Structure-oriented editors

- These failed because they
  - unnecessarily and overly constrained the users
    - one editor, one style
    - hard to integrate external tools
    - not bad for novice environments
  - tried to reduce the wrong cost
    - getting syntax (and static semantics) right isn't such a big deal
  - costly to produce (even after tons of research)
- Some language-oriented features have flourished
  - Ex: Color to mark constructs in editors

Present Status of CASE?



Commercialization

- 22 companies matched "CASE" in Company Profiles database
  - About 10,000 matched "software"
  - 23 matched "application development"
- 3 Yahoo CASE categories
  - 55-60 registered CASE pages in Yahoo
  - (35 Java categories, thousands of pages)

The business of CASE

- IDE (Software through Pictures)
  - Founded 1983
  - Acquired by Thomson-CSF 1996
    - ~$10M annual sales
- Rational
  - Founded 1982
  - Recently purchase Pure Atria
    - ~$91M annual sales (+ ~$132M annual from Pure Atria)

The business of CASE

- Popkin
  - Founded 1986
    - ~$15M annual sales
- Cayenne Software, Inc. (1996)
  - Merger of Bachman (1983) and CADRE (1982)
    - ~$14M annual sales
- StructSoft (TurboCASE/Sys)
  - Formed 1984
    - ~$6M annual sales

The business of CASE

- I-Logix
  - Founded 1987
    - » ~$10M annual sales
- Reasoning Systems
  - Founded 1984
    - » ~$20M annual sales

Some context

- SAP AG
  - R/3 Business Process (Re)Engineering
  - Founded 1972
    - » >$2B annual revenue
- Largest “application development” company in Company Profiles
  - Progress Software, 1981
  - ~$180M annual revenue

CASE quotations

- “Despite the many grand predictions of the trade press over the past decade, computer-assisted software engineering (CASE) tools failed to emerge as the promised `silver bullet.’”
  - Guinan, Cooprider, Sawyer; IBM Systems Journal, 1997
- “CASE tools are sometimes excessively rigid in forcing the user to input too much information before giving usable results back. CASE tools also typically don't adapt to multiple or in-house methodologies…”
  - www.confluent.com; 1997

Myth #1 of CASE

- *Integration is job #1*
- Integrating tools helps, but only if the tools are the “right” tools
- That is, integration is a second order effect, not a first order effect

Myth #2 of CASE

- *Graphics inherently dominate text*
  - “A picture is worth a thousand words”
- This is a complicated issue
  - Screen real estate
  - Sharing with other tools
  - Sharing with other people

Myth #3 of CASE

- *Software tools are more important than intellectual tools*
- False
  - Software tools are important but are generally a second order effect
  - Sometimes software tools can qualitatively change the world, although usually indirectly

Myth #4 of CASE

- *Tool adoption is a consumer problem not a producer problem*
- False
  - See my flame in WOW

Organizational issues (Orlikowski)

- "CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development"
  - MIS Quarterly Best Paper, 1993
- "[To] account for the experiences and outcomes associated with CASE tools, researchers should consider the social context of systems development, the intentions and actions of key players, and the implementation process following by the organization."

Myth #5 of CASE

- *Goal should be to change how software engineering is done*
- No, it should be to enhance how people are doing software engineering

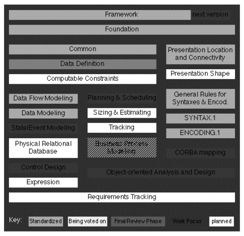
Myth #6 of CASE

- *The tools can handle creative aspects of software engineering*
- Tools frequently fail to be useful because they make poor judgments about what the human does well and what the computer does well

Technical problems

- Integration
- Representation
- Views
- Multiple languages
- External tools
- ...

CDIF: a standard (EIA)



- **C**ASE **D**ata **I**nterchange **F**ormat
  - EIA/CDIF 1994
- Members of the standards committee include
  - Boeing, Ford, ICL, Rational, IBM, and a dozen or so more

So?

- Should we stop research and development in environments and CASE?
- Certainly not
  - Good solutions to the technical issues could have broad impact
  - The underlying motivations for CASE still remain
- But the Holy Grail is farther away, not closer

The near future?

- “Lightweight” CASE tools may make significant headway
  - Visio, Visual Thought, etc.
  - Hundreds not thousands of dollars
- Start with a syntactic diagramming tool
  - Add value through added shapes
  - Integration with other tools through COM/OLE
- How far can they get?

Current projects of interest

- Desert (Brown, Reiss)
  - `www.cs.brown.edu/software/desert/`
- IP (Microsoft, Simonyi)
  - `www.research.microsoft.com/research/ip/main.htm`
- Atlantis (Columbia, Kaiser)
  - `www.psl.cs.columbia.edu/atlantis/atlantis.html`
- Endeavors (UCI, Taylor & Redmiles)
  - `http://www.ics.uci.edu/pub/endeavors/`
- …what else…?