

Graphical User Interfaces in Haskell

By Gideon Sireling


August 2011




Outline

- ▶ Introduction
 - ▶ Literature Review
 - ▶ Analysis
 - ▶ Aims
 - ▶ Design
 - ▶ Packaging
 - ▶ Testing
 - ▶ Conclusions
 - ▶ Evaluation
 - ▶ Future Work
- 


Introduction

- ▶ Imperative vs. Functional
 - ▶ Haskell: Purely functional
 - ▶ Line-of-Business vs. Academia and Financials
 - ▶ New Challenges
 - Parallelism
 - Correctness
 - ▶ Graphical User Interfaces
- 


Literature Review

- ▶ Haskell
 - ▶ IO Monad
 - ▶ Toolkits
 - Gtk2Hs
 - WxHaskell
 - QtHaskell
 - ▶ Functional Reactive Programming
 - Events
 - Behaviours
 - Reactive Behaviour
- 

Analysis

- ▶ Imperative Toolkits vs. Declarative FRP
 - ▶ Wherefore art thou FRP?
 - ▶ Autonomous Components vs. Global State
 - ▶ Domain-Specific Abstractions
 - ▶ IO Monad as Glue
- 

Aims

- ▶ Data Binding
 - One Way
 - Two Way
 - ▶ Reuse the Wheel
 - ▶ Small, Simple API
 - ▶ Back to the IO Monad
- 

Design

▶ Variable Interface

```
class Variable v where
  newVar    :: a -> IO (v a)
  readVar   :: v a -> IO a
  writeVar  :: v a -> a -> IO ()
```

▶ Bindings

```
data Binding a = Binding (a -> d) t (t -> d -> IO ())
```

▶ Simple Data Sources

```
data Source v a = Source {bindings :: v [Binding a], var :: v a}
```

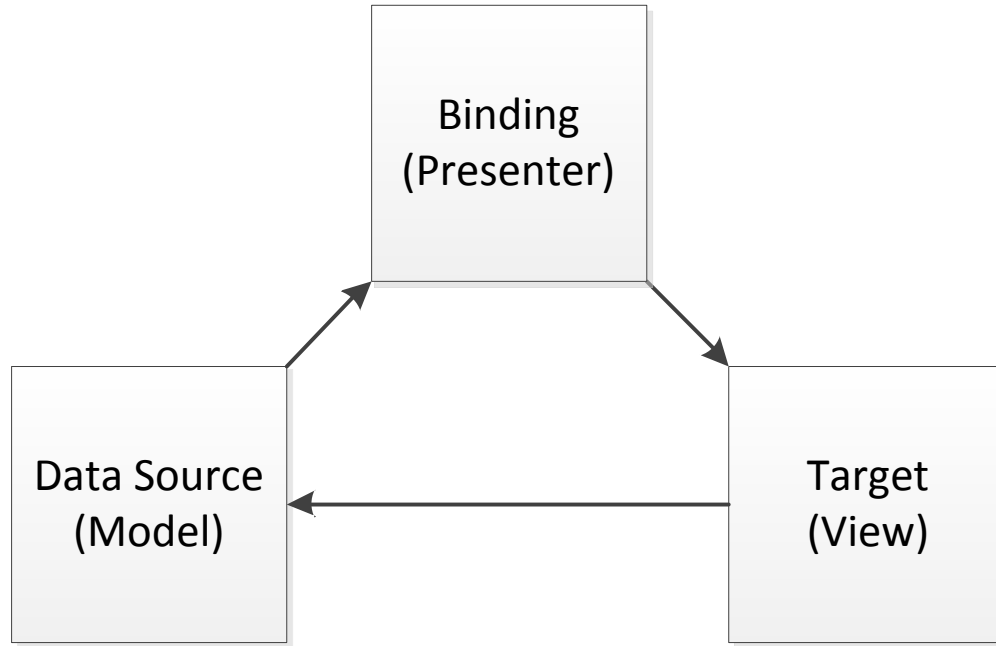
▶ Binding Lists

```
data BindingList v a = BindingList {source :: Source v a
                                     ,list  :: v [v a]
                                     ,pos   :: v Int}
```

▶ Binding Interface

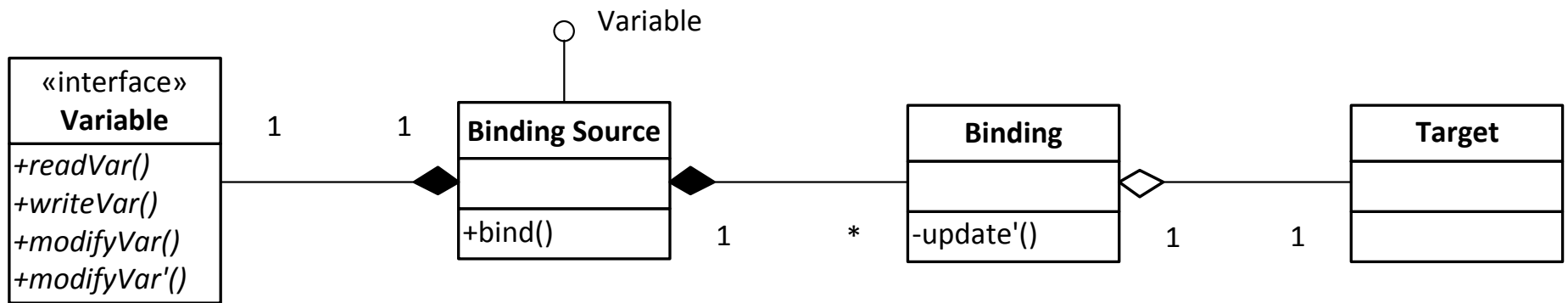
```
class Bindable b where
  bind :: b a -> Binding -> IO ()
```

Design



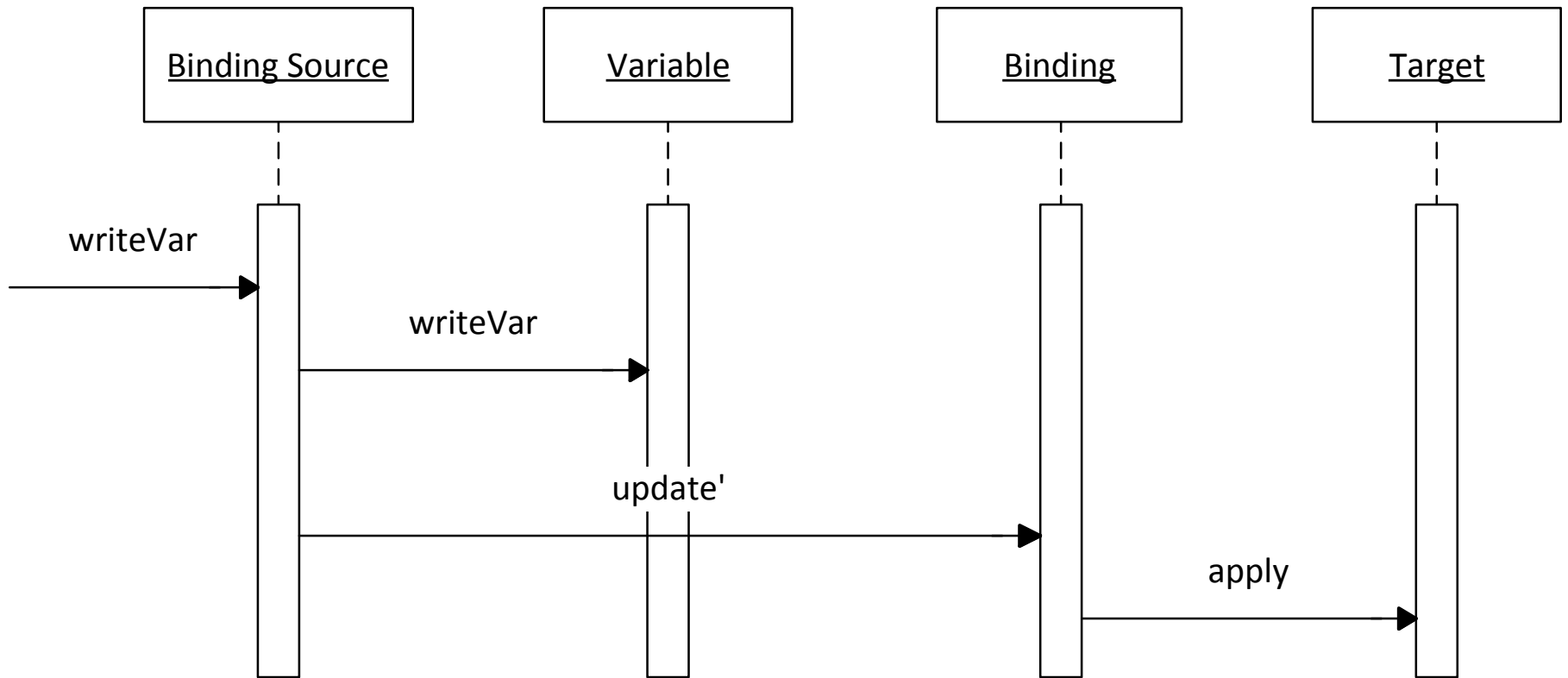
Model-View-Presenter

Design



Class Diagram

Design



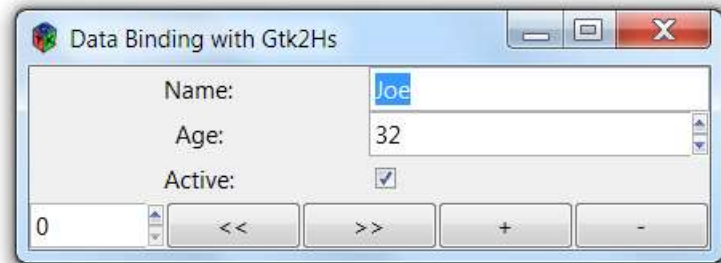
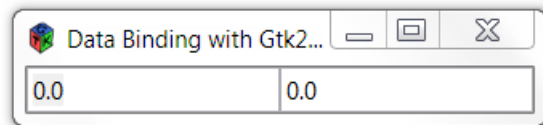
Sequence Diagram

Packaging


- ▶ **binding-core**
 - Core binding functionality
- ▶ **binding-gtk**
 - Gtk2Hs interface
- ▶ **binding-wx**
 - WxHaskell interface

Testing


- ▶ Unit Testing
 - HUnit
 - QuickCheck
- ▶ Integration Testing
 - Simple
 - List




Conclusions

- ▶ No one-size-fits-all abstraction for GUIs
 - ▶ IO Monad = Superglue
 - ▶ FRP = Animation, Arcade Games
 - ▶ Data Binding = Data Processing Applications
 - ▶ Haskell = Corporate Desktop?
- 


Evaluation

- ▶ Quixotic Quest
 - One abstraction to rule them all
 - ▶ Domain-specific frameworks
 - Build on existing toolkit bindings
 - ▶ Business applications
- 

Future Work

- ▶ Complex data binding
 - ▶ Master–detail relationships
 - ▶ Data Persistence
 - ▶ Mapping Functional Data Types to Relational Databases; the Saigon of Computer Science?
- 

Summary

- ▶ False Dichotomy?
 - Imperative Languages for Business
 - Functional Languages for Academia and Finance
 - ▶ Need more Functional Programming
 - Free lunches
 - Static verification
 - ▶ Functional programming languages must support imperative tasks
 - ▶ Data Binding for Data Processing Applications
 - ▶ It can be done!
- 

QUESTIONS?