Modeling user interfaces with Cal and Ptolemy

Hallvard Trætteberg, Associate Professor
Dept. of Computer and Information Sciences
Norwegian Univ. of Science and Technology
Designers use informal representations...

Engineers use models
UsiXML [Vanderdonckt] – A family of XML-based notations for UI elements
Pet shop [Palanque] – Modeling safety critical UIs with ICO PetriNets
Lots of pragmatic approaches
(read: non-academic and useful)

- XML-based formats for describing user interface layout and style
  - XHTML (W3C), XAML (Microsoft), JavaFX (Oracle), XUL (Mozilla)
  - template languages for web pages
- DSLs
  - Ecore-based: Eclipse 4’s workbench model, Wazaabi
  - Xtext-based: APPlause, MOBL, Agentry
- Application modeling
  - Esito’s Genova – business applications for the desktop and web
  - WebRatio - business applications for the web
- Standardization
  - WebML
  - IFML (in progress)
  - Model-Based User Interfaces (MBUI) Working Group
IFML – Interaction Flow Modeling Language

- OMG RFP
- Proposal by WebRatio++
- Abstract UI model
- Functional units and view containers
- Dataflow and control/activation signals
Diamodl

Mail folders

Selection 1

Selection of leaf from tree

Mailbox

Selection 2

Selection of element from set

Message

View message

folder list

mailbox content

single message

The image depicts a flowchart illustrating the process of managing a mailbox. The flowchart shows the steps involved in accessing and viewing mail, including navigating through folder lists, accessing mailbox content, and viewing individual messages.
But what is

• the *semantics* of the model (runtime behavior)?
• the *role* of the model (scope/interoperability)?
Example – web browser
Three iterations

- **Take 1** – hand-code Ptolemy actors and Java Swing toolkit
  - showed the feasibility of using Ptolemy
  - lot of work writing generic and configurable actors
  - a specialized actor language would be nice, e.g. Cal

- **Take 2** – Cal implementation for Ptolemy runtime library
  - thin layer on top of atomic actors to support Cal implementation
  - extra Cal constructs for event handling, as an actor can be triggered by data and widget events, in addition to input on ports

- **Take 3** – moved to Javafx toolkit
  - more Cal constructs for UI state update
  - improved thread handling
Take 1

- Event-driven, use DE Director
- Load UI with SwixmlResource
- Break cycles with TimeDelay

Issues
- hand-coding actors is difficult and tedious
- Swing is being replaced by Javafx
Application architecture

- The whole runtime state is captured as coordinated graphs of data
- The widget hierarchy is continuously rendered on a device
Javafx widgets with fxml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<BorderPane maxHeight="Infinity" maxWidth="Infinity" minwidth="100" minHeight="0" prefHeight="100" prefWidth="100">
  <HBox>
    <Label text="location" />
    <TextField id="browser_location_text" prefWidth="400.0" text="http://www.idi.ntnu"
      Button id="browser_location_goButton" mnemonicParsing="false" text="Go" />
  </HBox>
</BorderPane>
```
Model of the controller
Model of the controller

Network of (instances of) reusable actors
Generic actors, based on Diamodl

Semantics!
Cal – generic actors

```java
namespace org.ptolemy.xtext.examples.ui :

import java.util.Collection
import java.util.ArrayList

actor Variable<T>(T initialValue=null, boolean allowNull=true, (T) -> boolean validator)
T* setValue ==> T value :
    var T current = initialValue;
    function diff(T oldValue, T newValue) --> boolean :
        (oldValue != newValue && (oldValue == null || !oldValue.equals(newValue)))
    end
    function isValid(T value) --> boolean :
        (allowNull || value != null) && (validator == null || validator.apply(value))
    end
    action [newValue] any ==> [current] when current != old_current do
        val T value = newValue.values.head
        if (diff(current, value) && isValid(value)) {
            current = value
        }
    end
```

Widget actors, wrappers/abstractions

What is the essential function of widget?
Cal – widget actors

Widgets
Cal implementation

• Xtext and Xbase provide tight integration with Eclipse platform
  – editor with syntax highlighting, code completion, navigation, ...
  – can refer to and use Java APIs (standard, third-party, custom)

• Implementation liberties
  – expressions – Java-like, closures, syntactic sugar, due to Xbase
  – atomic actors – event specifications
  – network – inline atomic actors, data transforming relations

• Runtime state
  – referring to contextual data
  – updating contextual data
  – threads
Summary

• Ptolemy as a platform for exploring and experimenting with semantics for domain-specific languages

• Utilize Ptolemy and Cal for developing apps
  – language, architecture and tooling issues