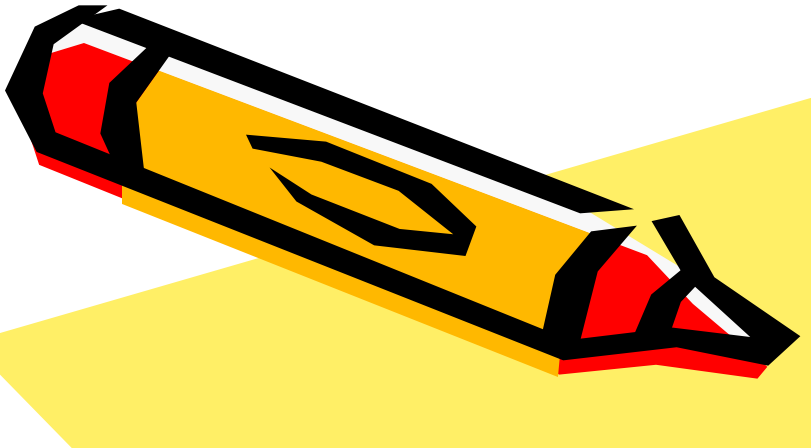
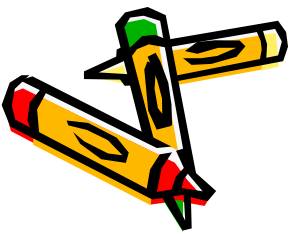


Web应用建模

Qiuyan Huo 霍秋艳
qyhuo@mail.xidian.edu.cn
Software Engineering Institute

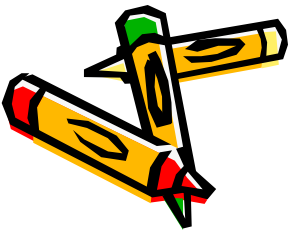
Web应用建模

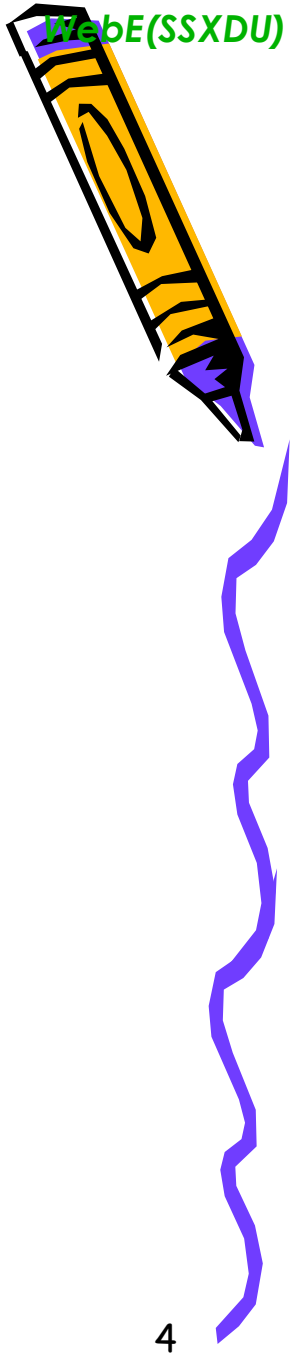
- 如何降低Web应用开发的风险和保证其质量
- 基于模型的方法提供了比特定（ad hoc）的Web应用开发更好的方法
- Web应用建模需要考虑
  - 静态和动态
  - 内容、超文本、展示、以及个性化适应性



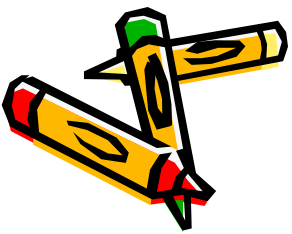
Web应用建模

- Web应用建模特性
- 模型驱动开发
- Web应用建模方法与工具
- 功能需求建模
- 内容建模
- 超文本建模
- 展示建模
- 适应性建模
- 总结与展望



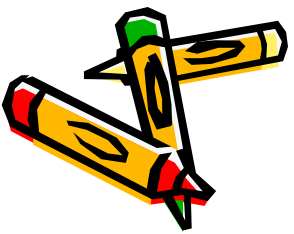


WEB应用建模特性



模型

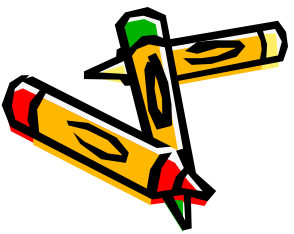
- 现实世界的抽象表达
  - 发现领域中的对象/概念
  - 给对象分配职责
- 思维工具
  - 降低复杂性
  - 文档化设计决策
- 沟通手段



建模的目的

- 足够详细的规格说明
  - 作为自动模型转换的基础
  - 实现/编码的输入
- 可读的系统结构和功能描述
- 有助于对系统进行可视化

- 必须的过程
  - 产出：表达系统相关方面的简化的、易于理解的模型



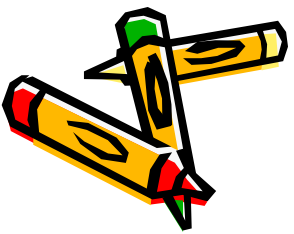
软件建模

- 计算机科学:范围跨越如下三维（orthogonal **dimensions**）
  - 层(Levels): 封装了“什么”和“如何”实现应用
  - 方面 (Aspects): 对象及其属性以及与其它对象的相互关系，函数和加工
  - 阶段 (Phases): 开发周期

UML的统一

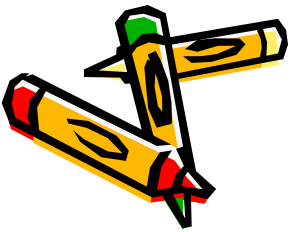
UML的统一

- UML (Unified Modeling Language): a set of modeling conventions that is used to specify or describe a software system in terms of objects.
  - The UML does not prescribe(指定) a method for developing systems—only a notation that is now widely accepted as a standard for object modeling.



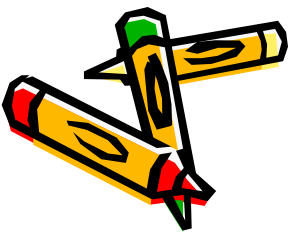
OO(object-oriented)原则

- OO 分析和设计
  - 分析: 发现领域中的对象/概念
  - 设计: 定义对象并确定对象如何交互以完成需求
- Key skill: 为对象分配职责



对象(Objects)

- 软件实体– 类似现实世界中的实体 – 具有状态(states)和行为(behaviors)
- States:
  - Variables store the states of an object's properties
  - Hidden from the outside world (data encapsulation)
- Behaviors:
  - Methods define the object's behaviors
  - Used by objects to communicate with other objects
- Classes
  - blueprints for creating objects of a particular type

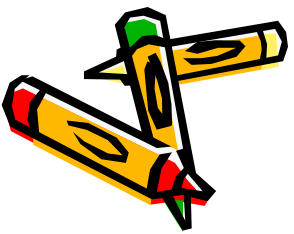


发现领域中的对象

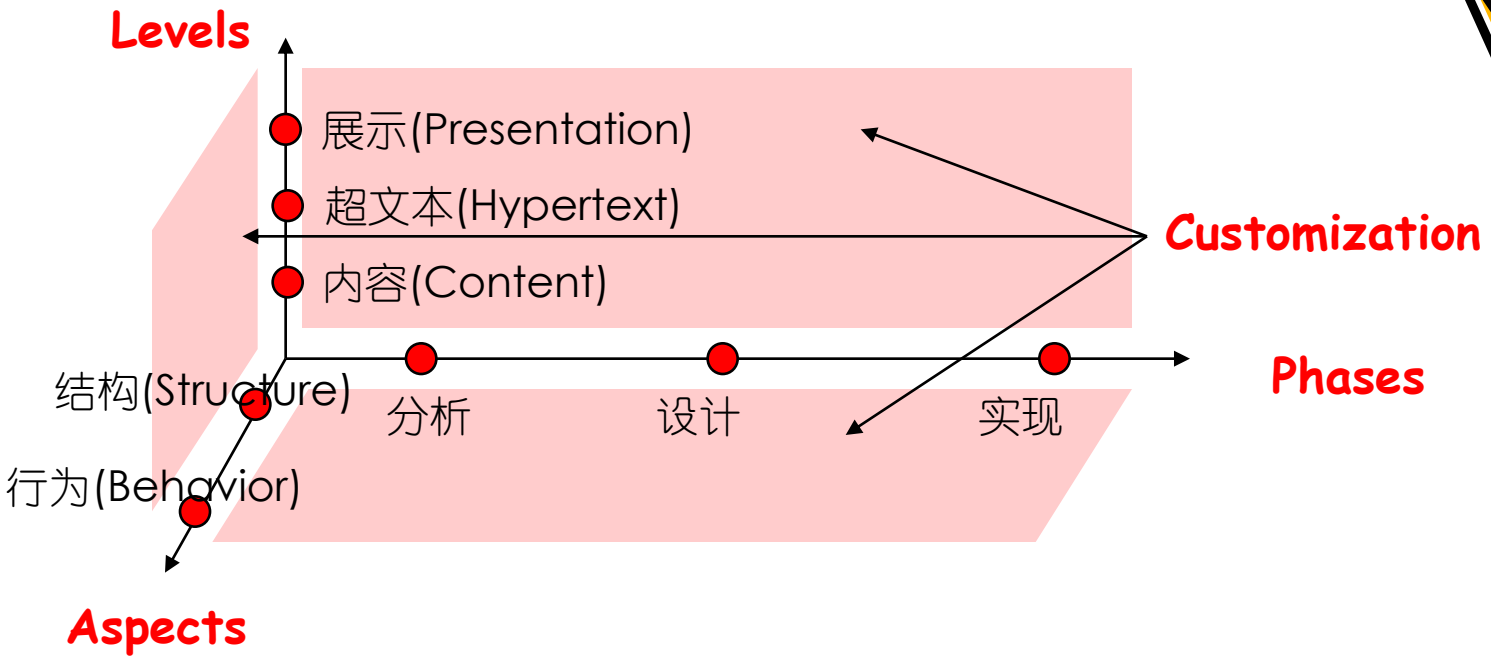
- 表达领域软件模型的方式应该与物理模型尽量相似
- 发现关键对象类
  - 如果有已存在的模型，重用
  - 分类列表
    - People, places, things
    - Transactions
    - Events
  - 识别名词短语
- 使用领域中常用词语来命名
  - i.e., 用户理解的术语

分配职责

- 职责是一个角色相关的责任/义务或特定行为
- 对象做什么?
  - Doing something itself
  - Pass actions (messages) to other objects
  - Controlling & coordinating the activities in other objects
- 对象知道什么?
  - Private, encapsulated(*封装的*) data
  - Its related objects
  - Items it can derive or calculate



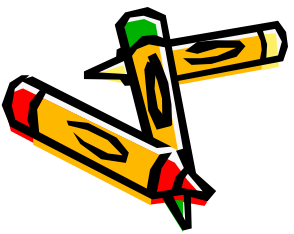
Web应用建模



- 层 (Levels)– 信息,结点以及相互之间的链接结构, UI & 页面布局
- 方面 (Aspects) – 和软件一样：结构和行为
- 阶段 (Phases) – 和应用类型有关的开发方法
- 适应性 (个性化、定制，Customization)– 上下文(Context)

分层的优点

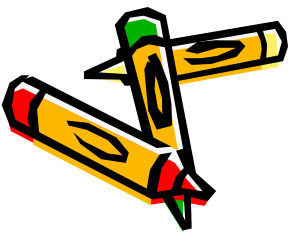
- 降低复杂性
- 模型演化
- 分层的模型栈
  - 不同的超文本结构位于相同内容之上
  - 不同展示模型位于相同超文本模型之上
- 不同的建模目标
  - 内容模型:展示信息结构，无冗余
  - 超文本建模:适当的冗余，如：通过多条路径访问到信息
  - 展示层建模:统一的展示结构



方面

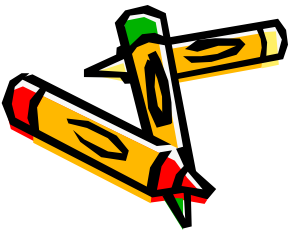
- 采用OO方法, 对每一层进行结构和行为建模
- 结构和行为模型之间的关联,和要实现的Web应用的类型有关
  - 主要是静态(static)信息的Web应用需要比较少的行为建模
  - 高交互性(interactive)的Web应用,如e-commerce应用提供search engine、订购等功能, 需要更多的行为建模

推荐用统一的建模形式对结构和行为进行建模，以使不同层之间可以进行映射。 (i.e., 依赖一种CASE工具)

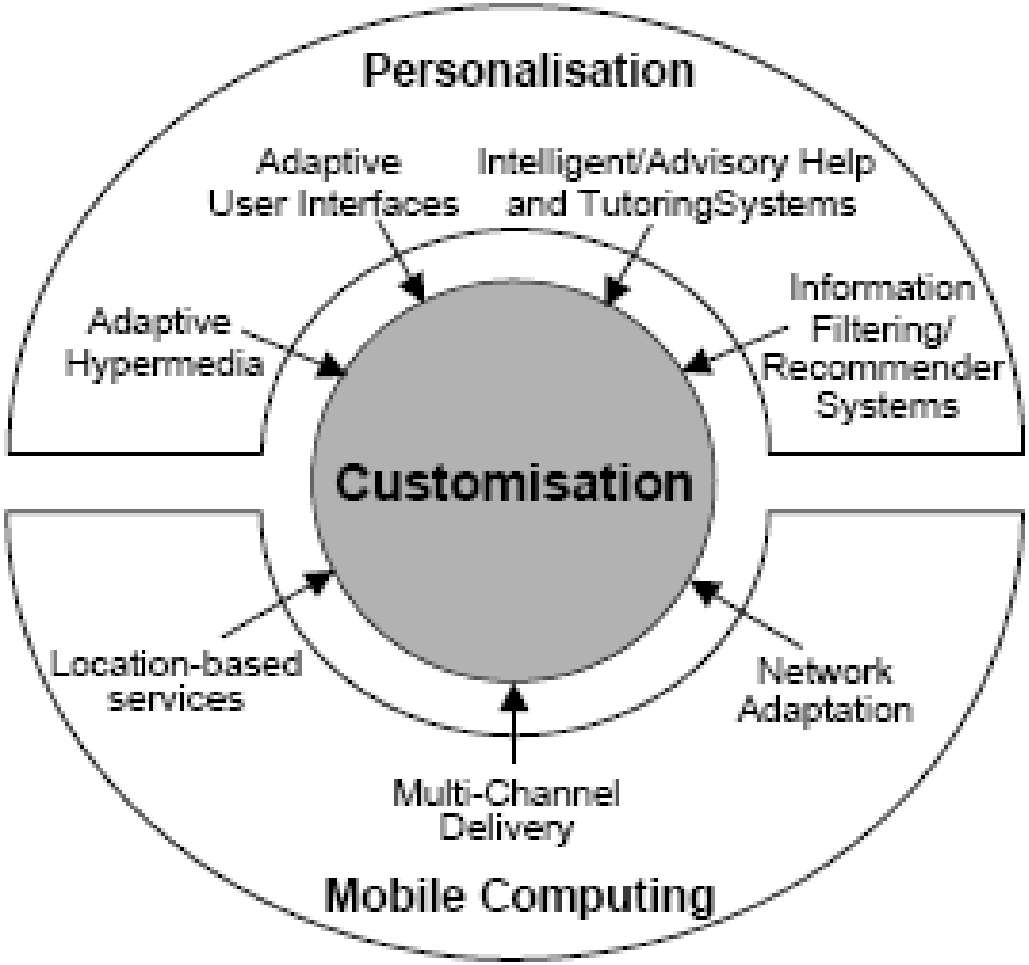


阶段

- 与Web应用的类型有关:
  - 信息驱动方法（如，从内容建模开始）
  - 展示驱动方法（如，从建模应用的展示开始）
- Web工程中基于模型的开发方法
  - 有时候和Web项目的实践需求有些冲突
  - 能确保解决方案的持久性
  - 更好地进行沟通

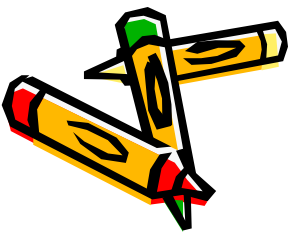


适应性


Personalisation
Adaptive
User Interfaces
Intelligent/Advisory Help
and TutoringSystems
Adaptive
Hypermedia
Information
Filtering/
Recommender
Systems
Customisation
Location-based
services
Network
Adaptation
Multi-Channel
Delivery
Mobile Computing

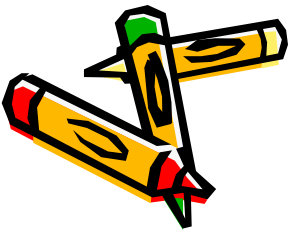
适应性

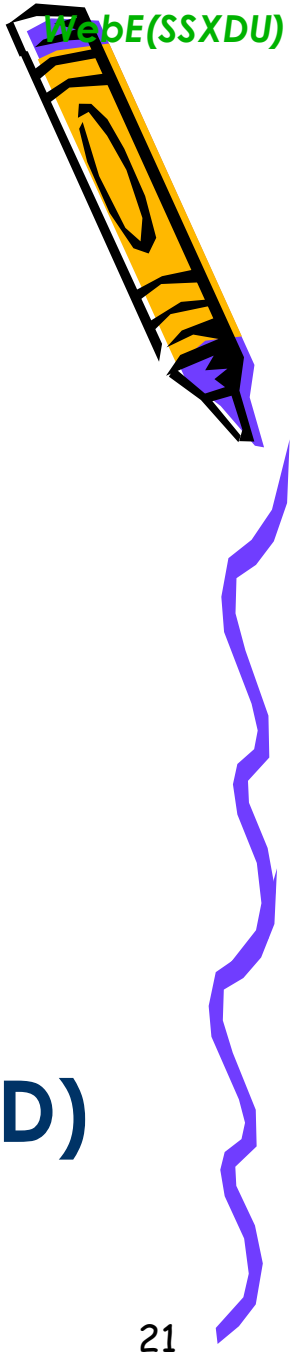
- Web应用针对个性化和移动计算需求进行自适应
- 影响着Web应用建模的其它三维
- 处理上下文信息看作是独立的第四维特性——适应性
- 主要处理为什么和何时进行适应
- 特定属性的具体化
- 描述应用的环境和应用自身的一些方面



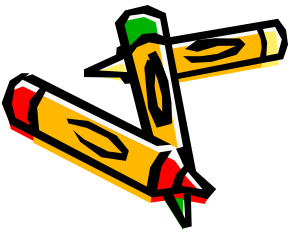
Web应用建模

- 建模内容、超文本和展示的静态和动态方面
  - 更好地理解系统
  - 满足协同开发项目的需要
  - 满足用户需求
  - 开发过程可控
  - 系统可持续性发展



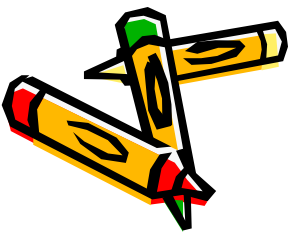


模型驱动开发
(MODEL DRIVEN DEVELOPMENT, MDD)

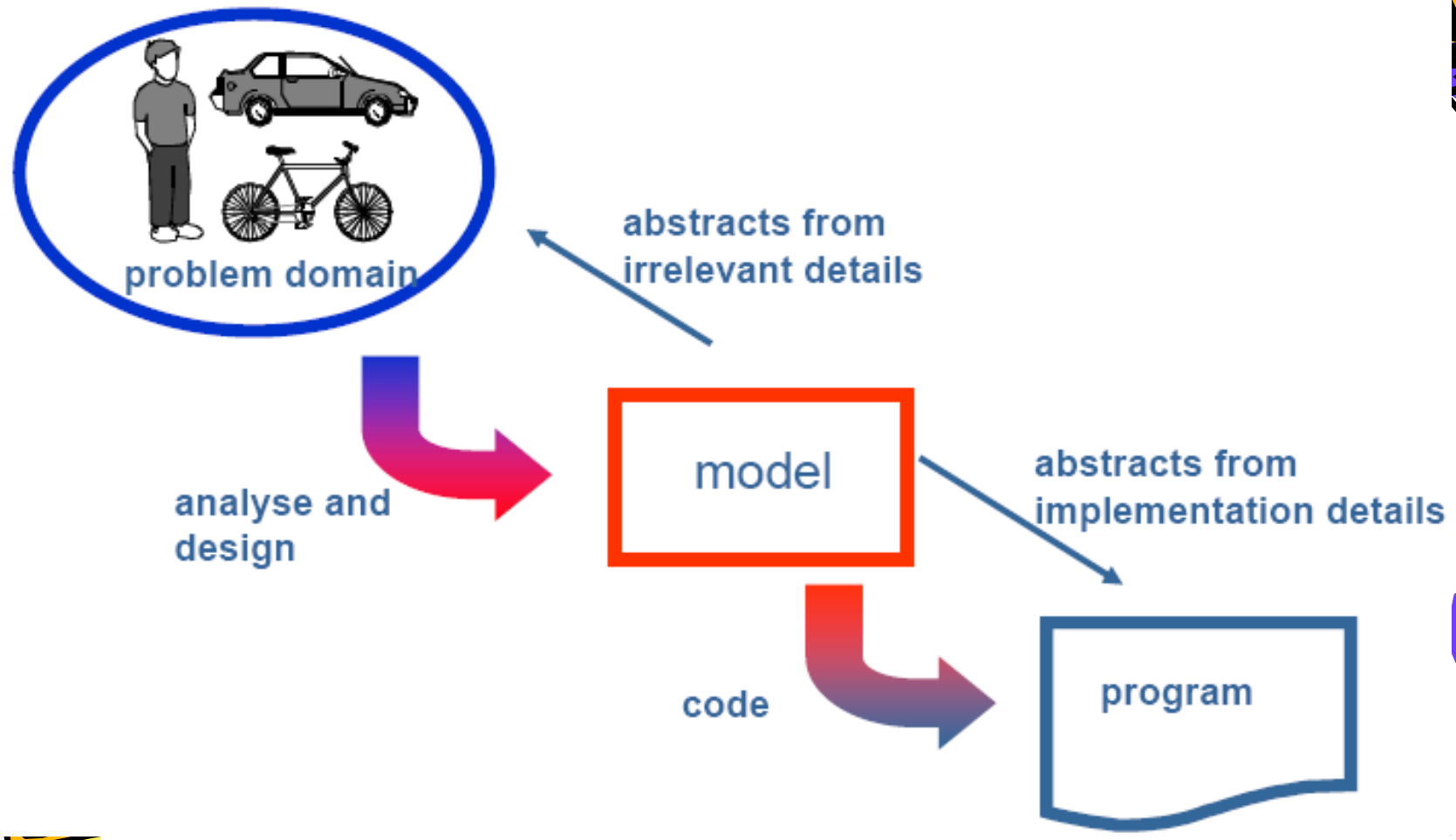


模型驱动开发

- 以建立模型为主要手段的一种开发方法
  - 以模型作为开发过程中的主要制品，模型可以自动或半自动生成元数据和实现代码。
  - 不仅仅强调在开发中使用模型，也强调在整个开发过程中所有开发阶段之间的转换。模型之间的转换提供了系统的整个生命周期中不同模型之间的自动转换能力。

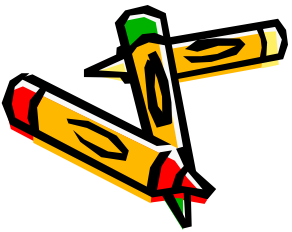


Model-Driven Development (MDD)
Model-Driven Engineering (MDE)

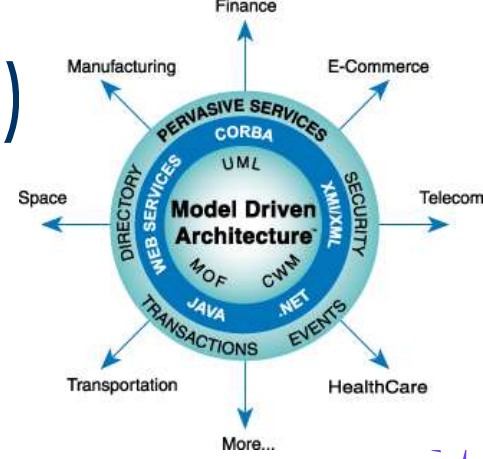


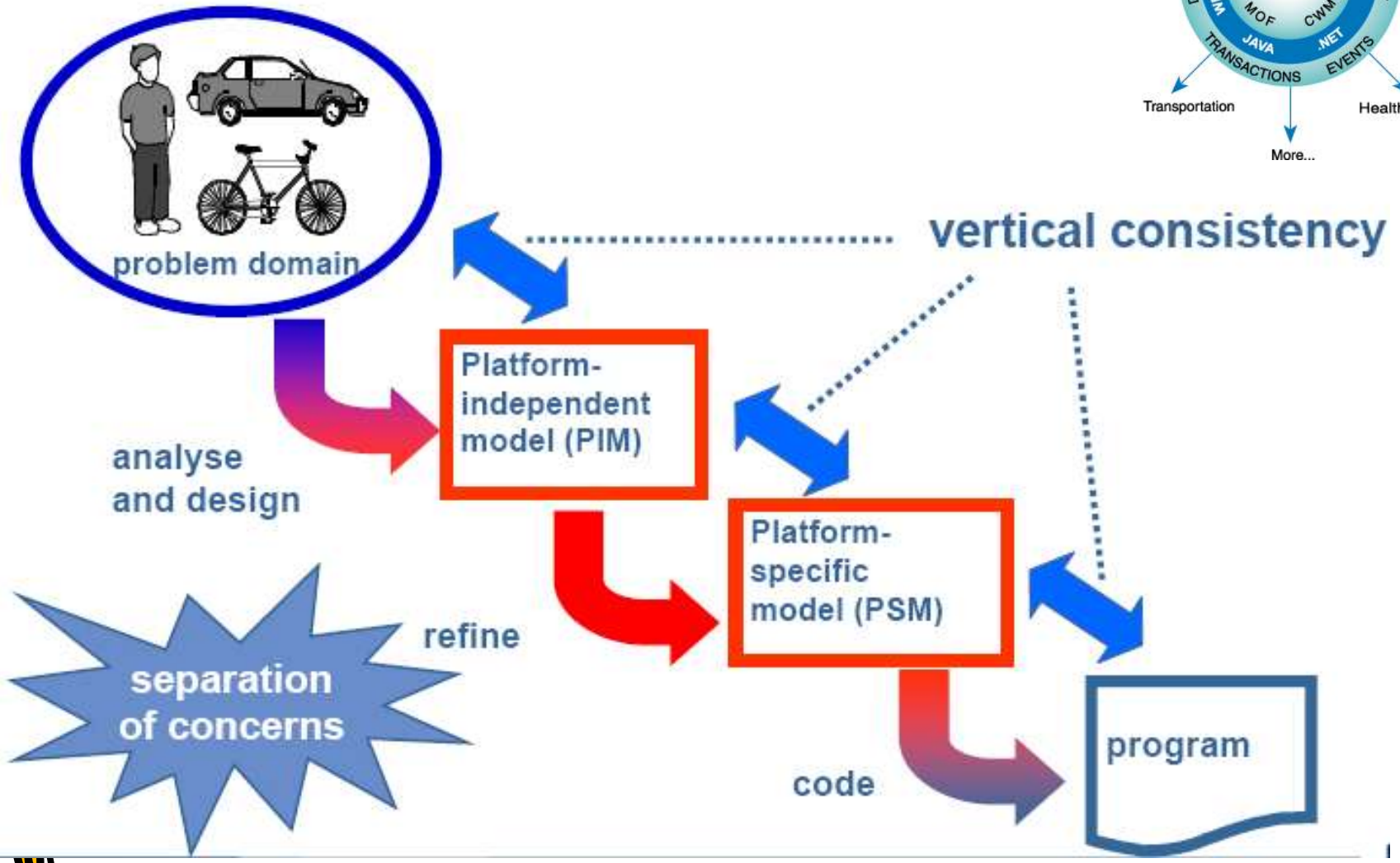
Technological Obsolescence(荒废、退化)

We don't want anymore to pay such a high price for simply moving our information system to a new middleware platform (COM, CORBA, Java, HTML, XML, DotNet, etc.) when our business system stays stable.
We are prepared to pay a last price for building the abstract models of our business and services that will guarantee us against technological obsolescence.
From there, any platform provider will also have to provide the mapping solutions from standard business models before we buy.



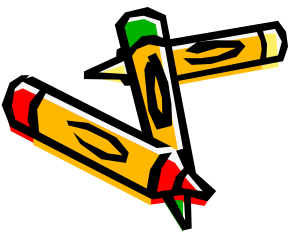
Model-Driven Architecture (MDA) -模型驱动架构





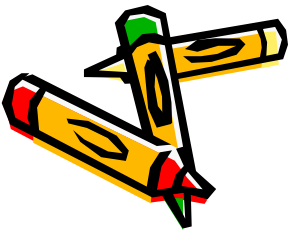
敏捷模型驱动开发（Agile MDD, AMDD）

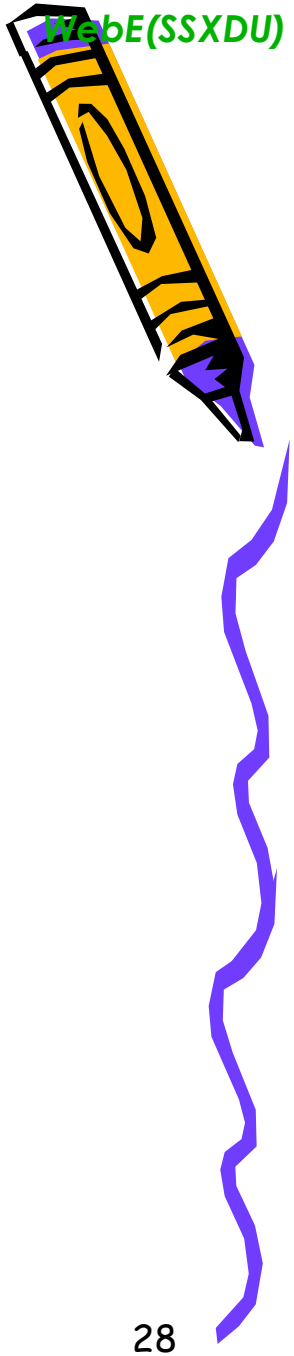
- 不强调在编写或生成代码之前创建大量的模型，而是创建一定程度上够用，甚至只有很少的模型，即敏捷建模（Agile Modeling，AM），然后就进行编码。
- 在编码实现模型之后，或者编码过程中，再根据需要迭代进行进一步建模。



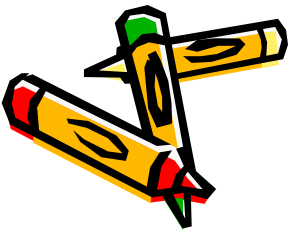
MDD的好处

- 更好地指导Web应用的开发
- 提高Web应用开发的效率
- 提升Web应用的质量
- 灵活性，即当Web技术演化而引入新技术时它的灵活性。

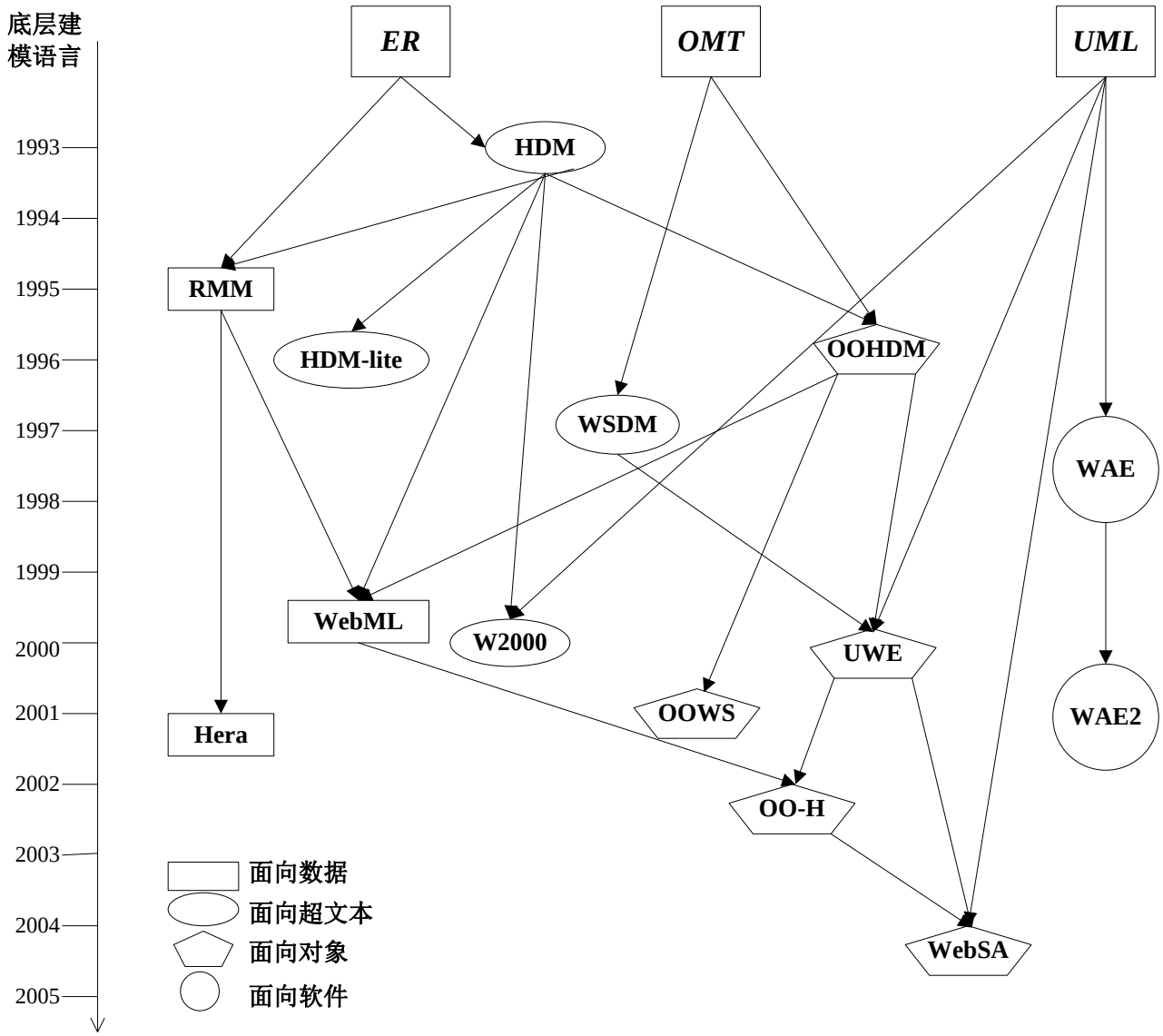




WEB应用建模方法与工具

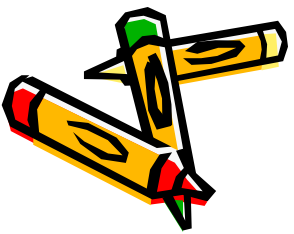


建模Web应用的方法


底层建模语言
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
ER
OMT
UML
HDM
RMM
HDM-lite
OOHDM
WSDM
WAE
WebML
W2000
UWE
OOWS
WAE2
Hera
OO-H
WebSA
面向数据
面向超文本
面向对象
面向软件

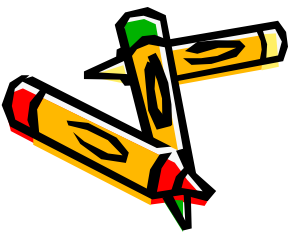
建模Web应用的方法

- 便于将Web应用模型分为领域模型、导航模型和展示模型，经过概念建模、逻辑建模、物理建模和实现四个过程完成Web应用开发
  - 领域模型、导航模型和展示模型分别描述系统的一个不同侧面，可以看成是Web模型的不同视图
  - 领域模型描述Web应用中领域对象及其关系，是导航模型的基础
  - 展示模型描述Web页面展示形式，是导航对象和导航行为的最终体现
  - 导航模型描述了Web应用的导航特性，并衔接领域模型和展示模型



UWE (UML-based Web Engineering)

- UWE: 基于UML的Web工程
- UWE扩展了UHDM
- UWE符号
  - 与UML兼容
  - 综合建模工具
  - http://uwe.pst.ifi.lmu.de/index.html

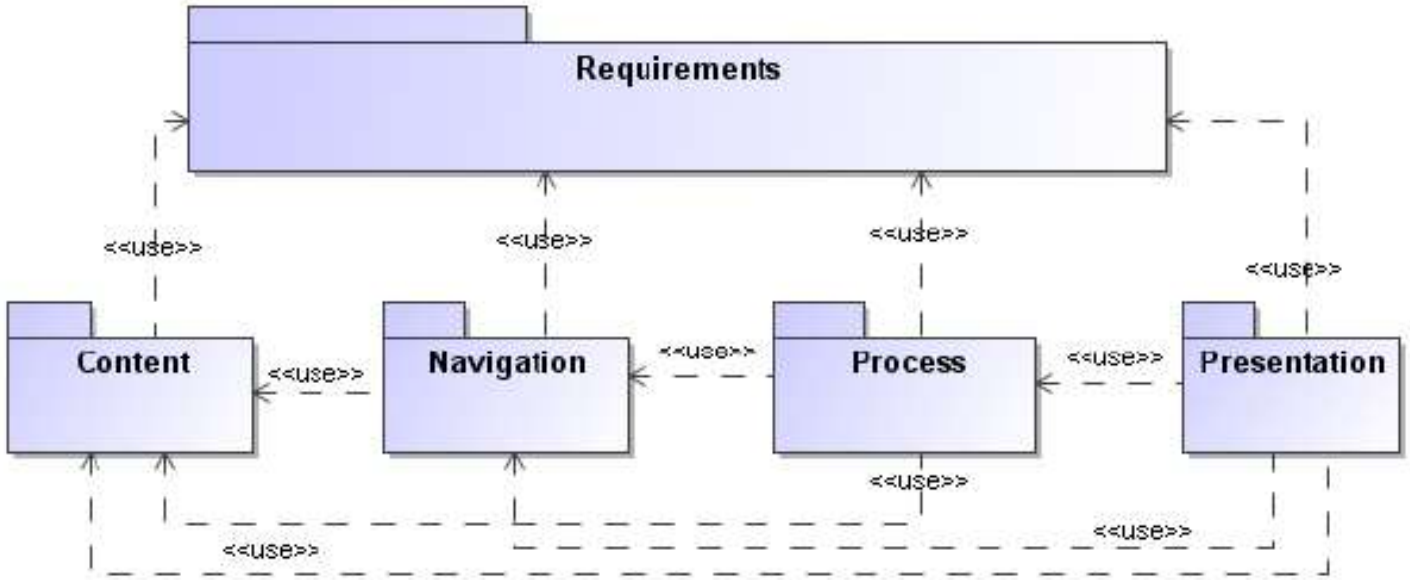


UWE Methodology(*方法学*)

- UWE强调“分而治之”
  - 内容、导航结构、业务过程、展现
- UWE关注
  - *系统化* (Systematization)
  - *个性化* (Personalization)
  - *半自动化*Web应用*生成* (Semi-automatic generation)
    - UWE通过定义不同类型之间的模型转换，以从PIM产生PSM和生成可运行程序，从而实现模型驱动开发过程
- UWE是一个面向对象的、迭代的建模方法，关注系统化、个性化的开发和生成Web应用

UWE模型

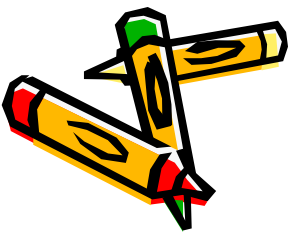
- 模型：需求、内容、导航、过程、展示
- 扩展的链接：导航链接（«navigation link»）、过程链接（«process link»）和外部链接（«external link»）



UWE元模型

需求模型

- UML的用例图
  - 由UML建模元素及元素之间的关系构成
    - 用例和角色
    - 角色和用例之间的“关联”关系、用例之间的“包含（«include»）”与“扩展（«extend»）”关系、角色之间或用例之间的继承关系
- 导航业务过程用例用**«navigation»**
- 例如，在新闻系统中浏览新闻视频



内容模型

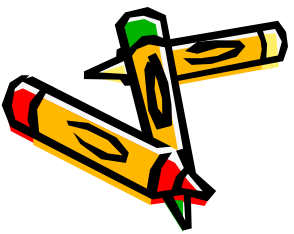
- UML类图： Web应用和领域有关的信息的规格说明
- 静态视图
- 尽量忽略应用程序的导航、展示和交互等方面的内容，仅仅表达应用程序的概念框架
- 例如：
  - 在视频库中包含的视频按分类（如娱乐、体育等）
  - 每部视频具有名称、时间、简介、图片、类型（娱乐或体育等）等
  - 将视频库设计为VideoCollection类，视频设计为Video类，则Video和VideoCollection类中包含向视频库中添加视频或从视频库中删除视频的方法

导航模型

- 展现Web应用系统的超文本结构，用结点（node）和链接（link）进行表示
- 具有导航性的类，用扩展的stereotype «navigation class»来表示信息获取
  - 如VideoCollection或Video
- 用«process class»定义发生事务处理的导航节点
  - 如AddVideo和RemoveVideo
- 用关联关系建模直接链接，尤其是关联关系<<process link>>的一头是过程类
- 一些专门的导航节点，以便于组织链接
  - 一个导航类的一些实例通过«index»类表示
  - 一些可选的链接用«menu»类来表示
- 用户交互
  - UML的动作， «userAction»

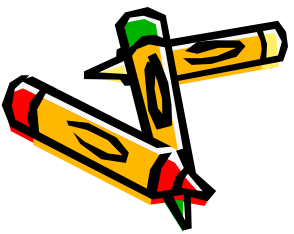
导航模型

- 导航结构模型
  - 超文本结构，即内容模型中包含的类和对象映射为超文本中节点（页面或文档）以及这些节点之间的链接
  - 内容模型的上层视图
- 导航访问模型
  - 描述哪些节点可以通过导航方式来访问，以及如何通过导航访问节点



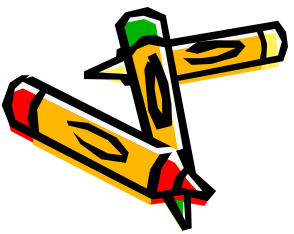
展示模型

- Web应用用户界面（UI）设计
- 描述用户可见类对象和存取结构（如索引、向导、菜单、查询）在什么位置出现、以什么面貌出现
  - 结构类构造为«presentationGroup»模型
  - UI元素（如文本、图片和按钮等）构造为«text»、«image»和«button»等



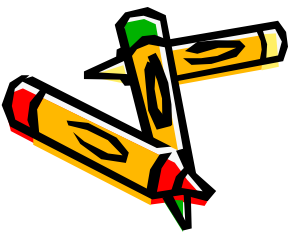
适应性模型

- UWE采用面向方面建模（Aspect-Oriented Modeling, AOM）技术进行适应性建模。
  - 系统功能和个性化方面系统地进行分离



模型一致性

- UWE中不同模型之间具有一致性约束或生成关系
  - 导航模型的基础是底层内容模型，超文本模型或多或少和内容模型密切相关
  - 展示模型和导航模型之间的映射，通常认为一个节点的所有实例将在展示层展现，并且需要考虑Web应用中用户的交互行为
  - 适应性模型影响前面所有的Web应用建模过程，变化可能会在一层或者也会影响几层

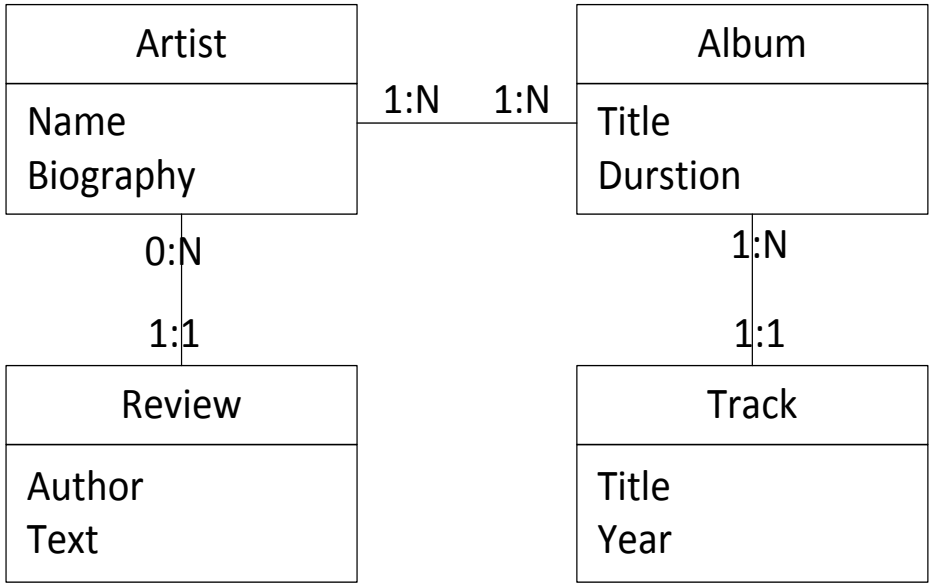


WebML(Web Modeling Language)

- Web建模语言
- 图形符号和XML语法进行描述
- 主要目标：
  1. 运用高层的描述表示Web应用的结构
  2. 对同一内容建立多个视图
  3. 分离信息内容。将页面、导航呈现从它们的合成中分离出来，单独定义和演进或求精
  4. 将设计过程中收集到的媒体信息保存到数据存储中，在Web应用生命周期中这些信息将被用于动态产生Web页面
  5. 为支持个性化策略和一对一的Web应用，可明确地建立用户或用户组模型
  6. 能规范地描述与Web应用内容修改相关的数据维护操作以及任意的与外部服务的交互

WebML:结构模型(Structure Model)

- 描述Web应用的数据内容：实体及其之间的关系

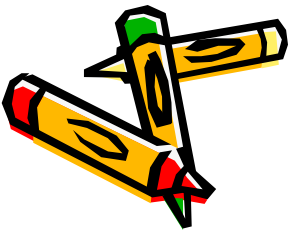


WebML:超文本模型

- 描述Web应用中的超文本
- 每个不同的超文本定义一种Web应用视图
- 组成模型（Composition Model）
  - 组成超文本的页面
  - 内容单元：数据、数据集（multi-data）、索引、过滤器、滚动和链接
- 导航模型（Navigation Model）
  - Web页面间的链接关系拓扑模式
  - 表示Web页面和内容单元如何链接而形成超文本结构
  - 上下文无关链接HYPERLINK/上下文相关链接INFOLINK

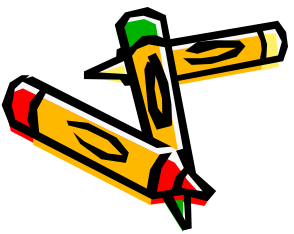
WebML:展示模型（Presentation Model）

- Web页面的物理外观和感觉
- 由抽象XML语法实现
- 页面特定的或是通用的



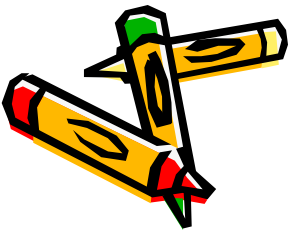
WebML:个性化模型（Personalization Model）

- 适应性数据的记录模式
- 两类用户：用户（User）和组（Group）



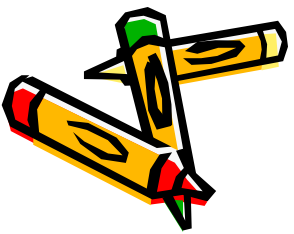
HDM-lite

- HDM
  - 结构化链接（Structural link）、透视链接（Perspective link）和应用链接（Application link）
- 不同设计“维”:内容、导航/交互和展示
- 扩展HDM
  - 强调自动化的开发过程和Web应用的自动生成
  - 主要是定义结构、导航和展示，分别用HyperBase、Access、展示模型来表达
- 工具
  - Auto Web系统工具



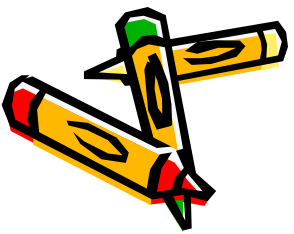
OOHDM

- OOHDM（Object-Oriented Hypermedia Design Method，面向对象超媒体设计方法）
- 迭代步骤
  - 捕获需求：OO方法将一组UID映射为概念模型
  - 概念设计：领域的概念模型
  - 导航设计：超媒体应用的导航结构
  - 抽象界面设计：根据界面类定义的“感知对象”来构造抽象界面模型
- 设计工具：
  - OOHDM-Web（只读性的Web应用）

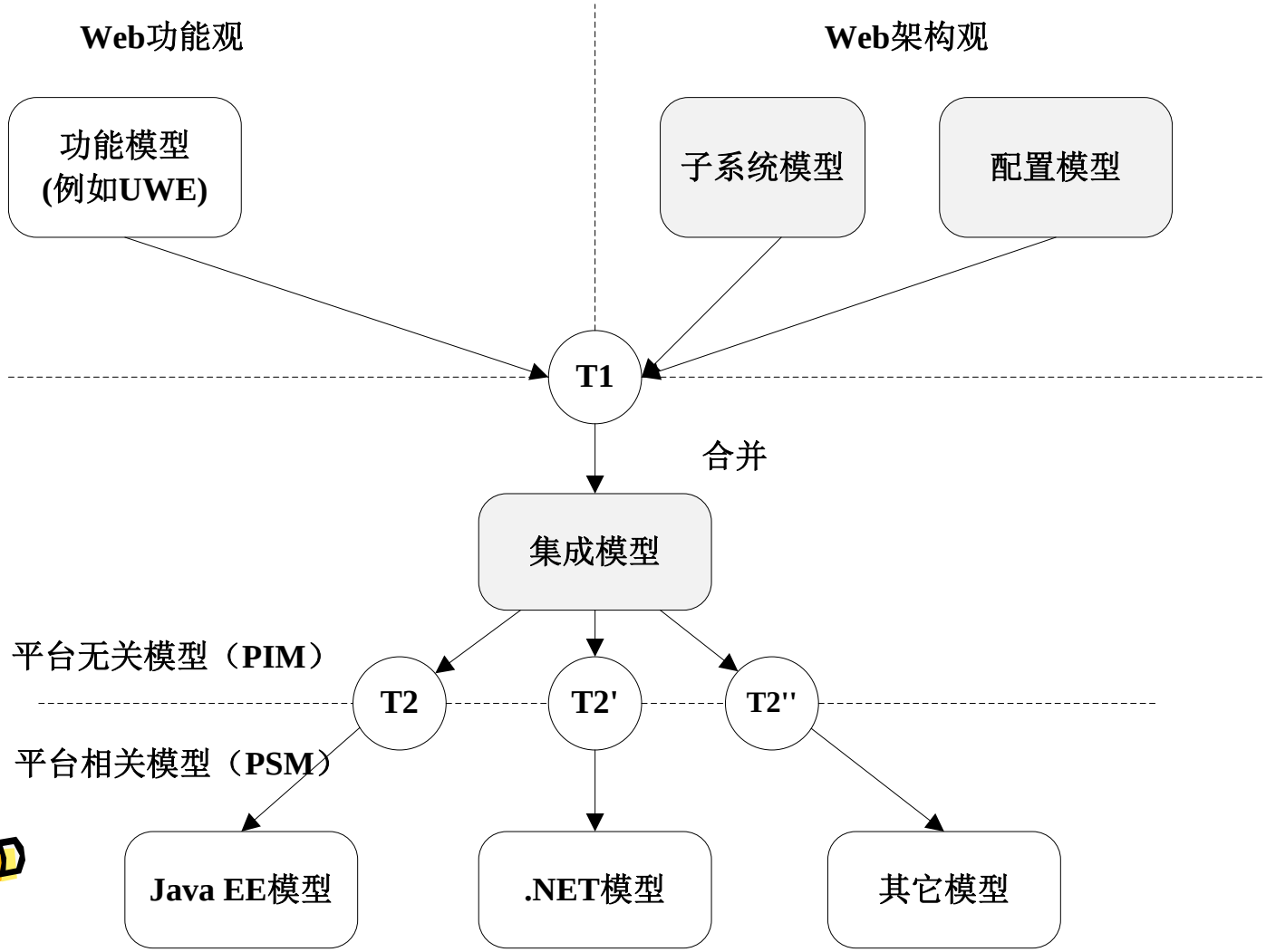


WebSA（Web Software Architecture）

- 一种Web领域中基于MDA泛型的模型驱动方法
- 基于UML、UWE以及OO-H方法
- 通过引入软件架构，使系统的功能需求和非功能需求都得到较好的满足，即建模Web应用架构
- 特性：
  - 提高了Web应用的开发速度
  - 使得Web应用接口与已存在模块的集成更加容易
  - 减少了Web应用的开发成本
- 使用UML模型表达Web应用以及QVT转换

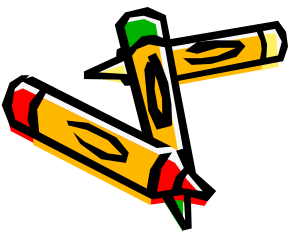


WebSA


Web功能观
Web架构观
功能模型
(例如UWE)
子系统模型
配置模型
T1
合并
集成模型
平台无关模型（PIM）
T2
T2'
T2''
平台相关模型（PSM）
Java EE模型
.NET模型
其它模型

其它方法

- RMM
- WebComposition
- OntoWebber
- W2000
- OOWS
- Araneus（ADM）
- OO-H
- WSDM
- CBSD

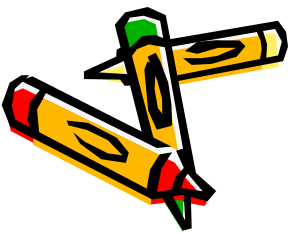


Web应用建模方法范型

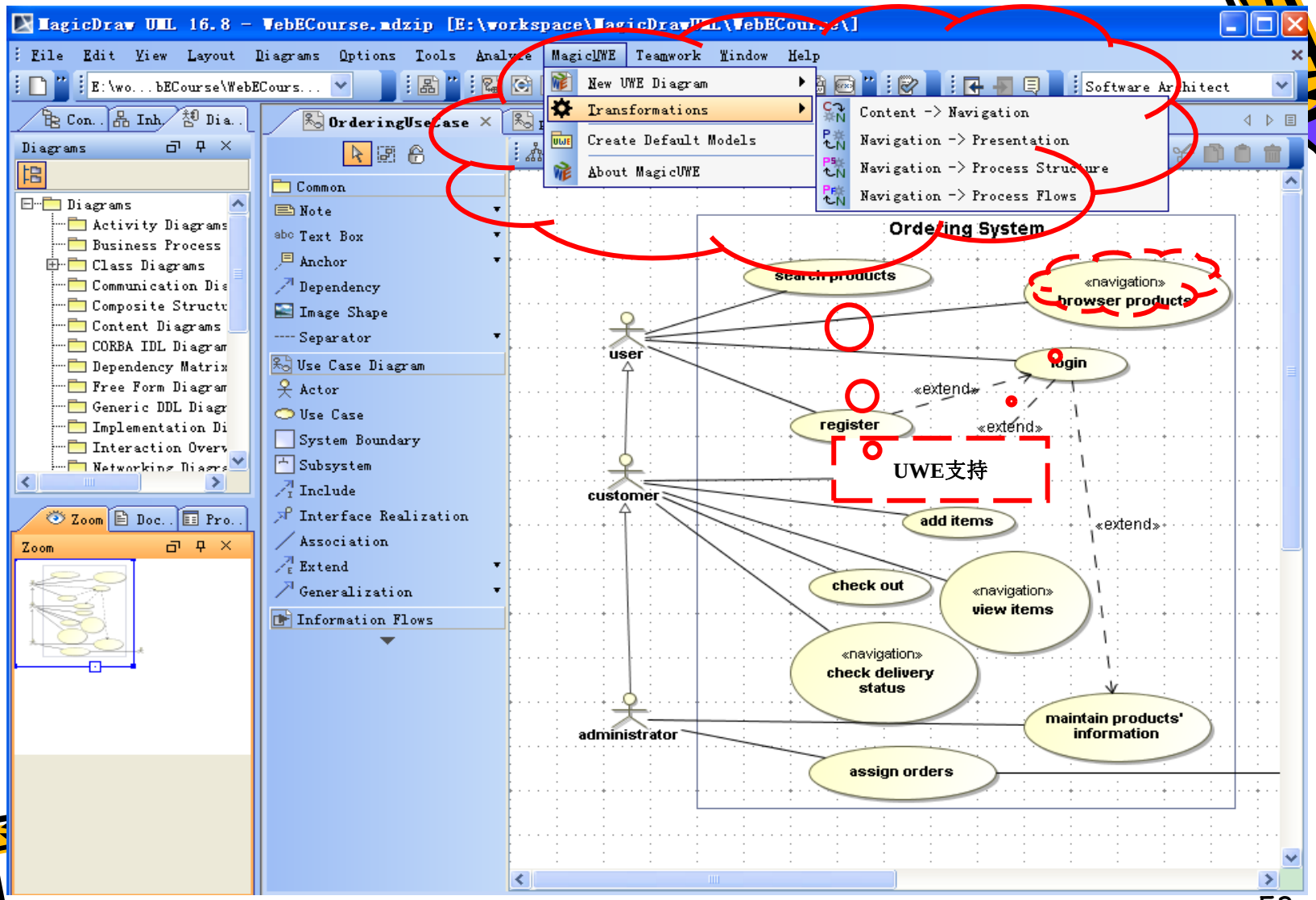
- 面向数据方法：
  - 基于E-R模型的方法，起源于数据库系统，主要关注数据库驱动的Web应用，如RMM、Hera、WebML，以及OntoWebber和ADM
- 面向超文本方法：
  - 关心Web应用的超文本特性，起源于超文本系统，如HDM、W2000、HDM-lite和WSDM
- 面向对象方法：
  - 基于OMT或者UML，其中UML获得更大青睐，如OOHDM、UWE、OOWS、OO-H和WebComposition
- 面向软件方法：
  - 将Web应用看作传统软件开发，使用沿着软件软件工程的技术，如基于UML技术的WAE及其改进版WAE2

建模工具：UWE工具

- MagicUWE
- ArgoUWE
- UWE4JSP
- tidyDiagram
- UWEet (palette)
- UWE2FACPL
- MagicSNP



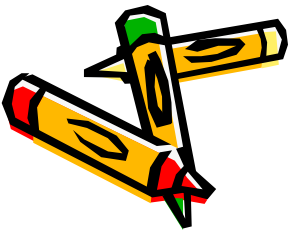
MagicUWE



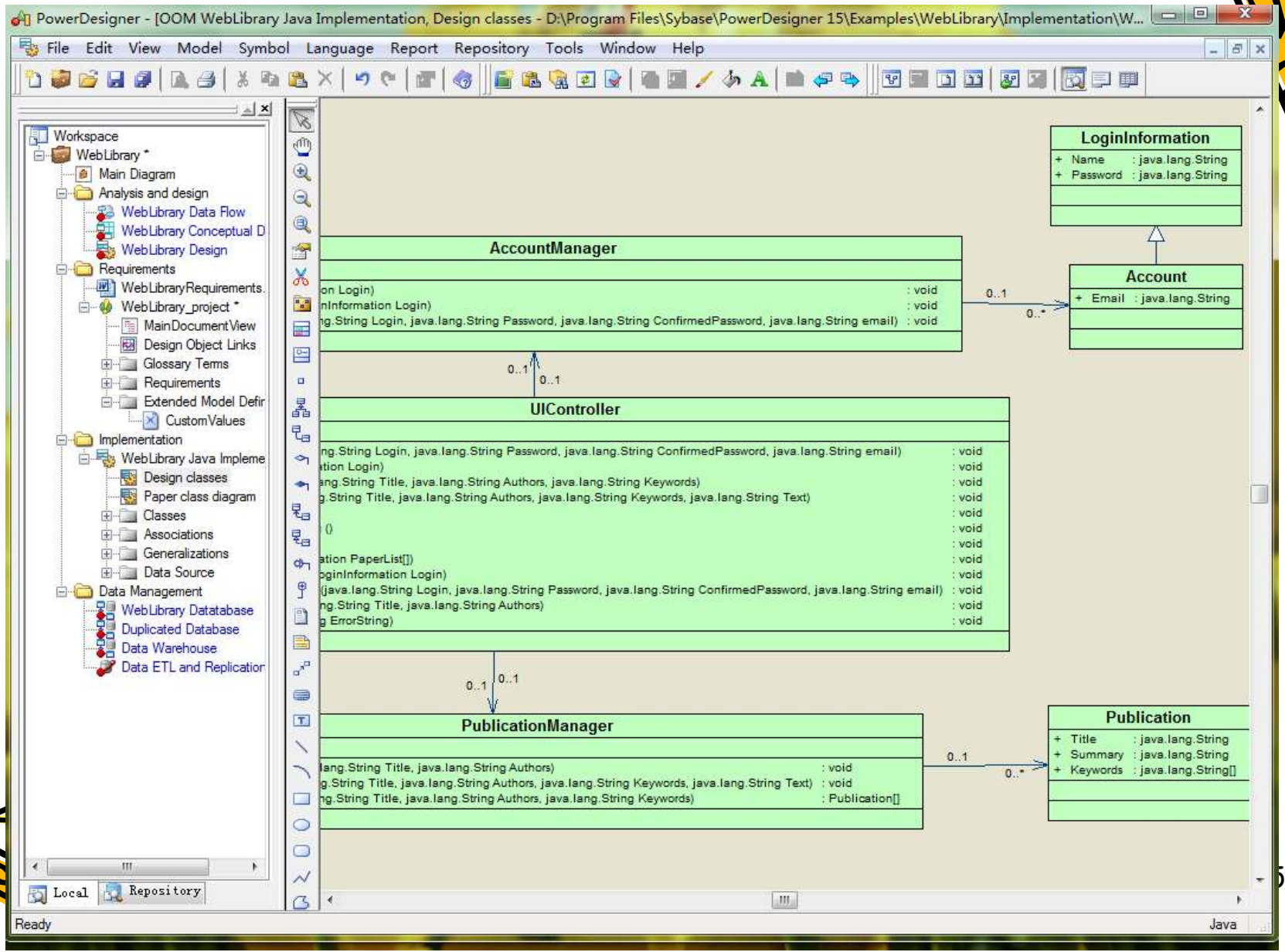
return

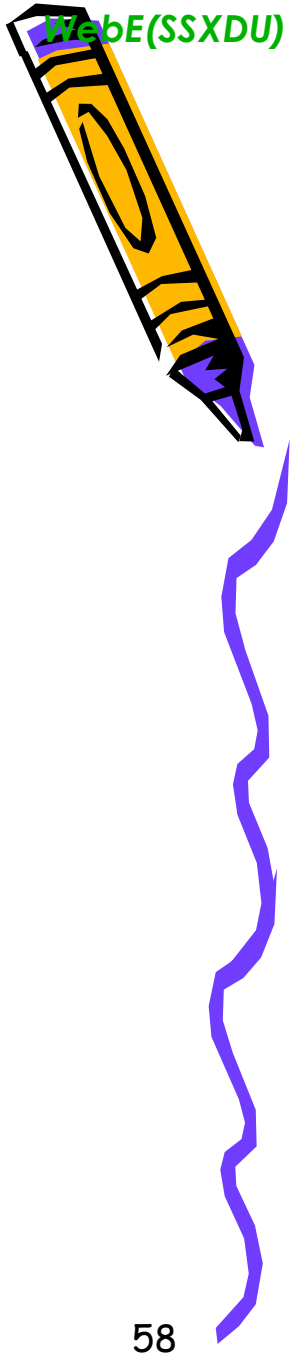
建模工具：WebML工具

- WebRatio： Eclipse插件
  - 对Web应用以及Web服务开发提供了一组完整的模型驱动工程方法
  - 直接将模型转化为可运行代码
  - Eclipse的优势

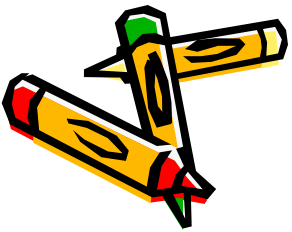


建模工具：PowerDesigner 15



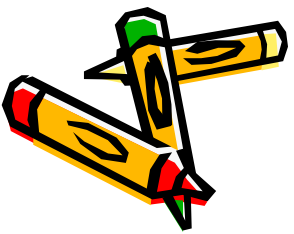


功能需求建模



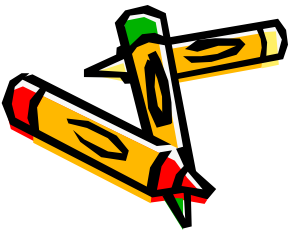
功能需求建模

- 采用UML用例图全局功能建模
- 基于参与者的视图
  - 至少一个参与者是人且通常是匿名的
- 通过UML activity进行精化
- 两类需求
  - 功能
  - 导航(Navigational, Web应用典型需求)
- stereotype **<<navigation>>**

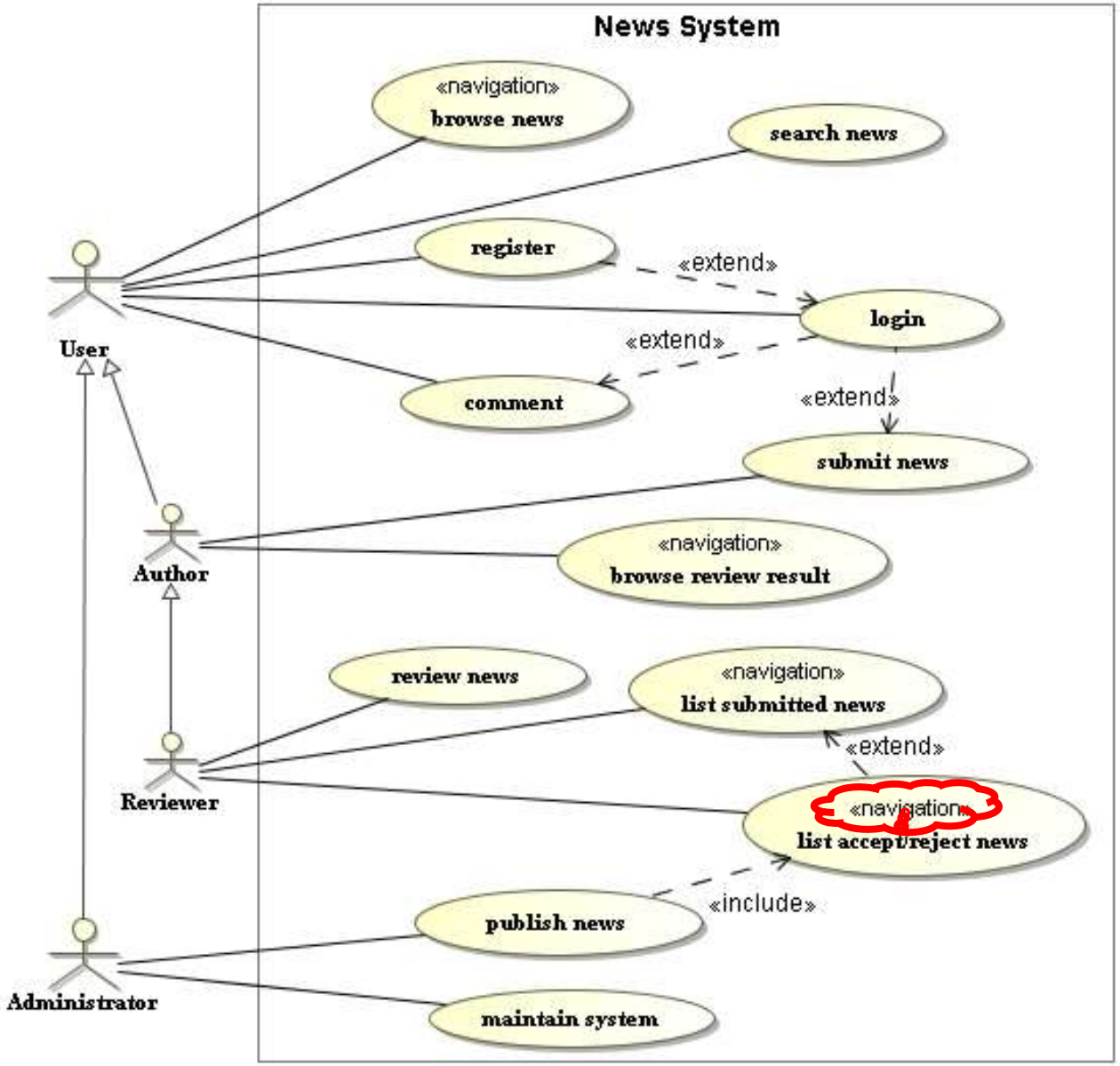


绘制用例图

- 确定系统参与者
  - 谁将使用该系统的主要功能？
  - 谁将需要该系统的支持以完成其工作？
  - 谁将需要维护、管理该系统，以及保持该系统处于工作状态？
  - 系统需要处理哪些硬件设备？
  - 与该系统交互的是什么系统？
  - 谁或什么系统对本系统产生的结果感兴趣？
- 获取用例
  - 特定参与者希望系统提供什么功能？
  - 系统是否存储和检索信息，如果是，由哪个参与者触发？
  - 当系统改变状态时，是否通知参与者？
  - 是否存在影响系统的外部事件？
  - 哪个参与者通知系统这些事件？

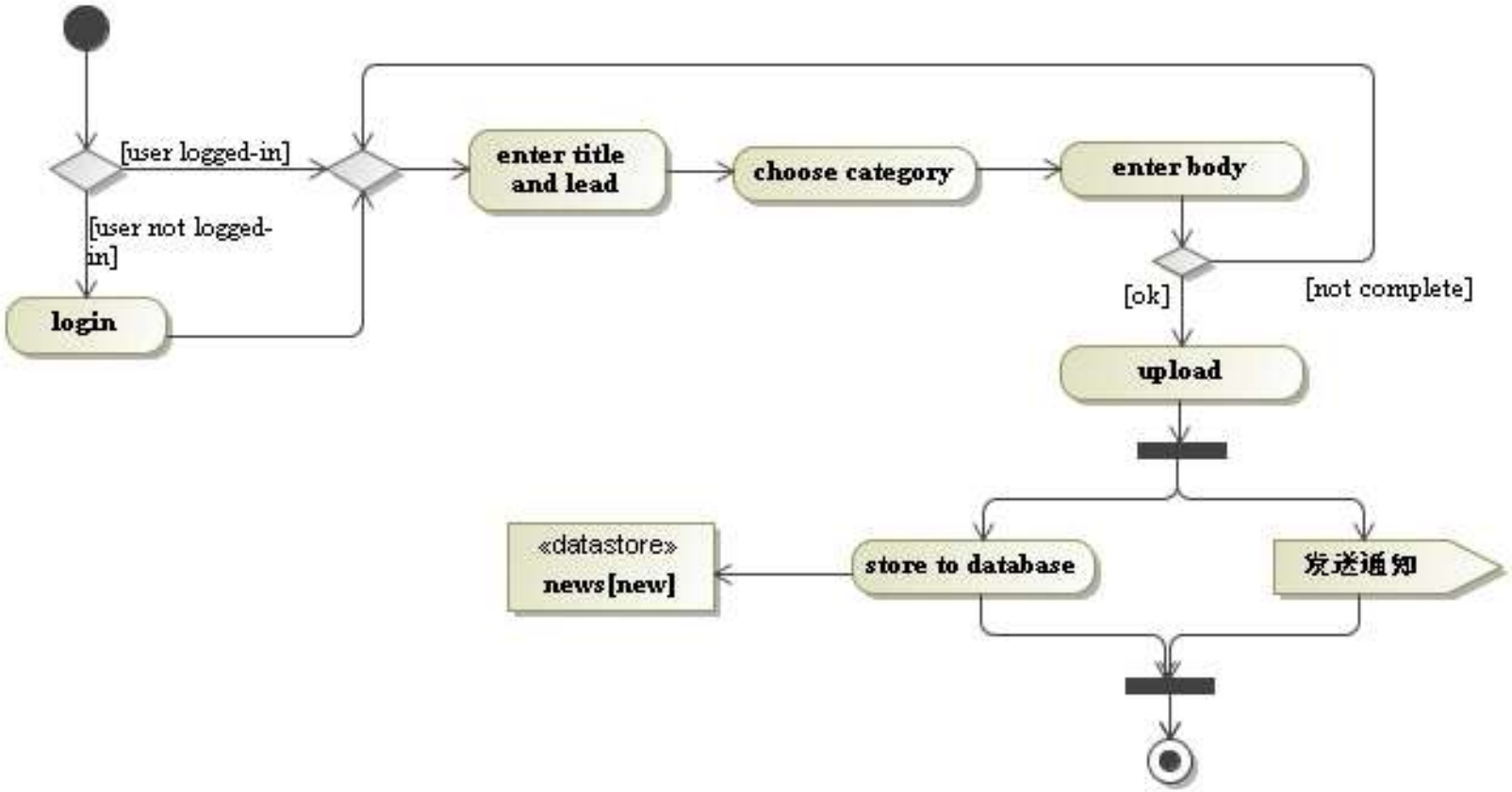


新闻系统用例图


News System
«navigation»
browse news
search news
register
«extend»
login
«extend»
comment
«extend»
submit news
User
Author
«navigation»
browse review result
review news
«navigation»
list submitted news
«extend»
Reviewer
«navigation»
list accept/reject news
«include»
publish news
Administrator
maintain system

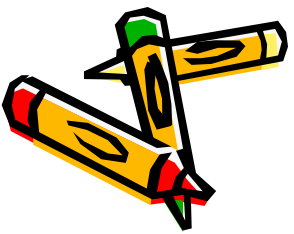
绘制活动图

- 表达复杂业务逻辑的用例进一步精化建模



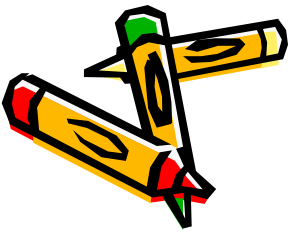
新闻投稿活动图

内容建模

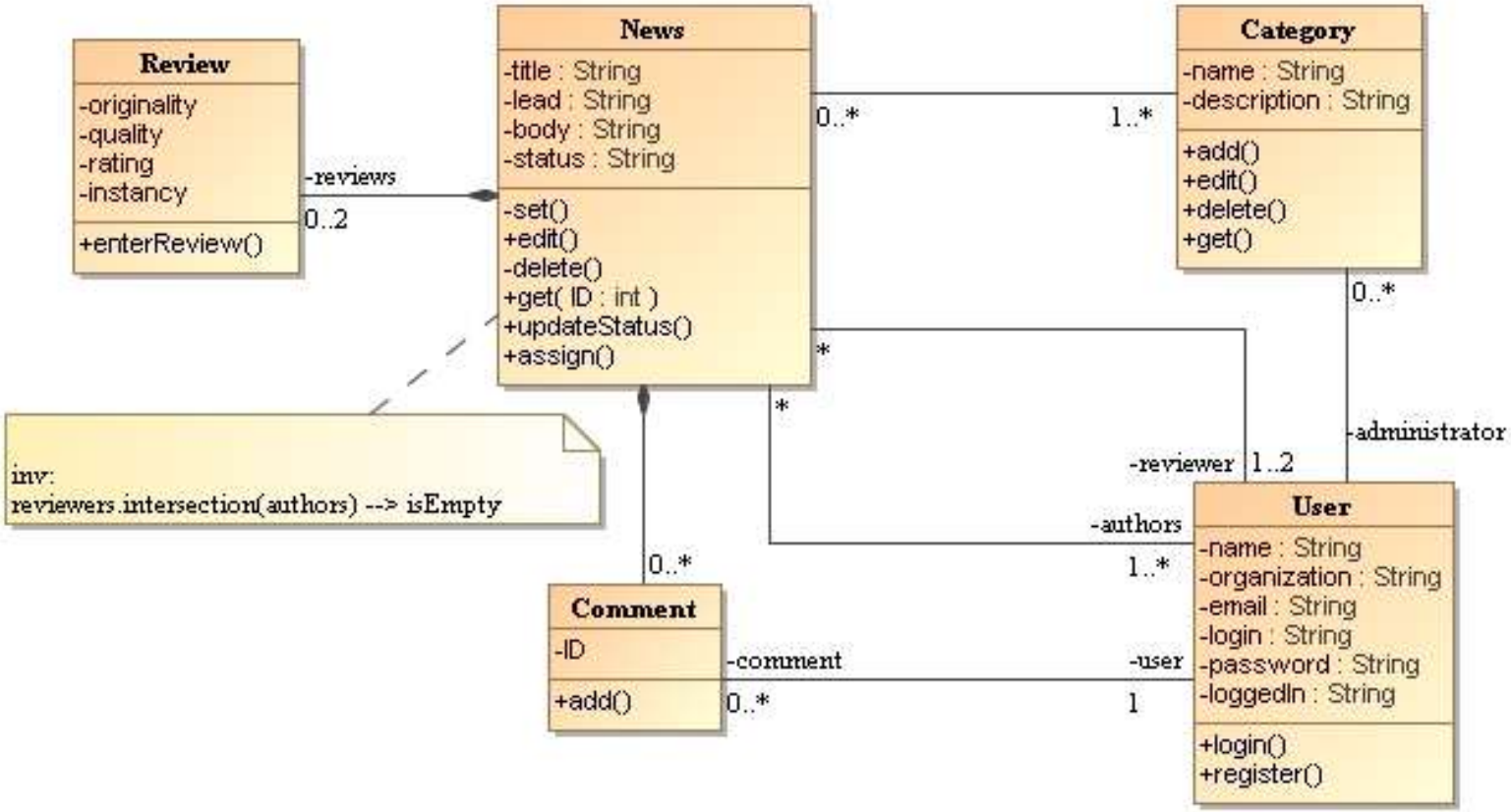


内容建模

- 目标是将从需求工程中决定的Web应用信息和功能需求转换为模型
  - 图形化信息的结构(i.e., information objects) 和行为方面
  - 不关注导航
- 内容的结构方面
  - 问题域模型
  - UML类图
- 内容的行为方面
  - 根据Web应用的类型和复杂程度
  - UML状态图(state charts)/交互图(interaction diagrams)
- 只是内容层, i.e.,
  - No hypertext or presentation建模!

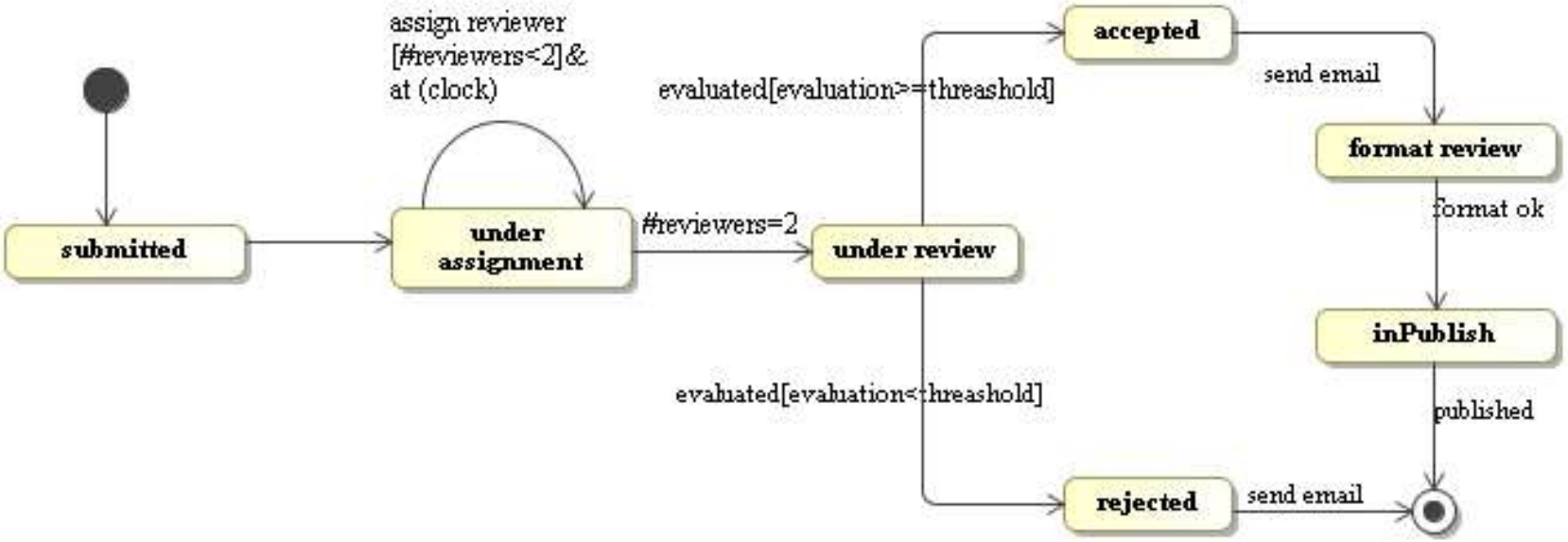


内容建模:静态建模

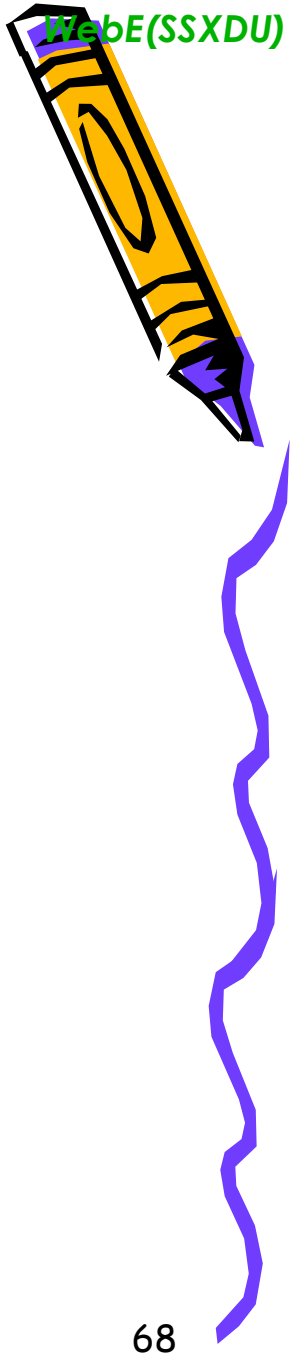


新闻系统类图

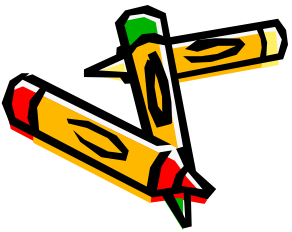
内容建模:动态建模



新闻类的状态图



超文本建模



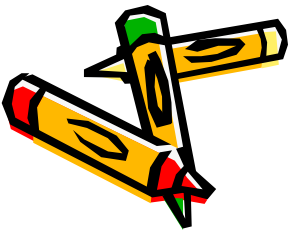
超文本建模

- 目标是通过Web应用的内容构建导航，因此也是导航建模
  - 建模节点和超文本结构
  - 建模导航路径

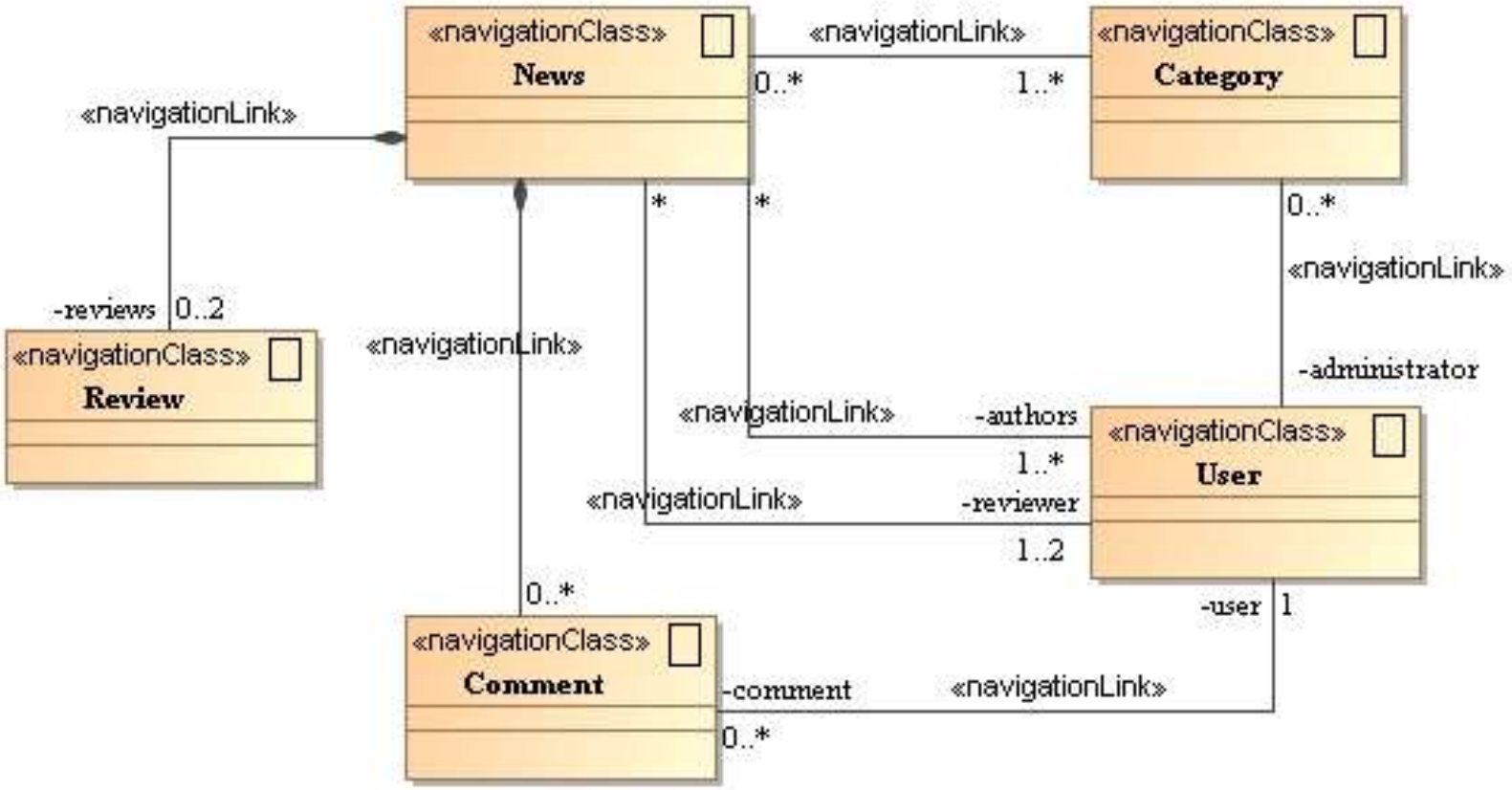
- 产出
  - 超文本/导航结构模型：导航类图
    - 表述超文本的结构, i.e.,内容模型可以通过导航来访问
  - 超文本访问模型(Access model)
    - 使用访问模型中的访问元素精化超文本结构模型
  - *针对用户角色, i.e., 新闻作者, 审稿人, 管理员.*

超文本建模：静态建模

- 以内容模型为基础
  - 类和对象在超文本中表示为节点
  - 转换规则和按需添加的一些链接
- 特定的符号：如UWE
  - «navigation class»：导航节点
  - «navigation link»：导航链接
  - «process link»：过程链接
  - «external link»：外部链接



超文本建模：静态建模

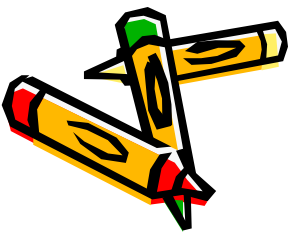


管理员视图的超文本结构模型

建模超文本结构模型的步骤

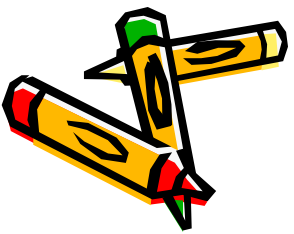
1. 为每个导航相关的内容类定义导航类
2. 为内容模型相关的关联、聚合和组合定义导航链接
3. 在内容模型中添加重数和角色名
4. 根据需求分析的情景添加额外的导航链接
5. 添加额外的导航链接作为快捷方式

(note: 1, 2 和3可以自动化进行)



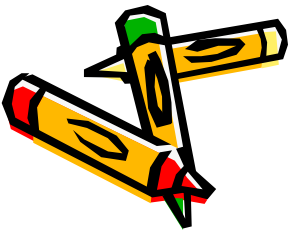
超文本建模：动态建模

- 超文本访问模型：如何通过导航访问到节点
- 访问结构:UWE
  - <<menu>>访问不同类型的节点
  - <<index>>访问一个节点对象集
  - <<query>>搜索节点
  - <<guided tour>> 按序在节点之间漫游
  - <<home>>指向Web应用的主页
  - <<landmark>>陆标，指向一个所有节点都可达到节点

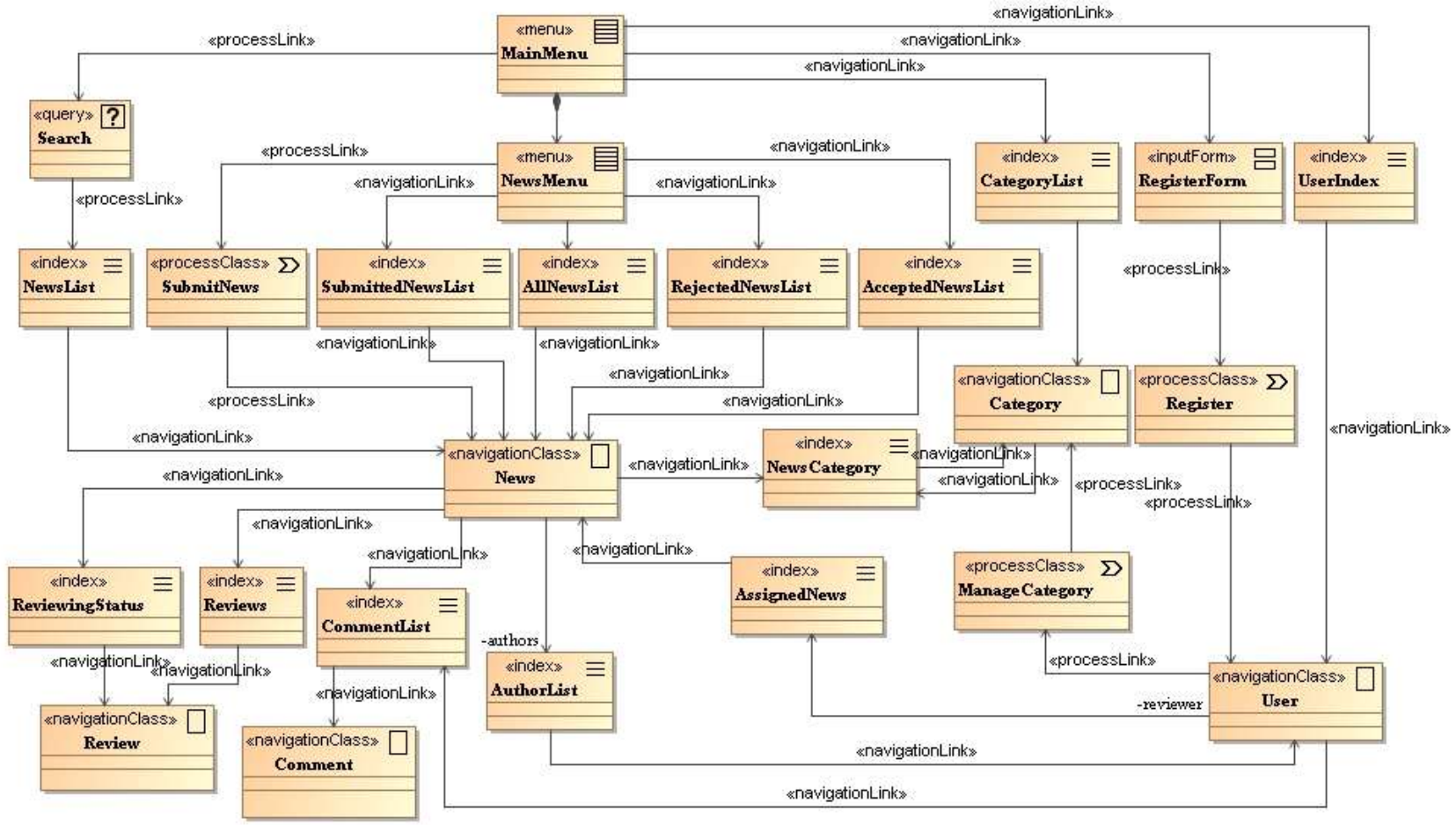


超文本访问模型

- (自动)从超文本结构模型导出访问模型的方法
  - 对导航链接度大于1的引入<<index>>
  - 对外向导航链接大于1的每个类，引入<<menu>>
  - 用角色名作为外向导航链接的菜单项

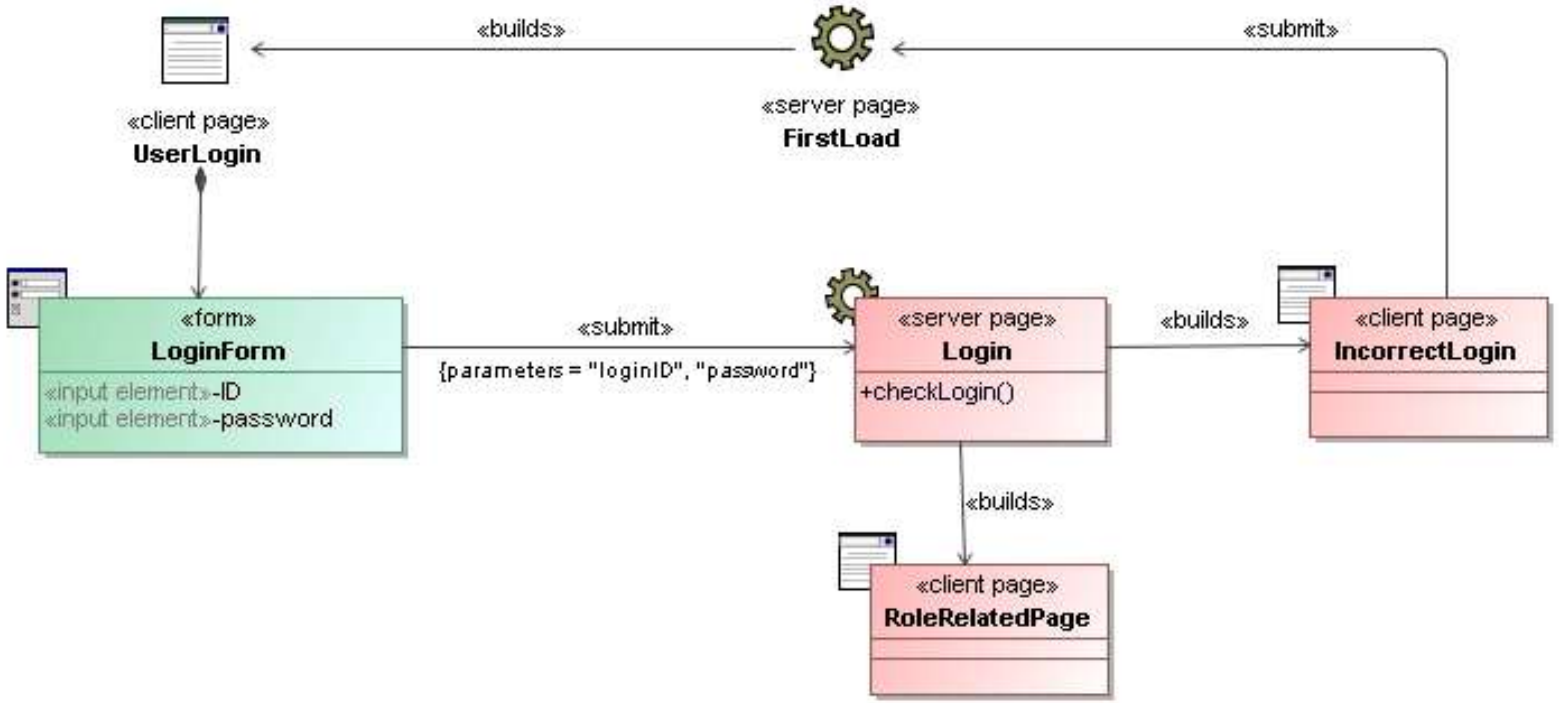


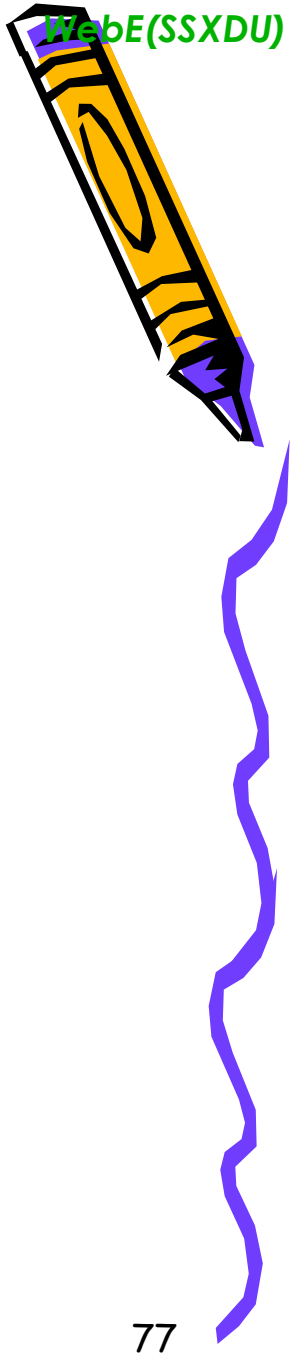
超文本建模：UWE超文本访问模型



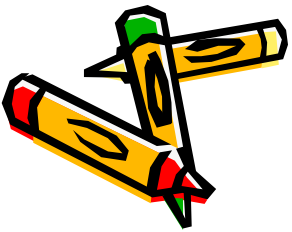
简化访问模型

超文本建模： Web模型


«builds»
«submit»
«client page»
UserLogin
«server page»
FirstLoad
«form»
LoginForm
«input element»-ID
«input element»-password
«submit»
{parameters = "loginID", "password"}
«server page»
Login
+checkLogin()
«builds»
«client page»
IncorrectLogin
«builds»
«client page»
RoleRelatedPage

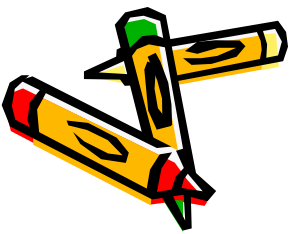


展示建模



展示建模

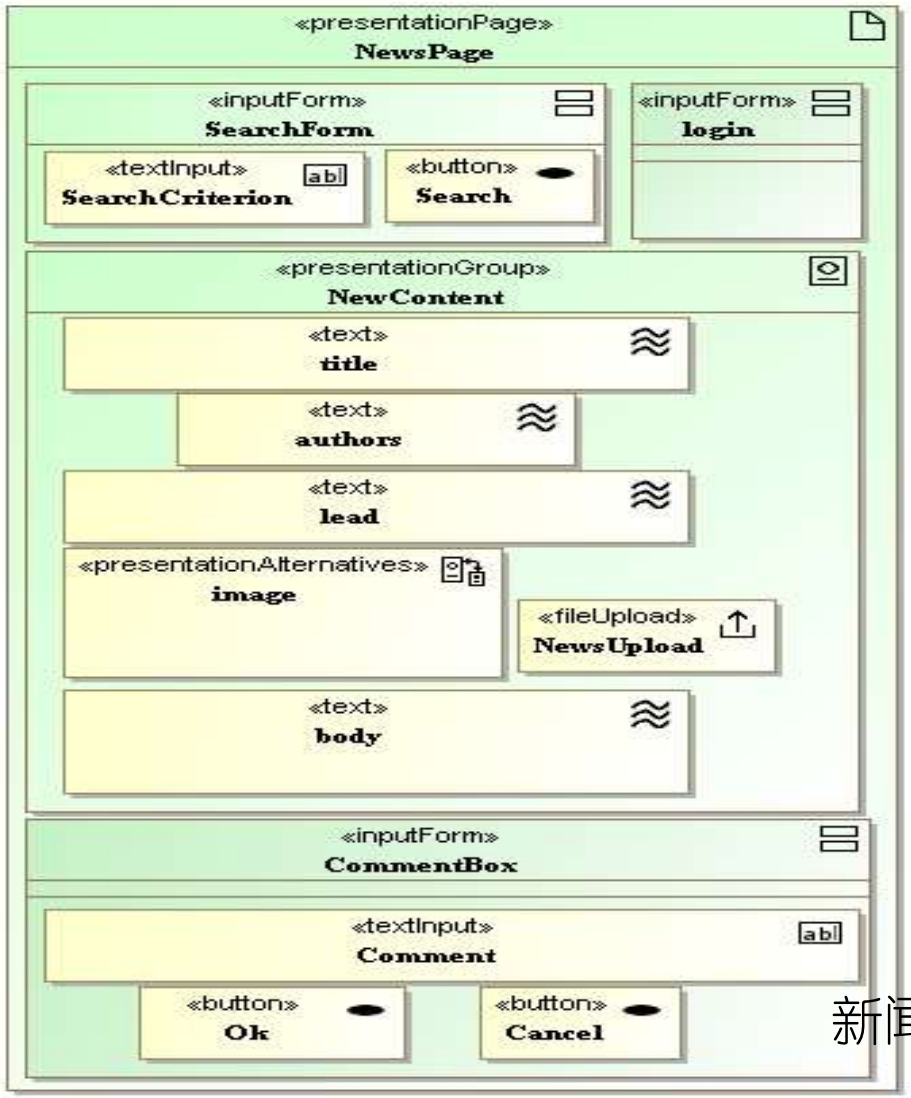
- 目标
  - 对Web页面的结构和行为进行建模
    - 简单且可自我解释
- 特性
  - Web页面的层次组织，包含展示元素
- 产出
  - 静态展示模型
    - 通过建模页面中的通用元素（如，页眉和页脚）产生一致的展示概念，并且展示每个页面的组成以及其中包含的字段、文本、图片、表单等内容
  - 动态交互模型
    - 用户界面的交互特性，例如点击某个按钮可以激活应用逻辑的某个功能，同时要考虑提供给用户必要的导航路径、用户访问历史等信息



展示建模：UWE静态建模

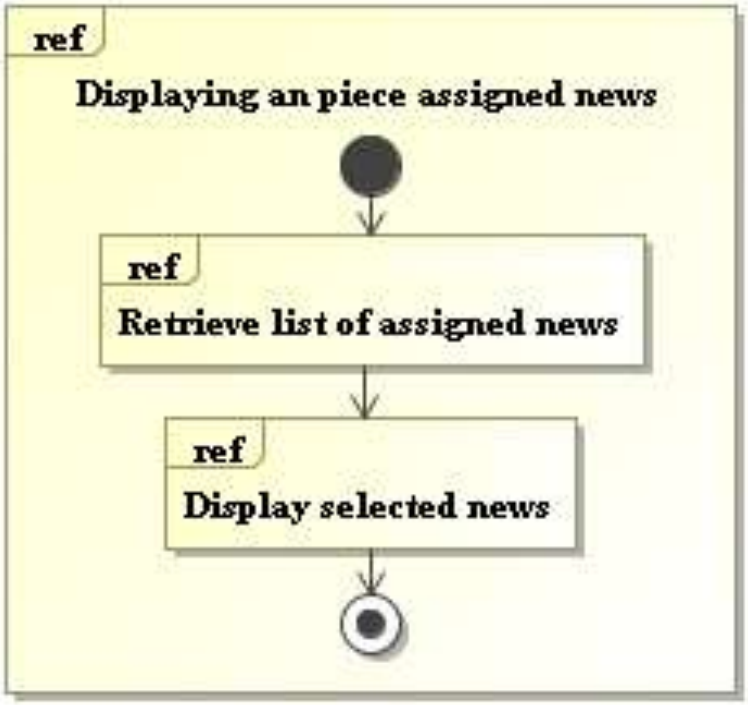
- <<page>>
  - 展示页面，用户看到的最大单位，可以包含不同展示单元
- <<presentation unit>>
  - 展示单元：将一些页面元素组织在一起，表达页面的逻辑片段，表示超文本模型中的一个节点
- 展示元素
  - <<text>>
  - <<image>>
  - <<audio>>
  - …

展示建模：UWE静态建模



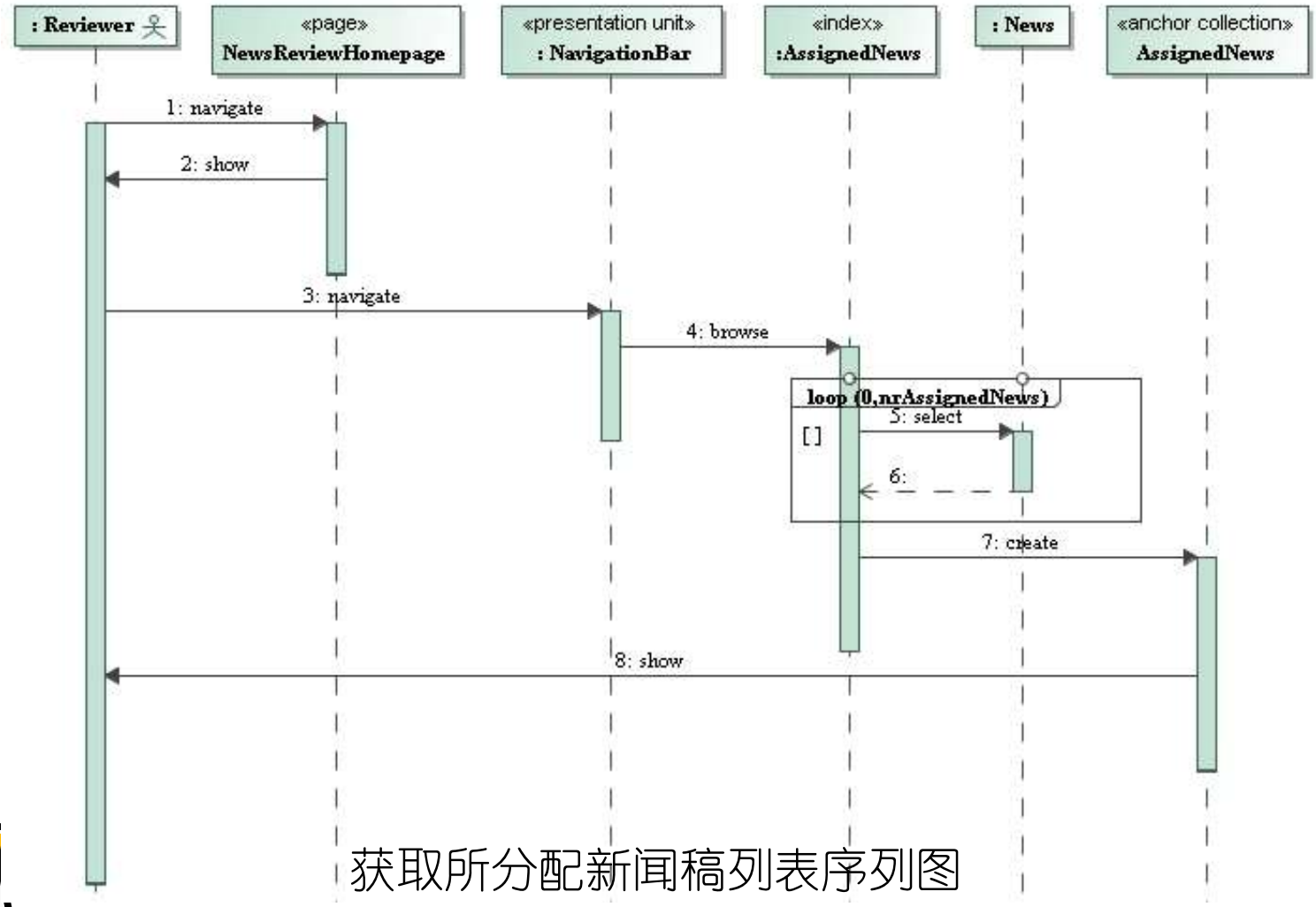
新闻页面的简单展示模型

展示建模：动态建模



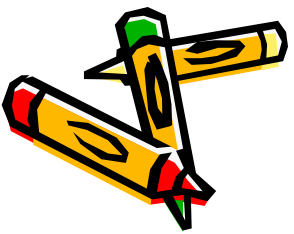
显示所分配新闻稿交互图

展示建模：动态建模



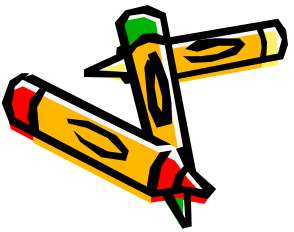
获取所分配新闻稿列表序列图

适应性建模

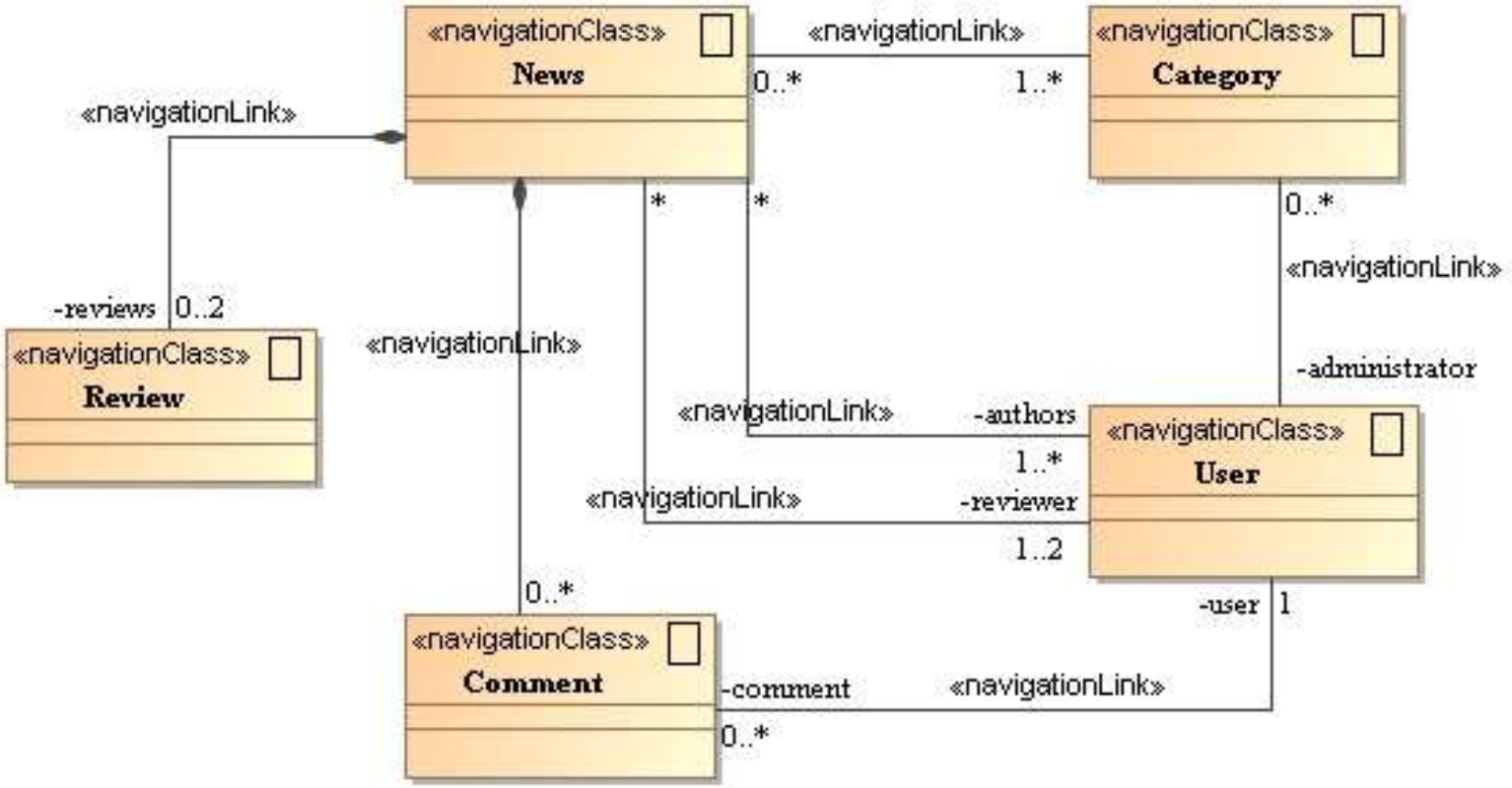


适应性建模（Customization Modeling）

- 目标
  - 根据用户上下文（context）特性，给用户提供合适的展示
- 方法
  - 静态建模: 不同上下文不同模型
  - 动态建模: 一个模型+ 适应性规则
- 产出
  - 适应性模型



适应性建模：静态建模



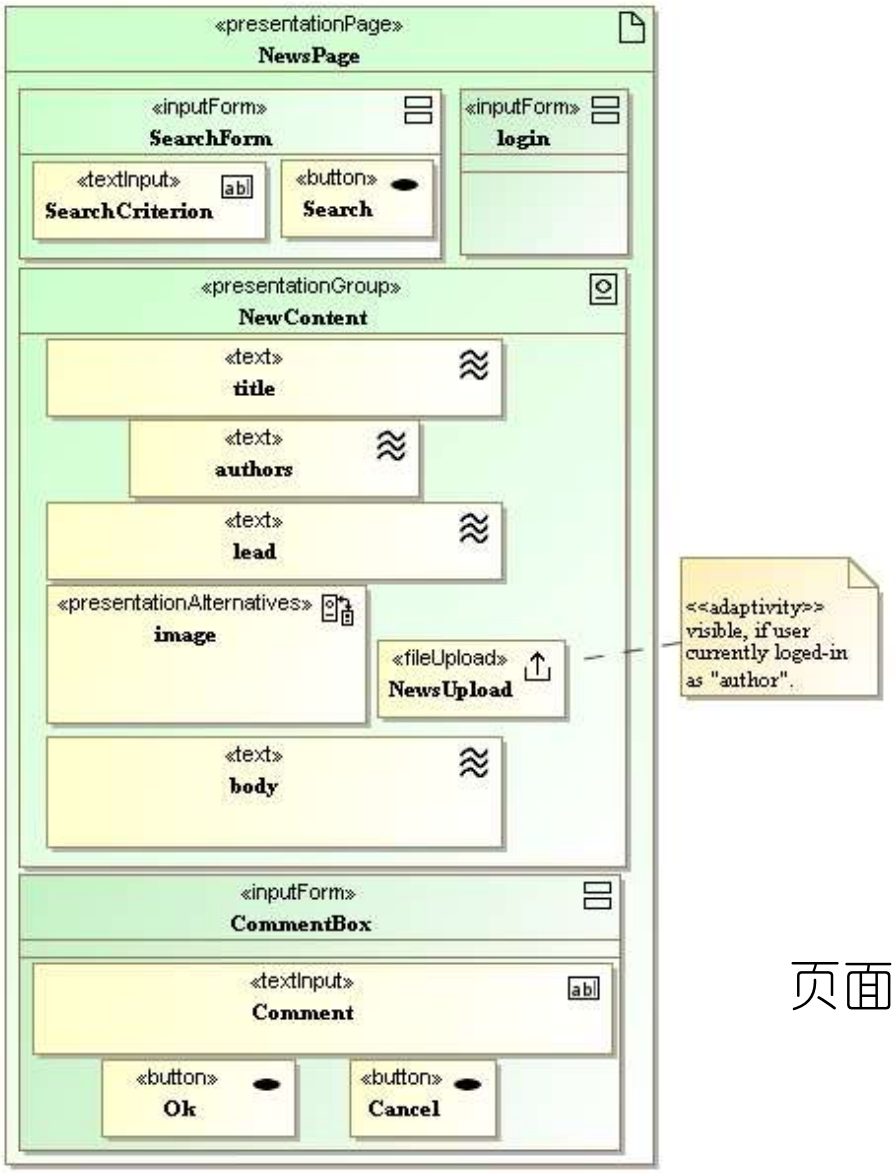
新闻管理员用户角色的超文本结构

多种模型的变化以及可能的模型数量的增加

适应性建模：动态建模

超文本模型中索引的动态适应

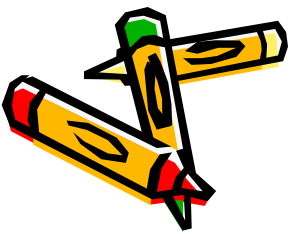
适应性建模：动态建模

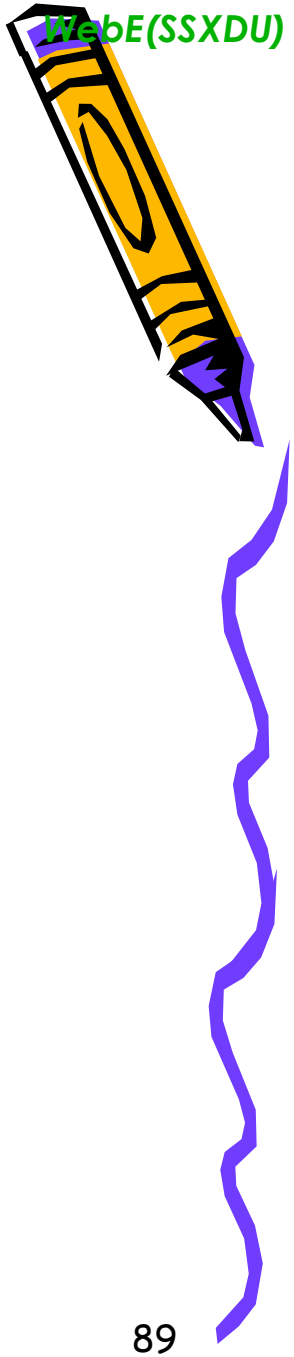


页面的动态适应

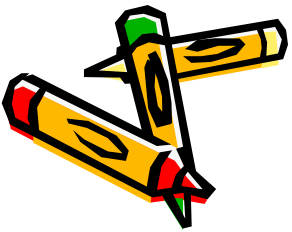
适应性建模：UWE

- AOM进行适应性建模
  - 使得系统功能和个性化方面系统地进行分离，并减少冗余
  - UWE区分内容、超文本和展示层的不同个性化建模
  - 建模元素«Navigation Annotation»



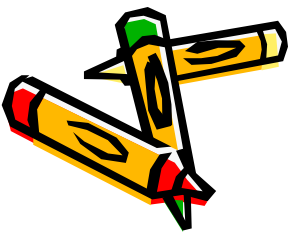


总结与展望



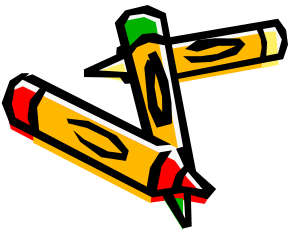
总结

- Web应用建模的特性包括层、方面、阶段和适应性4个方面
- 模型驱动开发
- 建模方法和支持工具



展望

- 统一Web建模语言（UWML）？
- 支持MDD的工具
  - 支持建模符号
  - 支持模型驱动开发的开发过程
- 建模方法
  - 定义清楚的指南和方法
  - 在很大程度上考虑敏捷方法
  - 与自动生成相协调
- 考虑运行平台和发布框架
- 模型重用
- 工作流建模
- 上下文：移动设备
- Web服务的建模



Project Task：Task4

- Web应用建模
  - 根据本章内容，完成Web应用建模
- 展示的主要内容之一
  - 10 minutes
  - ~8 slides