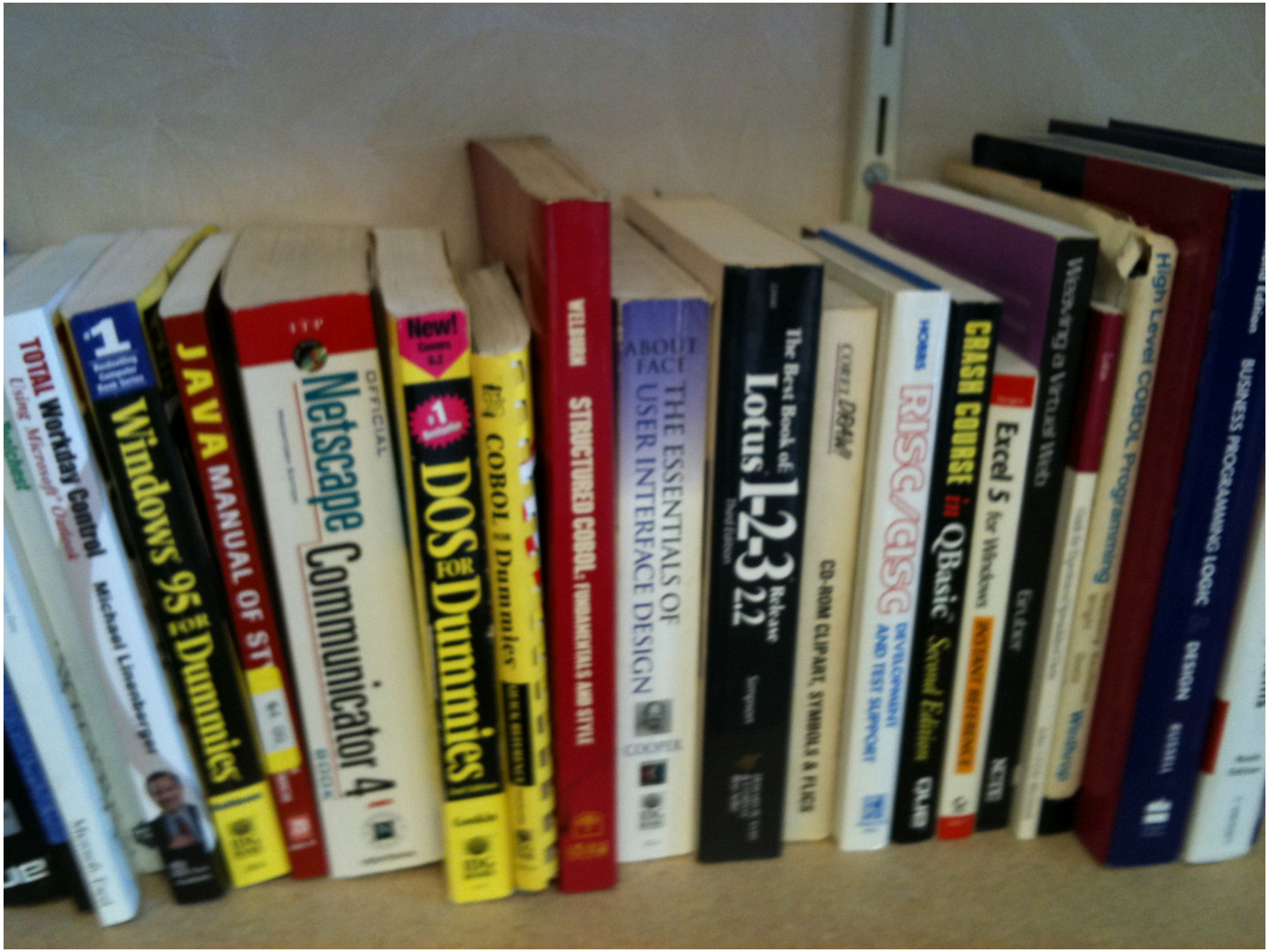
SEMAT Language Track

Language track proposal

- General introduction
- Each participant may provide a short statement on the language aspects of software enginering
  - ✓ 1 minute maximum
  - ✓ may be one sentence
  - ✓ may be five keywords
  - ✓ may be controversial
  - ✓ may be consensual
- Of course, the statements will be related to the SEMAT vision paper
- All statements will be recorded and send back to participants (scriber JM Favre).

Software technology changes rapidly



18 ± 3 Software Technology Maturation

- « *The magic number Eighteen Plus or Minus Three* », William E. Riddle, ACM Sigsoft, April 1984
- 15 to 20 years to mature a technology to the point that it can be popularized and disseminated to the technical community at large

Cost Models

0-1966- appearance of first collection of cost-related data
1-1976- appearance of a usable system (Price S)
2-1978- alternative systems are available (for example, COCOMO)
3-1981- publication of Boehm's text

Smalltalk-80

0-1965- Kay's thesis defines concept of a personal computerized notebook
1-1972- preliminary version of Smalltalk is available
2-1976- major new version of Smalltalk appears
3-1981- other companies start porting the Smalltalk-80 system to their computers
4-1983- Smalltalk-80 available as a commercial product

SREM

0-1968- ISDOS system demonstrates applicability of attribute-value-relation approach to pre-implementation activities
1-1973-74- first concrete definition of the SREM system appears
2-1977- first release of the SREM system
3-1981- Vax version available

Unix

0-1967- appearance of the Multics system
1-1971- initial versions of Unix available
2-1973- Unix system debuts at Sigops conference
3-1976- collection of papers appears and system begins to be widely used in academic community
4-1981- announcement of Unix System III

0 1 2 3 4
4 — Substantial Evidence of Value and Applicability
3 — Shift to Usage Outside of Development Group
2 — Usable Capabilities Available
1 — Definition Via Seminal Paper or Demonstration System
0 — Emergence of Key Idea

Knowledge-based Systems

0-1965- appearance of artificial intelligence systems providing intelligent assistance (for example, Dendral)
1-1973- appearance of systems containing a knowledge base (for example, Hearsay)
2-1970-00- appearance of knowledge-based systems that can be routinely used for problem-solving tasks (for example, R1)

Software Engineering

0-1960- inadequacy of existing techniques for large-scale software development noted in several projects (for example SAGE)
1-1968- concept of software engineering is articulated at Workshop on Software Engineering at Garmisch Partenkirchen
2-1973-74- general collections of papers appear and policy guidelines are established in various communities
3-1978-79- texts and generally usable systems supporting software engineering appear (for example, the SREM system)
4-1983- use of software engineering shifts to a larger community through actions such as the DeLauer directive and the definition of a Software Engineering Institute

Verification

0-1966- Floyd's paper on program correctness analysis
1-1971- King's demonstration system appears
2-1975- multiple systems are available
3-1979- usage of some systems shifts to application groups

Compiler Construction

0-1961- Iron's paper on compiler generation
1-1967- review paper by Feldman and Gries
2-1970- usable systems appear (such as the XPL system at Stanford)
3- - (cannot be determined)
4-1980- appearance of production-quality compiler-compilers

Metrics

0-1972- publication of book on Halstead metrics
1-1977- results of trying to measure various empirical and analytic measures appear

Abstract Data Types

0-1968- initial report on information hiding
1-1973- appearance of some languages using idea of abstract data types (for example, TOPD design language)
2-1977- major publication on the subject and frequent appearance of the concept in new programming languages (for example, CLU)
3-1980- use of abstract data types in other technologies (such as in the Affirm program verification system)

Structured Programming

0-1965- Dijkstra's paper on programming as a human activity
1-1969- paper on structured programming by Dijkstra at the First NATO-sponsored Workshop on Software Engineering
2-1972-73- concept is widely discussed and presented in papers
3- - (cannot be determined)
4-1976- publication of first introductory text based on structured programming

SCR Methodology

0-1968- appearance of concepts such as information hiding and communicating sequential processes
1-1976- completion of feasibility demonstration by NRL with positive experiences
2-1978-79- appearance of training material and models of usage
3-1982- methodology moved to a variety of other organizations

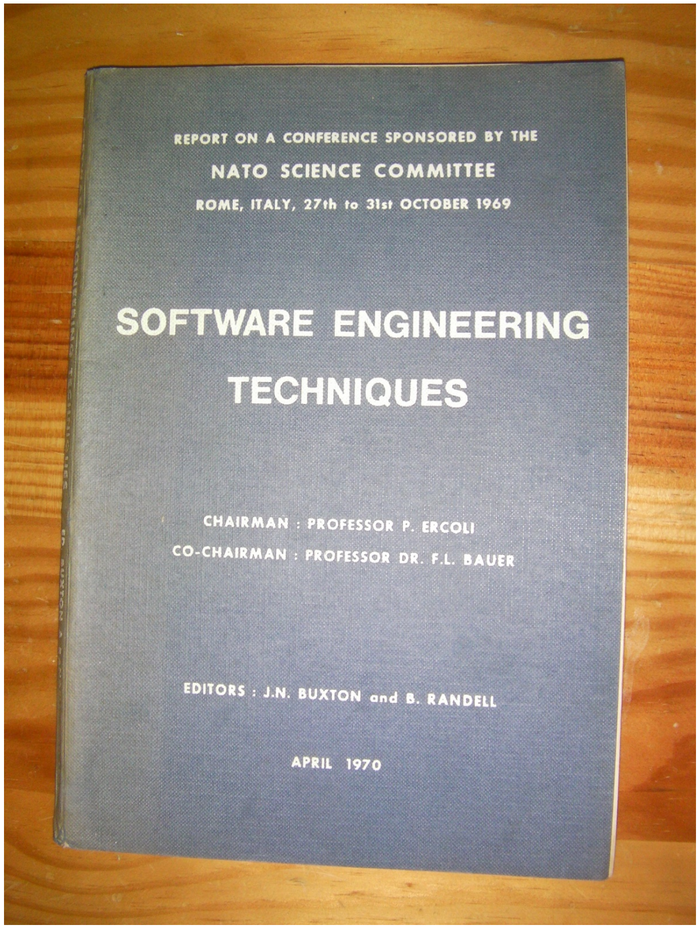
DOD-STD-SDS

0-1967- initial articulation of phased approaches to software development
1-1980- contract signed for development of DOD-STD-SDS

AFR 800-14

0-1972- basic need for policy and specific guidance is documented
1-1973- initial draft policy is published
2-1974- policy guidance is published
3-1974- final draft is available
4-1975- regulation and instructions for its use are officially published

Only 3 technology maturation cycles



Software Engineering

0-1960- inadequacy of existing techniques for large-scale software development noted in several projects (for example SAGE)

1-1968- concept of software engineering is articulated at Workshop on Software Engineering at Garmisch Partenkirchen

2-1973-74- general collections of papers appear and policy guidelines are established in various communities

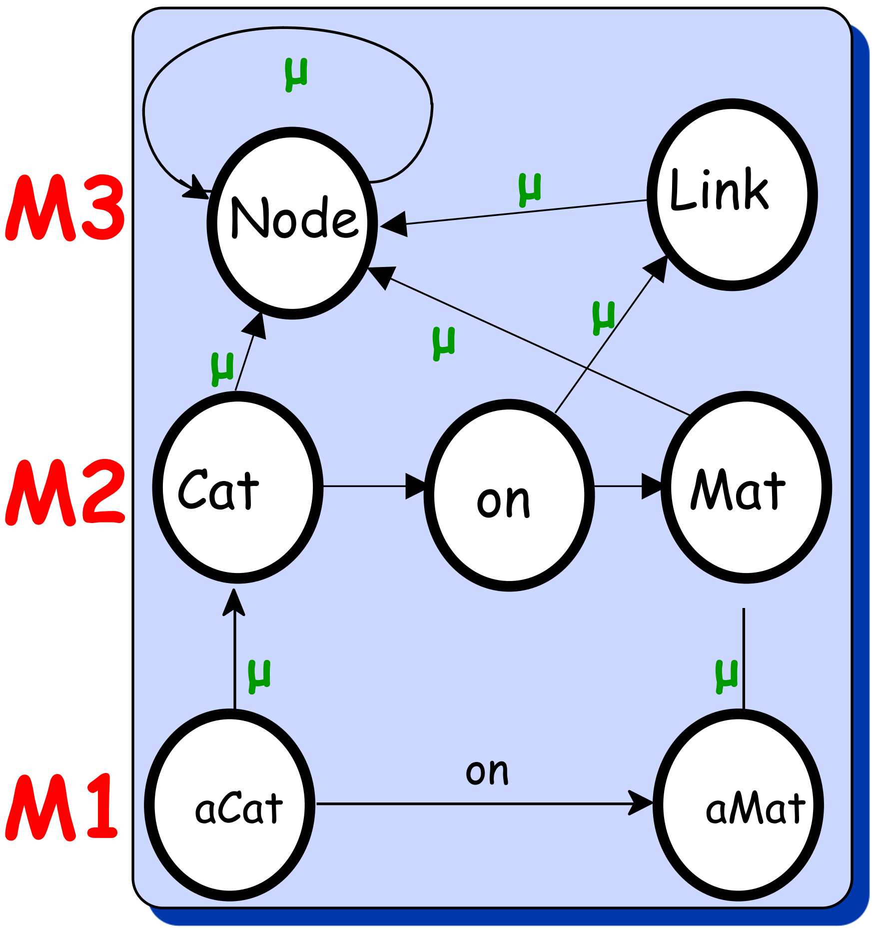
3-1978-79- texts and generally usable systems supporting software engineering appear (for example, the SREM system)

4-1983- use of software engineering shifts to a larger community through actions such as the DeLauer directive and the definition of a Software Engineering Institute

Which Languages for Software Engineering?

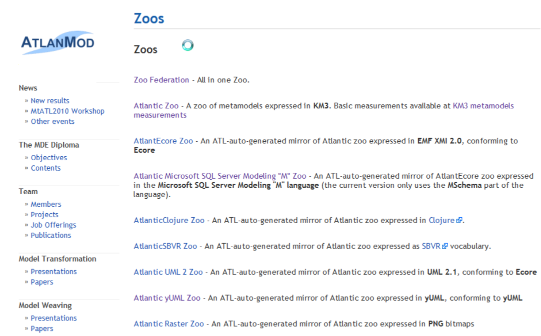
- ✓ Procedural, functional, object-oriented, rule-based?
- ✓ Formal (Z, B, VDM, Petri) or not (Basic)
- ✓ General Purpose or Domain Specific (DSLs)
- ✓ Executable or Non-Executable
- ✓ For process or product
- ✓ For business or IT (e.g. BPMN & UML)
- ✓ For professional (Eiffel) or end-users (Excel)
- ✓ For objects, rules, events, process, goals, etc.
- ✓ For code or data
- ✓ Normative or Proprietary
- ✓ Textual, Visual, Tabular, Form-based, …
- ✓ Grammar-based, metamodel-based, schema-based, …
- ✓ etc.

Various representations



Metalanguages (EBNF, XML Schema, MOF, ECORE, …)

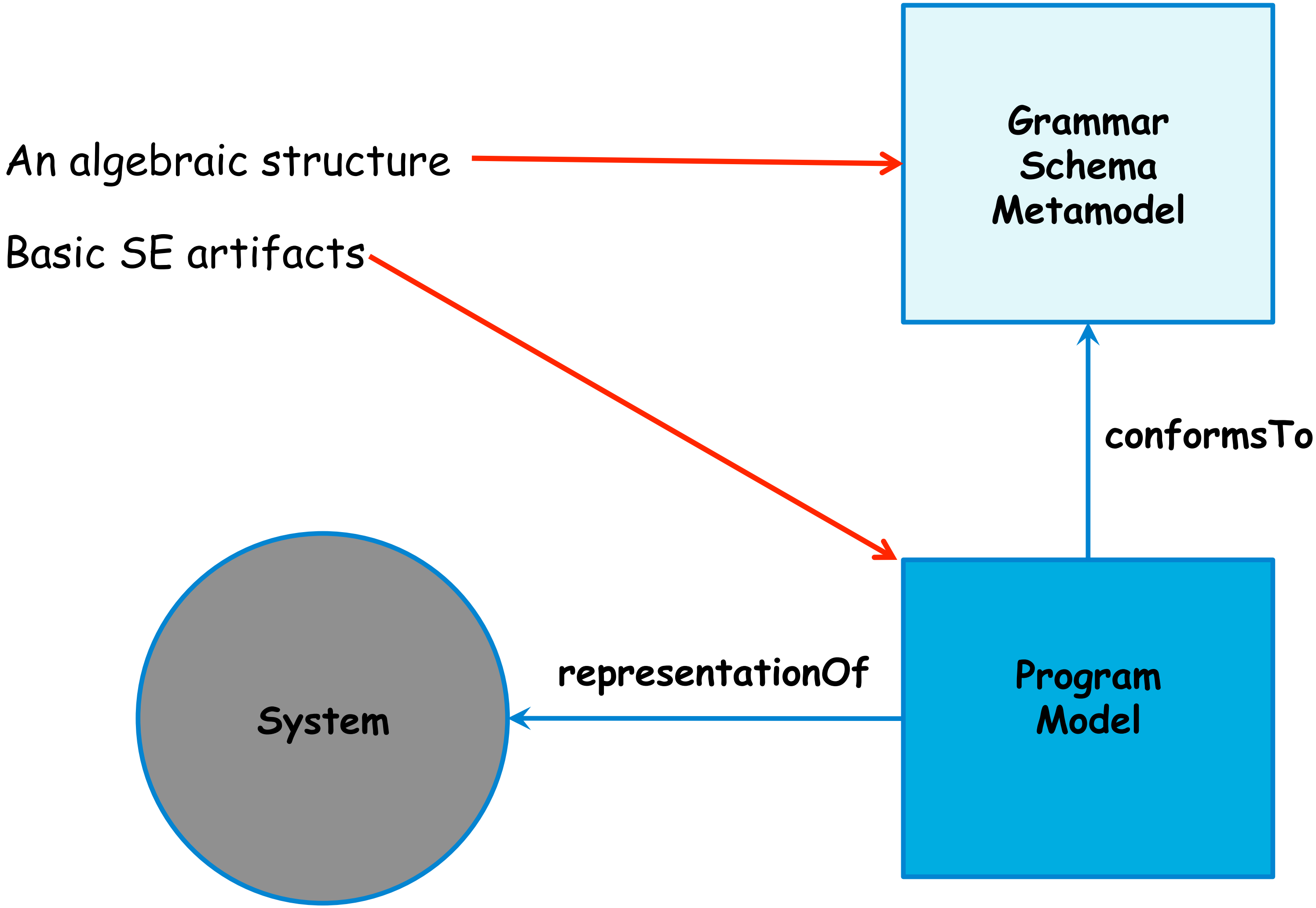
Language Libraries, Repositories



Basic artifacts (programs, etc.)

$\exists x, \exists y : Cat(x) \wedge Mat(y) \wedge on(x,y)$

Representation and Conformance



Taking the *representation* relation seriously

"*What about the [relationship between model and real-world]? The answer, and one of the main points I hope you will take away from this discussion, is that, at this point in intellectual history, we have no theory of this [...] relationship*".

Brian Cantwell Smith **The Limits of Correctness**;
a paper prepared for the Symposium on Unintentional Nuclear War, Fifth Congress of the International Physicians for the Prevention of Nuclear War, Budapest, Hungary, June 28 - July 1, 1985.

See also "***On the origin of objects***"

Robin Milner's Grand Challenge

Language is the raw material of software engineering, rather as water is the raw material for hydraulic engineering…

A more thorough science-based approach to informatics and ubiquitous computing is both necessary and possible. We often think in terms of models, whether formal or not. These models, each involving a subset of the immense range of concepts needed for ubiquitous computer systems, should form the structure of our science…

Even more importantly, the relationships (either formal or informal) among them are the cement that will hold our towers of models together. For example, how do we derive a model for senior executives from one used by engineers in designing a platform for business processes, or by theoreticians in analyzing it?