Principles of Software Construction: Objects, Design and Concurrency

Object-Oriented Analysis

15-214

toad

Christian Kästner Charlie Garrod

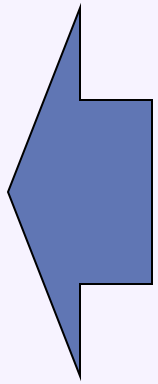
School of Computer Science

institute for SOFTWARE RESEARCH

Learning Goals

- Ability to model concepts in a domain and their relationships
- Use an adequate notation to document a domain model
- Distinguish problem and solution space, distinguish domain models from object models and implementation models

The design process

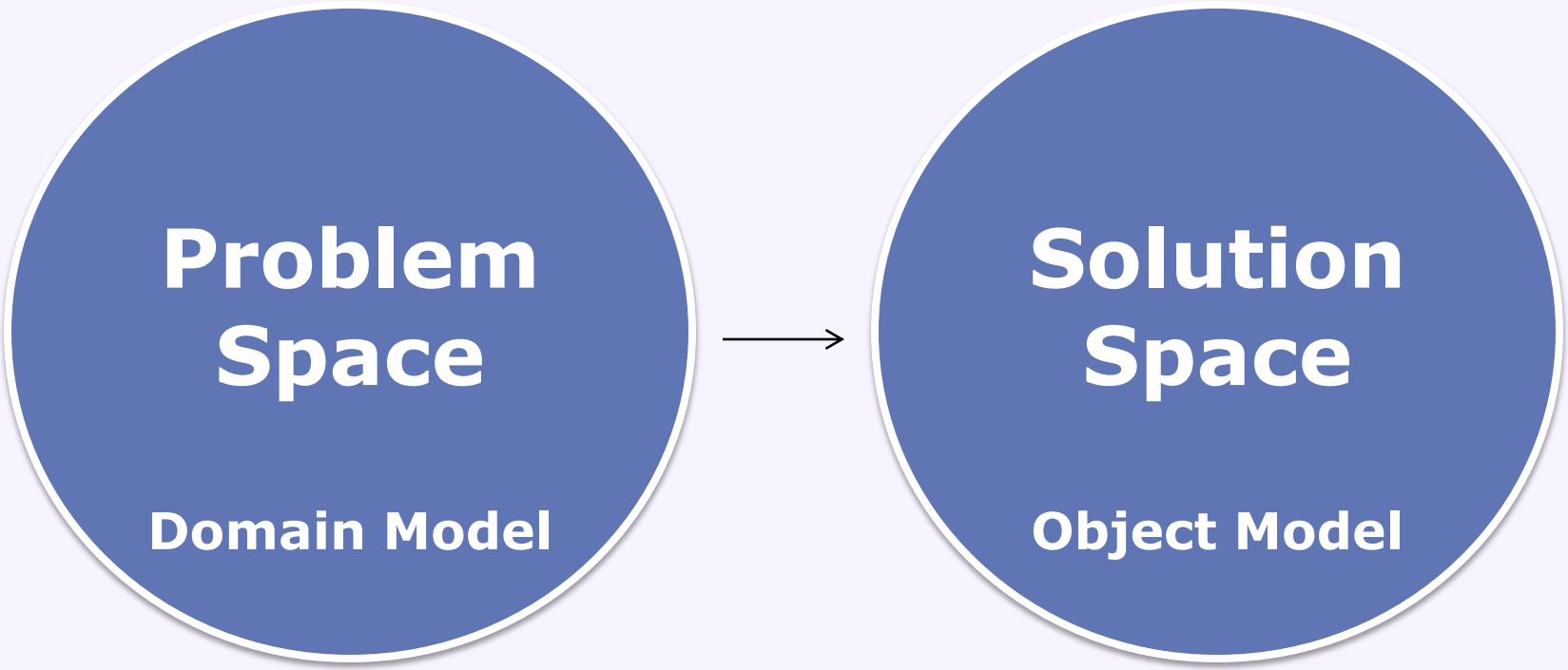


1. Object-Oriented Analysis
   - Understand the problem
   - Identify the key **concepts** and their relationships
   - Build a (visual) vocabulary
   - Create a **domain model** (aka conceptual model)
2. Object-Oriented Design
   - Identify **software classes** and their relationships with *class diagrams*
   - Assign responsibilities (attributes, methods)
   - Explore **behavior** with *interaction diagrams*
   - Explore design alternatives
   - Create an **object model** (aka design model and design class diagram) and **interaction models**
3. Implementation
   - Map designs to code, implementing classes and methods

Object-Oriented Analysis

- Find the concepts in the problem domain
  - Real-world abstractions, not necessarily software objects
- Understand the problem
- Establish a common vocabulary
- Common documentation, big pictutre
- For communication!
- Often using UML class diagrams as (informal) notation

- Starting point for finding classes later



- Real-world concepts
- Requirements, Concepts
- Relationships among concepts
- Solving a problem
- Building a vocabulary

- System implementation
- Classes, objects
- References among objects and inheritance hierarchies
- Computing a result
- Finding a solution

Running Example

Running Example

- **Point of sale** (**POS**) or **checkout** is the place where a retail transaction is completed. It is the point at which a customer makes a payment to a merchant in exchange for goods or services. At the point of sale the merchant would use any of a range of possible methods to calculate the amount owing - such as a manual system, weighing machines, scanners or an electronic cash register. The merchant will usually provide hardware and options for use by the customer to make payment. The merchant will also normally issue a receipt for the transaction.
- For small and medium-sized retailers, the POS will be customized by retail industry as different industries have different needs. For example, a grocery or candy store will need a scale at the point of sale, while bars and restaurants will need to customize the item sold when a customer has a special meal or drink request. The modern point of sale will also include advanced functionalities to cater to different verticals, such as inventory, CRM, financials, warehousing, and so on, all built into the POS software. Prior to the modern POS, all of these functions were done independently and required the manual re-keying of information, which resulted in a lot of errors.

http://en.wikipedia.org/wiki/Point_of_sale

Read description carefully, look for nouns and verbs

- **Point of sale** (**POS**) or **checkout** is the place where a retail transaction is *completed*. It is the point at which a customer makes a payment to a merchant in *exchange* for goods or services. At the point of sale the merchant would use any of a range of possible methods to *calculate* the amount owing - such as a manual system, weighing machines, scanners or an electronic cash register. The merchant will usually provide hardware and options for use by the customer to *make* payment. The merchant will also normally *issue* a receipt for the transaction.
- For small and medium-sized retailers, the POS will be customized by retail industry as different industries have different needs. For example, a grocery or candy store will need a scale at the point of sale, while bars and restaurants will need to *customize* the item sold when a customer has a special meal or drink request. The modern point of sale will also include advanced functionalities to cater to different verticals, such as inventory, CRM, financials, warehousing, and so on, all built into the POS software. Prior to the modern POS, all of these functions were done independently and required the manual re-keying of information, which resulted in a lot of errors.

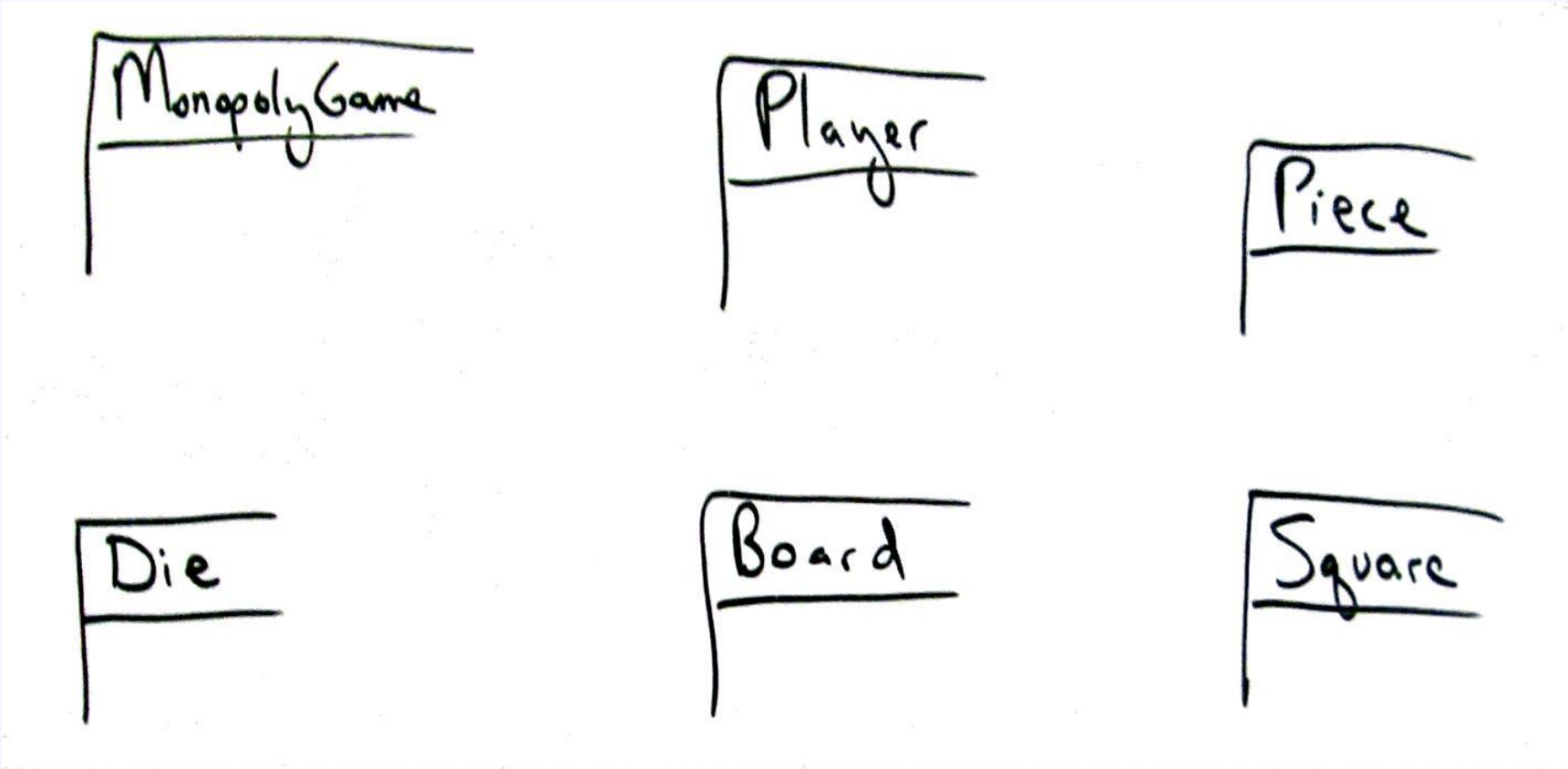
http://en.wikipedia.org/wiki/Point_of_sale

First Steps toward a Conceptual Model

Identify concepts

Register	Item	Store	Sale
Sales LineItem	Cashier	Customer	Ledger
Cash Payment	Product Catalog	Product Description	

Conceptual Model



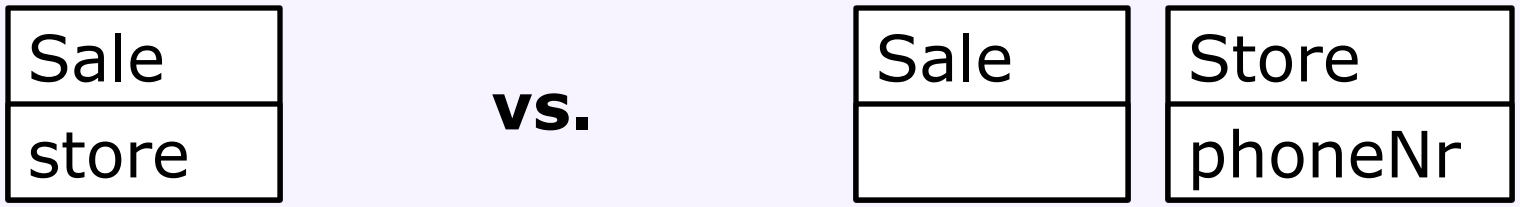
Hints for Identifying Concepts

- Read the requirements description, look for nouns
- Reuse existing models
- Use a category list
  - tangible things: cars, telemetry data, terminals, …
  - roles: mother, teacher, researcher
  - events: landing, purchase, request
  - interactions: loan, meeting, intersection, …
  - structure, devices, organizational units, …
- Analyze typical use scenarios, analyze behavior
- Brainstorming
- Select larger set first, narrow down later if necessary

Organize Concepts

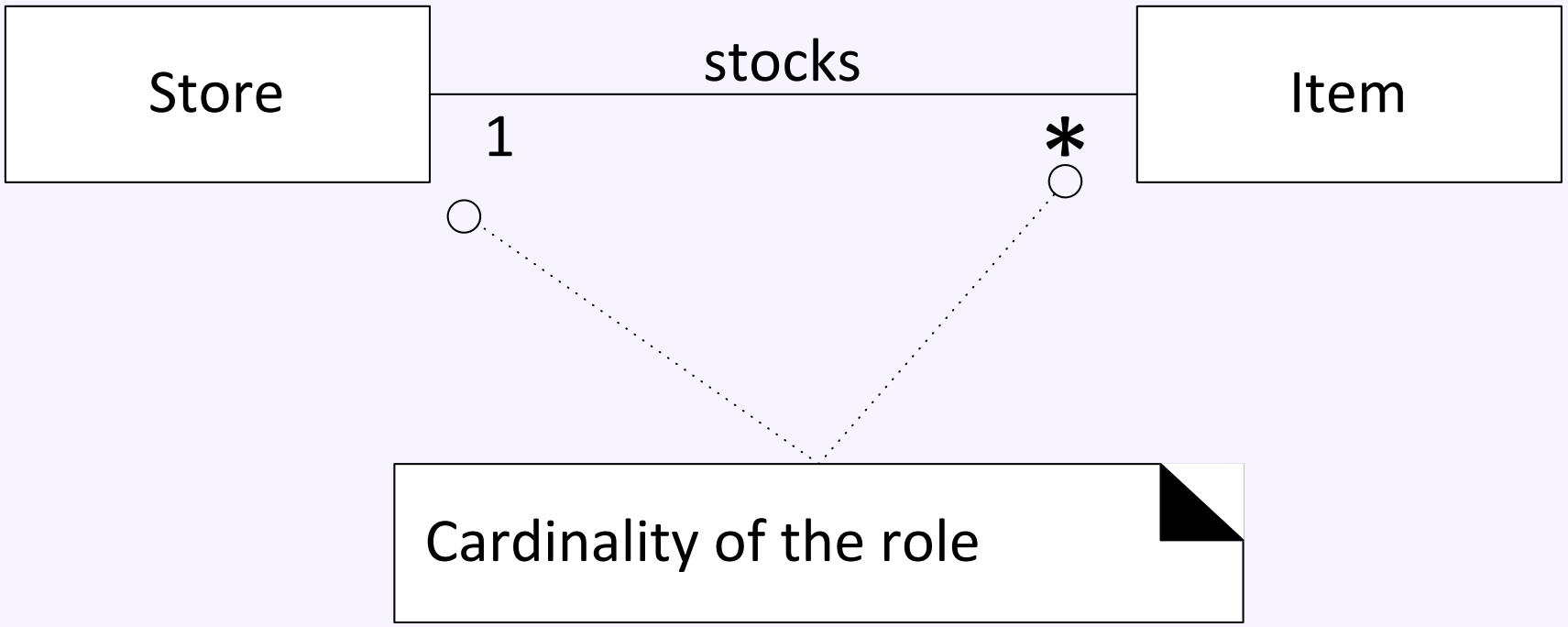
- Identify related elements
- Model relationships through inheritance ("is a") or associations ("related to")

Classes vs. Attributes



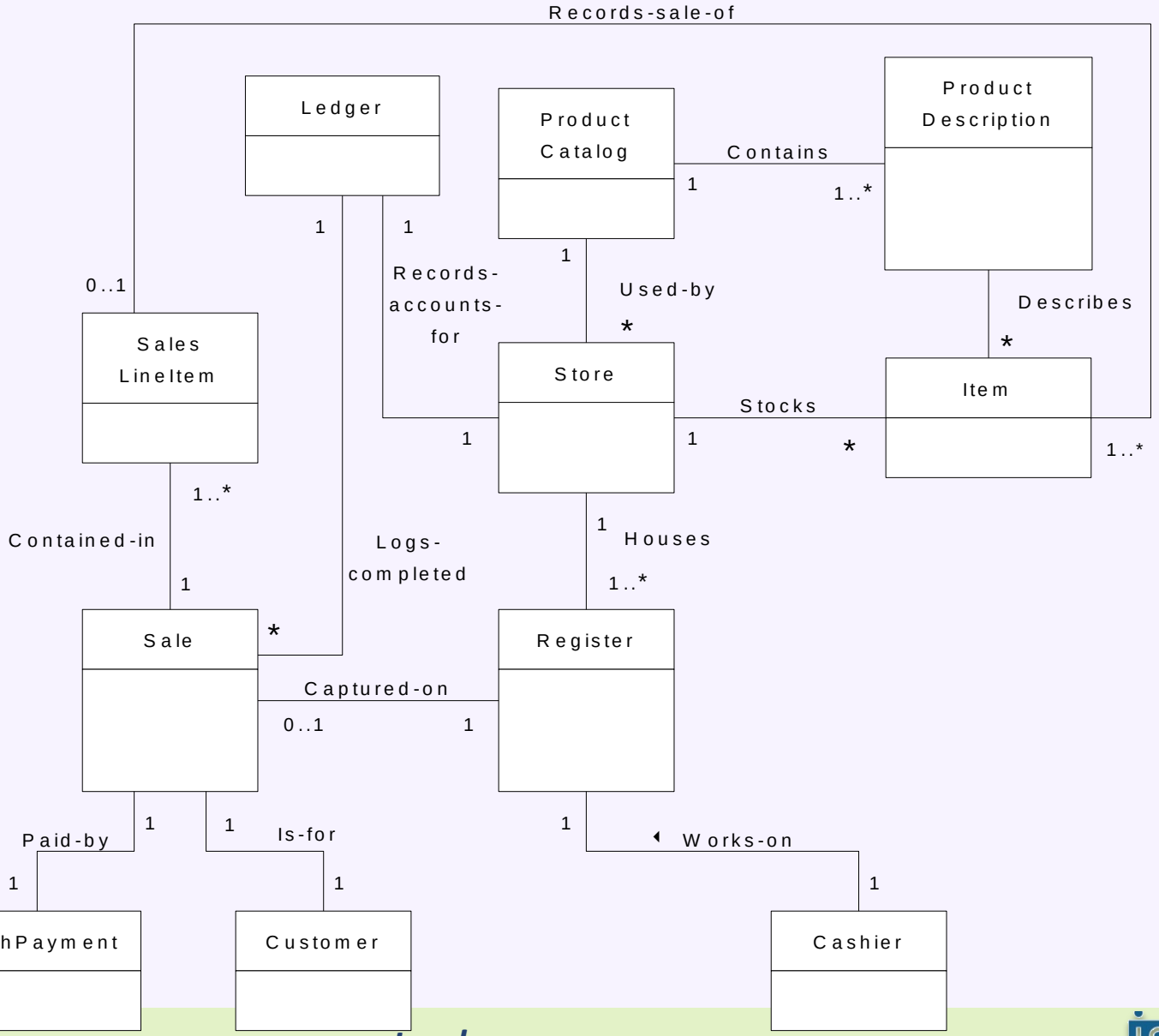
- "If we do not think of some conceptual class X as text or a number in the real world, it's probably a conceptual class, not an attribute"
- Avoid type annotations

Associations



- When do we care about a relationship between two objects? (in the real world)
- Include cardinality (aka multiplicity) where relevant

Conceptual Model (example, excerpt)



Outlook: Lowering the Representational Gap (Congruency)

- Classes in the object model and implementation will be inspired by domain model
  - similar names
  - possibly similar connections and responsibilities
- Facilitates understanding of design and implementation
- Eases tracking and performing of changes

Hints for Object-Oriented Analysis

- A domain model provides vocabulary
  - for communication among developers, testers, clients, domain experts, …
  - Agree on a single vocabulary, visualize it
- Focus on concepts, not software classes, not data
  - ideas, things, objects
  - Give it a name, define it and give examples (symbol, intension, extension)
  - Add glossary
  - Some might be implemented as classes, other might not
- There are many choices
- The model will never be perfectly **correct**
  - that’s okay
  - start with a partial model, model what's needed
  - extend with additional information later
  - communicate changes clearly
  - otherwise danger of "analysis paralysis"

Documenting a Domain Model

- Typical: UML class diagram
  - Simple classes without methods and essential attributes only
  - Associations, inheritances, … as needed
  - Do not include implementation-specific details, e.g., types, method signatures
  - Include notes as needed
- Complement with examples, glossary, etc as needed
- Formality depends on size of project
- Expect revisions

Remember: Three perspectives of class diagrams

- **Conceptual**: Draw a diagram that represents the concepts in the domain under study
  - **Conceptual classes** reflect concepts in the domain
  - Little or no regard for software that might implement it
- **Specification**: Describing the interfaces of the software, not the implementation
  - **Software classes** representing candidates for implem.
  - Often confused in OO since classes combine both interfaces and implementation
- **Implementation**: Diagram describes actual **implementation classes**

Understanding the intended perspective is crucial to drawing and reading class diagrams, even though the lines between them are not sharp

Summary

- **Object-oriented analysis** identifies relevant **concepts** in the **problem space**
- Documented in a **domain model**
  - conceptual classes, associations, attributes
  - extra glossary, examples, as needed
- Keep notation simple and implementation independent