Computer Aided Programming
Enabling Software at Scale

Armando Solar-Lezama

MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

Automation with a human touch

Computer Aided Engineering is a combination of techniques in which man and machine are blended into a problem solving team, intimately coupling the best characteristics of each.

S.A. Meguid 1986
Integrated Computer-aided Design of Mechanical Systems
The challenges of big software

- Big software is an ecosystem
- No one understands it in its entirety
- Challenges
  - Help programmers leverage their limited understanding to contribute to the ecosystem
  - Maintain confidence that critical system properties will be preserved
The problem with scale

OO Frameworks revolutionized programming
  - designed around flexibility and extensibility

Overall this was a good thing
  - facilitates reuse
  - new applications deliver rich functionality with little new code

But, there were unintended consequences
  - functionality is atomized into very small methods
  - proliferation of classes and interfaces
  - “Ravioli” code

Example: Eclipse Syntax Highlighting

Different lexical elements highlighted in different colors
What we know

TextEditor

ITokenScanner

SkEditor

SkScanner

TextEditor.setTokenScanner();

How do editors and Scanners Meet?

SkEditor

SkScanner

DamageRepairer

PresentationReconciler

(1)

(2)

(3)
How do editors and Scanners Meet?

Very **complicated**!

```java
class SkConfig extends SourceViewerConfiguration {
    public IPresentationReconciler getPresentationReconciler(…) {
        DefaultDamagerRepairer dr = new DefaultDamagerRepairer(new SkScanner());
        PresentationReconciler rcr = new PresentationReconciler();
        rcr.setDamager(dr, …); rcr.setRepairer(dr, …);
        return rcr;
    }
}

class SkEditor extends TextEditor {
    SkEditor() {
        setSourceViewerConfiguration(new SkConfig());
    }
}
```

**We can synthesize this code!**
Data Driven Synthesis

- The key problem is coping with scale
  - program is too big & complex to fully analyze statically

- Synthesizer must use data
  - database captures the accumulated insight of project members

MatchMaker approach

- Observation 1:
  Interaction between two objects usually requires a chain of references between them.

Our goal is to find the important code pieces that work together to build the chain
MatchMaker approach

- Observation 2:
  Often helpful to imitate the behavior of sibling classes.

- Observation 3:
  We have data about many runs with many different editors
  - (A1 ∧ A2) - B

Trace 1:
- A1 = {Important code for forming critical chain 1}

Trace 2:
- A2 = {Important code for forming critical chain 2}

Trace 3:
- B = {All code in this trace, which forms no critical link}
Algorithm

- Find critical chain in one trace:
  - iterate over the snapshots
  - find the earliest pointer dereference chain from $X$ to $Y$.
    - $X$: object of TextEditor's subclass
    - $Y$: object of ITokenScanner's subclass
- Thin slicing connects critical chain to code
- Result is a tree of important calls
- Compare trees from many different instances
  - Search for similarities and differences

Database

- Currently very rudimentary
- Track
  - method enter/exit,
  - heap load/store,
  - class hierarchy.
- Many events can be safely ignored
- Also contains periodic heap snapshots
- Lots of data, but manageable
  - between 3 and 7 MB per second of real-time execution
How long does this take?

- Searching for relevant data could be expensive
  - but it parallelizes easily
  - indexing can help a lot
  - right now our databases are small, so this takes < 30 sec

- The rest is easy after the right data is found
  - finding the critical path takes < 20 sec
  - building the call tree takes about 30 sec
  - tree matching takes < 1 sec

Take Home

- Modern OOP frameworks are
  - flexible
  - extensible
  - and very very complex.

- Hard to match classes so they work together

- MatchMaker uses data to synthesize code
PROGRAMMING WITH DELEGATION

Delegating Cross Cutting Concerns

- Critical properties are cross-cutting concerns
  - enforced by different bits of code scattered through the system

- Cross-cutting concerns make software complex
  - don't fit natural abstraction boundaries
  - often come as an afterthought in software design

- What if we could delegate them?
  - let programmer worry about the core functionality
  - and let the synthesizer deal with the cross-cutting concerns
Ex: Controlling Information Flow

[Image of a sign-in screen]

Ex: Controlling Information Flow

[Image of an email interface]
Ex: Controlling Information Flow

Info-Flow is a cross cutting concern

- Changes required throughout the code to enforce even simple policies.

- Poor match for traditional techniques
  - Aspect oriented programming is not “smart” enough
How was this fixed?

```php
class Mailer {

    var $hideSensitive;

}
```

- Mailer has sole responsibility for composing e-mails.
- `$hideSensitive` determines whether to show pwd
  - similar fields protect other forms of private information,
    e.g. reviews

An account has been created for you at the %CONFNAME% submissions site, including an initial password.

- Site: %URL%
- Email: %EMAIL%
- Password: %PASSWORD%

```php
$password = ($this->hideSensitive ? "HIDDEN" : $contact->password);
if ($what == "%PASSWORD%") return $password ;
if ($what == "%EMAIL%")    return $this->_expandContact($contact, "e");
```

An account has been created for you at the POPL 2011 submissions site, including an initial password.

- Site: [http://www.cs.tau.ac.il/conferences/popl11/](http://www.cs.tau.ac.il/conferences/popl11/)
- Email: asolar@csail.mit.edu
- Password: GoOdPwD
How was this fixed?

○ Program must create one message to display

```
$rest["hideSensitive"] = true;
$show_preparation =
    Mailer::prepareToSend($template, $contact, $rest);
$show_preparation->displayBody();
```

○ And a different one to send

```
$rest["hideSensitive"] = false;
$preparation =
    Mailer::prepareToSend($template, $contact, $rest);
$preparation->send();
```

This is too complicated!

○ Too many points of failure
  - programmer could
    • output without using the message class
    • pass the wrong flag
    • forget to create multiple versions of a message
    • use the wrong version of the message

○ Not to mention the design took a lot of work
Programming with delegation

- What if we could ignore the issue altogether

```php
$message = Mailer::expandTemplate($template, $contact);
$message->displayBody();
$message->send();
```

- And delegate the information flow control to a high-level policy

```php
foreach( x in users)
    assert flowout.user != x ➔ x.getPwd() == "HIDDEN"
```

Programming with delegation

- How do we allow the policy to be enforced?
  - preferably with minimal changes to the simple code

```php
function expandTemplate($t, $contact){
...
    $t = replace($t, "%PASSWORD%", $contact->getPwd());
...
}

function getPwd(){
    return delegate($this->password) ;
}
```

- Delegated expression gives the system control
Semantics of Delegation

```
    function getPwd()
      return delegate($this->password) ;
    }
    
    $t = replace($t, "%PASSWORD%", $contact->getPwd());
```

Semantics of Delegation

```
    foreach( x in users)
      assert flowout.user != x 
      x.getPwd() == "HIDDEN"
```
How does it work?

- Program uses **Symbolic Values** to represent data under the control of the runtime
- Runtime tracks logical relationships between symbolic values and program data
- Runtime uses an SMT solver to derive values for symbolic data

Status

- We have a runtime to do the blended symbolic/concrete execution
  - Performance is comparable to running an interpreted language
- We are formalizing the language semantics
- Working on a full language design
Conclusion

It’s time for a revolution in programming tools
- Unprecedented ability to reason about programs
- Unprecedented access to large-scale computing resources
- Unprecedented challenges faced by programmers

Successful tools can’t ignore the programmer
- programmers know too much to be replaced by machines
- but they sure need our help!