MDD in Practice

www.lukas-renggli.ch
Lukas Renggli

- **Software Engineer at Google**
  YouTube Video Analytics
- **SCG Alumni**
  Bachelor, Master and PhD
- **Open-Source Communities**
  Core-developer of Seaside
  Author of Magritte and Pier
Any sufficiently complicated program contains at least one meta-model.
Roadmap

1. Protocol Buffers
   Meta-data model for serialization

2. OmniBrowser
   Meta-model for tool building

3. Google Web Toolkit
   Model-driven web architecture

4. Magritte
   Generic meta-model
Protocol Buffers
Meta-data model for serialization
Describe your data format, to share and evolve it.
Protocol Buffers

- Encode structured data
- Language-neutral
- Platform-neutral
- Extensible
- Efficient
Protocol Buffer IDL

```protobuf
message Person {
  required int32 id = 1;
  required string name = 2;
  optional string email = 3;
}
```
message Person {
  required int32 id = 1;
  required string name = 2;
  optional string email = 3;
}

protoc

Java

Python

C++
Serialize in C++

```cpp
Person person;
person.set_name("John Doe");
person.set_id(1234);
person.set_email("jdoe@example.com");
fstream output("myfile", ios::out | ios::binary);
person.SerializeToOstream(&output);
```
Deserialize in Python

```python
file = open("myfile", "rb")
person = Person()
person.ParseFromString(file.read())
file.close
print "Name: ", person.name
print "E-mail: ", person.email
```
message Person {
  required int32 id = 1;
  required string name = 2;
  optional string email = 3;
}

message Person {
  required int32 id = 1;
  required string name = 2;
  repeated string email = 3;
  repeated Phone phone = 4;
}
Demo
Why not XML/JSON?

- Protocol Buffers are
  - 3—10 times smaller
  - 20—100 times faster
  - consistent code generators
  - less ambiguous
  - evolvable
Discussion

- Describe your data once, and use it across platforms and languages
- Evolution supported by design
- Used at Google for almost all
  - file formats (data storage)
  - remote procedure protocols (RPC)
OmniBrowser

Meta-model for tool building
Create a first-class description of the domain.
Object subclass: #OBBrowser
instanceVariableNames: 'panels announcer cmdFactories'
classVariableNames:"
poolDictionaries:"
category: 'Omnibrowser-Kernel'
Variables of OB Browser

- announcer
- cmdFactories
- panels

OB Browser>>initialize
OB Browser>>announcer
OBCodeBrowser>>subscribe
OREnvironmentBrowser>>setMetaNode node:
Object subclass: #OBBrowser
  instanceVariableNames: 'panels announcer cmdFactories'
  classVariableNames: ''
  poolDictionaries: ''
  category: 'OmniBrowser-Kernel'
Environment
OB-Morph

OB-Browser

Object subclass: #OBBrowser
  instanceVariableNames: 'panels announcer cmdFactories'
  classVariableNames: 
  poolDictionaries: 
  category: 'OmniBrowser-Kernel'
Object subclass: #OBBrowser
    instanceVariableNames: 'panels announcer commandFactories'
    classVariableNames: ',
    poolDictionaries: ',
    category: 'OmniBrowser-Kernel'
```ruby
Object subclass: #OBBrowser
  instanceVariableNames: 'panels announcer cmdFactories'
classVariableNames: "
poolDictionaries: "
category: 'OmniBrowser-Kernel'
```

The image shows a window titled `OBBrowser` with a list of classes and functions available in the OB-Mars environment. The window is divided into sections labeled `Instance` and `Class`, with various items listed under each section. Some of the classes listed include `OBBuilder`, `OBColumn`, `OFan`, `OBList`, `OBMetaEdge`, `OBMetaNode`, `OBNode`, `OBPlatform`, and `OBSubtree`. The functions listed under these classes include `accessing`, `building`, `callbacks`, `initializing`, `navigating`, `opening`, and `updating`. There are also methods like `addPanel`, `announce`, `announcer`, `buildGroup.on`, `buildOn`, `close`, `cmdFactories`, `commandSelectors`, `currentNode`, `currentOrRootNode`, and `defaultBackgroundColor`.
Discussion

- Model the concepts of your domain
- Domain-specific reflection
- UI independent
- Can be limiting or slow
- People might get meta-confused
Google Web Toolkit

Model-driven web architecture
Write your application in Java, run your application in JavaScript
public class Main implements EntryPoint {
    public void onModuleLoad() {
        Label label = new Label("Hello World");
        RootPanel.add(label);
    }
}
Is this really a model?

```java
public class Main implements EntryPoint {
    public void onModuleLoad() {
        Label label = new Label("Hello World");
        RootPanel.add(label);
    }
}
```
Is this really a model?

public class Main implements EntryPoint {
    public void onModuleLoad() {
        Label label = new Label("Hello World");
        RootPanel.add(label);
    }
}
Is this really a model?

```java
public class Main implements EntryPoint {
    public void onModuleLoad() {
        Label label = new Label("Hello World");
        RootPanel.add(label);
    }
}
```
public class Main implements EntryPoint {
    public void onModuleLoad() {
        Label label = new Label("Hello World");
        RootPanel.add(label);
    }
}
# GWT Translation

<table>
<thead>
<tr>
<th></th>
<th>Mozilla</th>
<th>Chrome</th>
<th>Safari</th>
<th>IE</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td><img src="image.png" alt="PNG" /> <img src="image.js" alt="JS" /> <img src="image.css" alt="CSS" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>German</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>
</tr>
</tbody>
</table>

*Thursday, 20 October 11*
GWT Translation

<table>
<thead>
<tr>
<th></th>
<th>Mozilla</th>
<th>Chrome</th>
<th>Safari</th>
<th>IE</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>German</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
</tbody>
</table>

...
Discussion

- Write JavaScript using a high-level widget library in a well defined (statically typed) language
- Translate and optimize code towards specific browsers
- Debugging, kind of works but you don’t want to know how
Magritte
Generic Meta Meta-Model
Describe your objects once, use the descriptions everywhere.
## Address Object

<table>
<thead>
<tr>
<th>:Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>street = 'Schützenmattstrasse'</td>
</tr>
<tr>
<td>plz = 3012</td>
</tr>
<tr>
<td>place = 'Bern'</td>
</tr>
<tr>
<td>canton = 'Bern'</td>
</tr>
</tbody>
</table>
Address Class

:Address
street = 'Schützenmattstrasse'
plz = 3012
place = 'Bern'
canton = 'Bern'

Address
street: String
plz: Integer
place: String
canton: String
Address Description

:Address
  street = 'Schützenmattstrasse'
  plz = 3012
  place = 'Bern'
  canton = 'Bern'

:Container
  description
    :StringDescription
      label = 'Street'
    :NumberDescription
      label = 'PLZ'
      required = true
      range = 1000..9999
    :StringDescription
      label = 'Place'
      required = true
    :SingleOptionDescription
      label = 'Canton'
      required = true
      sorted = true
      options = #('Bern' 'Zurich' ... )
Address Description

:Address
- street = 'Schützenmattstrasse'
- plz = 3012
- place = 'Bern'
- canton = 'Bern'

:Container
- description

:StringDescription
- label = 'Street'

:NumberDescription
- label = 'PLZ'
- required = true
- range = 1000..9999

:StringDescription
- label = 'Place'
- required = true

:SingleOptionDescription
- label = 'Canton'
- required = true
- sorted = true
- options = #( 'Bern' 'Zurich' ... )
result := anAddress asMorph
addButtons;
addWindow;
callInWorld
Seaside Rendering

result := self call: (anAddress asComponent
  addValidatedForm;
  yourself)
Other Applications

- Viewer building
- Editor building
- Report building
- Documentation
- Data validation
- Query processing
- Object filtering
- Object serialization
- Object copying

- Object indexing
- Object initialization
- Object extension
- Object adaption
- Object customization
- Code generation
  
  and much more ...

Thursday, 20 October 11
Why is it useful?

- Describe once, get everywhere.
- Extensibility of classes is ensured.
- Context dependent descriptions.
- End-user customizable.
- Developer configurable.
Why is it cool?

- Be more productive
- Lower coupling
- Do more, with less code
- Do more, with less hacking

- Empower your users
Thursday, 20 October 11
Run-time Adaptive Object Model
End users

customizability
Demo
Discussion

- Very powerful and flexible
- Runtime adaptive code
- End-user programmable code
- Can cause meta meta-confusion
- Can be slower than hardcoding
Any sufficiently complicated program contains at least one meta-model.