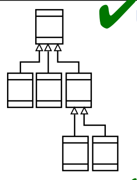
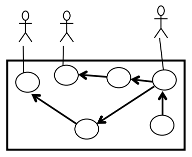
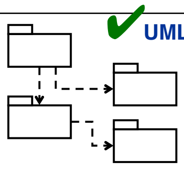
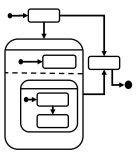
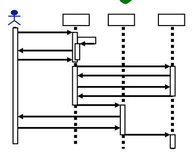
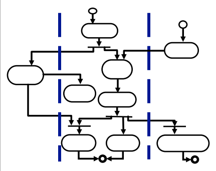
Lecture 8: "Use Case"-Driven Design

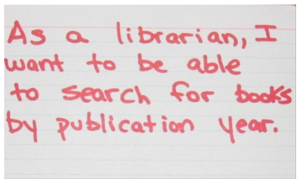
- → User Stories in Agile Development
- → Introducing UML into the Software Process
  - ➪ E.g. ICONIX
- → Domain Models
- → Use Cases

Refresher: UML Notations

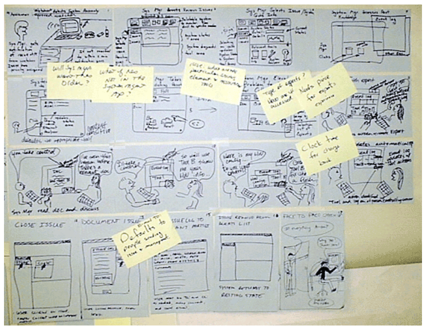
 ✔ **UML Class Diagrams** information structure; relationships between data items; modular structure for the system	 **Use Cases** user's view; Lists functions; visual overview of the main requirements
 ✔ **UML Package Diagrams** Overall architecture; Dependencies between components	 **(UML) Statecharts** responses to events; dynamic behavior; event ordering, reachability, deadlock, etc
 ✔ **UML Sequence Diagrams** individual scenario; interactions between users and system; Sequence of messages	 **Activity diagrams** business processes; concurrency and synchronization; dependencies between tasks;

What do users want?

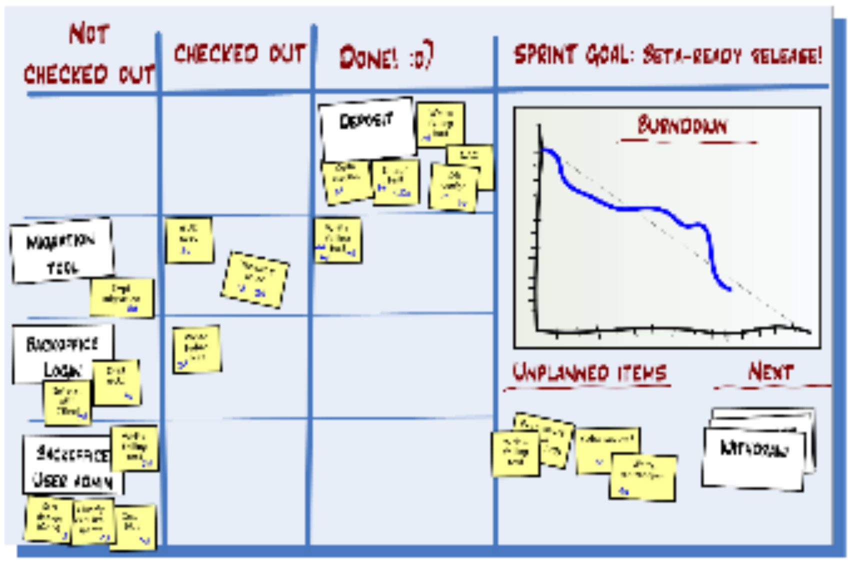
User Stories
- Used in XP, Scrum, etc.
- Identify the user (role) who wants it
- Typically written on notecards


As a librarian, I want to be able to search for books by publication year.

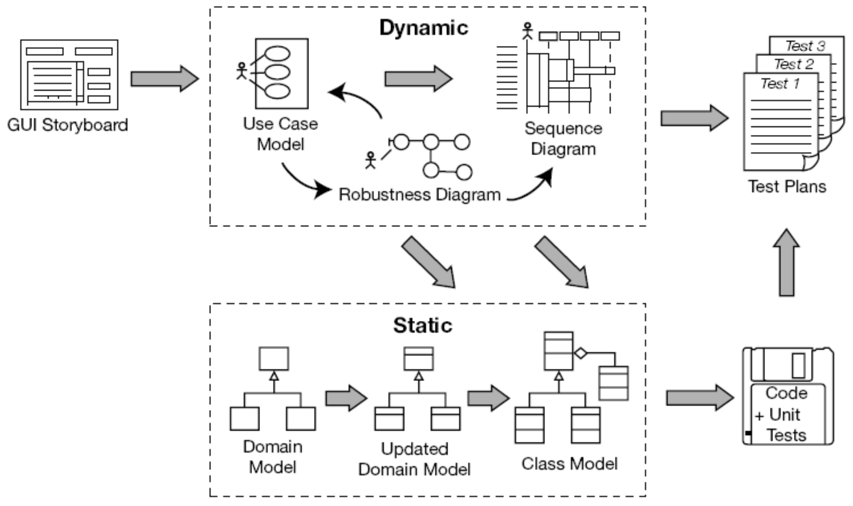
(User Interface) Storyboards
- Sketch of how a user will do a task
- Shows the interactions at each step
- Commonly used in UI Design



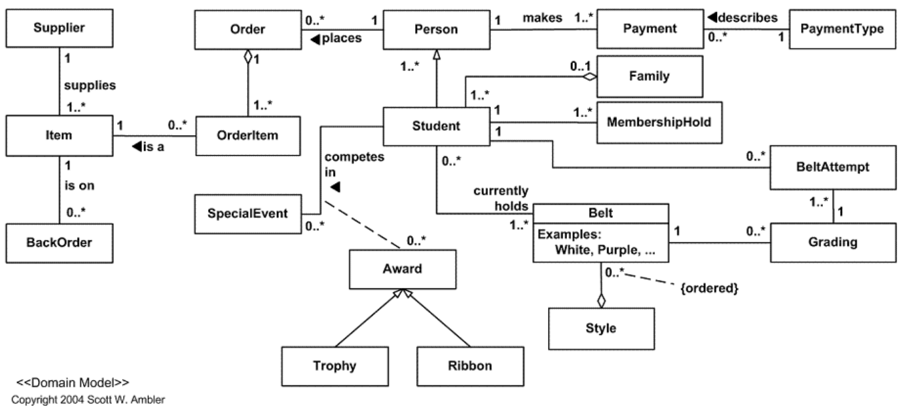
Use Cases
- Sets of user features
- UML diagram shows inter-relationships



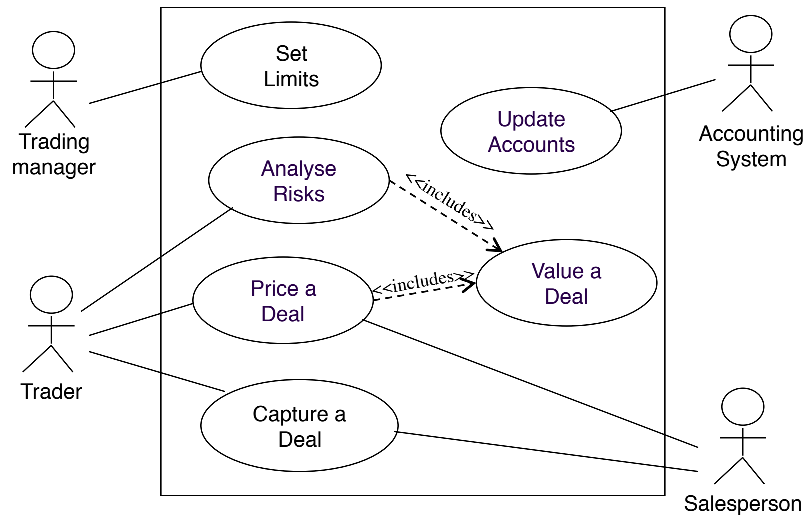
ICONIX process



Domain Model



Use Case Diagram



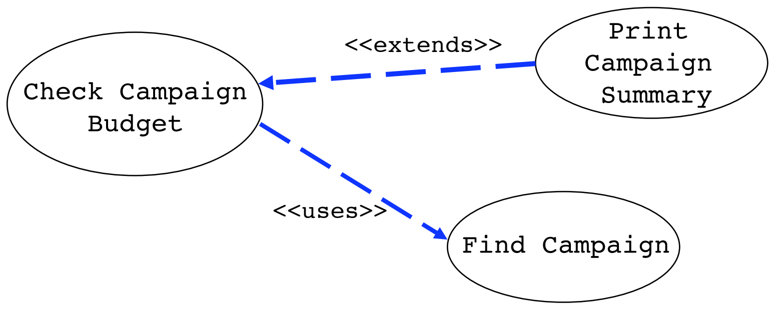
Relationships between Use Cases

<<extends>> when one use case adds behaviour to a base case

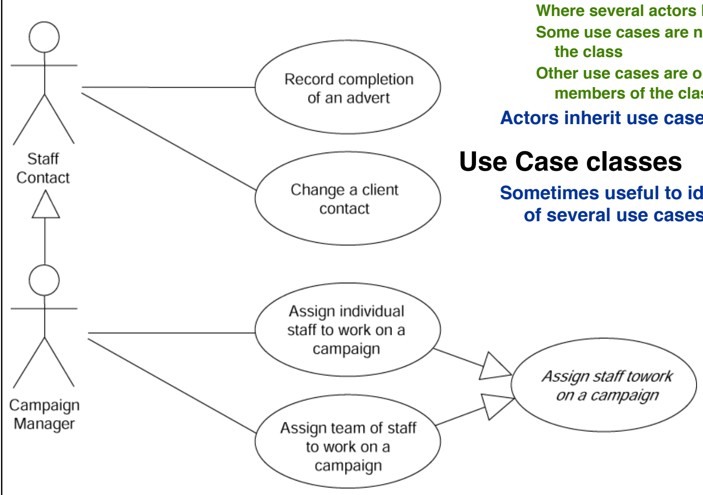
used to model a part of a use case that the user may see as optional system behavior; also models a separate sub-case which is executed conditionally.

<<uses>>: one use case invokes another (like a procedure call);

used to avoid describing the same flow of events several times puts the common behavior in a use case of its own.



Using Generalizations


Staff Contact
Campaign Manager
Record completion of an advert
Change a client contact
Assign individual staff to work on a campaign
Assign team of staff to work on a campaign
Assign staff towork on a campaign

Actor classes

- Identify classes of actor
  - Where several actors belong to a single class
  - Some use cases are needed by all members in the class
  - Other use cases are only needed by some members of the class
- Actors inherit use cases from the class

Use Case classes

- Sometimes useful to identify a generalization of several use cases

Describing Use Cases

For each use case:

- a "flow of events" document, written from an actor's point of view.
- describes what the system must provide to the actor when the use case is executed.

Typical contents

- How the use case starts and ends;
- Normal flow of events;
- Alternate flow of events;
- Exceptional flow of events;

Documentation style:

- Choice of how to elaborate the use case:
  - English language description
  - Activity Diagrams - good for business process
  - Collaboration Diagrams - good for high level design
  - Sequence Diagrams - good for detailed design

Detailed Use Case

Buy a Product

Main Success Scenario:
1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping information (address, next-day or 3-day delivery)
4. System presents full pricing information
5. Customer fills in credit card information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming email to customer

Extensions:
3a: Customer is Regular Customer
 .1 System displays current shipping, pricing and billing information
 .2 Customer may accept or override these defaults, returns to MSS at step 6
6a: System fails to authorize credit card
 .1 Customer may reenter credit card information or may cancel

Finding Use Cases

Browse through existing documents

- noun phrases may be domain classes
- verb phrases may be operations and associations
- possessive phrases may indicate attributes

For each actor, ask the following questions:

- Which functions does the actor require from the system?
- What does the actor need to do ?
- Does the actor need to read, create, destroy, modify, or store some kinds of information in the system ?
- Does the actor have to be notified about events in the system?
- Does the actor need to notify the system about something?
- What do those events require in terms of system functionality?
- Could the actor's daily work be simplified or made more efficient through new functions provided by the system?