Modeling the Conceptual Domain

CSCI 4448/5448: Object-Oriented Analysis & Design

Lecture 15

Acknowledgement & Materials Copyright

- I’d like to start by acknowledging Dr. Ken Anderson
- Ken is a Professor and the Chair of the Department of Computer Science
- Ken taught OOAD on several occasions, and has graciously allowed me to use his copyrighted material for this instance of the class
- Although I will modify the materials to update and personalize this class, the original materials this class is based on are all copyrighted © Kenneth M. Anderson; the materials are used with his consent; and this use in no way challenges his copyright

Working at the Conceptual Level

- One of the first steps in developing an OO design is discovery of the entities in your system that have responsibilities – that will turn into classes and objects
- A conceptual level of a university may include entities such as:
  - Students, instructors, professors, staff, classes, transcripts, registrar, buildings, classrooms, etc.
- How can you determine which of these entities will become part of your application’s class structures?

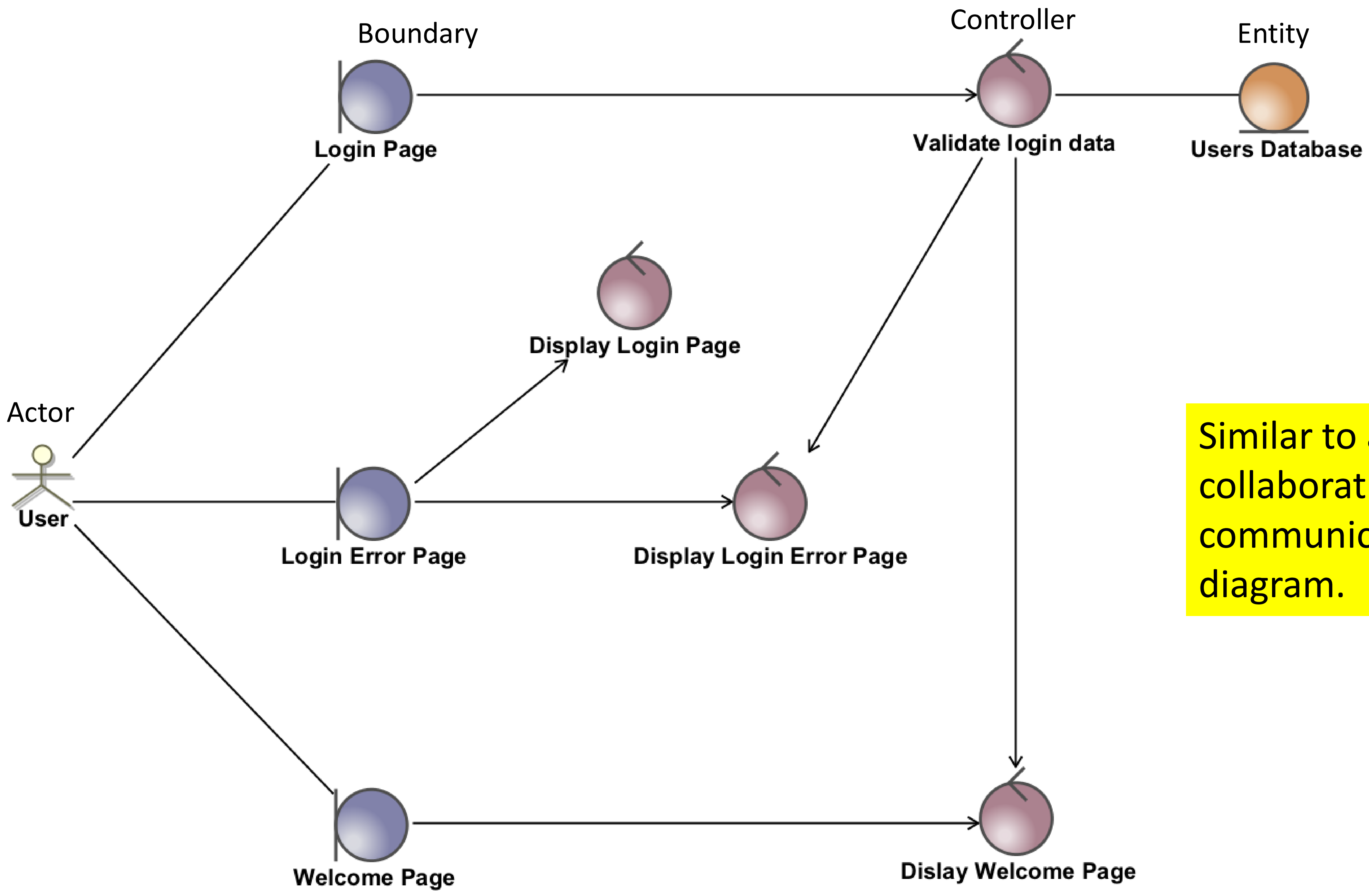
Conceptual Modeling Approaches

- You’ve seen one of these modeling approaches already –
  - Developing UML class and object diagrams (with other diagrams in support)
- Today we’ll look at some other approaches, including:
  - Robustness diagrams
  - Object Role Model (ORM) diagrams
  - Class Responsibility Collaborator (CRC) models
  - Logical data models (LDMs)
  - Analysis patterns
    - All from The Object Primer, Scott Ambler, 2005, Cambridge
- Later, we’ll also look at
  - Design Pattern-Driven Design
    - aka Thinking in Patterns
  - Commonality and Variability Analysis
  - Analysis Matrix

Robustness Diagrams

- This is an approach based on analysis of use cases
- Analyze the steps of use cases to ensure consistency with other use cases in your overall model
- You're trying to confirm the robustness of the use requirements for the system you're building
- Add Actors
- Add Boundary Elements for major UI elements (screens, reports)
  - Only talk to controllers and actors
- Add Entities for business concepts/support
  - Only talk to controllers
- Add Controllers for process management
  - Can talk to controllers, boundary, and entity objects, not actors
- Optionally add Use Case references to bridge activities on diagrams
  - https://docs.nomagic.com/display/MD190/Robustness+diagram

Robustness Diagrams



Similar to a UML collaboration or communication diagram.

- https://docs.nomagic.com/display/MD190/Robustness+diagram

Object Role Modeling

- A powerful method for designing and querying database models at the conceptual level, where the application is described in terms easily understood by non-technical users
- In practice, ORM data models often capture more business rules, and are easier to validate and evolve than data models in other approaches
- From http://www.orm.net/

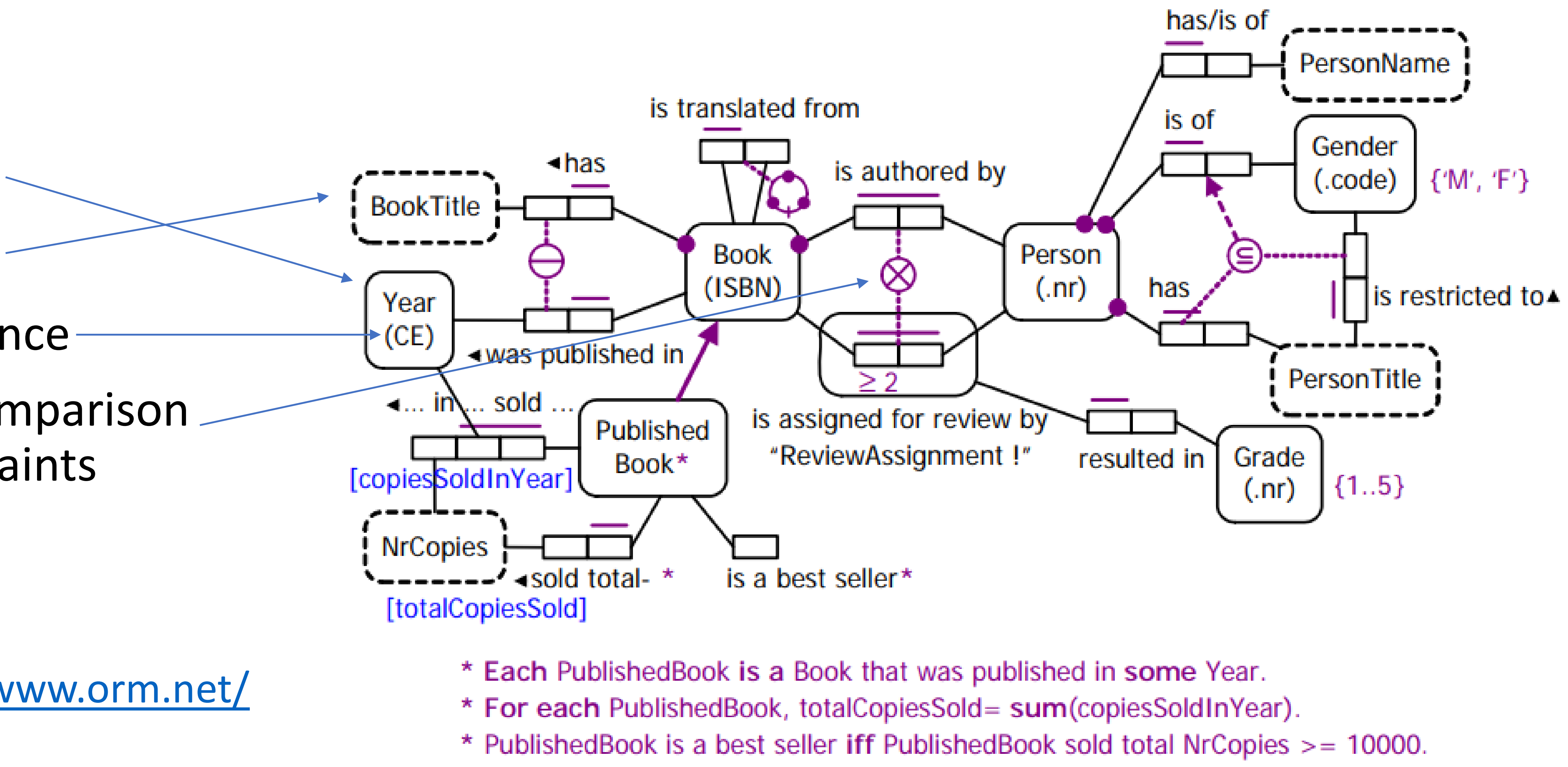
- I find ORM models to get complex quickly. I believe higher level diagrams, such as UML class diagrams are likely a better way to capture conceptual designs…

Object Role Model (ORM) diagrams

- Depicts objects, relationships, roles, constraints, and examples

- Entity
- Value
- Reference
- Set Comparison Constraints

- http://www.orm.net/

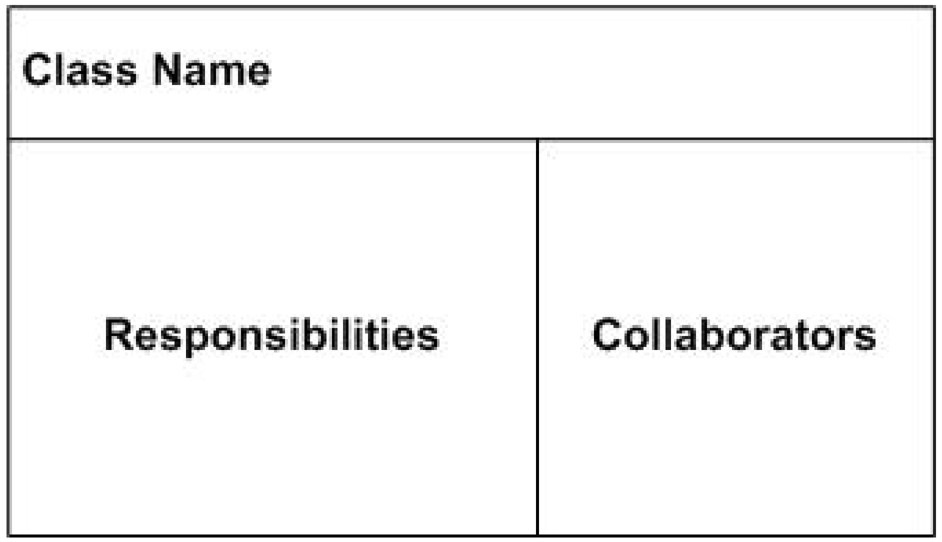


* Each PublishedBook is a Book that was published in some Year.
* For each PublishedBook, totalCopiesSold= sum(copiesSoldInYear).
* PublishedBook is a best seller iff PublishedBook sold total NrCopies >= 10000.

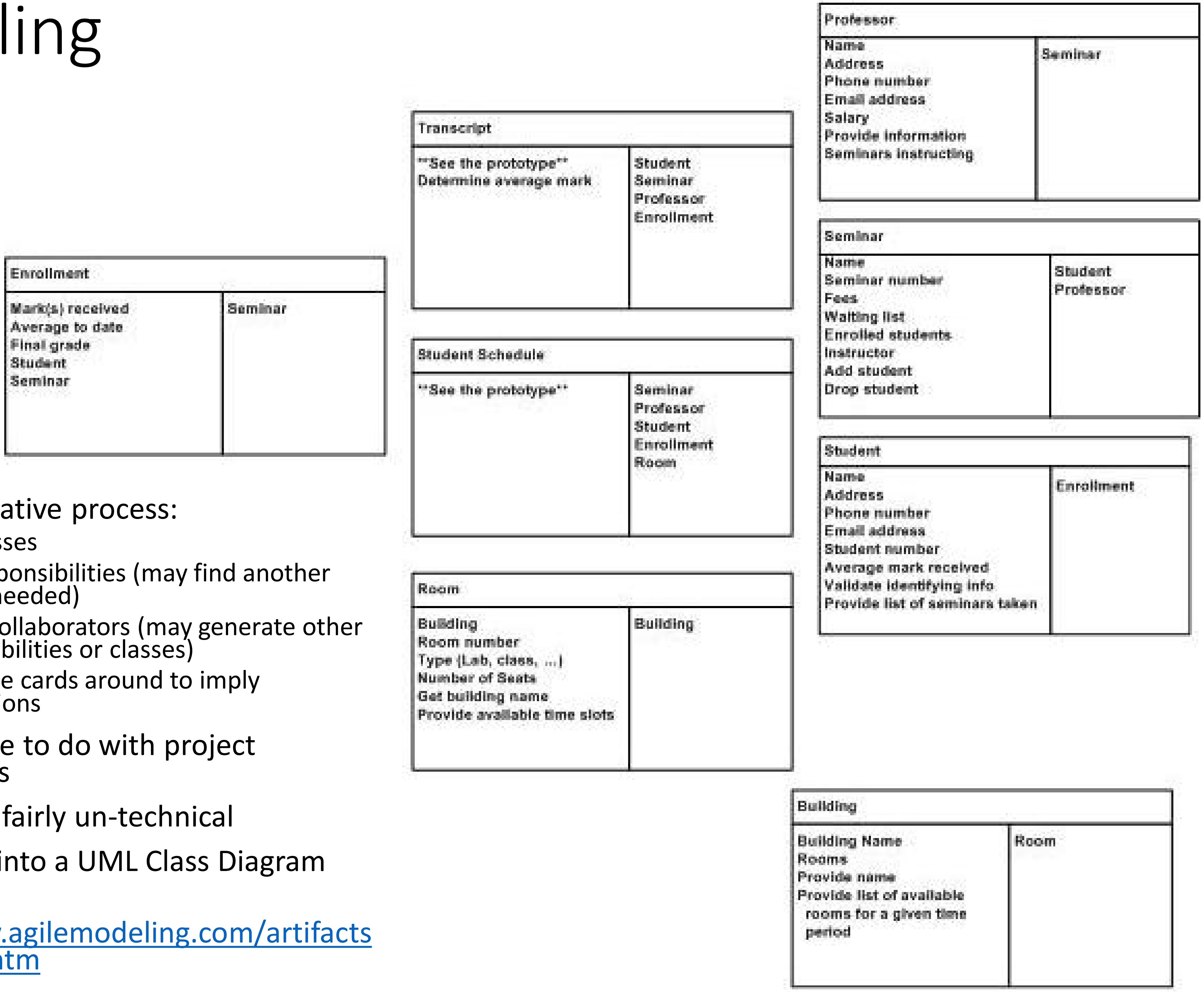
Fig. 2. An ORM schema for a book publishing domain

Class Responsibility Collaborator (CRC) Modeling

- Initially a teaching concept, became a modeling approach
- Classes are objects, people, places, things…
- Responsibilities are anything a class knows or does.
- Collaborators are anything you need to interact with to perform a responsibility
- From http://www.agilemodeling.com/artifacts/crcModel.htm



Class Responsibility Collaborator (CRC) Modeling



Professor	
Name Address Phone number Email address Salary Provide information Seminars instructing	Seminar

Transcript	
See the prototype Determine average mark	Student Seminar Professor Enrollment

Enrollment	
Mark(s) received Average to date Final grade Student Seminar	Seminar

Seminar	
Name Seminar number Fees Waiting list Enrolled students Instructor Add student Drop student	Student Professor

Student Schedule	
See the prototype	Seminar Professor Student Enrollment Room

Student	
Name Address Phone number Email address Student number Average mark received Validate identifying info Provide list of seminars taken	Enrollment

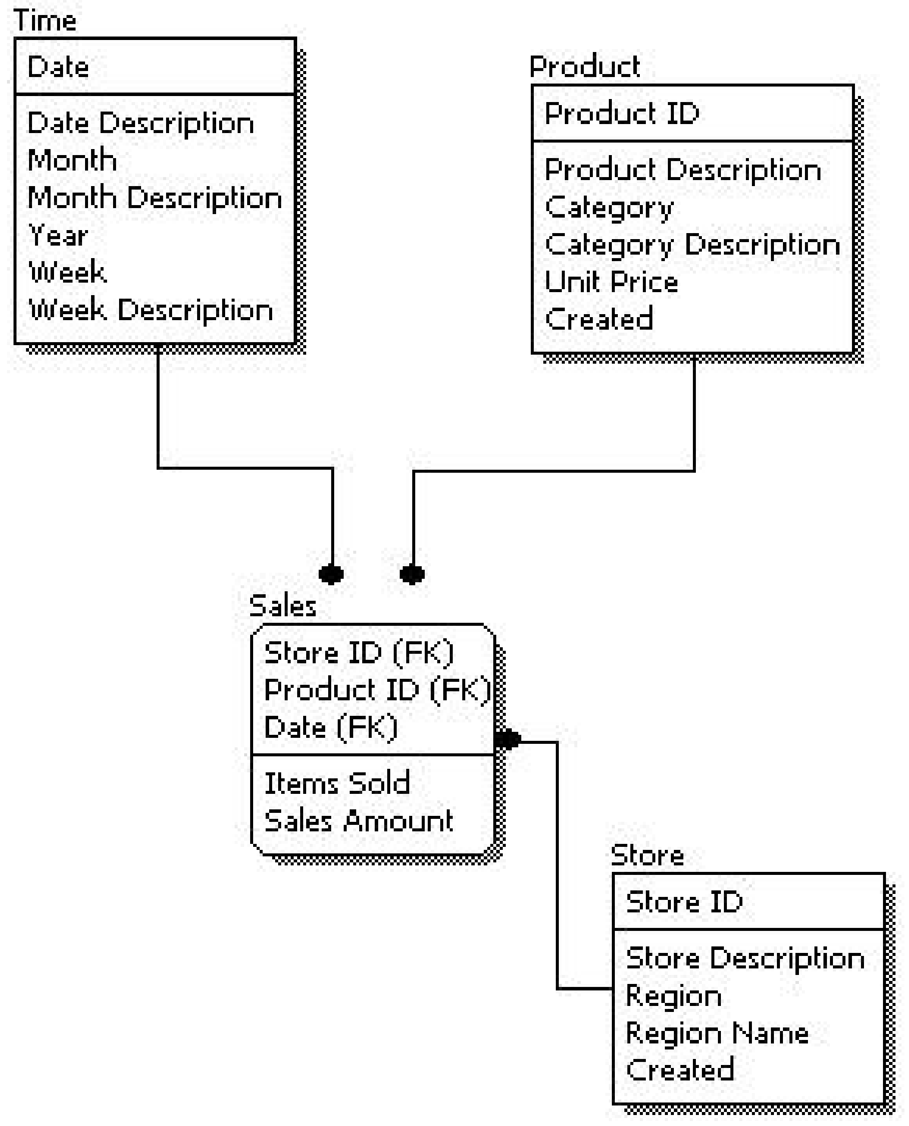
Room	
Building Room number Type (Lab, class, ...) Number of Seats Get building name Provide available time slots	Building

Building	
Building Name Rooms Provide name Provide list of available rooms for a given time period	Room

- General iterative process:
  - Find classes
  - Find responsibilities (may find another class is needed)
  - Define collaborators (may generate other responsibilities or classes)
  - Move the cards around to imply connections
- Nice exercise to do with project stakeholders
- Can remain fairly un-technical
- Will evolve into a UML Class Diagram
- From http://www.agilemodeling.com/artifacts/crcModel.htm

Logical Data Models (LDMs)

- Data focused model to describe data in detail without specifics of implementation in a database
- Includes all entities and relationships, with attributes and keys identified
- Database normalization can be applied to these models
- https://www.1keydata.com/datawarehousing/logical-data-model.html

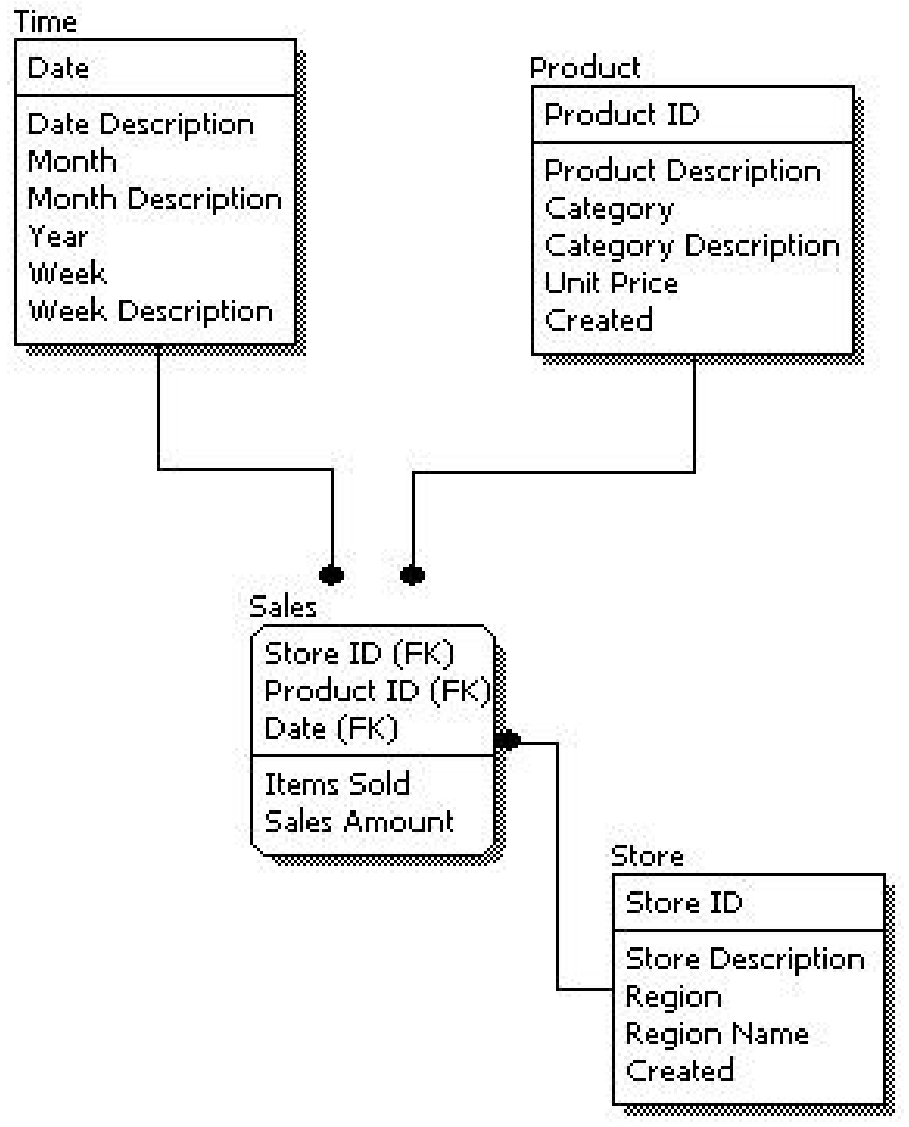


Logical Data Models (LDMs)

Typical process:

1. Specify primary keys for all entities
2. Find the relationships between different entities.
3. Find all attributes for each entity.
4. Resolve many-to-many relationships.
5. Normalization.

- https://www.1keydata.com/datawarehousing/logical-data-model.html



Analysis (or Process) Patterns

- A process pattern is a pattern which describes a proven, successful approach and/or series of actions for developing software
- Process Patterns have at least three types:
  - Task Process Patterns – steps to perform a typical task
    - Ex: Technical Review, Reuse First
  - Stage Process Patterns
    - Ex: Program, Rework
  - Phase Process Patterns
    - Ex: Initiate, Delivery
- Two published pattern books available
- Good example of an alternate pattern set for a different (if adjacent) discipline
- http://www.ambysoft.com/processPatternsPage.html

Analysis (or Process) Patterns

- Book outlines the serial phase steps in OO Development:
- Chapter 1: Introduction to the Object-Oriented Software Process
- Chapter 2: The Initiate Phase
- Chapter 3: The Define and Validate Initial Requirements Stage
- Chapter 4: The Define Initial Management Documents Stage
- Chapter 5: The Justify Stage
- Chapter 6: The Define Infrastructure Stage
- Chapter 7: The Construct Phase
- Chapter 8: The Model Stage
- Chapter 9: The Program Stage
- Chapter 10: The Test In The Small Stage
- Chapter 11: The Generalize Stage
- Chapter 12: Towards More Process Patterns.

- http://www.ambysoft.com/books/processPatterns.html

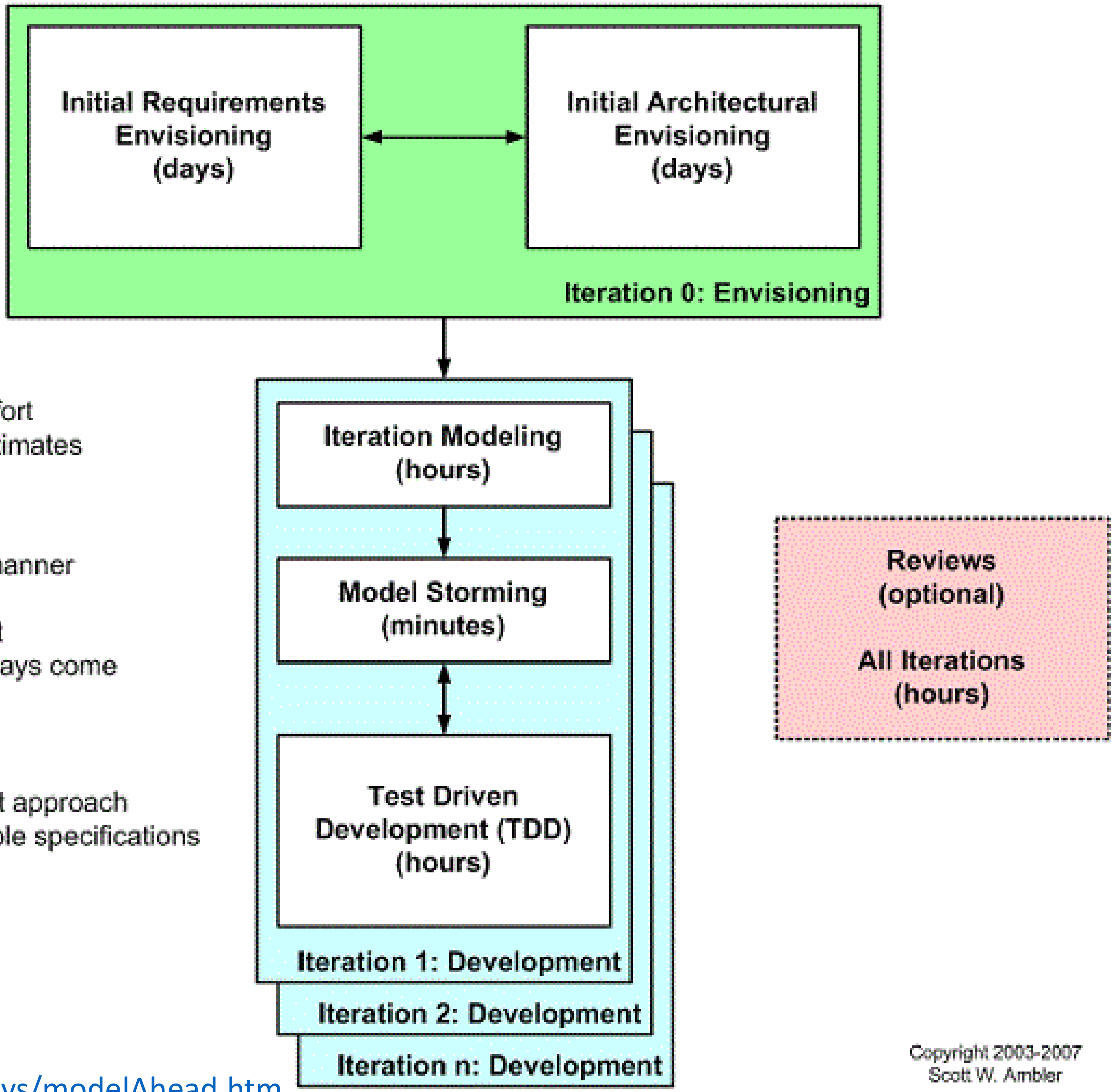
Model A Bit Ahead Pattern

- Identify the high-level scope
- Identify initial "requirements stack"
- Identify an architectural vision

- Modeling is part of iteration planning effort
- Need to model enough to give good estimates
- Need to plan the work for the iteration

- Work through specific issues on a JIT manner
- Stakeholders actively participate
- Requirements evolve throughout project
- Model just enough for now, you can always come back later

- Develop working software via a test-first approach
- Details captured in the form of executable specifications



- http://www.agilemodeling.com/essays/modelAhead.htm

Summary

- We will look at some other OO design approaches, but these are some that have been around for some time
- I think the **UML modeling cycle** is still a solid way forward
- I like **CRC cards** – can be a less technical way to explore a class/object model with non-technical stakeholders, we may see them again…
- **Robustness diagrams** may be an alternate to other UML methods if they appeal to you – mixes UIs and support in with use case models
    - You might also look at UML collaboration/communication diagrams. Good article on them in Code Magazine: https://www.codemag.com/article/0205051/UML-Collaboration-Diagrams
- There are other modeling approaches for databases to consider, **LDMs** are one, there is also the generic Entity-Relationship approach (ER modeling) that is similar and common for databases; as are Data Flow Diagrams (DFD)
- **ORMs** are fairly complex diagrams, you'd need to commit to understanding how they're best developed and applied – not that common in my experience
- **Analysis or process patterns** are just another example of trying to capture best practices in a pattern language, and might be helpful if you were putting together or improving development practices or processes for your team to do OO systems