An Assessment of Reuse Technology after Ten Years

James M. Neighbors
SADA

November 1994

Overview

- 11th anniversary of 1983 ITT Workshop on Reusability in Programming

- "The Draco Approach to Constructing Software from Reusable Components"

- Goals of this talk: reuse conference
  - concept rationalization for researchers, not a survey
  - questions for practitioners on what you will see here
  - predict next 10 years

- Major changes in the last 10 years
  - new very large modeling abstractions (GUIs, networks, database)
  - system developer bridges abstractions from problem domain to modeling abstractions

Libraries of Reusable Artifacts

- Black box (no modification) vs. white box (modification) reuse
- Black box reuse issues
  classification (abstraction), integration & search (selection)
  problems
- White box reuse additional issue
  specialization problem
- Granularity of reuse
  large grain versus small grain
- Basic library technology is mature.
- Libraries are a good first step
  simple approaches have serious limiting problems.
  minimum 15% cost improvement.

Library Problems

- Independent of artifact level of abstraction

- Primary failure of library concept.
  - easy to add an artifact to the library.
  - burden of search is placed on every user of the library.
  - burden of integration is placed on every user of the library.
  - burden of specialization is placed on every user of the library.
  - library parts are not explicitly connected.

- Library similar to large abstraction (say GUI) documentation.
  - hyper browsing is not enough - does not scale
  - protocols of use are required

- Book library analogy failure
  - science citation index a better model not card catalog.
  - searchable connections versus topics

What to Reuse

- Software Engineering lifecycle phases and % of effort [Boehm]
  requirements 6%, analysis 8%, design 29%,
  implementation 34%, testing 23%
  maintenance a reiteration

- Later phase artifacts based on earlier phase artifacts

- Reuse artifacts from as early in lifecycle as possible.
- Reuse requirements, analysis, and design - not code.

- Library now contains connections between different phase artifacts.

- "Library" is now a misnomer - it is no longer passive.
  repository or knowledge base might be better.

- Phase artifacts is mature, linkage between artifacts is not.

Using the "Library"

- No longer passive because of knowledge connections - a mechanism is required to navigate the library.

- Composition versus specialization
  - composition is integration on artifact use
  - specialization is integration planned at artifact addition to library

- The evils of glue code with composition
  - builds up / accumulates over maintenance - defensive programming
  - hard to maintain - high coupling because it crosses abstractions
  - leads to dead code

- Specialization is easier to use but harder to specify.

- There is always some composition.

Domain-Specific Knowledge

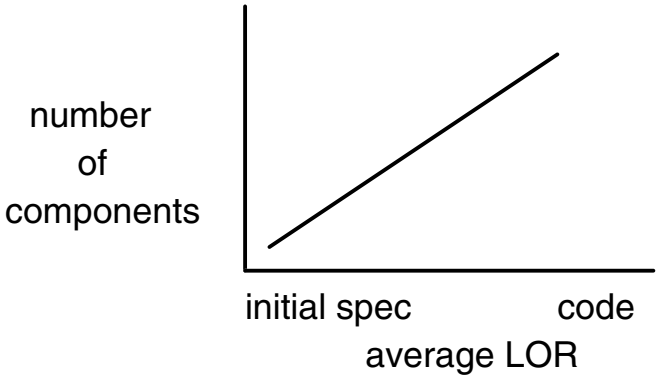
- Higher level of abstraction artifacts become problem domain specific
  - powerful system specification
  - avoids as much composition as possible
  - encourage as much specialization as possible

- Different approaches to doing and to results of domain analysis
  - languages - not mature
  - algebras - not mature
  - architecture/frameworks - mature
  - kits - mature
  - analysis reports - mature

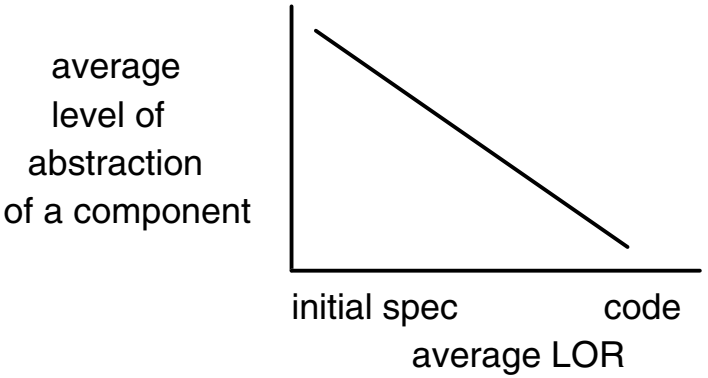
- Many uses of domain analysis: education, powerful specification and understanding

Abstraction and Refinement

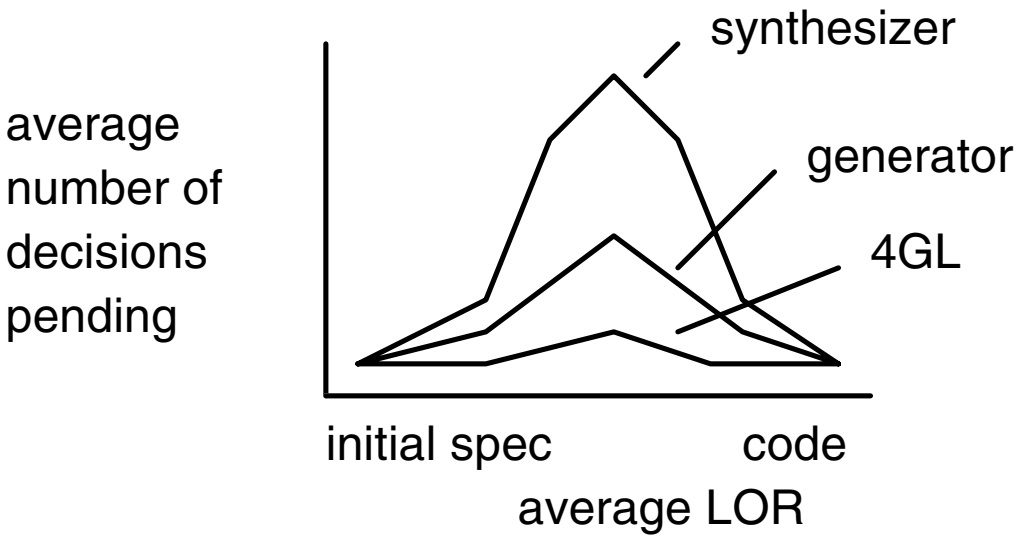
- Assume a connected knowledge base of domain-specific information
- New system is specified in problem domain terms
- Refinement is the process of making implementation decisions
  - level of abstraction (LOA) - length of artifact chain to compilable state with respect to a particular knowledge base.
  - level of refinement (LOR) - distance traversed down the chain
- Height of the decisions pending curve is the planning strength of the refinement mechanism needed.
- Mechanism planning strength is not power but flexibility
- Low LOA domain-specific artifacts can be powerful generators but inflexible.



Components vs. average LOR



average LOA vs. average LOR



Decisions vs. average LOR

Refinement Builds Structural Architecture

- Two forces shaping structural architecture
  - functional decomposition (stepwise refinement) - "...form...follows function"
  - encapsulation - layers of virtual machines
- Maximal information hiding is the deciding rule between them

- Different brands of encapsulation produce different structural architectures providing the same system function.

- MILs, dependency lists (make files), system distribution lists, programs executed at sys creation time, phases, initialization files, configuration files, documentation files, ...

- all refinement or simple library use builds a structural architecture - even if the user has no control over it.

Recap on Usage of Reuse Technology

- Organizational change technology is mature

- Library technology is mature
  recognize limitations
  knowledge base connectivity is not the usual model

- Domain analysis technology is somewhat mature.
  process is mature
  complex refinement processes not mature

- Issues of structural architecture not mature.

Organizational Questions

1. How does this scheme fit into my organization?

2. How is my version space impacted by using this method?
   issues over time

3. How is my configuration space impacted by using this method?
   issues of functions supported over platforms

4. How is the basic reuse cycle of abstraction, selection, specialization, and integration performed?

Specification of a System

1. What is the form of specification?

2. What is the LOA of the starting specification?

3. What is the result of domain analysis?

Refining Specification to Implementation

1. How is a consistent implementation maintained as the LOR increases?

2. How does the method / tool support various functional architectures?

3. How does the method / tool support various structural architectures?

4. How is specialization performed?

5. How do maintenance programmers maintain?

6. How do I guide refinement to different targets (e.g., implementations and simulations)?

Extending the Library / Knowledge Base

1. How is the library / knowledge base extended?

2. How are new subsystem interfaces (e.g., GUIs, DBs, Networks, OSs) specified?

The Next 10 Years

- More code level libraries - first step, 15% and disillusionment
- More abstraction towers - interoperation, high speed communications and OS.
- Successful but inflexible domain-specific generators
- Big painful glue, version & configuration failures - JCL déjà vu
- Formal algebra theory continues on - no successes
  perhaps in 20 to 50 years
- Not much knowledge based work will be done - too bad it is needed
- Slow shift in focus from productivity to flexibility
- Targets other than programming language source code - hardware/software co-generation (VHDL)

The Difficult Science and Business of Software

Beware "Technology Tabloid Journalism"

- "OLE 2.0 to be CORBA compatible."
- "Elvis sighted at burger stand with space aliens."

Beware the "paradigm / methodology du jour"

Computer Science has developed a good literature and it isn't cited much.

- Encapsulations of the 80's
- Artificial Intelligence of the 70's
- Classical Systems Analysis of the 60's
- Philosophy of the 50's

Thank you all for working in software - there are much easier things to do.