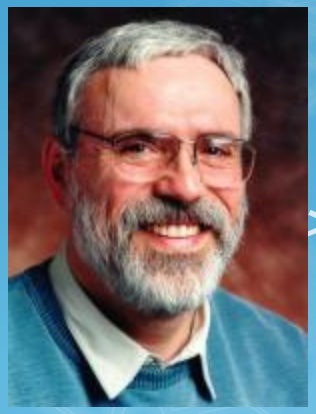
The Pragmatics of Model-Driven Development

Bran Selic, IBM Rational Software



Outline

- Define MDD
- Code generation (rant)
- Quality
- Pragmatics
- Conclusion + comments

Introduction

- Software is evolving slowly
- 3rd generation languages began in late 1950's
- Object Oriented Design
- "Even Classes and Inheritance are underused"

What is MDD?

- Using models as primary artifact
- Automation
- Model transformation
- Model verification
- Code generation
- NOT a set of rules to follow
- It is an IDEA – Where is the line?

Acronyms

- MDA
- MDE
- MDD
- MDSD
- MDSE

- MDE first coined by Stuart Kent (U of Kent, 2001).

Why MDD?

- Models are easier to:
  - Specify
  - Maintain
  - Understand

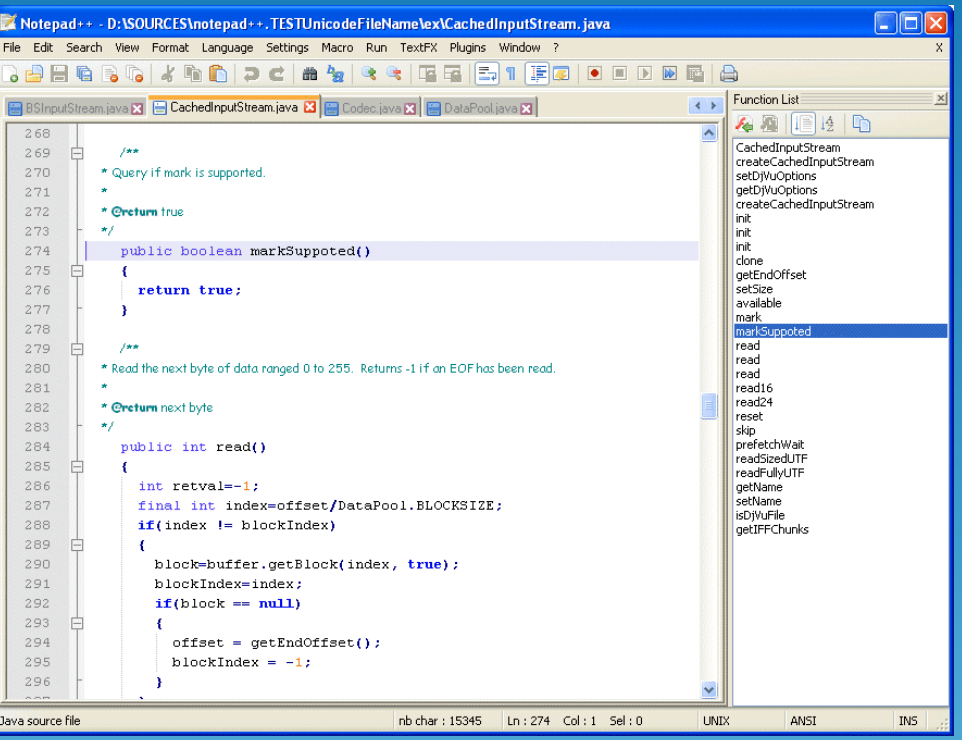
Why do we need it

- MDE aims to:
  - Improve short term productivity
  - Improve long term productivity
- Improve software ROI
- Reduce sensitivity to change – artifacts are used longer

Why do we need it

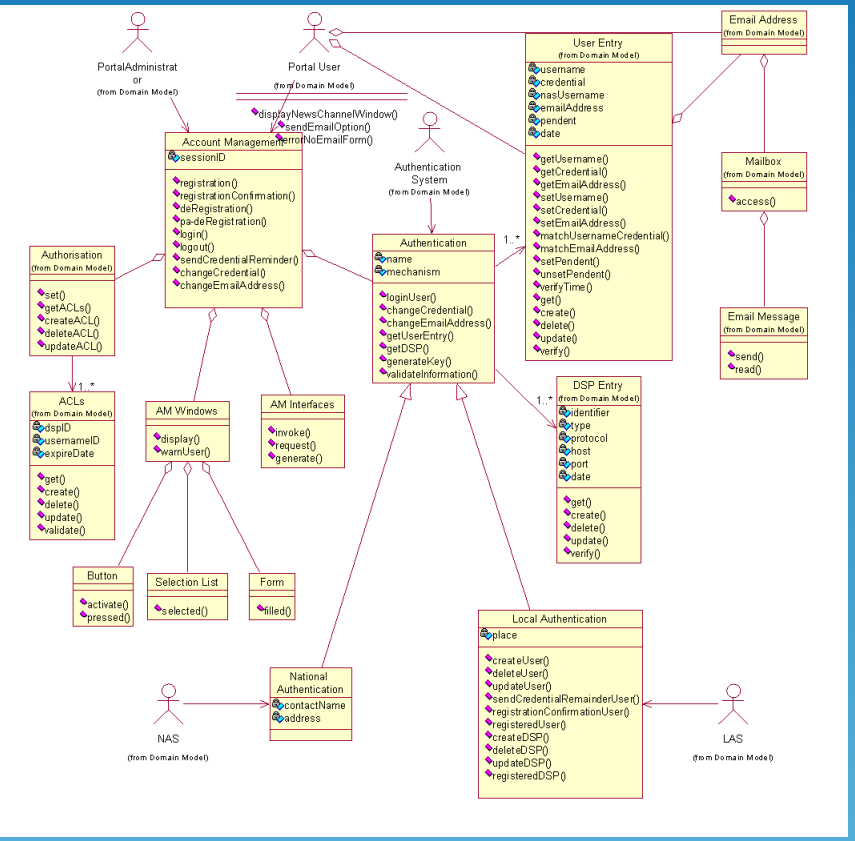
- Faster
- Less prone to errors
- Automated documentation
- Empowers DOMAIN EXPERTS

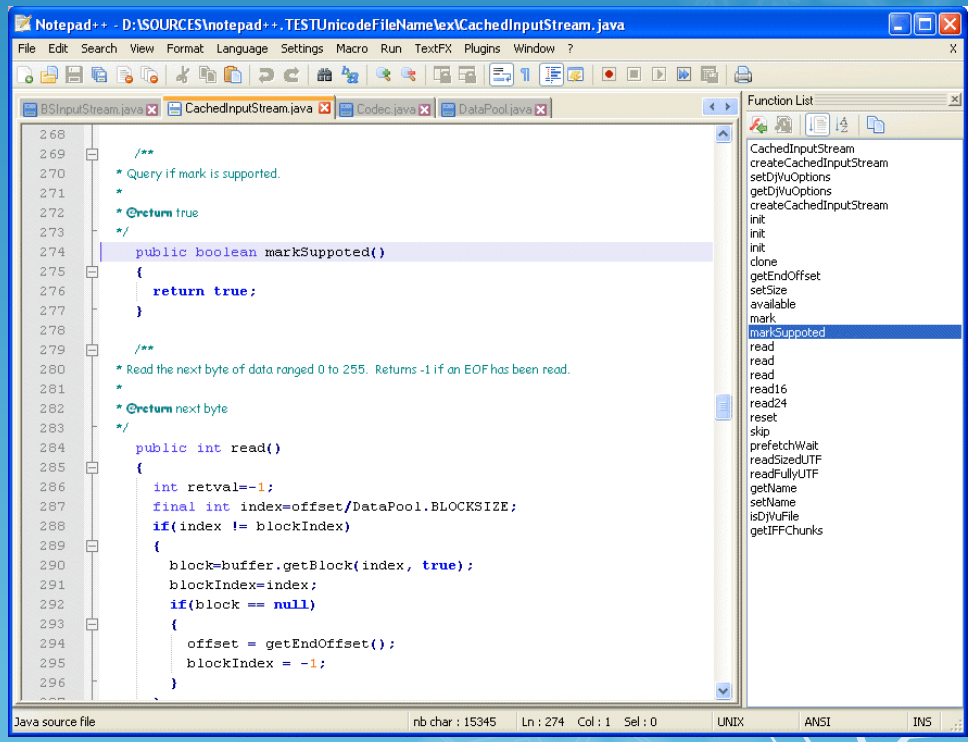
Code generation




```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2


C000                    ORG     ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START    LDS     #STACK

                        ***************************************
                        * FUNCTION: INITA - Initialize ACIA
                        * INPUT: none
                        * OUTPUT: none
                        * CALLS: none
                        * DESTROYS: acc A

0013                    RESETA  EQU     %00010011
0011                    CTLREG  EQU     %00010001

C003 86 13      INITA   LDA A   #RESETA   RESET ACIA
C005 B7 80 04           STA A   ACIA
C008 86 11              LDA A   #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A   ACIA

C00D 7E C0 F1           JMP     SIGNON    GO TO START OF MONITOR

                        ***************************************
                        * FUNCTION: INCH - Input character
                        * INPUT: none
                        * OUTPUT: char in acc A
                        * DESTROYS: acc A
                        * CALLS: none
                        * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04   INCH    LDA A   ACIA      GET STATUS
C013 47                 ASR A             SHIFT RDRF FLAG INTO CARRY
C014 24 FA              BCC     INCH      RECIEVE NOT READY
C016 B6 80 05           LDA A   ACIA+1    GET CHAR
C019 84 7F              AND A   #$7F      MASK PARITY
C01B 7E C0 79           JMP     OUTCH     ECHO & RTS

                        ***************************************
                        * FUNCTION: INHEX - INPUT HEX DIGIT
                        * INPUT: none
                        * OUTPUT: Digit in acc A
                        * CALLS: INCH
                        * DESTROYS: acc A
                        * Returns to monitor if not HEX input

C01E 8D F0      INHEX   BSR     INCH      GET A CHAR
C020 81 30              CMP A   #'0       ZERO
C022 2B 11              BMI     HEXERR    NOT HEX
C024 81 39              CMP A   #'9       NINE
C026 2F 0A              BLE     HEXRTS    GOOD HEX
C028 81 41              CMP A   #'A
C02A 2B 09              BMI     HEXERR    NOT HEX
C02C 81 46              CMP A   #'F
C02E 2E 05              BGT     HEXERR
C030 80 07              SUB A   #7        FIX A-F
C032 84 0F      HEXRTS  AND A   #$0F      CONVERT ASCII TO DIGIT
C034 39                 RTS

C035 7E C0 AF   HEXERR  JMP     CTRL      RETURN TO CONTROL LOOP
```

Code generation



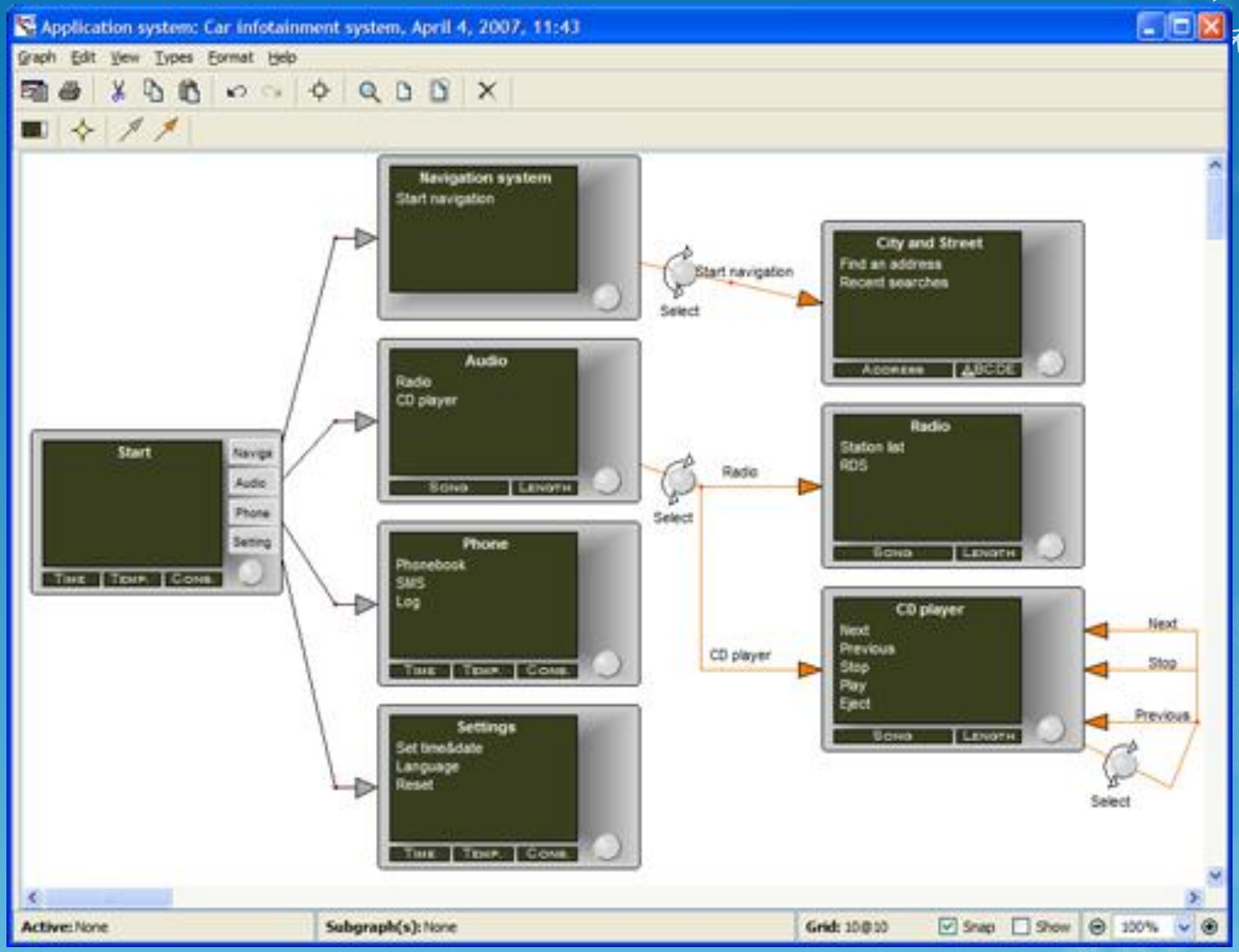


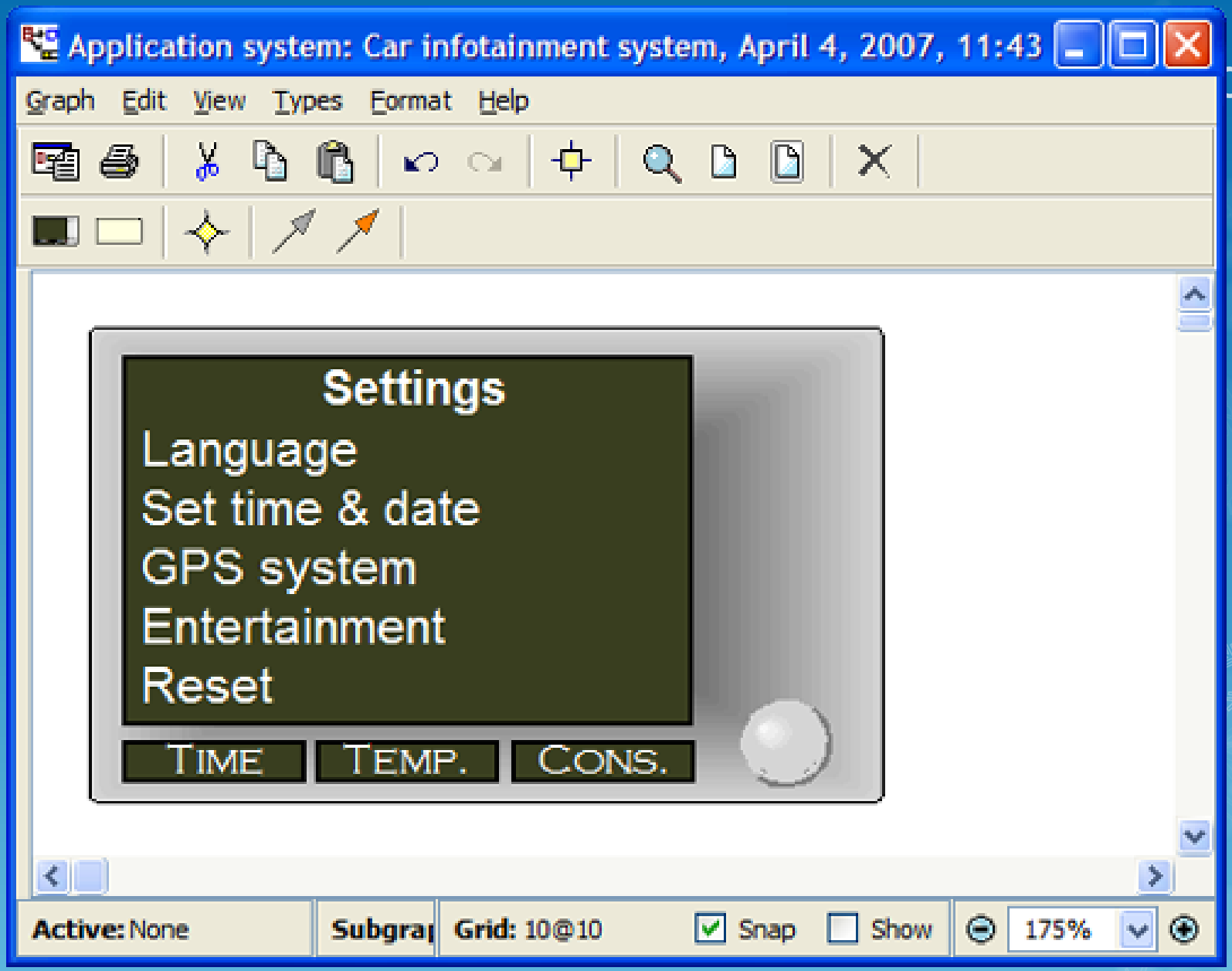
Code Generation

- Just skeletal?
- Relation of code to model
- Round trip engineering
- DSL code generation

Code Generation

"A key premise behind MDD
is that programs are automatically generated
from their corresponding models"


Application system: Car infotainment system, April 4, 2007, 11:43
Graph Edit View Types Format Help
Navigation system
Start navigation
Audio
Radio
CD player
Phone
Phonebook
SMS
Log
Settings
Set time&date
Language
Reset
Start
Navigat
Audio
Phone
Setting
Select
Start navigation
City and Street
Find an address
Recent searches
Select
Radio
Radio
Station list
RDS
CD player
CD player
Next
Previous
Stop
Play
Eject
Next
Stop
Previous
Select
Active: None
Subgraph(s): None
Grid: 10@10
Snap
Show
100%

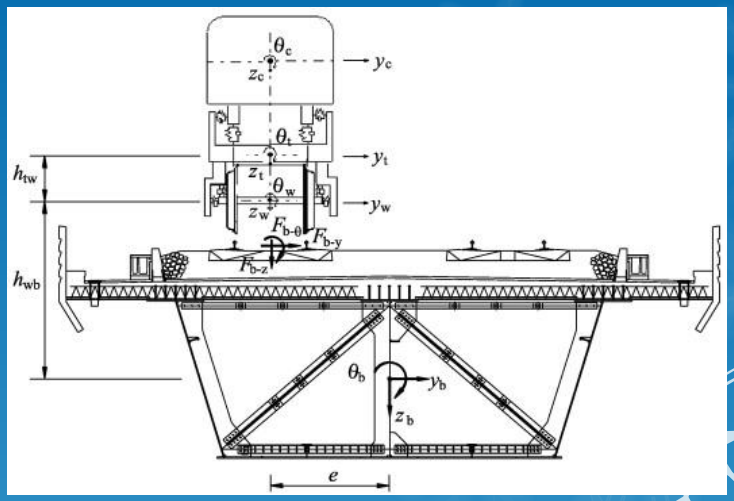

Application system: Car infotainment system, April 4, 2007, 11:43
Graph Edit View Types Format Help
Settings
Language
Set time & date
GPS system
Entertainment
Reset
TIME
TEMP.
CONS.
Active: None
Subgra
Grid: 10@10
Snap
Show
175%

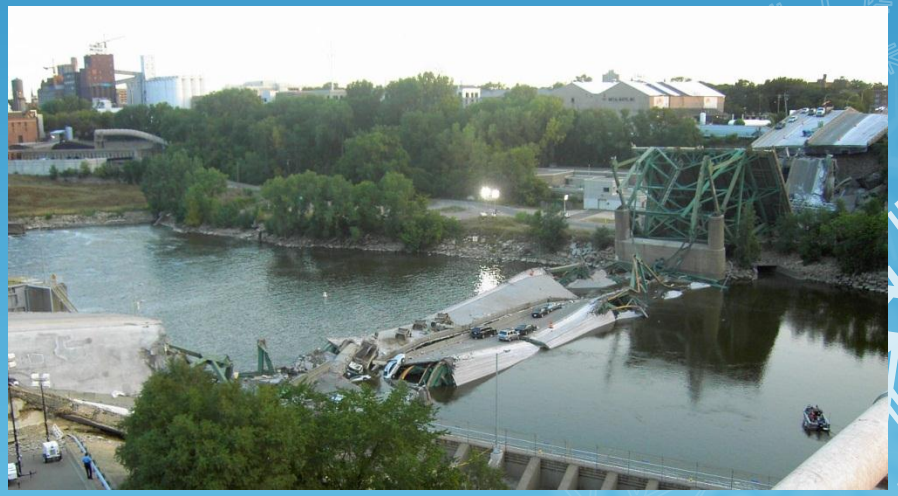
http://www.metacase.com/cases/autoinfo.html

And now back to the paper…

Quality of a Model

- Abstraction
- Understandability
- Accuracy
- Predictiveness
- Inexpensive


θ_c
z_c
y_c
θ_t
y_t
h_{tw}
z_t
θ_w
y_w
z_w
$F_{b-\theta}$
F_{b-y}
F_{b-z}
h_{wb}
θ_b
y_b
z_b
e



Executability

- Must have ability to:
  - Start
  - Stop
  - Resume
  - Automated instrumentation
- Executing + experimentation models helps learning
- “Hello world” raises confidence

Executability

Scalability

- Intended for large corporations / large software
- Metrics:
  - compilation time – full and partial
  - Code generation time
  - System size

Efficiency

- Modern compilers outperform manual coding
- Two categories:
  - performance
  - memory management
- Model to code is within 5%-15% efficiency of manual
- Will improve!
- Occasionally need to add manual code to generated code
- Used as an excuse to reject MDD

Integration

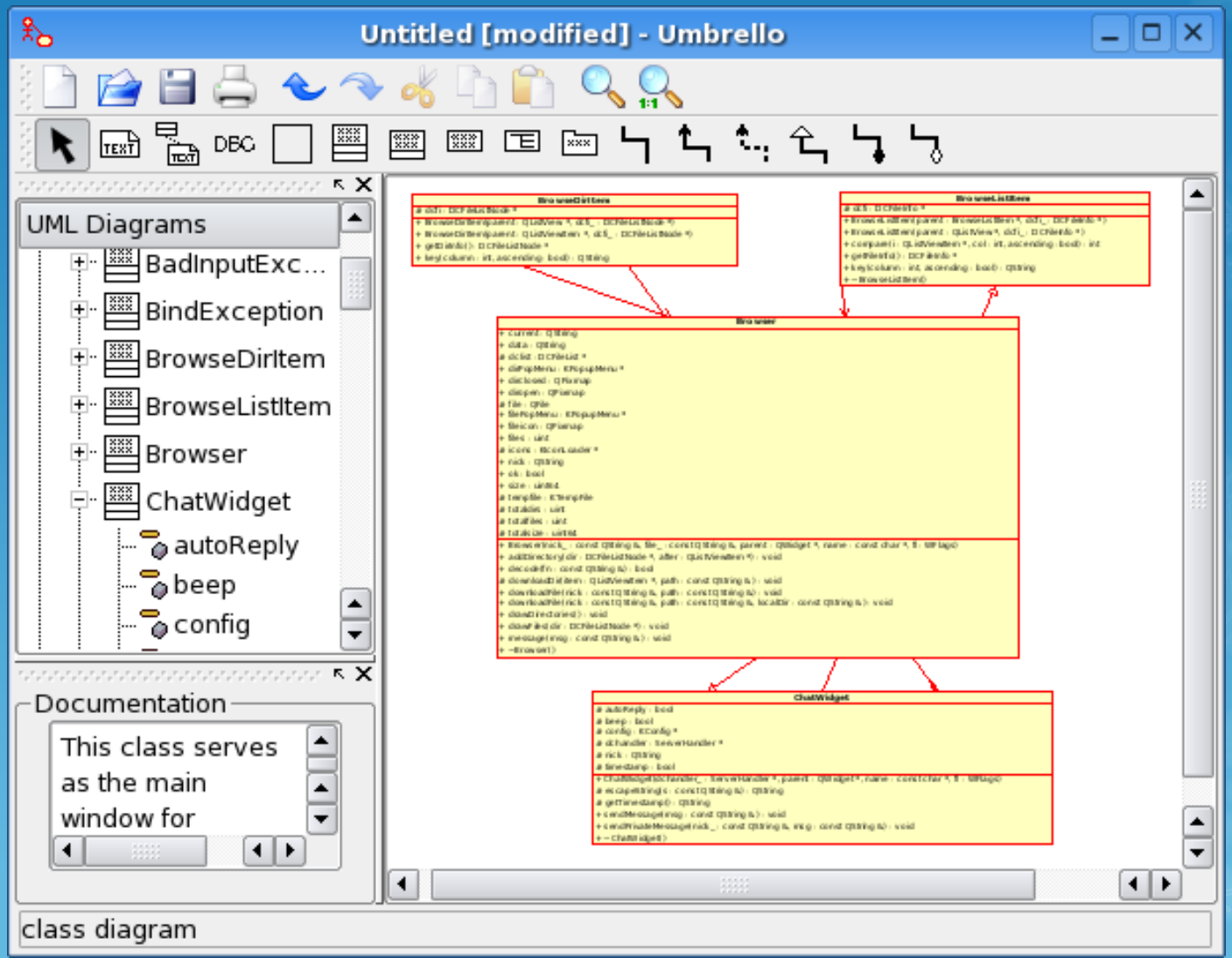
- Need compatibility with existing
  - Compilers
  - Build utilities
  - Debuggers
  - Code analyzers
  - Software versioning control systems
  - Legacy code

Integration

- Begin with small projects or extensions
- CANNOT require purchasing of new systems
- Customizable code generators
- Allow direct calls to such utilities from within the model

Why hasn’t it caught on?

CASE


Untitled [modified] - Umbrello
UML Diagrams
BadInputExc...
BindException
BrowseDirItem
BrowseListItem
Browser
ChatWidget
autoReply
beep
config
Documentation
This class serves as the main window for
class diagram

CASE Dejavu

- Uses:
  - Diagrammatic Tools
  - Information Repository
  - Interface Generators
  - Management tools
  - Process modelling
  - Others
- Promised:
  - high-quality,
  - defect-free
  - maintainable software products

CASE

- Enforced a way of
  - Working
  - Modelling
  - Coding
- Tools didn’t match existing coding practices
- Single tool – not tuned to developer needs
- Resulting code required too much modification
- If code needed editing – must manage TWO systems
- Will MDE encounter same pitfalls?

Now back to MDE

- Fears
  - Generated code not as efficient
  - Wastes time and money
- Focused too much on academics
- Large culture change
- How do we push it to industry?

Why hasn’t it caught on?

- Requires new skill set
- Models are abandoned when too much modification is required.
- High risk!
- Universities teach compilers and parsers. What about DSL, or code generators.

Why hasn’t it caught on?

- Lack of tools, or undeveloped (i.e version control)
- Using MDD with little experience
- Too many promises (platform independence)
- Learning curve
- Cost of creating the tool
- Are we simplifying the process but making it harder at the same time?

Why hasn’t it caught on

I believe the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation. We still make syntax errors, to be sure; but they are fuzz compared with the conceptual errors in most systems.
If this is true, building software will always be hard. **There is inherently no silver bullet.**

Why hasn’t it caught on

For almost 40 years, people have been anticipating and writing about "automatic programming," or the generation of a program for solving a problem from a statement of the problem specifications. Some today write as if they expect this technology to provide the next breakthrough.

“automatic programming always has been **a euphemism for programming with a higher-level language** than was presently available to the programmer”

Quick Demos

- Visual Studio / WPF
- Labview - http://www.youtube.com/watch?v=xpsQFvc7v5E&feature=related

Comments

- Introductory paper
- Not much new
- Who is the target audience?