Composable Software Research at Carnegie Mellon

## Plan for the next few weeks

- **Mon 1 Oct**
  - Research paradigms and pro forma abstracts. Re-read Newman 94 and Shaw01, look at paper-writing advice on resources page
- **Wed 3 Oct, Mon 8 Oct, Wed 10 Oct**
  - Four research papers per class, 15-minute presentations
  - Choose a significant paper in your chosen topic area that illustrates both good results and good research strategy
  - Classic papers are prime candidates
- **Mon 15 Oct**
  - Plan projects, revisit framework for analysis. Re-read Redwine/Riddle
- **Wed 17 Oct for 11 classes**
  - 1 hour presentation/discussion of topic area
  - 1/2 hour other topic: classic paper, unify pro forma abstracts...

### Current interests

<table>
<thead>
<tr>
<th>Who</th>
<th>Topic 1</th>
<th>Topic 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owen Cheng</td>
<td>SE for security</td>
<td>SW economics</td>
</tr>
<tr>
<td>George Fairbanks</td>
<td>Modeling environments</td>
<td>SW analysis</td>
</tr>
<tr>
<td>Tim Halloran</td>
<td>Tools &amp; environments</td>
<td>Maintenance</td>
</tr>
<tr>
<td>Beth Latronico</td>
<td>Reliability</td>
<td>SW safety</td>
</tr>
<tr>
<td>Paul Li</td>
<td>Testing</td>
<td>Economics driven</td>
</tr>
<tr>
<td>Annie Luo</td>
<td>Testing for reliability</td>
<td>Performance</td>
</tr>
<tr>
<td>Elissa Newman</td>
<td>Reverse engineering</td>
<td>Maintenance &amp; evolution</td>
</tr>
<tr>
<td>Vahe Poladian</td>
<td>SW analysis</td>
<td>SE for security</td>
</tr>
<tr>
<td>Charles Shelton</td>
<td>SW architecture</td>
<td>Middleware</td>
</tr>
<tr>
<td>Bridget Spitznagle</td>
<td>Middleware</td>
<td>Reliability, depend, rev eng</td>
</tr>
<tr>
<td>Dean Sutherland</td>
<td>Analysis for maintenance</td>
<td>Main/evol, analysis</td>
</tr>
</tbody>
</table>

### Current interests

<table>
<thead>
<tr>
<th>Topic</th>
<th>Responsible Party</th>
<th>Sounding Board</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW analysis</td>
<td>Vahe Poladian</td>
<td>George Fairbanks, Dean Sutherland</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Dean Sutherland (anal for maint)</td>
<td>Tim Halloran, Elissa Newman</td>
</tr>
<tr>
<td>Reliability</td>
<td>Beth Latronico</td>
<td>Annie Luo</td>
</tr>
<tr>
<td>Reverse engineering</td>
<td>Elissa Newman</td>
<td>Bridget Spitznagle</td>
</tr>
<tr>
<td>SE for security</td>
<td>Owen Cheng</td>
<td>Vahe Poladian</td>
</tr>
<tr>
<td>Tools &amp; environments</td>
<td>George Fairbanks</td>
<td>modeling environments, Tim Halloran</td>
</tr>
<tr>
<td>Middleware</td>
<td>Bridget Spitznagle</td>
<td>Charles Shelton</td>
</tr>
<tr>
<td>Testing</td>
<td>Paul Li, Annie Luo (for reliability)</td>
<td>Charles Shelton</td>
</tr>
<tr>
<td>SW architecture</td>
<td>Charles Shelton</td>
<td>Owen Cheng</td>
</tr>
<tr>
<td>SW economics</td>
<td></td>
<td>, Paul Li</td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td>Annie Luo</td>
</tr>
<tr>
<td>SW safety</td>
<td></td>
<td>Beth Latronico</td>
</tr>
</tbody>
</table>
Overview

Building on early information hiding results, formal models for data abstraction enable a new class of programming languages

- Today’s papers:
  - First formulations of program verification
    - (supplemental) Floyd 67, Hoare69
  - First formal models for abstract data types
    - Hoare 72: relation between abstraction and implementation
    - Linker/Zilles 74: ADTs (should have been 75: data types as algebras)
  - Integration of formal model with programming language
    - Wulf/London/Shaw 76: models can be incorporated in languages

- Second-generation formalism

- Guttag/Horning/Wing 85: address some of design problems of algebra
Composable Software Research at Carnegie Mellon

Growth in Specification Power

- 1950: prose
- 1960: mnemonics, macros
- 1970: higher-level languages
- 1980: structured programming
- 1990: extensible languages
- 2000: architectural chunks

Liskov/Zilles: Programming with ADTs (1974)

- Sorry, specified the wrong paper
- This paper is like Parnas72 and Booch86
  > “We have a better idea”
  > “Here’s an example”
- You can see the transition from the idea of information hiding to programming language support
  > Global variables were still an issue then
- The paper was influential

Hoare: Correctness of Data Reps (1972)

- Real-world problem: Stepwise refinement suggests a method but does not help with determining whether successive steps are correct.
- Research problem: Find a relation between specification and implementation that allows verification that the implementation will be faithful to the specification.
- Experiment: Develop formal model based on an abstraction function that maps between representation and abstraction and an invariant that expresses integrity constraints. Prove that it shows correctness as above.
- Result: Method proved formally
- Impact:
  > Set rules for this sort of verification: explicit abstraction function, invariant
  > Established “commutative diagrams” as technique for reasoning about SW

Hoare: Correctness of Data Reps

1. The concrete function is correct
2. Abstract pre & inv imply concrete pre & inv
3. Concrete pre & inv & post imply concrete pre & inv
4. Therefore the abstract function is correct
Hoare: Correctness of Data Reps

- Problem type: To-show (Is X always true of Y ?)
- Research model: Formal model of relation between abstract and concrete. [Perhaps more relevant to abstract data types than to stepwise refinement]
- Hypothesis: Essence of the relation between specification and implementation can be captured as mapping, modeling, and invariant.
- Strategy: Demonstrate method on example, prove it correct.
- Validation: Formal proof of method, based on induction over sequences of operations.

Liskov/Zilles: Specs for Data Abstraction (1975)

- Real-world problem: Formal specification and verification is promising, but it’s still too hard.
- Research problem: For a particular class of problems – data abstractions– and a matching specification technique, and show that formal reasoning is practical for this class of problems.
- Experiment:
  - State criteria for evaluation
  - Classify approaches: abstract models vs implicit definitions
  - Using stack as example, evaluate several classes of approaches
- Result: Algebraic axioms best satisfy criteria
- Impact:
  - The 1974 precursor paper introduced algebraic axioms for specification

Liskov/Zilles: Criteria for Specification Methods

- Formality, to allow mathematical reasoning
- Constructibility, so specifications can be written
- Comprehensibility, so a reader can understand them
- Minimality, so no extra information interferes and so what is done isn’t confounded with how it’s done
- Wide range of applicability, to maximize utility
- Extensibility, so change in concept has similar change in spec

Liskov/Zilles: Specs for Data Abstraction

- Problem type: Classification (What are the varieties of X, and how are they related ?) and implicitly To-show (Is X always true of Y ?)
- Research model: Stack
- Hypothesis: Algebraic axioms best satisfy criteria.
- Strategy: Compare specifications of stack.
- Validation: Careful argument
- The other story …
- Algebraic axioms can capture essential functionality of data abstractions
  - the big result of the 1974 paper
  - established by developing the algebras for some examples
Wulf & al: Construction and Verification (1976)

- Real-world problem: Poor language support for constructing understandable, maintainable programs whose specifications can be shown to match their implementations.
- Research problem: Design a programming language that both captures abstractions as integral part of the program and supports verification.
- Experiment:
  > Design programming language with data abstraction as central element
  > Address Hoare-style verification as integral part of language
  > Using stack as example, show that the language restrictions match the verification requirements
- Result: Verification support can be integrated in language design
- Impact:

Guttag & al: Larch (1985)

- Real-world problem: This is a second generation specification language for data abstractions.
- Research problem: Address shortcomings of first generation: monolithic nature of a complete algebra, mismatch between formalism and language mechanism.
- Experiment:
  > Develop a theory in which specification fragments exist independently
  > Show how these can be combined to obtain full specifications
  > Separate pure functionality from specs related to language mechanisms
- Result: This partitioning is possible and promising
- Impact:
  > Consolidated prior work on specifications for data

Wulf & al: Construction and Verification

- Problem type: Methods (How do I accomplish X?)
- Research model: Stack, Hoare-style verification of abstractions
- Hypothesis: Synergy between language design can specification.
- Strategy: Design language based on requirement to capture abstraction information for verification and future maintenance, dry-run it on “stack”.
- Validation: Full verification of example. Careful argument about language and about interaction among method, language, verification
- The other story …

Guttag & al: Larch

- Problem type: Improvement (What is a better way to accomplish X?)
- Research model: Various container abstractions, algebraic axioms for abstractions
- Hypothesis: Specification-writing can be made easier by supporting composable fragments.
- Strategy: Design new language and theory for specification of data abstraction, in which specification fragments are the primitive elements
- Validation: Demonstration of theory on a set of examples
Composable Software Research at Carnegie Mellon

Maturity: Progressive Codification Cycle

- Folklore
- Ad hoc solutions
- Improved practice
- New problems
- Models & theories
- Codification
- Improved practice
- New problems
- Models & theories
- Codification
- Ad hoc solutions
- Folklore