

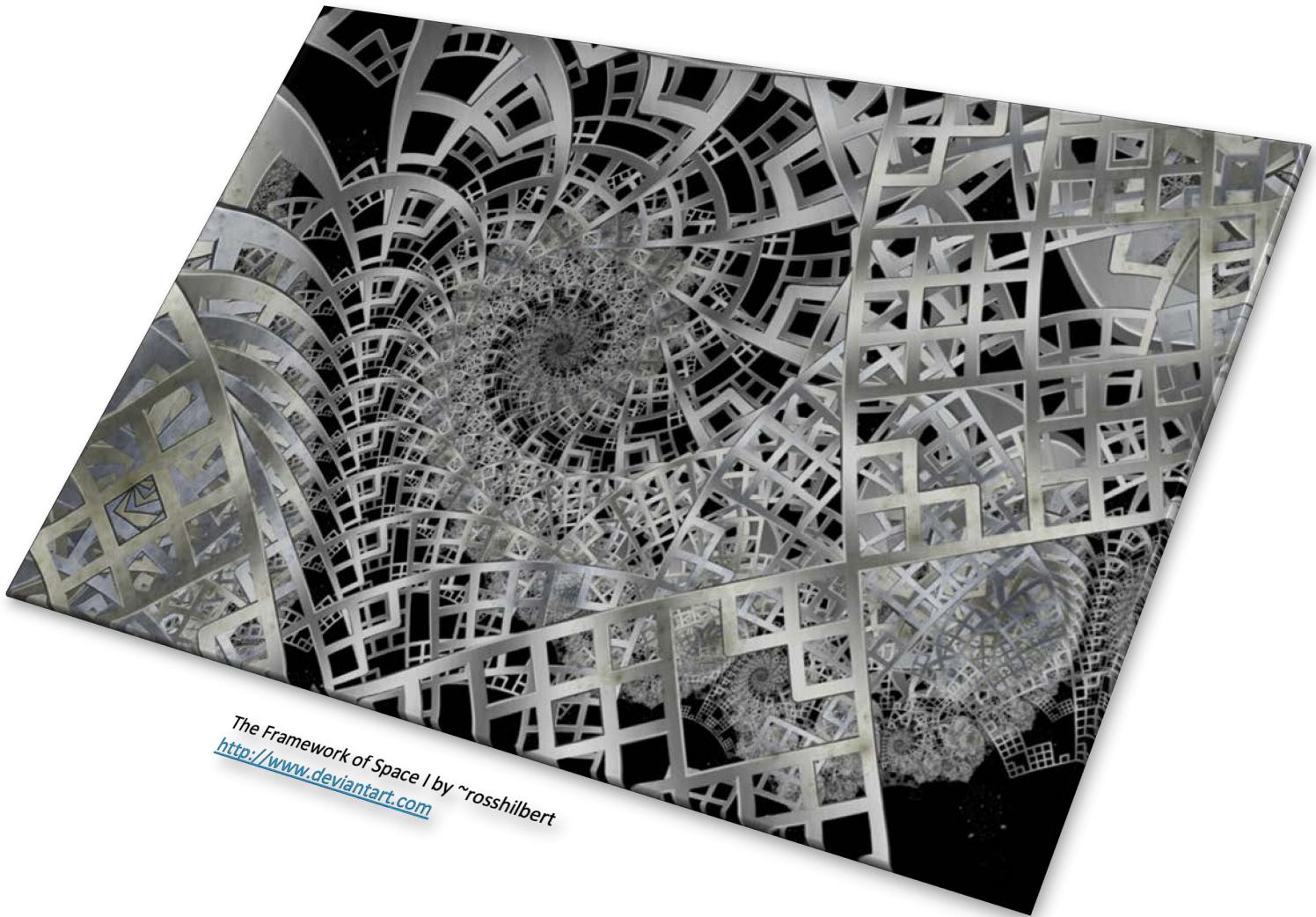
# Ingeniería web

## Tema 2

JLA 2021 v. 1.5

# Ingeniería de Aplicaciones web

- Marco de trabajo y metodologías ágiles
- Modelo HDM
- Modelo RMM
- Modelo OOHDM
- Modelo OOWS
- Modelo UWE

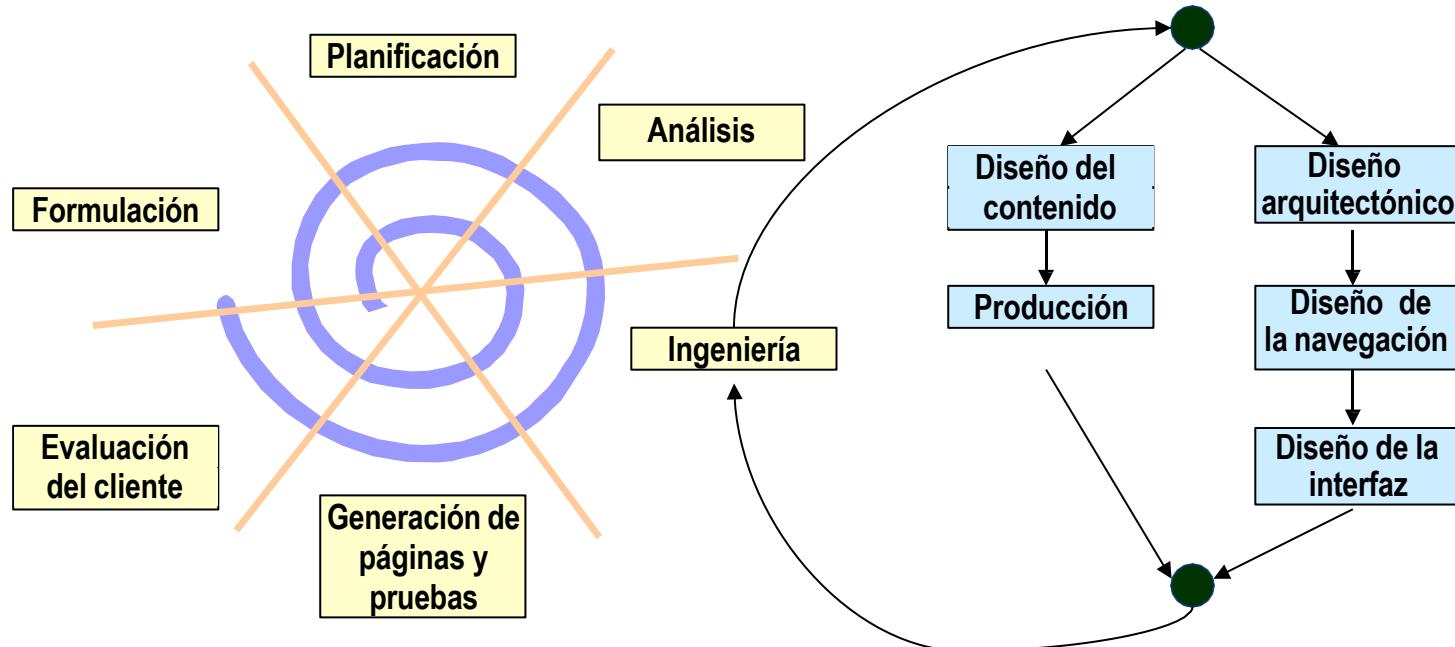


*The Framework of Space I* by ~rosshilbert  
<http://www.deviantart.com>

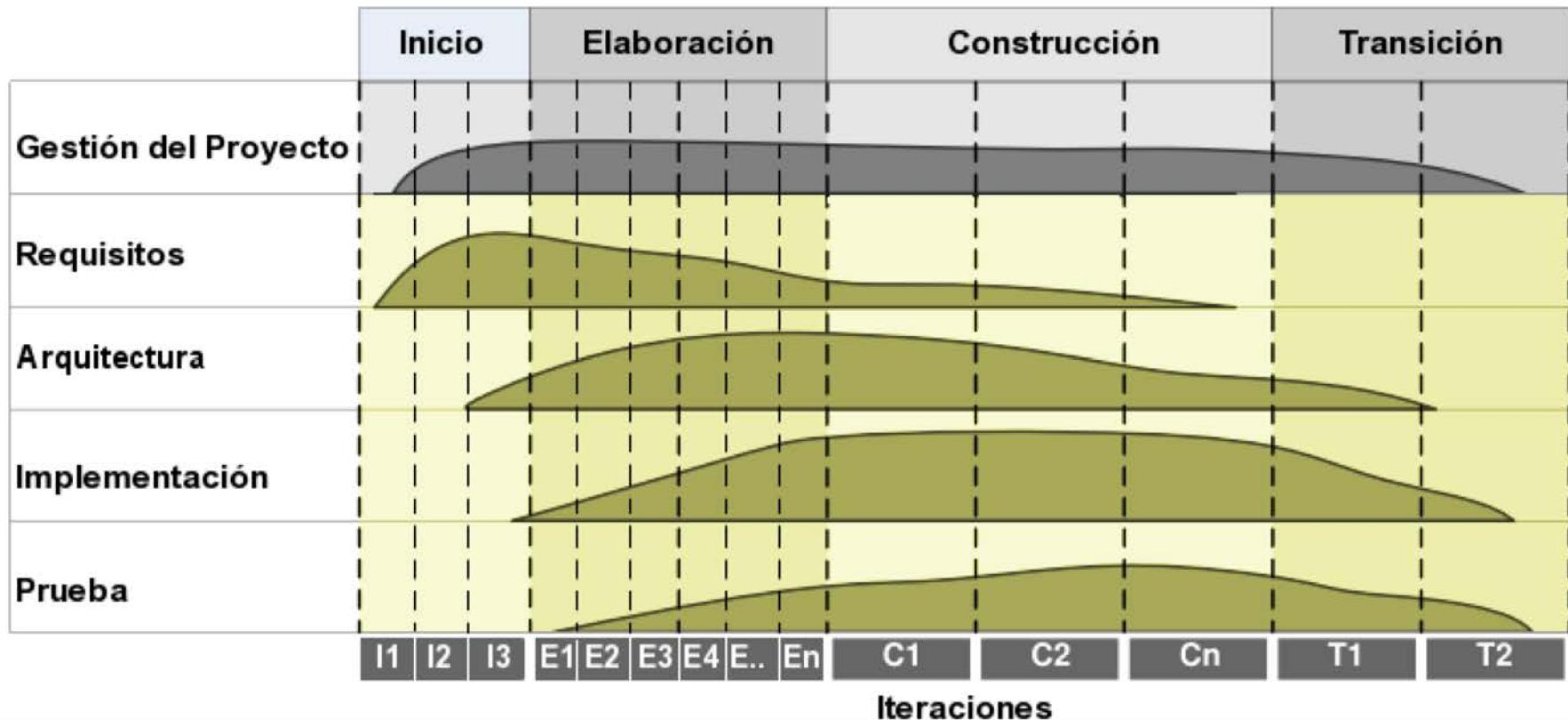
# 1. Marco de trabajo

# Proceso iterativo e incremental para la Ingeniería Web

- R. S. Pressman & B. R. Maxim (2015) proponen un proceso para la Ingeniería Web basado en el modelo en espiral de Boehm (1986)

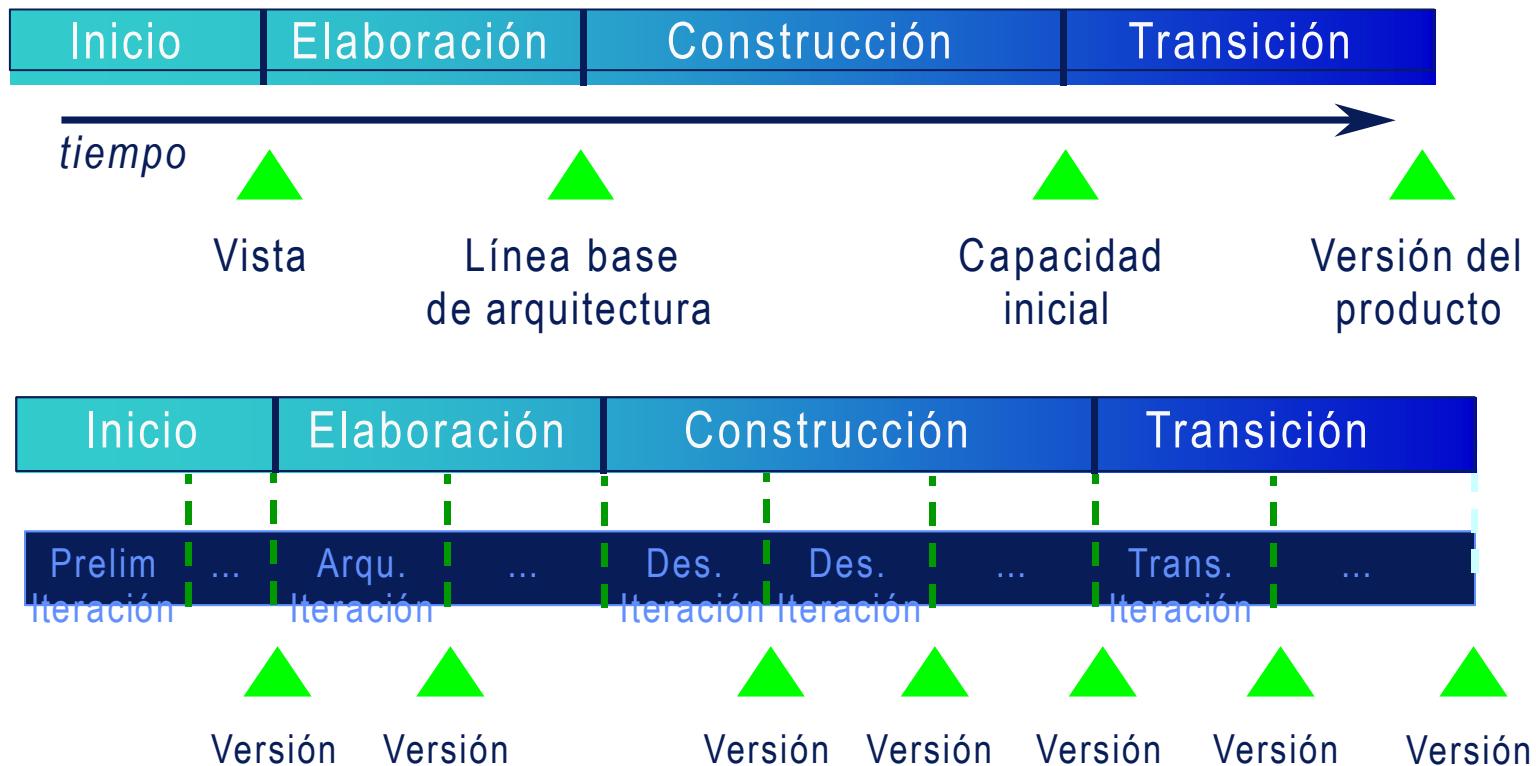


# Proceso Unificado

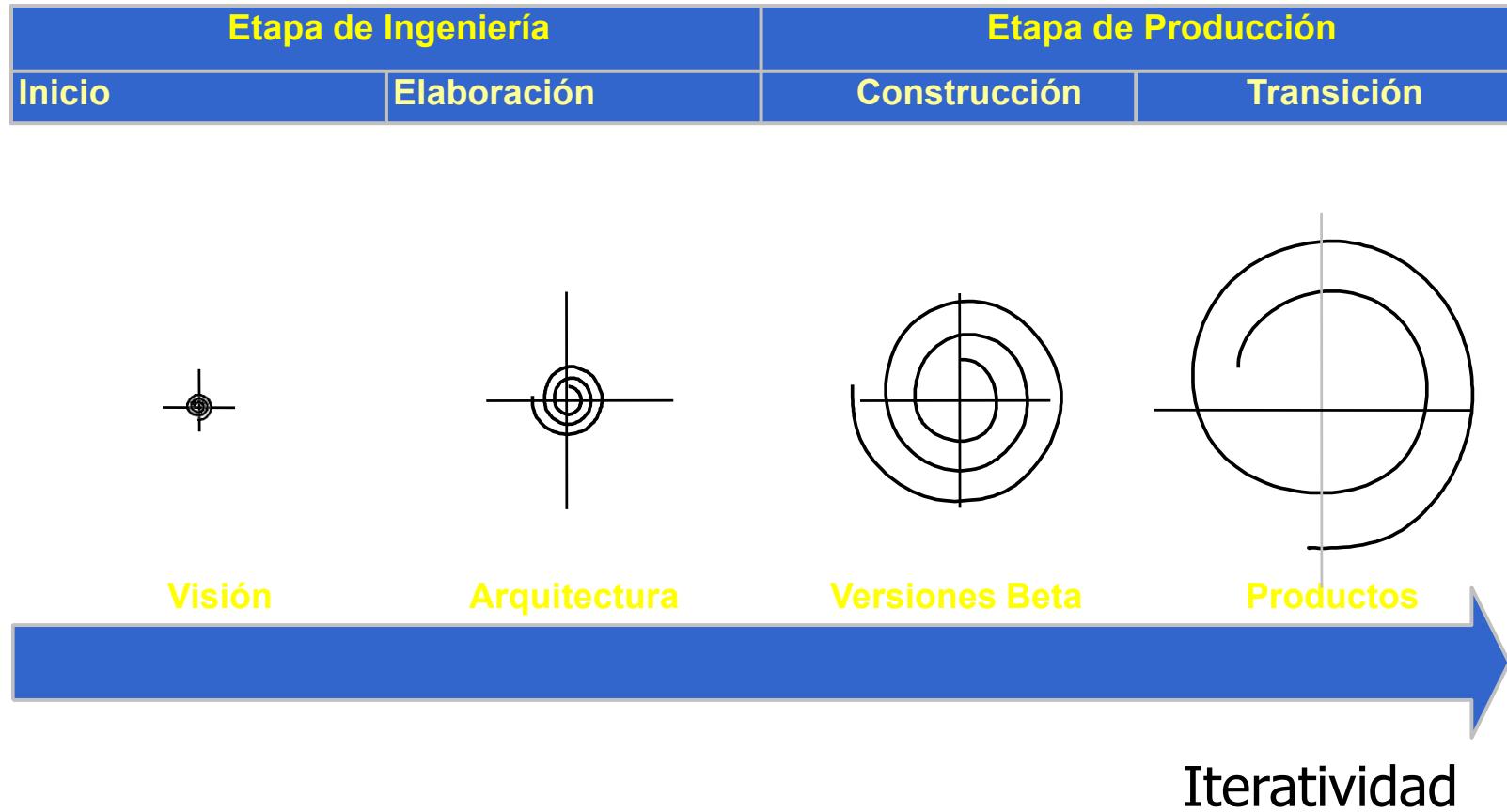


# Proceso Unificado

Cada ciclo concluye con una versión del producto para los clientes



# Proceso iterativo e incremental



## 2. Métodos ágiles



# Procesos ágiles

- Los sistemas web caracterizados, entre otras cosas, por su presión temporal priorizan el *cuándo* sobre el *qué*, como sucede en las aplicaciones tradicionales
- Esta reducción del ciclo de desarrollo del *software*, sin perder calidad ni capacidad de evolución y de mantenimiento, es uno de los retos a los que se enfrenta la Ingeniería Web



■ A los procesos *software* que intentan dar solución a estas circunstancias se les conoce con el nombre de procesos ágiles o procesos ligeros



# Procesos ágiles

- Las aproximaciones ágiles emplean procesos técnicos y de gestión que continuamente se adaptan y se ajustan a (Turk et al., 2002)
  - Cambios derivados de las experiencias ganadas durante el desarrollo
  - Cambios en los requisitos
  - Cambios en el entorno de desarrollo
- La agilidad en el desarrollo del *software* no significa únicamente poner en el mercado o en explotación los productos *software* más rápidamente
  - Esto choca frontalmente con los modelos de procesos tradicionales que son monolíticos y lentos, centrados en una única iteración o ciclo de larga duración

# Procesos ágiles

- La agilidad significa respuesta efectiva al cambio



incluye

- Estructuras de equipo y actitudes para facilitar la comunicación entre los miembros
- Énfasis en la entrega rápida de *software* funcional, para lo que resta importancia a los productos intermedios (lo cual siempre no es bueno)
- Adopción del cliente como parte del equipo de desarrollo
- Planificación incierta y, por tanto, flexible

# eXtreme Programming

- Controvertido enfoque de desarrollo de *software* basado en el modelo incremental
- Está indicado para
  - Equipos de tamaño mediano o pequeño
  - Requisitos imprecisos y cambiantes
- Características
  - El juego de la planificación
  - Versiones pequeñas
  - Metáfora
  - Diseño sencillo
  - Hacer pruebas
  - Refactorización
  - Programación en parejas
  - Propiedad colectiva
  - Integración continua
  - Cliente in-situ
  - Estándares de codificación

# eXtreme Programming vs Proceso Unificado

- Detractores del Proceso Unificado
  - Argumentan, en favor de las aproximaciones ágiles en general y de la Programación Extrema en particular, que el Proceso Unificado es un proceso pesado que obliga a hacer gran cantidad de actividades inútiles y antinaturales
- Seguidores del Proceso Unificado
  - El Proceso Unificado es un marco de trabajo abierto y flexible con un modelo de proceso riguroso, que se puede adaptar al proceso de desarrollo que se necesite, pero sin caer en el caos al que invita la Programación Extrema per se (Kruchten, 2001)

# eXtreme Programming vs Proceso Unificado

- El Proceso Unificado puede adaptarse para incluir algunas de las prácticas de la Programación Extrema, pero no son propuestas idénticas en ningún caso
- El Proceso Unificado permite construir procesos para dar soporte a proyectos que están fuera del alcance la Programación Extrema, tanto por su escala como por su tipo
- El Proceso Unificado es un proceso pesado por la completa descripción de una familia de procesos, que pueden ser tanto ligeros como pesados, mientras que la Programación Extrema es totalmente ligera porque pone su énfasis en la implementación, implicando que las cosas hay que descubrirlas, inventarlas o definirlas ad hoc

(Smith, 2001)

# Scrum

- Propuesto por Jeff Sutherland y desarrollado por Schwaber y Beedle (Schwaber, 2007)
- El conjunto de buenas prácticas de Scrum se aplica esencialmente a la gestión de proyecto
- Es un *framework*, por lo que es necesaria su adaptación en cada organización o incluso en cada equipo
- El objetivo es obtener resultados rápidos con adaptación a los cambios de las necesidades de los clientes
- Las principales características de Scrum
  - El desarrollo *software* mediante iteraciones incrementales
  - Las reuniones a lo largo del proyecto

**Scrum se basa en entregas parciales priorizadas por el beneficio que aporta al receptor final del producto**

# Scrum

- Actividades estructurales
  - Requisitos
  - Análisis
  - Diseño
  - Evolución
  - Entrega
- Dentro de cada actividad las tareas se organizan con un patrón de proceso denominado *sprint*
- El trabajo del *sprint* se adapta al problema y se define y modifica en tiempo real por el *equipo Scrum*
- Uso de patrones de proceso de demostrada eficacia en proyectos críticos, con plazos cortos y requisitos cambiantes

# Scrum

- Scrum define un ciclo de vida iterativo e incremental, que mejora la gestión del riesgo y aumenta la comunicación
- Se basa en tres pilares
  - Transparencia
    - Todos los aspectos del proceso que afectan al resultado son visibles para todos aquellos que administran dicho resultado
  - Inspección
    - Se debe controlar con la suficiente frecuencia los diversos aspectos del proceso para que puedan detectarse variaciones inaceptables en el mismo
  - Revisión
    - El producto debe estar dentro de los límites aceptables
    - En caso de desviación se procederá a una adaptación del proceso y del material procesado
    - Mecanismo de mejora continua, esto es, de control, para adaptarse y mejorar

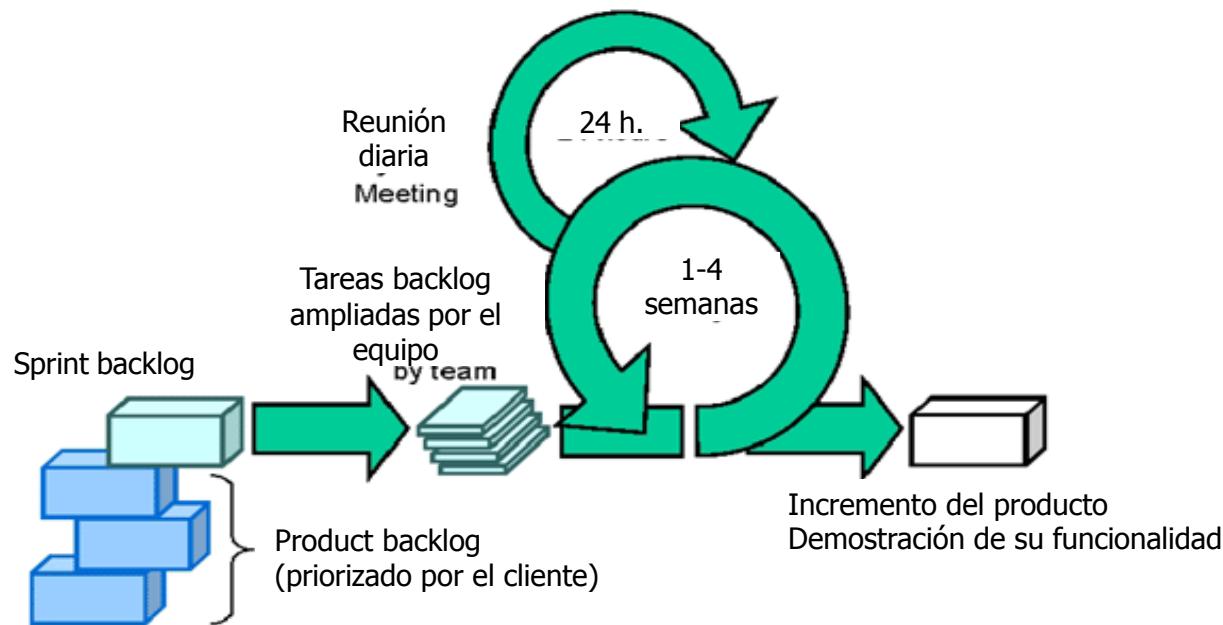
# Scrum

- Cada equipo Scrum tiene tres roles
  - *Scrum Master*. Es el responsable de asegurar que el equipo Scrum siga las prácticas de Scrum. Sus funciones
    - Ayudar a que el equipo y la organización adopten Scrum
    - Liderar el equipo Scrum para buscar la mejora en la productividad y la calidad de los entregables
    - Ayudar a la autogestión del equipo
    - Gestionar e intentar resolver los impedimentos con los que el equipo se encuentra para cumplir las tareas del proyecto
  - *Product owner*. Es la persona responsable de gestionar las necesidades que se quieren satisfacer mediante el desarrollo del proyecto. Sus funciones
    - Recolectar las necesidades (historias de usuario)
    - Gestionar y ordenar las necesidades
    - Aceptar el producto *software* al finalizar cada iteración
    - Maximizar el retorno de la inversión del proyecto
  - Equipo de desarrollo. Tiene las siguientes características
    - Autogestionado. El mismo equipo supervisa su trabajo (no existe el rol clásico de jefe de proyecto)
    - Multifuncional. Cada integrante del equipo debe ser capaz de realizar cualquier función
    - No distribuidos. Es conveniente que el equipo se encuentre en el mismo lugar físico
    - Tamaño óptimo. Al menos tres personas, máximo nueve, sin contar al *scrum master* ni al *product owner*

# Scrum

- Acciones de los patrones de proceso
  - *Retraso (pila de producto o product backlog)*: priorización de requisitos. Debe estar detallado de manera adecuada, estimado, emergente y priorizado
  - *Sprints*: unidades de trabajo requeridas para alcanzar un requisito. Es cada iteración. Se recomiendan iteraciones cortas (1-4 semanas) y cuyo resultado será un producto *software* potencialmente entregable. El equipo de desarrollo selecciona las historias de usuario que se van a desarrollar en el *sprint* para conformar así la pila de *sprint* (*sprint Backlog*). La definición de cómo descomponer, analizar o desarrollar este *sprint backlog* queda a criterio del equipo de desarrollo. Además, la lista de tareas se mantendrá inamovible durante toda la iteración
  - *Reuniones Scrum*: reuniones breves dirigidas por el *maestro Scrum*
  - *Demostraciones preliminares*: entrega de un incremento al cliente

# Scrum



# Scrum

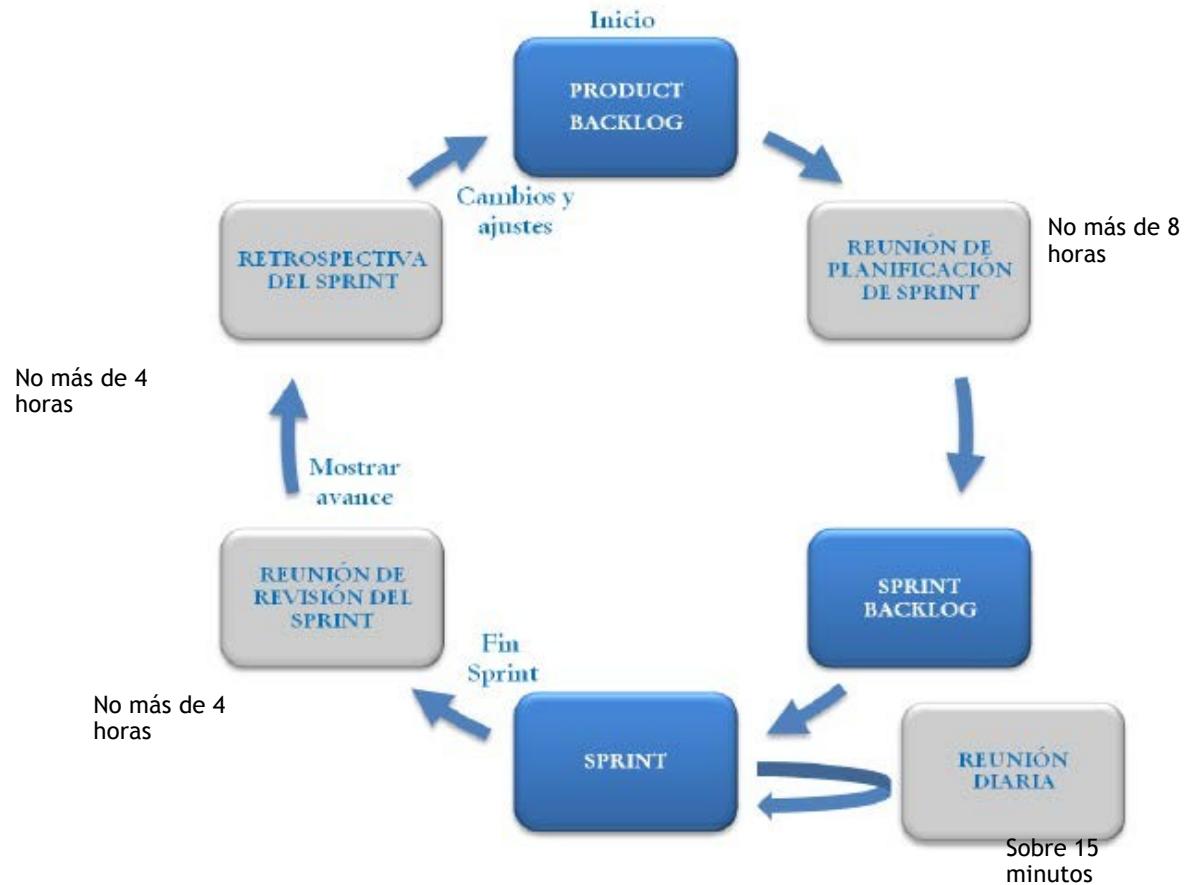
- Para mejorar la gestión de las historias de usuario y las tareas de cada *sprint* usualmente se utilizan pizarras u otros mecanismos que brinden información inmediata al equipo



(Garzás et al., 2012)

# Scrum

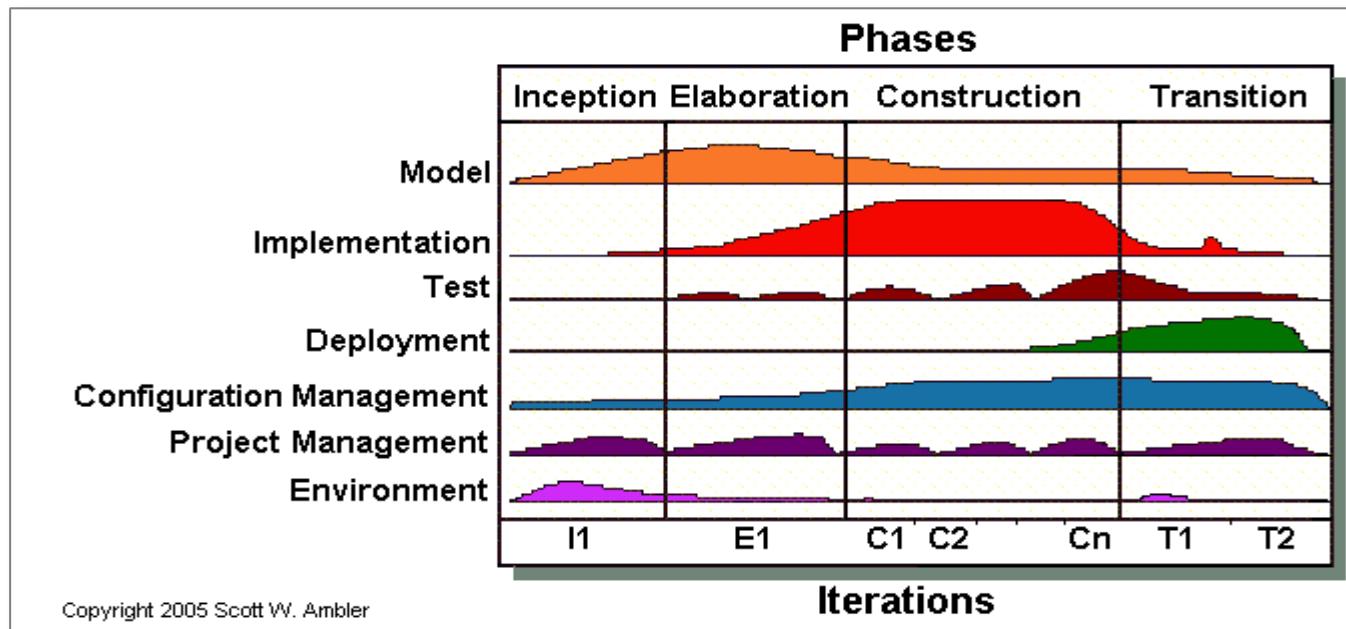
- Reuniones de Scrum



(Garzás et al., 2012)

# Proceso Unificado Ágil

- Versión simplificada del Proceso Unificado (PU) desarrollada por Scott W. Ambler (1994)
- Adaptación de PU a los principios de los procesos ágiles



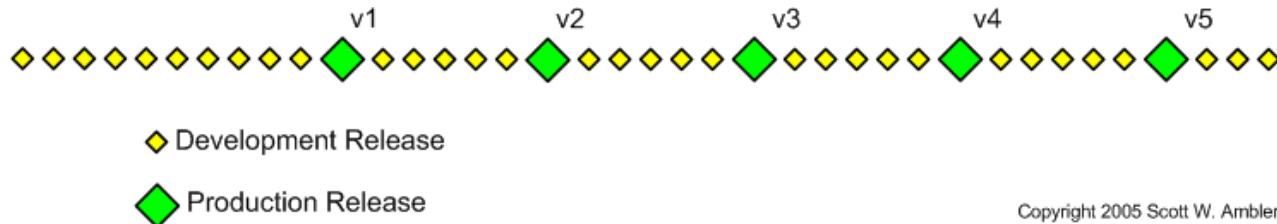
# Proceso Unificado Ágil

- **Disciplinas**

- **Modelo:** estudio del dominio del problema e identificación de una solución viable
- **Puesta en práctica:** transformación de los modelos en código ejecutable y realización un nivel básico de la prueba
- **Prueba:** realización de una evaluación objetiva para asegurar calidad
- **Despliegue:** planificación y entrega del sistema a los usuarios finales
- **Gestión de la configuración:** control de cambios y gestión de versiones
- **Gestión del proyecto:** gestión del riesgo, de recursos, de costes, organización de tareas, seguimiento, etc.
- **Ambiente:** actividades para asegurar que el proceso, las normas (estándares y directrices), y las herramientas apropiados (*hardware, software, etc.*) están disponibles

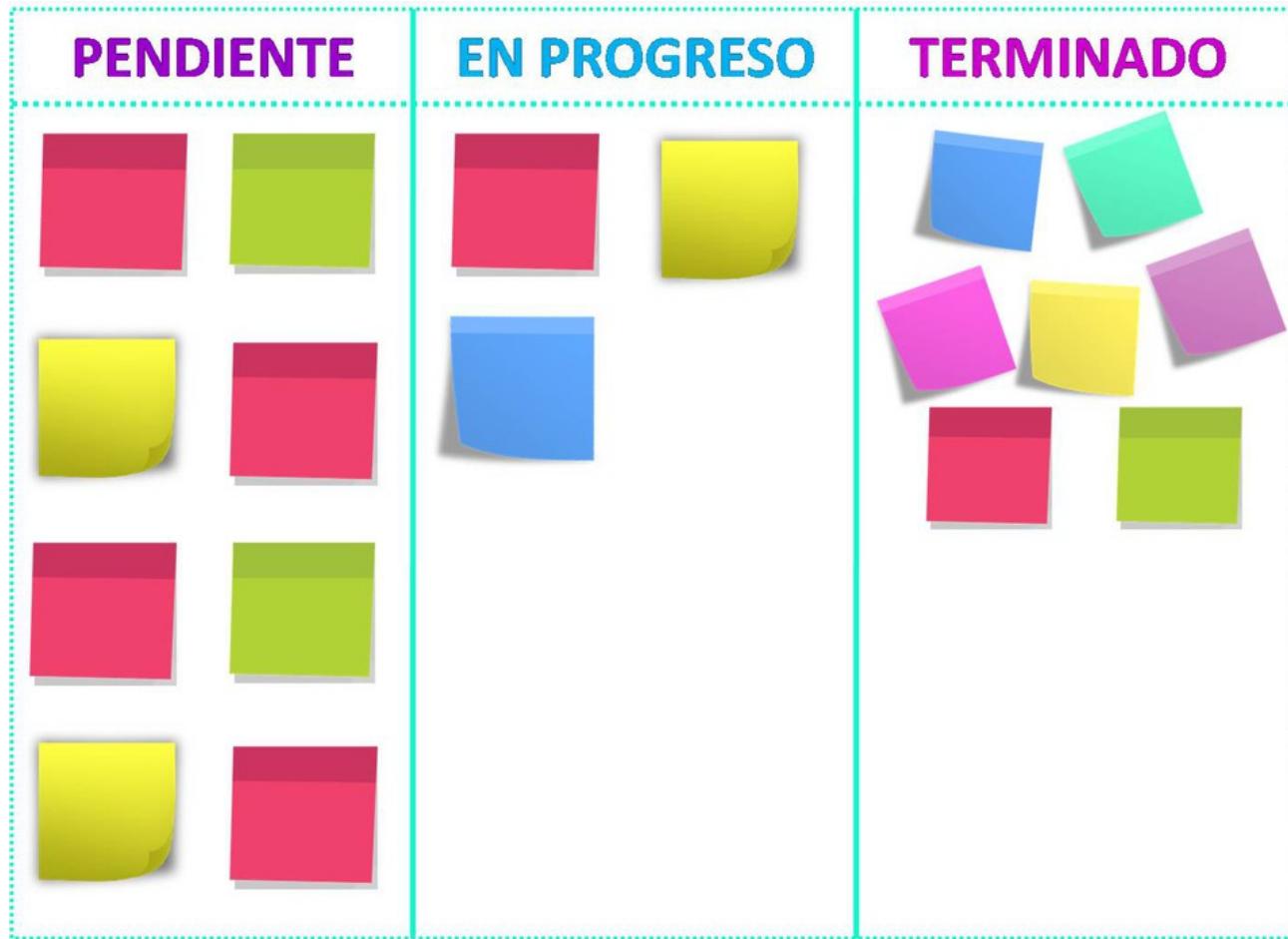
# Proceso Unificado Ágil

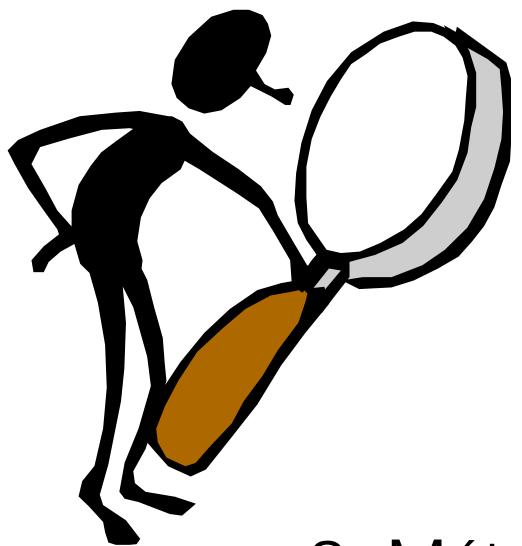
- Lanzamiento de versiones incrementales a lo largo del tiempo
  - Versiones de desarrollo
    - Se lanzan al final de cada iteración
  - Versiones de producción
    - Son versiones de desarrollo que han pasado procesos de aseguramiento de la calidad, prueba y despliegue
    - Se lanzan con menos frecuencia que las de desarrollo



# Kanban

**Tarjeta Visual:** Una tarjeta por tarea  
tarjetas Kanban,desplazar por cada una de las diversas etapas.





### 3. Métodos para la Ingeniería Web

# HDM

- HDM (*Hypermedia Design Model*) (Garzotto et al., 1993)
  - Uno de los primeros métodos desarrollados para la definición de la estructura y la interacción de las aplicaciones hipermediales
  - Se basa en la técnica de modelado conceptual Entidad/Relación
    - Extiende el concepto de entidad e introduce nuevas primitivas
      - Unidades (que se corresponden con el concepto de nodo)
      - Enlaces
  - Las entidades HDM tienen una estructura interna y semántica de navegación asociada
    - Especificación de cómo se puede llevar a cabo la navegación y de cómo se visualiza la información
  - Una entidad es una jerarquía de componentes
    - Los componentes están formados por unidades

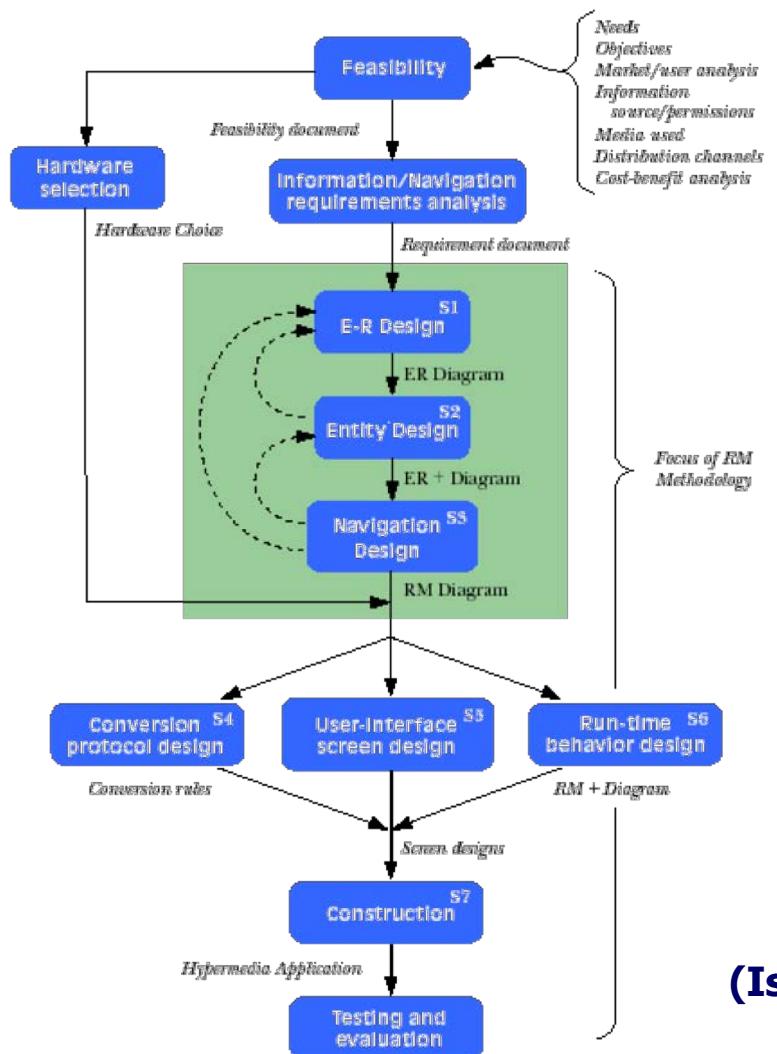
# HDM

- HDM (*Hypermedia Design Model*) (Garzotto et al., 1993)
  - Existen tres tipos de enlaces
    - Estructurales
      - Conectan componentes
    - De perspectiva
      - Conectan unidades
    - De aplicación
      - Conectan componentes y entidades
  - Existe otro tipo de entidades que dan acceso a las entidades de aplicación
    - Ofrecen puntos de entrada para comenzar la navegación
  - Para soportar el diseño de la presentación se definen
    - Ranuras (*slots*) - Una pieza atómica de información. Están compuestos de marcos
    - Marcos (*frames*) - Una unidad de presentación que se muestra al usuario

# RMM

- RMM (*Relationship Management Methodology*) (Isakowitz et al., 1995)
  - Se ocupa del diseño y construcción de aplicaciones hipermedia definiendo un proceso formado por siete pasos
    - Diseño entidad/relación
    - Diseño de vistas de información (*slices*)
    - Diseño de la navegación
    - Diseño de la interfaz de usuario
    - Diseño del protocolo de conversión
    - Diseño del comportamiento en ejecución
    - Construcción y prueba

# RMM



**(Isakowitz et al., 1995)**

# EORM

- EORM (*Enhanced Object Relationship Methodology*)  
(Lange, 1996)
  - Presenta un proceso iterativo que se basa en la riqueza del modelo objeto para representar relaciones entre objetos (enlaces) como objetos
    - Utiliza para ello notación OMT (Rumbaugh et al., 1991)
  - Este método se basa en tres *frameworks*
    - Clase
      - Consiste en definiciones reutilizables de clases
    - Composición
      - Consiste en definiciones reutilizables de clases de enlaces
    - Interfaz gráfica de usuario

# OOHDM

- OOHDM (*Object-Oriented Hypermedia Design Method*)  
(Schwabe y Rossi, 1995; Rossi, 1996)
  - Conlleva cinco actividades
    - Obtención de requisitos
    - Modelado conceptual
    - Diseño navegacional
    - Diseño de la interfaz abstracta
    - Implementación
  - Las actividades se llevan a cabo mediante un proceso incremental e iterativo, que hace uso de prototipos
    - Los modelos objetos se construyen en cada paso mejorando los de las iteraciones anteriores

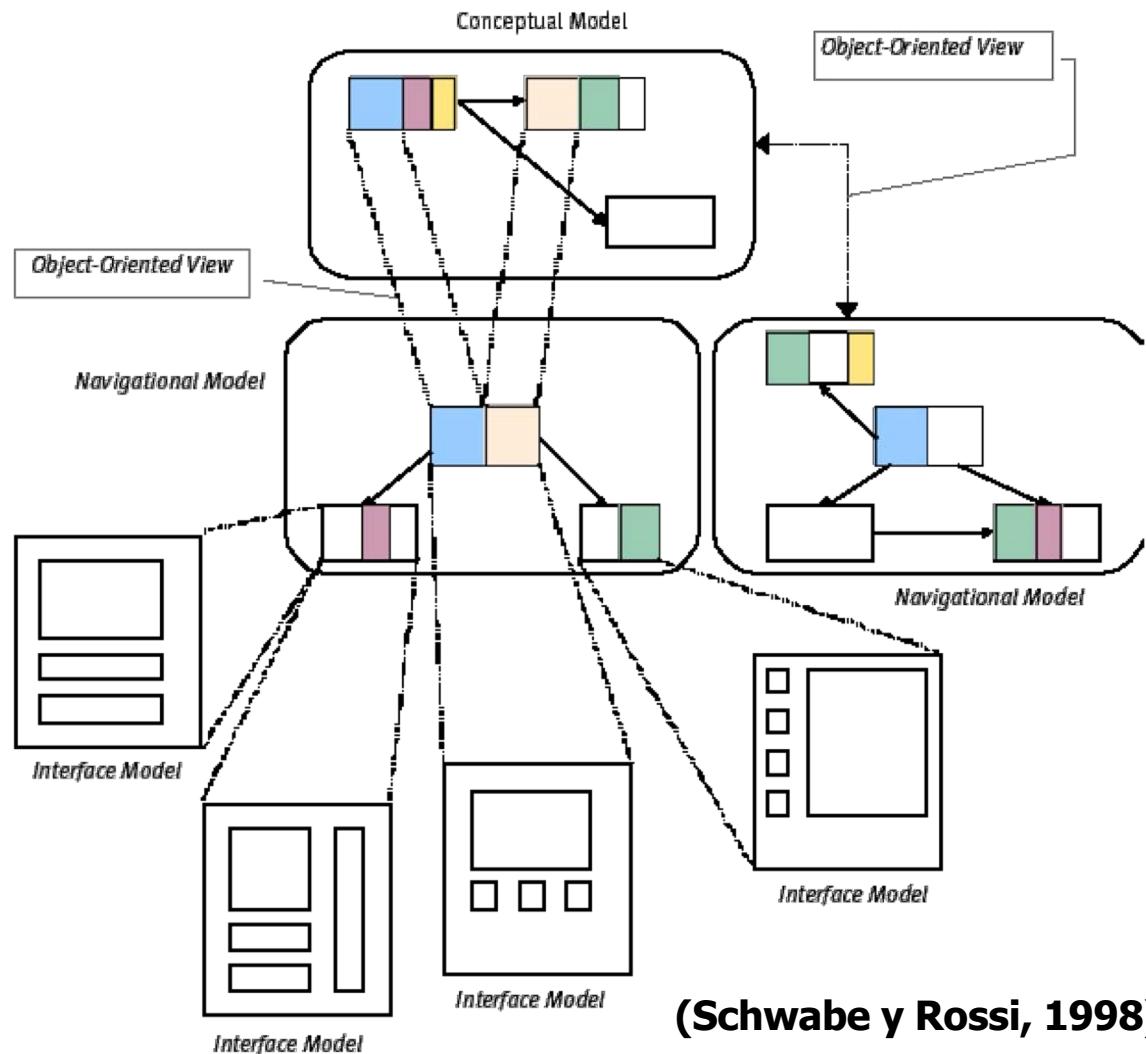
# OOHDM

- OOHDM (*Object-Oriented Hypermedia Design Method*)  
(Schwabe y Rossi, 1995; Rossi, 1996)
  - El modelado conceptual se lleva a cabo con un diagrama de clases
    - Primero utilizando notación OMT (Rumbaugh et al., 1991) y posteriormente notación UML (OMG, 2004)
    - Este método ve a una aplicación web como una vista del modelo conceptual
    - Las clases que definen las vistas son las clases navegacionales
    - Así, el modelo conceptual incluye dos tipos de objetos
      - Los que se perciben como nodos en el modelo navigacional
      - Los que ofrecen un soporte computacional a la aplicación web
        - Encapsulan comportamiento como algoritmos, acceso a bases de datos...
    - Este modelo puede servir como base a muchas aplicaciones y no incluye ninguna información específica de navegación
      - Esto es una clara diferencia con EORM (Lange, 1996)

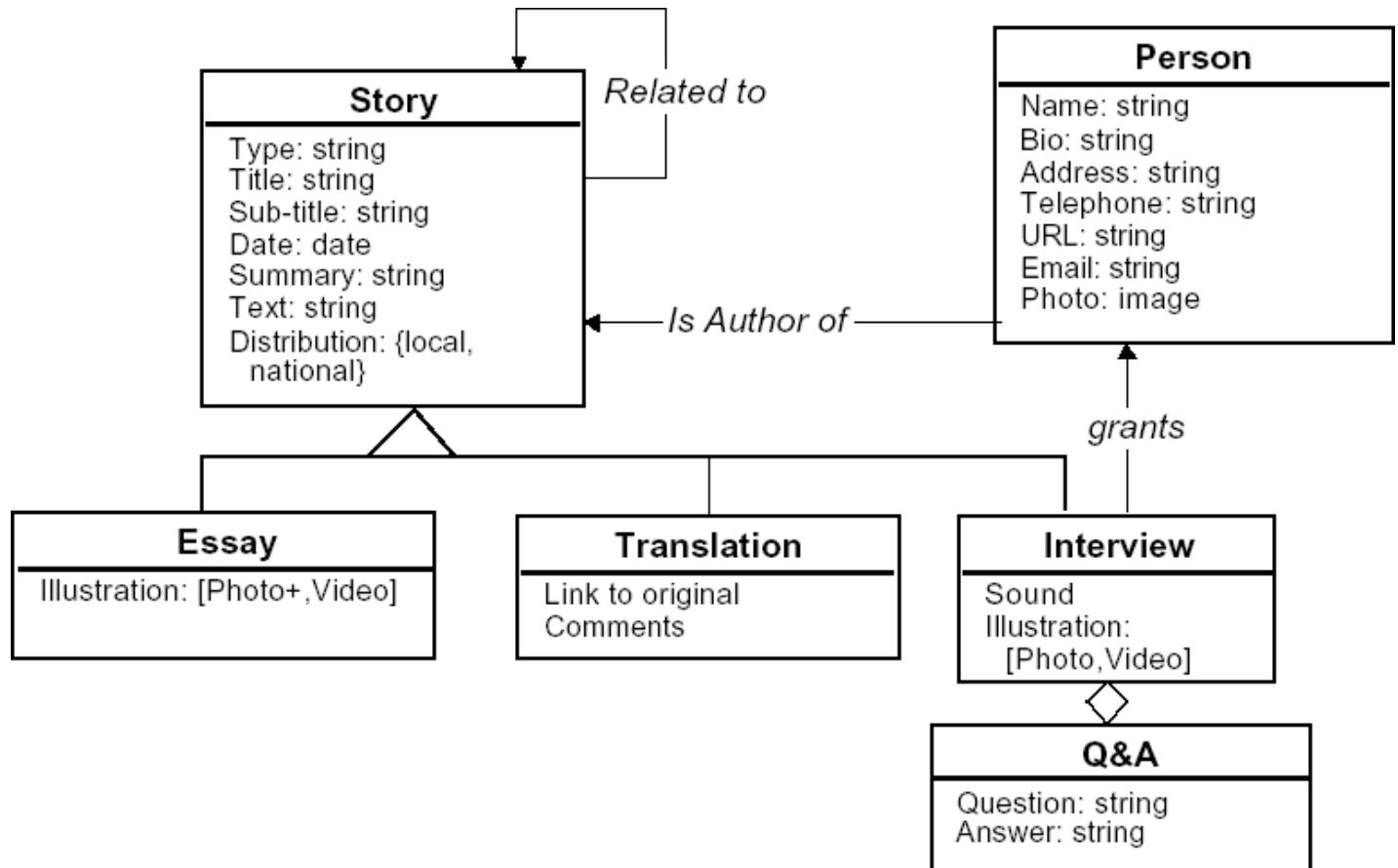
# OOHDM

- OOHDM (*Object-Oriented Hypermedia Design Method*)  
(Schwabe y Rossi, 1995; Rossi, 1996)
  - El concepto de contexto navegacional se introduce para describir la estructura de navegación
    - Permite diferentes agrupaciones de objetos navegacionales con el propósito de navegar por ellos en diferentes contextos
    - El acceso a estos elementos navegacionales se modela mediante estructuras de acceso, como índices por ejemplo
    - Se utiliza una notación propia para la representación del esquema del contexto navegacional
  - El modelo de interfaz abstracta es el resultado de especificar los objetos de interfaz que el usuario percibe
    - OOHDM utiliza ADVs (*Abstract Data Views*) para modelar los aspectos estáticos de la interfaz de usuario, utilizando diagramas de transición de estados para modelar los aspectos dinámicos

# OOHDM

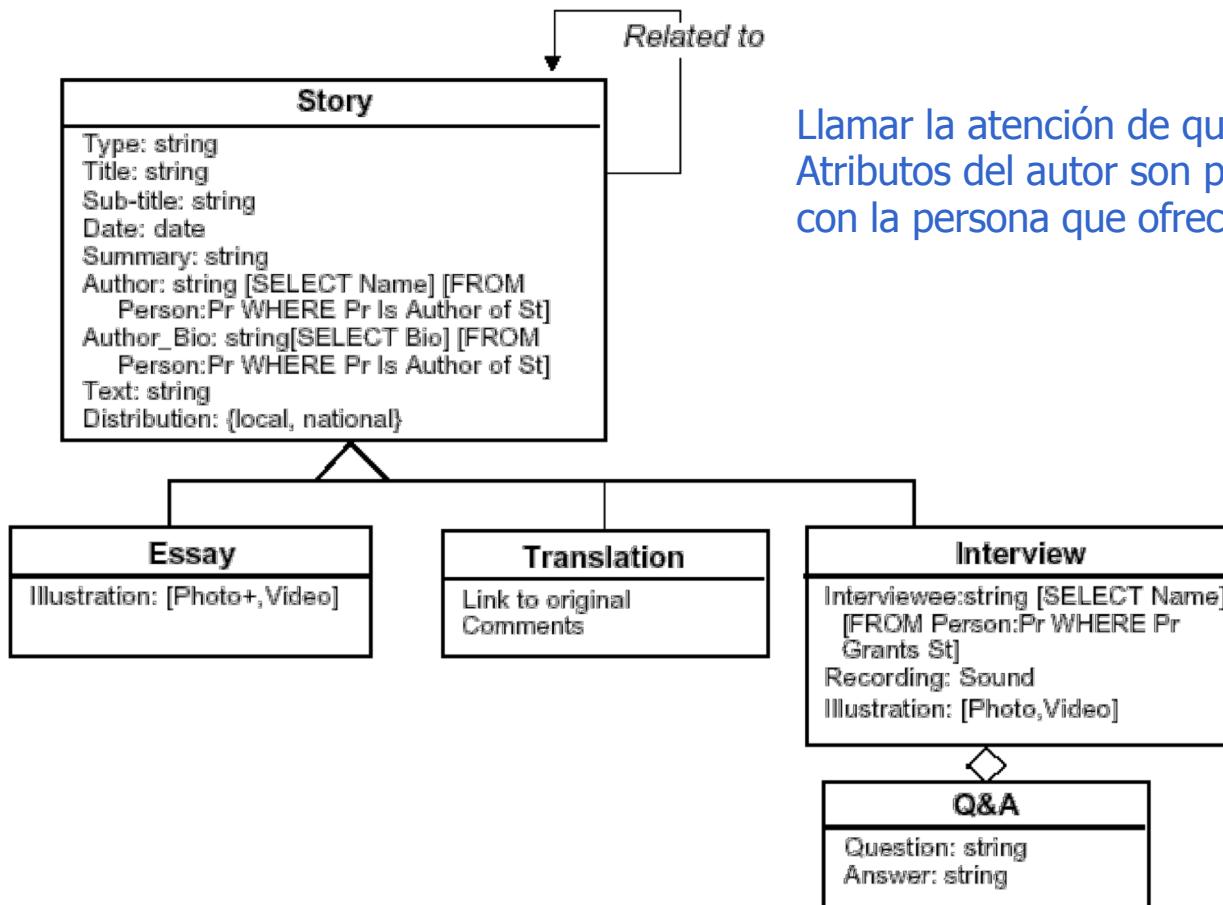


# OOHDM



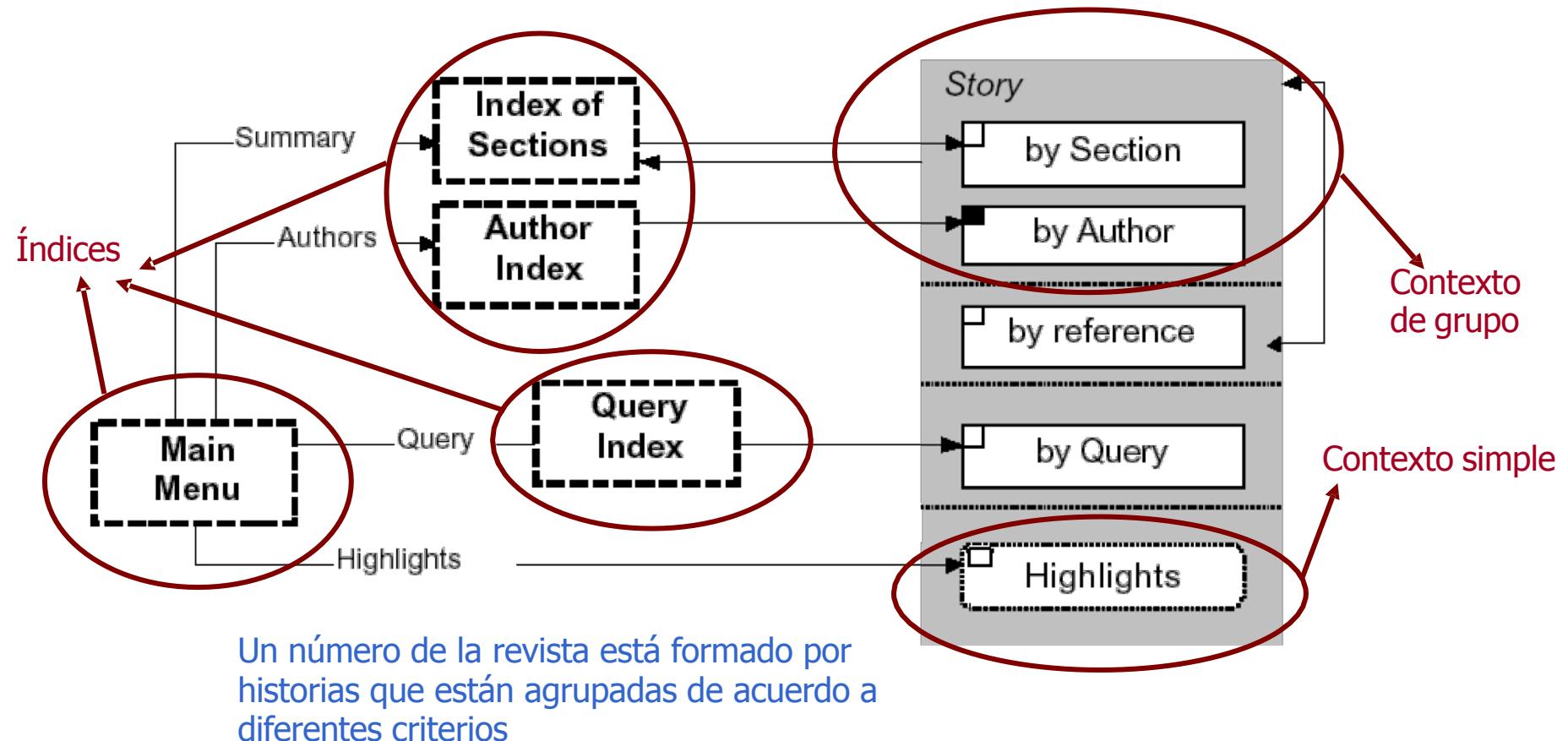
Modelo conceptual de una revista en línea (Schwabe y Rossi, 1998)

# OOHDM



Modelo navegacional de una revista en línea (Schwabe y Rossi, 1998)

# OOHDM



Esquema del contexto navegacional de una revista en línea (Schwabe y Rossi, 1998)

# OOWS

- OOWS (*Object-Oriented Approach for Web Solutions Modeling*) (Pastor et al., 2001a)
  - Método de desarrollo de aplicaciones web que extiende OO-Method (Pastor et al., 2001b)
    - Se introducen nuevas características navegacionales
  - Notación basada en UML (OMG, 2017)
  - Dos grandes fases
    - Especificación Conceptual
    - Generación Automática

# OO-HMethod

- OO-HMethod (Gómez et al., 2000; Cachero et al., 2000)
  - Método genérico para el desarrollo de la estructura semántica de las aplicaciones web
    - Se centra en las actividades globales (*authoring in the large*), esto es, en las clases y estructuras
    - Se despreocupa del contenido de los nodos de información (*authoring in the small*)
  - Extiende OO-Method (Pastor et al., 2001b)
    - Extiende la notación para expresar un modelo abstracto de interfaz de usuario
    - Se captura la información a que cada tipo de usuario (agente) puede acceder, así como los caminos de navegación entre vistas de información
  - El desarrollo de una aplicación web se aborda tanto a nivel conceptual como a nivel de ejecución
  - El modelo de navegación se captura en el NAD (*Navigation Access Diagram*)
  - La estructura y visualización de la interfaz se expresan en el APD (*Abstract Presentation Diagram*)

# OO-HMethod

- OO-HMethod (Gómez et al., 2000; Cachero et al., 2000)
  - Tanto el NAD como el APD capturan la información relevante gracias a un conjunto de patrones definidos en el Catálogo de patrones de OO-HMethod (Cachero et al., 2000)
  - La diferencia principal entre OO-HMethod y OOWS es que este último es una extensión plena de OO-Method
    - OOWS incluye completamente la especificación funcional, mientras que OO-HMethod sólo especifica la interfaz
    - Otra diferencia aparece en el modelo de navegación, concretamente en el concepto de nodo
      - OO-HMethod asocia diferentes diagramas de acceso a cada tipo de usuario, pero los nodos (clases de navegación) se limitan a presentar información de una sola clase
      - Los nodos en OOWS (contextos navegacionales) pueden trabajar con vistas de varias clases, mostrando la información apropiada para el usuario en cada momento

# SOHDM

- SOHDM (*Scenario-based Object-oriented Hypermedia Design Methodology*) (Lee et al., 1998)
  - Comprende seis fases
    - Análisis de dominio
    - Modelado de objetos
    - Diseño de vistas
    - Diseño de navegación
    - Diseño de la implementación
    - Construcción
  - Tiene similitudes con RMM (Isakowitz et al., 1995), EORM (Lange, 1996) y con OOHDM (Schwabe y Rossi, 1998)
  - Difiere en que usa escenarios
    - Los escenarios se definen en el análisis de dominio y se utilizan para el modelado de objetos

# SOHDM

- SOHDM (*Scenario-based Object-oriented Hypermedia Design Methodology*) (Lee et al., 1998)
  - El diseño de vistas consiste en determinar vistas que vienen generadas de un única clase o de asociaciones entre clases
    - Las vistas son similares a los contextos de OOHDM
    - Hay tres clases de vistas
      - Vistas base
        - Se genera de una única clase
      - Vistas de asociación
        - Se extrae de una relación de asociación
      - Vistas de colaboración
        - Proviene de una relación de colaboración
  - El diseño de la navegación emplea escenarios para determinar la estructura de los nodos
  - Las estructuras de acceso a los nodos (ASN - *Access Structure Nodes*), junto con las vistas, se denominan unidades de navegación
    - Las ASNs son similares a las primitivas de acceso de RMM
  - Este método define su propia notación gráfica

# WSDM

- WSDN (*Web Site Design Method*) (De Troyer y Leune, 1997)
  - Aproximación centrada en el usuario que define objetos de información basándose en los requisitos de información de los usuarios de una aplicación web
  - Conlleva tres fases principales
    - Modelado de usuario
    - Diseño conceptual
    - Diseño de la implementación
  - Modelado de usuario
    - Los usuarios de la aplicación web se identifican y se clasifican de acuerdo a sus intereses y preferencias de navegación
    - El punto de comienzo es la descripción del dominio, teniendo en cuenta las actividades de usuario
    - Cada perfil de usuario potencial es descubierto y se plasma en una clase

# WSDM

- WSDN (*Web Site Design Method*) (De Troyer y Leune, 1997)
  - Diseño conceptual
    - Dos fases
      - Modelado de objetos
        - El modelado de objetos conlleva tres pasos: modelado de objetos de negocio, modelado de objetos de usuario y modelado de objetos de perspectivas
      - Diseño navegacional
        - El modelo navegacional consiste en un número de caminos de navegación, una por cada perspectiva, expresando cómo los usuarios de un perspectiva particular pueden navegar a través de la información disponible
        - Se describe en término de componentes y enlaces
        - Se distinguen tres tipos de componentes: navegación, información y externos
        - Cada camino de navegación tiene tres capas: contexto, navegación e información
        - Este tipo de diseño de navegación provoca aplicaciones con una estructura muy jerárquica
    - Diseño de la implementación
      - Crea un *look&feel* eficiente y consistente con el modelo conceptual
      - Se dan pocas recomendaciones en este apartado
    - Combina una notación propia con OMT (Rumbaugh et al., 1991)

# HFPF

- HFPF (*Hypermedia Flexible Process Modeling*) (Olsina, 1998)
  - Abarca las siguientes vistas o perspectivas del proceso de desarrollo de una aplicación web
    - Vista funcional
      - Conjunto de fases y actividades para desarrollar tareas (encontrar usuarios, clases, casos de usos, etc.)
    - Vista metodológica
      - Define un conjunto de constructores de proceso para aplicarlos a diferentes actividades (análisis conducido por casos de uso, diseño conceptual y de navegación basado en OOHDML, etc.)
    - Vista de información
      - Para producir un conjunto de artefactos (modelo de navegación, modelo físico, etc.) que se requieren en diversas tareas
    - Vista de comportamiento
      - Representa la parte dinámica del modelo de proceso, tomando decisiones sobre la secuencia y la sincronización de tareas, iteraciones, incrementos, etc.
    - Vista organizacional
      - Define aspectos tales como roles, organización de equipos, mecanismos de comunicación, etc.

# HFPM

- HFPM (*Hypermedia Flexible Process Modeling*) (Olsina, 1998)
  - La lista de tareas que prescribe esta metodología para el desarrollo de una aplicación web son
    - Modelado de requisitos del *software*
    - Planificación del proyecto
    - Modelado conceptual
    - Modelado de la navegación
    - Modelado de la interfaz abstracta
    - Empleo de patrones de diseño
    - Captura y edición de los datos multimedia
    - Modelado físico
    - Validación y verificación
    - Empleo de criterios cognitivos
    - Aseguramiento de la calidad
    - Gestión y coordinación del proyecto
    - Documentación
  - Cubre todas las fases y actividades esenciales de un proyecto hipermedia
  - La notación se basa en OOHDML

# OO/Pattern

- OO/Pattern (Thomson et al., 1998)
  - Similar a HFPM (Olsina, 1998)
  - Propone utilizar diseño OO y patrones para el diseño de la navegación y de la presentación
  - Difiere de HFPM en que no cubre todo el ciclo de vida de una aplicación web
    - No se incluyen aspectos de gestión de proyectos, prueba y mantenimiento
  - La utilización de patrones presenta interesantes ventajas
    - Proceso bien definido
    - La documentación puede ser reutilizada
    - Se facilita el mantenimiento

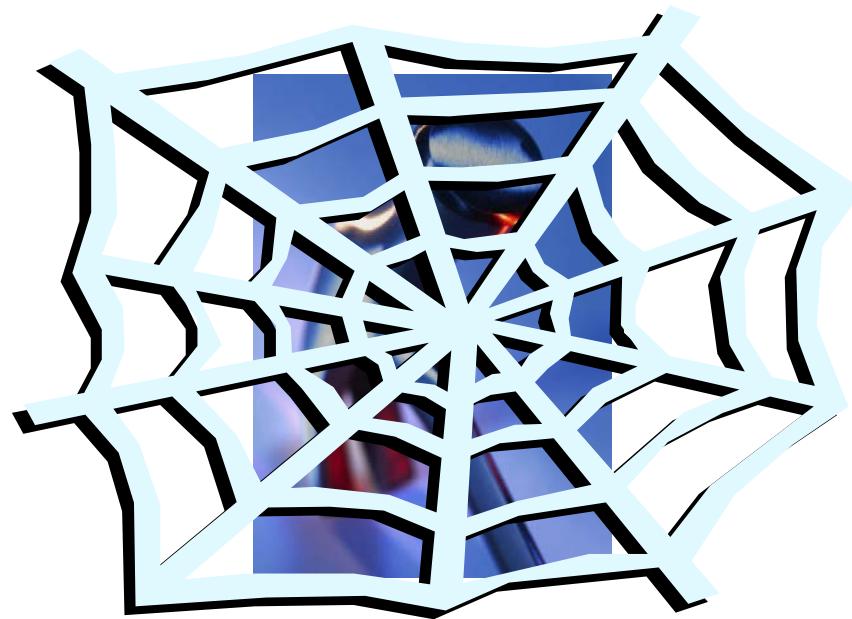
# OO/Pattern

- OO/Pattern (Thomson et al., 1998)
  - Presenta los siguientes pasos
    - Casos de uso
    - Diseño conceptual
    - Diseño de colaboraciones
    - Definición de clases
    - Diseño de la navegación
    - Implementación
  - Elementos innovadores
    - Análisis de casos de uso para diferentes tipos de usuarios
    - Diseño de colaboraciones basándose en los casos de uso definidos y en el diseño conceptual
    - Diseño de la navegación basada en patrones

# WAE

- WAE (*Web Application Extension for UML*) (Conallen, 1999)
  - Incluye estereotipos UML para el modelado de elementos arquitectónicos específicos de la Web
  - El proceso propuesto está basado en RUP (*Rational Unified Process*) (Kruchten, 2000)
  - Proceso centrado en la arquitectura y la implementación (incluye ingeniería inversa), pero ofrece poco soporte sistemático a la construcción de la estructura de navegación de la aplicación web, así como a sus aspectos de presentación

#### 4. OOWS



# Objetivos

**“Modelado conceptual de  
aplicaciones web...”**



# Objetivos

- Técnicas de **Modelado Conceptual** proporcionan un enfoque metodológico y sistemático a la especificación de aplicaciones tradicionales
- Los métodos de diseño orientados a objetos que utilizan técnicas de modelado conceptual **no** proporcionan primitivas para especificación de la navegación, presentación...
- ¿Cómo eliciar y representar la semántica navegacional en modelos conceptuales?
- Ampliar la etapa de **Modelado Conceptual** introduciendo los **Modelos de Navegación** y de **Presentación**

# Objetivo: Un método para la construcción aplicaciones web

The figure consists of three vertically stacked screenshots of the Amazon.com website, each with annotations in yellow boxes:

- Left screenshot (Navigation Capture):** Shows the left sidebar with categories like Books, Electronics, and Health & Beauty. A large black arrow points from the sidebar to the search bar on the main page. A yellow box contains the text "... permita capturar la navegación ...".
- Middle screenshot (Information Visualization):** Shows a search for "origami". A yellow box contains the text "... especificar búsquedas ...". Below it, another yellow box contains the text "... tratar la visualización de información ...".
- Right screenshot (Service Execution):** Shows a product page for "Math in Motion: Origami in the Classroom K-8". A yellow box contains the text "... y la ejecución de servicios ...".

# ¿Qué es OOWS?

- OOWS (*Object-Oriented Approach for Web Solutions Modeling*)  
(Pastor et al., 2001a)
- Una aproximación para definir **semántica de navegación** en modelos Orientados a Objeto
- **Ampliación de un Método OO** de producción de *software* “tradicional”, llamado OO-Method
- Utiliza la **notación UML** (adaptada)
- Extiende el **proceso de desarrollo** de sistemas *software* propuesto por OO- Method (Pastor et al., 2001b)
- Define **primitivas navegacionales y de presentación de información** integradas en el Modelado Conceptual OO-Method

# OOWS. Proceso de desarrollo

- El método OOWS extiende el proceso de desarrollo definido por OO-Method
- Incluye nuevas etapas donde se captan los requisitos de navegación y presentación de información
- Dos grandes fases
  - 1. **Especificación Conceptual**, se construye una especificación completa del sistema, **Esquema Conceptual**
  - 2. **Generación Automática**, a partir de la especificación se construye una solución *software* funcionalmente equivalente basada en patrones de traducción

# OOWS. Proceso de desarrollo

## Modelado Conceptual

- La fase de Modelado Conceptual abarca
  - 1. **Especificación de Requisitos**
    - Usa notación UML (Casos de Uso)
    - Recoge
      - La funcionalidad que debe proporcionar el sistema
      - Los diferentes tipos de usuarios que pueden interactuar con el sistema
      - La asociación de usuarios-funcionalidad
    - Sirve como base para la construcción del Esquema Conceptual

# OOWS. Proceso de desarrollo

## Modelado Conceptual

- **2. Modelado Conceptual**

Se construyen a partir de la especificación de requisitos los modelos

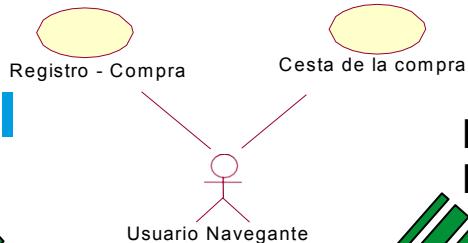
- **M. Objetos:** Define la estructura y las relaciones estáticas entre clases identificadas en el dominio del problema
- **M. Dinámico:** Se describen las posibles secuencias de servicios y los aspectos relacionados con la comunicación interobjetual
- **M. Funcional:** Captura la semántica asociada a los cambios de estado entre los objetos motivados por la ocurrencia de eventos o servicios
- **M. Navegación:** Define la semántica navegacional asociada las clases de los objetos del modelo
- **M. Presentación:** Captura los requisitos básicos de presentación de información, orientado a ambientes web. Está fuertemente basado en el modelo de navegación y permite definir la estructura lógica de presentación de los objetos navegacionales

OO-Method

Extensión  
OOWS

# OOWS. Proceso de desarrollo

## Especificación Conceptual

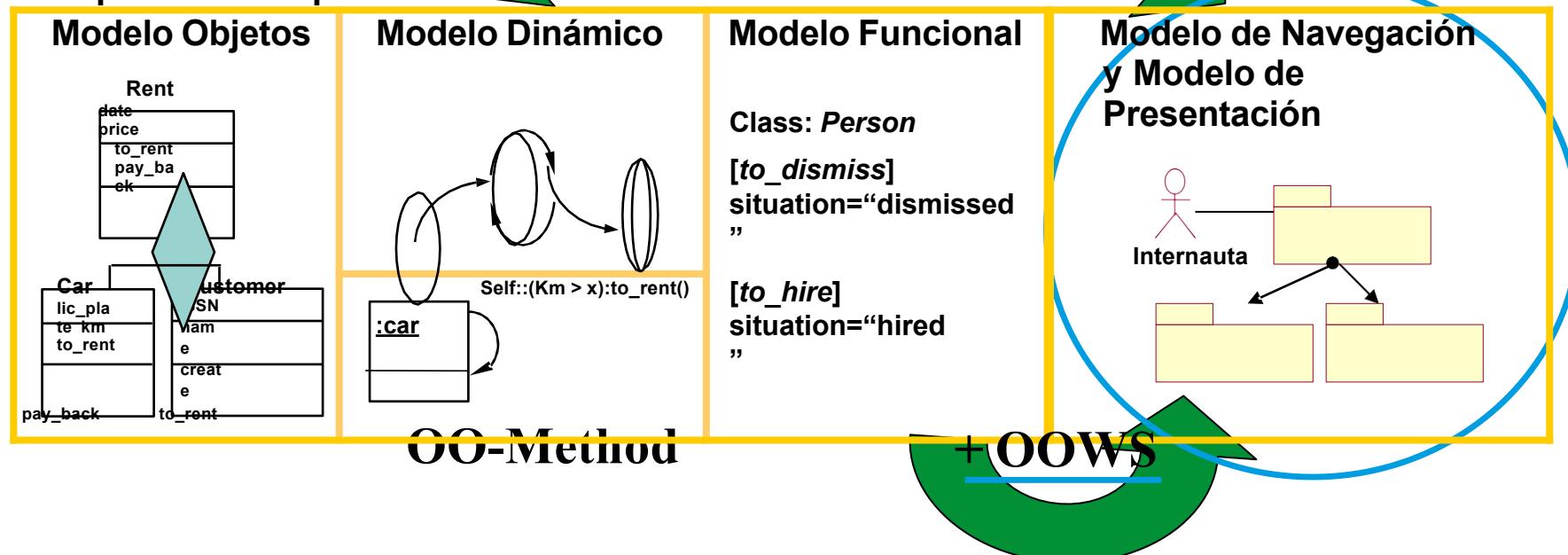


1

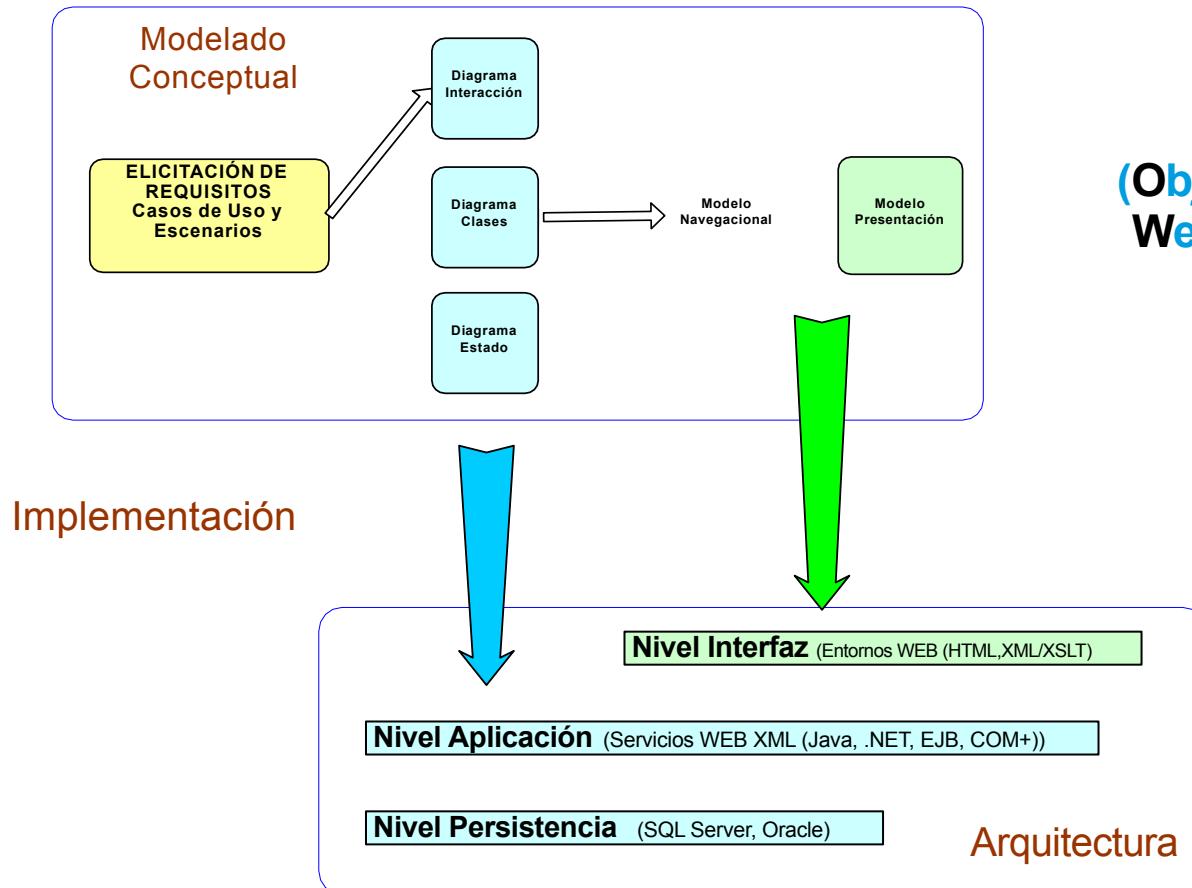
## Especificación de Requisitos

## 2 Construcción del Esquema Conceptual

con expresividad navegacional y de presentación de información

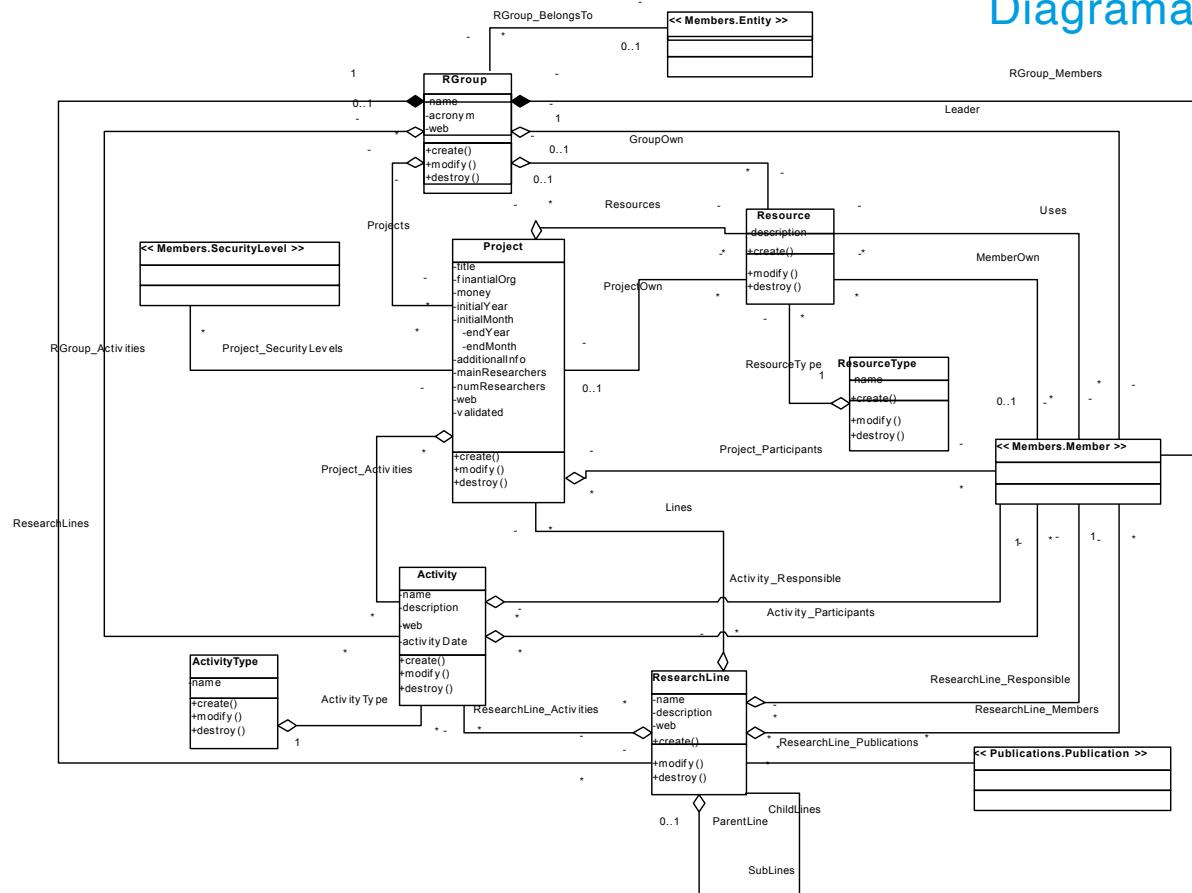


# Propuesta Metodológica



# Esquema conceptual

Diagrama de Clases



# Modelo de navegación

- Especificación de las **características navegacionales** de una aplicación web
  - En base a un Modelo de Objetos y a los requisitos de navegación
- Utiliza una notación basada en UML
- Se construye a partir de las **primitivas de abstracción** navegacionales
  - Representación de la Navegación
  - Especificación de Búsquedas
  - Tratamiento de la visualización de la información (presentación)
  - Ejecución de Servicios
  - Personalización de información de los distintos usuarios
- Integrado con las restantes vistas del esquema conceptual
- Define y estructura el **acceso** de los diferentes **usuarios** con el **sistema**, en función de su objetivo
  - Considera el punto de vista de cada **perfil de usuario** identificado previamente en el Modelo de Objetos

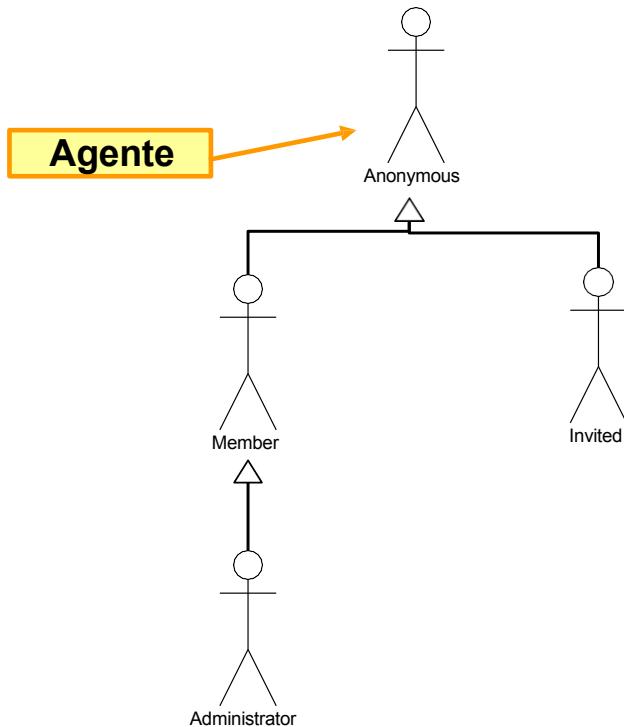
# Modelo de navegación

- Construye un **grafo navegacional** asociado a cada usuario formado por
  - **Nodos**
    - Unidades de interacción que proporcionan acceso a datos y funcionalidad relevante para el usuario
  - **Enlaces**
    - Relación de alcance entre nodos para conseguir cierto objetivo

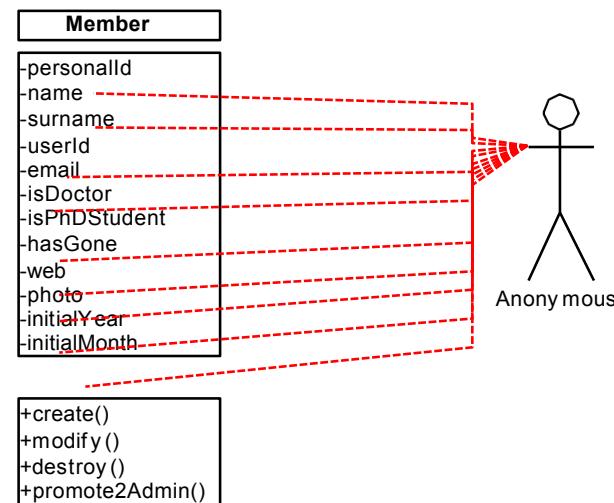
**Navegación** es el cambio de nodo conceptual al activar un enlace navegacional

# Modelado de navegación

## Diagrama de Agentes



## Visibilidad



# Modelo de Navegación

- **Primitivas de Abstracción Básicas**

- **Mapa Navegacional**

- “Visión Global de una aplicación web según un perfil de usuario”

- **Contexto de Navegación**

- “Conjuntos de objetos que el usuario irá navegar”

- **Vínculo de Navegación**

- “Indica la navegación entre contextos de navegación”

- **Clase Navegacional**

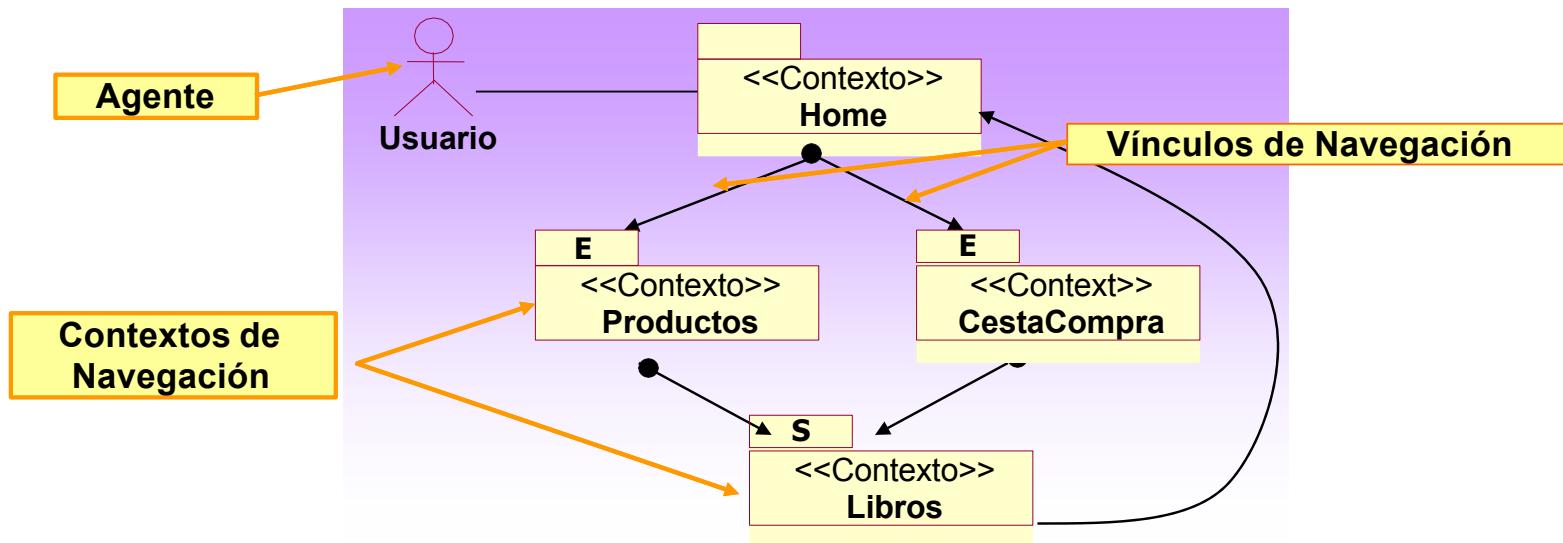
- “Contenido de la información por el cual los usuarios navegarán”

- **Relaciones**

- “Maneras de navegar para acceder al contenido de la información”

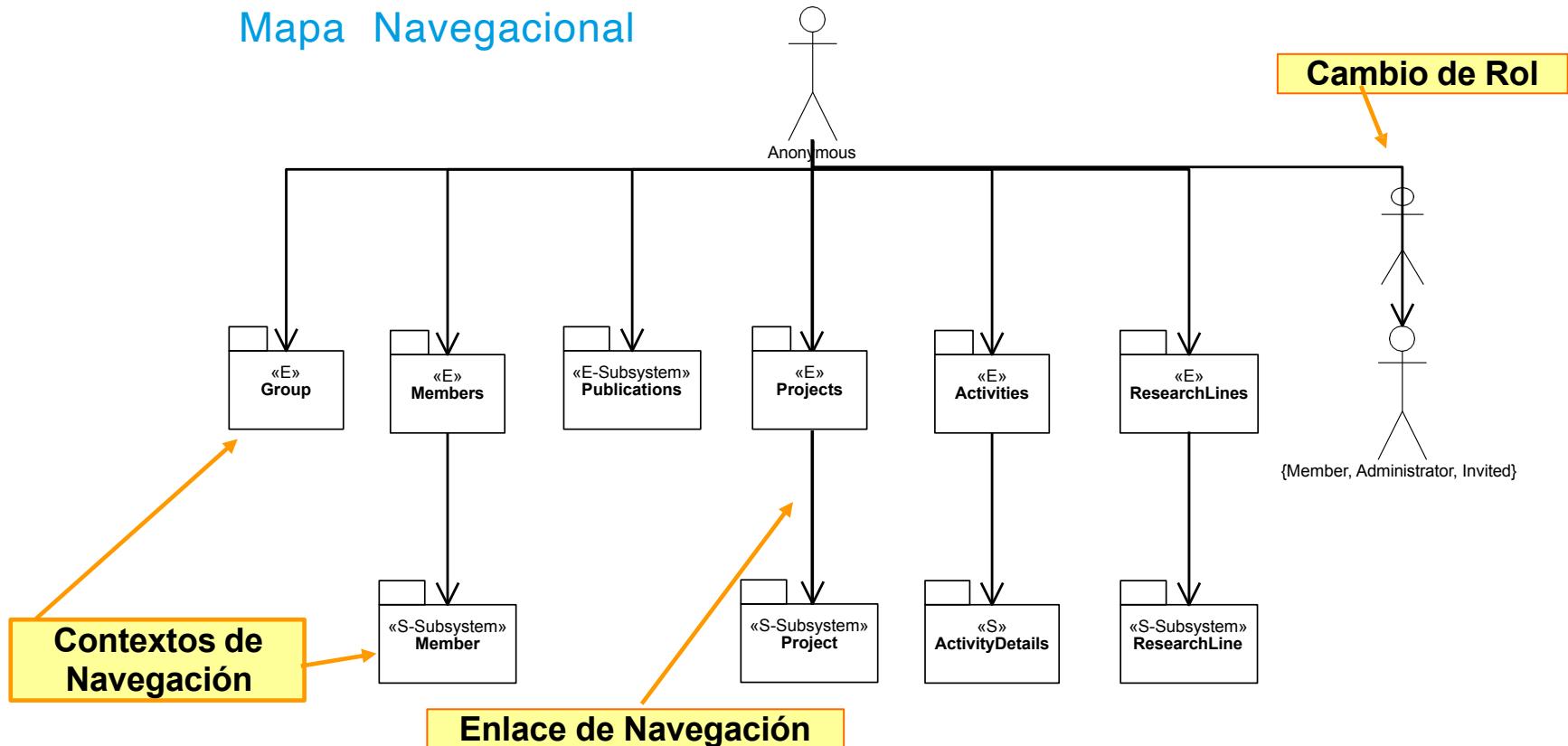
# Primitivas de abstracción. Mapa de navegación

- El **Modelo de Navegación** está compuesto por un conjunto de **mapas de navegación**
  - Define el sitio web
- Asociado a un agente del **Modelo Conceptual**
  - Visión global del sistema para cada tipo de usuario
- Grafo Navegacional formado por
  - Contextos de Navegación (nodos)
  - Vínculos Navegacionales (arcos)



# Mapa de navegación

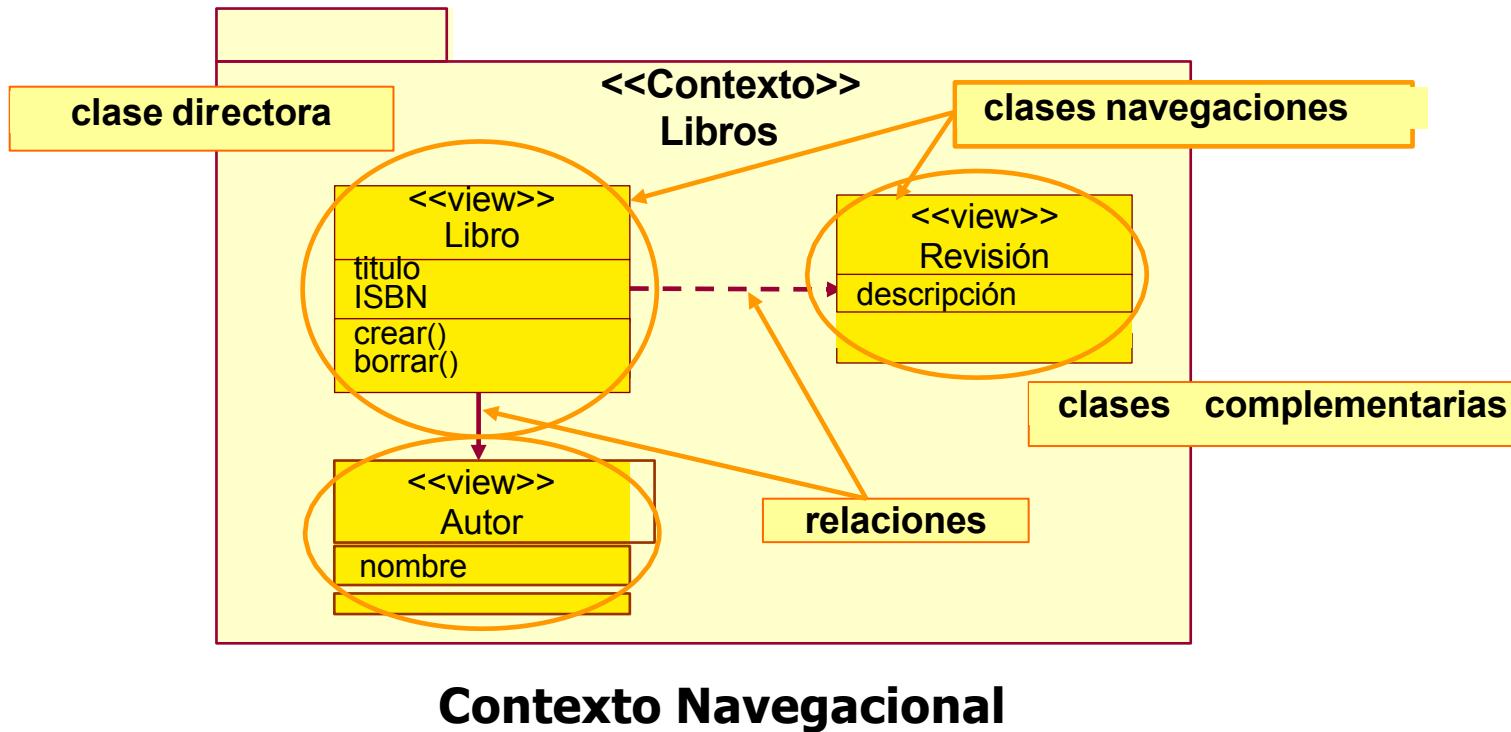
Mapa Navegacional



# Primitivas de abstracción. Contexto Navegacional

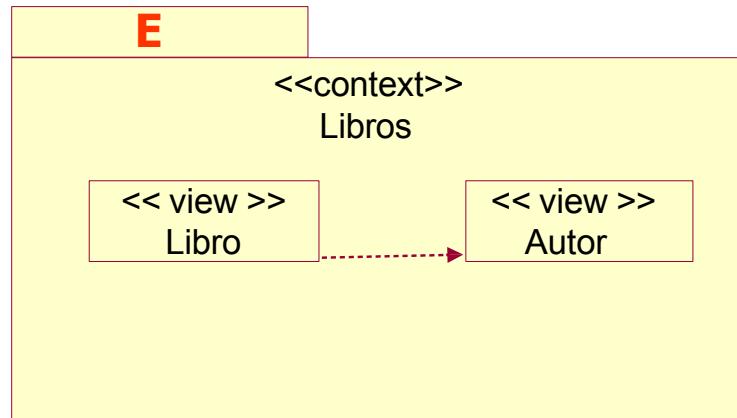
- **Unidad de Interacción Abstracta** básica con el usuario
- Representa una **vista parcial** del **sistema** adecuada para una determinada actividad
  - Es el punto de vista que un usuario tiene de un **subconjunto** del modelo de objetos
- Proporciona **acceso a datos y funcionalidad** asociados con el **usuario** propietario del mapa
- Está compuesto por
  - **Clases navegacionales**: Recuperan información del sistema
  - **Relaciones navegacionales**: Complementan la información de las clases navegacionales
- Gráficamente es un paquete UML estereotipado con la palabra reservada «**context**»

# Primitivas de abstracción. Contexto Navegacional



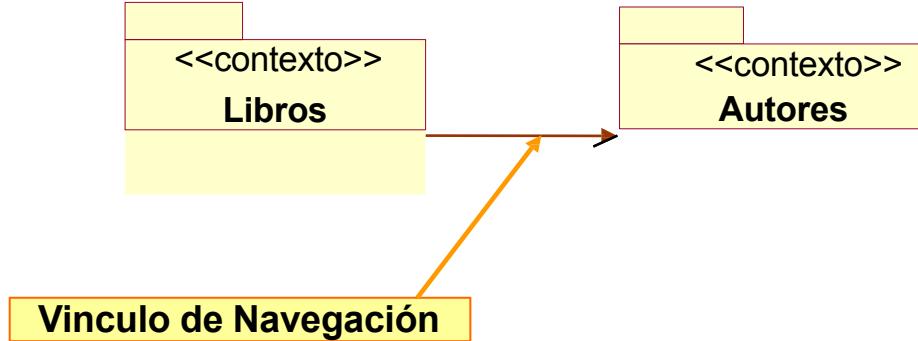
# Primitivas de abstracción. Contexto Navegacional

- Los contextos tienen un **carácter navegacional** que permite **estructurar** la navegación por el sistema
- El carácter de los contextos pueden ser
  - **Secuencia:** Sólo son accesibles siguiendo uno de los caminos de navegación especificados
  - **Exploración:** Son accesibles desde cualquier ubicación en la aplicación



# Primitivas de abstracción. Vínculo Navegacional

- Define una **relación de alcance** (navegación) entre Contextos de Navegación
- Definido implícitamente a partir de las **relaciones navegacionales** definidas dentro de los contextos y por el **carácter de los contextos** (de exploración o de secuencia)



# Modelo de Navegación. Ejemplo de mapa navegacional

**Contextos de Navegación**

**Vínculos de Navegación**

**Contextos de Navegación**

The diagram illustrates the navigation model on Amazon.com through three screenshots:

- Screenshot 1:** Shows the homepage with a red circle highlighting the main navigation bar (Welcome, Store Directory, Books, Music, DVD, Electronics, Tools & Hardware, Health & Beauty, Toys & Games) and the sidebar (BROWSE section).
- Screenshot 2:** Shows a search result page for "origami" with a red circle highlighting the search bar and the main navigation bar.
- Screenshot 3:** Shows a detailed product listing for "origami" with a red circle highlighting the main navigation bar.

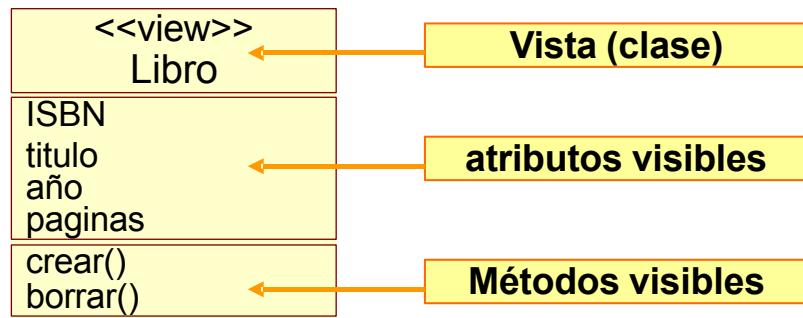
Arrows indicate the flow of navigation links between these different contexts on the site.

73

# Primitivas de abstracción.

## Clase Navegacional

- Proyecciones de **visibilidad** sobre clases existentes en el Modelo de Objetos con respecto a
  - **Atributos**: Datos del sistema visibles que por el usuario
  - **Servicios**: Funcionalidad ejecutable por el usuario
- Gráficamente son clases UML estereotipadas con la palabra reservada « **view** »

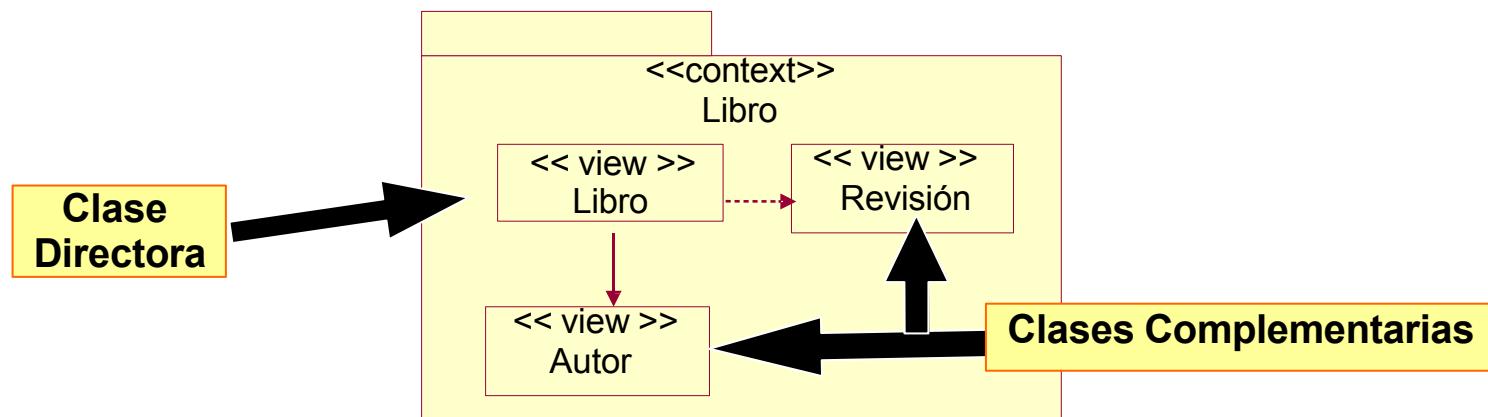


# Primitivas de abstracción.

## Clase Navegacional

- Existen de dos tipos

- **Clase Directora:** Es la clase principal de un contexto. Existe una única por contexto (obligatoria). El contexto se centra en presentar información y funcionalidad de esta clase
- **Clases Complementarias:** Su utilidad es complementar la información de la clase directora. Pueden aparecer varias por contexto (no son obligatorias)



# Primitivas de abstracción.

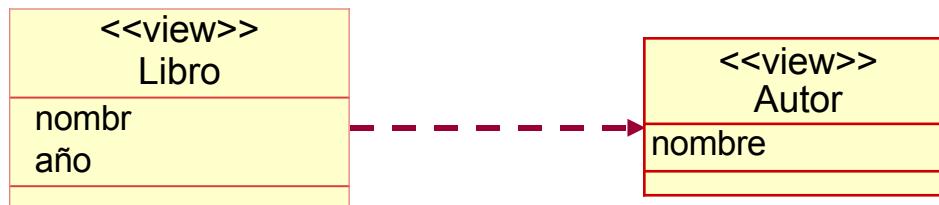
## Relación Navegacional

- Es una **relación binaria unidireccional** existente entre dos clases de un contexto
- Se define sobre una relación **agregación o herencia** entre dos clases del **Modelo de Objetos**
- **Complementa la información** sobre la clase de la cual parte la relación, recuperando la población relacionada
- Dos tipos
  - Relaciones de Dependencia Contextual
  - Relaciones de Contexto

# Primitivas de abstracción. Relación Navegacional

- **Relación de Dependencia Contextual**

- Indica la existencia de una relación entre dos clases de un contexto, pero no define una semántica navegacional entre ellas
- Complementa la clase navegacional origen con su población relacionada
  - Indica una recuperación de información relacionada de las instancias de la clase complementaria
- Gráficamente se representa mediante una línea discontinua



En este caso, sólo se recuperará información de los libros y de sus autores (utilizando la relación de agregación existente en el modelo) pero no se proporcionará un enlace con otro contexto

# Primitivas de abstracción.

## Relación Navegacional

- **Relación de Contexto**

- Complementa la clase navegacional origen con su población relacionada
- Define un vínculo navegacional entre contextos, indicando la dirección de navegación
- Implica necesariamente la existencia de un contexto navegacional (destino) en el que la clase directora es la clase destino de la relación
- Gráficamente se representa mediante una línea continua



Se verá información de los libros y de sus autores (utilizando la relación de agregación existente en el modelo) y además se permitirá alcanzar el contexto Autores

# Primitivas de abstracción. Relación Navegacional



- Las relaciones navegacionales poseen atributos adicionales
  - **Atributo de contexto:** Contenido de navegación destino del enlace
  - **Atributo de enlace:** Atributo que se usará en la conexión con la clase destino. Es opcional
  - **Atributo de rol:** Nombre de la relación estructural en el Modelo de Objetos a la que se refiere la relación. Sólo es necesario en caso de ambigüedad<sup>1</sup>

<sup>1</sup> Cuando exista más de una relación entre dos clases en el Modelo de Objetos

# Modelo de Navegación. Ejemplo de relación de contexto

amazon.com

VIEW CART | WISH LIST | YOUR ACCOUNT | HELP

WELCOME STORE DIRECTORY BOOKS MUSIC DVD ELECTRONICS TOOLS & HARDWARE HEALTH & BEAUTY TOYS & GAMES

SEARCH BROWSE SUBJECTS BESTSELLERS NEW & FUTURE RELEASES COMPUTERS & INTERNET E-BOOKS & DOCS BARGAIN BOOKS RARE & USED

SEARCH Books GO!

Math in Motion: Origami in the Classroom by Barbara Pearl

Our Price: \$24.95

Availability: Usually ships within 24 h

See larger photo

Buy & Sell Used Items

Get it for less! Order it used

I have one to sell! Sell yours here

Spiral-bound

Crane Publishing; ISBN: 0964792435

Amazon.com Sales Rank: 49,525

Average Customer Rating: ★★★★☆ Based on 13 reviews. Write a review.

Rate this item to get personal recommendations

Customers who bought this book also bought:

Definición de navegación al Contexto "Autores"

BUY FROM AMAZON.COM

Add to Shopping Cart (you can always remove it later)

<<Context>>

Libros

<<view>> Libro

ISBN  
Título  
Fotografía  
Precio  
Disponibilidad  
Índice\_ventas

[Autores]

<<view>> Autor

nombre

Información de la clase complementaria "Autor"

Información de la clase directora "Libro"

# Construcción del Modelo de Navegación

**INPUT: Modelo Conceptual + Requisitos Navegación**

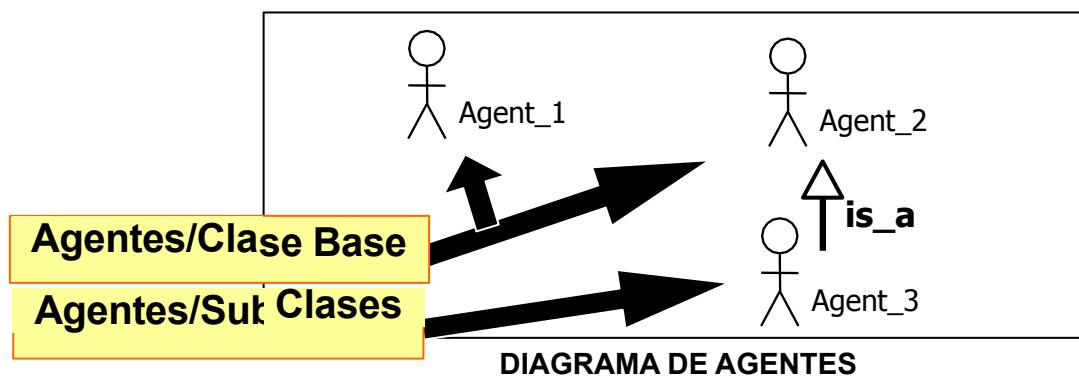
**OUTPUT: Modelo de Navegación**

1. Identificación de los **agentes** del Sistema y sus relaciones (herencia) entre sí, en base al **Modelo de Objetos** definido
2. Construcción de los **mapas navegacionales** para cada agente detectado, según los **requisitos de navegación**. Se pueden reutilizar contextos por relaciones entre agentes

# Construcción del Modelo de Navegación

## 1. Identificación de Agentes

- Buscar en el Modelo de Objetos los agentes del sistema
- Detectar las relaciones entre los agentes (reutilización navegacional)
  - Construir los **árboles de agentes**, donde aparece cada agente y sus relaciones con los demás
  - Estos árboles están compuestos de
    - Agentes/Clases Base
    - Agentes/SubClases



# Construcción del Modelo de Navegación

## 2. Construcción de los Mapas

- El **Modelo de Navegación** está compuesto por los **Mapas Navegacionales** definidos
- Se pueden seguir dos **estrategias** para la **construcción** del Modelo de Navegación
  - **Top-Down**: Para cada agente, se da un boceto del mapa navegacional y siguiendo su estructura se refina cada contexto declarado
  - **Bottom-Up**: Para cada agente, se construyen las piezas básicas (contextos) y a partir de éstas se obtiene su mapa navegacional

# Construcción del Modelo de Navegación

## 2. Construcción de los Mapas

- Estrategia **Top-Down**
  - Para cada Agente
    - Se define un mapa de navegación abstracto
      - Se declaran los contextos navegacionales que existirán y las relaciones de alcance entre ellos
      - Se hereda el Mapa de Navegación del agente/Clase Base si el actual es un agente/SubClase, se modifica y extiende
    - Se refina cada contexto en base al mapa de navegación propuesto

El Mapa de Navegación se construye de manera explícita y se va refinando a medida que se construyen los contextos

# Construcción del Modelo de Navegación

## 2. Construcción de los Mapas

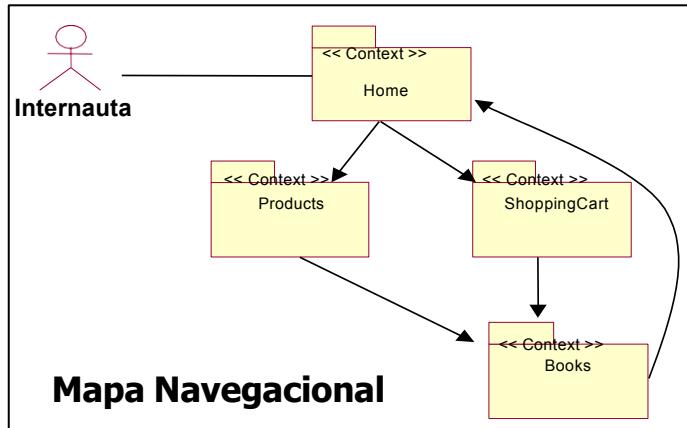
- **Estrategia Bottom-Up**
  - **Para cada Agente**
    - Construir los contextos de navegación que capturen los requisitos de navegación detectados (en la especificación de requisitos del sistema)
  - **Para los Agente/SubClase**
    - Se hereda el Mapa Navegacional de su clase base y se modifica y extiende

El **Mapa de Navegación** para cada agente se construye automáticamente a partir de los contextos especificados

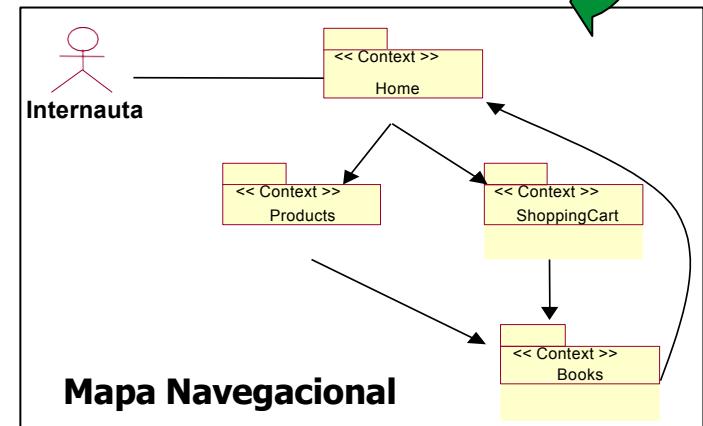
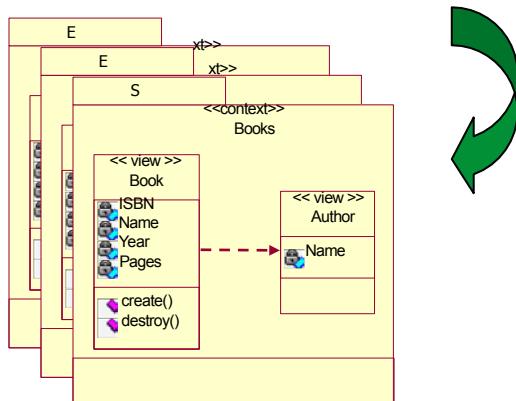
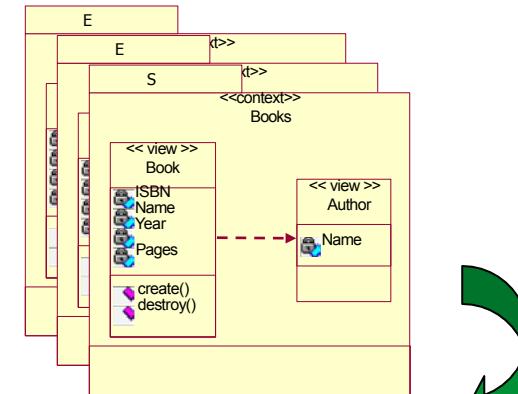
# Construcción del Modelo de Navegación

## 2. Construcción de los Mapas

### Estrategia Top-Down



### Estrategia Bottom-Up



# Modelo de Presentación

- Tras la especificación del Modelo de Navegación se construye el Modelo de Presentación
- Este modelo recoge la **semántica de presentación de información** del sistema
- Se basa en definir el **modo de presentación asociado** a cada **UIA** (Unidad de Interacción Abstracta) definida por el Modelo de Navegación
- Asocia **patrones de presentación** a los elementos que aparecen en estos nodos navegacionales

# Modelo de Presentación. Patrones de presentación

- **Patrón de Presentación**

- Define la estructura lógica de presentación de información a la población a que se aplica
- Se puede aplicar a
  - Clase Directora
  - Relaciones Navegacionales
- Cuatro tipos, en función de las cardinalidades y el tipo de las relaciones interobjetuales

- Registro
- Tabular

Para relaciones “1 a 1”

- Maestro-Detalle
- Árbol

Recursivamente, el detalle ha de tener un tipo

Para relaciones “1 a muchos” o “muchos a muchos”

Indicado también para relaciones reflexivas

# Modelo de Presentación. Patrones de presentación

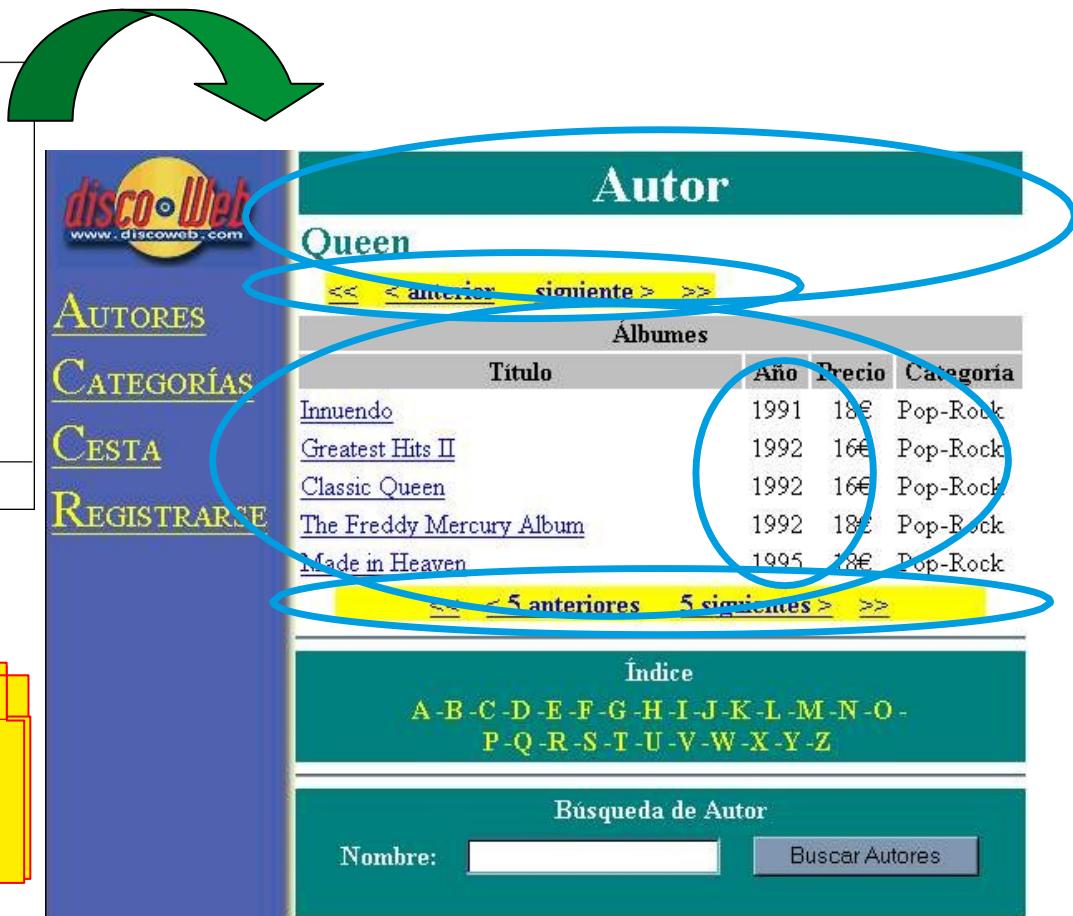
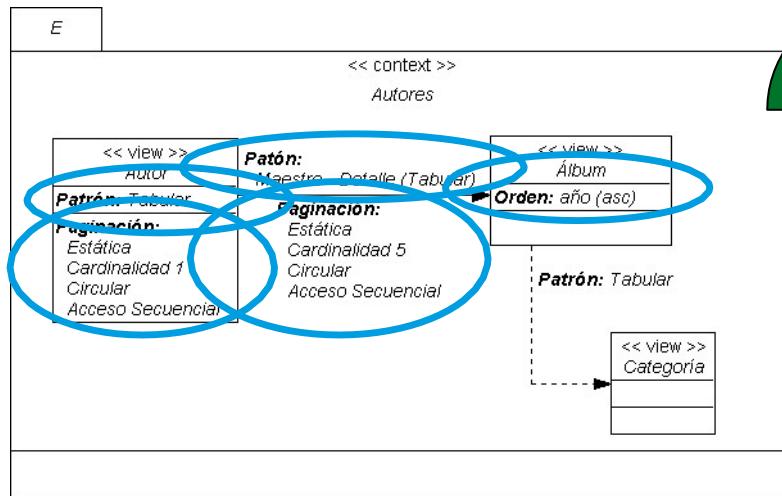
- Patrón de **Criterio de Ordenación**
  - Permite definir una **ordenación de la población** de una clase atendiendo a un criterio
  - Este criterio deberá estar en **función de propiedades** (atributos) de alguna **clase** del contexto
  - Se puede aplicar a
    - Clases Navegacionales, indicando cómo se recuperarán las instancias de estas clases
    - Estructuras de Acceso y Mecanismos de Búsqueda, para ordenar los resultados obtenidos
  - Existen de dos tipos: **Ascendente y Descendente**
  - En caso de especificación de **varios atributos**, la **ordenación es jerárquica**

# Modelo de Presentación. Patrones de presentación

- Patrón de **Paginación**
  - Define un *scrolling* de información, creando **bloques lógicos** en los que las instancias son “troceadas”
  - Se especifica una **cardinalidad**, o número de instancias a recuperar
  - Puede ser **estática** o **dinámica**, en función de si el usuario puede o no modificar la cardinalidad
  - Existen dos tipos
    - De **acceso secuencial**, cuando desde un bloque lógico sólo se puede ir al siguiente, al anterior, al primero o al último
    - De **acceso aleatorio**, cuando desde un bloque lógico se puede acceder directamente a cualquier otro
  - Se puede definir como **circular**, indicando que el siguiente bloque lógico al último es el primero y viceversa
  - Se aplica a
    - A la **clase directora**: Permite restringir el número de instancias de la clase principal que se recuperarán
    - A las **relaciones navegacionales**: Restringiendo el número de instancias de objetos relacionados que se recuperarán

# Modelo de Presentación. Patrones de presentación

## Ejemplo



Criterio de Ordenación Ascendente  
Paginación aplicada a una relación  
navegacional. Se recuperan objetos  
secuencialmente en grupos de 5

# UWE

- UWE (*UML-based Web Engineering*) (Koch, 2000; Hennicker y Koch, 2000)
  - Soporta desarrollo de aplicaciones web prestando especial atención en sistematización y personalización (sistemas adaptativos)
  - Consiste en una notación y en un método
    - La notación se basa en UML (OMG, 2004)
      - Para aplicaciones web en general y para aplicaciones adaptativas en particular
    - El método consta de seis modelos
      - Modelo de casos de uso para capturar los requisitos del sistema
      - Modelo conceptual para el contenido (modelo del dominio)
      - Modelo de usuario
      - Modelo de navegación que incluye modelos estáticos y dinámicos
        - Modelo de estructura de presentación, modelo de flujo de presentación, modelo abstracto de interfaz de usuario y modelo de ciclo de vida del objeto
      - Modelo de adaptación

# UWE

Modelado de requisitos:

Casos de uso:

Estereotipos Browsing and processing

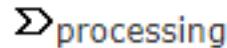
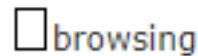
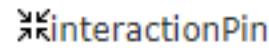


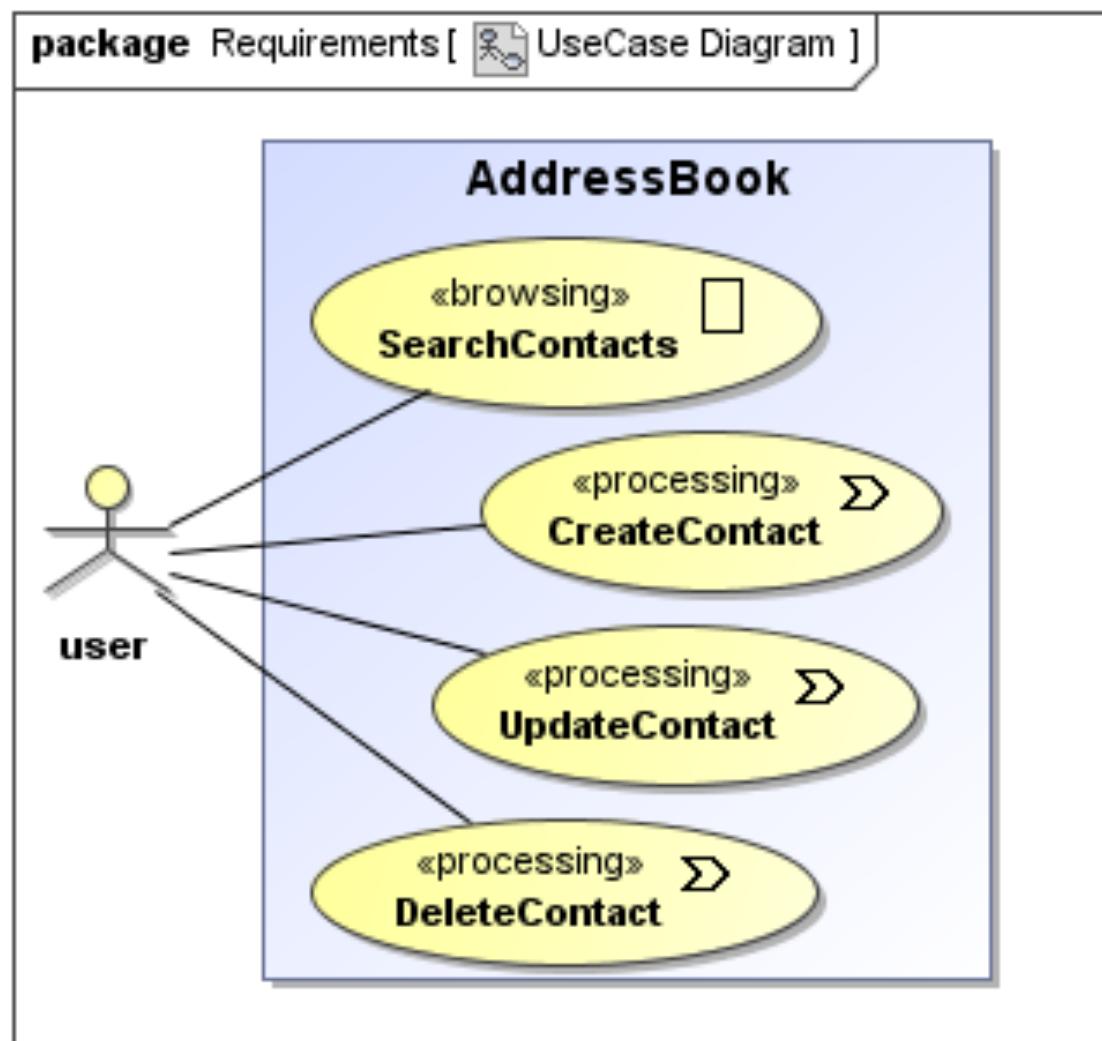
Diagrama de actividades: cada caso de uso se describe más detalladamente mediante un proceso. Es decir, las acciones que son parte de un caso de uso así como los datos presentados al usuario y aquellos requeridos como entrada de datos pueden ser modelados con precisión como actividades.



# UWE

Modelado de casos de uso:

Ejemplo agenda:



# UWE

Diagrama de actividades:

Los dos esterotipos «user Action» y «system Action» pueden ser usados análogamente al flujo de procesos. El estereotipo «user Action» es usado para indicar interacciones de usuario en la página web iniciando un proceso o respondiendo to un explícito requisito de información. Por lo contrario, «system Action» describe acciones que son ejecutados por el sistema. Ambos tipos de acciones pueden ser insertados usando la toolbar.

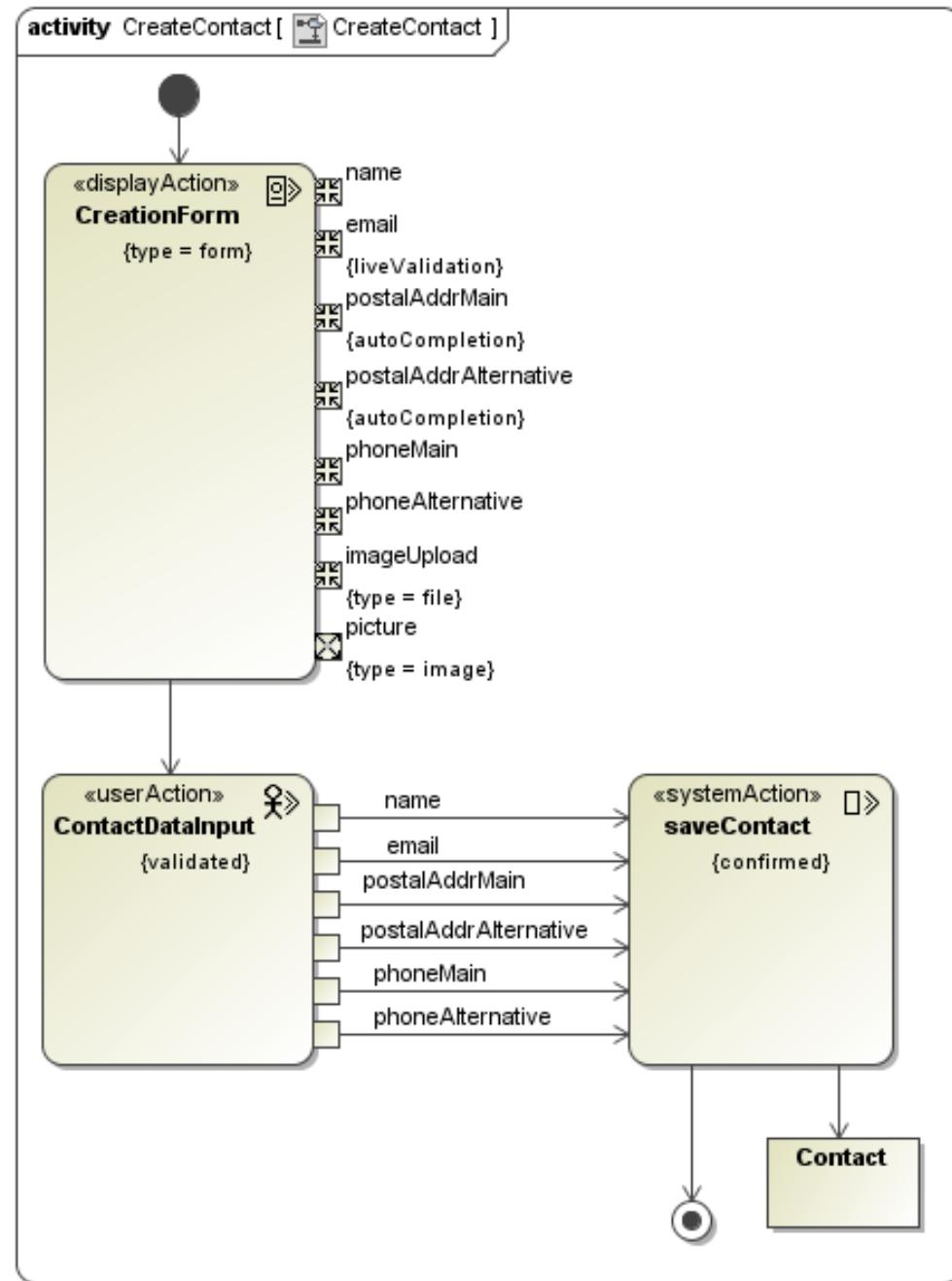
el estereotipo «display Action», mientras que los dos pines de acción estereotipados «interaction Pin» y «display Pin» son usados para modelar la entrada y la salida de datos.

Finalmente el estereotipo «navigationAction», puede ser usado para modelar opciones de navegación y los elementos asociados de presentación.

# UWE

## Diagrama de actividades

Crear contacto:



# UWE

Diagrama de actividades:

Buscar contacto

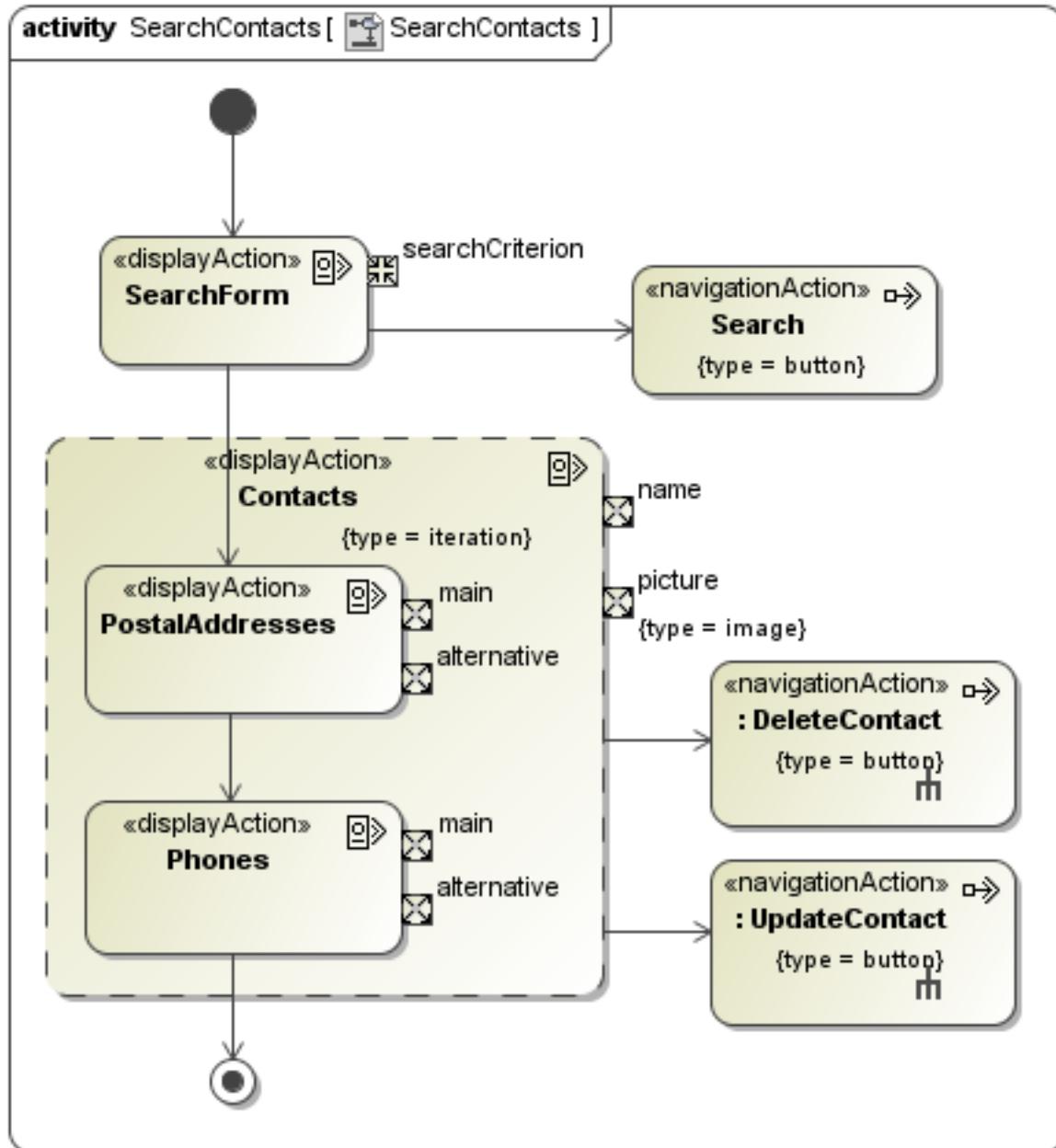
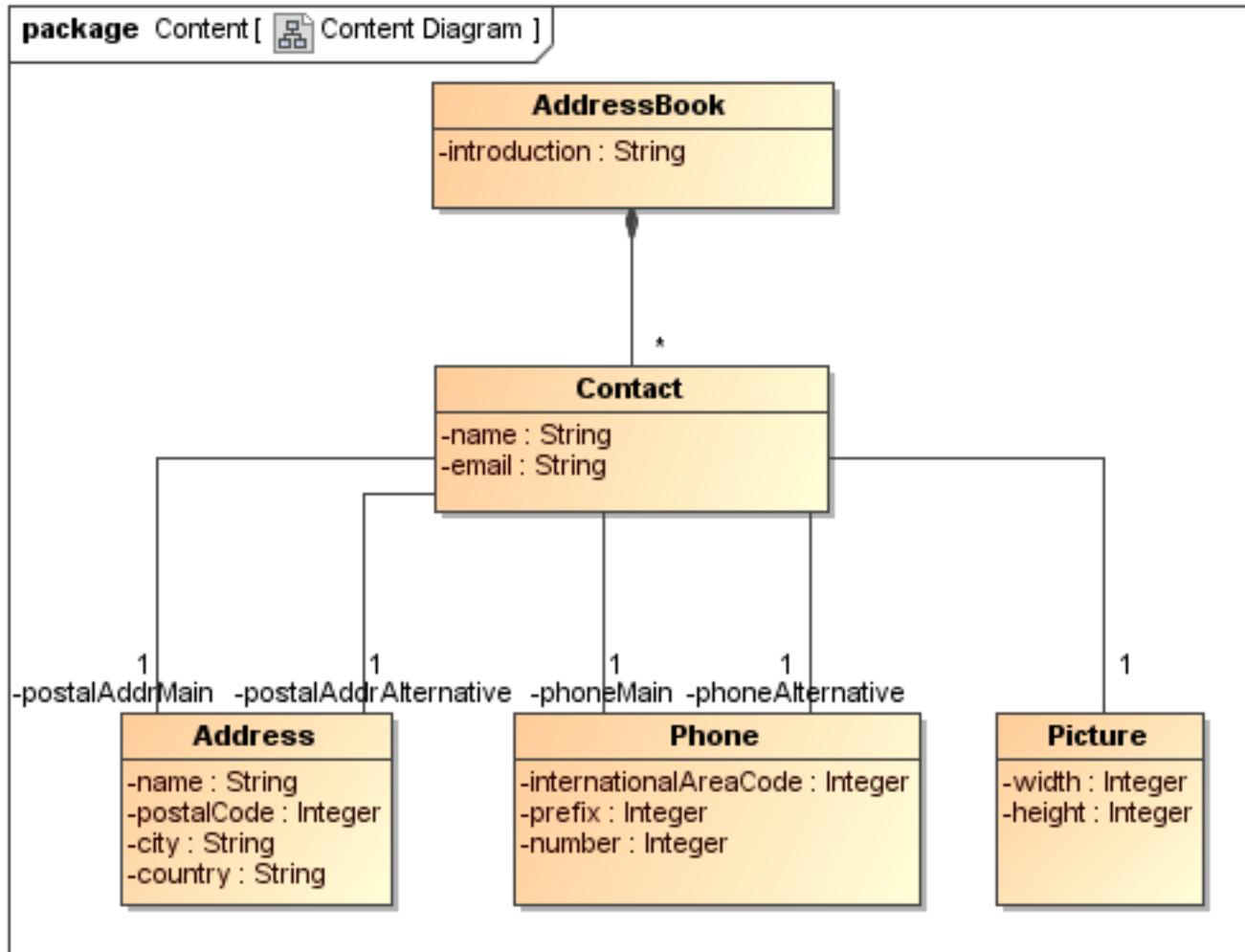
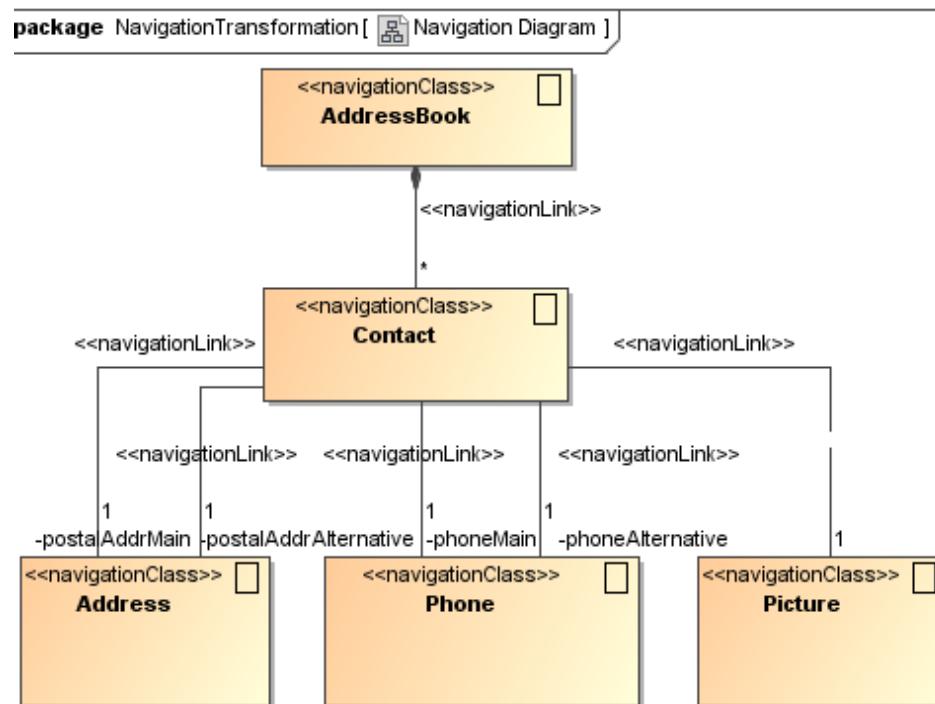


Diagrama de "contenido". diagrama de clases "normal":



# UWE

En un sistema para la web es útil saber como están enlazadas las páginas. Ello significa que necesitamos un diagrama conteniendo nodos (nodes) y enlaces (links). UWE provee diferentes estereotipos, ejemplo Para los nodos y enlaces son usados los estereotipos «navigationClass» and «navigationLink»:



UWE