The Model Driven (R)evolution

Richard Mark Soley, Ph.D.
Chairman and CEO
Object Management Group, Inc.
Modeling Changes Everything!

- Throw out those pesky objects!
- Toss away your silly compilers!
- No more boring coding!
- All your software pain gone forever!

It’s a REVOLUTION!
Everything Old is New Again

- Unfortunately I’m old enough to remember
  - Artificial Intelligence
  - Object Technology
  - Distributed Computing
  - XML
  - Web Services
  - Enterprise Service Bus
  - Service Oriented Architecture

- This technology does everything! It makes miracles, changes water to wine…
Move to Model Driven Everything!

(That’s a model, driving, get it?)
Pictures from Mars!

• Um, did that require a PIM?
OK, Calm Down

- Got that out of your system?
- Have we seen this before?
Everything Old IS New Again

- Refactoring design
- Object orientation
- Service orientation
- Legacy transformation
- Business process re-engineering
What is the Point?

• Reuse
• Interoperability
• Portability
• Maintainability
• Productivity
• Business Alignment
What is the Priority?

**Analysis, Design, Development, Test & Deployment:** 10%

**Maintenance & Integration:** 90%

*Lesson: Software lifecycle costs are in the back end.*
Where is the Current Focus?

• Initial development productivity
  – Wizards
  – Generators
  – Even open-source

• Flash vs. form
  – Demo programs
  – Whiz-bang user interfaces
  – GUI’s, even on the server

• MDA focusing where the pain is
Because Otherwise We’re All Just...

...roadkill on the information highway!!
We Must Be Able To…

- Capture enduring design
- Separate capture of process from engineering of implementation
- Automate the latter as much as possible
- Design-in agility

- The key ideas: **enduring**, **automated** and more importantly **agility**
**What is “Model Driven”?”**

- Graphical description of process
  - Captures *design* with a minimum amount of artifacts caused by the *language*
  - Separates *modeling* and *transformation*
  - Automates (somewhere from part to all) creation of implementation artifacts (schemas, deployment descriptors, programming language text, scripts, etc.)
Haven’t We Seen This Before?

• Well, yes: we have a clever name for tools that take precise, more abstract descriptions and transform them automatically to precise, less abstract (more concrete) descriptions.
Compilers!
We Owe it all to John Backus

- This clever technology actually dates to 1954: SPEEDCODING and FORTRAN
John Backus’ Pain

• Coding for the IBM Selective Sequence Electronic Calculator (SSEC) was painful (especially due to the lack of index registers and floating point)
• Backus considered programming “hand-to-hand combat with the machine”
• His solution: SPEEDCODYING, an assembly-language aid to automate translation of pseudo-index registers and pseudo-floating point
The Birth of High-Level Language

- For IBM’s new “supercomputer” (the 704), something better had to be done
- Backus’ team came up with the FORmula TRANslating system (FORTRAN) in ‘54
- They called it automatic programming 😊
FORTRAN: Yes, it’s an HLL

• That was 1954, this is now
• Perhaps FORTRAN isn’t considered high-level today, but it’s still hugely successful
• The key idea was to maintain precision but raise the level of abstraction
• FORTRAN programmers worried about the algorithm (well, more at least), while...
• ...compiler developers worried about the transformation.
Resistance Was Futile

• Most programmers “knew” that they could write better code themselves (some were right)
• Many more people became programmers (but they were programming abstract “FORTRAN machines,” not 704’s)
• The day parentheses died 😞
Modeling Isn’t New

• Just the next higher abstraction level
And It’s Fractal

- Why just three levels?
- CIM’s, PIM’s and PSM’s
Everything Old is New Again

– All the problems Backus faced are with us:
  • Is the generated code (artifact) as good as hand-generated?
  • How do you debug something you’ve never seen?
  • Who owns, controls and tests the transformations?
  • How do you audit models?
– Those of us who remember IBM 360’s remember:
  • Program in FORTRAN…
  • …but debug a core dump.
Graphical Language Are Scary

- Real Programmers Don’t Draw

<table>
<thead>
<tr>
<th>SPACE TELESCOPE SCIENCE INSTITUTE</th>
<th>STRUCTURE EQUATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>)aRelativistic Structure Equations</td>
<td>aRelativistic Structure Equations</td>
</tr>
<tr>
<td>x+(P[1]÷u)×3 a Relativity Parameter</td>
<td>x+(P[1]÷u)×3 a Relativity Parameter</td>
</tr>
<tr>
<td>q+ux('EOS =NEwTON np,x,3*P)*3 a Density</td>
<td>q+ux('EOS =NEwTON np,x,3*P)*3 a Density</td>
</tr>
<tr>
<td>D=4×o(S+P[2]×2)×q a Mass Continuity</td>
<td>D=4×o(S+P[2]×2)×q a Mass Continuity</td>
</tr>
<tr>
<td>r&lt;1+P[4]÷c×q,÷0÷P[2]÷x,3÷l,×2,1÷1−2÷c÷g,c,P[2÷2]</td>
<td>r&lt;1+P[4]÷c×q,÷0÷P[2]÷x,3÷l,×2,1÷1−2÷c÷g,c,P[2÷2]</td>
</tr>
<tr>
<td>D=D,-g×P[3]÷S÷q÷x÷r a HS Equilibrium</td>
<td>D=D,-g×P[3]÷S÷q÷x÷r a HS Equilibrium</td>
</tr>
</tbody>
</table>

The first write-only language?
All the Same Structures

• But of course all of the things we find in the text world are in the graphical modeling world too:
  – Flexible
  – Pluggable models (libraries)
  – Standard models
  – Patterns of usage

• We’ve just moved all of them up a level (or more) of abstraction
Many of the Same Problems

- Bad models are easy to build
- The wrong design does the wrong thing
- Still need some sort of development methodology for consistency and quality
- Architecture is a good idea
- Training is required
Don’t Ignore the Costs

- This is a sea change for most development teams
- Jobs may sort out differently than currently
- Audit requirements based on code have to be updated
- *Training is required*; certification too
- Integration with current methodology is critical
- That old code just isn’t going away
- *Don’t tell me you’ve never seen embedded assembly code?*
Modeling: Key Concepts

- **Emphasis on transformation techniques**
  - Based on a standard metamodeling framework; there will be many metamodels, and plenty of modeling languages (including UML)
  - Clear semantics, expressed consistently
  - Potentially many levels of abstraction

- *Enduring architectures* are the focus
  - Maintenance and integration aren’t pretty, but they are the main job of IT

- **Graphical languages as well as textual ones**
  - Some generic, some domain-specific, just like the textual language world
Generation Isn’t Everything

• Sometimes we’ll be able to generate all the
  – Code
  – Schemas
  – Deployment descriptors
• Sometimes we won’t; but we’ll still have the modeling values of
  – Clear, sharable graphical expression
  – Flexible transformation for agile retargeting
  – An enduring description of the system
• Architecture matters (that’s why MDA)
• (That’s what you call engineering)
Developer Roles Change

• Developers become more productive, not redundant, with focus on:
  – Requirements Analysis
  – Analyst/Designers
  – Architects
  – Analyst/Programmers
  – Testers
  – Maintainers/Integrators

• All sharing a language or set of languages with a common underpinning

Who’s Doing It

• Modeling has quietly changed the world
  – Up to 1997, dozens of languages, dozens of tools, a US$30MM market
  – From 1997, an initial common language (UML), one base metamodeling framework (MOF), dozens of tools (Microsoft, Rational, etc.)
  – From 2001, a sea change in IDE’s:
    • Open Source (Eclipse NetBeans, Poseidon)
    • Standardized (Adaptive, Codagen, Data Access, IBM, iO, MID, Sun, many others)
    • Even proprietary 😊
  – Today a US$4B market
Conclusions

- Every IDE supports model-driven today
- You need to look into it now
- Even if you plan to use a DSL, your organization needs to understand standardized frameworks (UML, MOF)
- Standards for infrastructure (MDA, UML, MOF) exist; many vertical standards exist and more are in development (that’s what DSL’s are!)
- The “real hacker” of tomorrow is the transformation developer
- Don’t forget: people still write assembly code
OMG’s Take on Modeling

• A standardized architecture, MDA
  – UML, MOF, XMI, CWM, QVT: the right starting points for enduring, agile, transformable systems
  – Vertical-market standards (domain-specific models) in many areas
• http://www.omg.org/mda/
One Final Word

“Not all evolution mandates revolution”
Leo McGarry
The West Wing
Conclusions

• Ask me no questions, I’ll tell you no lies:
  – OMG: http://www.omg.org/
  – Me: soley@omg.org
  – This presentation:
    http://www.omg.org/~soley/mdr.ppt