Smalltalk-based Visual Programming Tools

Janelle Carroll

VisaulAge and IBM Smalltalk Technical Support

Acknowledgments

Thanks to Martin Nally, chief architect and lead designer of VisualAge, for the extensive help in developing this presentation

What this talk is

- About visual programming tools like Visual Smalltalk* and VisualAge**
- From a user’s perspective
- About deploying real applications
- About current products

*Visual Smalltalk is a trade mark of Digitalk
**VisualAge is a trade mark of IBM

What this talk is not

- An academic talk
- About future directions
- A sales pitch for VisualAge
- Totally objective

Some questions to answer about these tools

- What are they good for?
- Do they scale?
- How do they fit with Smalltalk?
- Who should use them?
- How do they fit with existing tools and code?
- How do they fit with existing designs and methodologies?

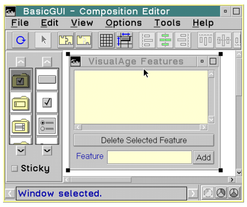
History of visual programming

- Mainframe screen painters (e.g. ISPF)
- GUI-builder tools (e.g. Smethers-Barnes Prototyper)
- NextStep
- Visual Smalltalk and VisualAge

Some simple examples of uses of visual programming

- GUI
- Simple Applications
- Simple Database Applications

Example 1: building a GUI layout

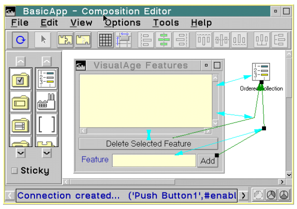


Window selected.

GUI layout: why does this work?

- Because the GUI layout is inherently visual (WYSIWYG)
- Because the GUI layout is a static arrangement of subparts

Example 2: building a minimal application



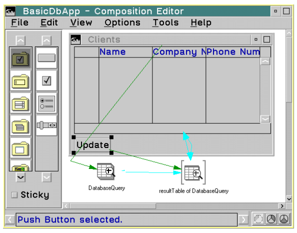
BasicApp - Composition Editor

Connection created... ('Push Button1',#enabl

A minimal application: why does this work?

- Because the processing components are a static arrangement of subparts
- Because the behavior can be expressed as an interaction between subparts

Example 3: building a simple database application



BasicDbApp - Composition Editor

Push Button selected.

A simple database application: why does this work?

- Because we designed a set of database parts for VisualAge
- Because the database components are a static arrangement of subparts
- Because the behavior is expressed as an interaction between subparts
- Because the query part integrated a custom visual programming environment

Advantages of visual programming

- Much higher level of expression
- Allows domain experts to do part or all of application development
- Allows an incremental adoption of OO and skill-building
- Very powerful for many aspects of application development
  - GUI, Reports, Database access, Transaction invocation, many others

Who should use Visual Programming Tools?

- Teams who want higher-level tools and productivity
- Companies looking to bridge to OO technology
- OO Teams looking to involve non OO specialists
  - UI or graphics designers, Domain experts, Database experts, Prototypers

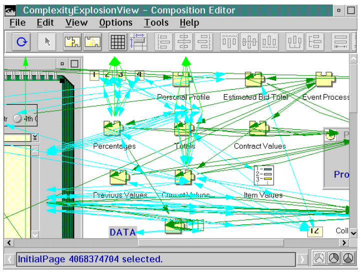
What should they use Visual Programming Tools for?

- Prototyping
- JAD
- Deployment of production applications

Difficulties of visual programming

- Not effective for every problem
- Tendency to skip design
- Scaling of complexity
- Debugging and performance tuning

Example 4: complexity explosion



InitialPage 4068374704 selected.

Complexity explosion (cont'd)

- Visual spaghetti code is easily created
- Fundamental questions:
  - Does this example illustrate problems with Visual Programming, or does Visual Programming illustrate problems with this example?
  - Can Visual Programming scale?

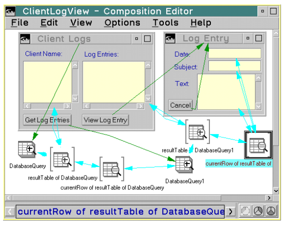
Complexity explosion: why does this not work?

- Design problems
  - Poor factoring: many low-level objects and connections should be encapsulated
  - Poor design of custom processing elements
  - View is entire application
- Sub-optimal use of visual programming techniques
  - Many objects and connections for simple initialization

The 3 Most Effective Solutions

- Design
- Design
- Design

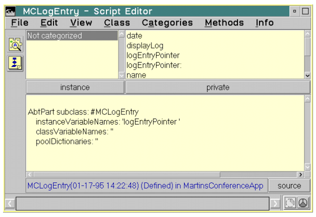
Refining a simpler example


ClientLogView - Composition Editor
File Edit View Options Tools Help
Client Logs
Client Name:
Log Entries:
Get Log Entries
View Log Entry
Log Entry
Date:
Subject:
Text:
Cancel
DatabaseQuery
resultTable of DatabaseQuery
currentRow of resultTable of DatabaseQuery
resultTable
DatabaseQuery1
currentRow of resultTable of
DatabaseQuery1
currentRow of resultTable of DatabaseQue

Techniques for managing complexity

- Writing custom model parts in the Smalltalk language
- Writing simple scripts in the Smalltalk language
- Building reusable visual programs and using them as subparts
  - Visual Parts
  - Non-Visual Parts

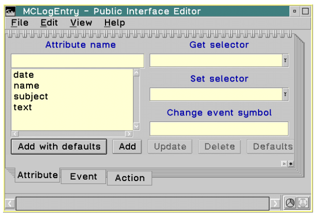
Example 5: building a custom part in Smalltalk



```
AbtPart subclass: #MCLogEntry
   instanceVariableNames: 'logEntryPointer '
   classVariableNames: ''
   poolDictionaries: ''
```

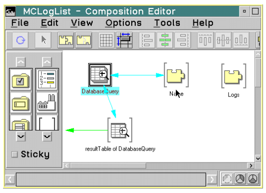
MCLogEntry(01-17-95 14:22:48) (Defined) in MartinsConferenceApp

Example 5 (2): building a custom part in Smalltalk

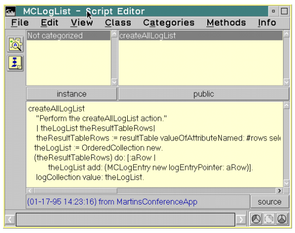


date
name
subject
text

Example 6: building a reusable Non-visual part



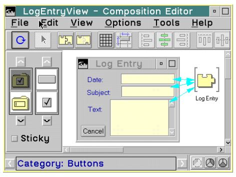
Example 7: writing a script



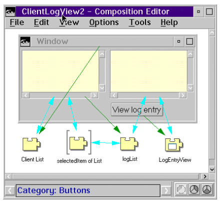
```
createAllLogList
   "Perform the createAllLogList action."
   | theLogList theResultTableRows|
   theResultTableRows := resultTable valueOfAttributeNamed: #rows sel
  theLogList := OrderedCollection new.
  (theResultTableRows) do: [:aRow |
       theLogList add: (MCLogEntry new logEntryPointer: aRow)].
   logCollection value: theLogList.
```

(01-17-95 14:23:16) from MartinsConferenceApp

Example 8: building a reusable Visual Part



Putting it together



Writing a script

- First level of integration with Smalltalk
- Scripting skill level, not arbitrary Smalltalk design and implementation

When to write a script

- Initializing variables
- Complex Boolean conditions
- Complex cross-field input validation
- Iterations

Building a reusable Visual or Non-Visual Part

- Can be performed entirely visually
- Usually the first step towards an object design
- Teaches a number of OO concepts: an object, its external interface and usage, and its implementation without requiring any OO language skills

When to build a reusable Visual or Non-Visual Part

- Whenever there is a chance of reuse (almost always)
- Whenever a logical entity can be extracted to reduce complexity and facilitate maintenance

Building Custom Parts in Smalltalk

- Second level of integration with Smalltalk
- Key feature distinguishing Smalltalk-based tools from many other RAD tools
- Allows limitless flexibility while retaining benefits of Visual Programming
- Allows implementation of arbitrary designs with Visual Programming
- Allows mixed skill levels on teams

When to build a custom part in Smalltalk

- If the part is not easily decomposed into a static arrangement of subparts
- If the behavior of the part cannot be expressed easily in terms of interactions between its subparts
- If the part has many complex logical decision points or iterations
- If the part requires ‘low-level coding’ (e.g. calling system services)

Examples of custom parts in Smalltalk

- A domain object (customer, bank account, oil well...)
- A spell-checker
- A loan interest rate calculator
- A wrapper of a set of system APIs

VisualAge choices for custom parts implementation

- Inherit from Object (VisualAge will create generic wrapper at runtime)
- Inherit from Object but implement VisualAge part protocol
- Inherit from AbtPart
- Write custom wrapper

Custom parts that inherit from Object

- Essentially free integration
- Some performance overhead and flexibility restrictions
- Works well for existing classes
  - Currently used in VisualAge for OrderedCollection, Dictionary etc.

Custom parts that implement part protocol

- Requires a little more implementation
- Gives best flexibility and performance
- Works well for existing classes

Custom parts that inherit from AbtPart

- Essentially free integration
- Gives best flexibility and performance
- Works best for new implementation targeted to VisualAge
  - Currently used in VisualAge for many standard parts

Custom part wrappers for existing parts

- Gives excellent flexibility
- Requires maintenance of wrapper
- Small performance overhead
- Used to enhance VisualAge integration of existing classes
  - Currently used in VisualAge for all GUI parts

Integrating non-Smalltalk code

- C and COBOL DLLs can be integrated as parts
- Network connections can be integrated as parts
- CICS transaction objects can be integrated as parts

Integration with Smalltalk Tools

- ■ Configuration Management and Team Programming
  - ◆ Each vendor provides integration with its tools
- ■ Debugging
  - ◆ High-level debugging tools available
  - ◆ Still sometimes necessary to debug at the Smalltalk level (VisualAge)
  - ◆ Smalltalk debugger pops up for some Visual Programming errors (VisualAge)

Integration with Smalltalk Tools (cont’d)

- ■ Performance profiling
  - ◆ Works with existing Smalltalk profilers.
  - ◆ Still necessary to profile at the Smalltalk level (VisualAge)
- ■ GUI Builders
  - ◆ WindowBuilder/Pro screens can function as VisualAge parts

How do they fit with existing designs and methodologies

- ■ Mapping existing design elements to Visual Programming concepts
  - ◆ In VisualAge each part is a class
- ■ Model-View separation
  - ◆ VisualAge variable parts allow models to be passed to views.
- ■ Control Objects
  - ◆ VisualAge gives support for deferred update operations, objects that implement changes

Summary of Visual programming tools

■ - Good for prototyping, JAD and production applications.

- Allow a broad range of specialties and skill levels
- Provide an incremental adoption path for OO
- Can provide a significant productivity gain

Summary of Visual programming tools (2)

■ - Can scale to large, sophisticated apps

- Attention to design and training avoid incomprehensible, unmaintainable programs.
- It helps to have expert Smalltalk programmers to create custom parts
- Learning to create simple, well-designed, reusable parts with Visual Programming is a key to success, especially as applications grow

Summary of Visual programming tools (3)

■ - They integrate smoothly with Smalltalk

- They allow scripts in the Smalltalk language to supplement Visual Programs
- They allow custom parts to be created in Smalltalk

Summary of Visual programming tools (4)

- They integrate well with existing tools and code
  - They provide a powerful integration environment for existing code
  - They complement existing skills in Smalltalk
  - They integrate with existing debuggers, profilers and team and configuration management tools

Summary of Visual programming tools (5)

- - They can be used with existing designs and methodologies?
  - Designs created with existing methodologies can be implemented or partially implemented with Visual Programming.
  - Pieces of a design that don’t lend themselves to Visual Programming can be implemented in straight Smalltalk (or a 3GL programming language) and easily integrated with Visual Programming.

Summary of Visual programming tools (6)

- Everyone should use them