Incorporating MDA into the Development Process

Terry Merriman

M^2VP

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?

- Separating Modeling Activities
  - Modeling the Solution
  - Modeling the Problem
  - Modeling the Approach
- Separating Behavioral Types
  - Functional Behavior
  - Non-functional Behavior
  - Semantic Behavior
  - Idiomatic Behavior
- 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*

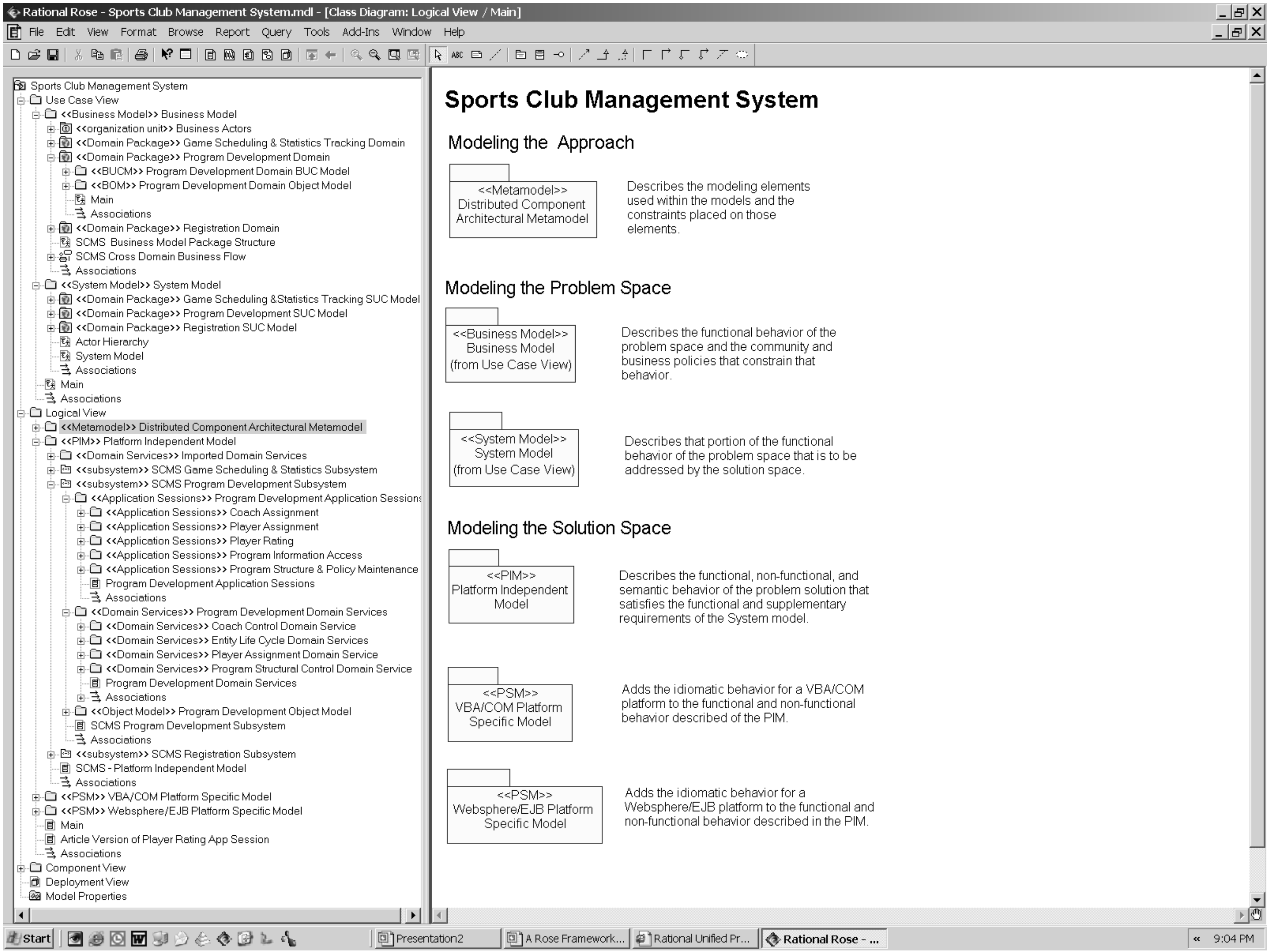
- Architecture
  - “The highest level concept of a system in its environment. The architecture of a software system (at a given point in time) is its organization or **structure of significant components** interacting through interfaces, those components being composed of successively smaller components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

RUPRMODPMDA

- 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



Rational Rose - Sports Club Management System.mdl - [Class Diagram: Logical View / Main]

- Sports Club Management System
  - Use Case View
    - <<Business Model>> Business Model
      - <<organization unit>> Business Actors
      - <<Domain Package>> Game Scheduling & Statistics Tracking Domain
      - <<Domain Package>> Program Development Domain
        - <<BUCM>> Program Development Domain BUC Model
        - <<BOM>> Program Development Domain Object Model
        - Main
        - Associations
      - <<Domain Package>> Registration Domain
      - SCMS Business Model Package Structure
      - SCMS Cross Domain Business Flow
      - Associations
    - <<System Model>> System Model
      - <<Domain Package>> Game Scheduling &Statistics Tracking SUC Model
      - <<Domain Package>> Program Development SUC Model
      - <<Domain Package>> Registration SUC Model
      - Actor Hierarchy
      - System Model
      - Associations
    - Main
    - Associations
  - Logical View
    - <<Metamodel>> Distributed Component Architectural Metamodel
    - <<PIM>> Platform Independent Model
      - <<Domain Services>> Imported Domain Services
      - <<subsystem>> SCMS Game Scheduling & Statistics Subsystem
      - <<subsystem>> SCMS Program Development Subsystem
        - <<Application Sessions>> Program Development Application Sessions
          - <<Application Sessions>> Coach Assignment
          - <<Application Sessions>> Player Assignment
          - <<Application Sessions>> Player Rating
          - <<Application Sessions>> Program Information Access
          - <<Application Sessions>> Program Structure & Policy Maintenance
          - Program Development Application Sessions
          - Associations
        - <<Domain Services>> Program Development Domain Services
          - <<Domain Services>> Coach Control Domain Service
          - <<Domain Services>> Entity Life Cycle Domain Services
          - <<Domain Services>> Player Assignment Domain Service
          - <<Domain Services>> Program Structural Control Domain Service
          - Program Development Domain Services
          - Associations
        - <<Object Model>> Program Development Object Model
        - SCMS Program Development Subsystem
        - Associations
      - <<subsystem>> SCMS Registration Subsystem
      - SCMS - Platform Independent Model
      - Associations
    - <<PSM>> VBA/COM Platform Specific Model
    - <<PSM>> Websphere/EJB Platform Specific Model
    - Main
    - Article Version of Player Rating App Session
    - Associations
  - Component View
  - Deployment View
  - Model Properties

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 those 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

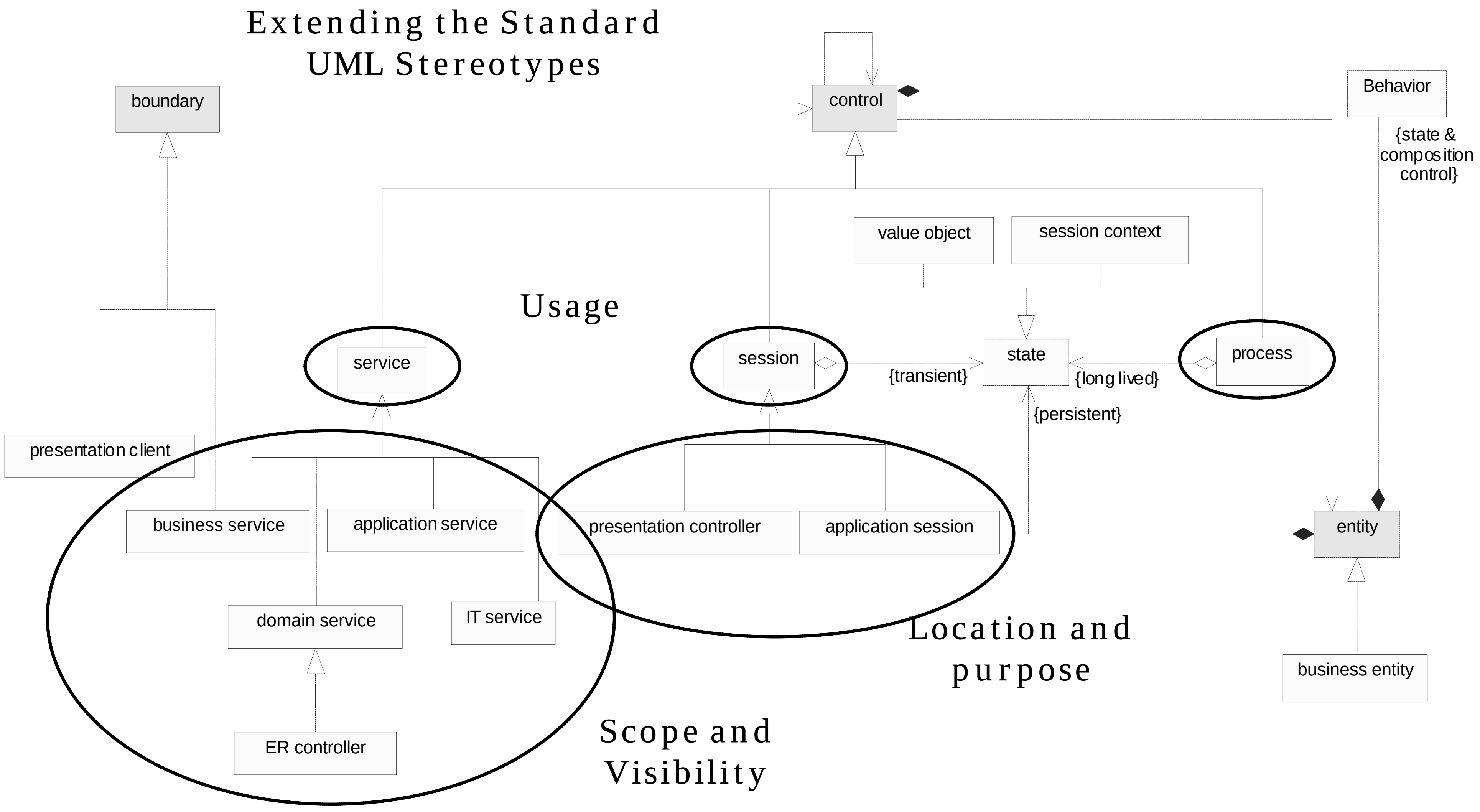
Adds the idiomatic behavior for a VBA/COM platform to the functional and non-functional behavior described of the PIM.

<<PSM>>
Websphere/EJB Platform
Specific Model

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

Modeling the Approach
Metamodel

(Metalevel M2)



Extending the Standard UML Stereotypes

Usage

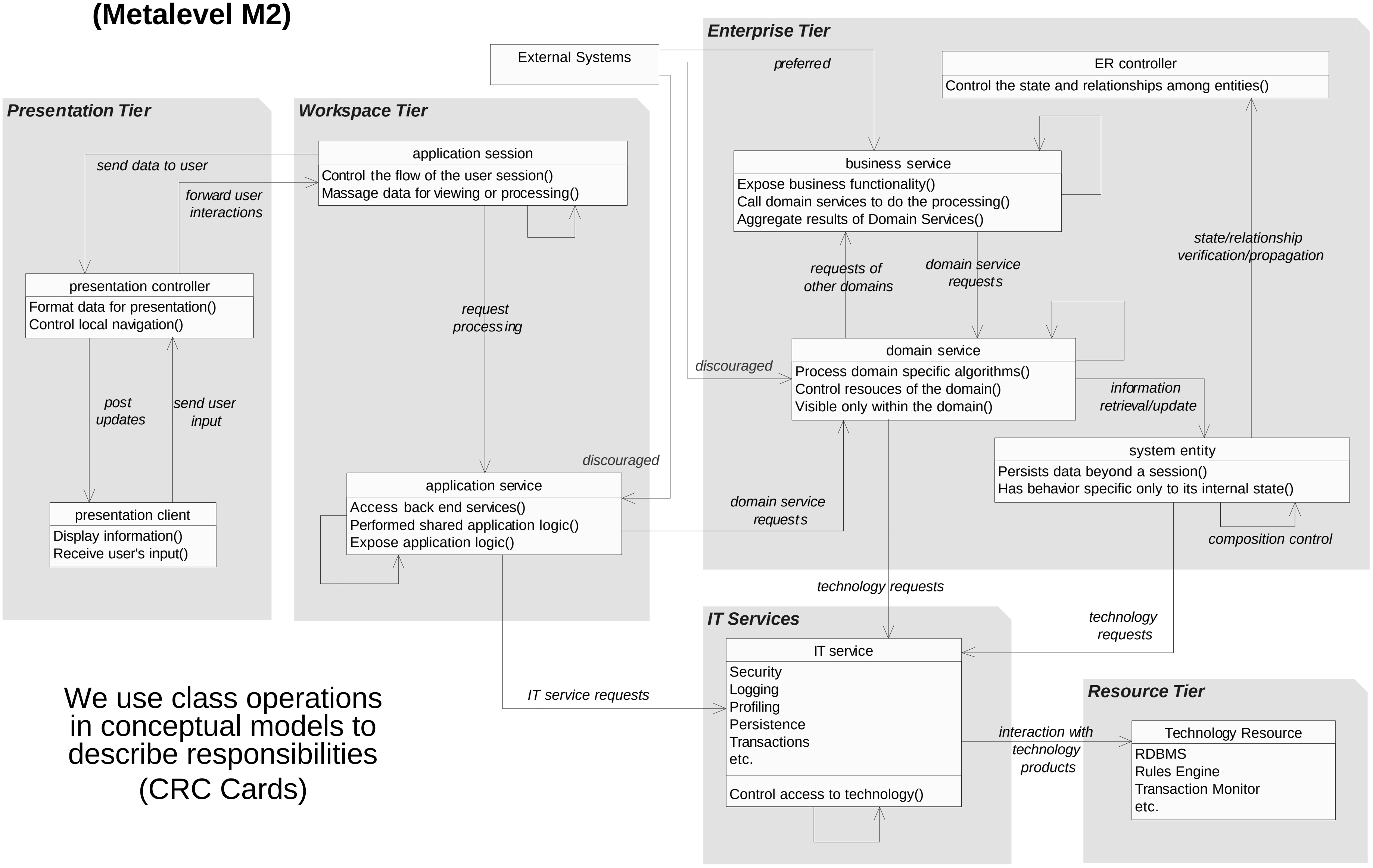
Location and purpose

Scope and Visibility

Modeling the Approach

“Rules of Engagement”

(Metalevel M2)



Presentation Tier

presentation controller
- Format data for presentation()
- Control local navigation()

presentation client
- Display information()
- Receive user's input()

Workspace Tier

application session
- Control the flow of the user session()
- Massage data for viewing or processing()

application service
- Access back end services()
- Performed shared application logic()
- Expose application logic()

External Systems

Enterprise Tier

ER controller
- Control the state and relationships among entities()

business service
- Expose business functionality()
- Call domain services to do the processing()
- Aggregate results of Domain Services()

domain service
- Process domain specific algorithms()
- Control resouces of the domain()
- Visible only within the domain()

system entity
- Persists data beyond a session()
- Has behavior specific only to its internal state()

IT Services

IT service
- Security
- Logging
- Profiling
- Persistence
- Transactions
- etc.
- Control access to technology()

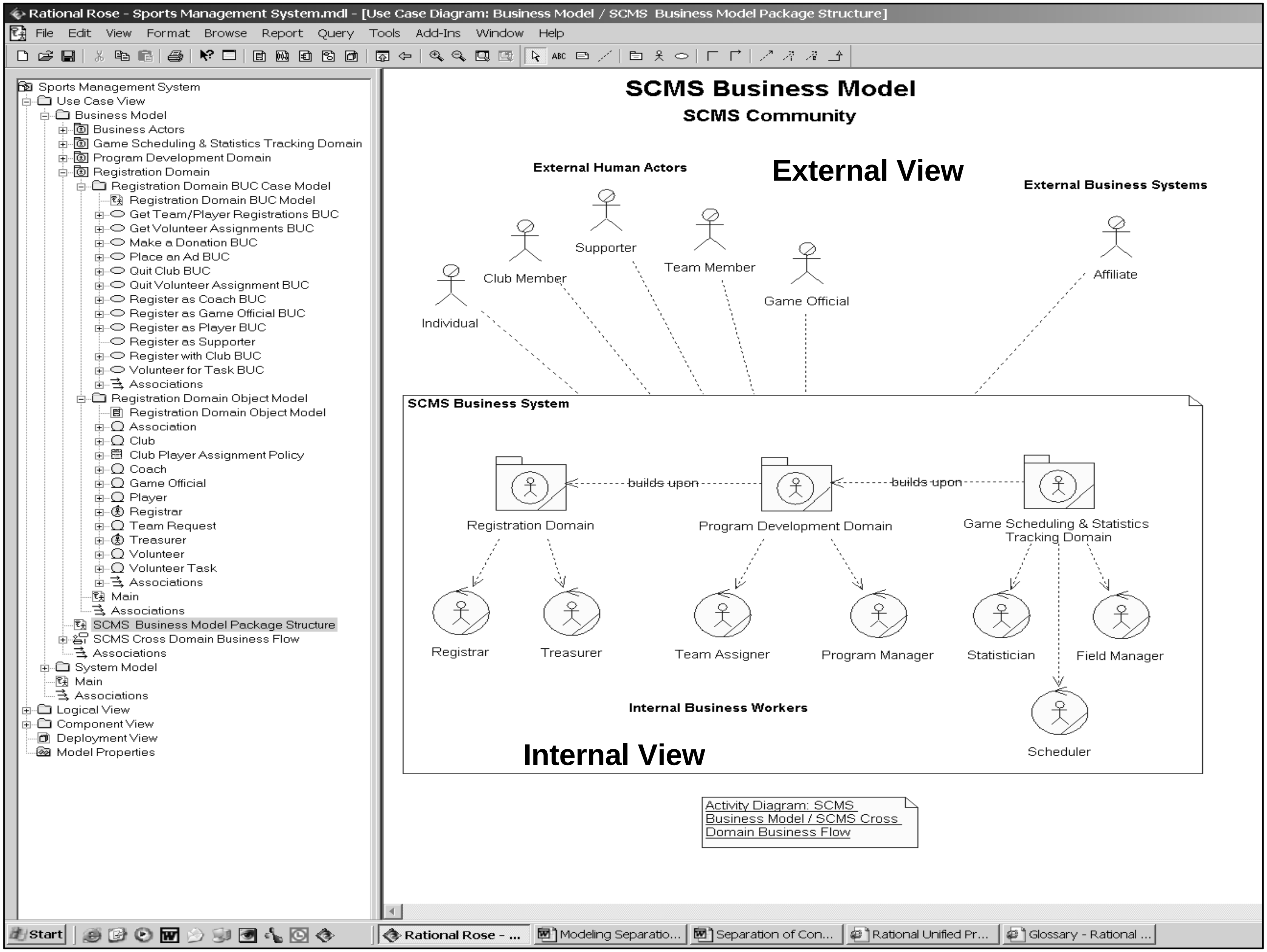
Resource Tier

Technology Resource
- RDBMS
- Rules Engine
- Transaction Monitor
- etc.

We use class operations in conceptual models to describe responsibilities

(CRC Cards)

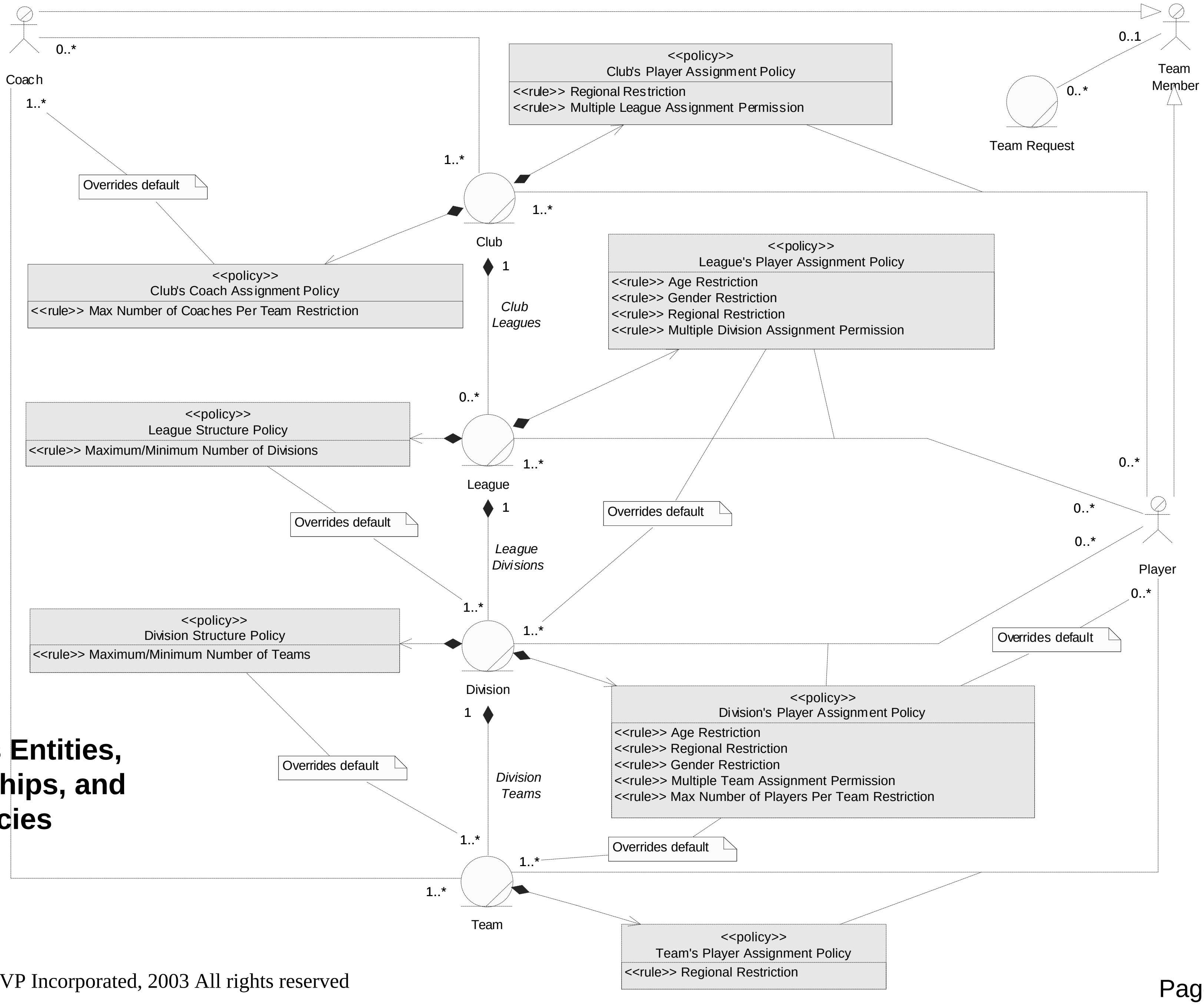
Modeling the Problem Space
EVP - Business Model


Rational Rose - Sports Management System.mdl - [Use Case Diagram: Business Model / SCMS Business Model Package Structure]
File Edit View Format Browse Report Query Tools Add-Ins Window Help
ABC
Start
Rational Rose - ...
Modeling Separatio...
Separation of Con...
Rational Unified Pr...
Glossary - Rational ...

- Sports Management System
  - Use Case View
    - Business Model
      - Business Actors
      - Game Scheduling & Statistics Tracking Domain
      - Program Development Domain
      - Registration Domain
        - Registration Domain BUC Case Model
          - Registration Domain BUC Model
          - Get Team/Player Registrations BUC
          - Get Volunteer Assignments BUC
          - Make a Donation BUC
          - Place an Ad BUC
          - Quit Club BUC
          - Quit Volunteer Assignment BUC
          - Register as Coach BUC
          - Register as Game Official BUC
          - Register as Player BUC
          - Register as Supporter
          - Register with Club BUC
          - Volunteer for Task BUC
          - Associations
        - Registration Domain Object Model
          - Registration Domain Object Model
          - Association
          - Club
          - Club Player Assignment Policy
          - Coach
          - Game Official
          - Player
          - Registrar
          - Team Request
          - Treasurer
          - Volunteer
          - Volunteer Task
          - Associations
        - Main
        - Associations
      - SCMS Business Model Package Structure
      - SCMS Cross Domain Business Flow
      - Associations
    - System Model
    - Main
    - Associations
  - Logical View
  - Component View
  - Deployment View
  - Model Properties

Modeling the Problem Space

IVP Invariant Schema



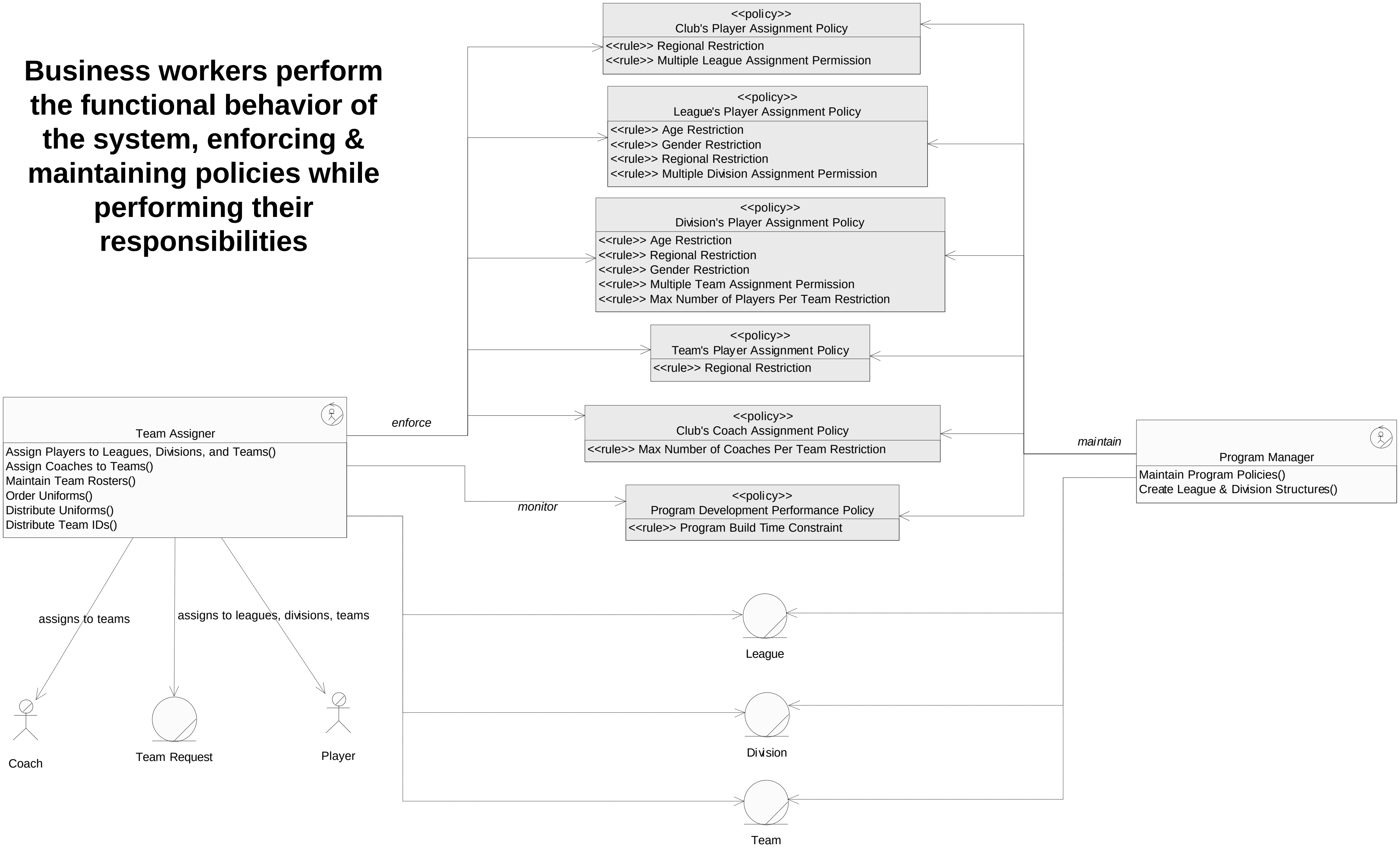
Coach
0..*
1..*
Team Member
0..1
Team Request
0..*
<<policy>>
Club's Player Assignment Policy
<<rule>> Regional Restriction
<<rule>> Multiple League Assignment Permission
Overrides default
1..*
Club
1..*
<<policy>>
Club's Coach Assignment Policy
<<rule>> Max Number of Coaches Per Team Restriction
1
Club Leagues
<<policy>>
League's Player Assignment Policy
<<rule>> Age Restriction
<<rule>> Gender Restriction
<<rule>> Regional Restriction
<<rule>> Multiple Division Assignment Permission
0..*
<<policy>>
League Structure Policy
<<rule>> Maximum/Minimum Number of Divisions
League
1..*
0..*
1
Overrides default
Overrides default
0..*
0..*
Player
League Divisions
1..*
<<policy>>
Division Structure Policy
<<rule>> Maximum/Minimum Number of Teams
1..*
0..*
Overrides default
Division
<<policy>>
Division's Player Assignment Policy
<<rule>> Age Restriction
<<rule>> Regional Restriction
<<rule>> Gender Restriction
<<rule>> Multiple Team Assignment Permission
<<rule>> Max Number of Players Per Team Restriction
1
Division Teams
Overrides default
1..*
Overrides default
1..*
1..*
Team
<<policy>>
Team's Player Assignment Policy
<<rule>> Regional Restriction

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



<<policy>>
Club's Player Assignment Policy
<<rule>> Regional Restriction
<<rule>> Multiple League Assignment Permission

<<policy>>
League's Player Assignment Policy
<<rule>> Age Restriction
<<rule>> Gender Restriction
<<rule>> Regional Restriction
<<rule>> Multiple Division Assignment Permission

<<policy>>
Division's Player Assignment Policy
<<rule>> Age Restriction
<<rule>> Regional Restriction
<<rule>> Gender Restriction
<<rule>> Multiple Team Assignment Permission
<<rule>> Max Number of Players Per Team Restriction

<<policy>>
Team's Player Assignment Policy
<<rule>> Regional Restriction

<<policy>>
Club's Coach Assignment Policy
<<rule>> Max Number of Coaches Per Team Restriction

<<policy>>
Program Development Performance Policy
<<rule>> Program Build Time Constraint

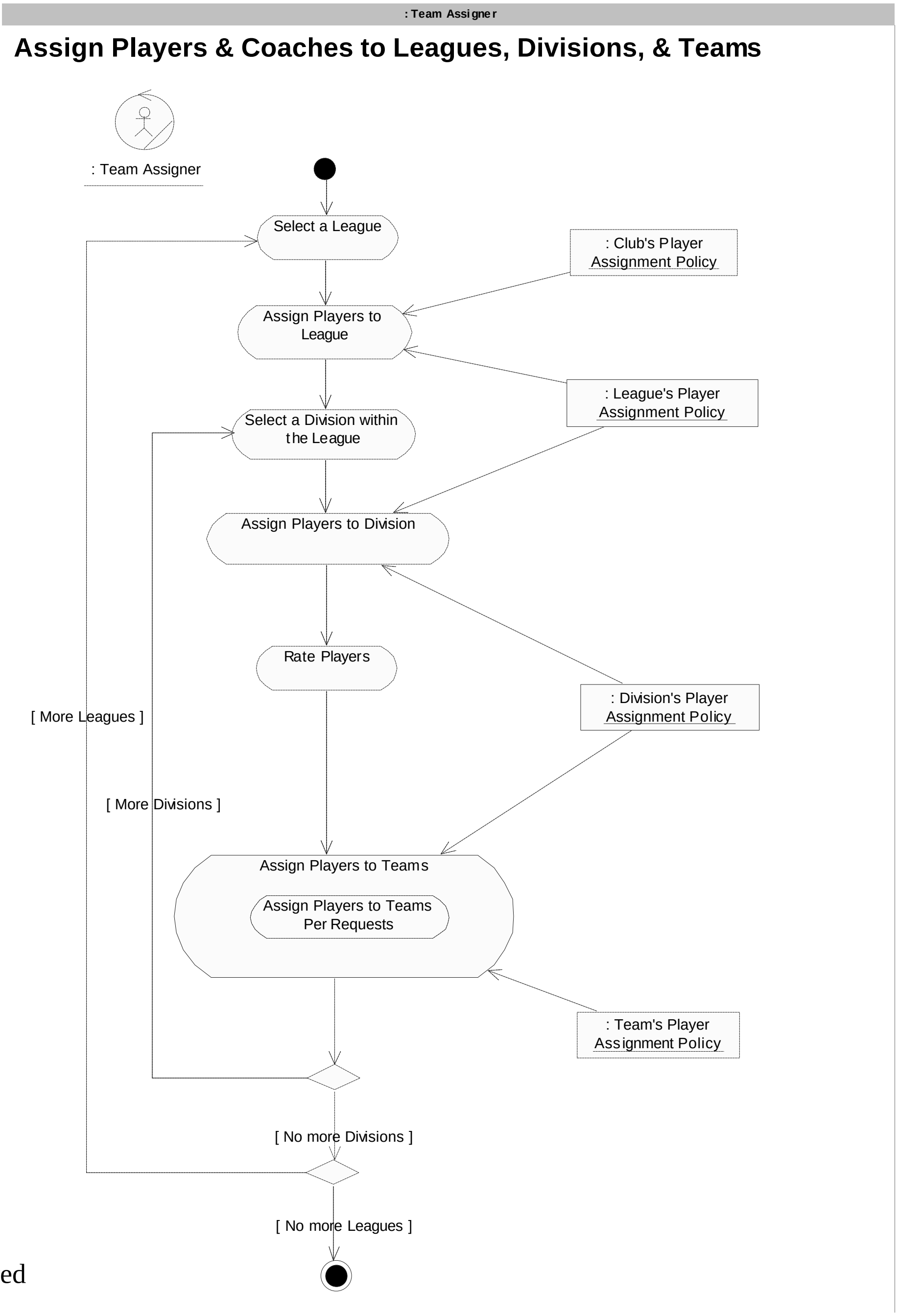
Team Assigner
Assign Players to Leagues, Divisions, and Teams()
Assign Coaches to Teams()
Maintain Team Rosters()
Order Uniforms()
Distribute Uniforms()
Distribute Team IDs()

Program Manager
Maintain Program Policies()
Create League & Division Structures()

Modeling the Problem Space

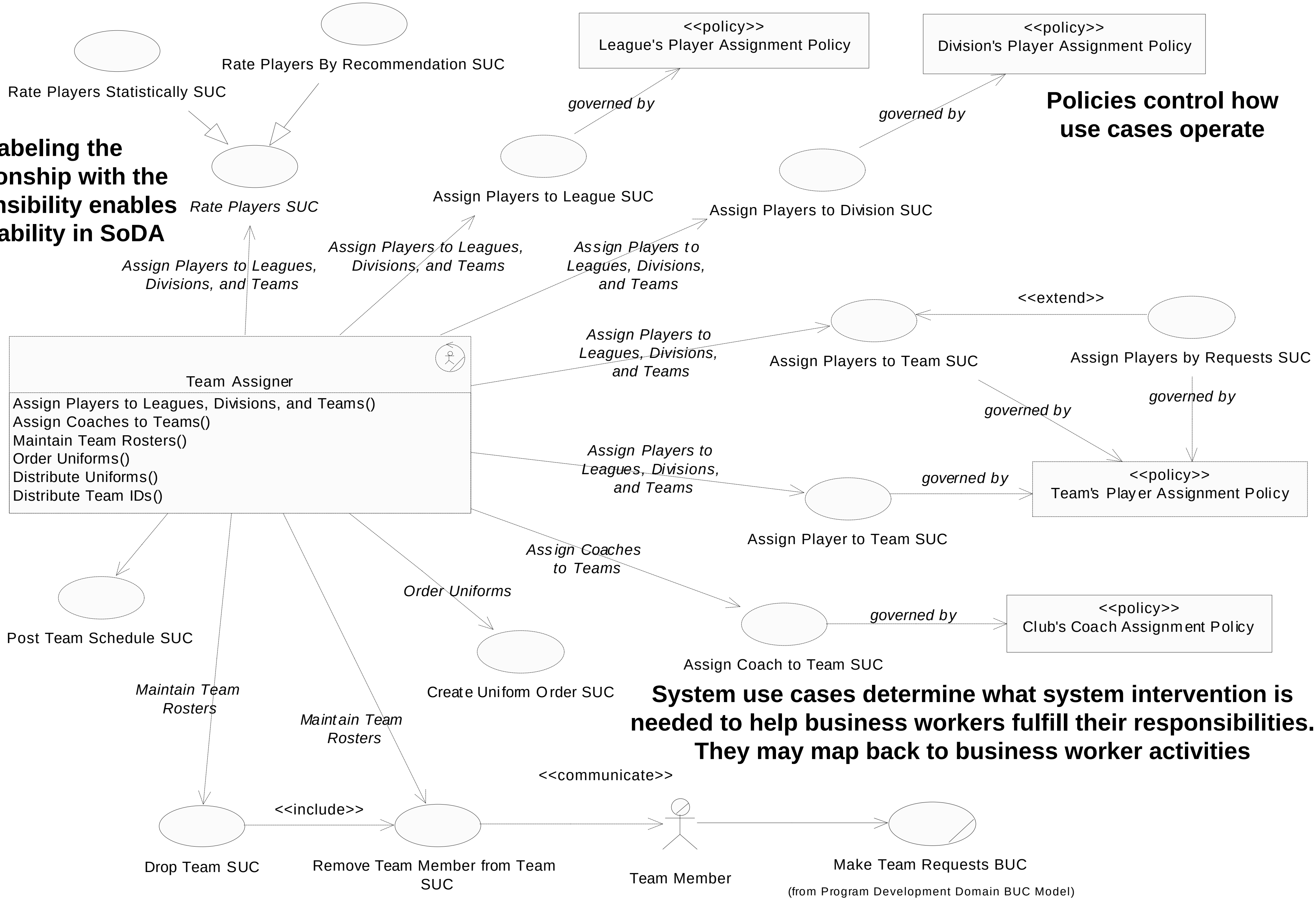
CVP - Business Workers

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


: Team Assigner
Assign Players & Coaches to Leagues, Divisions, & Teams
: Team Assigner
Select a League
: Club's Player Assignment Policy
Assign Players to League
: League's Player Assignment Policy
Select a Division within the League
Assign Players to Division
Rate Players
: Division's Player Assignment Policy
[More Leagues]
[More Divisions]
Assign Players to Teams
Assign Players to Teams Per Requests
: Team's Player Assignment Policy
[No more Divisions]
[No more Leagues]

Scoping the Problem Space

Mapping Business Workers’ Responsibilities to System Use Cases


Rate Players By Recommendation SUC
Rate Players Statistically SUC
Rate Players SUC
<<policy>>
League's Player Assignment Policy
<<policy>>
Division's Player Assignment Policy
governed by
Assign Players to League SUC
governed by
Assign Players to Division SUC
Assign Players to Leagues, Divisions, and Teams
Assign Players to Leagues, Divisions, and Teams
Assign Players to Leagues, Divisions, and Teams
<<extend>>
Assign Players to Leagues, Divisions, and Teams
Assign Players to Team SUC
Assign Players by Requests SUC
Team Assigner
Assign Players to Leagues, Divisions, and Teams()
Assign Coaches to Teams()
Maintain Team Rosters()
Order Uniforms()
Distribute Uniforms()
Distribute Team IDs()
governed by
governed by
Assign Players to Leagues, Divisions, and Teams
governed by
<<policy>>
Team's Player Assignment Policy
Assign Player to Team SUC
Assign Coaches to Teams
Order Uniforms
governed by
<<policy>>
Club's Coach Assignment Policy
Post Team Schedule SUC
Create Uniform Order SUC
Assign Coach to Team SUC
Maintain Team Rosters
Maintain Team Rosters
<<communicate>>
<<include>>
Drop Team SUC
Remove Team Member from Team SUC
Team Member
Make Team Requests BUC
(from Program Development Domain BUC Model)

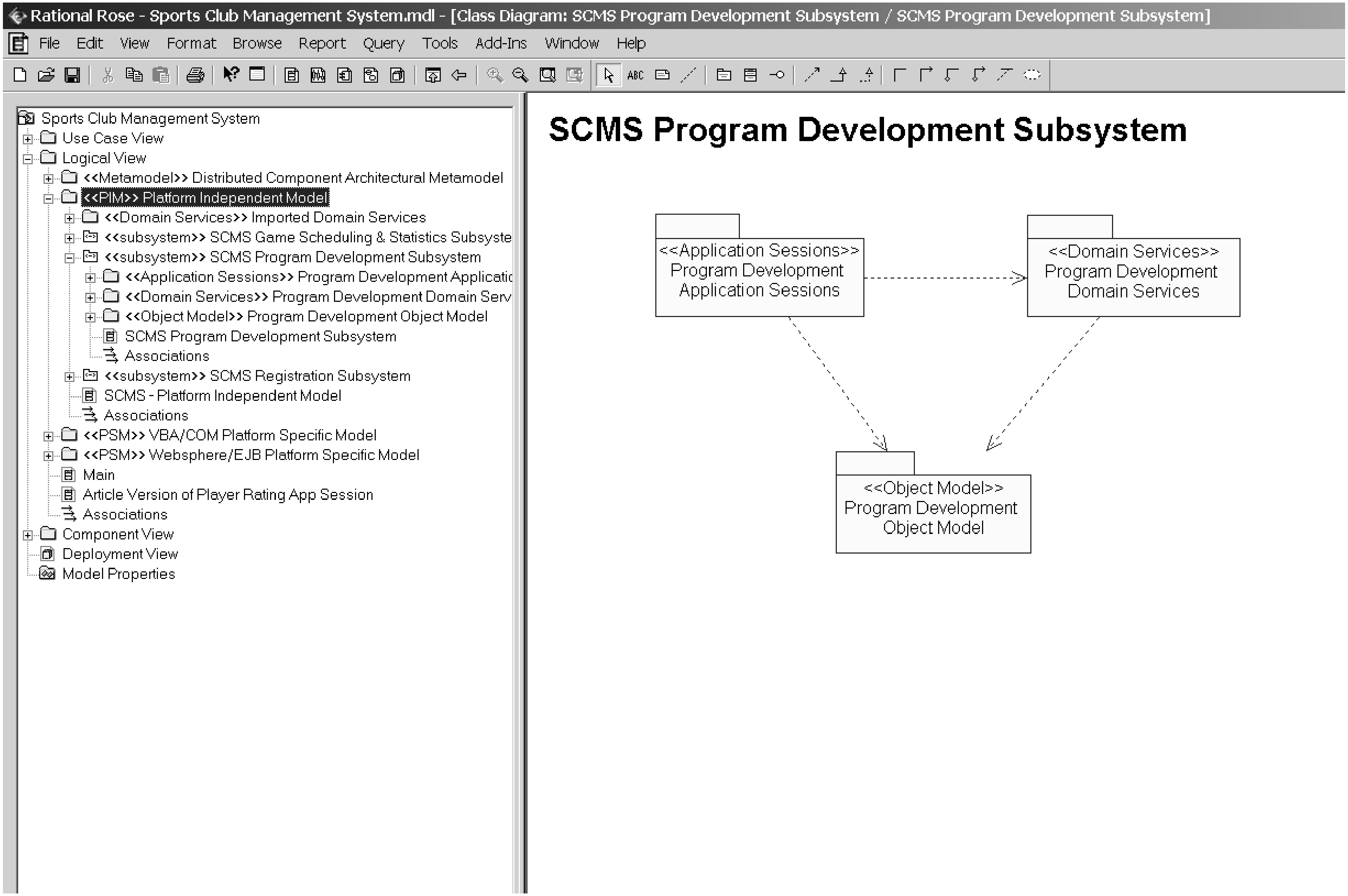
Labeling the relationship with the responsibility enables traceability in SoDA

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

Modeling the Solution Space

Platform Independent Model



Rational Rose - Sports Club Management System.mdl - [Class Diagram: SCMS Program Development Subsystem / SCMS Program Development Subsystem]

- Sports Club Management System
  - Use Case View
  - Logical View
    - <<Metamodel>> Distributed Component Architectural Metamodel
    - <<PIM>> Platform Independent Model
      - <<Domain Services>> Imported Domain Services
      - <<subsystem>> SCMS Game Scheduling & Statistics Subsyste
      - <<subsystem>> SCMS Program Development Subsystem
        - <<Application Sessions>> Program Development Applicatic
        - <<Domain Services>> Program Development Domain Serv
        - <<Object Model>> Program Development Object Model
        - SCMS Program Development Subsystem
        - Associations
      - <<subsystem>> SCMS Registration Subsystem
      - SCMS - Platform Independent Model
      - Associations
    - <<PSM>> VBA/COM Platform Specific Model
    - <<PSM>> Websphere/EJB Platform Specific Model
    - Main
    - Article Version of Player Rating App Session
    - Associations
  - Component View
  - Deployment View
  - Model Properties

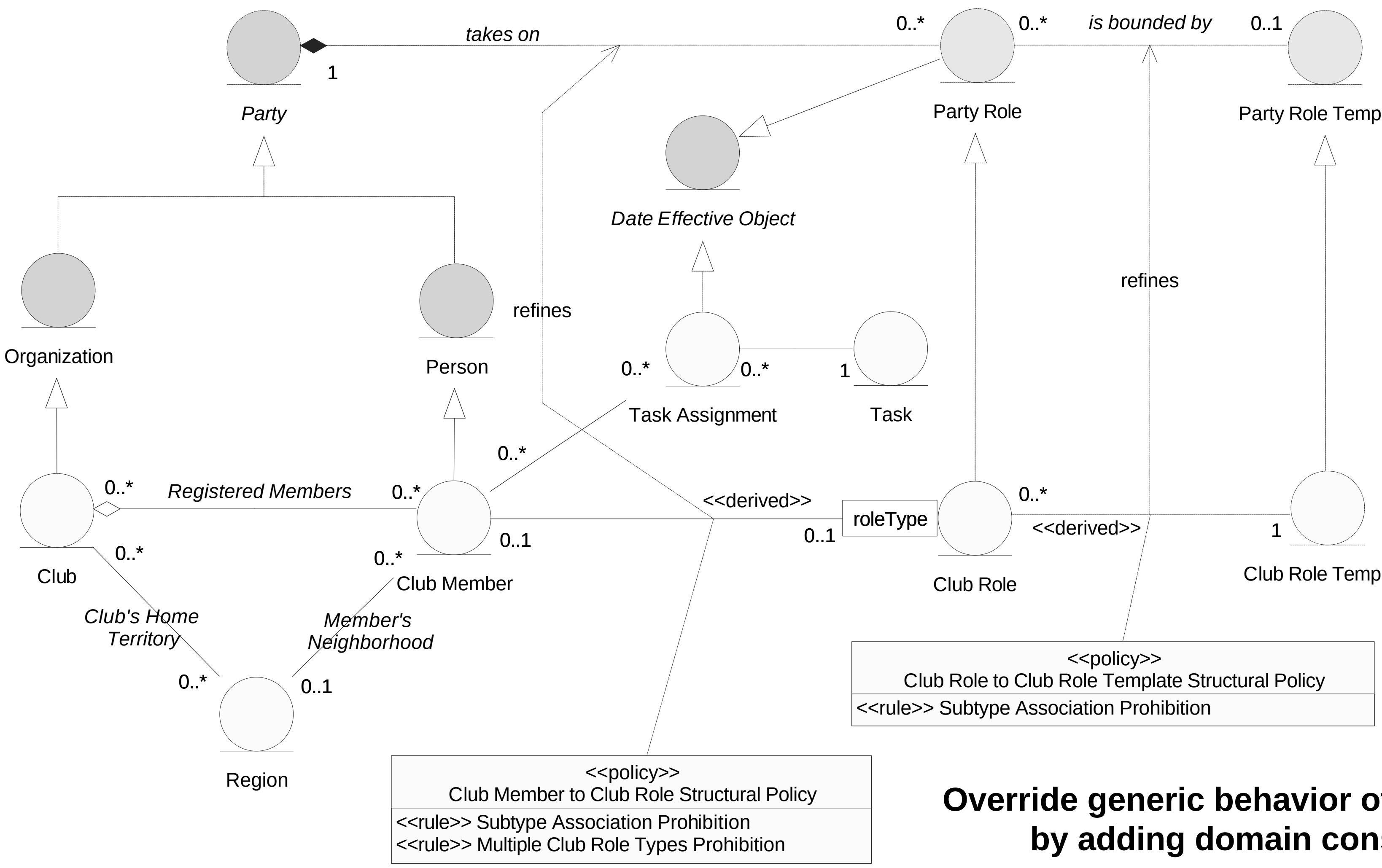
SCMS Program Development Subsystem

<<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)

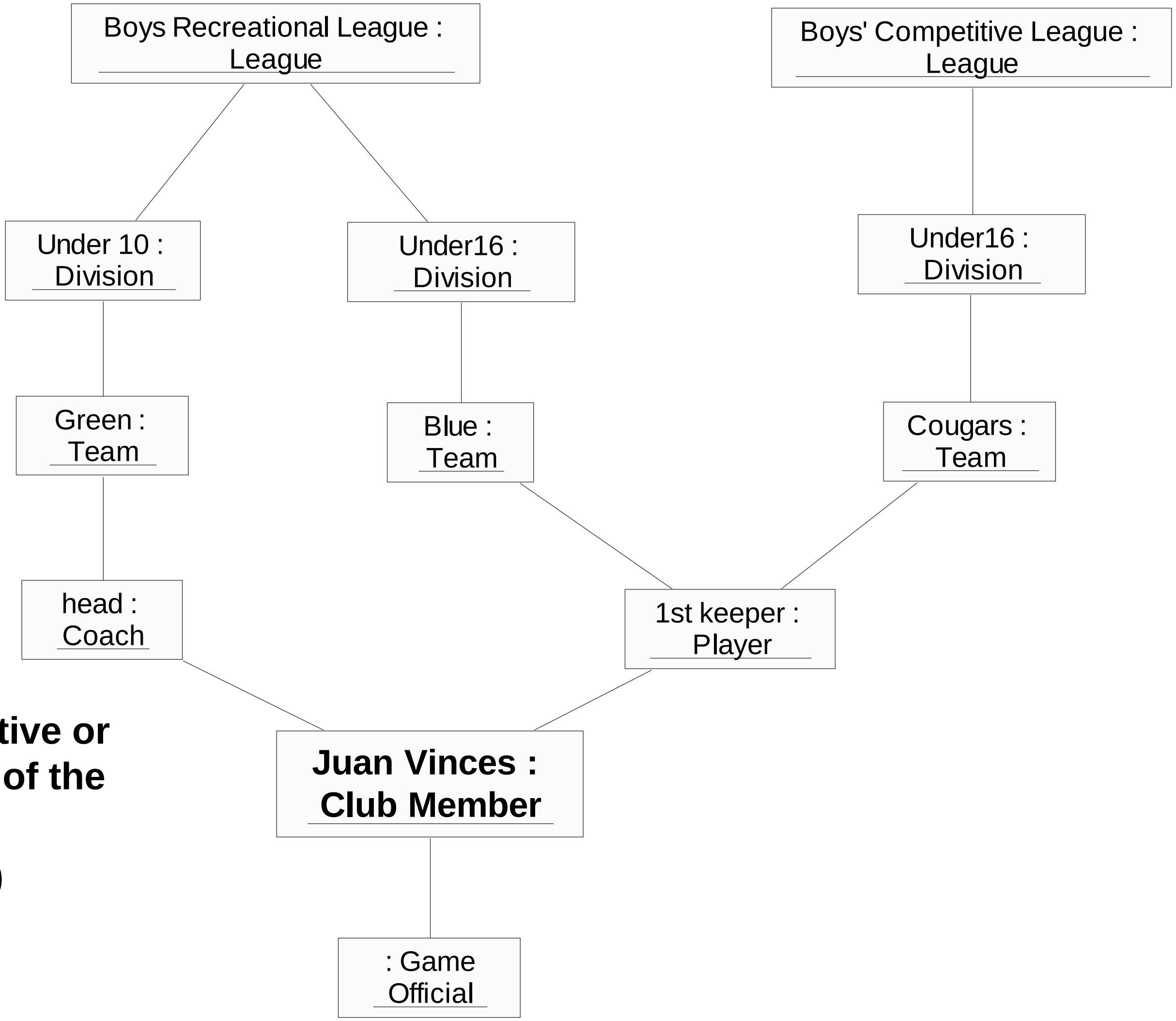


<<policy>>
Club Role to Club Role Template Structural Policy
<<rule>> Subtype Association Prohibition

<<policy>>
Club Member to Club Role Structural Policy
<<rule>> Subtype Association Prohibition
<<rule>> Multiple Club Role Types Prohibition

Override generic behavior of frameworks by adding domain constraints

Modeling the Solution Space
PIM IVP Static Schema



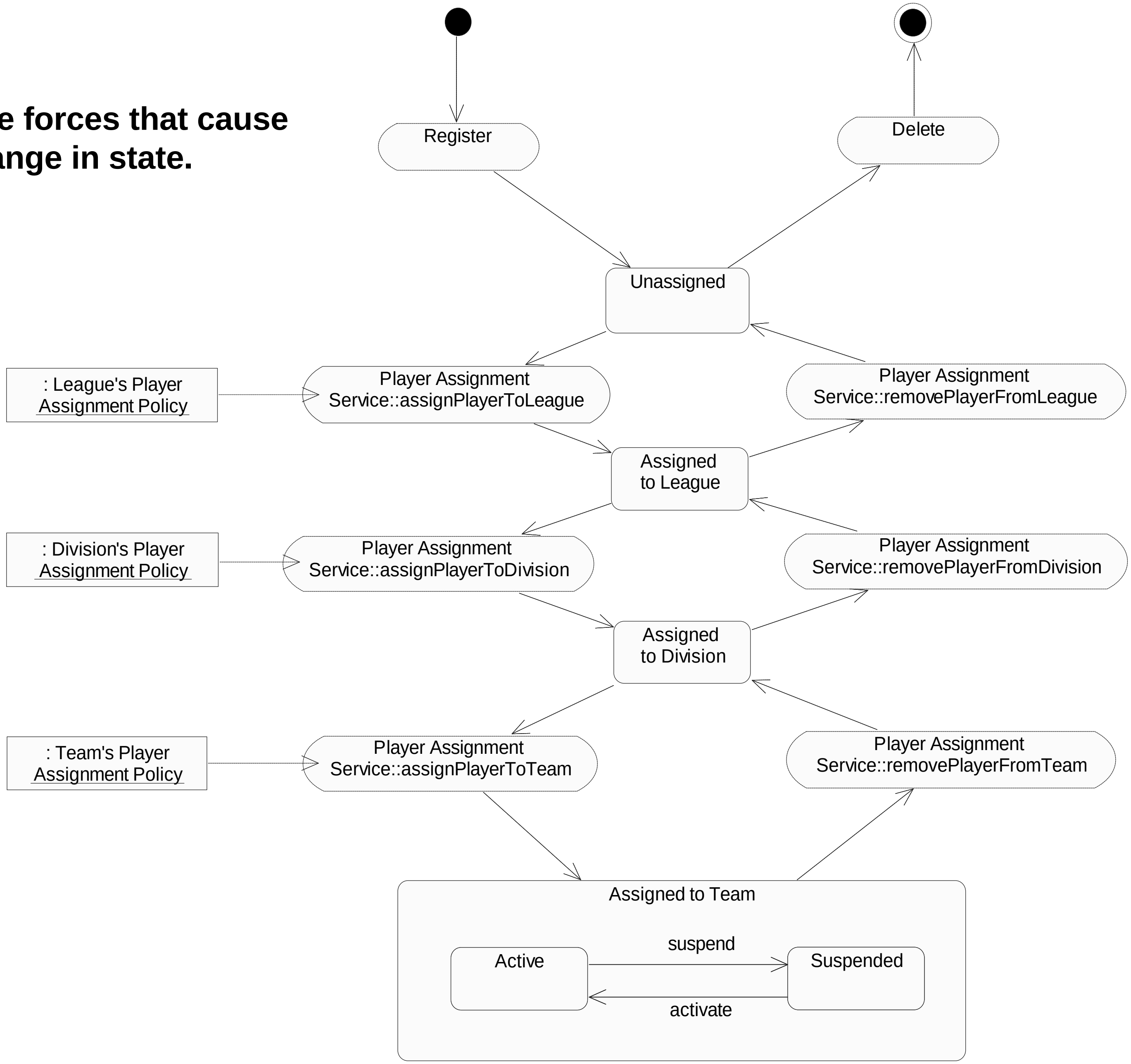
Shows a representative or exceptional sample of the system state.

(Metalevel M0)

Modeling the Solution Space

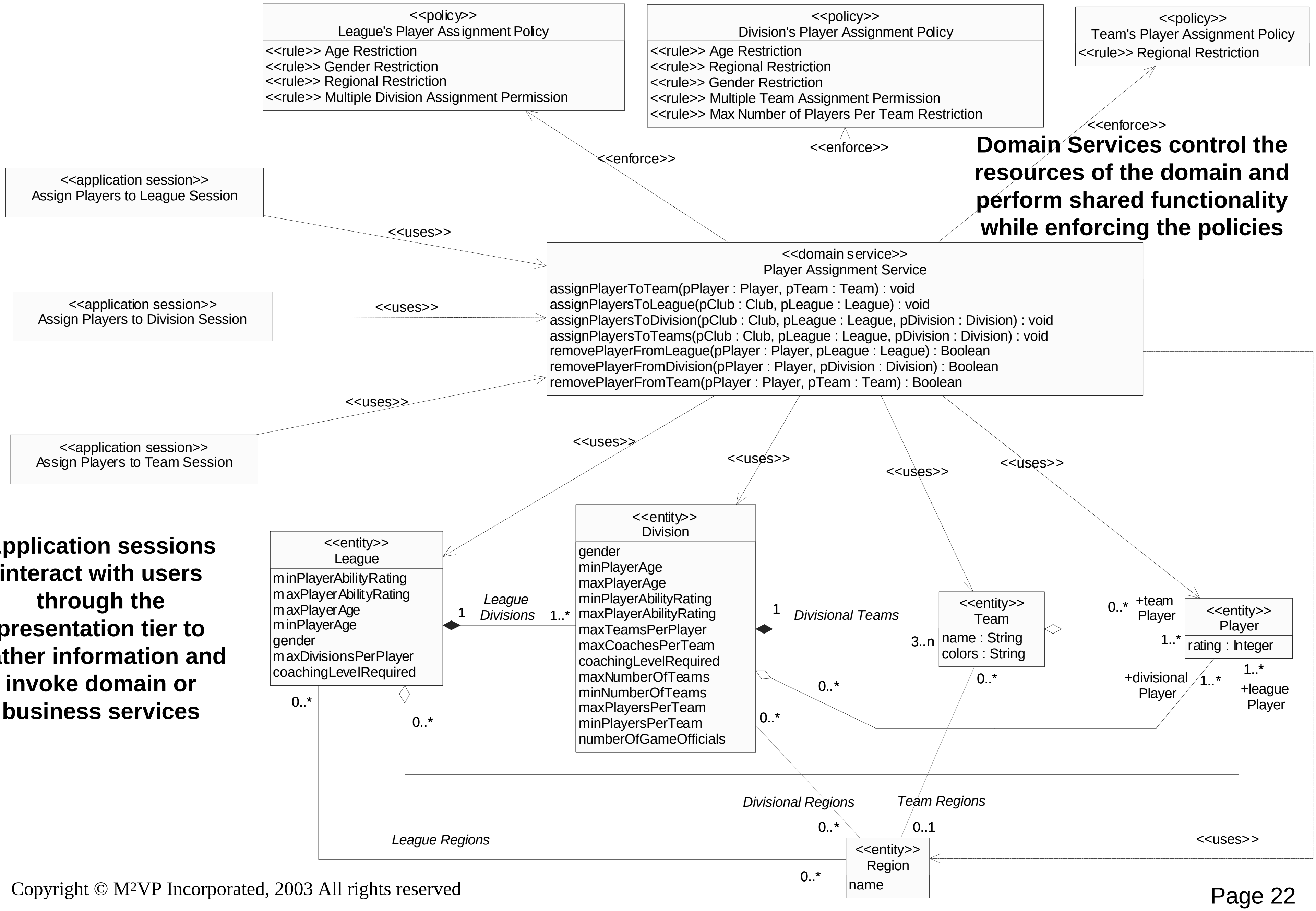
PIM IVP Dynamic Schema

Shows the forces that cause change in state.


Register
Delete
Unassigned
: League's Player Assignment Policy
Player Assignment Service::assignPlayerToLeague
Player Assignment Service::removePlayerFromLeague
Assigned to League
: Division's Player Assignment Policy
Player Assignment Service::assignPlayerToDivision
Player Assignment Service::removePlayerFromDivision
Assigned to Division
: Team's Player Assignment Policy
Player Assignment Service::assignPlayerToTeam
Player Assignment Service::removePlayerFromTeam
Assigned to Team
Active
suspend
Suspended
activate

Modeling the Solution Space

PIM CVP - Application Sessions & Domain Services



<<policy>>
League's Player Assignment Policy
- <<rule>> Age Restriction
- <<rule>> Gender Restriction
- <<rule>> Regional Restriction
- <<rule>> Multiple Division Assignment Permission

<<policy>>
Division's Player Assignment Policy
- <<rule>> Age Restriction
- <<rule>> Regional Restriction
- <<rule>> Gender Restriction
- <<rule>> Multiple Team Assignment Permission
- <<rule>> Max Number of Players Per Team Restriction

<<policy>>
Team's Player Assignment Policy
- <<rule>> Regional Restriction

<<application session>>
Assign Players to League Session

<<application session>>
Assign Players to Division Session

<<application session>>
Assign Players to Team Session

<<domain service>>
Player Assignment Service
- assignPlayerToTeam(pPlayer : Player, pTeam : Team) : void
- assignPlayersToLeague(pClub : Club, pLeague : League) : void
- assignPlayersToDivision(pClub : Club, pLeague : League, pDivision : Division) : void
- assignPlayersToTeams(pClub : Club, pLeague : League, pDivision : Division) : void
- removePlayerFromLeague(pPlayer : Player, pLeague : League) : Boolean
- removePlayerFromDivision(pPlayer : Player, pDivision : Division) : Boolean
- removePlayerFromTeam(pPlayer : Player, pTeam : Team) : Boolean

<<entity>>
League
- minPlayerAbilityRating
- maxPlayerAbilityRating
- maxPlayerAge
- minPlayerAge
- gender
- maxDivisionsPerPlayer
- coachingLevelRequired

<<entity>>
Division
- gender
- minPlayerAge
- maxPlayerAge
- minPlayerAbilityRating
- maxPlayerAbilityRating
- maxTeamsPerPlayer
- maxCoachesPerTeam
- coachingLevelRequired
- maxNumberOfTeams
- minNumberOfTeams
- maxPlayersPerTeam
- minPlayersPerTeam
- numberOfGameOfficials

<<entity>>
Team
- name : String
- colors : String

<<entity>>
Player
- rating : Integer

<<entity>>
Region
- name

Domain Services control the resources of the domain and perform shared functionality while enforcing the policies

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

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->includes(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 Player.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->includes(pPlayer)

-- *The player is in the division.*

post: pPlayer.division->includes(pDivision)

English and OCL provide declarative descriptions of operation processing

References & Bibiliography

- [Putnam] Janis R. Putnam, *Architecting with RM-ODP*, Prentice Hall PTR (2001)
- [RUP] Rational Unified Process, Version 2002.05.00, Rational Corporation
- [BASS] Len Bass, Paul Clements, Rick Kazman, *Software Architecture in Practice*, Addision Wesley Longman, Inc. (1998)
- [MDA] Object Management Group, *Model Driven Architecture (MDA), Document number ormsc/2001-07-01* (2001) available at: http://www.omg.org/mda
- Jos Warmer and Anneke Kleppe, *The Object Constraint Language*, Addision Wesley Longman, Inc. (1999)
- Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, *Pattern – Oriented Software Architecture*, John Wiley & Sons, Ltd. (1996)
- Rebecca Wirfs-Brock, Brian Wilkerson, Lauren Wiener, Rebecca Brock, Designing Object-Oriented Software, Prentice Hall PTR (1990)
- Grady Booch, James Rumbaugh, Ivar Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley Longman, (1999)
- Ivar Jacobson, Grady Booch, James Rumbaugh, *The Unified Software Development Process*, Addison Wesley Longman, (1999)
- James Rumbaugh, Ivar Jacobson, Grady Booch, *The Unified Modeling Language Reference Manual*, Addison Wesley Longman, (1999)
- Terry Merriman, Anatomy of a Platform Independent Model, (2003), Cutter Consortium, http://www.cutter.com

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