Agenda

- Introduction
- Evolution of Languages
- What is Abstraction
- How can Models help us?
- Model Driven Development
- Demo
- Conclusions
- Q & A
Speaker Introduction

- Ashok Nare
  - Technology Advisor, Enterprise Architect & Consultant
  - Currently working with various Startups
  - Over 15 years of experience in technology in a variety of roles (hands-on, technical, managerial and executive)

- Core Competencies
  - Business & Technology Alignment
  - Enterprise Solution Architecture
  - Model Driven Development
  - Service Oriented Architecture
  - Emerging Technologies

- LinkedIn Profile: [http://www.linkedin.com/in/ashoknare](http://www.linkedin.com/in/ashoknare)
- Blog: [http://www.ashoknare.com](http://www.ashoknare.com)
- Twitter: [@ashoknare](http://twitter.com/ashoknare)
Evolution of Languages

Source: http://www.eecs.ucf.edu/~leavens/ComS541Fall97/hw-pages/history/
Evolution of Languages

- Evolution of programming languages
  - Machine language to Assembly language to higher level languages such as C++, Java, C#, Ruby, etc.
  - More time was spent on understanding “how” to solve the problem in early languages (understand the language)

- Evolution of tools, frameworks and application servers
  - Abstraction and reuse of common services

- Each language and framework raised the level of Abstraction by hiding low level details

Assembly
- move.l #helloworld,-(A7)
- move #9,-(A7)
- trap #1
- addq.l #6,A7
- move #0,-(A7)
- trap #1
- helloworld:
- dc.b "Hello World!",$0d,$0a,0

Java
- public class HelloWorld {
  - public static void main(String[] args) {
    - System.out.println("Hello, World!");
  - }
  - }

Ruby
- puts "Hello, World!"
What is Abstraction?

- Abstraction is concentration on relevant aspects of the problem and ignoring those that are not important.
- Focus on solution to the problem by working with concepts and terms that are familiar to the problem space and ignoring the low level details.
- Abstraction is the key to building modern complex software with multiple moving parts.
- Model based development is the natural next step in the evolution of Abstraction.
How can Models help us?

- Models provide a simplified abstraction of the complexity in the target domain
  - Models provide an abstraction layer that focuses on the higher level concepts of the domain and decouples “what” from “how”
  - Models can be Visual or non-visual
  - Different models provide different views of the problem domain
- Used in Daily Life, Science & Engineering
  - Ex: Maps, Engineering (CAD/CAM), Architecture (Structural Modeling)
- Used in software engineering primarily for white boarding, communication and analysis & design
- What if Models are 100% semantically complete instead of merely being design artifacts!
Model Driven Development (MDD)

- **What is MDD?**
  - A software development approach that uses models to capture application logic during the development of end-to-end enterprise applications.
  - Forrester’s Definition: “An iterative approach to software development where models are the source of program execution with or without code generation.”

- **MDD Objectives**
  - Raise the level of abstraction for application development
  - Reduce development time and improve application quality
  - Reduce maintenance cost and Total Cost of Ownership of enterprise applications
Model Driven Development

- **How**
  - Use models to implement application logic
  - The domain model (in Abstract Design) is the implementation model
  - Use automation to generate executables from the implementation model in runtime or build time

- **Approaches**
  - OMG Model-Driven Architecture (MDA)
  - Executable Models
Model Driven Architecture (MDA)

**Pros**
- Reusable PIM models
- Methodical approach to software development
- More direct input from Application Analysts/Architects
- Marginal reduction in development time

**Cons**
- PIM models cannot be validated
- PSM are platform/architecture dependent, which makes up for the majority of the artifacts
- Transformation/code-generation is an integral part of the development process
- Generated code still require further development
Executable Models (xUML)
Executable Models (xUML)

Pros
- Abstract Design is 100% semantically complete and reusable
- Models can be immediately validated
- Significant reduction in development and maintenance time
- Code generation is optional and after-the-fact

Cons
- Supports a single architecture
- Depends on runtime environment
- No access to generated code
Executable Models (xUML)

The Model is the Executable
Advantages

- Captures application logic in platform independent UML models
- Simplifies application development by reducing the number of required skills in the underlying technologies, specifications and standards
- Provides immediate validation of business requirements
- Improves communication among stakeholders
- Protects business IP investments from evolving technologies
- Radically reduces the development, time, cost and effort of business applications and Web Services
- Increases application agility to better align with continuously changing business needs
MDD Vendors

- MDA
  - IBM’s Rational Software Architect
  - Interactive Objects (ArcStyler)
- Executable UML
  - Intelliun Corporation (The Virtual Enterprise)
  - Kennedy Carter (iUML)
  - CARE Technologies (OLIVANOVA)
  - Mentor Graphics (BridgePoint UML Suite)
  - E2E Technologies, Ltd. (E2E Bridge)
The Virtual Enterprise (VE)

- Built on NetBeans

- Support for multiple devices including Web Browser, PDA, Web Services

- All application logic is captured in UML notation (class and activity diagrams)

- Support for wide range of relational databases and auto-generation of database schema & OR-Map

- Application Logic Layer:
  - VE/Server
  - J2EE
  - Java VM

- Interface Layer:
  - Web Browser
  - PDA
  - Web Service

- Persistence/Resource Layer:
  - Database
  - Web Service
  - 3rd Party Application

- Build web applications using visual UML models
- Instant execution of UML models
- Dynamic generation of web interface and WSDL
- WYSIWYG web personalization
- Support for embedded Java code and Jar files
- Support for embedded SQL statements
- Unit/Remote testing framework
<table>
<thead>
<tr>
<th>VE/Designer</th>
<th>VE/Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Develop web applications using UML models</td>
<td>Runs on any J2EE web and/or application server</td>
</tr>
<tr>
<td>Instant execution of UML models and validation of application logic</td>
<td>Runs on a any Java supported platform including Unix, Linux, Windows and AS/400</td>
</tr>
<tr>
<td>Dynamic generation of the web interface and Web services</td>
<td>Supports wide range of relational databases including MS-SQL, Oracle, DB2/UDB, MySQL, Pervasive, and Sybase</td>
</tr>
<tr>
<td>WYSIWYG web personalization</td>
<td>Supports SOAP and WSDL in both client and server scenarios</td>
</tr>
<tr>
<td>Dynamic generation of object-relational database mapping</td>
<td>Supports REST in both client and server scenarios</td>
</tr>
<tr>
<td>Formula auto-completion</td>
<td>Supports JMS for messaging and events</td>
</tr>
<tr>
<td>Support embedding Java code and JAR files</td>
<td>Provides full localization</td>
</tr>
<tr>
<td>Support embedding hand-coded SQL statements/stored procedures</td>
<td></td>
</tr>
<tr>
<td>Unit/remote testing framework</td>
<td></td>
</tr>
</tbody>
</table>
Demonstration of Model Driven Development using Intellun’s Virtual Enterprise platform

Web-based billing module

- Submit Invoice
  - Capture line items with quantity, unit price and extended price

- List Invoices
  - Edit, delete specific invoices
Approach Summary

- Application logic is captured in platform independent UML models
- Models are immediately executable as they’re developed (no code generation, compilation, and deployment required)
- The development focus is always on the domain model, where interface and persistence is auto generated and can be later customized
- Code generation is optional and after-the-fact
- Code generation is done via templates that can be customized to control language, coding style, design patterns, and technology choices
Myths of MDD

- Modeling can only be used during the analysis and design phases
- Only objective of modeling is to generate code
- MDD is similar to CASE Tools
- Models are not sufficient to develop an enterprise application
- MDD tools are not mature and can only be used for small apps and prototypes
- MDD tools are expensive
State of MDD

- Many active vendors and products
- Proven to work – Several MDD applications in production!
- MDD and SOA
  - Model Driven Service Development
- MDD and BPM
  - Model Driven Business Process Management
- MDD continues to build momentum
  - Increased interest and involvement of industry leaders
- Emerging standards for Executable UML
  - OMG’s RFP for Concrete Syntax for a UML Action Language
OMG’s MDA Resources

Intelliun’ Virtual Enterprise
- Corporate: [http://www.intelliun.com](http://www.intelliun.com)
- Technical Overview: [http://www.intelliun.com/Products/TheVirtualEnterprise](http://www.intelliun.com/Products/TheVirtualEnterprise)