Domain Models and Object Oriented Analysis

"The quintessential object-oriented step in analysis ... is the decomposition of a domain of interest into individual conceptual classes or object" [Larman, Ch 9]

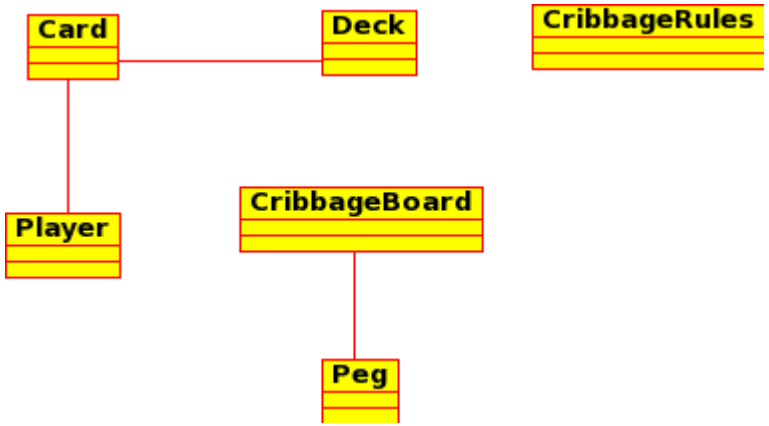
"A domain model is a representation of real-world conceptual classes, not of software components."

Domain modeling is a technique used to understand the project problem description and to translate the requirements of that project into software components of a solution. The software components are commonly implemented in an object oriented programming language.

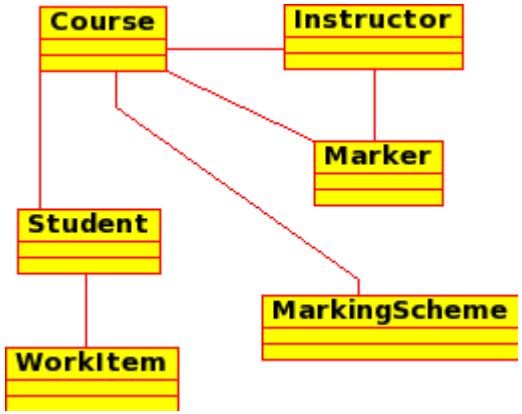
A domain model contains conceptual classes, associations between conceptual classes, and attributes of a conceptual class. "Informally, a conceptual class is an idea, thing, or object".

The model is shown as a class diagram.

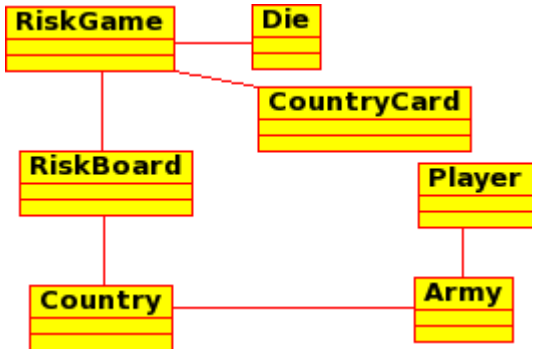
Cribbage Domain Model



Grade Report



Risk



Domain models

Domain models are the initial artifacts created in Object-Oriented-Analysis. An object has:

- identity,
- state, and
- behaviour.

An object can be

- related to other objects and
- complex (with sub-objects).

A class is a description of a set of objects that share the same attributes, operations/responsibilities, relationships and semantics.

Identifying objects in the problem domain is used to identify conceptual classes in the problem domain. Conceptual classes model entities in the problem domain, not in the software domain.

Identifying Conceptual Classes

1. Modify or reuse an existing model.
2. Use a conceptual class category list.
3. Identify noun phrases.

Categories in no particular order.

- Physical or tangible objects: quiz, game piece, die
- Specification, designs, or description of things: game piece image, marking scheme
- transaction: game move, order item, work item grade
- roles of people: marker, instructor, player
- container of things: game board, card deck
- events: start turn, finish game
- rules and policy: games ruler, submit assignment policy
- records: work item grade

Using noun phrases

Conceptual class can be identified by studying the use case looking for relevant noun phrases. Just like search term used in an Internet search engine, not all nouns are relevant.

Some heuristics for identification are:

- terms that developers or users need to clarify in order to understand the use case,
- recurring nouns (higher frequency),
- real world entities that the system needs to track,
- real world activities
- data sources or sinks

Category List for Associations

- A is a physical part of B. Memory is part of a CPU.
- A is a logical part of B. The p tag is grouped by a div tag.
- A is contained in B. A card is in a deck.
- A is a description of B. A house plan describes a house.
- A is a line item of B. A part in a part list.
- A is reported in B. An error event is reported in a log.
- A is a member of B. A faculty member is a member of a department.
- A used B. Tax rules are used to calculate the total in a point of sales terminal.
- A communicates with B. A computer communicates with a printer.

Heuristics for Identifying Associations

Some heuristics for identifying associations are:

- examine verb phrases,
- ensure that the roles and association names are clear,
- only add an association if it impoves the understanding of the domain,
- wait until the list of associations are stable before considering the multipicity,

Identifying Attributes

An attribute is a logical data (property) of an object (e.g., my eyes are hazel). Most attributes can be represented by simple data types (which are?).

Some heuristics for identifying attributes:

- an attribute is part of the state of an object (a car's speed is 100 km/h, weight of a work item)
- attributes are required by the use case (i.e., ignore irrelevant attributes).

Judgment is required to separate attributes from associated classes.

Making a domain model

1. Identify conceptual classes
2. Draw the class diagram
3. Add any associations between classes
4. Add attributes (properties) to the classes

Larman suggest that domain modeling should be similar to map making.

- use existing names (do not invent your own)
- exclude irrelevant features (that is the basis of modeling)
- do not add things that are not there.

Domain Modeling Practice

Use Case Name: create marking scheme

Actor: Instructor

Precondition: None

Flow of events: Basic Path

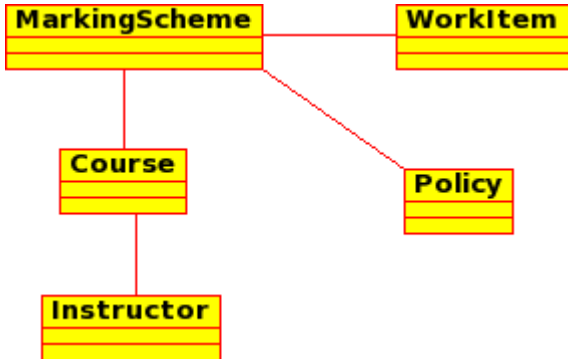
1. The instructor has selected create marking scheme.
2. The system prompts the instructor to enter the course name and the semester that the course will be offered.
3. The instructor enters the information.
4. The system then prompts the instructor to add a work item name, its maximum score, and its weight.
5. The instructor provides the information.
6. The system prompts the instructor to continue adding work items.
7. If the instructor agrees, step 4 is started again.
8. The instructor is then prompted for the late submission policy.
9. The instructor enters the policy.
10. The system saves the marking scheme and the use case ends.

Postcondition: A marking scheme has been created for a course in the specified semester.

A list of possible nouns or none phrases are: scheme, marking scheme, course, name, instructor, semester, work item, value, weight, policy, submission policy, system

Which nouns are conceptual classes, which are attributes to the classes? We can try using a conceptual class category list. Some of the category list items are: physical objects, transactions, things in a container, rules and policies, and records.

Domain model for create marking scheme



Which classes could be dropped?

More Domain Modeling Practice

Use Case Name: manage cribbage play

Actor: Cribbage player

Precondition: The system has registered ether 2 or 3 cribbage players.

Flow of events: Basic Path

1. The system informs the players that play is about to start and assigns the order of play randomly among the players.

2. While a player's score has not exceed 120, or has not quit:

    1. The games system shuffles the deck and deals the cards. Two player are dealt six cards, while three players are dealt five cards.
    2. In a two person game, each player selects two cards and places the cards into the crib. In a three person game, each player selects one card to place in the crib. The fourth card for the crib is dealt from the deck. Each player will receive the points in the crib in order. Only one player will receive the points during each round.
    3. The game deals one more card face up. This card is used to determine a hand's score.

    4. While the players have cards to play, the current player,

1. selects a card and plays it on the board, a card can only be played if the card value when added to the current sum does not exceed 31,
2. if the player could not play, the game announces Go, the current player's turn is skipped, and the next player in sequence is allowed to play, the events begin at the beginning of the while sequence,
3. the game system checks if the played card sequence results in any points being scored, if so, the points are added to the current player's score,
4. the next player in the play order becomes the current player.

5. The game then analyzes each players hand and the crib for the points present. The card turned face up is included in the analysis. The points are assigned to each players score.

Postcondition: The games is finished either by one player withdrawing or one player winning.

The nouns or noun phrases are: player, hand, crib, deck, card, points, and game.

A relevant set of conceptual classes could be: game, player, hand, and crib.

Discuss why points and deck were dropped.