SYNTHESIS FROM FORMAL PARTIAL ABSTRACTIONS

Hamid Bagheri – PhD Proposal
University of Virginia, Computer Science
Developing complex software systems is difficult...
Goal and Contribution

To provide engineers with methods and tools that help them (1) to better make tradeoff decisions, (2) to develop sound architectures, and (3) to implement architectures upon complex, modern software platforms
Our Approach

- Use general-purpose relational specification language
- Use partial modeling
- Use general-purpose relational logic model finder
- Employ architectural description as intermediate representation
Our approach has significant potentials to address the problems we have identified in software engineering practice.
Research components

- Synthesizing architectural models from abstract application models
- Synthesizing Partial code frameworks from application architectures
- Automated Formal Exploration of tradeoff spaces
  - In the context of object-relational mappings
  - In a more general context of architectural decision making
Outline

- A review of each research component
  - Technical Merit
  - Evaluation Strategy
- Related Work
- Conclusion
FORMAL SYNTHESIS OF ARCHITECTURAL MODELS FROM ABSTRACT APPLICATION MODELS
How do architects do this? Can it be formalized? Can it be automated?
Software engineers have long considered architectural style as a separate design decision.

Application models can be mapped to architectures in many styles.
- Example: KWIC in two styles, Parnas, 1972

These important mappings have not previously been identified as an important subject of study.

Practiced informally, intuitively, implicitly.
Goals

- Develop rigorous account of these mappings
- Exploit that formalism to automate architecture synthesis
- Deliver synthesis capabilities in model-driven tools
Approach

- Formalize application type and architectural style
- Define mapping predicates taking applications of given types to set of architectures in given architectural styles
- Employ relational logic model analyzers to automate these mappings
Architectural Map

\[ t : \text{AppType} \]
\[ m : \text{AppModel} \]
\[ s : \text{ArchStyle} \]
\[ \text{map}_{(t,s)}(m) \]
\[ \text{conforms} \]
\[ \text{refines} \]
\[ \{a_i : \text{ArchModel}\} \]
Tool Support for Architecture Synthesis
Evaluation

- Tested the idea by exhibiting architectural maps for several application types and architectural styles.
- Check consistency: informally produced results in the literature.

<table>
<thead>
<tr>
<th></th>
<th>Comp. Fun.</th>
<th>State-Driven</th>
<th>SCC</th>
<th>ACF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipe-And-Filter</td>
<td>KWIC, LL</td>
<td></td>
<td></td>
<td>KWIC</td>
</tr>
<tr>
<td>Object-Oriented</td>
<td>KWIC, LL</td>
<td>KWIC</td>
<td>MIDAS, LL</td>
<td></td>
</tr>
<tr>
<td>Implicit Invocation</td>
<td>KWIC, LL</td>
<td>MIDAS, LL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Rows represent application types; columns represent architectural styles.
Related Work

- Formal approaches to architectural evolution
  - *Architectural evolution patterns*. Tamzalit and Mens, 2010
  - *Evolution styles*. Garlan et al., 2009
    - Horizontal vs. Vertical mapping

- Formalization of architectural styles
  - Kim and Garlan, 2010
  - Wong et al., 2008

- Separation of Concerns
  - *Flexible Packaging*. Deline 1999
    - Introduce and formalize architectural map, automate architecture synthesis
SPECIFICATION-DRIVEN SYNTHESIS OF ARCHITECTURAL CODE FRAMEWORKS
Can we do a better job mapping software models to programs based on commodity platforms?
Model Driven Engineering Solution

- Design domain specific languages (DSL)
- Develop model transformer for each DSL
- Model system in DSL, transformed to code
Problems with MDE Approach

- Designing language syntax and semantics is hard
- Hand-crafting translators is costly, error-prone
- Full system modeling in custom languages is hard
practically synthesize platform-specific frameworks (partial implementations) from formal architectural models without domain-specific languages or hand-crafted translators
Approach

- General-purpose relational logic language
  - Rather than designing specialized DSL

- General-purpose relational logic model finder
  - Rather than developing custom-built model interpreters
  - Augmented by a model to code-fragment binding mechanism

- Partial (architectural) system modeling
  - Rather than modeling the whole system
Pol: Synthesis Approach

Evaluation

- Feasible to synthesize architectural code frameworks without the need for custom-built code generators.

- The choice of object-oriented frameworks as outputs helps limit the impacts of changes in models.

- Performance of approach implementation using relational logic model finder reasonable.
CyberHealth

- A laboratory-scale operational model for nation-wide health information systems

- Its development has involved multiple modern platforms:
  - **Restlet** for Restful web services
  - **HornetQ** for message distribution
  - **Oauth 2.0** for authorization
  - **CometD** for reverse Ajax
  - **CouchDB** for data persistence
Preliminary Results

- Implementation over 12 thousand lines of code
- Use object-oriented framework to separate synthesized and hand-written code yielded a manageable change process
  - Architectural changes did not overwrite hand-written code
- Performance:
  - Time to synthesize code frameworks
  - The synthesis time for a bound of 30 was ~100 seconds
  - Certainly a reasonable outcome
Platforms and Architectural Styles


✓ Maintain separation between arch. styles and platform constraints

Filling the gap between architectures and implementations

- *ArchJava*. Aldrich, ICSE 2002
- *Archface*. Ubayashi, Nomura, and Tamai, ICSE 2010

✓ No support for arch. styles, or app. synthesis
✓ Separate architectural concerns from implementation details

Model-driven code generation

- *Framework-specific modeling languages*. Antkiewicz et al., 2009

✓ Share emphasis on explicit definition of mappings otherwise implicit in code
✓ Use formal analyzers to automate mapping
FORMAL SYNTHESIS OF TRADEOFF SPACES FOR OBJECT-RELATIONAL MAPPING
How do expert engineers select a mapping from a space of possible mappings?
Problem

- Developing object-relational (OR) mappings that achieve desired quality is difficult
- Automatic generation of mappings using fixed patterns
  - Restrict applicable mapping strategies
  - Lead to sub-optimal results
- Manual design of mappings
  - Achieve much higher quality
  - Require expertise in OR mapping strategies
Goal

Develop an approach and a tool capability that provide both the quality benefits of human experts and the productivity benefits of automated synthesis.
A formal automated approach for exhaustive \textit{synthesis} of OR mappings and their \textit{classification} into quality equivalence classes
Basic Idea

- To formalize object-relational mapping (ORM) strategies using relational logic
  - ORM strategies to manage association relationships
  - ORM strategies to manage inheritance relationships
  - Fine-grained strategies applicable to classes within inheritance hierarchy
- To use relational model finders to generate space of possible mappings w.r.t ORM strategies
**Spacemaker High-Level View**

**Formal object model**

```
module personObjectModel
open objectModel

one sig Person extends Class {}
  attrSet = identifier + name
  id = identifier
  no parent
  isAbstract = No 

one sig name extends Attribute {}
  type in string
```

**Open ORM Strategies**

```
open ORMStrategies
open personObjectModel

UnionSubclass[Manager]
genericMixedStrategy[Clerk, Student, Employee]
```

**User-specified Constraints**

**Object-Relational Mapping Strategies**
(formalized)

**Candidate from each class**

Plus the measures of quality attributes

Evaluation

- Feasibility
  - To formalize object-relational mapping strategies
  - To automate the formal synthesis of an exhaustive set of mapping alternatives

- Performance of technology implementation based on a bounded model finder is reasonable
Preliminary Experimental Results

- Several case studies including an e-commerce system object model
- Tradeoff space consists of hundreds of thousands of mappings
- Spacemaker generates tradeoff space < an hour
- Certainly a reasonable performance

<table>
<thead>
<tr>
<th>ObjectModel</th>
<th>Solutions</th>
<th>Eq.Classes</th>
<th>T[min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>137,000</td>
<td>67</td>
<td>46</td>
</tr>
<tr>
<td>Asset</td>
<td>124,000</td>
<td>31</td>
<td>&lt;1</td>
</tr>
<tr>
<td>Order</td>
<td>93,000</td>
<td>31</td>
<td>16</td>
</tr>
</tbody>
</table>

E-commerce experiment performance statistics
Proposed Tasks

- Develop a back-end parser
  - Integrating formally precise ORM synthesis with industrial OR mapping tools
- Provides end-to-end tool chain
- Enables validation of approach w.r.t constraints imposed by industrial tools
Related Work

- **Object-relational mappings and their quality impacts**
  - Understanding OR mapping: A framework based approach. Ireland et al., 2009.

- **Derivation of database-centric implementations from formal specification**
  - Defining a Map between specifications and implementations. Cunha et al., 2009.

✓ Use formal specifications to generate the space of mappings
AUTOMATED FORMAL EXPLORATION OF ARCHITECTURE TRADEOFF SPACES
Problem

- No best architecture meets conflicting quality requirements

Software Architects

- Consider several design alternatives
- Evaluate alternatives in dimensions of multiple quality criteria
- Identify a solution exhibiting a good trade-off

- Labor-intensive, time-consuming
- Lead to sub-optimal results
Goal

Develop an approach and a tool capability that help engineers understand and make multi-dimensional tradeoff decisions at the architecture level.
A formal automated approach for generating the space of architectures and classifying architectural alternatives in dimensions of relevant quality metrics
Proposed tasks

- **Tool implementation:** develop a prototype tool that implements architecture tradeoff space exploration
  - Synthesis of architectures in multi-dimensional architectural styles
  - Supporting extensions to handle various architectural quality assessments

- **Evaluation:**
  - Apply prototype tool to several case studies from literature
  - Measure computational time for architecture tradeoff space exploration and analysis
Conclusion
Contributions

- Identifies architectural map as important subject of study
  - Enables Formalizing and automating architectural synthesis
- A novel approach for practical synthesis without custom languages or translators
- A novel formal technique to tradeoff space synthesis
- Open new path to apply such a formally-specified technique in other disciplines, architectural decision making
Research Timeline

Research Task 1
- Architectural Maps
- Extending to AO
- Monarch Tool
- Architectural Tactics

Research Task 2
- Core implementation mapping
- Pol Synthesis tool & Evaluation

Research Task 3
- Formalizing ORM strategies
- Spacemaker method & tool

Research Task 4
- Arch. tradeoff space exploration


Thank You