



# The Goal

---

- **Facilitate Visual Programming**
  - **User Interfaces can be designed with no programming required**
  - **Applications can be assembled with no programming required**

*Personal computers can be assembled without a soldering iron. Why can't applications be assembled without programming?*

# The Reason

---

- **Use the right skills for the right job**
  - User interface specialists should be doing user interface design
  - Business analysts should work with customers and UI specialists for application flow
  - Core object designers should be working on threading models, design patterns and core business objects

*Professionals are most productive when they are doing the things they want to do.*

# JavaBean Review

---

- **Observer Design Pattern**
  - JavaBeans decouple interactions between objects by passing events instead of direct method calls
- **JavaBeans are modeled in terms of:**
  - Events (observer design pattern)
  - Properties
  - Methods

*In some integrated development environments (for example VisualAge), the properties generate different code than adding a field. Properties generate a default constructor and a different name. Methods don't create bean info and are not available in the beanbox unless they are created on the BeanInfo page.*

# The Context

---

- **Proper abstractions within the application are vital**
  - The object modeler should have the skills to determine the interfaces between components within the application
  - Abstractions are required to enable the core objects modeler to separate their work from the work of the user interface specialists and from the application assembler
  - *Design Patterns* is useful as a reference to learn abstraction techniques

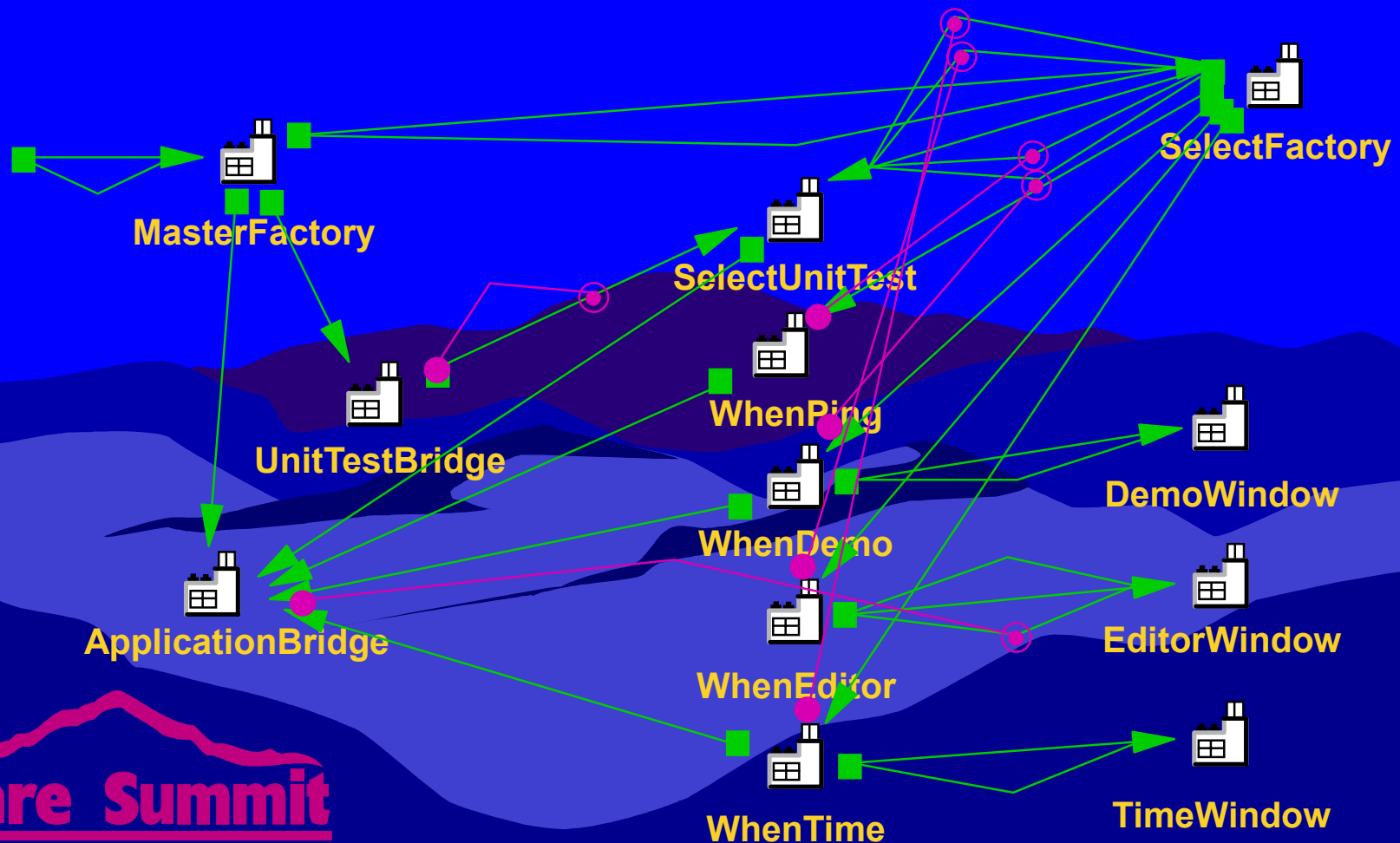
# The Context (Continued)

---

- **Visual programming makes the most sense in the context of an applications framework**
  - Layer 0 — Completely decoupled, generic objects
  - Layer 1 — Clusters of loosely-coupled objects
  - Frameworks Layer — Collections of clusters with much pre-determined behaviour.
  - Visual Layer — JavaBean-based layer that maps into the frameworks. It provides:
    - Wiring to functions in the frameworks
    - Notification of significant events, such as requests for information from the user or result sets from prior processing requests

# Demo

- Visually programming non-user interface parts is a reality...



# The Techniques

---

- **User Interface Layout**
  - Use bean box for visual layout of user interface components (*ala VisualBasic*)
  - Interactions between the components are also assembled using “wiring beans”

# The Techniques (Continued)

---

## ■ Wiring Blocks

- HBlock. No logic, just a place to collect a sequence of connections.
- HIfBlock. Conditional execution
  - User input validation in user interface beans
  - Basic logic and flow in non-interface beans

# The Techniques (Continued)

---

## ▪ Event Bridges

- Transition between “object clusters”
- True abstraction between application layers. There is no coupling between user interface and application model except the event bridge
- These are 90% generated by the bean box and 10% hand-coded
- These are key to effective design since they define what capabilities are available to analysts and user-interface designers
  - The bean box is unforgiving of changes
  - The addition of a method to the event listener “breaks” everything wired to it

# The Techniques (Continued)

---

## ▪ Event Bridges (Part 2)

### – Standard Programming Idiom

- Create the event(s) derived from `java.util.EventObject` that define the parameters/objects that need to pass between the JavaBean clusters (*i.e.* between the pages in the beanbox)
- Create the listener(s) derived from `EventListener` that define the interaction between the components.
- Create the JavaBean event bridge by adding the listeners as events and creating a method corresponding to each method in the listener

### – A sample follows...

# Transition Strategies

---

- **Visually-enabling code doesn't require that we start from scratch**
  - **Adapter design pattern**
    - **Create a specialized bean that calls your existing code, but presents a methods/events/properties**