Incorporating MDA into the Development Process

Terry Merriman
M²VP
Today’s Changing Environment

・ The way companies do business is changing more rapidly than ever
・ The technology that supports them is changing even faster
・ Development efforts are often split across different teams on different continents
・ How can IT departments satisfy these fluid requirements when the ground beneath them is moving and the communication among participants is sketchy?!!
How Can We Keep Up?

- When faced with an enemy of superior forces…
  - Divide and Conquer
- In our case, apply…
  - Separation of Concerns
- Using Industry & De Facto Standards
  - Rational Unified Process
  - Reference Model for Open Distributed Processing
  + Model Driven Architecture
  = Separation of Concerns in Modeling
What Concerns?

่าย Separating Modeling Activities
  ◆ Modeling the Solution
  ◆ Modeling the Problem
  ◆ Modeling the Approach

่าย Separating Behavioral Types
  ◆ Functional Behavior
  ◆ Non-functional Behavior
  ◆ Semantic Behavior
  ◆ Idiomatic Behavior

่าย Integrating Modeling Activities
  ◆ Creating traceability links between elements of different models
RUP

“The Rational Unified Process® or RUP® product is a software engineering process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization.” [RUP]

It describes various modeling disciplines and the types of models employed by each

- Business Modeling
- Use Case Modeling
- Analysis & Design
- Implementation
RM-ODP

“RM-ODP is a standard for modeling object-based distributed processing architectures that separates concerns and simplifies the specification of heterogeneous open distributed processing systems” [Putnam 2001]

Transparencies
- Access, Failure, Location, Migration, Persistence, Relocation, Replication, Transaction

ODP Functions
- Management, Coordination, Repository, Security

Frameworks
- Interfacing, Binding, Interception, Behavioral Semantics

Viewpoints
- Viewpoints focus on the needs of a particular audience, abstracting out details that do not add clarity to the subject under inspection.
- RM-ODP prescribes five viewpoints as necessary and sufficient to describe the model of a system
  - Enterprise VP, Information VP, Computational VP, Engineering VP, Technology VP
**MDA**

- “The MDA defines an approach to IT system specification that separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform.” [MDA]
- Employs the Object Constraint Language to provide precision without code
- Is built upon a set of industry standards
- Provides separation of concerns through its Platform Independent and Platform Specific Models
Model Driven Architecture

**Architecture**

“The highest level concept of a system in its environment. The architecture of a software system (at a given point in time) is its organization or **structure of significant components** interacting through interfaces, those components being composed of successively smaller components and interfaces.” [RUP]

**Architectural Style**

“A description of **component types** and a **pattern** of their runtime control and/or data transfer. A style can be thought of as a set of constraints on an architecture – **constraints on the component types and their patterns of interaction** – and these constraints define a set or family of architectures that satisfy them.” [BASS 1998]

“Rules of engagement” described by a Metamodel
Putting It All Together
RUPRMODEPMDA

A modeling framework that...
- Helps to organize the models
- Enables traceability across the models
- Provides a documentation style that provides flowing, understandable specifications
- Breaks the effort into...
  - Approach
    - Metamodel
  - Problem Space
    - Business Model
    - System Model (aka Use Case Model)
  - Solution Space
    - Platform Independent Model
    - Platform Specific Models
Putting It All Together

Sports Club Management System

Modeling the Approach

<<Metamodel>>
Distributed Component Architectural Metamodel

Describes the modeling elements used within the models and the constraints placed on these elements.

Modeling the Problem Space

<<Business Model>>
Business Model
(from Use Case View)

Describes the functional behavior of the problem space and the community and business policies that constrain that behavior.

<<System Model>>
System Model
(from Use Case View)

Describes that portion of the functional behavior of the problem space that is to be addressed by the solution space.

Modeling the Solution Space

<<PIM>>
Platform Independent Model

Describes the functional, non-functional, and semantic behavior of the problem solution that satisfies the functional and supplementary requirements of the System model.

<<PSM>>
VBA/COM Platform Specific Model

Adds the idiomatic behavior for a VBA/COM platform to the functional and non-functional behavior described in the PIM.

<<PSM>>
WebSphere/J2EE Platform Specific Model

Adds the idiomatic behavior for a WebSphere/J2EE platform to the functional and non-functional behavior described in the PIM.
Modeling the Approach
Metamodel

Extending the Standard UML Stereotypes

Usage

Location and purpose
Scope and Visibility

Behavior
{state & composition control}

entity

presentation client

business service
application service
domain service
IT service
ER controller

presentation controller
application session

session
{transient}
state {long lived}
{persistent}

value object
session context

process

boundary

control

service
We use class operations in conceptual models to describe responsibilities (CRC Cards)
Modeling the Problem Space

EVP - Business Model

SCMS Business Model

External View

External Human Actors

- Club Member
- Supporter
- Team Member
- Game Official
- Affiliate

Individual

Internal View

Internal Business Workers

- Registrar
- Treasurer
- Team Assigner
- Program Manager
- Statistician
- Field Manager
- Scheduler

Activity Diagram: SCMS Business Model/SCMS Cross Domain Business Flow
Modeling the Problem Space
IVP Invariant Schema

Business Entities, Relationships, and Policies
Modeling the Problem Space
CVP - Business Workers

Business workers perform the functional behavior of the system, enforcing & maintaining policies while performing their responsibilities.
Business workers perform the functional behavior of the system, enforcing & maintaining policies while performing their responsibilities.

Modeling the Problem Space
CVP - Business Workers

Assign Players & Coaches to Leagues, Divisions, & Teams

- Select a League
  - Assign Players to League
    - Select a Division within the League
      - Assign Players to Division
        - Rate Players
          - Assign Players to Teams
            - Assign Players to Teams Per Requests
              - No more Divisions

- More Leagues
  - No more Leagues
  - More Divisions
    - No more Divisions

- Club's Player Assignment Policy
- League's Player Assignment Policy
- Division's Player Assignment Policy
- Team's Player Assignment Policy
Scoping the Problem Space

Mapping Business Workers’ Responsibilities to System Use Cases

Labeling the relationship with the responsibility enables traceability in SoDA

System use cases determine what system intervention is needed to help business workers fulfill their responsibilities. They may map back to business worker activities

Policies control how use cases operate

Team Assigner
- Assign Players to Leagues, Divisions, and Teams
- Assign Coaches to Teams
- Maintain Team Rosters
- Order Uniforms
- Distribute Uniforms
- Distribute Team IDs

Post Team Schedule SUC

Maintain Team Rosters

Maintain Team Rosters

<<include>>

Drop Team SUC

Remove Team Member from Team SUC

<<communicate>>

Make Team Requests BUC

(from Program Development Domain BUC Model)

<<policy>>
League's Player Assignment Policy

<<policy>>
Division's Player Assignment Policy

governed by

governed by

Assign Players to Leagues, Divisions, and Teams

Assign Players to League SUC

Assign Players to Division SUC

Assign Players to Team SUC

Assign Players by Requests SUC

Assign Coach to Team SUC

<<extend>>

<<extend>>

Club's Coach Assignment Policy

Policies control how use cases operate

System use cases determine what system intervention is needed to help business workers fulfill their responsibilities. They may map back to business worker activities

Scoping the Problem Space

Mapping Business Workers’ Responsibilities to System Use Cases
Modeling the Solution Space
Platform Independent Model

SCMS Program Development Subsystem

<<Application Sessions>>
Program Development Application Sessions

<<Domain Services>>
Program Development Domain Services

<<Object Model>>
Program Development Object Model
Modeling the Solution Space
PIM IVP Invariant Schema

Extend/standardize behavior with functional patterns (Party Management)

Add non-functional behavior, often through the use of design patterns or frameworks (Object Reflection Pattern)

Override generic behavior of frameworks by adding domain constraints
Modeling the Solution Space
PIM IVP Static Schema

Boys Recreational League : League

Under 10 : Division

Green : Team

head : Coach

Juan Vinces : Club Member

: Game Official

Boys' Competitive League : League

Under16 : Division

Blue : Team

1st keeper : Player

Cougars : Team

Shows a representative or exceptional sample of the system state.
(Metalevel M0)
Modeling the Solution Space
PIM IVP Dynamic Schema

Shows the forces that cause change in state.

Register → Unassigned
Unassigned → Assigned to League
Assigned to League → Player Assignment Service::assignPlayerToLeague
Player Assignment Service::assignPlayerToLeague → Assigned to Division
Assigned to Division → Player Assignment Service::assignPlayerToDivision
Player Assignment Service::assignPlayerToDivision → Assigned to Team
Assigned to Team → [Active, Suspended]
Active suspend Suspended activate
Modeling the Solution Space
PIM CVP - Application Sessions & Domain Services

Application sessions interact with users through the presentation tier to gather information and invoke domain or business services.

Domain Services control the resources of the domain and perform shared functionality while enforcing the policies.
Modeling the Solution Space
Declarative Definition of an Operation

Player Assignment Service: assignPlayersToDivision( pClub : Club, pLeague : League, pDivision : Division) : void

Pre-conditions

-- The Player must belong to the Division's League.
pre: pPlayer.league \exists(pPlayer.league = pDivision.league)

-- The Player must not already belong to the Division.
pre: NOT pDivision.player \in(pPlayer)

-- The Player must not already belong to the maximum number of Divisions permitted by the Division's League
pre: pPlayer.division[pDivision.league] \size() < pDivision.league.maxDivisionsPerPlayer

-- The Player's age must be within the Division's age range.
pre: pDivision.minPlayerAge <= pPlayer.clubMember.age AND pPlayer.clubMember.age <= pDivision.maxPlayerAge

-- If the Division has a Region requirement, the Player's Region must fall within the Division's range of Regions.
pre: pDivision.region \size() > 0 IMPLIES pDivision.region \exists(region = pPlayer.region)

-- The Division must not have the maximum number of Players already assigned to it.
pre: pDivision.player \size() < pDivision.maxPlayersPerTeam * pDivision.maxNumberOfTeams

-- The number of Divisions within the League that the Player is already on must be less than the League's maxDivisionsPerPlayer.
pre: pPlayer.division[pDivision.league] \size() < pDivision.league.maxDivisionsPerPlayer

-- If the Division is not Coed, then the Player's gender must match the Division's.
pre: pDivision.gender ! = 'C' implies pPlayer.clubMember.gender = pDivision.gender

-- If the Division has ability requirements, then the Player's rated ability must be within them.
pre: pDivision.maxPlayerAbility > 0 IMPLIES
    pDivision.minPlayerAbility <= pPlayer.rating AND pPlayer.rating <= pDivision.maxPlayerAbility

Post-conditions

-- The division includes the player.
post: pDivision.player \in(pPlayer)

-- The player is in the division.
post: pPlayer.division \in(pDivision)

English and OCL provide declarative descriptions of operation processing.
References & Bibliography

[*] [RUP] Rational Unified Process, Version 2002.05.00, Rational Corporation
[*] Rebecca Wirfs-Brock, Brian Wilkerson, Lauren Wiener, Rebecca Brock, Designing Object-Oriented Software, Prentice Hall PTR (1990)

Terry Merriman
M²VP
610-429-2999
tmerriman@m2vp.com