MODEL TALK: A Framework for Developing Domain Specific Executable Models

Atzmon Hen-tov and Lior Schachter
Pontis Ltd., Israel

Joint Work With:

David H. Lorenz
The Open University of Israel
The Challenge:
Telco-grade Dependability with Extreme Agility

- **Telecommunications Business Support System (BSS)**
  - Customers: Communication service providers
  - Product: Marketing delivery platform

- **Problem Space**
  - Strict extra-functional requirements
  - Pervasive customization
  - Frequent updates

- **Solution Space**
  - Third-party components (Main-stream J2EE technologies)
  - Domain-specific model-driven development
  - Product-line software engineering
ModelTalk Facts

- **Interpretive Approach**
  - Short edit-execute cycle
  - Minimum changes to binary code
  - Meta-model changes are automatically reflected in the tool

- **Commercial Experience**
  - 50 developers; >20 systems
  - 50 TPS/CPU.
  - Response time:
    - 70 ms average,
    - 250 ms 99%

- **Customization**
  - Time and effort dropped by an order of magnitude
  - 82% declarative
Agenda

✓ Introduction
✓ ModelTalk Facts

• The ModelTalk Approach
• ModelTalk in Action
• Conclusion
### The ModelTalk Approach (1/3)

<table>
<thead>
<tr>
<th>Model (XML)</th>
<th>Code (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Structure and static state</td>
<td>- Behavior</td>
</tr>
<tr>
<td>- Instances, Classes, Meta-classes</td>
<td>- ModelTalk managed Java classes</td>
</tr>
</tbody>
</table>

The ModelTalk Approach (1/3)

- **Metaclasses**
- **Classes**
- **Instances**
ModelTalk Dependency Injection (2/3)

- Model Driven Dependency Injection
- Model instances are constructed and injected into Java instances
ModelTalk Adaptability (3/3)

<table>
<thead>
<tr>
<th>Model (XML)</th>
<th>Code (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Mapping permits “holes” on the Java side</td>
<td></td>
</tr>
<tr>
<td>- Holes enable Java-less model change process</td>
<td></td>
</tr>
</tbody>
</table>

- Injection in Java side
- No Class in Java side

The Open University of Israel

Pontis
The ModelTalk Architecture

- Model sources
- Model processors
  - Problems view
  - Auto completion
  - Navigation views
  - Re-factor tools
- Consistent model
- Compiled code
- Code processors
- Source code
- Runtime platform
  - Model VM
  - Code VM
Agenda

- Introduction
- ModelTalk Facts
- The ModelTalk Approach
  - ModelTalk in Action
  - Conclusion
ModelTalk in Action

• **Objective**
  – Look and feel of the ModelTalk IDE

• **Example**
  – Customizing the Pontis application for OOPSLA in less than 10 minutes…
  – OOPSLA Happy Hour promotion
    • $20 discount on selected OOPSLA tutorials during Oct 19-20, 2008.
Part I: The Programmer Perspective

- Create a HappyHour instance
- Add the HappyHour instance to the DB
- Customize UI labels
- Manipulate the HappyHour instance in the GUI

Model-Code IDE → Easier assimilation
Model-compilation → Controlled declarative changes
Meta-object extensibility → Meta-Data = Data
Model class: “HappyHour”
Instance-of: “HappyHourMeta” (metaclass)
Extends: “BusinessTemplate”

Class metadata

Field metadata

Modeling navigation views
The model contains dozens of metaclasses, thousands of classes and ten of thousands of instances.
Upon a change to the model, the compiler is invoked to perform cross-model validation. An incremental model change takes no more than a few seconds. Errors are reported in the IDE standard problems view.

Model instance: “OOPSLA_HH”
Instance-of: “HappyHour”
Extends (i.e., prototyping): “HappyHour_Base”

Auto-completion for attributes values, tag names, etc.
A "Profile sheet" provides a comprehensive view of a model element. Inherited fields are displayed in gray color and fields that were defined in the model element are displayed in black color.
Technical DSLs are defined to improve the communication within the development team.
Part II: The Non-Programmer Perspective

- Create new reference code Tutorial in the GUI
- Create new OOPSLA event Tutorial purchase Event in the GUI
- Send an event to the Pontis system and receive a discount

Model VM → Runtime modeling capabilities
Interpretive → Short cycle
Non-programmers modeling workbench is form based. Changes to the model are automatically reflected in: O/R mapping layer, GUI, External API (Web-Service).
The new (dynamic) model class can now be used in a business rule (just as a regular class) and influence the execution of the system.
Conclusion

ModelTalk = MDD + Dependency Injection + Meta-modeling

- ModelTalk integrates MDD, Dependency Injection and Meta-Modeling to form an interpretive, Domain Specific Modeling framework.
ModelTalk: A Framework for Developing Domain Specific Executable Models

Contact info:

Lior Schachter, Senior Software Architect, Pontis Ltd., Israel, liors@pontis.com
Atzmon Hen-tov, Chief Software Architect, Pontis Ltd., Israel, atzmon@pontis.com
Professor David Lorenz, The Open University of Israel, lorenz@openu.ac.il