Desarrollo de software dirigido por modelos

M.I. Capel

ETS Ingenierías Informática
y Telecommunicación
Universidad de Granada
Email: manuelcapel@ugr.es

Desarrollo de Software

Índice

Índice

Índice

Índice

Concepto de desarrollo dirigido por modelos

- En IS se utilizan *modelos* desde hace décadas:
  - los modelos han de ser algo más que diagramas–guía para ayudar a programar,
  - aparición de *herramientas de modelado* (CASE-80's, etc.)
  - capacidad de *generación de código* a partir de los modelos.
- El *modelo* se convierte en una forma de programación

[OMG 2001]
"*Model Driven Architecture* is a *framework* for software development defined by the Object Management Group (OMG). The key of MDA is the importance of models in the software development process. Within MDA the software development process is driven by the activity of modeling your software system".

Caracterización del desarrollo dirigido por modelos

Ideas fundamentales:

- No elimina la necesidad de programar algoritmos
- Las herramientas han de proporcionar varios *lenguajes de acciones* específicos
- UML no especifica la sintaxis concreta de un lenguaje de este tipo!!!!
- Existen varios enfoques para abordar un desarrollo dirigido por modelos, uno de ellos es MDA.

Model Driven Architecture (MDA)

Ciclo de Desarrollo

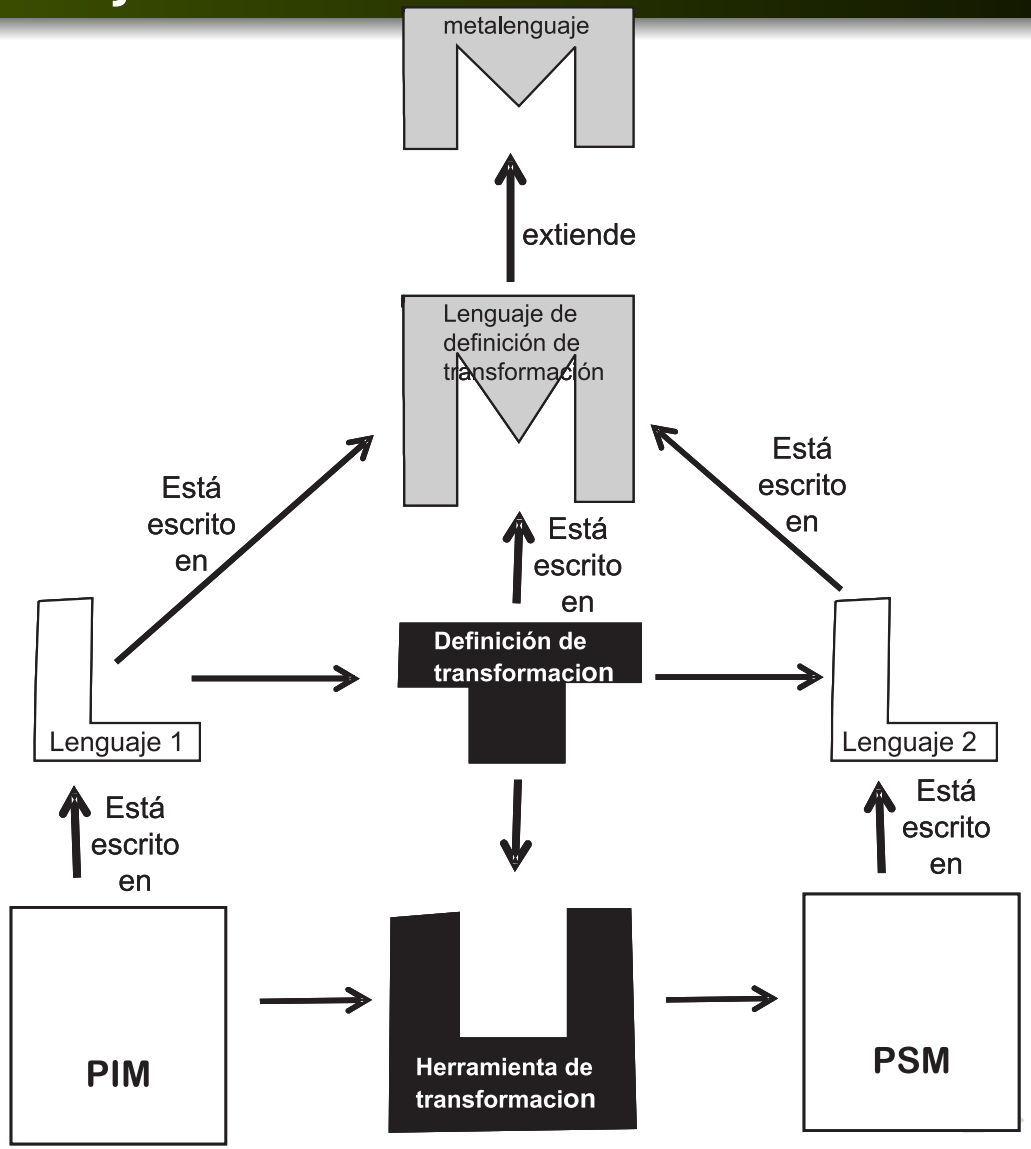
1. Modelo Computacionalmente Independiente (CIM)
2. Modelo Independiente de la Plataforma (PIM)
3. Modelo Dependiente de la Plataforma (PSM)
4. Código

Herramientas de Transformación

Transformaciones en MDA

-Tienen una importancia fundamental dentro del proceso de desarrollo de software
-Se aplican sucesiva e iterativamente
-Hasta conseguir un *modelo ejecutable* del sistema objetivo

Marco de trabajo básico con MDA



Transformaciones en MDA

Dependen de los siguientes elementos

- *Cómo se define la Transformación*
- Invariancia de la definición de transformación sobre cualquier modelo de entrada
- *Definición de una transformación*
- *Regla de transformación*
- Las transformaciones han de *preservar la semántica* del modelo fuente en el modelo destino.

Motivación

Desarrollo basado en transformaciones entre modelos

- Falta de interoperabilidad entre las tecnologías de desarrollo de software actuales
- Facilidades de evaluación / desempeño del software
- Evolución tecnológica muy rápida, difícil de gestionar: obsolescencia frecuente
- Independencia de la *lógica de negocio* respecto de las tecnologías

Importante

Las transformaciones en MDA dan respuesta estas necesidades de los usuarios de tecnologías.

Información adicional

Fuente y destino de las transformaciones

- No existen limitaciones en cuanto a los lenguajes, en principio, siempre se pueden encontrar transformaciones
- Podrían utilizarse los mismos lenguajes: *refactorización* del software
- El compartir el mismo lenguaje no garantiza un propósito común o igual semántica de los modelos
- Los *perfiles de UML* son lenguajes nuevos y diferentes entre sí

PIM y PSM

Importancia

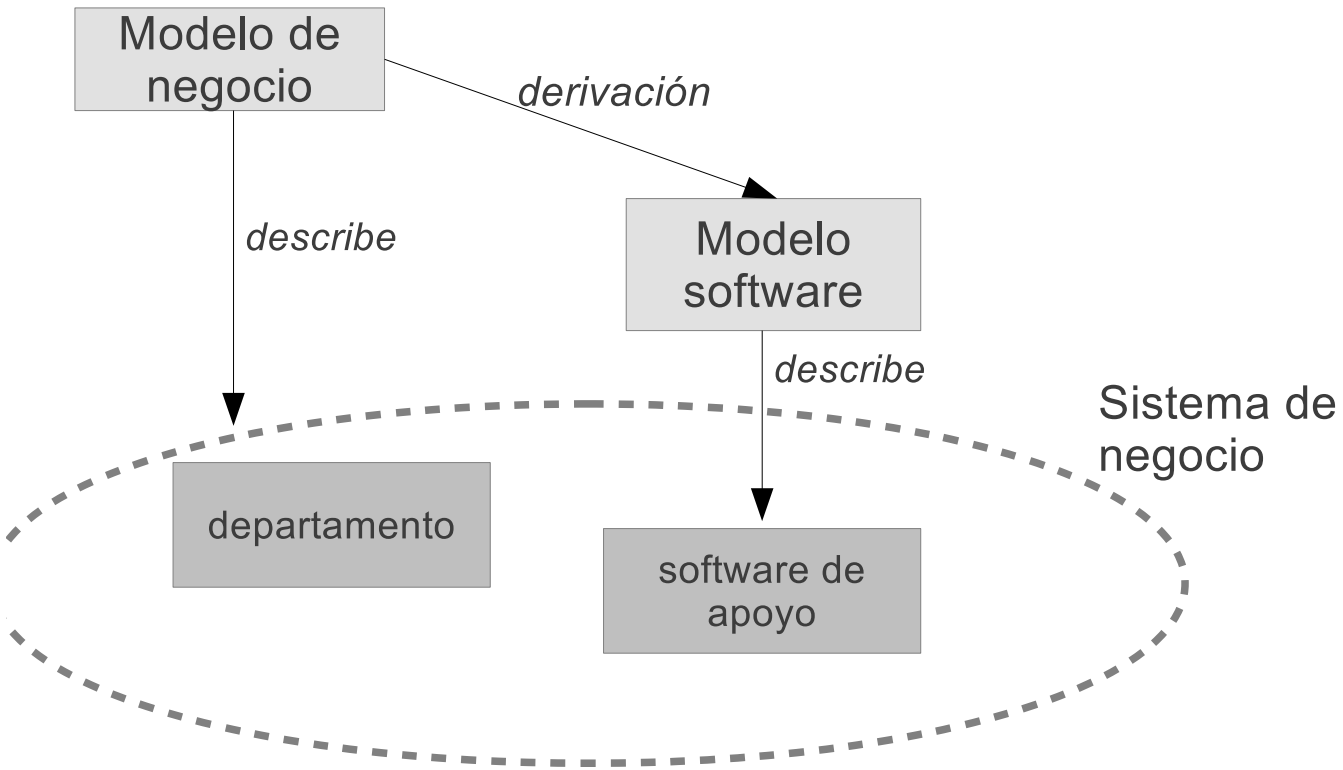
- Obtener el PIM y el PSM son los elementos más importantes para el programador
- Un diseñador de software pondrá todo su interés en desarrollar un buen PIM
- La siguiente etapa: PIM-> PSM se puede llevar a cabo automáticamente con la herramienta adecuada
- Un PIM, a menudo, puede generar múltiples PSMs con *puentes* para pasar a ellos

Modelos de software y de negocio

Significado de los modelos

- Sólo se puede preservar si el modelo es expresable en los dos lenguajes: *fuente* y *destino*
- Los dos lenguajes pueden coincidir y tendríamos un tipo especial de transformación denominada *refactorización*
- Perfiles UML

Tipos de modelos



Selección

- Depende del nivel de detalle que consideremos del sistema
- Los modelos de negocio no indican nada de los sistemas software
- CIM: modelo independiente del software, para describir un sistema de negocio

Tipos de modelos

Dinámicos y estructurales

- Vista de casos de uso
- Vista de interacción
- Vista de diagrama de estados
- Vista de diagrama de clases

Modelos específicos e independientes de la plataforma

Características

- Resultan muy difíciles de distinguir
- PIM y PSM son términos relativos
- Importancia de la plataforma donde se instala el sistema objetivo

Diseño con MDA

Etapas

1. Diseño Modular
2. Arquitectura del Diseño
3. Diseño de la Interfaz de Usuario
4. Obtención del PSM
5. Diseño detallado y específico para lenguajes de programación

Diseño modular

Cómo se hace:

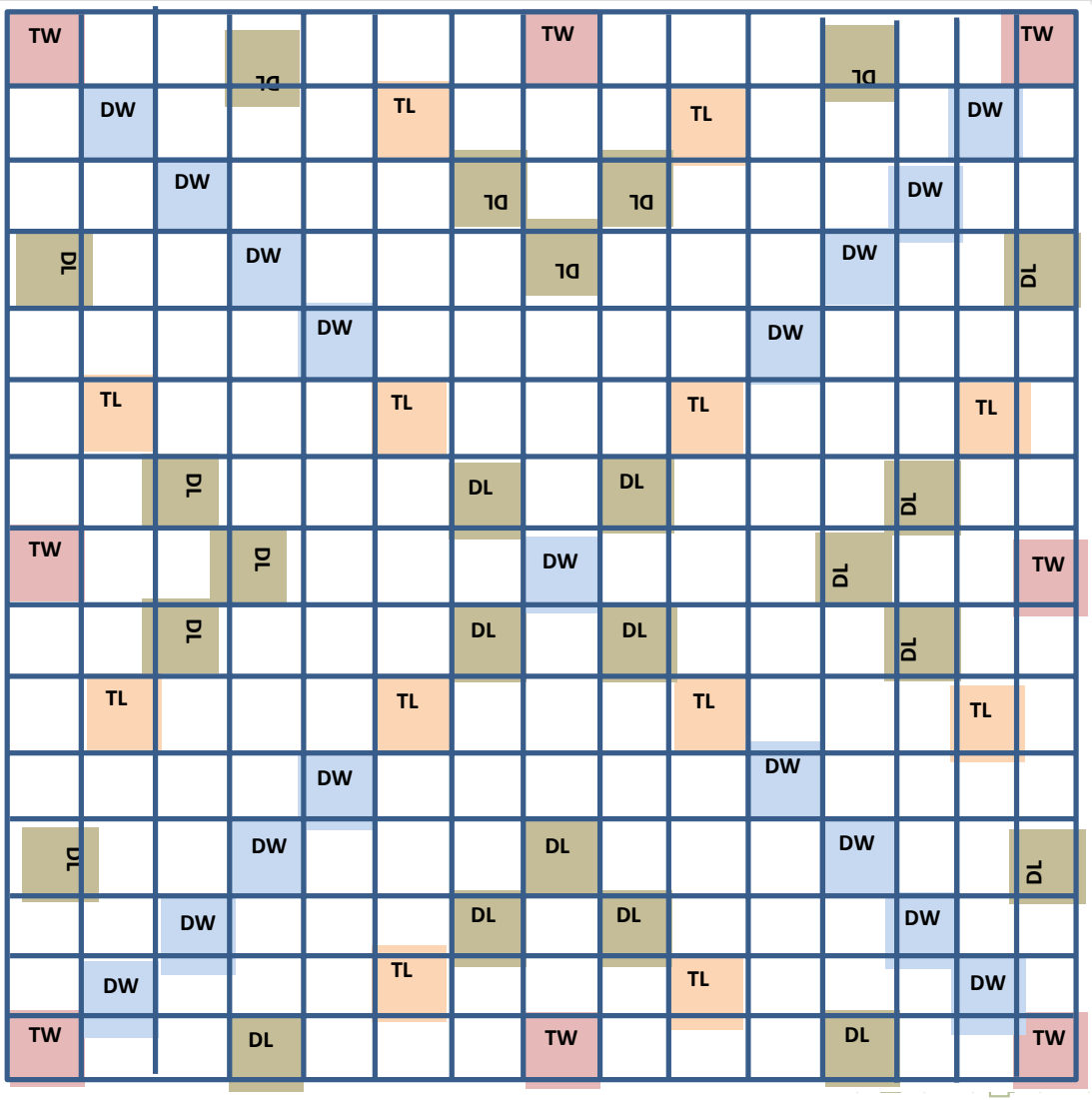
Se utiliza lenguaje natural o descripciones en UML con el objetivo de definir las responsabilidades de los diferentes módulos, sus operaciones, los datos a los que acceden y el interfaz.

Ejemplo: juego del Scrabble

Descripción del juego

Scrabble es un juego parecido a un *crucigrama* en el que cuatro jugadores consiguen puntos formando palabras, utilizando para ello piezas de una letra que se colocan como en un puzzle, sobre un tablero de 15x15 casillas.

Tablero de Scrabble



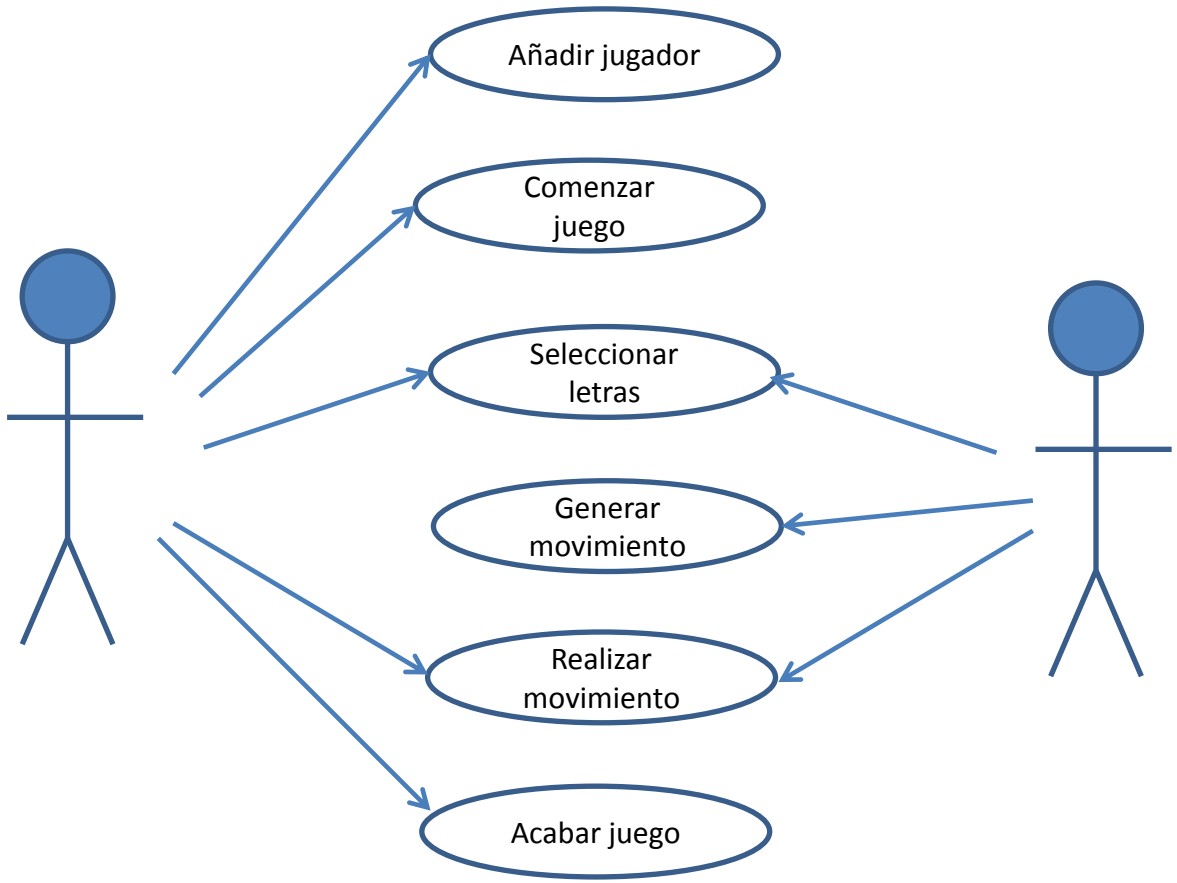
Movimiento en Scrabble

F : nueva letra

F : letra existente

Nuevas palabras formadas:
ha
Ira
atoro
foz
es

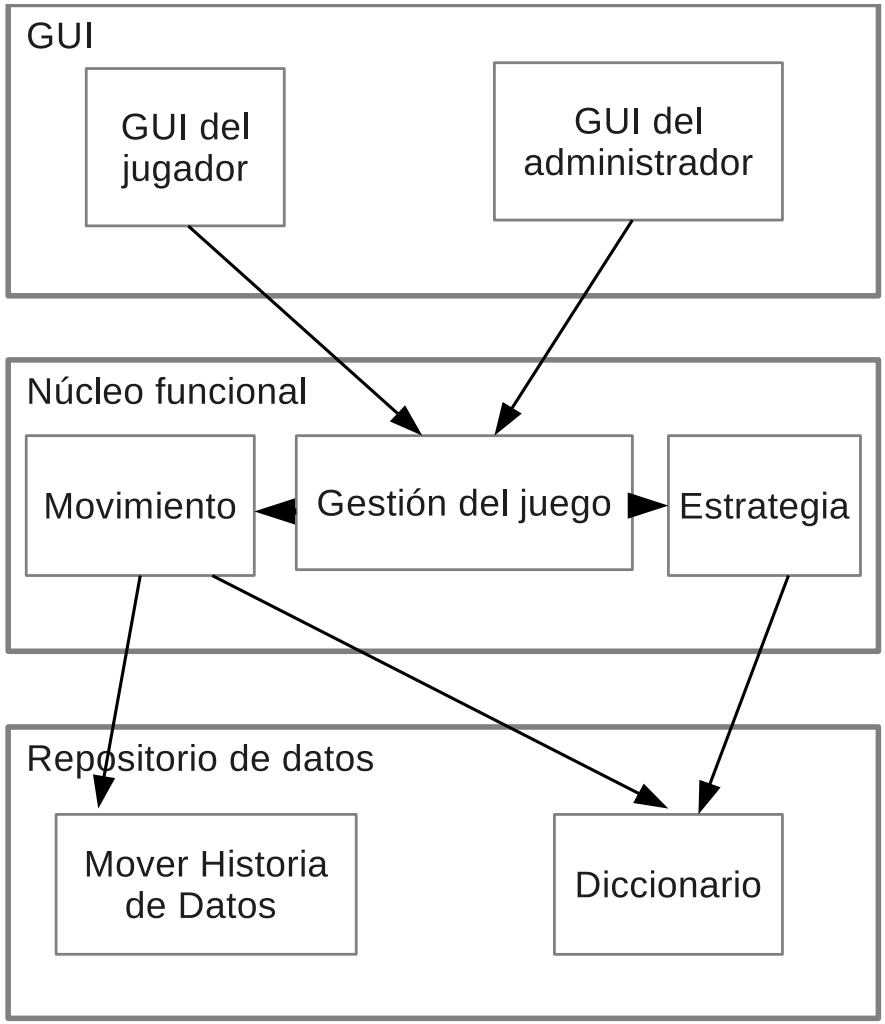
Descripción casos usos de un jugador



Descripción de los módulos

```
Module Move;
MANAGES DATA : Move, LetterMove, Word
CONSTRAINTS
  C1: m : Move => m.letterMoves.x.size = 1 or
                  m.letterMoves.y.size = 1
  C2: ...
OPERATIONS
  addLetterMove(m: Move, lm: LetterMove)
  validateMove(m: Move, n: Integer): Boolean
  findWords(m: Move, b:Board)
  calculateScore(m: Move, b:Board): Integer
// post: m.letterMoves.size < 7 =>
//          result= m.wordsFormed.getScore(b).sum &
//          m.score = m.wordsFormed.getScore(b).sum
// post: m.letterMoves.size = 7 =>
//         result = m.wordsFormed.getScore(b).sum + 50 &
//         m.score = m.wordsFormed.getScore(b).sum + 50
```

Sistema Scrabble



Arquitectura de Diseño

Principio General

Los elementos que pertenezcan al mismo grupo deben tener *más en común* entre ellos que con los elementos de un grupo distinto.

Objetivo de diseño

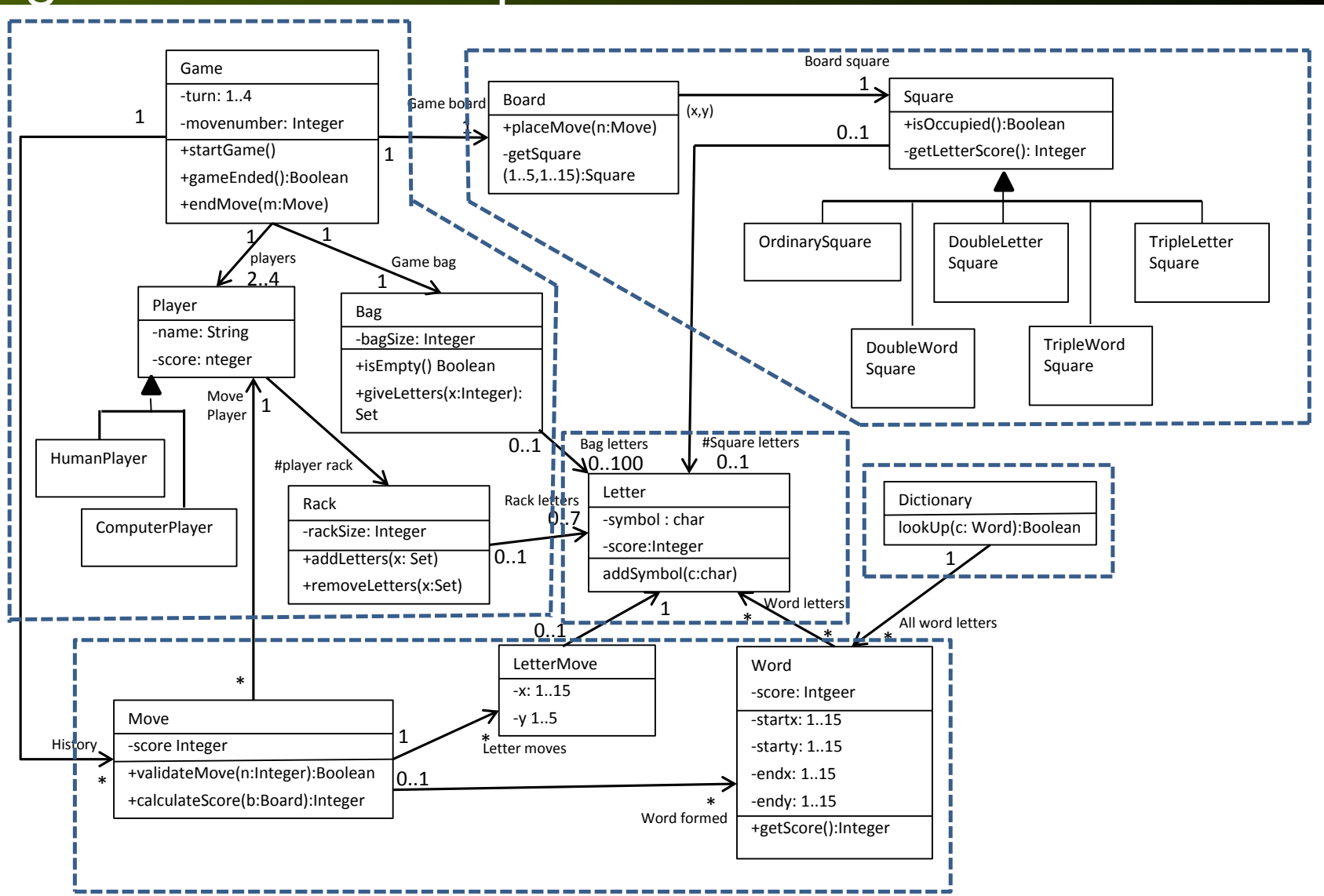
Un módulo o un subsistema debe representar un conjunto cohesivo de elementos de modelado, con un propósito claro, y que posea un papel en el sistema más grande.

Estrategias de diseño

Criterios de agrupación

- **Subordinación** entre clases
- **Coherencia Lógica** dentro del grupo
- **Coherencia funcional** y casos de uso
- **Agregación**
- **No solapamiento**
- **Minimizar las dependencias** entre los grupos

Diagrama de clases para Scrabble



Estrategias de diseño para GUIs

Objetivos

- Especificación de apariencia, comportamiento y aspecto independientes de la plataforma
- El comportamiento del GUI se expresa con *Statecharts*
- La estructura: mediante un diagrama de clases utilizando *patrones*

Statechart

Acoplamiento débil entre módulos

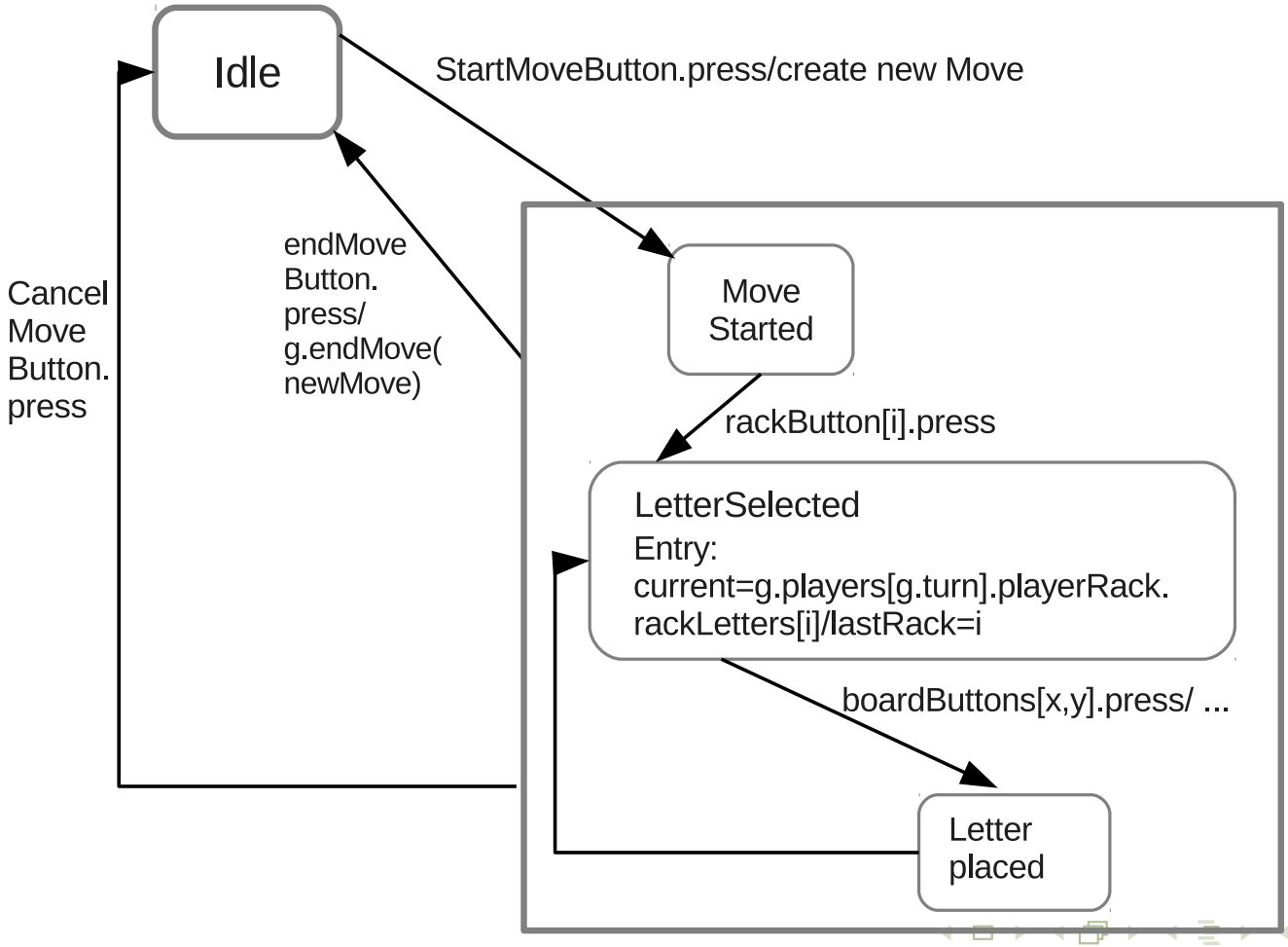
Permite a los módulos el ser desarrollados y refinados de una forma relativamente independiente entre ellos

Conceptos GUI independientes de la plataforma

Concepto GUI	**XHTML**	**Swing (Java)**
button	button	Jbutton
text field	$< \text{inputtype} = '' \text{text}'' / >$	JTextField
checkboxes	$< \text{inputtype} = '' \text{checkbox}'' / >$	JCheckBox
radio buttons	$< \text{inputtype} = '' \text{radio}'' / >$	JRadioButton
selection lists	$< \text{select...} / >$	JComboBox
table	table	JTable
label	text	JLabel
frame	frame	JFrame
dialog	form	JDialog

Tabla: Correspondencia entre conceptos GUI

Statechart para creación de un movimiento en Scrabble



Transformación de PIM a PSM

Principio director

Puesto que los lenguajes de programación y las plataformas software difieren significativamente en su semántica, hay que tener cuidado con que el significado correcto del PIM sea también correctamente expresado en el PSM.

Técnicas de transformación

Aspectos a considerar para hacer buenas transformaciones

- Las restricciones definidas sobre los datos han de mantenerse
- Preservar la interpretación de las expresiones booleanas y de las restricciones OCL
- Mantener la semántica de *sobrecarga* y *redefinición* de los métodos, tanto en el lenguaje fuente como en el de destino
- Reglas características propias de los lenguajes (p.e.: ausencia de *herencia múltiple*)
- Conservar la dirección de navegación de las asociaciones
- No modificar las anotaciones de *visibilidad* de los elementos incluidos en paquetes y clases

Transformaciones a PSM–Java

Reglas a respetar

- *Sobrecarga de métodos*: no es aplicable a dos métodos con el mismo nombre pero diferentes tipos de salida
- El polimorfismo en Java (tiempo de ejecución) sólo se aplica al objeto en el cual se ejecuta el método, nunca a los parámetros.
- Métodos estáticos no pueden redefinirse en subclases.
- Es inválido intentar redefinir la *visibilidad* de un método en una subclase.
- Si una clase contiene un método abstracto, ella misma ha de ser también abstracta.
- Las *interfaces* no pueden contener constructores o atributos cuyo ámbito sea la instancia.

UML como lenguaje para PIM

Fortalezas y debilidades

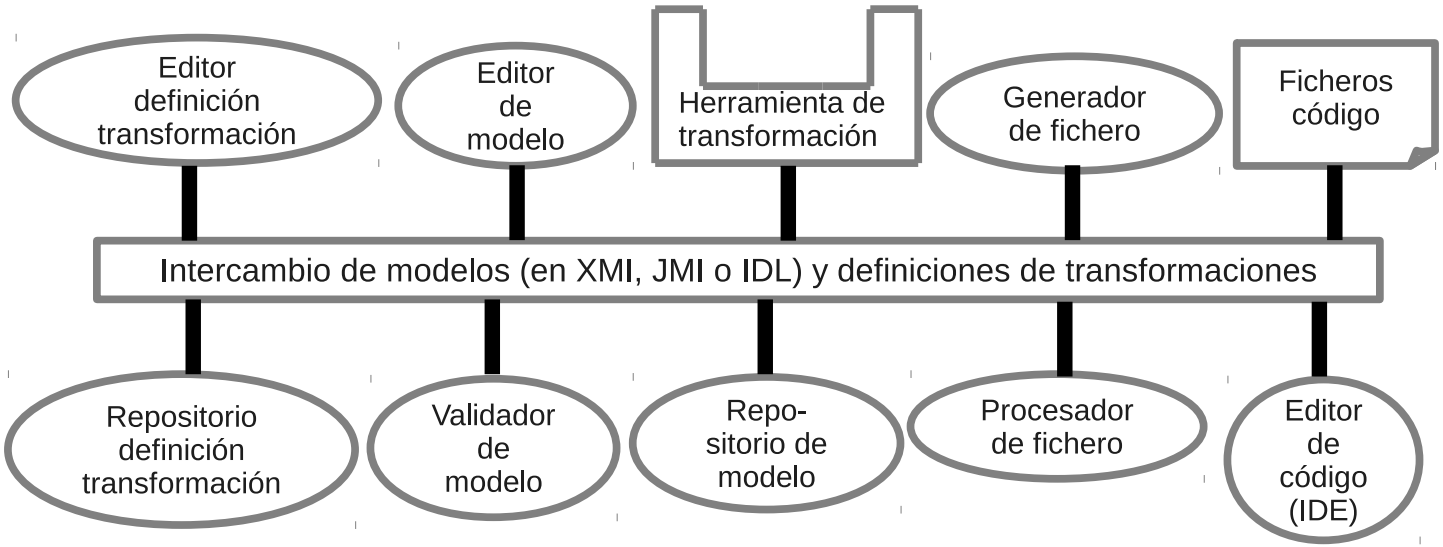
- Descripción de la estructura del sistema
- Falta de capacidad descriptiva de aspectos comportamentales o dinámicos
- UML Ejecutable: *Action Semantics*
- UML + OCL(*Object Constraint Language*)

Herramientas

Tipos:

- Editor de Código (IDE)
- Archivos de código (procesador, generador)
- Repositorio del modelo
- Editor de modelos
- Validador de modelos
- Editor–repositorio definición transformaciones

Funcionalidad de las herramientas para un entorno MDA



Desarrollo de software ágil

Objetivo fundamental

“Working software is valued over comprehensive documentation"

Programación Extrema (XP)

Características

- Se ha de tener un sistema funcionante todo el tiempo
- Desarrollo incremental y en pequeños pasos
- Definición de *marcadores*
- “Modelado Extremo"

Proceso Unificado Racional

RUP

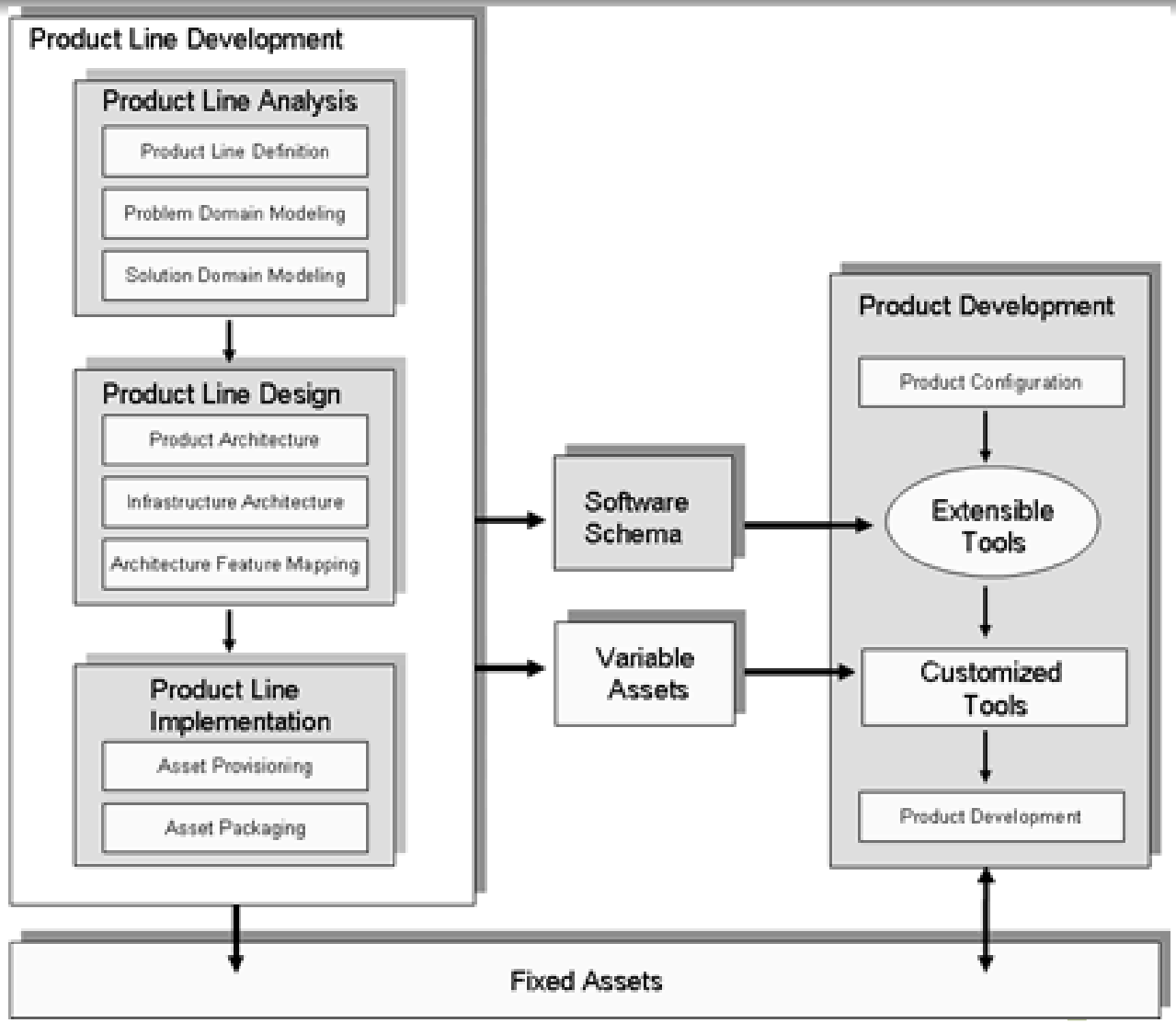
- Se trata de un modelo de desarrollo muy completo
- Actualmente es estándar en grandes proyectos
- Los artefactos RUP pueden considerarse modelos MDA si encontramos una descripción UML adecuada para describirlos

Factoría

Características

- Colección de software (plantillas, configuraciones, entornos y vistas) para crear tipos de software específico
- Aplican principios de la industria manufacturera a la producción de software
- Se habla de *líneas de* producto y arquitecturas
- Eliminación de la codificación en el nivel de aplicación
- Líneas–guía y ejemplos

Estructura de una factoría de software



Componentes de una factoría

- **Esquema de factoría**: documentos XML
- **Implementación de referencia**
- **Guia arquitectónica y patrones**
- **“How-To"**
- **Recetas**
- **Plantillas**
- **Componentes**

Construcción de un producto software con una factoría

Actividades:

- Análisis del problema
- Especificación del problema
- Diseño del producto
- Implementación del producto
- Despliegue del producto
- Testeo del producto

Desarrollar software con factorías

Beneficios

- Consistencia de las líneas de producto
- Calidad
- Productividad
- Valor añadido (negocio,arquitecturas,desarrollo,operaciones)

Producción y factorías de software

Situación actual:

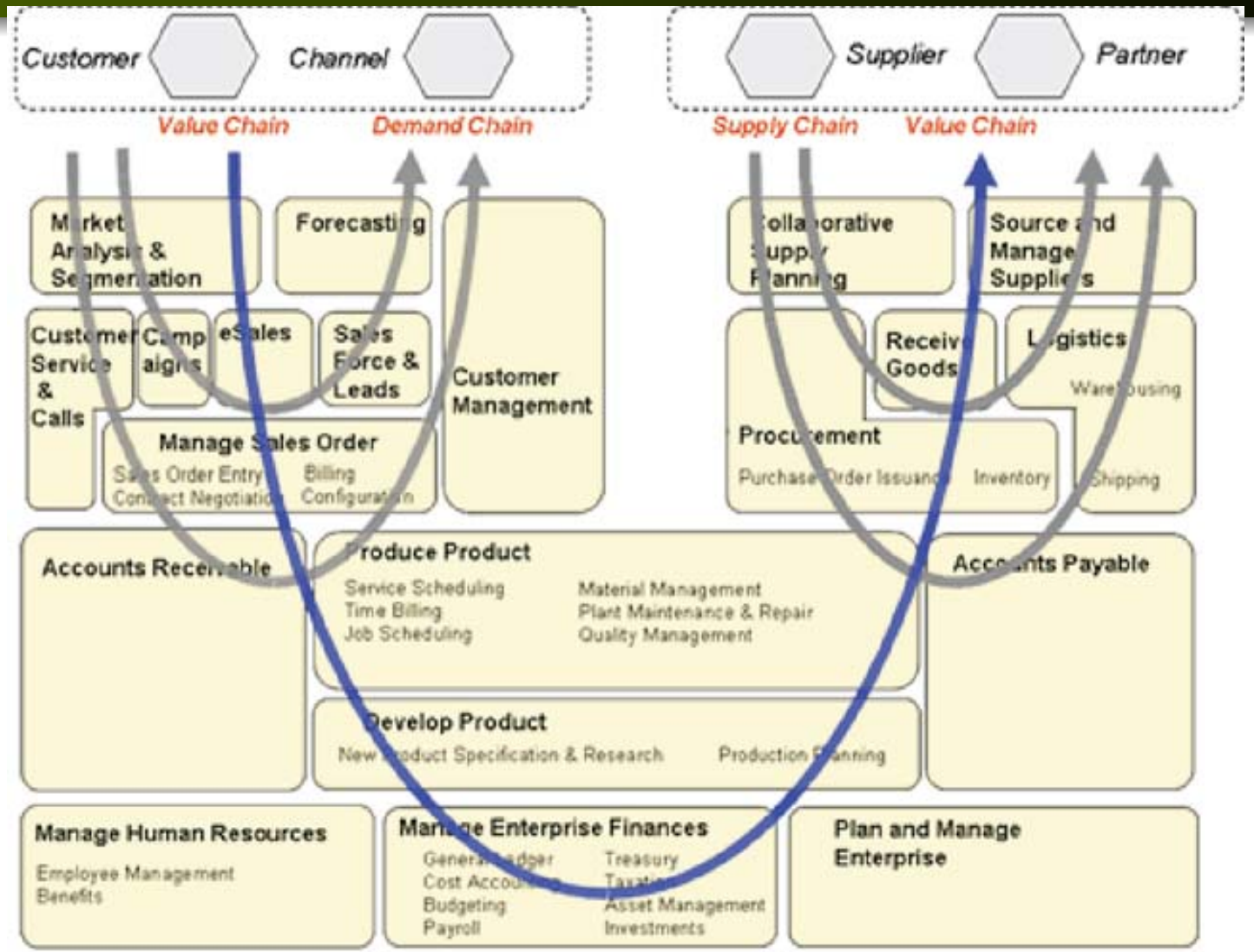
-Menos de la tercera parte de un sistema software es desarrollado desde cero.
-Los desarrolladores, en lugar de escribir código nuevo, obtienen la funcionalidad que necesitan de componentes–software pre-existentes.

Producción masiva de software

Elementos

- Cadenas de suministro de componentes–software: suministradores vs. desarrolladores/consumidores
- Especificación de requisitos, especialización y *tercerización*
- Convergencia de arquitecturas software entre procesos suministradores y consumidores de componentes
- Propagación de restricciones asociadas a componentes–software

Cadena de valor de Porter



Producción masiva de software II

Integración de la cadena de valor de Porter:

"La producción masiva de software se podrá conseguir cuando los suministradores de componentes consigan optimizar sus cadenas de valor.

Implementaciones

- Marco de trabajo .NET de Microsoft
- Enterprise Framework Factory (EFx Factory)
- Actualmente de Microsoft:
  1. Share Point Software Factory (2010) y Visual Studio(2008/2010)
  2. Smart Client Software Factory (2008) para Visual Studio
  3. Web Service Software Factory (2010)
  4. Mobile Client Software Factory (2006)

Modelado de Negocio

"Business Modeling *buzzword*"

- Difusión:
  - Más de 5.9 millones de enlaces a documentos en inglés,
  - Más de 1.1 millones de enlaces a documentos en español.
- En librerías digitales especializadas:
  - *ACM digital library:* 20,000 hits para "business process modeling"
  - *IEEE digital library:* 750 hits para "business modeling"

Modelado de Negocio

Modelo de negocio

Abstracción acerca de cómo funciona un negocio. Depende del(os) experto(s) que lo realice(n). Vista simplificada de la estructura del negocio para difundirlo o mejorarlo. Se considera parte de la definición de requerimientos del sistema de información al que da lugar.

Modelado de Negocio

Negocio

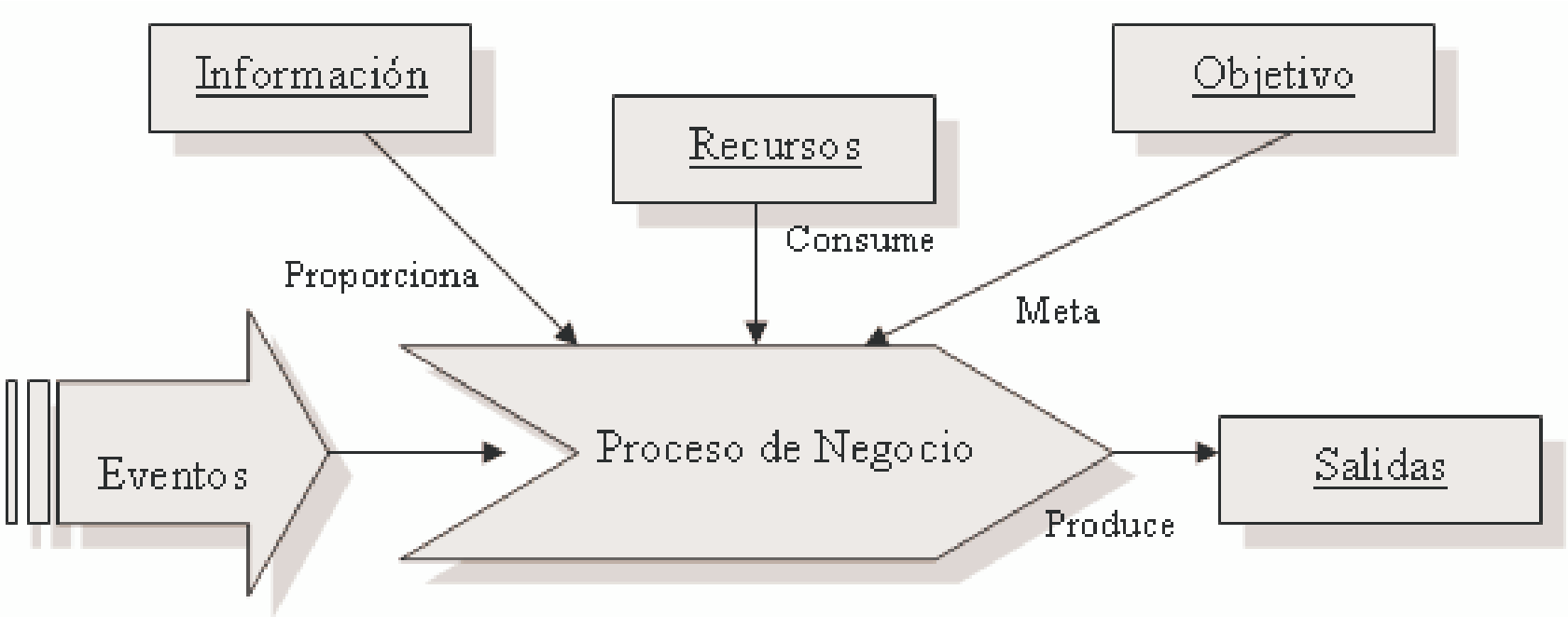
Conjunto de actividades de una empresa que posea o use recursos, tenga varias metas y si se lleva a cabo correctamente crea valor (normalmente económico) para la empresa.

Motivación de utilizar la tecnología de PN

Beneficios

- Los PN son el patrimonio más valioso de una empresa
- Comprender mejor los mecanismos clave de un negocio.
- Actuar como las bases para la creación de sistemas de información adecuados que soporten al negocio.
- Dar soporte a la mejora de la estructura y operación del negocio actual.
- Mostrar la estructura de un negocio innovador.
- Experimentar con nuevos conceptos de negocio o copiar o estudiar un concepto usado por una empresa competidora.
- Identificar oportunidades de subcontratación de productos o suministros.

Proceso de Negocio



Bibliografía Fundamental


Eriksson, H. and Penker, M. (2000).
Business Modelling with UML.
OMG Press–John Wiley, New-York, N.Y.


Greenfield, J. (2004).
Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools.
Microsoft.


Klepple, A., Warmer, J., and Bast, W. (2004).
MDA Explained. The model driven architecture: practice and promise.
Addison Wesley–Pearson Education.


Lethbridge, S. and Balcer, M. (2002).
Executable UML: A Foundation for Model-Driven Architectures.
Addison-Wesley – Longman Publishing Company, Boston, MA, USAS.