POL:
SPECIFICATION-DRIVEN SYNTHESIS OF ARCHITECTURAL CODE FRAMEWORKS FOR PLATFORM-BASED APPLICATIONS

Hamid Bagheri, Kevin Sullivan
University of Virginia Computer Science
Example: Collaborative Echo

- RESTful, Publish-Subscribe Architecture
- Implemented on Restlet and CometD Platforms
Can we do a better job mapping software models to programs based on commodity platforms?
Standard Approach

- Expressive, domain-specific modeling language
- Hand-crafted, possibly retargetable translator
- Transform complete model to complete, hidden code
Problems With This Approach

- Designing languages (syntax & semantics) is hard
- Hand-crafting translators is costly and error-prone
- Need for full system modeling is overly constraining
- Approach is too top-down, not particularly agile
Our Goal and Contribution

Practical synthesis of platform-specific frameworks (partial implementations) from formal architectural specifications without domain-specific languages or hand-crafted translators
Our Approach: How We Do It

- Use established, general-purpose relational specification language
  - Instead of custom, domain-specific modeling languages
  - We have instantiated this approach using Alloy [Jackson et al, MIT]

- Accept partial modeling: architecture, platforms
  - Instead of complete system and platform models

- Use established, general-purpose relational logic model finder
  - Instead of custom, domain-specific translators
  - Augmented by a model to code-fragment binding mechanism

- Synthesize understandable frameworks, complete by hand
  - Instead of synthesizing complete, and “hidden” implementations
Mapping Process

Reusable

Architectural Style Specifications

from

Implementation Mapping Specifications

to

Target Platforms Metamodels

Design Fragments

Bindings

Architectural Model

Mappeing Engine

Generated Platform-specific Implementation Model

ILParser

Automatically Synthesized

Generated Architectural Code Framework

Application Developer
Hypotheses

- Technically Feasible
- Industrially Useful
- Accommodates Evolution
- Performance Adequate
Experiment, Results, Significance

- **Experiment:** laboratory-scale model of national-scale health data exchange on standard platforms
  - Rest, Authentication, Web-based Pub-Sub, etc.

- **Results:** Data provide early support for hypotheses

- **Significance:** A synthesis approach with potential to reduce costs of model-driven software engineering while extending its applicability to agile processes
Rest of Talk

- Illustrative Toy Example
- Details of Evaluation
- Related Work
- Conclusion
Toy Example
Example: Collaborative Echo

- Collaborative Echo/Chat Application
- Publish-Subscribe Architectural Style
- Built on Restlet and CometD Platforms
module Echo

open Cnc       // Component-n_Connector Style
open ServerPush // ServerPush Style

one sig EchoSystem extends System{}{
    components = Echo
    connectors = EchoServerPushConnector
    attachments = EchoServerPushService->MessageHandler
}

one sig Echo extends Component{
    messageHandler: one MessageHandler
}

abstract sig MessageHandler extends Port{}{
    portProcesses = GetMessage + PushMessage
}

fact {
    // Each port belongs to exactly one component
    all m: MessageHandler| one m->messageHandler & m->ports = m->messageHandler
}

// Process
one sig GetMessage extends Process{}{
    Message = data
}

one sig Message extends Data{}

one sig EchoServerPushConnector extends ServerPushConnector{}{
    roles = EchoServerPushService
}

// User interactions through web clients based on server-push technique
one sig PushMessage extends ServerPushAnnounce{} // A Role Process

one sig EchoServerPushService extends ServerPushPublish{} // A Connector Role
    roleProcesses = PushMessage
module Restlet

abstract sig Restlet {}
abstract sig Handler extends Restlet{}
sig Resource extends Handler{}
abstract sig Connector extends Restlet{}
sig Server extends Connector{}
sig Router extends Restlet{
    attachedResources: some Restlet
    }
    attachedResources in Application + Resource
}
sig VirtualHost extends Router {}
sig Application extends Restlet{
    router: one Router,
    resources: some Resource
    }
    router.attachedResources in resources
}
Implementation Mapping

```prolog
pred mapping() {

    // For each system, there is a Restlet component managing the system's components
    all s: System | one c: Restlet/Component | c.handle = s & &
                   c.applications.handle = s.components & & s.~handle = c

    // For each component in the architecture, there is a Restlet application managing resources
    // associated with each port of the component
    all c: Cnc/Component | one app: Restlet/Application | app.handle = c & &
                           c.ports in (app.resources.handle <=: Port) & & c.~handle = app

    #Restlet/Application = #Cnc/Component

    // For each port in the architecture, there is a Restlet resource handled by the Restlet application
    // associated with the port's component application managing resources associated with each port of the component
    all p: Port | one restletResource: Restlet/Resource | restletResource.handle = p & &
                 p.~handle = restletResource

    // for each connector of type ServerPushConnector, there is a BayeuxServer (CometD server) that handles it
    all connector: ServerPushConnector | one cometDServer: BayeuxServer |
       cometDServer.handle = connector & & connector.~handle = cometDServer

    // for each role of ServerPushConnectors, there is a corresponding CometdService
    // associated with the CometD server that handles the role's connector
    all role: ServerPushConnector.roles | one cometd: CometdService | cometd.handle = role & &
                                        cometd.bayeux = role.~roles.~handle // BayeuxServer

    #CometdService = #ServerPushConnector
}
```
Design Fragment

<design-fragment>
  <name>CometdServiceSetup</name>
  <goal>Setup a Cometd service that extends the Abstract Bayeux Service</goal>
  <framework-provided class="org.cometd.server.CometdService;org.cometd.bayeux.server.*;java.util.*"/>
  <programmer-required string="ServiceName;PackageName"/>
  <code-snippet>
    public class $ServiceName$ extends CometdService {

    protected String channel;

    public $ServiceName$(BayeuxServer bayeux, String channel) {
        super(bayeux, channel);
        this.channel = channel;
    }

    public $ServiceName$(BayeuxServer bayeux) {
        super(bayeux, "$ServiceName$);
    }

    public void publish(JSONObject body) throws JSONException {
        ...
    }

    }
  </code-snippet>
</design-fragment>
<bindings name="Echo">
  <binding design-fragment="CometdServiceSetup">
    <IL-Variables cometdServiceName="impl_mapping/CometdService"/>
    <PackageName>Echo</PackageName>
    <ServiceName>IL:cometdServiceName+CometdService</ServiceName>
  </binding>
</bindings>
Architectural Code Framework

Synthesized platform-independent interface

```java
public interface IEcho {
    public abstract void GetMessage(Message);
}
```

Synthesized platform-specific Class

```java
package Echo;

import ...

public class EchoCometdService extends AbstractService {
    protected String channel;

    public EchoCometdService(ToManyServer bay eux, String channel) {
        super(bay eux, channel);
        this.channel = channel;
    }

    public EchoCometdService(ToManyServer bay eux) {
        super(bay eux, "EchoCometdService");
    }

    public void publish(JSONObject body) throws JSONException {
        Json2Java jj = new Json2Java();
        HashMap<String, Object> bodyMap = new HashMap<>(jj.toObjectMap(body, bodyMap));
        HashMap<String, Object> dataMap = new HashMap<>();
        dataMap.put("body", bodyMap);
        ServerChannel srvChannel = getBay eux().getChannel(channel);
        if (srvChannel != null)
            srvChannel.publish(getServerSession(), dataMap, null);
    }
}
```
package Echo;

public class Echo implements IEcho {

    @Override
    public void GetMessage(Message m){
        MessageImpl message = (MessageImpl) m;

        EchoCometdService cometdService = EchoCometd.getInstance().PushMessageService;
        try {
            cometdService.publish(message.toJSONString());
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
Experiment & Results
Case Study

- CyberHealth: A laboratory-scale operational model of nation-wide health information systems
Manageability of System Evolution (Experiments)

- Step 0: Model based on hand-written system
- Step 1: Add server-push
- Step 2: Add Logging
- Step 3: Change systemic data flows
Feasibility & Potential Industrial Utility

- Significant architectural styles, platforms, functions
- Over 12 thousand LOC, more than 34% synthesized
Manageability of Evolution

- Object-oriented approach to separating synthesized and hand-written code yielded a manageable change process.
- Architectural changes did not overwrite hand-written code.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp. 1</td>
<td>40</td>
<td>52</td>
<td>450</td>
<td>96</td>
</tr>
<tr>
<td>Exp. 2</td>
<td>12</td>
<td>16</td>
<td>108</td>
<td>0</td>
</tr>
<tr>
<td>Exp. 3</td>
<td>84</td>
<td>49</td>
<td>704</td>
<td>570</td>
</tr>
</tbody>
</table>
Performance

- Time to synthesize code frameworks
- Alloy scopes from 10 to 30
- Synthesis bound of 30 was ~100 seconds
- Certainly a reasonable outcome
- Warning: Single data point
Related Work
Related Work

- Platforms and Architectural Styles
  - Middleware-induced architectural styles, Di Nitto and Rosenblum, ICSE’99
  - Formal modeling of EJB, Sousa and Garlan, 2001
  - Specification language for framework constraints (SCL), Hoo and Hoover, 2006

- Filling the gap between architectures and implementations
  - ArchJava, Aldrich, ICSE 2002
  - Archface, Ubayashi, Nomura, and Tamai, ICSE 2010
  - Prism-MW architectural framework. Malek et al., 2005

- Synthesis of MDE code generators
  - Framework-specific modeling languages, Antkiewicz et al., 2009
  - XTEAM, Edwards and Medvidovic, 2008
  - Santos, Koskimies, Lopes, Journal of Systems and Software, 2010

- Separation of Synthesized and Hand-Crafted Code
  - 1.x-way mapping, Zheng and Taylor, ICSE 2012
  - Cassou et al., ICSE 2011
Conclusion
An approach to reducing costs in model-driven development and extend it to more agile processes
Thank You