Outline

• Overview of Model Driven Architecture (MDA)
• Compare and Contrast MDA with traditional software engineering
• Perspectives
• An Example
• Model Checking, Animation, and Simulation
Overview of Model Driven Architecture (MDA)
Model Driven Architecture (MDA)

MDA is a 20+ year-old design-to-delivery approach that uses models* to define, design, construct, deploy, maintain and modify systems.

- 1,381 books about it on Amazon.com
- Non-proprietary set of standards
- EDU’s that cover MDA (US only): UM, George Mason UMBC, MIT, Harvard, more
- Multiple Vendors enable the standard with tools: No Magic, IBM, SPARX, SAP, others (soon: Microsoft)
- Uses open and standard modeling languages and notations.
- For more MDA information: [www.omg.org/mda/faq_mda.htm](http://www.omg.org/mda/faq_mda.htm)

* A Model can represent: business motivation, policy, regulation, influencers, regulations, events, agents, process, data, building, services, cloud, security...
Experience with MDA - Analysts Reports

• Forrester¹:
  – “57% of total sampled organization using MDD² …” 2008 “After an initial ramp up period, MDD results in applications with flexible architectures.”
  – “MDD allows us to halve or quarter development time compared with a traditional approach. (Fiducia, financial services firm)”³
  – “Modeling is still a crucial step when building an SOA …”⁴

• Gartner⁵:
  – “Modeling using a formal language leads to a robust and consistent models which facilitate user validation and lower most important defects…”
  – “Automatic code generation results in a short process giving the user the opportunity for an in-depth validation of the business rules while still in the modelling stage of the application.”

• Adopted by HL7, UN/CEFACT, SWIFT, ACORD, INCOSE, and others
  – Nearly 100% of commercial software done using some form of MDA⁶
  – Model Based Systems Engineering (MBSE). Using MDA for model-based Hardware/software⁷

2. MDD refers to Model Driven Development. MDD includes Model Driven Architecture and Development. VLH
3 Giuduce, Pp. 3
4. Giuduce Pp. 5, 17
5. Gartner Group Benchmark. CARE Technologies Deutschland GmbH. 2005
6. OMG straw poll of Independent Software Vendors
7. INCOSE 2007 Symposium
Compare and Contrast
Traditional Approach to Architecture and Engineering
Document-Centric, Linear, and “Separation of Concerns” by Conceptual Firewall
Model-Centric
One View: An Open, Continuous, Model based delivery process

Diagram showing the flow from Requirements, Business Model: Use Cases, Sequences and Tasks, Allocate Requirements, Feature, Rules, etc., to Business Object Model through Model Repository, with intermediate steps including Deployment, Generate interfaces and Implementation Connectors, Implementation independent description completed, Domain Classes Specified, Conceptual Classes, and Finish code (or connect COTS implementation); allocate to Environment.
Compare and Contrast this …
MDA Perspectives
An Enabler of Concurrent Processes “Threads”

- **Thread**: The step-by-step creation of a specific set of model elements necessary for system of service delivery
- **Cross-Thread Collaboration**: the model-enabled interactions and transformations that cross threads (e.g., how the specification of a message payload affects SERVICE specification)

### Full Lifecycle MDA-based Work “Threads”

- a. Requirements Elicitation, Analysis, Modeling, Measurements, and Allocation
- b. Business Process Modeling to Business Object Allocation
- c. Data Modeling integration into other views; Payloads, Entities, and OCL
- d. Behavioral Model detailing of Activities and Transition to Structural Modeling
- e. Transition of Behavioral Model element to Structural Elements
- f. Model Checking, Model Animation, and integrated Modeling and Simulation
- g. Delivery Spec/Code Generation and Buildmastering
- h. Reverse Engineering, refactoring, and Forward Engineering
MDA Perspectives
Cross-Thread Collaboration means *Model Elements from one thread are used in others*

- **Unit of Work (UOW):** The step-by-step creation of a specific set of model elements necessary for system of service delivery
- **Cross-Thread Collaboration:** the model-enabled interactions and transformations that cross threads (e.g., how the specification of a message payload affects SERVICE specification)

### Full Lifecycle MDA-based Work “Threads”

<table>
<thead>
<tr>
<th>Thread</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>Requirements Elicitation, Analysis, Modeling, Measurements, and Allocation (UoW: 1 Requirement)</td>
</tr>
<tr>
<td>c.</td>
<td>Data Modeling integration into other views; Payloads, Entities, and OCL (UoW: Payload; Entity; Subject Area to Bus. Obj.)</td>
</tr>
<tr>
<td>d.</td>
<td>Behavioral Model detailing of Activities and Transition to Structural Modeling (UoWs: Use Case; Sequence Diagram)</td>
</tr>
<tr>
<td>e.</td>
<td>Transition of Behavioral Model element to Structural Elements (UoW: Business Object; Class/service)</td>
</tr>
<tr>
<td>f.</td>
<td>Model Checking, Model Animation, and integrated Modeling and Simulation (UoW: Error, KPP, MOE, MOP)</td>
</tr>
<tr>
<td>g.</td>
<td>Delivery Spec/Code Generation and Buildmastering (UoW: Code; JUNITs)</td>
</tr>
<tr>
<td>h.</td>
<td>Reverse Engineering, refactoring, and Forward Engineering (UoW: Model Elements)</td>
</tr>
</tbody>
</table>
MDA Perspectives
A set of integrated modeling artifacts

Full Lifecycle MDA-based Work Threads
a. Requirements Elicitation, Analysis, Modeling, Measurements, and Allocation
b. Business Process Modeling to Business Object Allocation
c. Data Modeling integration into other views; Payloads, Entities, and OCL
d. Behavioral Model detailing of Activities and Transition to Structural Modeling
e. Transition of Behavioral Model element to Structural Elements
f. M&S and Model Simulation
g. Delivery Spec/Code Generation and Buildmastering
h. Reverse Engineering, refactoring, and Forward Engineering

Integrated MDA Model Viewpoints and Elements

1. Integrated Traceability and Transformation between Models
2. MDA Viewpoints are “Collaborations” of Models to represent the concerns of specific stakeholders
3. Model Profiles are “templates” for describing various aspects of a a Viewpoint
4. Model-based Views use Model Profiles as the basis for detailed models of the system
5. Association-based, not SDLC-phase constrained
6. Based upon an extensible MetaMetaModel that contains “templates” (Model Profiles) for each View
MDA Perspectives

An Example of Model Artifacts: Model Repository from a Real Project

Metadata Repository (MDR)

- Bi-Directional Traceability Map
  - Update Process: Requirements, User Interface, Subsystem, Use Case, Service Activity, Sequence, Class, Entity, Component diagrams, code, TDD...
  - Model versioning, security, model differencing
  - Industry standards (UML, SoaML, BPMN...)

MDR Portal

- Search/Discover models, model elements, transformations...
- Customizable dashboard, and Reports
- Model differencing reports
- Visual Traceability, impact analysis maps: Requirements to Design Models to Code changes

Code

- Java, XML Schemas, DDL, WSDL, Struts

- Generate executable code for Application, Service, Database
- Generate familiar SDLC deliverables (Technical Design Document)
An Example: Requirements Elicitation through Allocation
# Requirements Thread Description

<table>
<thead>
<tr>
<th>Description</th>
<th>Current</th>
<th>MDA</th>
<th>Impact and Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Requirements Elicitation</td>
<td>Current performed offline in Word documents and gathered through meeting minutes</td>
<td>Do in small sets into requirements repository and then released into MDA environment</td>
<td>Improves flow of requirements Low</td>
</tr>
<tr>
<td>2 Requirements Analysis</td>
<td>Done against a word document</td>
<td>Done as part of the model environment</td>
<td>Formalizes the requirements analysis process Medium risk</td>
</tr>
<tr>
<td>3 Requirements Modeling and Measurements</td>
<td>Not currently done</td>
<td>Done as a part of the modeling environment</td>
<td>Profound improvement to requirements to be allocated for delivery Moderate risk</td>
</tr>
<tr>
<td>4 Requirements Allocation</td>
<td>DECC currently uses Magic Draw</td>
<td>Minimal additional change</td>
<td>Explicit traceability and outcome binding Moderate risk</td>
</tr>
<tr>
<td>5 Specification Generation</td>
<td>Manual allocation of requirements to a single type of specification</td>
<td>System-controlled inclusion of requirements and requirement models to a type of specification</td>
<td>Explicit and tool controlled requirement to spec generation Low risk</td>
</tr>
</tbody>
</table>
### Requirements Elicitation

Example is the elicitation of a security requirement for Identity Management

<table>
<thead>
<tr>
<th>PP</th>
<th>Threats</th>
</tr>
</thead>
</table>
| 1  | User impersonation  
User exceeding assigned authorization |
| 2  | Undesired use of an object implementation  
Request/response repudiation  
Disclosure of “eyes-only” data |
| 3  | Unprotected security-unaware applications  
Unwanted revelation of Client machine existence |
| 4  | Object masquerade  
Client masquerade  
Object misuse of user authorizations  
Inter-operable Object Reference (IOR) tampering  
Disclosure of request contents  
Modification/destruction of request contents |
| 5  | Network eavesdropping  
Message tampering  
Inability to cross network boundaries (e.g., firewalls) |
| 6  | Unprotected security-unaware applications  
Too many object interfaces and implementation to manage individually |
| 7  | Unauthorized disclosure of specific information to client  
Request/response repudiation  
Protection of “eyes-only” data |

... 

As specified in authoritative sources  
capec.mitre.org/data/index.html#Definition or NIST equivalent

A ‘Protection Point’ specifies a place within the architecture where one or more threats occur and particular protection patterns apply.

Threats and Attacks are based upon authoritative sources.
Example of Requirements Model
Model the Requirements
Decompose into elementals, associate with others, and Type it
Example allocation Requirement to a Model element

- Example allocation Requirement
- User_Consumer
- Component_Consumer
- Consumer
- Get Service Discovery Service
- Component_Consumer
- User_Consumer
- Design Time Bind Service Interface
- Design Time Incorporate Component
- Incorporate Component
- Implement
- Get Service
- Discovery Service
- Query Federated Discovery Services
- Session-Based Service Bind
- UnBind Service
- Apply Service Instance resolution
- Criteria
- Apply implementing API resolution
- Criteria
- «include»
- «include»
- «include»
- «include»
- «communicate»
- «communicate»
- «communicate»
- «communicate»
- «communicate»
- «communicate»
- «communicate»
- «communicate»
Another Example:
Transition to Structural Architecture Elements

Roles: Basis for Classes

Characteristics: identity is established by Metadata Lookup

Basis for Services and Service Interfaces
Model Checking, Model Animation and Model Simulation
Model Checking

Apply rules to model elements and their associations. Check for correctness.
Model Animation

Establish Model Animation Criteria, timings & Wireframes

Animate models.
Set timing and load factors so models simulate SLAs. Perform M&S
There is more, but that’s enough

- JUNIT Generation
- Value based modeling
- Bi-directional transformations between UML and BPMN
- Ontology modeling (OWL compliant resultant models in UML/sysML
- Model Profiles for DoDAF, The Open Group Architecture, HL-7, etc.
- More…

But that’s enough…
THANK YOU