System Analysis and Design

Eighth Edition

Alan Dennis, Barbara Wixom, Roberta M. Roth

Chapter 6

Moving into Design

Objectives

- Explain the initial transition from analysis to design.
- Create a system specification. ✓
  - Already in the process of doing this as part of analysis.
- Describe three ways to acquire a system: custom, packaged, and outsourced alternatives.
- Create an alternative matrix.

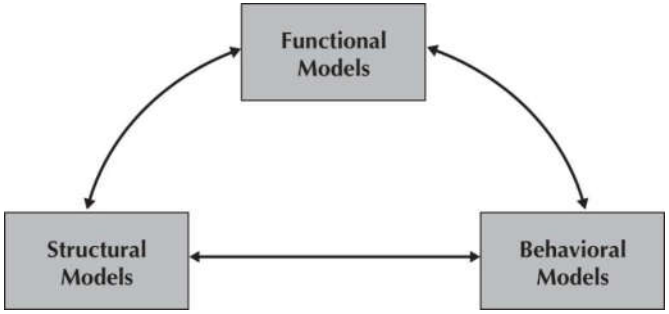
Content Outline

- Design phase
  - Decide *how* to build the system
  - Create ***system requirements*** that describe all technical details for building the system
    - This process was already started as part of the efforts we have done to date.
    - We created a draft Software Requirements Specification\
- Verify Analysis Models
- Design Artifacts
  - Update SRS, as agreed to with Stakeholders.
  - Final Deliverables are the Design Model and the Final SRS
  - Conveys exactly what system the development team will implement during the implementation phase

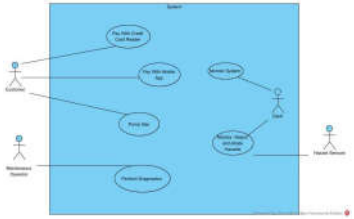
Transition from Requirements to Design

- In ***systems analysis*** we figure out…
  - What the business needs
- In ***system design*** we figure out…
  - How to build the system that fulfills those needs
- All the “logical” work from systems analysis is converted to the “physical”

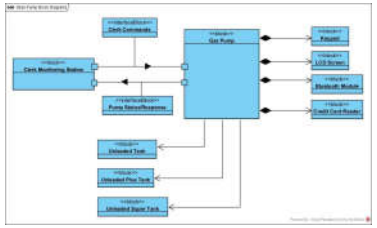
Models Thus Far



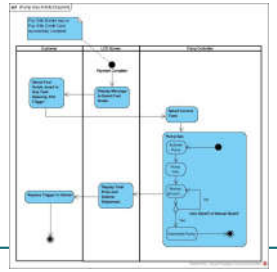
- **Functional Model** – Describe System to be built.
  - User Requirements Gathering Process
  - Use Case Cases
  - Use Case Diagram
  - Use Case Template Description



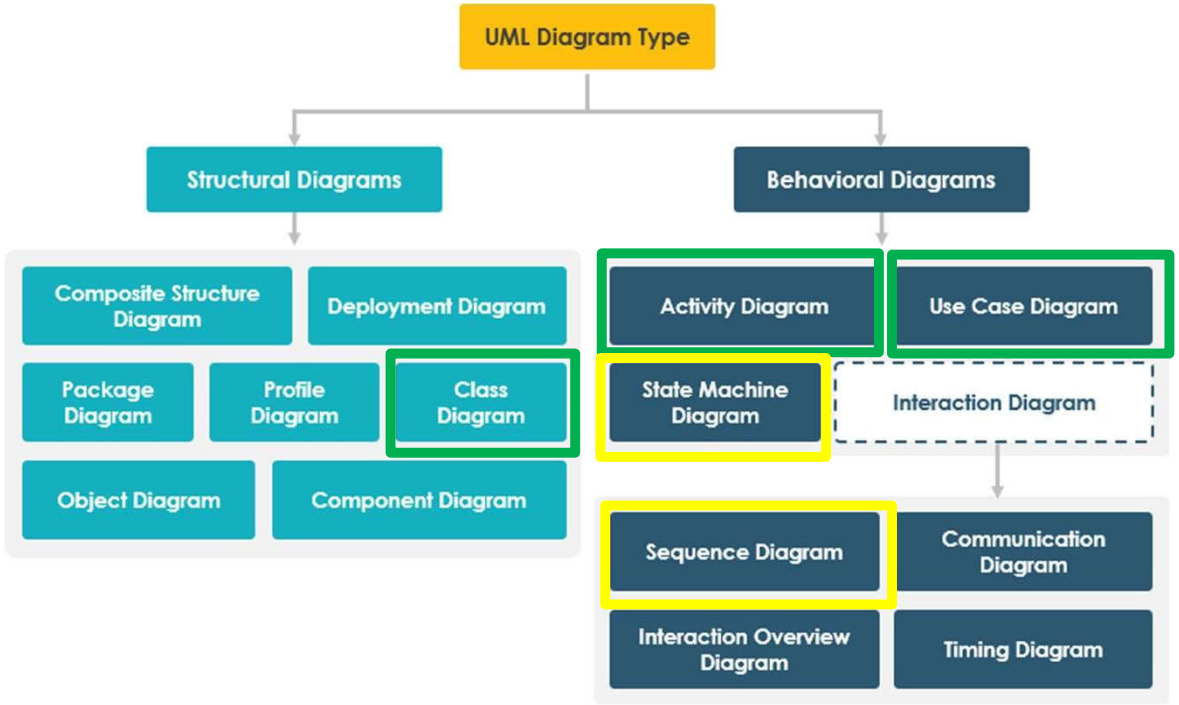
- **Structural Model** – Describe Overall Structure That Helps to Understand The System Context
  - Classes/Blocks
  - System Block Diagram
    - Class Diagram
    - Block Definition Diagram



- Behavioral Model – Describe User Interactions of the System to Model Requirements
  - Use Case Scenarios
  - Activity Diagrams



UML Diagram Types



UML Diagram Overview

Created

Will Create

Class Diagrams

- Same thing, more detail.
- During system analysis, class diagrams can be used to depict high level system components to help visualize the requirements in Activity Diagram.
- The “system level” classes are not meant to be implemented. They are only seen as a means to an end. The “end” being an analysis model for viewing our requirements.
- During design, software engineers view the analysis and develop classes that will be used to actually *implement* the *requirements* through *software level* capabilities
- Requirements analysis models are meant to be the “what”, software design information is the “how”

CRC Cards

- Class – Responsibility – Collaboration
- Used to document the responsibilities and collaborations of a class.
- Low-tech approach that can complement a typical high-tech Unified Process approach that uses Computer-Aided Software Engineering (CASE tools).
- We use an extended form of the CRC card to capture all relevant information associated with a class.

- https://agilemodeling.com/artifacts/crcmodel.htm

Responsibilities

- **Knowing** responsibilities are those things that an instance of a class must be capable of knowing.
  - An instance of a class typically knows the values of its attributes and its relationships.
- **Doing** responsibilities are those things that an instance of a class must be capable of doing.
  - An instance of a class can execute its operations, or
  - It can request a second instance, which it knows about, to execute one of its operations on behalf of the first instance.

Collaborations

- The structural model describes the objects necessary to support the business processes modeled by the Use Cases.
- Most Use Case implementations involve a set of several classes, not just one class.
  - These classes form collaborations.
- Collaborations allow the analyst to think in terms of clients, servers, and contracts.
  - A client object is an instance of a class that sends a request to an instance of another class for an operation to be executed.
  - A server object is the instance that receives the request from the client object.
  - A contract formalizes the interactions between the client and server objects. This "contract" is in the form of the public methods of the server class.

Back to Nouns and Verbs

- For design level class diagrams, review your uses cases, activity diagrams and even requirements.
- Pick out the nouns and verbs as before.
- “Discover” you initial set of class, attributes and operations
- Then, step inside these classes…..

Anthropomorphism

- Pretending inanimate objects have human characteristics.
- Step inside the object, and ask yourself these questions:
  - Who or what are you?
  - What do you know?
  - What can you do for me?
- Answer these questions while filling out the CRC Cards

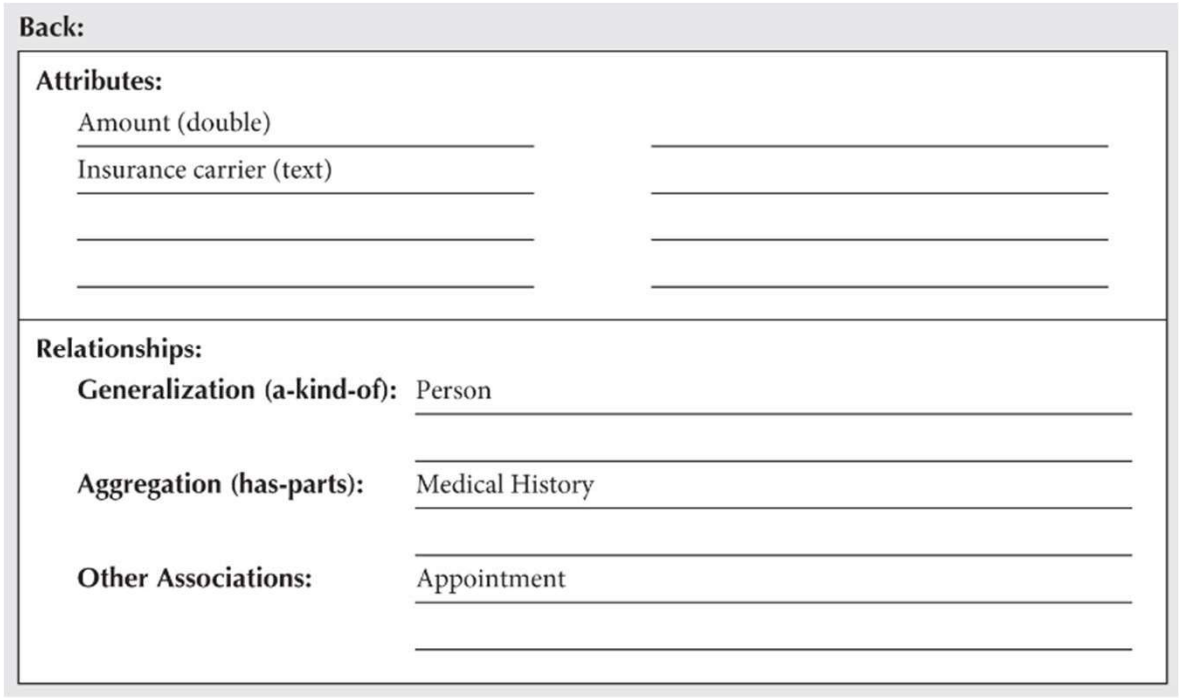
CRC Card Elements - Front

Front:

Class Name: Old Patient	**ID:** 3	**Type:** Concrete, Domain
Description: An individual who needs to receive or has received medical attention		**Associated Use Cases:** 2

Responsibilities	**Collaborators**
Make appointment	Appointment
Calculate last visit	
Change status	
Provide medical history	Medical history

CRC Card Elements - Back


Back:

Attributes:

Amount (double)

Insurance carrier (text)

Relationships:

Generalization (a-kind-of): Person

Aggregation (has-parts): Medical History

Other Associations: Appointment

CRC Modeling

https://agilemodeling.com/artifacts/crcmodel.htm

So how do you create CRC models? Iteratively perform the following steps:

•**Find classes**. Finding classes is fundamentally an analysis task because it deals with identifying the building blocks for your application. A good rule of thumb is that you should look for the three-to-five main classes right away, such as Student, Seminar, and Professor in Figure 4 (refer to webpage). I will sometimes include UI classes such as Transcript and Student Schedule, both are reports, although others will stick to just entity classes. Also, I'll sometimes include cards representing actors when my stakeholders are struggling with the concept of a student in the real world (the actor) versus the student in the system (the entity).

•**Find responsibilities**. You should ask yourself what a class does as well as what information you wish to maintain about it. You will often identify a responsibility for a class to fulfill a collaboration with another class.

CRC Modeling (cont)

•**Define collaborators**. A class often does not have sufficient information to fulfill its responsibilities. Therefore, it must collaborate (work) with other classes to get the job done. Collaboration will be in one of two forms: a request for information or a request to perform a task. To identify the collaborators of a class for each responsibility ask yourself “does the class have the ability to fulfill this responsibility?”. If not then look for a class that either has the ability to fulfill the missing functionality or the class which should fulfill it. In doing so you’ll often discover the need for new responsibilities in other classes and maybe even the need for a new class or two.

•**Move the cards around**. To improve everyone’s understanding of the system, the cards should be placed on the table in an intelligent manner. Two cards that collaborate with one another should be placed close together on the table, whereas two cards that don’t collaborate should be placed far apart. Furthermore, the more two cards collaborate, the closer they should be on the desk. By having cards that collaborate with one another close together, it’s easier to understand the relationships between classes.