Outline

- Introduction
- Related Work
- Contributions
- Implementation
- Validation
- Conclusion & Future Work
Context - Constrained Devices

- Cannot locally support an IDE & Dev-Tools
- Have different HW/SW Configurations from Dev-Machines
Context - Debugging Constrained Devices

- Emulators
- Post-Mortem Analysis
Context - Debugging Constrained Devices

- Emulators
- Post-Mortem Analysis
Remote Debugging

Is a solution that is distributed in a nature:

- **Impact on productivity** due to re-deployments [ZeroTurnAround 2011]
- **Lacks facilities otherwise** available in a local setting (e.g. O-Centric Debugging [Rescia 2012b])
Research Questions

- What are the properties of an ideal remote debugging solution?
- Given these properties which model for remote debugging can exhibit them?
- What are the trade-offs between this ideal model and a real world implementation?
Thesis Statement - Properties

An ideal remote debugging solution should support

- Interactiveness
- Instrumentation
- Distribution
- Security
Interactiveness

the ability to **incrementally update** all parts of a remote application without losing the running context (i.e. without stopping the application).

Add/Rem **Packages**, Add/Rem **Classes**, Add/Rem **Fields**, Edit **Hierarchy**, Add/Rem **Methods**
Instrumentation

the ability to alter the semantics of a running process in order to assist debugging

*Method/Statement* Execution, *Class* Instantiation/Field Read/Write
*Object* Read/Write/Send/Receive/Argument/Store/Interact
the ability of a debugging solution to adapt its framework while debugging a remote target

No-Distribution / Fixed Middleware / Extensible Middleware / Adaptable Middleware
Security

the availability of prerequisites for security mechanisms such as authentication and access restriction

Internal / External / Target-Side / Client-Side
Outline

- Introduction
  - Related Work
- Contributions
- Implementation
- Validation
- Conclusion & Future Work
### Related Work - Overview

<table>
<thead>
<tr>
<th>Feature</th>
<th>JPA</th>
<th>.NET</th>
<th>GDB</th>
<th>DCE</th>
<th>.REBEL</th>
<th>SMALLTALK</th>
<th>BIFROST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactiveness</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Instrumentation</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>Distribution</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Security</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Related Work - Overview

(A) **None** of the existing solutions met all of our criteria

(B) But **reflective debugging** proved superior to other approaches in a local setting

<table>
<thead>
<tr>
<th></th>
<th>JAVA</th>
<th>DCE</th>
<th>JPDA</th>
<th>.NET</th>
<th>JREBEL</th>
<th>SMALLTALK</th>
<th>BIFROST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(B) Since **reflective debugging** proved superior to other approaches in a local setting

<table>
<thead>
<tr>
<th></th>
<th>.net</th>
<th></th>
<th>JRebel</th>
<th></th>
<th>BIFROST</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td>3</td>
<td>12</td>
</tr>
</tbody>
</table>

(C) Investigate Remote Reflection Design Patterns
Remote Reflection

Remote Proxy / Remote Facade / Mirrors
Mirrors - Explicit Meta-Objects

- Encapsulation
- Stratification
- Ont. Correspondance

Mirror on: anObject

Indirection
Design Patterns - Criteria

- Extensibility, Re-use
- Distribution
- Identity [Bracha 2010]
- Meta-recursion [Denker 2008]

MIRRORS can be seen as an extension to both the remote proxy and the remote facade patterns
Mirrors - Open Issues

- **Mirrors and the Problem of State** - debugging meta-information in cohesive language kernels
- **Mirrors and Intercession** - advanced instrumentation support while debugging
Outline

- Introduction
- Related Work
  - Contributions
- Implementation
- Validation
- Conclusion & Future Work
Our Proposals

- A solution to the problem of Reflective-Data [Maes 1987b] in the context of mirrors [Bracha 2004]

- The definition of a model for remote debugging that can exhibit the properties of: interactiveness, instrumentation distribution and security
From a language design perspective meta-objects should be both:

- **Pluggable** as mirrors are
- **State-Full** as 3-KRS meta-objects
MetaTalk Implementation

- **MetaTalk-VM** is written in Pharo
- **MetaTalk-Compiler** relies on Petit-Parser
- **Object-Model** inspired by ObjVLisp

http://www.squeaksource.com/MetaTalk/
Our Proposals

- A solution to the problem of Reflective-Data [Maes 1987b] in the context of mirrors [Bracha 2004]
- The definition of a model for remote debugging that can exhibit the properties of: interactiveness, instrumentation distribution and security
Interactiveness - through a mirror-based remote meta-level that is causally connected to its target
Core Model

Development Side

Mirror
- targetObject: Object

ObjectMirror
- rtMirror: RunTimeMirror
  + instVarAt(anIvName: String): ObjectMirror
  + ...

RunTimeMirror
- runTime: RTDebuggingSupport
  + objectinstVarAt(forObject: Object, anIvName: String): ObjectMirror
  + ...

Target Side

Object

Point

RTSupport
+ objectinstVarAt(forObject: Object, anIvName: String): Object
+ ...

RunTimeDebuggingSupport
runTimeDebuggingSupport

Mirror reflects on

......

instance of

mirrorOn

nAPoint

aRunTimeMirror

aPoint

Development Side Reflects on Target Side
Interactiveness

(1) Structural Reflection

ObjectMirror
- rtMirror: RunTimeMirror
+ setClass(aClassMirror: ClassMirror): ClassMirror

EnvironmentMirror
+ newPackageNamed(aPackageName: String): PackageMirror
+ removePackageNamed(aPackageName: String): PackageMirror

PackageMirror
+ newClassNamed(aClassName: String): ClassMirror
+ removeClassNamed(aClassName: String): ClassMirror

ClassMirror
+ setSuperClass(aClassMirror: ClassMirror): ClassMirror
+ addInstVarName(anIvName: String): ClassMirror
+ deleteInstVarName(anIvName: String): ClassMirror
+ addMethod(methodName: String, source: String): MethodMirror
+ deleteMethod(methodName: String): MethodMirror

MethodMirror
+ recompileWithSource(src: String): MethodMirror

(2) Computational Reflection

Mirror
- targetObject: Object

RunTimeMirror
- runTime: RunTimeDebuggingSupport

ExceptionMirror

ProcessMirror

ContextMirror
+ saveAndContinue(newSrc: String): ContextMirror
Our Proposal - Overview

Instrumentation - through reflective intercession by reifying the underlying execution environment
Our Proposal - Overview

**DEVELOPER'S END**
- MODEL OF THE DEBUGGED APP
- DEBUGGER / IDE
- MIDDLEWARE

**TARGET**
- DEBUGGED APPLICATION
- RUN-TIME DEBUGGING SUPPORT
- MIDDLEWARE

- Distribution - *through an Adaptable Middleware*
Distribution
Our Proposal - Overview

- **Security** - security by decomposing and authenticating access to reflective facilities
Security

Development Side

SecurityPolicy

Process

RemoteEnvironment
- address: String

Reflection
+ onEnvironment(environmentAddress: RemoteEnvironment) : EnvironmentMirror

Mirror
- targetObject: Object

ObjectMirror
- rtMirror: RunTimeMirror

RunTimeMirror
- runTime: RunTimeDebuggingSupport

ObjectIntrospectionMirror
+ instVarAt(anIvName: String): ObjectIntrospectionMirror
+ ...

ObjectIntercessionMirror
+ instVarAtPut(anIvName: String, aValue: ObjectIntercessionMirror): ObjectIntercessionMirror
+ ...

EnvironmentIntrospectionMirror

EnvironmentIntercessionMirror

InterspectionMirrorForAPoint

ProcessIntrospectionMirror

ProcessIntercessionMirror

Instance of

InterspectionMirrorForAPoint
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>JAVA</th>
<th>.NET</th>
<th>GDB</th>
<th>DCE</th>
<th>JRebel</th>
<th>SMALLTALK</th>
<th>BIFROST</th>
<th>MERCURY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactiveness (6)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Instrumentation (13)</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>Distribution (+++)</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+++</td>
</tr>
<tr>
<td>Security (4)</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Related Work
- Contributions
  - Implementation
- Validation
- Conclusion & Future Work
<table>
<thead>
<tr>
<th>Mercury-Core</th>
<th>Mercury-Ui (Alexandria)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seamless</td>
<td>MetaStackVM</td>
</tr>
</tbody>
</table>

- [http://ss3.gemstone.com/ss/Mercury-Prototype.html](http://ss3.gemstone.com/ss/Mercury-Prototype.html)
- [http://ss3.gemstone.com/ss/Seamless.html](http://ss3.gemstone.com/ss/Seamless.html)
- [http://ss3.gemstone.com/ss/mSVM.html](http://ss3.gemstone.com/ss/mSVM.html)
Meta-Recursion - \textit{mStackVM}

Interpreter
on: MessageReceived
for: anObject
do: [:reifications |
    anObject incrementMessageCounter:
    anObject
    perform: reifications selector
    withArguments: reifications arguments]
Interpreter
on: MessageReceived
for: anObject
do: [:reifications :reflectogram |
  reflectogram disable.
anObject incrementMessageCounter.
  reflectogram enable.
  reflectogram
  returnValue: reflectogram defaultAction]
MetaStackVM - Reflectogram

Interpreter
on: MessageReceived
for: anObject
do: [:reification:
  reflectogram
  anObject incrementMessageCounter.
  reflectogram
  reflectogram
  +.. + enable + disable + remove +...
  + defaultAction
  + returnValue:
  +...
  + processMetaLevel
  + objectMetaLevel
  +...
  + obj:perform:
}
Implementation Trade-offs

Supporting Interactiveness and Instrumentation

- Through Local Reflection
- Through Virtual-Machine support
- Through Byte-code manipulation
Benchmark based on Tanter [Tanter 2003]

- No Instrumentation
  - Bifröst: 1x
  - Mercury: 1x
- Disabled Instrumentation
  - Bifröst: 1x
  - Mercury: 1x
- Enabled Instrumentation
  - Bifröst: 35x
  - Mercury: 8x
Outline

- Introduction
- Related Work
- Contributions
- Implementation
- Validation
- Conclusion & Future Work
Experimental Setting

Constraint Device (A)
Phone
- WiFi: 10.1.160.92
- USB: PORT-FWD
- Galaxy Nexus - (1.2 GHz / 1GB RAM)
- Android 4.3

Constraint Device (B)
Tablet
- WiFi: 10.1.160.158
- USB: PORT-FWD
- Galaxy Tab - (1.0 GHz / 1GB RAM)
- Android 4.0

Constraint Device (C)
Remote Server
- ETH: 10.1.10.81
- WIFI: 10.1.160.92
- USB: PORT-FWD
- HP Workstation - (2.3 GHz / 4GB RAM)
- Ubuntu 12.04

Mercury IDE

Developer-Machine
- ETH: 10.1.10.206
- WIFI: 10.1.160.116
- USB: PORT-FWD
- MacMini - (2.3 GHz / 4GB RAM)
- Ubuntu 12.04
Alexandria

PROTOTYPE

sleep tight, don't let the source bugs bite
Objectives

- **Verify the applicability** of Mercury for these constrained debugging targets.
- **Illustrate how** a debugging session benefits from Mercury’s properties

Case-Study I
Remote Agile Debugging

Case-Study II
Remote Object Instrumentation
Remote Agile Debugging

Phone: '/charger'
Tablet: '/default.prop'
Server: '/User/.profile-xmind-portable-201212250029'
Remote Agile Debugging

**Constraint Device (A)**
- **Developer-Machine**
  - Mac-Mini - (2.3 GHz / 4GB RAM)
  - Ubuntu 12.04

**Constraint Device (B)**
- **Constraint Device (A)**
  - **Tablet**
    - Galaxy Tab - (1.0 GHz / 1GB RAM)
    - Android 4.0

**Constraint Device (C)**
- **Remote Server**
  - HP Workstation - (2.3 GHz / 4GB RAM)
  - Ubuntu 12.04

**Network Connections**
- **ETH:** 10.1.10.206
- **WIFI:** 10.1.160.158
- **USB:** PORT-FWD

**Mercury IDE**

**FileBrowserTest**
- + suffixOf: aString
- + testSuffixWithDot
- + testSuffixWithoutDot
- + testLongFilePath

**TestCase**
Remote Agile Debugging

Initial Error

Re-produced Errors from test cases

Re-produced Errors as Failed Assertions
Objectives

- **Verify the applicability** of Mercury for these constrained debugging targets.
- **Illustrate how** a debugging session benefits from Mercury’s properties

Case-Study I

Remote Agile Debugging

Case-Study II

Remote Object Instrumentation
Remote Object Instrumentation

Phone: '/charger'
Tablet: '/default.prop'
Server: '/User/.profile-xmind-portable-201212250029'

FileDirectory dot
Remote Object Instrumentation

Mirror of Target Object

Mirror of Interacting Object

Optional Condition Of Meta-Action

Semantical Event

Meta-Action
Remote Object Instrumentation

1. Assertion Halted on Semantic Event

2. Involving two interacting remote objects
Results

Droid-Browser

- acct
  14 October 2013 12:54:16 pm
- cache
  8 September 2013 10:52:48 am
- charger
  1 January 1970 1:00 am
- config
  14 October 2013 12:54:17 pm
- data
  9 September 2013 4:26:30 pm

Upload Files to Cloud
Outline

- Introduction
- Related Work
- Contributions
- Implementation
- Validation
  - Conclusion & Future Work
Summary - Contributions

- **Identification** of four desirable properties for remote debugging: *interactiveness, instrumentation, distribution and security*.
- The **definition of a model** for remote debugging (**Mercury**) that exhibits these desirable properties.
Contributions

- **The reification** of a previously illustrative notion (that of the *reflectogram* [Tanter 2003])

- **Prototype implementation** of our model for remote debugging in the context of reflective languages.

- **Implementation of an adaptable middleware** [David 2002] for supporting distribution (**Seamless**).

- **Implementation of a dedicated VM** for Pharo (**MetaStackVM**) for advanced intercession facilities.
Diffusion of Results

- **Submitted/Published:** Nikolaos Papoulias, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse and Luc Fabresse. *Towards Structural Decomposition of Reflection with Mirrors*. In Proceedings of International Workshop on Smalltalk Technologies (IWST’11), Edinburgh, United Kingdom, 2011. 105


Future Work

- Language and **Virtual-Machine Debugging** in the Same Model

- Integration of **Automated** Debugging Techniques (e.g. delta-debugging) in **Developer-Driven Debugging**
Thank you!
Some time left?
Groovy!!
Collaborations

- **Seamless as a library** for the Continuous integration services of Pharo.
- **The MetaTalk model as a case-study** for bootstraping OO - languages.
- **Mercury integration** with the PhaROS robotic middleware (on-going effort).
Design Patterns - Mirrors

- Explicit meta-object
- Abstract class / Interface
- Factory
- Facade and Bridge
Conditional Meta-Action

[:reifications :reflectogram |
  reifications trigger halt.
  reflectogram
    override: true;
    returnValue: reflectogram defaultAction.
  ]
Seamless Initialization

SeamlessDeamon class>>newDefaultWithGlobalAccess

^ self new
  buildWithTransporterClass: SeamlessSocketStreamTransporter
  transcoderClass: SeamlessFuelTranscoder
  proxyClass: SeamlessFastDNUProxy
  garbageCollectorClass: SeamlessDefaultGarbageCollector
  andAuthenticationManager: ((SeamlessAuthenticationManager new)
    addGroup: [...] withPolicy: (SeamlessDistributionPolicy
      newWithEntryPoint: [...] classesToPassByValue: [...] classesToPassByShallowCopy: [...] andSecurityPolicy: (SeamlessSecurityPolicy
        newWithClassesToPassByReference: [...] classesNotToReference: [...] includingMessages: [...] excludingMessages: [...] );
JMercury - Our model on top of Java

- JPDA + DCE VM (Interactiveness)
- Reflex / ASM / JavaAssist (Instrumentation)
- Cajo Project (Distribution)
- Decomposed Hierarchy of Mirrors / Closer integration with SecurityManager (Security)
Emulators - Field Experience

- **IPhone/Android emulators** (different models - versions of OSes - gyroscopes - touch gestures ...)

- **Car-Team experience -- RoboShop 2013 Demo** (unanticipated changes - people walking by - glass walls ...)