Vending Machine

Terry Corlet, Josip Lozina and Stephan Brumme

June 3rd, 2004
Agenda

1. Introduction
2. Process
3. Implementation
4. UML Critique
5. Conclusion
Introduction - I

- identifying the **objective**
• setting up a plan
  – who?
  – when?
  – where?
  – what?

• decide on notation
  – UML

• team member roles
  → certain experiences?
• **iterative** approach
  
  - start with a **simple** model
  
  → refine to build the final model

  -- we came up with **8 revisions**!
• identifying requirements
  – what is clear?
    • 80 items available
    • several buttons
    • and many more ...

  – what is ambiguous?
    • which currency?
      – 5 cent rounding
      – smallest accepted coin
    • items out of stock
    • tracking sales
    • and many more ...

- CRC modelling
  - class diagram is integral
  - look for adequate names
  - semantics
- identifying patterns
  - well-known techniques
  - often directly mapped to UML structures
- Class diagram

- **TransactionController**
  - has a balance
  - contains coin

- **Product**
  - has a product
  - contains products

- **VendingMachine**
  - has a keypad
  - has a transaction controller
  - has displays
  - has a cancel button

- **Balance**
  - has a balance

- **CoinValidator**
  - has a coin validator

- **Coin**
  - contains coin

- **KeyPad**
  - contains numeric buttons

- **Button**
  - contains alpha buttons

- **NumericButton**
  - contains numeric buttons

- **AlphaButton**
  - contains numeric buttons

- **CancelButton**
  - contains numeric buttons
- Use case diagram

Diagram:

- Actor 1
- System
  - Purchase product
  - Select product
  - Pay for product
• State diagram
• Collaboration diagrams
  – mapping state diagram to OO structure
  – several diagrams!
• Sequence diagram
  – emphasize temporal relationships
Implementation

- done in C#

- to test robustness of our design

```csharp
using System;
using System.Collections;

namespace VendingMachine
{
    public class VendingMachine : Observer
    {
        const int numberOfRows = 8; // number of rows
        const int numberOfColumns = 10; // number of columns
        const string coordinate = "Ready"; // Ready message
        const string zeroFormat = "0.00"; // 0.00 message
        const string outOfOrder = "Out of Order"; // Out of order message
        const string nilStock = "Nil Stock"; // Nil stock message
        const float maxBalance = 10000.00f; // Maximum machine balance

        Hashtable products = new Hashtable(); // holding products
        float machineBalance = 0.0f; // machine balance
        Display purchaseDisplay = new Display(); // display for amount due
        Display selectionDisplay = new Display(); // display for selection messages
        Button cancelButton = new Button(); // button for cencel transaction
        CoinValidator coinValidator = new CoinValidator(); // coin validation
        TransactionController transactionManager; // for transactions
        Keypad selectionPad; // for user selections

        // default constructor
        public VendingMachine()
        {
            selectionPad = new Keypad(this);
            transactionManager = new TransactionController(this);
            display(purchaseDisplay);
        }
    }
}
```
UML Critique - I

- UML is not the customer’s language!
  - UML is a mix of several notations

- too many structural details
  - and no consistent level of detail (use cases vs. class diagram)

- no unique algorithm to design UML diagrams
  - hundreds of possibilities to model a problem with UML

- hard to draw diagrams without software (Visio)
UML Critique - II

- UML violates basic rules of visualization
  - human perception
  - Bertin’s variables

- no way to verify and validate requirements

→ try to take a look at competing modelling languages!
Conclusion

- all three team members have different background
  - ... but **UML** helped to speak the same language!

- initial effort to learn UML
  - but the last meetings were quite efficient and effective
  - time initially spent paid off

- software tools seem to get better and better

- developer community accepts UML
  - now essential **skill** of advanced developers
Questions