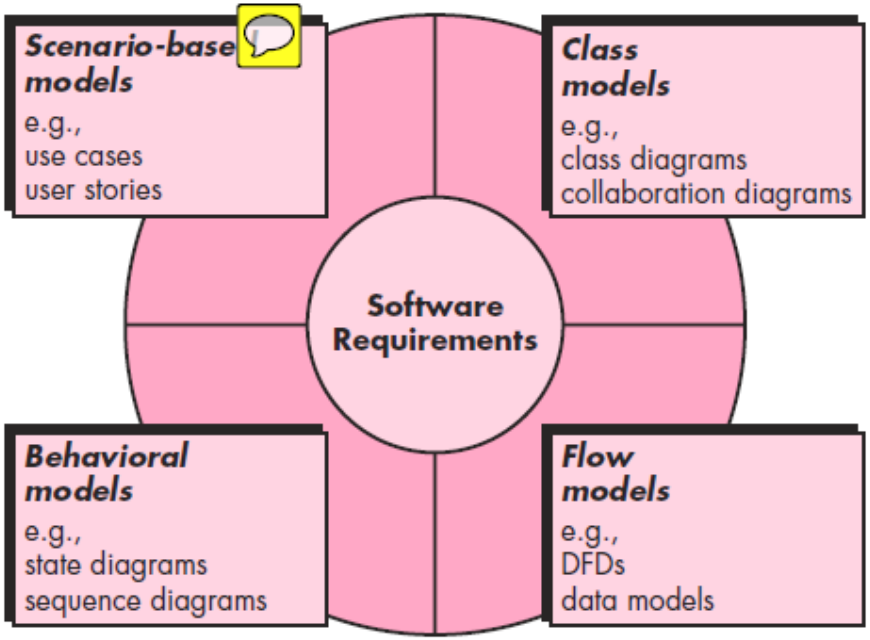
Software Engineering

Lecture 3, Week-3

Analysis Modeling for WebApps (I)

What is Analysis Modelling??

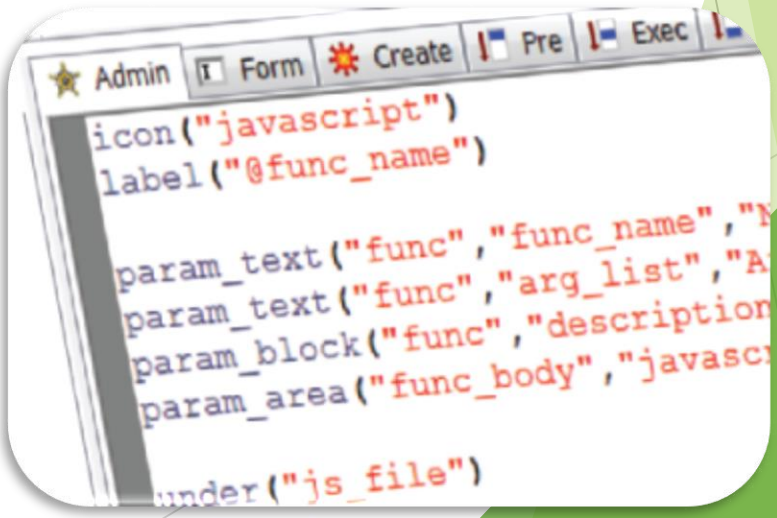
- Often Programmers/Web Developer go right to the development phase without really understanding:
  - What they are trying to build
  - How they want to build it
- However some modelling and software engineering can make the development process much smoother.
- It comes down to asking yourself as a developer:
  - Do you understand the requirements of the problem?
  - Represent the requirements in an understandable form that describe your system (graphical, template based)



Necessities

Modeling and disciplined software engineering can:

- Make the software development process much smoother
- Ensure that the Web system is more maintainable in the future
  - Easier to follow code
  - Remove complexity

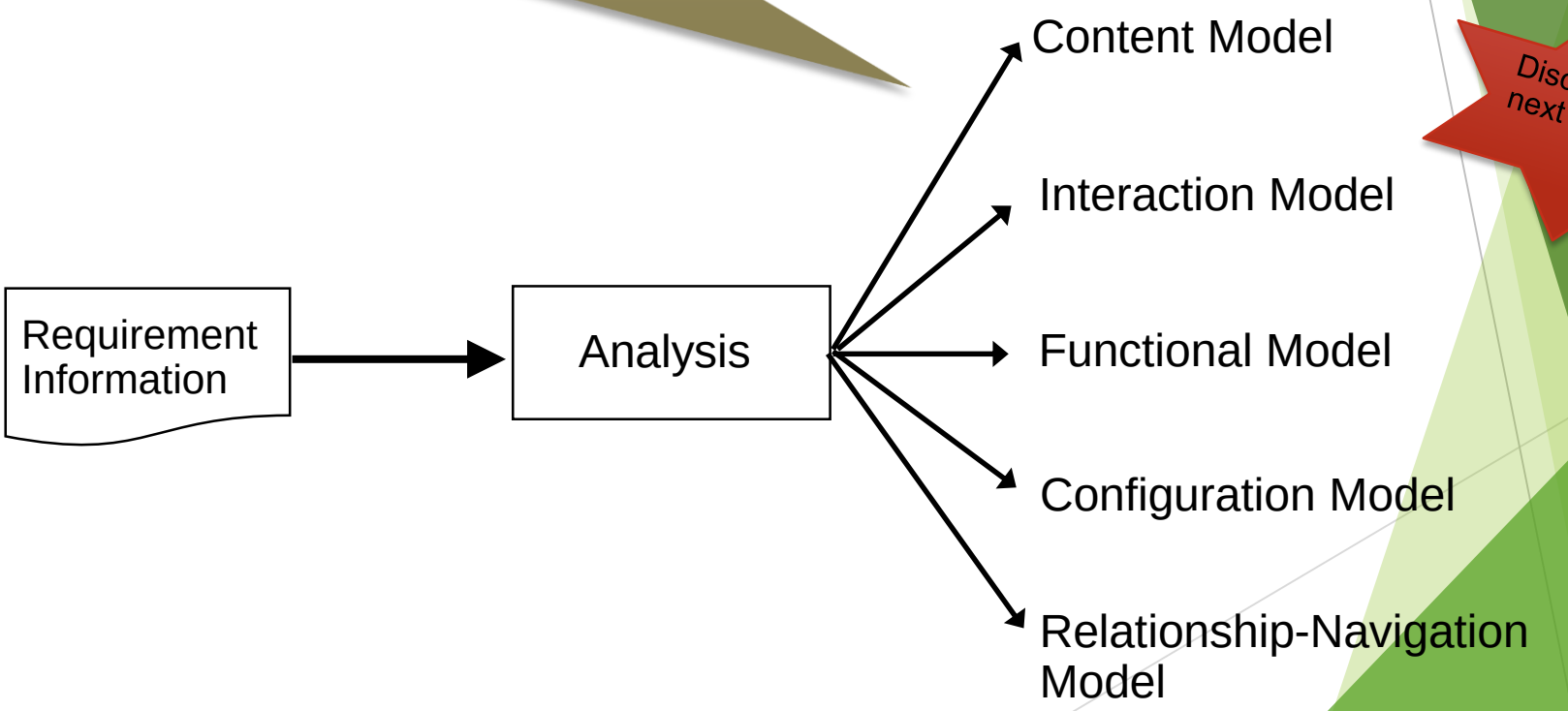


When Do We Perform Analysis?

- In some WebE situations, analysis and design merge together.
- However, a specific targeted analysis activity will occur when:
  - The WebApp to be built is large and/or complex
  - The number of stakeholders is large
  - The number of Web engineers and other contributors is large
  - The goals and objectives (determined during formulation) for the WebApp will affect the business' future financial situation
  - The success of the WebApp will have a strong bearing on the success of the business

Input and Output of Analysis

- From our requirements analysis we will have some models that are outputs from this stage.



Analysis Models

- Content Analysis.
  - The full spectrum of content to be provided by the WebApp is identified, including text, graphics and images, video, and audio data. ***Data modeling*** can be used to identify and describe each of the data objects, Data tree.
- Interaction Analysis.
  - The way in which the user interacts with the WebApp is described in detail. Use-cases can be developed to provide detailed descriptions of this interaction, sequence diagram, state diagram, activity diagram, swimlane diagram.
- Functional Analysis.
  - This defines the operations that will be applied to WebApp content and other processing functions. All operations and functions are described in detail (analysis level –activity diagram of process description).
- Configuration Analysis.
  - The environment and infrastructure in which the WebApp resides are described in detail. Relationship-Navigation Analysis. Appropriate links between content objects and the functions that provide user-required capabilities.

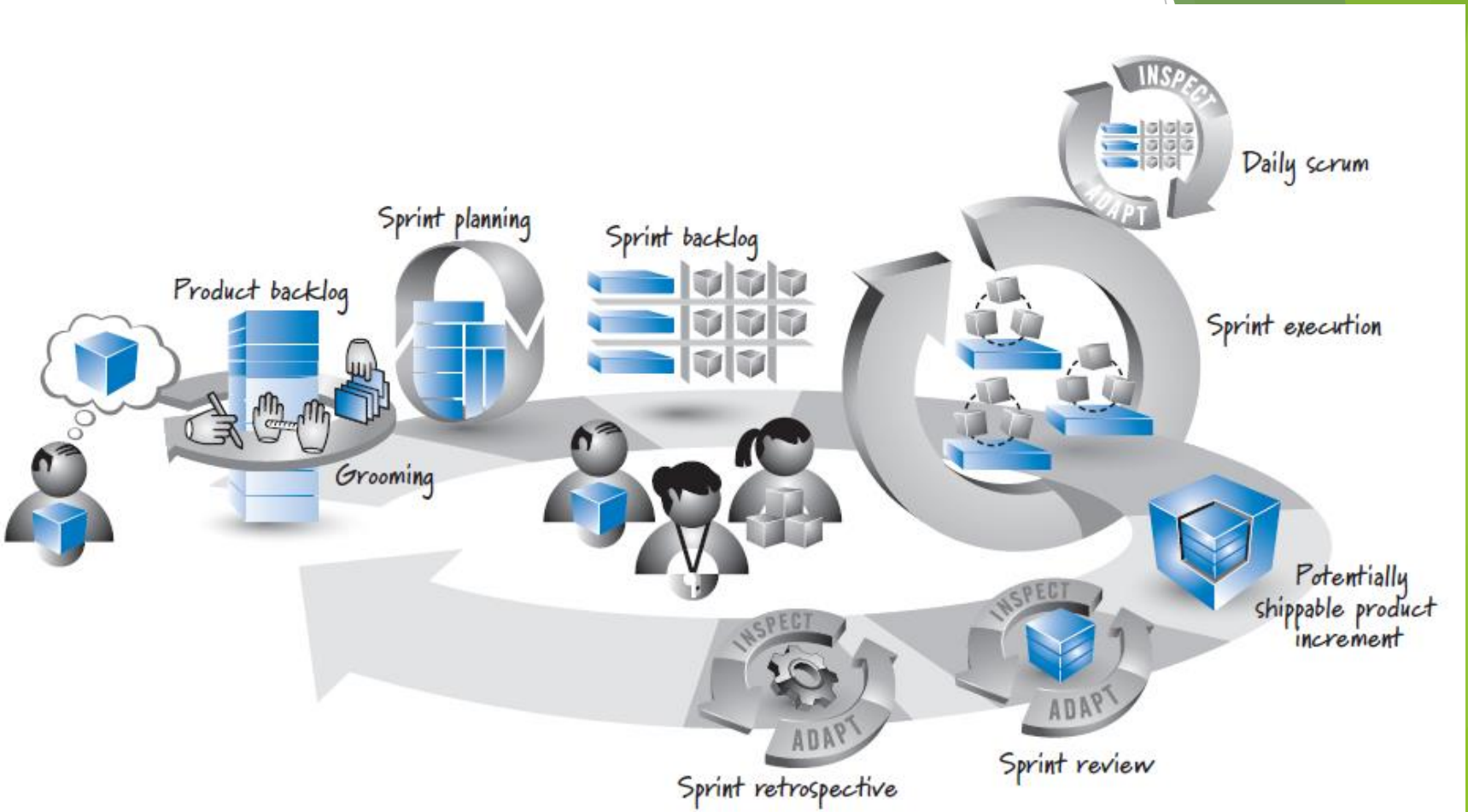
When Do We Perform Analysis?

- **In some WebE situations, analysis and design merge together.**
- **However, a specific targeted analysis activity will occur when:**
  - **The WebApp to be built is large and/or complex**
  - **The number of stakeholders is large**
  - **The number of Web engineers and other contributors is large**
  - **The goals and objectives (determined during formulation) for the WebApp will affect the business' future financial situation**
  - **The success of the WebApp will have a strong bearing on the success of the business**

In Scrum, Envisioning is part of requirement analysis

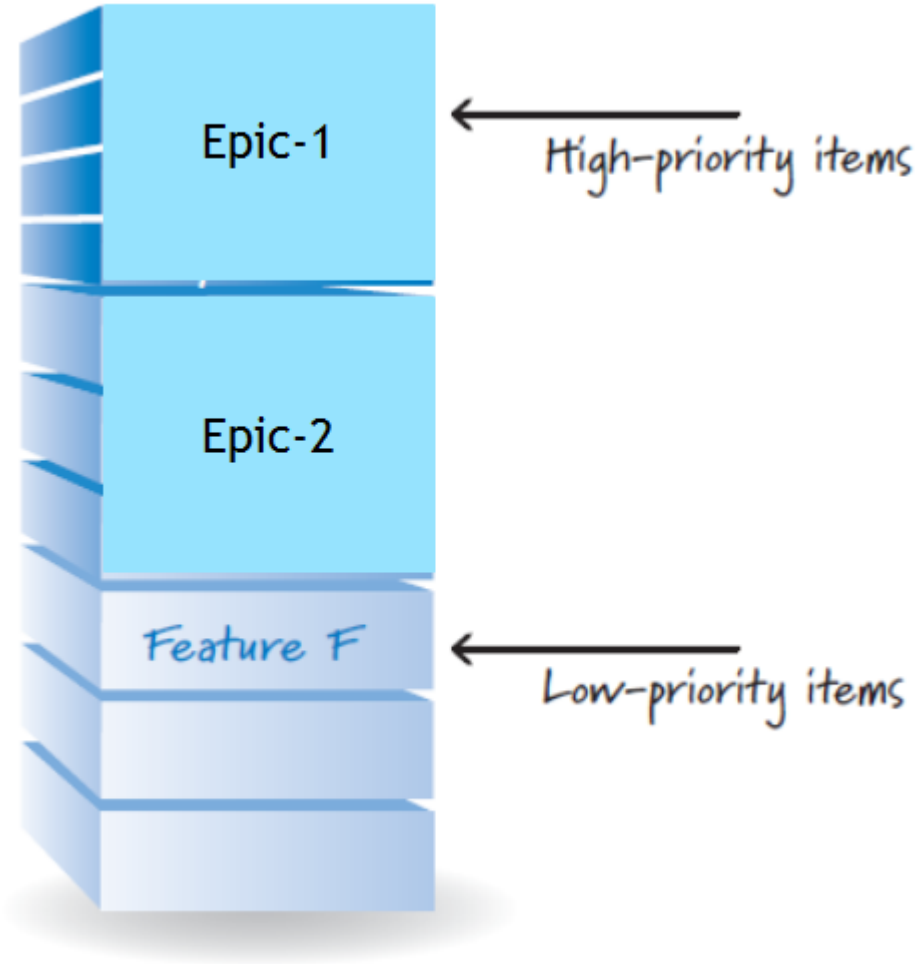
- Analysis steps starts from envisioning steps
- Envisioning sessions is the primary step in scrum which has mainly two parts.
  - **The first part:** is known as ideation.
  - **The second part:** is known as the high level product backlog creation.
- Ideation session (First part): This finds the product idea: Identify problems (pain points), market review, competitor reviews, internal analysis of current system, identify needs, create projects
  - This ideation of envisioning outputs:
    - the vision and project scope,
    - product perspective (high level architecture of product).

Deriving detail requirements from Initial product backlog



Reference of image: Book-- **Essential Scrum—Kenneth S. Rubin**

Initial Product backlog with Epics and Features



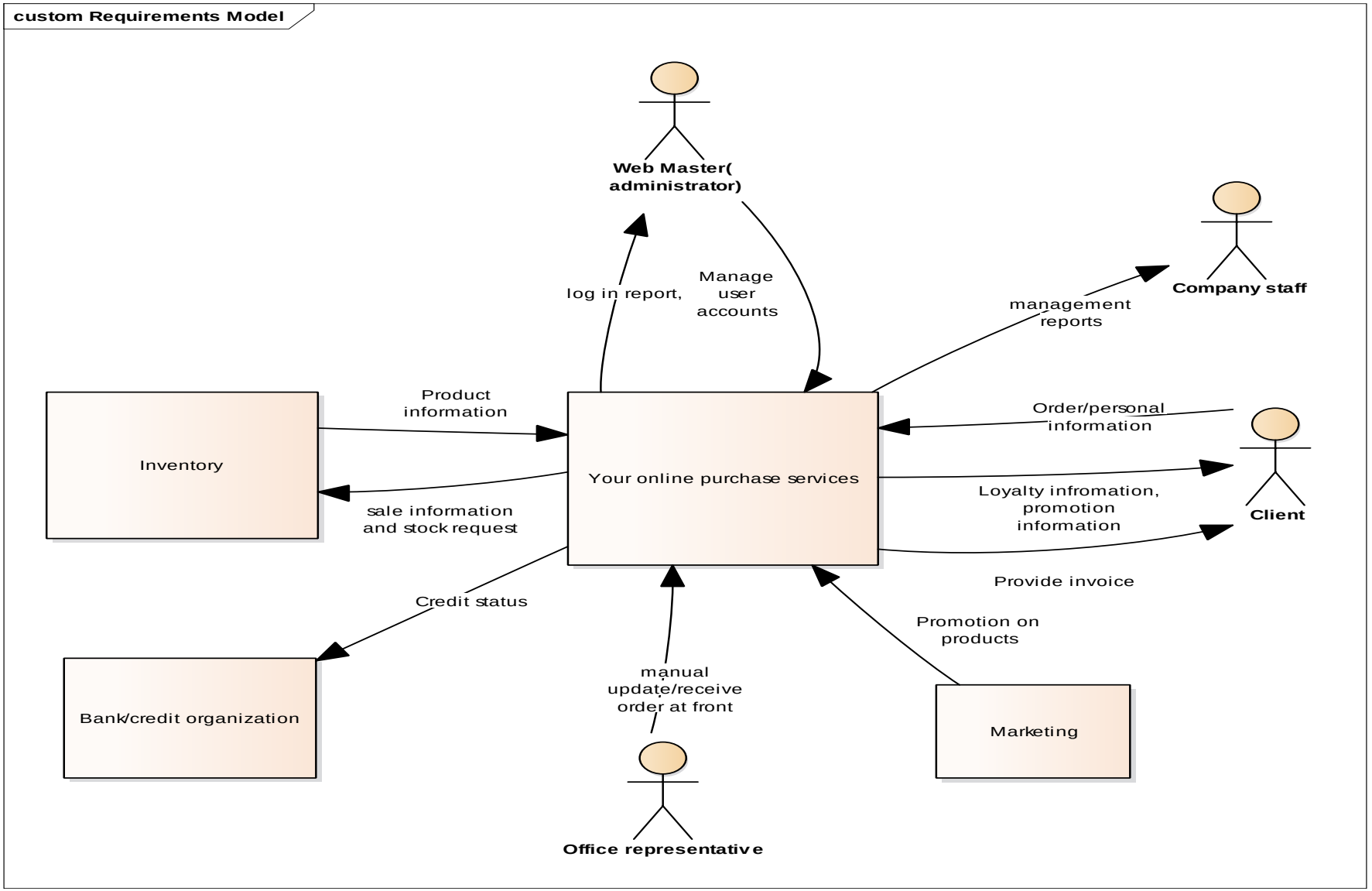
Reference of image: Book-- **Essential Scrum—Kenneth S. Rubin**

Deriving detail requirements from Initial product backlog: Backlog Grooming

Image---The agile age-Managing projects effectively using agile scrum---Brian vanderjack

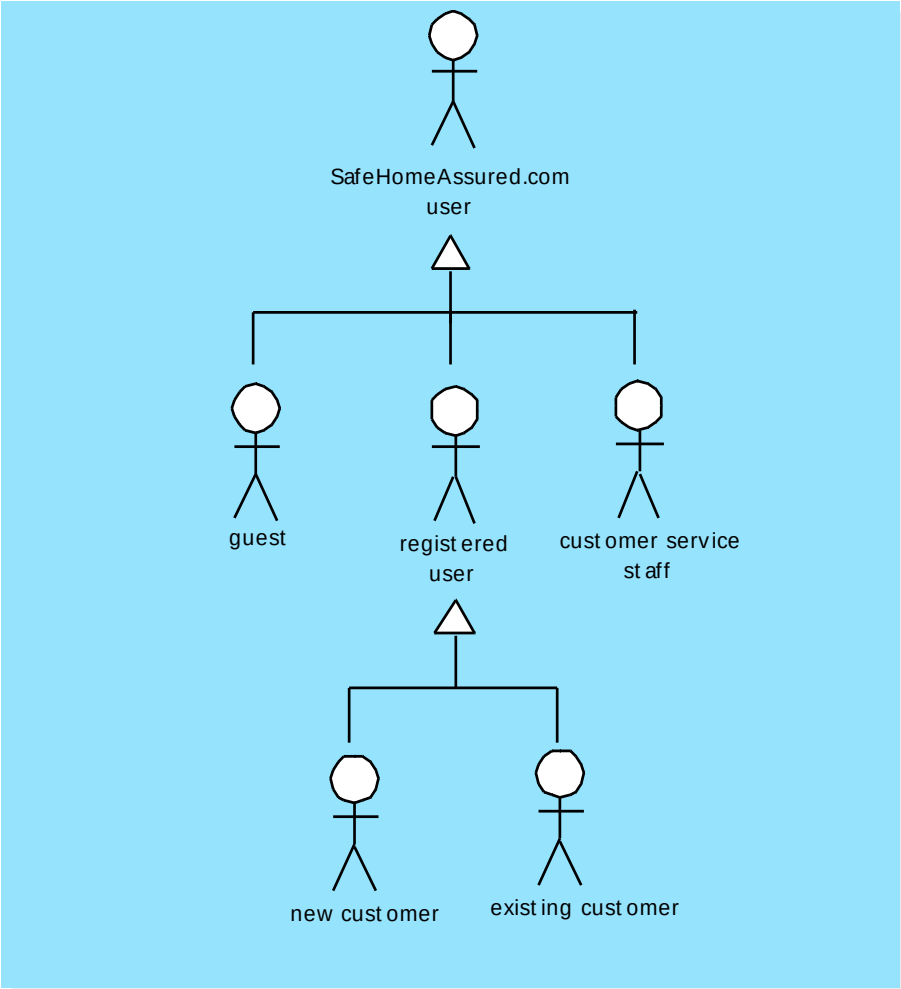
- Happens before sprint planning meeting
- Participants: Owner, development team, stakeholders, Scrum master, Scrum team members
- Use prototype, CASE tool, draw use case diagram, use case description, activity diagram
  - Identify detail requirements, prepare the product backlog

In Backlog grooming stage, use modelling tools



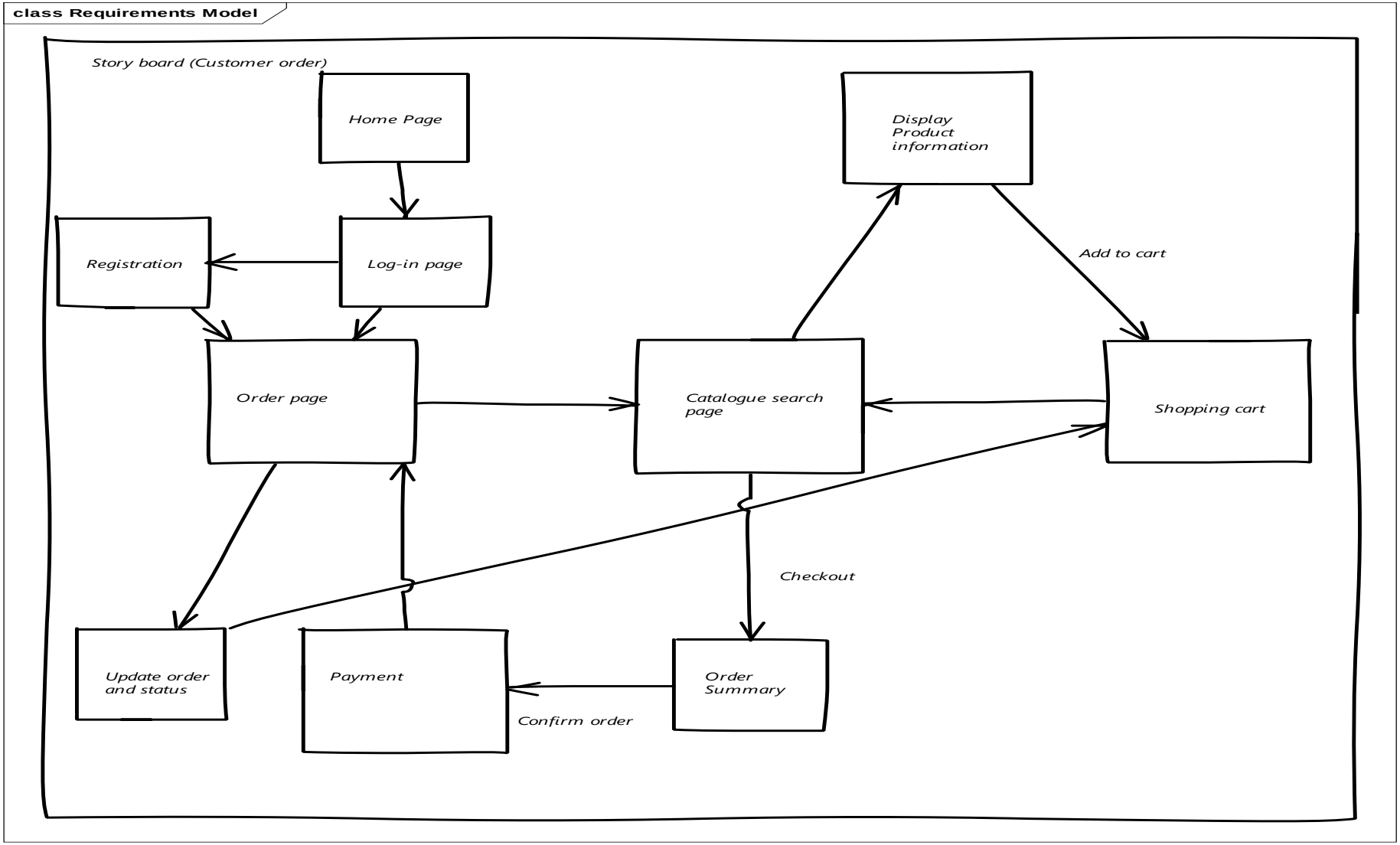
Requirement Analysis - User Hierarchy

Do users have different ranks? Which content will be visible to which user group?

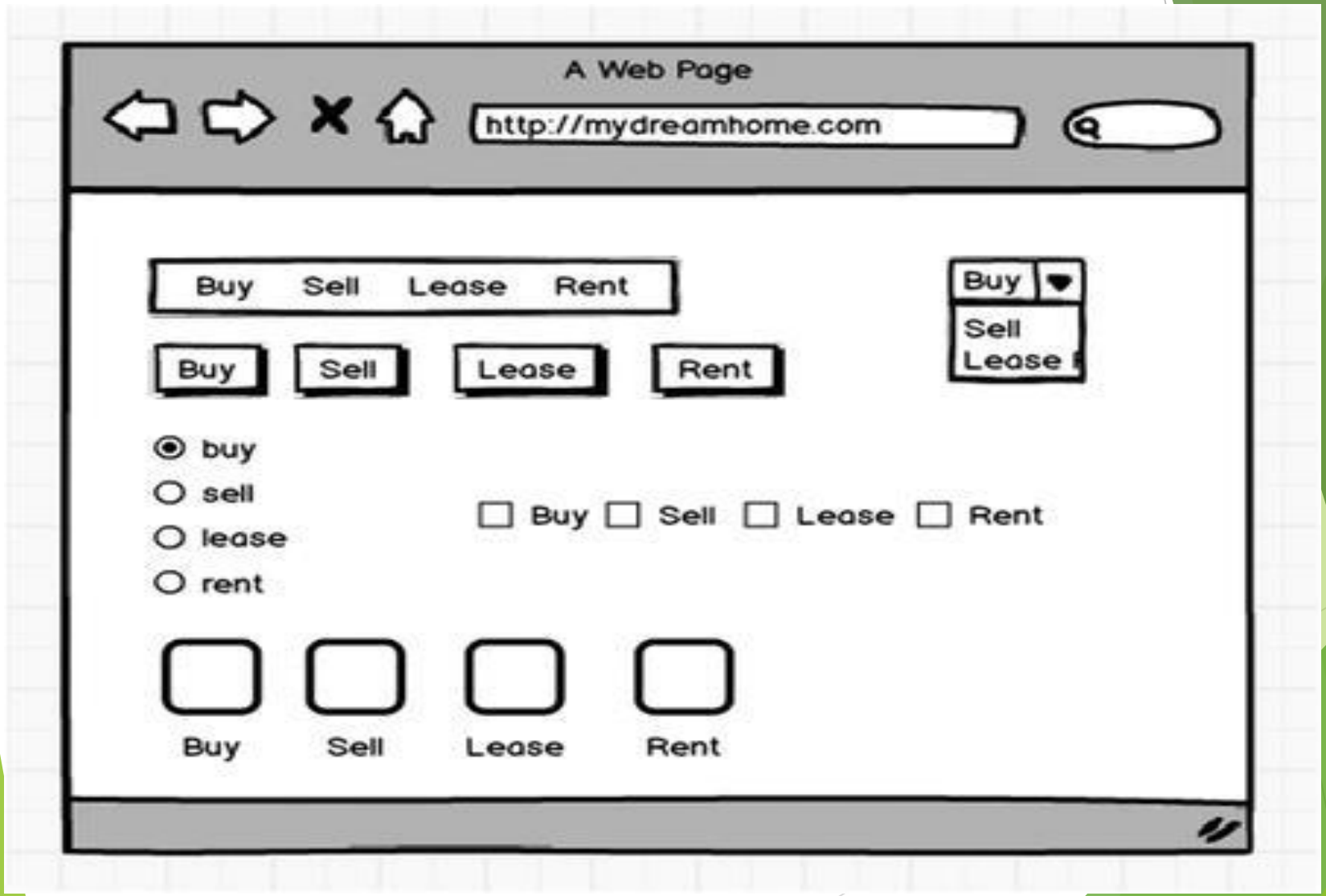


An example of Storyboard

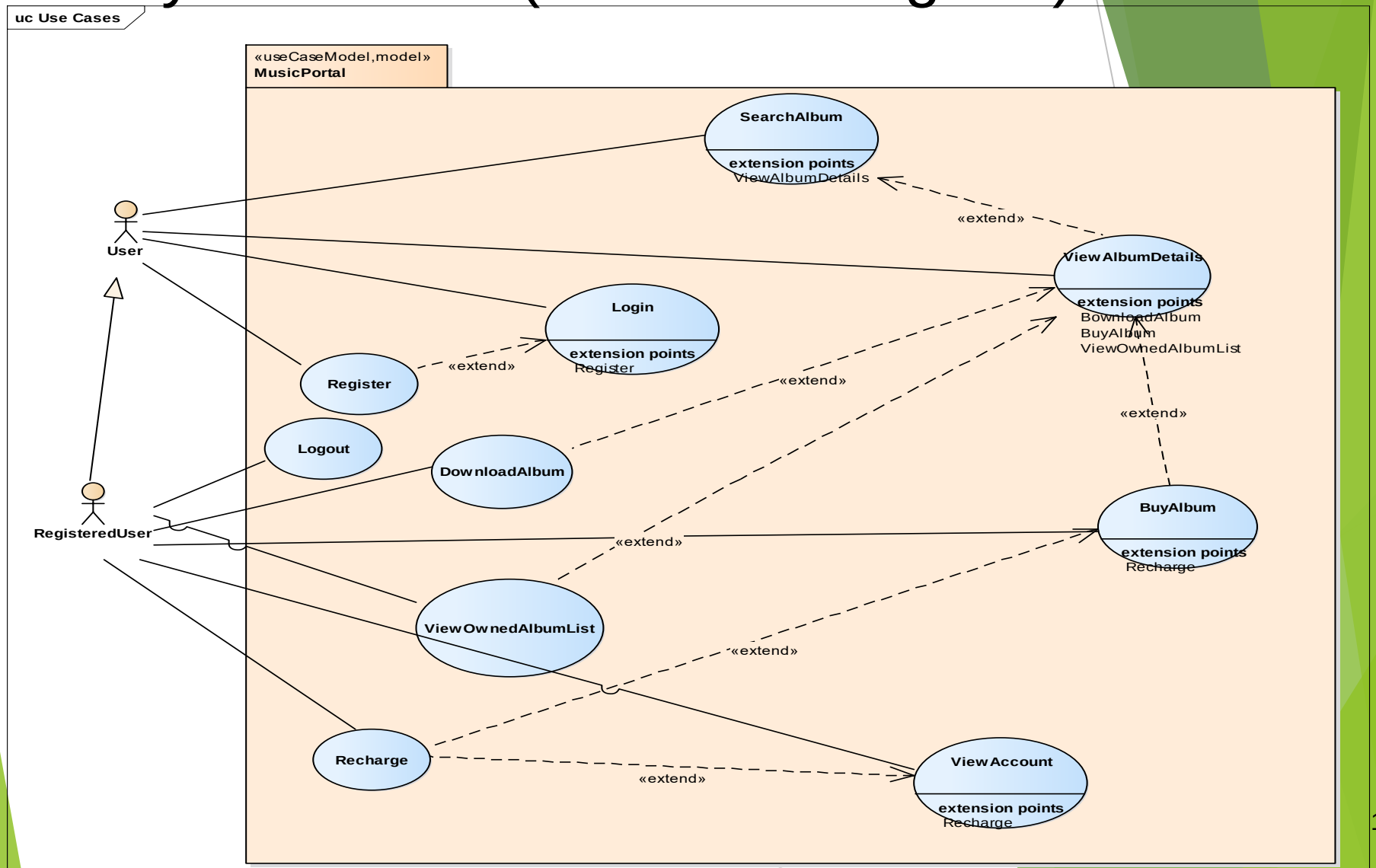
- prototype an entire scenario from beginning to end of an epic include a UI screen for a event that user will be experienced or initiated by the user simplify the interaction problem, think different ways



An example of Mock-ups



Analysis Models (use case diagram)



Identifying Use Cases

- Techniques
  - **Event decomposition technique**
    - decompose the epics into activities/use cases by identifying business events
  - **Identify user goals**
    - A use-case achieves a discrete goal for the user!

Requirement Analysis – Use Case Diagram-

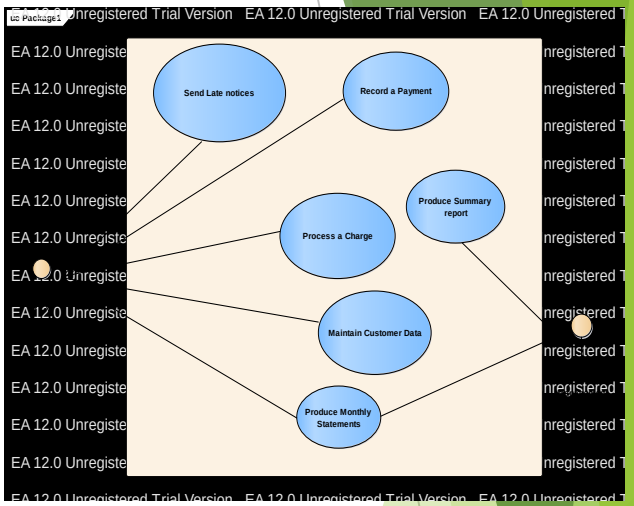
- Use cases define interactions between external actors and the system to attain particular goals.
- A **use case** is a depiction of a system's behavior or functionality under various conditions as the system responds to requests from users.
  - **Use case** represents a single system function
  - A use-case captures some user visible function
  - A use-case achieves a discrete goal for the user!

Actors: Actors are the type of users that interact with the system.

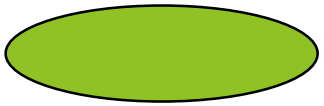
Goals: What are the activities and alternatives involved in attaining a goal of a specific task.

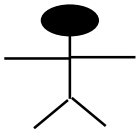
Definitions and Symbols for use case

- **System boundary** includes all the relevant use cases.
  - A boundary is the dividing line between the system and its environment.
  - Use cases are within the boundary.
  - Actors are outside of the boundary.
  - Represented as a box
- **Connection** is an association between an actor and a use case.
  - Depicts a usage relationship
  - Connection does not indicate data flow
  - Actors are connected to use cases with lines.
  - One use case is connected to other use case using an arrow



Definitions and Symbols

Use Case 

Actor 

Boundary 

Connection 

Include relationship 

Extend relationship 

Use case: Include

Includes

You have a piece of behavior that is similar across many use cases!

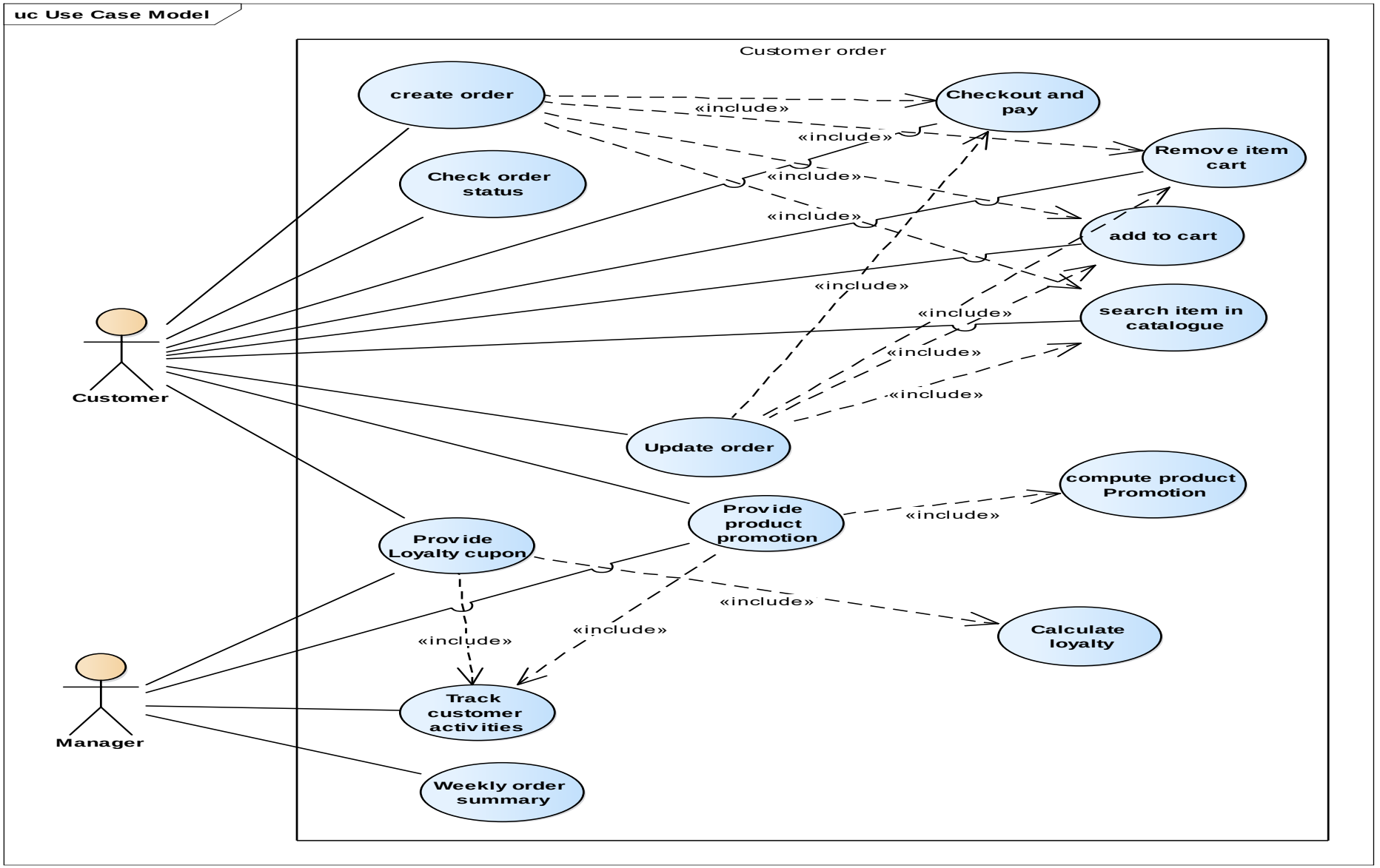
- Break this out as a separate use-case and let the other ones “include” it!
- Examples include!
  - Valuation!
  - Validate user interaction!
  - Sanity check on sensor inputs!
  - Check for proper authorization!

Use case: Extends

- *Extend relationship:*
- A use-case is similar to another one but does a little bit more!
  - Put the normal behavior in one use-case and
    - the exceptional behavior in another!

- Capture the normal behavior!
- Try to figure out what can go wrong in each step!
- Capture the exceptional cases in separate use cases!
- Makes it much easier to understand

In Backlog grooming stage use modelling tools



Requirement Analysis - Refining the Use Case Model

- Use-cases need to be organized effectively, so we can review them on an on-going basis.
- *They need to be:*
  - *Comprehensible*—all stakeholders have to understand the purpose of them
  - *Cohesive*—the use cases address the functions that are closely related to one another
  - *Loosely coupled*—functions or classes that collaborate with one another, are kept to a minimum.

Use Case Description (part-1)

Use case name	**Create an online order in Web application**
Use case Scenario	online order scenario
Use case Triggering event	User/registered customer go to shopping web site and logs on/ new customer clicks new account button/link/menu option
Scenario description	Registered customer logs on. He/she chooses the buy menu for ordering, then selects the product from catalogue form, mention quantity of products and their details. Web application takes the input from user and create entry in online cart and user add more items to the cart. User decides to finish and selects a payment option. Web application takes payment information, verifies those, finalize the order and provides a tax invoice to the user.
Primary actor	User/Customer
Related use cases	Includes: Create a new online account, search item in catalogue
Precondition	User need to be online, item catalogue need to be available
Post condition	• Web app will create an order, an invoice will be produced • Inventory items must have the item's quantity updated for successful order. • Order must be associated with a customer. • If unsuccessful, any uncompleted transaction need to be rollback, customer need to be advised

Use Case Description (part-2)

Flow of activities	Actor	Web application
	1. User go to the shopping web site of Webapp and clicks the log on link in the page.	
	2 If he/she is new, he/she clicks the register link, provide appropriate information to create customer account	2.1Create an account record for a new user
	2.q.Customer logs on to the purchase page	2q1. Validate user name and password 2q2. Switch to the catalogue page and present to user
	3 Customer find items using catalogue and checks product detail	3.1 Switch to product search page
	4 If customer finds the product item as required, he/she enter the quantity of item and selects the add button to create an order and add the item to the shopping cart of the order form	4.1 Adds selected item to the cart
	5 If he/she wants more item to purchase, he/she continue shopping by the following steps from step 3 to step 4	5.1 Presents cart summary and total prices

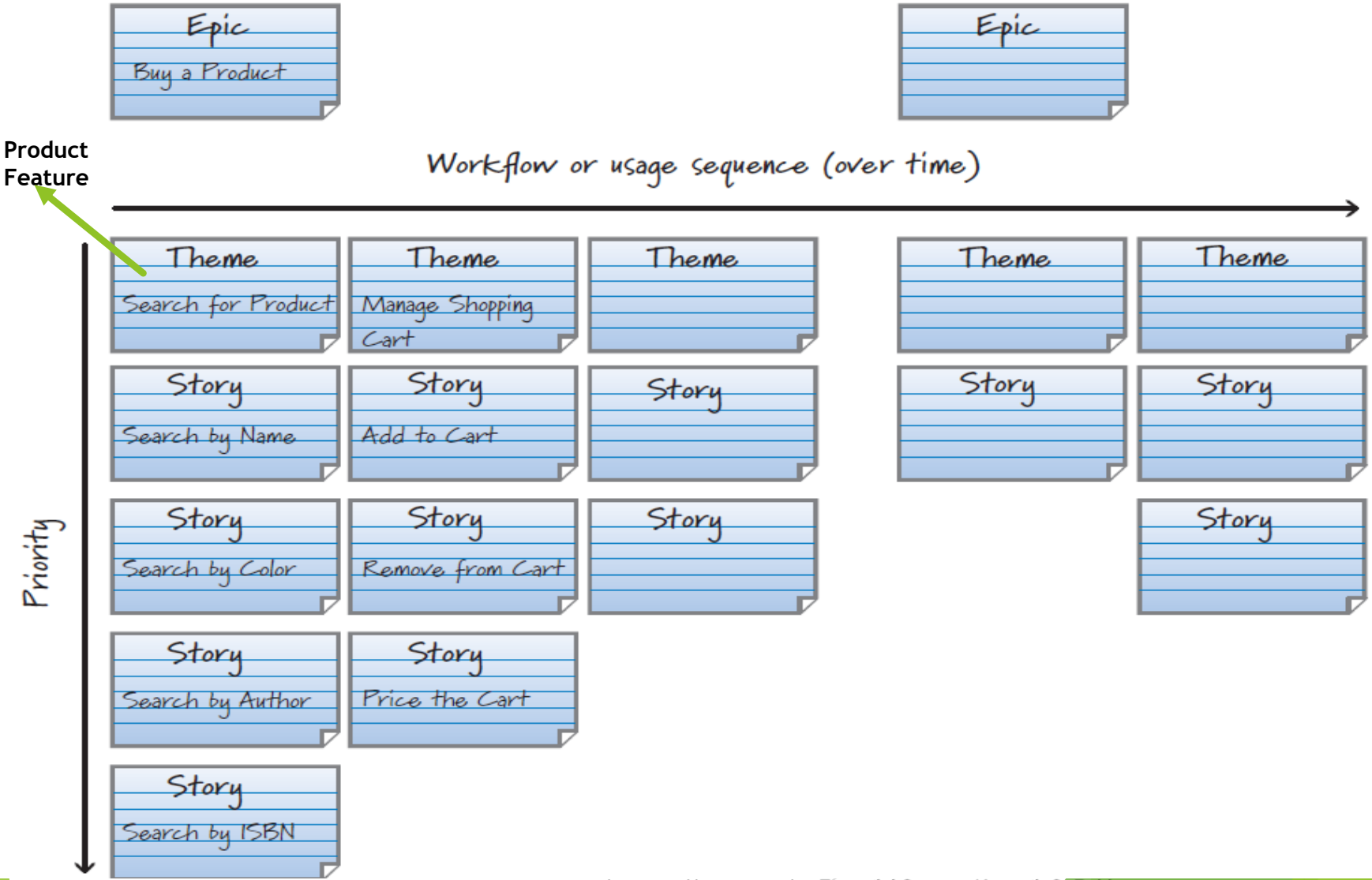
Use Case Description (part-3)

Flow of activities (cont.)	Actor	Web application
	6. When he/she decides to stop add item, clicks end of order button,	
	7. He/she would like to change (remove/update quantity) to shopping cart, then clicks update cart	
	8 When he/she wants to finish shopping, he/she checks out by clicking checkout link/button	8.1 Presents total cost including the delivery cost and display payment form
	9. He/she chooses payment option, provides payment information and shipping address	
	10 He/she confirm order by clicking confirm button/link.	10.1 Takes the input form payment form, checks customer's credit status, Processes payment, provides order confirmation information and an tax invoice

Use Case Description (part-4)

Exception conditions	**2q.1 If user forget the password, then** **a. Web app generates a temporary password and sends that by e-mail, advising to change password at next login, or contact by e-mail to office staff.**
	10.1. If credit status is not OK or wrong information is invalid, web app does not process, then a. facility to re-enters payment information must be available and user need to confirm order again, or b. order can be cancelled by user, or c. The order can be marked as a future order by saving in customer purchase profile
	Student can Add more exception condition

Deriving detail requirements (Story maps)



Reference of image: Book-- **Essential Scrum—Kenneth S. Rubin**

The Content Model

- A representation that provides a clear indication of the content that is required to support a usage scenario.
  - Think about your different scenario's – what content do we need

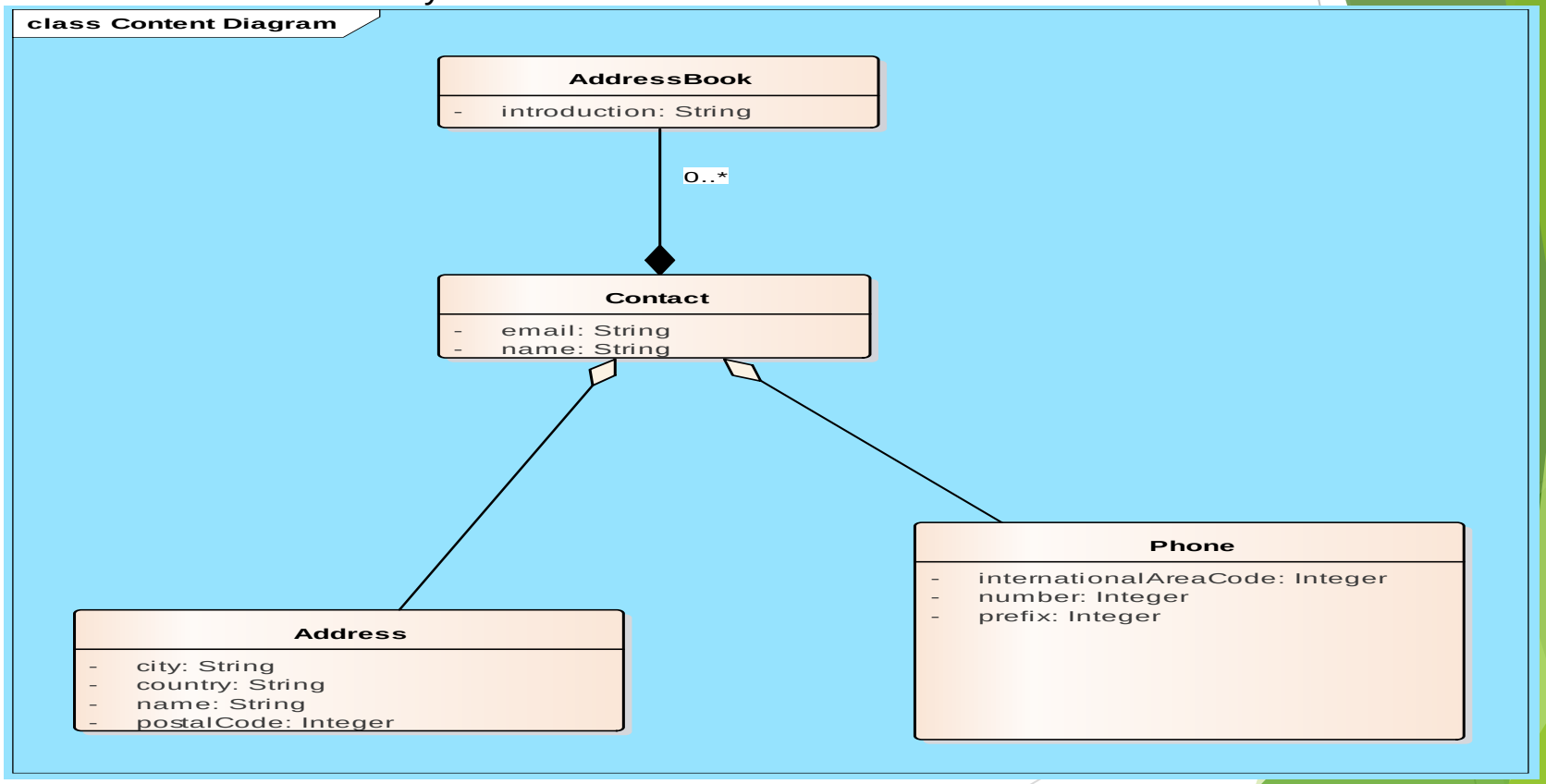
- **Structural elements of the content model**
  - **Content objects.**
  - **All analysis classes – user visible entities that are created or manipulated as a user interacts with the WebApp.**

A content object is any item of solid information that is to be <u>presented to an end user</u>, such as:

Textual description of a product
An article
An action photograph
A short video of a speech
A collection of PowerPoint slides
etc.

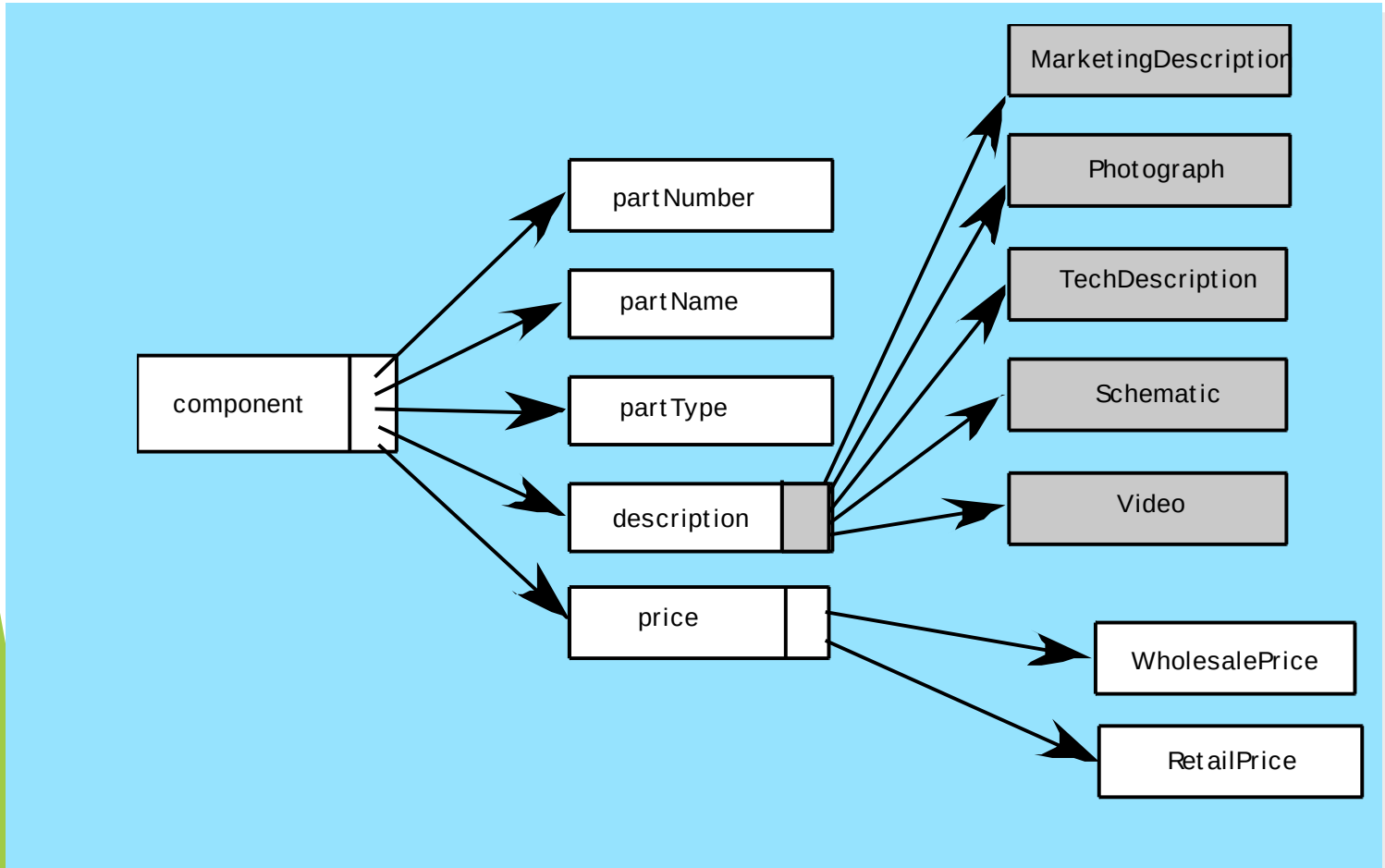
The Content Model

- In some cases, you can establish a better content model with deeper analysis using such things as:
- Entity relationship diagrams - graphical representation of ***entities*** and their relationships to each other.
- UML Class model for analysis classes



Data Tree

Data trees - a hierarchical *tree* structure of all the data used in each component of your WebApp.

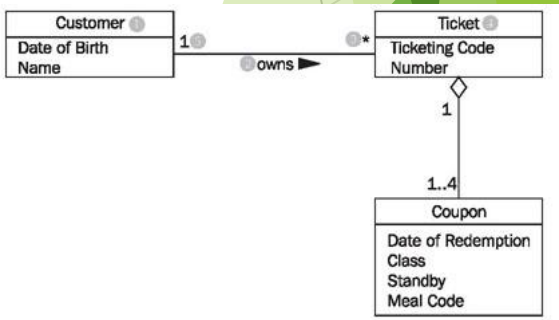


Analysis Classes

- Analysis classes manifest themselves in one of the following ways.
  - External entities: other system, people that produce or consume information to be used by the WebAPP
  - Things: that are part of information domain for the problem: report, display
  - Events: an order that occur within the context of user interaction
  - Role of users: customer support, salesperson
  - Organizational units: team group
  - Places
  - Structure: that defines a class of objects: sensors, monitoring devices
- Components of analysis classes:
  - Attributes
  - Operations
  - Collaborations
    - Identify relationship between classes

Identifying Classes

- General classifications for a potential class
  - External entity (e.g., another system, a device, a person)
  - Thing (e.g., report, screen display)
  - Occurrence or event (e.g., movement, completion)
  - Role (e.g., manager, engineer, salesperson)
  - Organizational unit (e.g., division, group, team)
  - Place (e.g., manufacturing floor, loading dock)
  - Structure (e.g., sensor, vehicle, computer)

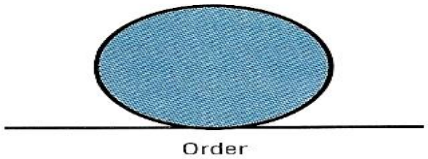


Class responsibilities and collaboration (CRC) modelling

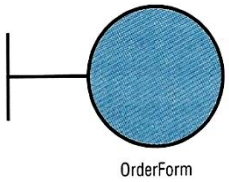
- Analysis classes have “responsibilities”
  - *Responsibilities* are the attributes and operations encapsulated by the class
- Analysis classes collaborate with one another
  - *Collaborators* are those classes that are required to provide a class with the information needed to complete a responsibility.
  - In general, a collaboration implies either a request for information or a request for some action.
  - [Ref: 2]

CRC modelling ([Ref: 2])

- Entity classes, also called model or business classes, are extracted directly from the statement of the problem (e.g., FloorPlan and Sensor).



- Boundary classes are used to create the interface (e.g., interactive screen or printed reports) that the user sees and interacts with as the software is used.

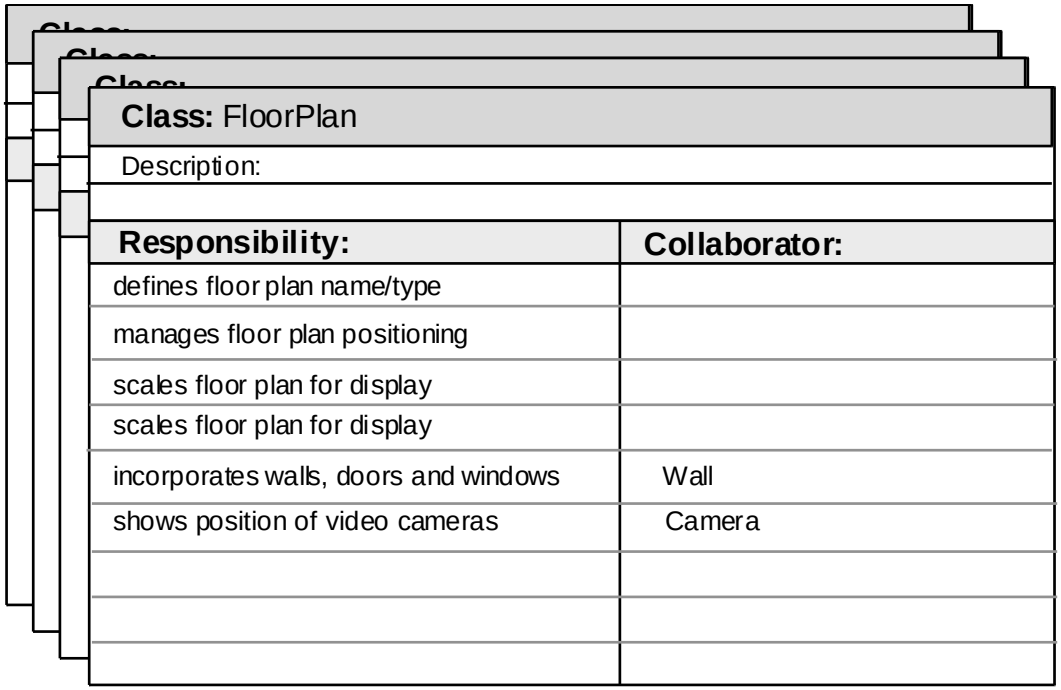


- Controller classes manage a "unit of work" from start to finish. That is, controller classes can be designed to manage
  - the creation or update of entity objects; the instantiation of boundary objects as they obtain information from entity objects; complex communication between sets of objects; validation of data communicated between objects or between the user and the application.

CRC (Ref:2)
Responsibilities and Collaborations

- Responsibilities:
  - System intelligence should be distributed across classes to best address the needs of the problem
  - Each responsibility should be stated as generally as possible
  - Information and the behavior related to it should reside within the same class
  - Information about one thing should be localized with a single class, not distributed across multiple classes.
  - Responsibilities should be shared among related classes, when appropriate.
- Collaborations
  - Classes fulfill their responsibilities in one of two ways:
    - A class can use its own operations to manipulate its own attributes, thereby fulfilling a particular responsibility, or
    - a class can collaborate with other classes.
  - Collaborations identify relationships between classes
  - Collaborations are identified by determining whether a class can fulfill each responsibility itself

CRC card (Ref:2)



Class: FloorPlan

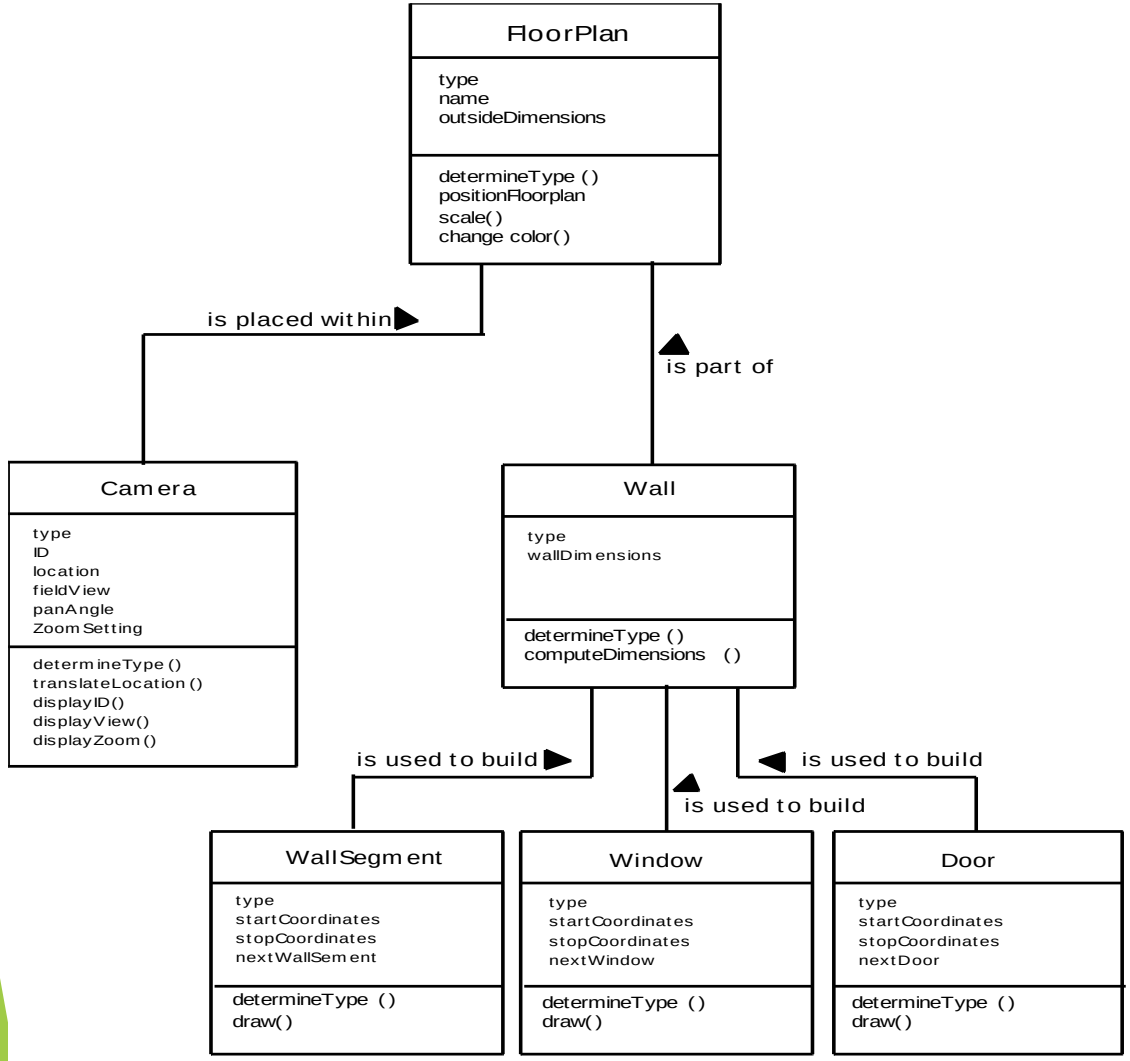
Description:

Responsibility:	**Collaborator:**
defines floor plan name/type	
manages floor plan positioning	
scales floor plan for display	
scales floor plan for display	
incorporates walls, doors and windows	Wall
shows position of video cameras	Camera

Identifying analysis classes using CRC

1. Select a use case, use problem domain class to identify a responsible class for this use case
2. Select a controller class from whom this object receive first message and handle system's internal process, you may need more than for other use cases
3. Create CRC cards by performing textual analyses from this use-cases description and role playing of this use case
4. Role play of this use case by thinking about what happens and which is helpful.
   I. Role playing involves thinking of your self as one of the instances (the role) and walking through what you would do.
   II. You can even have several people and let each of them play a class as you walk through a use case. It becomes a performance of the use case.
5. In the CRC card left side, write the actions/responsibility as the methods for this class, piece of information as attributes.
6. As you progress in role play, you may identify that this class needs information from other classes. Write their **class-name (attribute name) format** on right side of CRC
7. Update the CRC cards of the other class or create new CRC cards for other class as you identify
8. Mention in those CRCs about other classes' responsibilities and attributes as you role play of this use-case
9. **Repeat 1-9 for all other use cases**
10. Review the structural model for missing and/or unnecessary classes, attributes, operations, and relationships.

CRC card (Ref:2)/Class diagram



Readings

[1] R. S. Pressman and D. Lowe: *Web Engineering, A Practitioner's Approach,* McGraw-Hill, 2009.

- **Chapter 6: The Modeling Activity**
- **Chapter 7: Analysis Modeling for WebApps**

(concentrate on the topics covered in the lecture)

[2] Roger S. Pressman : Software Engineering: A Practitioners Approach. (8th Edition, 2014)

[3] Essential Scrum—Kenneth S. Rubin

[4]The agile age-Managing projects effectively using agile scrum---Brian vanderjack