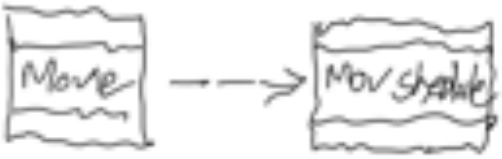
Relationships Between Objects and classes

Relationships

1. Decide info that objects know about other objects
2. Show how changes in 1 object affect another
3. Show membership in classes subclasses and superclasses

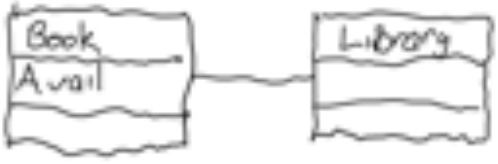
Dependency

- Weaker relationship
- One object depends on another
- eg like a movie shows depends on the movie schedule class



Association

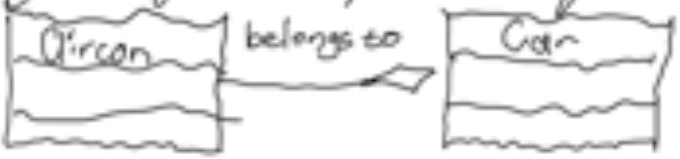
- Stronger relationship
- Changes made to one class affects the associated one
- When a book gets borrowed
  - Library class changes
    - Availability Status



Aggregation

- Stronger than Association & Dependency

- Object forms part of another



Composition

- Object made up of other object

Inheritance

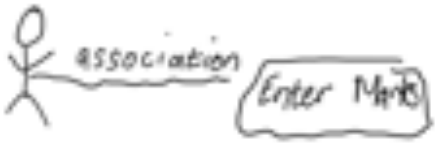
- Able to derive attributes from another object
- When a class extends another

1.3.5.3 Modeling Objects (UML)

1- Describe Usecase models

- Steps found in process or business

fuhtion
- External entities called actors
  - Request a process
    - called initiating a usecase
    - eg. instructor enter marks



2- Analysts and users define requirements

Use case diagram
- graphical representation
  - Between system & user
  - Shows what system will do not how

Use Case Components

Usecase Description

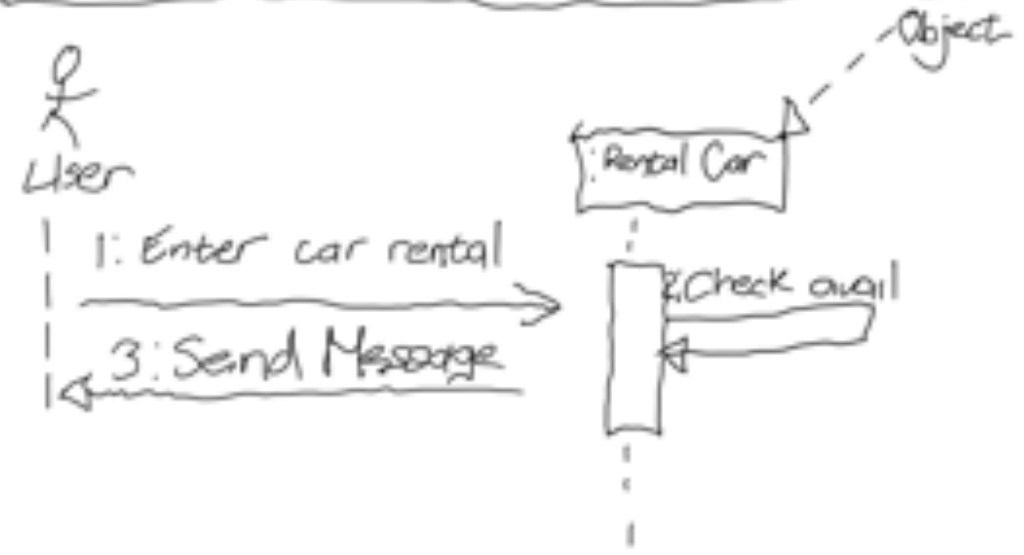
- Name of Usecase
- Succesful Completion
- Alternative to Completion
  eg. No cars to rent

- Precondition

- Postcondition
- Assumption

1.3.5.33 Sequence Diagrams

- Each usecase has desc of what happens during execution
- Desc known as scenario
- Describe usecase scenario with sequence diagram



- Show order in which obj.. interact in a certain timeframe

- Object — Rectangle
- Lifeline
- MESSAGES
- Focus — when messages are sent/received

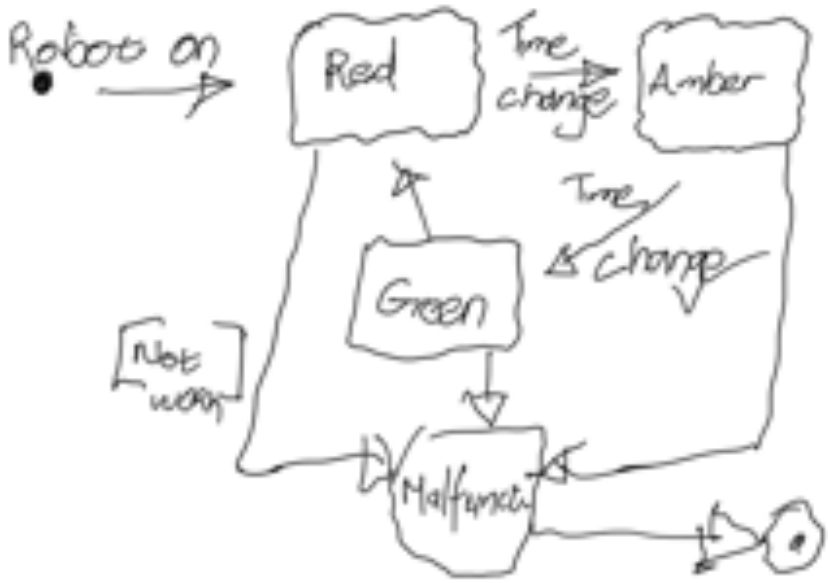
- time at which obj interact detect type where focus placed

Class Diagra

- logical model represents a single use case diagram in great detail
- Class divided into 3 sections
- cardinality (no of instances class can as
- multiplicity with another class)

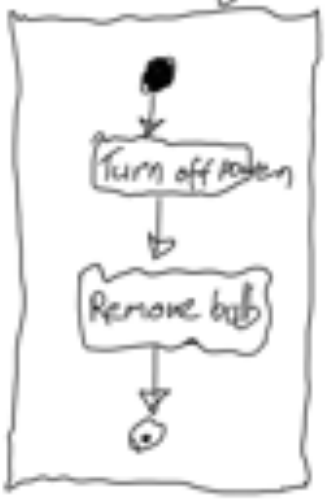
State Transition Diagrams

- State change based on events



Activity Diagram

- Show actions/events in order that they happen



1.3.6 Transition to Sys design

Software Alternatives

- In house
- Buying a software package
- Customize software package

- Horizontal application Software
  - Used by various orgs.
- Vertical application
  - focus on industry like bank

1.3.6.11 Concluding Sys Analysis Phase

System Requirements doc.

- New sys requirements
- Various options considered
- Single specific recommendation
- What sys dev must deliver
- gets presented to manager

Introduce System

- Introduce
- Summarize options considered
  - advantage
  - disadvantage
- Introduce Recommendation

1.3.6.13 Moving From Sys Analysis To Design

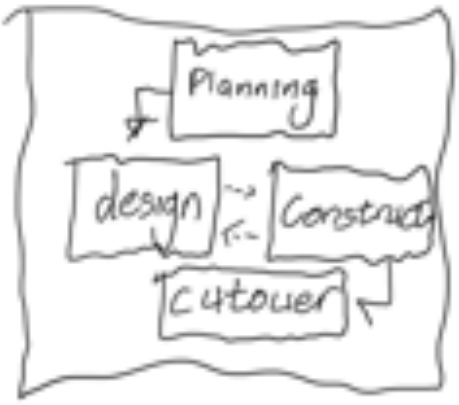
- Logical design (Model)
  - Reflects tasks system must perform not how they perform
  - Done during analysis phase
- Physical Design
  - Describes process necessary for implement
  - Based off logical design

Prototyping

- Sys Prototyping
  - Rapid app development (RAD)
  - Contains all features in the end

Rapid App Dev

- Associated with object orient.
- Reduce time spent at end of SDLC
- 4 phases
- Planning
- design
- construction
- cutover
- cutover
  - solution is presented and implemented



RAD

- Develop and deliver as quickly & cheaply as possible
- Same issues as prototyping
- Specifications gathered from JAD
  - documentation may be neglected
- SRD must be studied in great detail
  - first step of design phase

Unit 5 - Sys Design

- System Objectives
  - Reliable
    - cater for errors
      - input
      - processing
      - human
      - hardware
  - Maintainable
    - Easy to scale
      - add features
      - adapt to changing tech
  - Effective
    - Meet specific requirements
    - constraints
    - Users must approve of software
- Main aim of sys
  - Reliable
  - maintainable
  - effective
  - understand logical design
  - good sys design

    reliable maintainable effective

1.4.3.2 Considerations for Sys design

- later for interactions
  - user
  - data
  - processing

1.4.3.2 Users

- How will user use sys.
- Must be involved in development of system
- Key Points
  - Not be distracting
  - Straightforward
  - Output
    - clear
    - Easy to understand
    - Contain relevant / necessary info.
  - Interface
    - clear instructions for input
    - error handling
- future requirements will change
- Avoid hardcoding except for things like
- Hardcoded stuff is prone to outdate
- Parameters (user)
  - Very flexible
  - Increase reusability
  - Specified each time query is run
  - User could ask for same display

for 1 or 2 months
- Keys can start with default value
- Always keep user in mind

1.4.3.2.2 Data

- Capture should take place asap to avoid delays and errors
- verification as data entered
- Make use of codes
- Log of data entries should be kept
- Data should only be entered once
- Dont duplicate data

Processing 1.4.3.2.3

- Modular design
  - made up of modules which are processing components
    - form part of a larger program/process
  - Eg object orientation, classes and objects become the modules
  - easier to maintain
  - easier to upgrade
  - Reusable
  - developed, tested and corrected easily

UI 1.4.4

- User centered system

- focus on users
- focus on system functions
- HCI (human coputer interaction
- Easy to use and learn
- Test with prototypes
- Interface tech
  - operational structure
- Ergonomics
- how will people & computer interact

Data Design 1.4.5

- Data object - & diagram relationship
- file orientated systems
- use data files
- Also known as file processing sys
- Data redundancy
  - data duplication
- Data Integrity
  - Keep all data up to date
- Rigid data structure
  - file processing systems
  - Innefficient and slow when getting data from other platforms

DB Systems 1.4.5.2

- DBMS (db management sys) (MSSQL)
- QBE (query by example)
- Schema
  - describes entire db (fields, tables, data) (relationships, desc.)
- SQL

Subschema: limited view of info to different people

Data Warehouse

- Subject orientated
  - Organised according to topics
    - eg employees, customers, inventory
- Integrated
  - all data follows a standard
    - format
    - naming convention
- Non Volatile
  - Can't be updated by end user
  - Refreshed by sys customers use
- Time Variant

  Has a timeframe which can serve as timeline

- Data warehouse stores data centrally (transactionally)

Data Mining

- Software that searche data for patterns
- eg seasonal products

Data Design Terms 1.4.5.5.2

Entity
- Person, place, event or thing

-field
- A fakt or attribute about entity
- Common field
- Attribute appears in more than 1 entity

Pet Id (PK)
(FK)

- Record (tupple)
- Entity made up many tupples (key:val)
- describes single entity
- either arranged in files or tables

- File
- Collection of records that relate to same entity (in object orientation)

Table -

- 2 dimensions

Primary Key

- unique tupple
- Known as minimal key

Candidate Key

Can also sometimes be used as a PK

- Think ID Number and so CustId generated by system
- PK Should generally picked from field that has least data

Foreign Keys

- relationships formed by common keys
- Must exist as PK in other table
- can also favor PK in other tables
  - a unique combo of PetID, BreedID on another table
- Secondary keys
  - Fields that can be used to retrieve data
  - Values dont need to be unique
  - Type is a good example of

Secondary keys

Combination Keys

- made up of multiple fields

Relationships

- Logical links
- How entities relate to each other

1

E.R.D

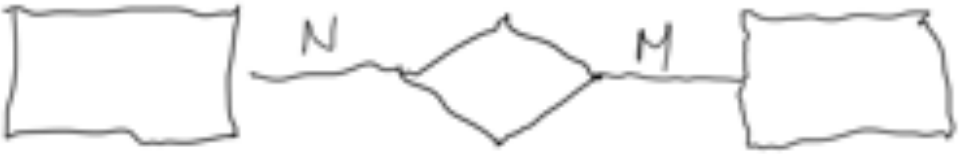

Noun Entity


Verb Relationship

Relationship Notation

one to one

1:M



N:M

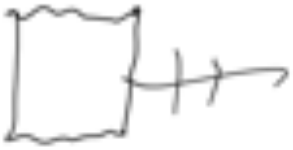
Crows Foot Notation

Single Line

- 1 instance

Double Line

1 instance only



One or Many

Zero, 1 or many

Construction of ERD

Steps:

1- Identify entities
2- Identify entities, attributes, events, transactions
3- Determine type of interaction
4- Draw ERD

1.4.5.9 Storing Data

- Lots of diffs between physical & logical data
- Smallest data
  - 1 bits is 1 binary digit
  - a set of bytes = 1 fields, data element or data item

Logical Record

- Record Name)
- Fields

Physical Record

Made of 1 or more logical records

- Also known as a block
- Smallest unit of data os can access
- Done by moving record from file/table to ? for buffer
- Buffer is section in comp. mem to store data
- Since the physical record or block is is made of logical records
  - Contains something called blocking factor
    - no of logical records in physical record

Formats for storing data

EBCDIC

- Mostly used on mainframe

ASCII

Unicode

- ints as chars
- 16 bit for each char
- important in sys design

Binary

- Stores numeric data more efficient than char

Application Architecture

- ERP
  - Enterprise Resource Planning
    - Establish strategy for resource planning
    - ERP defines an specific architecture
    - Includes standard that must be used in UI, data, network & processing
    - Describes platform
      - hardware & software env.

Supply Chain Management

- ERP sys extended to suppliers & customers

Archi. Planning

- 3 main functions
  - store & access data
  - programs to control processing logic
  - interfaces

Server - Client Architecture

- Trans server
- Web server
- DB Server
- Object Server

Client Types

- fat clients
- thin clients
- fat clients means client handles most of the processing
  - Simpler
  - Less expensive
  - Resembles file server architecture
  - Easier to dev
- Thin Client
  - Program code stays on server
  - Maintenance cost lower

Client-Server Tiers 1.4.6.8

-

- 2 tier
- App logic shared between client & server
- 3 tier
  - Client
  - App Server
  - Server (Data)

Completing Sys Design Phase

- Sys design specs
  - Everything needed incl. staff etc.
  - Doc is written for devs and staff
- Should include
  - Cover Page
  - T.O.C
  - Management Summary
    - Overview of objectives
    - Cost Summary
  - Details of the sys components
    - input
    - output
    - UI
    - program design
    - file & db design
    - network specs
    - other docs
      - O-O diagram
    - equip. requirements
    - Time & Cost
    - Appendices
    - Art Index

Application Development 1.5.3

- quality checking
- dev the app
- testing
- documenting

Quality Checking

- easier to find issues earlier on
- ISO (Int long for standardization)
  - Guidelines
    ISO 9000-3 which describes framework to assure quality when developing & maintaining software

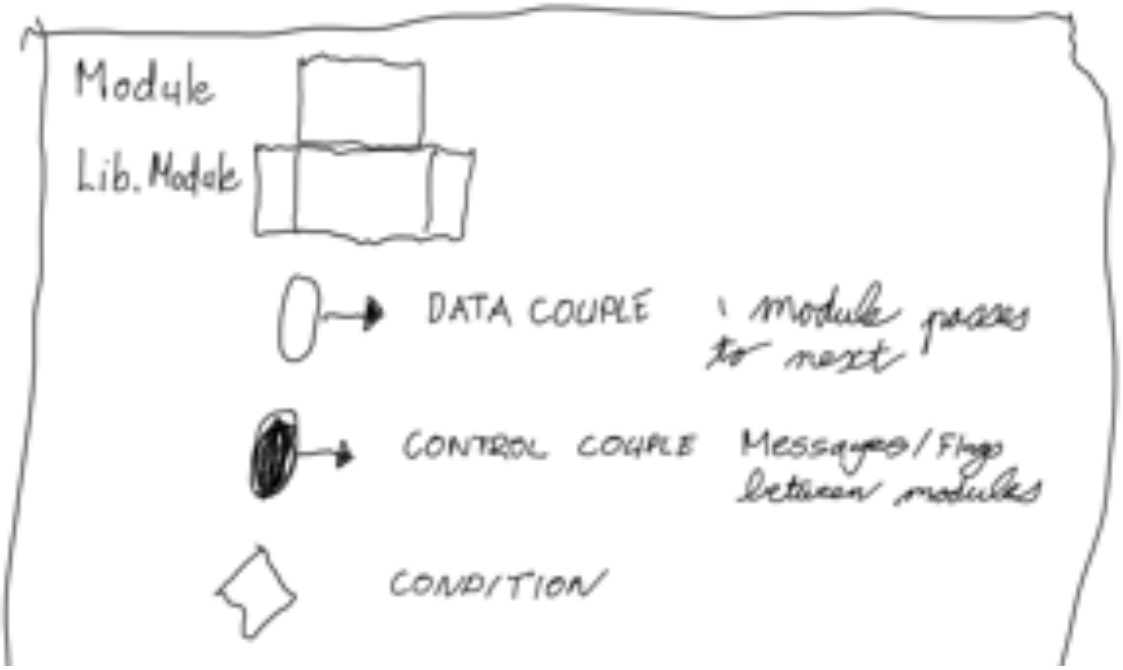
Developing The App

- Software engineering is a dev process that stressed solid design, effective structure accurate docs and testing
  - Always first plan design

Structured App Dev

Top down design

- General → More detailed design

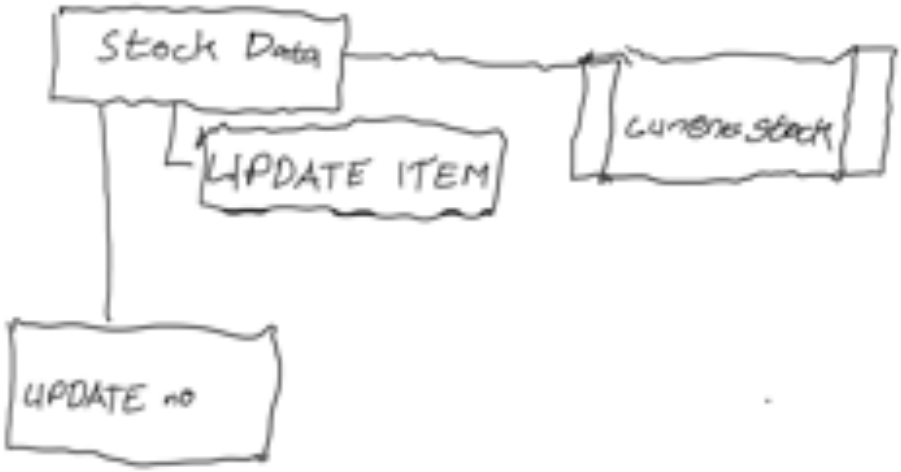


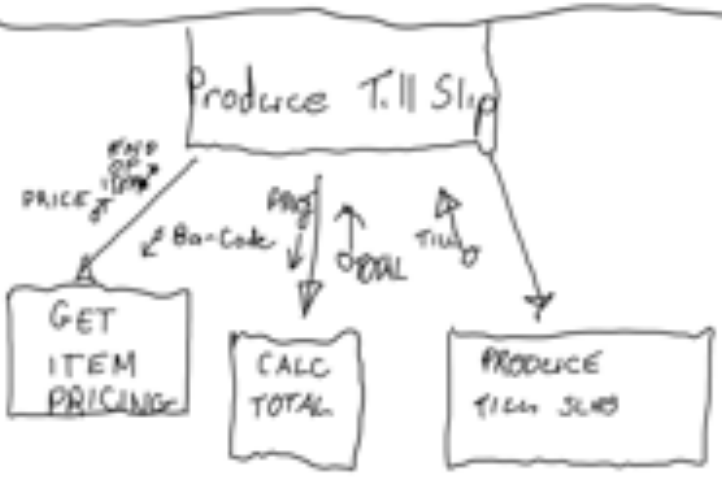
→ LOOP

Control Module

- decide when sub modules operate

Structure Chart





↗ - data

in structure chart.

- Cohesion
  - Amt of processing module has to do and scope of module
  - If module performs single task, its considered cohesive

O-O Appr Dev

- Object model can translate into oo lang directly
- A structure chart keeps track of relationship between modules
- Analyse sequence and state diagrams

Testing

- Stub testing
  - mock out dependencies like vars
  - either an entry or exit point
- Test Plan
  - How
  - When
  - Who
  - What
  - Should cater for all possible situations
- Integration Testing
  - 2 or more apps depend on each other
- Acceptance testing
- System Testing

- Test all situations that could occure

Documentation 1.5.26

- Program Docs
  - Starts in analysis phase
  - Includes overall docs that began in early SDLC phase
    - process desc.
    - Screens
    - Report Layout
    etc..
- Sys Docs
  - Describes funct. of sys
    - Implementation
    - Includes
      - Request that initiated proj.
      - D.F.D.s
      - Object model
      - Data Dictionary
      - Screen Layout
      - Source Docs
        - flow Charts
        - UML Diagrams
- Operations Docs

- Scheduling info
- Input / Output files
- Non standard forms required
- Security stuff
- Report Distribution
- Error messages / notifications
- User Docs
  - Sys Overview
  - Source Documentation (contents, prep, samples)
  - Menus & Data Screens
  - Reports
  - Security Info
  - Input, output, processing req.
  - FAQ
  - Audit trail
  - Who to contact

Approval From Management

- Test results handed to manager
  - Results
  - Status of docs
  - Testers & Test summary
  - Schedules
  - Budgets
  - Staff Rep

- Environments 1.5.4.1
  - Hardware & Software form env or platform
  - prod environment also called operational environment
  - Should be tested on staging
- Implementing New Sys.
  - Sys Changeover
    - Direct Cutover between 2 sys

      Old sys → NEW SYS
    - Pilot Operation
      - Only implemented in certain part of Org.
      - First group of users called pilot site
    - Parallel Sys
      - Both old & new operate same time
      - Only new sys outputs used when possible
    - Phased
      - New sys in stages
- Post-Implementation Eval 1.5.4.6
  - Review by users
    - by org.

- Achieve goals set out
- NB for future projects

- Usually Includes
  - User Satisfaction
  - Completness, Accuracy, Response times
  - S.T learns performance
  - How effective training was
  - Reliable and maintainable
  - Costs
  - Same techniques used in evaluation
  - Timing is NB.
    - Dont use biased people
    - Wait a bit so people can get used to it

Final Report

- Report
  - Should contain all docs

Maintainance

- adaptive
  - add features
- Corrective
  - fix errors

- Preventative
- Perfective
  - Improve efficiency

Releasing Maintainance Change

- Version control system
  - New version with changes called maintainiace release

Selection 2: oo analysis & design

- Classical approach used structured methods
  - Program flow charts
  - large programs had large flowcharts
  - can either follow data flow or course of action
- OO evolved where both data & actions equally important
  - standard design notation
    - UML
    - model structured and OO code

U.M.L

- UML endorsed by FAANG
- OMG.COM
- Most common diagrams

- class
- object
- state
- sequence
- activity
- collaboration
- components
- deployment
- Packages
- notes
- stereotypes
- patterns

- OO models business system

- SDLC (PADIO)

P - Planning

- Sys Request

A - Analysis

- Set out sys requirements

D - Design

A model needs to be created that shows new working situation

I - Implementation

- Building sys

O - Operations & Support

Analysis & Design Team

- OO Analysis & Design can't be done by 1 person

- More users involved in dev process the better the outcome

- Facilitator
  - runs session
    - Makes sure everyone contributes
  - Needs some OO design skills

Business Domain Expert

- People on the floor using the product most
  - Not really managers
  - Know more intimate details of what should happen

Object Design Analysis

- 1 or 2 people who know object design
- Scribes

Analysis & Design Tasks

- Modelling team
  - Vision statement
  1. Object analysis
     - Team asks exactly what wanted
     - Puts together analysis of the requirements of the sys.
     - How sys meant to behave captured in usecases
       - Captured with usecase diagrams

- Shows interaction user ⇔ software

2- CRC (class, responsibility, collaborator)

- Model
  - Identifies various objects with which users interact
  - depicted with use case diagram
  - CRC cards converted to class diagrams

Waterfall vs Iterative

- Waterfal
  - each phase signed off and considered final
  - One vision statement is considered final, it cannot be changed
  - We then move on to requirements analysis
- Iterative Approach
  - Sometimes scope estimate is to wide or narrow
  - The change in the vision statement feeds back into requirements analysis

Iterative approach to modeling

Unit 2 Obj. Analysis

A use case defines how sys is used and what its expected to do

Also called requirements analysis

Usecase Analysis 2.2.1

- Detailed specs of operations and tasks carried out by the program
  - What will it do and how
  - How will sys be used
  - Communicate sys functionality
  - Model how sys interacts with surroundings
    - At which point sys & user exchange info
  - Sys or program being developed seen as black box
    - How sys achieves goal / results not NB yet
  - Examine how sys reacts from outside
  - Usec represent sys as whole ors sys in sys
  - Primary flow be events should first be documented & modeled
    - high priority & high risk first
  - Boundry of usecase and more detailed usecase explanation include
    - UI look & feel
  - Define what app outcome should be

  Defined in supporting docs.

- A usecase has a name
- detailed desc called a scenario
  - Written in simple clear lang.
- Iterating design involves testing by actual users during development stage
- Usecase model 1% of doc.

Actor

- Anything/one interacting with sys
  - External to sys aka api or human

areas

- Has goals sys should help deliver upon
- Passive actors just receive data
- A person who interacts with sys in 2 ways considered 2 actors
- A usecase can be said to model any 1 or anything (called an actor)

Use cases & Scenarios

- Start with verb (drawing cash)
- A scenario is an instance of a usecase that provides details on how usecase happened from start → end
- Each usecase may have 1 scenarios
- Scenarios grouped on usecase they are serving/describing
- Essential and elaborate scenarios
  - Deliberately excludes all reference to exact nature of UI
    - eg. usecase check mail
    - an essential scenario
      - client opens mail client
  - Elaborate scenario
    - Includes UI details
    - Many elaborate scenarios for essential scenario
      - Launch using mouse or keyboard
        - 2 seperate scenarios.

Primary & Alt scenarios

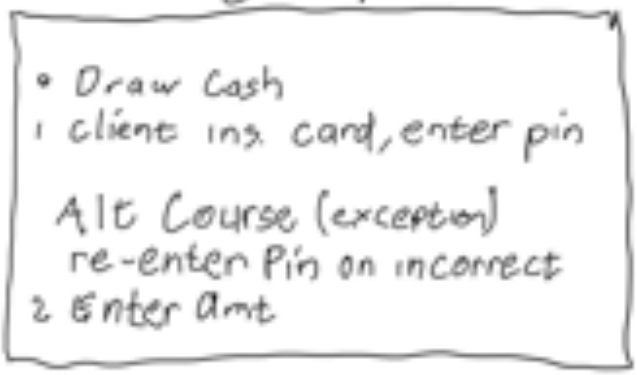
Capture primary all goes well scenario

Primary scenario for draw cash

- Check if user has cash
- Enter pin

Alternative scenario

- User gets pin wrong



USE CASE SCENARIO

- How to do a usecase analysis
  - Usually iterative
  - Overlaying vision may change
- Finding the actors 2.2.2.1

  Simple route to doing use case analysis
  - Start with vision statement
    - Allow team members to Id the actors and systems that will interact with new sys
    - Brainstorm with actors

- 1 person can serve as multiple actors if they interact differently with same sys
- Remember actors are external inputs or something external receiving sys output

Writing Usecases 2.2.2.2

- Happens once actor brainstorm is complete
  - Evaluation begins
- Ask each actor what role they play and determine if still necessary
- If actor has role to play
  - usecases and scenarios written
- Active voice used to describe usecases and scenarios
  - User enters code rather than code added by user
    - From this usecase, enter code is derived
- Way to also determine actor
  - 90 business or main events likely to occur in sys.
  - If actor has role then use cases & scenarios are written for actor
- Who or what initiates usecase (occr)
- Who, what benefits from usecase

Writing Scenarios

- follow event response model

- When scenario written
  - include initiating & benefiting actors

• Draw cash
1 Insert pass & card
  Sys verifies card & pass
  Alt course
    If pass denied 3 times confiscate card
2 Client enters amt to withdraw
  Sys verifies client has funds
    Alt course
      display no funds
  Sys verifies machine has funds
    Alt
      display machine empty

2.2.2.4 Questions

Finding Actors
- Who suggested sys
- Where will app be used in business
- Who maintains and supports sys
- Who/what provides info needed
- Who/what uses produced info from sys
- Who will want to look info up
- Does 1 person play multiple roles

- Do multiple people play same role

Finding Usecases

- Main business event of system
- What does a certain actor do
- Will actor capture info or want reports and data
- What needed to support/maintain sys
- Any external changes that actor would need to enter into sys
- Are all requirements met by these usecases

Usecase Diags 2.2.3

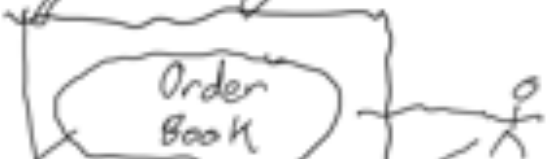
The usecases can be represented graphically Show relationship between actor and usecase

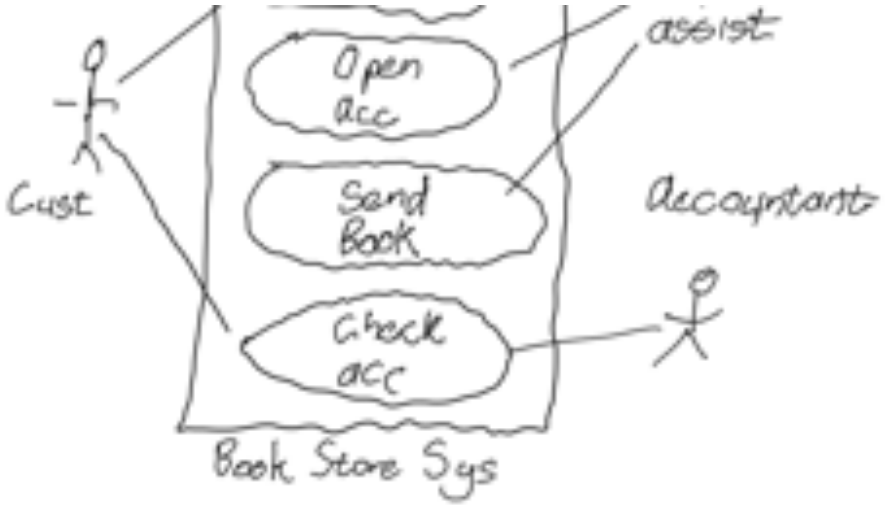
—— Interaction

- Shown using block

Lower order usecase diags likely exist They focus on finer actions

Advantages of use case analysis





Example

Track Marks & Payments

First way

- Determine main events in sys
- Write usecases
- Write Primary scenarios and any alts
- Identify actors
- Draw usecase diag

Alternatively

- Find actors
- Write usecases
- Write pri/alt scenarios
- Draw usecase diag

First Method 2.2.5.1

- Tracking marks and payments of students
- Writing usecases
  - Enter student mark
  - * people view course mark of student
  - Payment entered
  - Payment queried
- Below are the derived usecases

- Enter mark
- Display mark
- Enter payment
- Query payment

Writing Primary Scenarios & Alt

- Enter Mark

Instructor inputs student No, module no and mark

The sys checks if student exists & is doing module

Alt course

Student no doesnt exist
- display error msg
- Returns to menu

Alt course
- Student not doing course disp. message

- Enter Payment

Admin enters student no and payment info

Sys checks student exists then adds payment to schedule

Alt course

Student no doesnt exist, throw message

- Display Mark

User enters student & course no To see mark after requesting a mark be displayed

Sys checks if student no exists & course taken by student then points report

Alt course

Student no doesnt exist, display msg

- Query Payment Status

Admin carr query acc or payment
User enters student no

Sys checks user no, displays acc

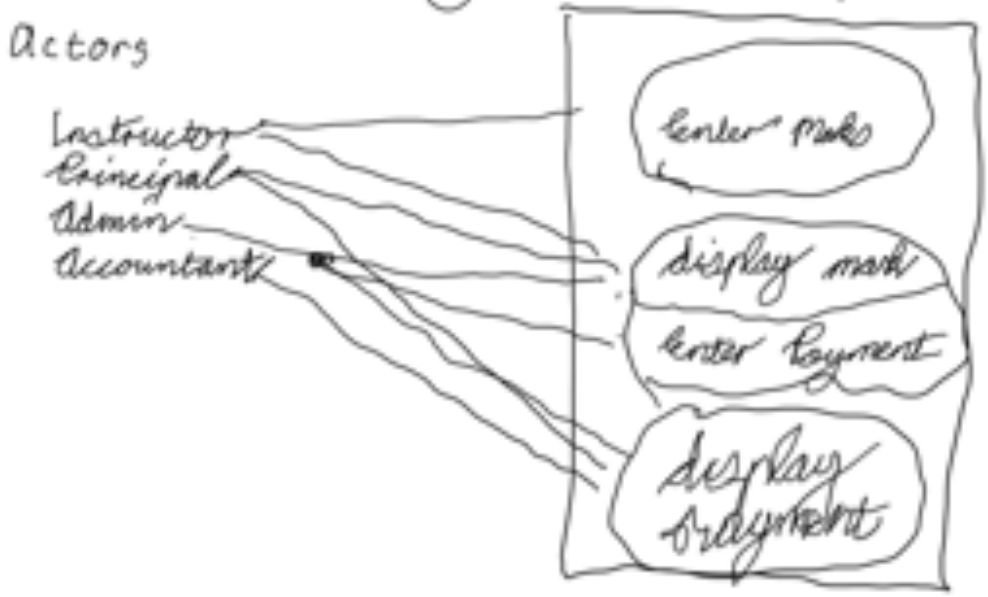
Alt course

Student no doesnt exist, throw error

Finding Actor

Determing by who/what inputs data based on usecase decided

Use case Diagram 2.2.5.1.4



Second Method

Finding the actor

- Brainstorming could produce actor list
  - Instructor
    Admin
    Accountant
    principal
- Writing usecases

Take each actor, create usecases and scenarios

- Same usecases should be produced
- Same with primary scenarios & alt
- Same with usecase diagram

First Method

- Find main events for sys

Second

- Brainstorm actors base usecases on them aswell as scenarios

Object Design

Object Analysis

- focus on understanding the problem
- Getting out reqs for new sys
- Transfer understanding of to design that can be implemented

with code

Objects & classes

class

- Is template
  - Abstract description
  - define attributes & behaviours (characteristics)
  - Blueprint of a concept
  - class diagm
    - Reflect class relationships

Object
- A thing classified and defined by class
- either;
  - Physical
  - Conceptual } entities
  - Software
- consists of
  - ID
  - State (is something on or off or set)
  - Behaviour (Predefined operations obj. can do)
    - assignMark()

- Usecase thought of as a class
- classes are identified from usecases

Attributes & Behaviour

- Attributes (adjectives)
- Behaviours (verb)

- Attributes
  - known as characteristics/state
- Behaviour
  - What an object can do
  - Is a function
  - Obj data & obj functions make up software object
  - Obj also has data
  - Obj is instance of class
  - Class contains a list of data members and member functions
  - Objects exist in memory, classes do not

- class describes thing that will be created

Objects, Classes, Instances

- Objects instances of classes
- class definition describe object
- A class should capture one key concept or set of data
- Classes should hide data from other objects
- Other classes should only reference functions of a class not data
  - Called Encapsulation
    - Allows interface to be visible to to other methods
      - Accessor function = public function
        - gives data to other classes
    - Benefits
      - Get to change data without knowing how its done in the background
      - Write code for class independent of other classes

Identifying objects & classes 2.3.1.4

1 look for objects (using CRC cards
2 Group objects
3 Form classes based on general characteristics

CRC Modelling

- At heart lies use cases and scenarios
- Stands for class, responsibility collaborator

CRC cards 2.3.2.1

- Back used for notes

CLASS	
RESPONSIBILITY	COLLABORATORS

Class data members Functions		

- Nouns in usecases make good class names
- Capitalise first letter
- Singular noun
- CamelCase for multi word classes
- Hard to name classes could be poorly defined
- Either source of classes is usually interface between new and old sys
  - Allows for transformation of data from 1 set to another
    eg data input in menu class
    - Menu class associate with other classes to perform funct offered by menu
- Easier to pick out common classes first
- To find responsibilities ask what a class knows and does
- Look at data members
- And what functions it will have
- Finding collaborators for a class involves looking for classes that have info required to fulfill responsibility of the class
- A class should have 2+ collab class

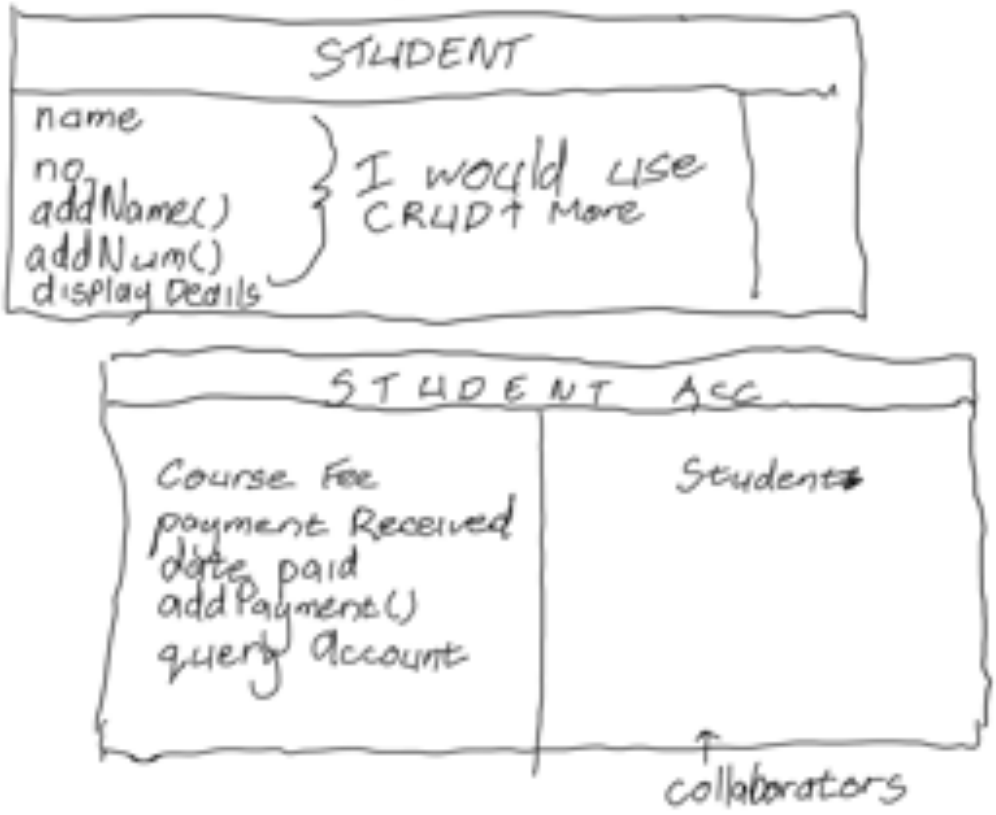
Something is wrong when

- class has no responsibilities
- Have responsibility many classes
- All classes collaborate with all other classes

Using CRC Modelling

- Based on usecases & scenarios developed in object analysis phase
- Normally 4-6 people
  - users, analysts and facilitators

1 Start with 1 scenario, the main 1 (basics)
2 CRC cards created
  - ask questions from user & class view
  - questions asked to users about everyday environment.
  - lay cards out so busiest is in the middle and collab classes are close together
  - not the best for massive systems
  - One big advantage is that it .. leads to class diags
  - Programmers don't IA classes, users & others attending do

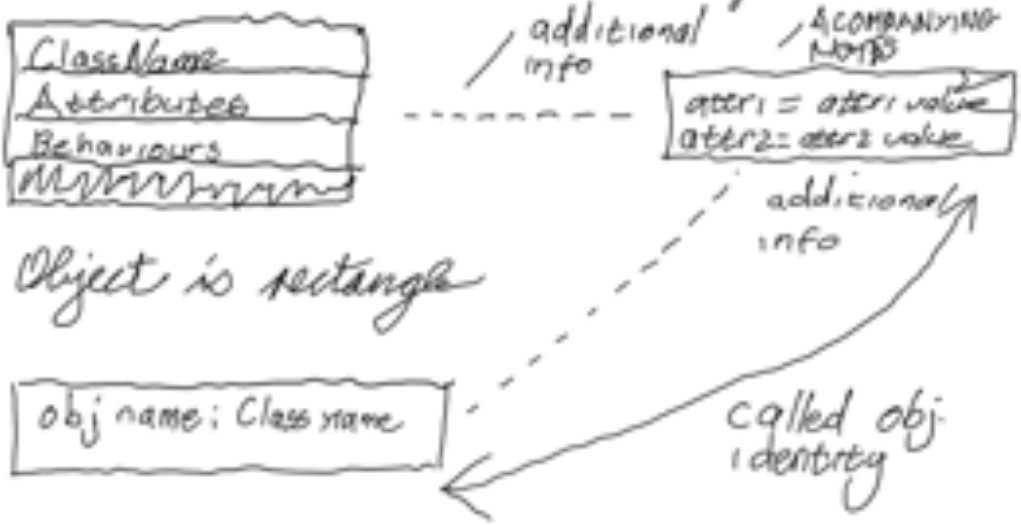


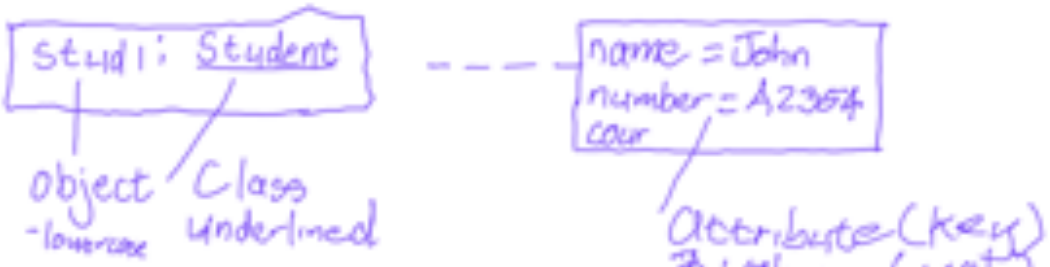
Object Relationships & Class diags

2.4.1 Object relationships

types

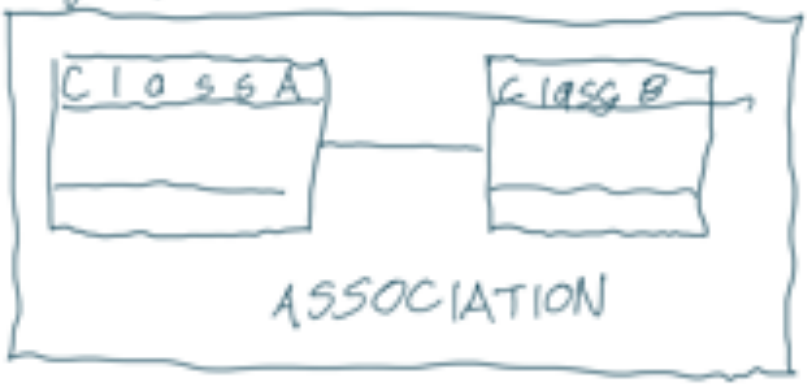
- association
- containment
- generalisation
- multiplicity
  - adds detail to relationship





Association 2.4.1.1

classes relate through association
- used when 1 class uses another to fullfill tasks



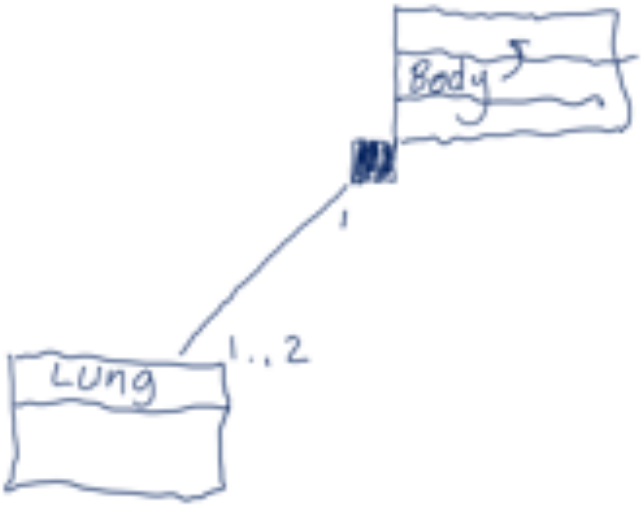
Containment 2.4.1.2

- Object composed of subobjects

- types
- Aggregation
  - Containment where sub-objects made of various parts
  - Often see the term has
    - car has tyres, doors etc
  - Shown with open diamond
  - All these parts aren't compulsory to populate

Composition

Parts are mandatory
Indicated by solid diamond
Parts get destroyed when whole gets destroyed
- Objects need to exist simultaneously



Gui design good example
- Window needs buttons etc for it to work

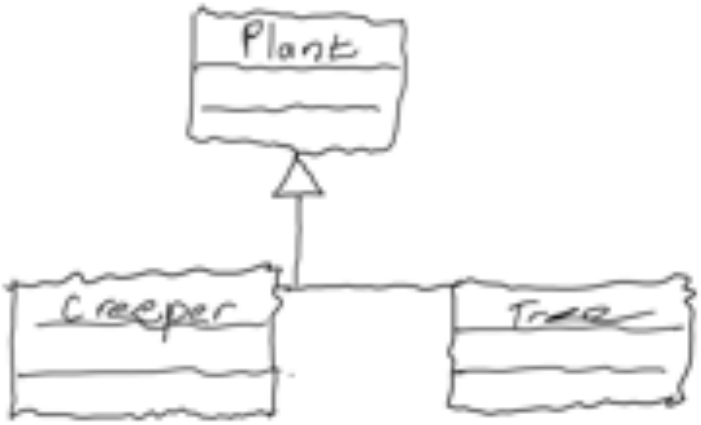
Multiplicity
- Cardinality
  - How many instances of 1 obj relates

is around

Multiplicity between 2 classes

Generalization 2.4.1.4

- Defined at class level
  - 1 class shares structure of another
  - (Kind of) or (- of) relationship
  - Known as inheritance
  - Plant is generalisation of creeper
  - Inherits attributes and functions from base or superclass
  - derived class or subclass is the class that inherits
  - Single inheritance
    - Subclass only inherits from 1 base class
  - Multi inheritance
    - Inherits from >1 base class
  - Generalisation is shown by a hollow headed arrow



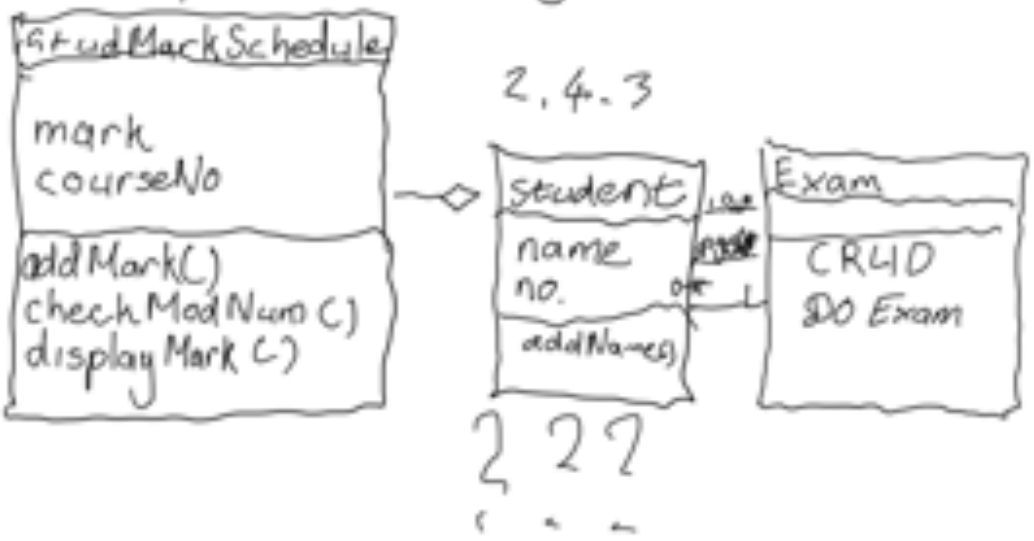
Inheritance should be used sparsely

- Buttons on gui great candidate for inheritance

Class Diagrams

express relationship between classes

- Way to capture the model that will be developed in a diagram



Object Interaction

- Object in real world take on a state
  - Represented by attr. values
- Object interaction code will call method on other object
- UML collab and sequence diagrams
  - Show dynamic (changing) behaviour of systems
  - both show diff. view of same obj
  - Sequence
    - objects ordered timewise (what happens + time)
  - Collab (when & how long)
  - Show structure of objects (how they are related)
    - How they are linked
    - maybe data flow
  - UML diagrams
  - Obj diagrams represented with collab diagram without interactions

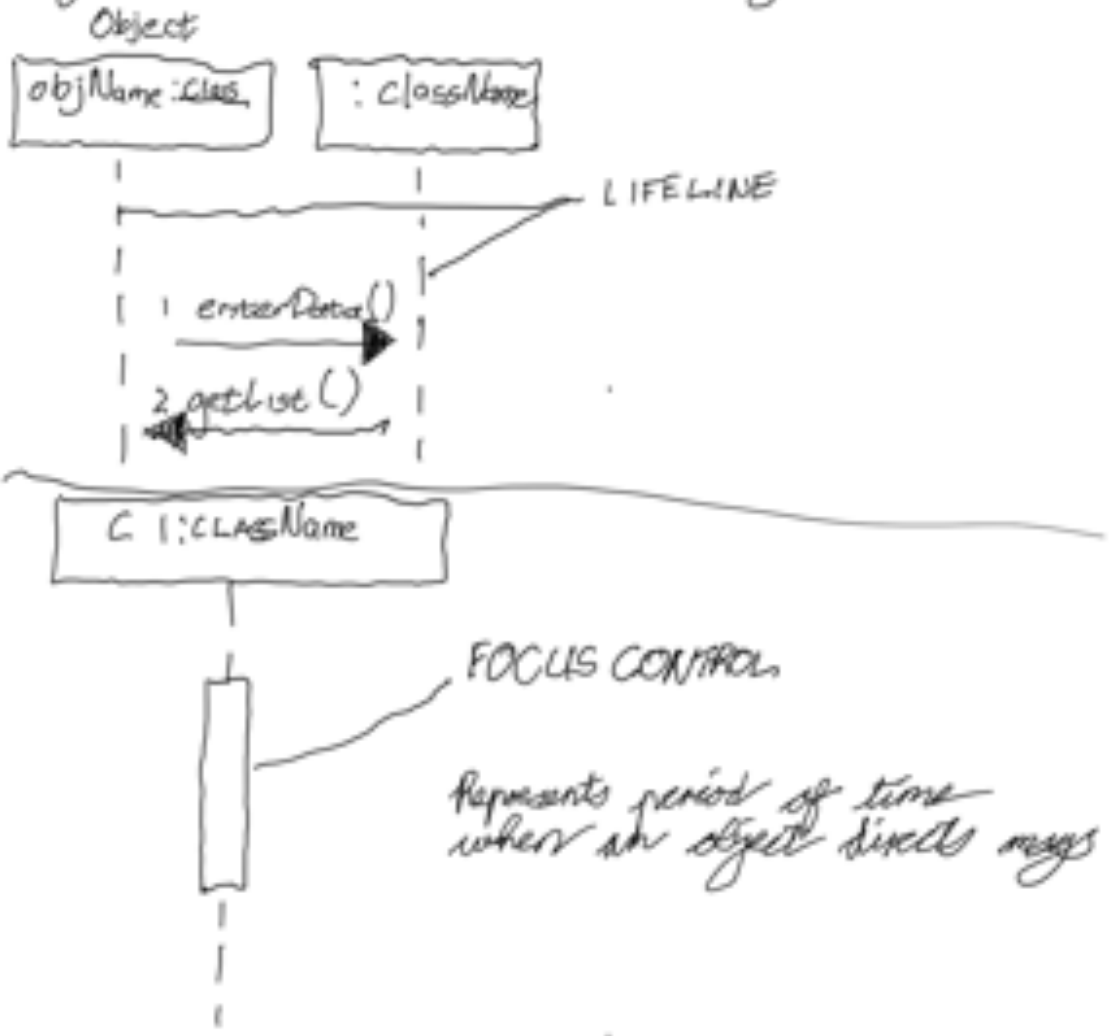
Sequence Diagrams 2.5.1

- During analysis phase
  - capture system use

- capture scenarios

- During design phase
  - How interaction is achieved between objects
  - Usecase scenario provide sequence of events as obj coms over time, this is shown in sequence diagrams
  - Sequence diagram focuses on time
  - Events occurs from top to bottom
  - Emphasis on sequence diagram is interaction between objects and not whats happening in objects

Symbols used in sequence diagrams



- Lifeline shows how long object lives
- Phrases in usecase scenario converted to function names in sequence diag.

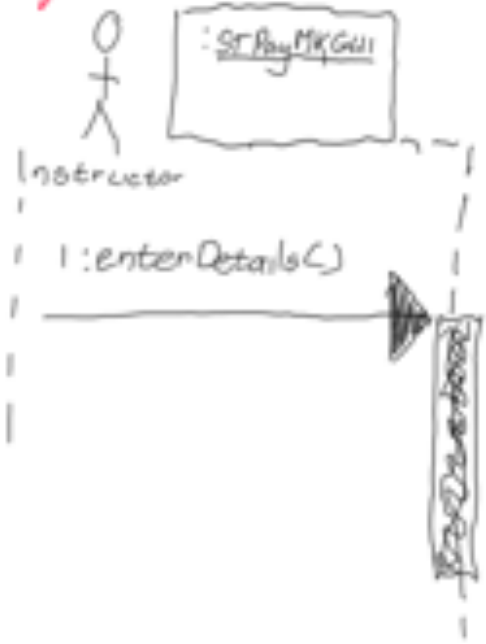
2.5.3 A Controller

Between OS and printer and [illegible]

considered controller

- Special class that handle running of the program.

In structured code controller would be a function from which other functions are called.



Collaboration Diagram 2.5.5

- Same interaction that sequence diag. shows from diff angle
- sequence diagram is top down (side)
- (collab diagram) = bird eye view
- Aims @ showing which objects collaborate with others to achieve the objective
- Shows how objects that send & or receive data are organised but does not show logic
- Object represented with rectangle

can also be represented

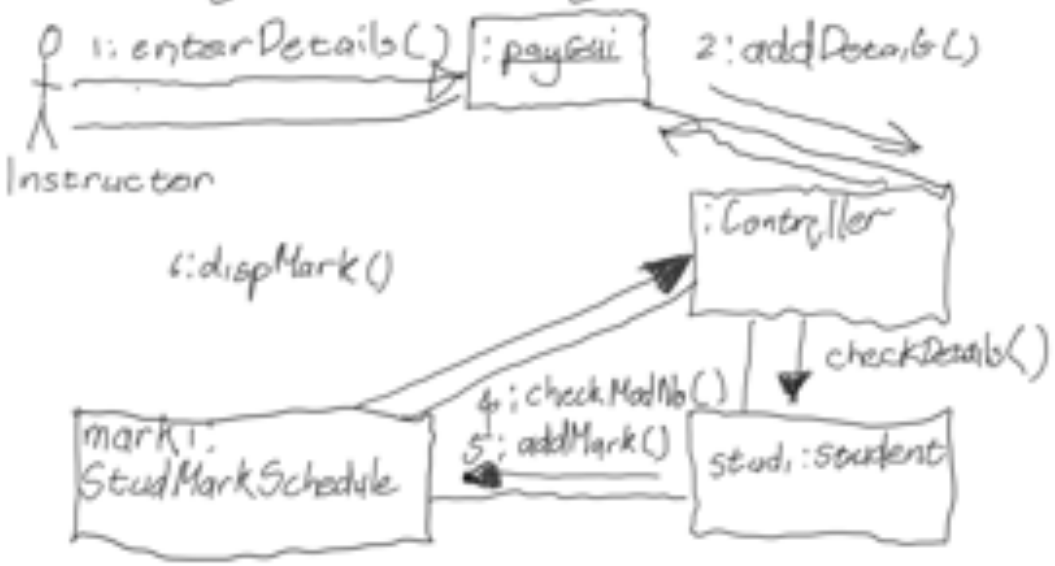
Link

enterData(id, Mark)

message

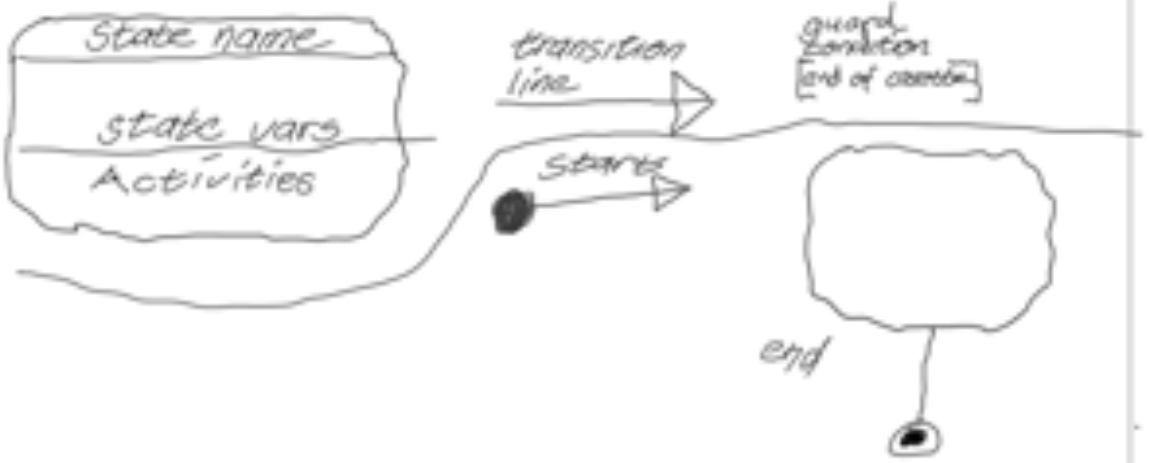
- Collab digrams will be easier if components are drawn in order

- 10 objects
- draw rectangles in logical order such as being placed in corner
- connect objs with lines
- add msg that the obj sends & receives



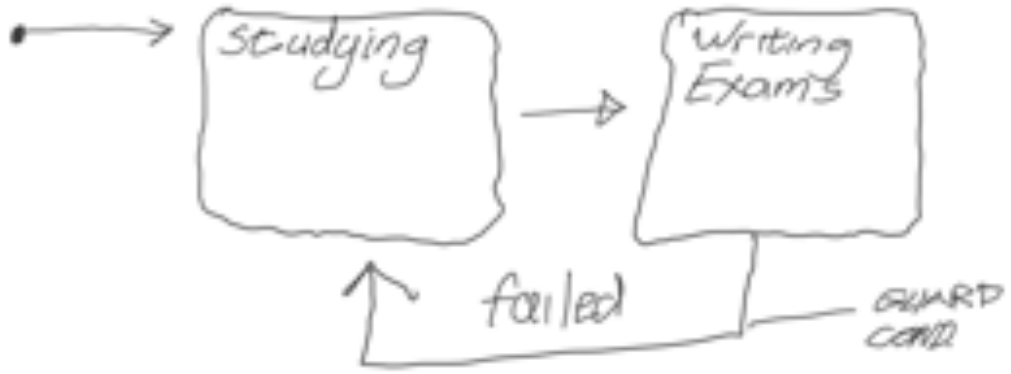
State Diagrams

- also known as state machine
- capture workflow
- shows lifecycle of an obj from creation to destruction



Guard Conditions

- Allow object state only in certain state when condition is met
- eg. Student failing will trigger studying condition



Interaction Diagrams

- Show no logic of a process
- Logic found in focus of control shown as a program flowchart

Where logic could be shown in sequence diag.

- In UML logic is shown in activity diagram
  - shows beginning, the steps, decision points, branches, end of process and things causing obj to transition
  - Similar to flow chart but shows parallel processing
- Start early in project
- May be used to help id components of system model

- May also show the whole sys flow of control from 1 activity to another
- May also be used to id classes

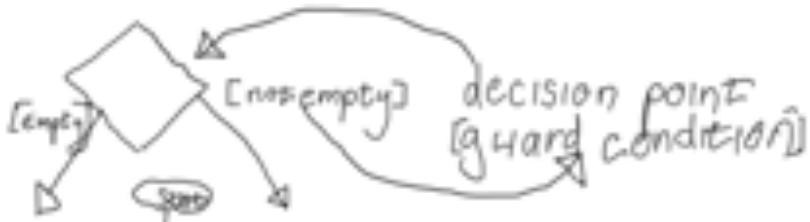
2.7.1 Symbols used in activity Diagram

activity

start point

end point

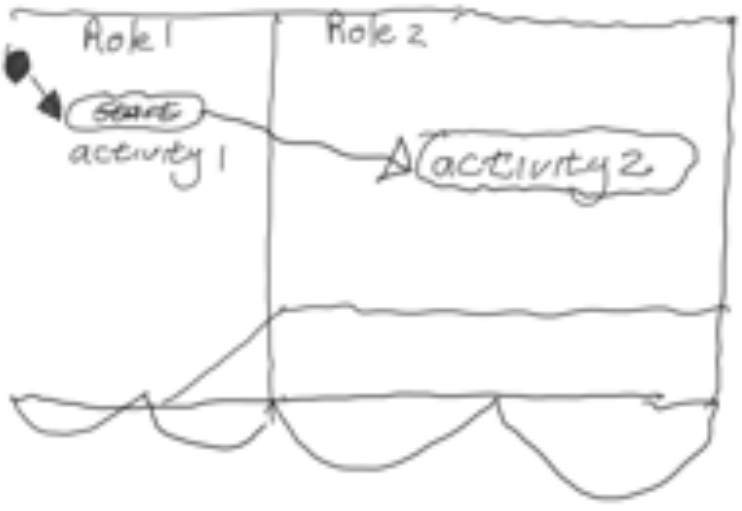
transition from 1 activity to another



two or more paths leave

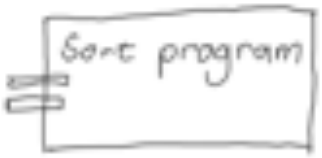
another bar would show where they join

Concurrent paths



Swimlanes

Component & deployment diagrams



A component, with function desc

Menu — Usr interface.
- connected to node by solid line

«word»
node shown in guillemets
- devices may be shown as nodes

Node

----------> Relationship between components

_____ association between nodes

 Internet

Component Diagram

- How software components
  - organised
  - relate to / another
- Describe physical things that make up sys
- Software component
  - independant piece of code
  - reuseable

- Interface may also be shown as a little circle

Deployment diags

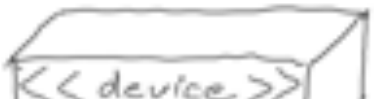
- Shows how hardware is setup

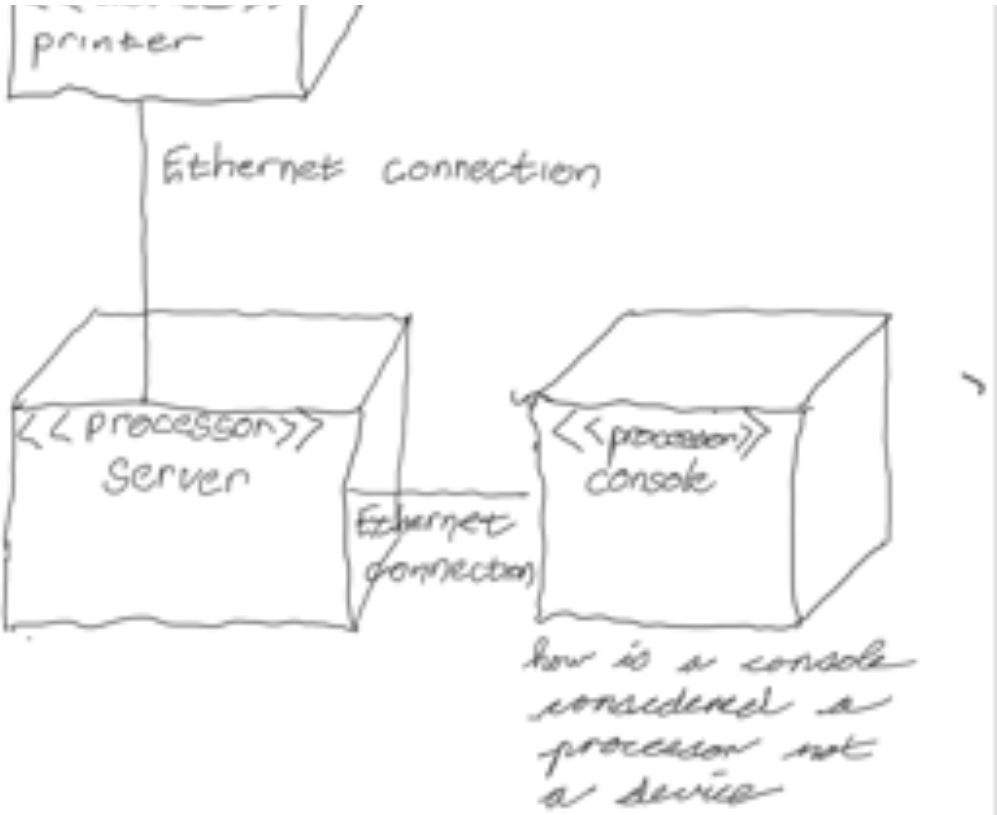
- Shows physical architecture of systems with devices
- Term used to describe computer resource is node
- Represented as a cube deployment dialogue
- 2 types of nodes
  - Processors
  - Devices
- processor thought of as brain, says that can execute components
- Type of code written in guillemots

2.8.3 Not entirely accurate remove lines (Just redraw this!

2.8.3.1 Diff between nodes and components

- Processor nodes exec components
- Components part of execution of sys
-



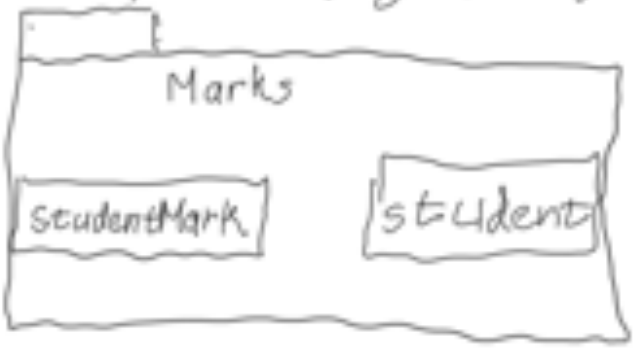


how is a console considered a processor not a device

Unit 9 - Packages, Stereotypes and patterns

Packages

- Elements of a sys may be organised into groups
- Represented as tabbed folders
- examples of groups (package)
  - Staff
  - all parts of an exam system
- Name is descriptive of group of content



2.9.2 Stereotypes

- New element retaining original characteristics of element it was created from
- Interface is represented with the word

interface between guillemets

- Stereotypes may be used as headings for lists

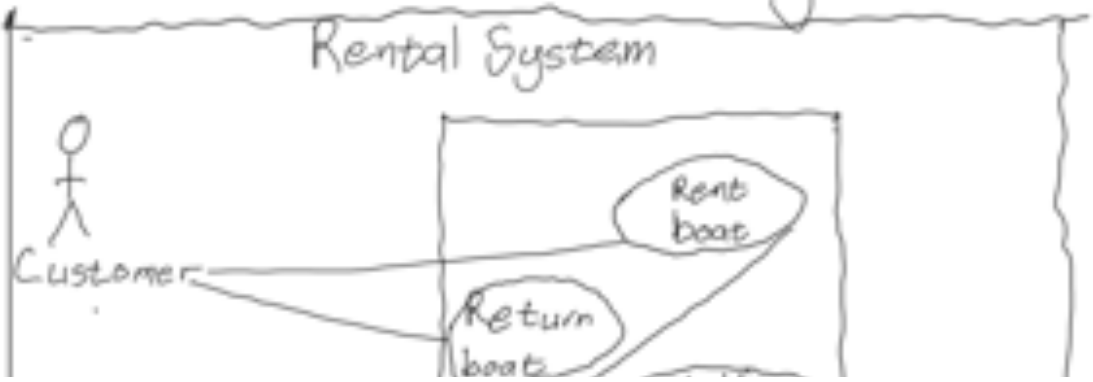
- Pictures can also be used as stereotypes eg. keyboard

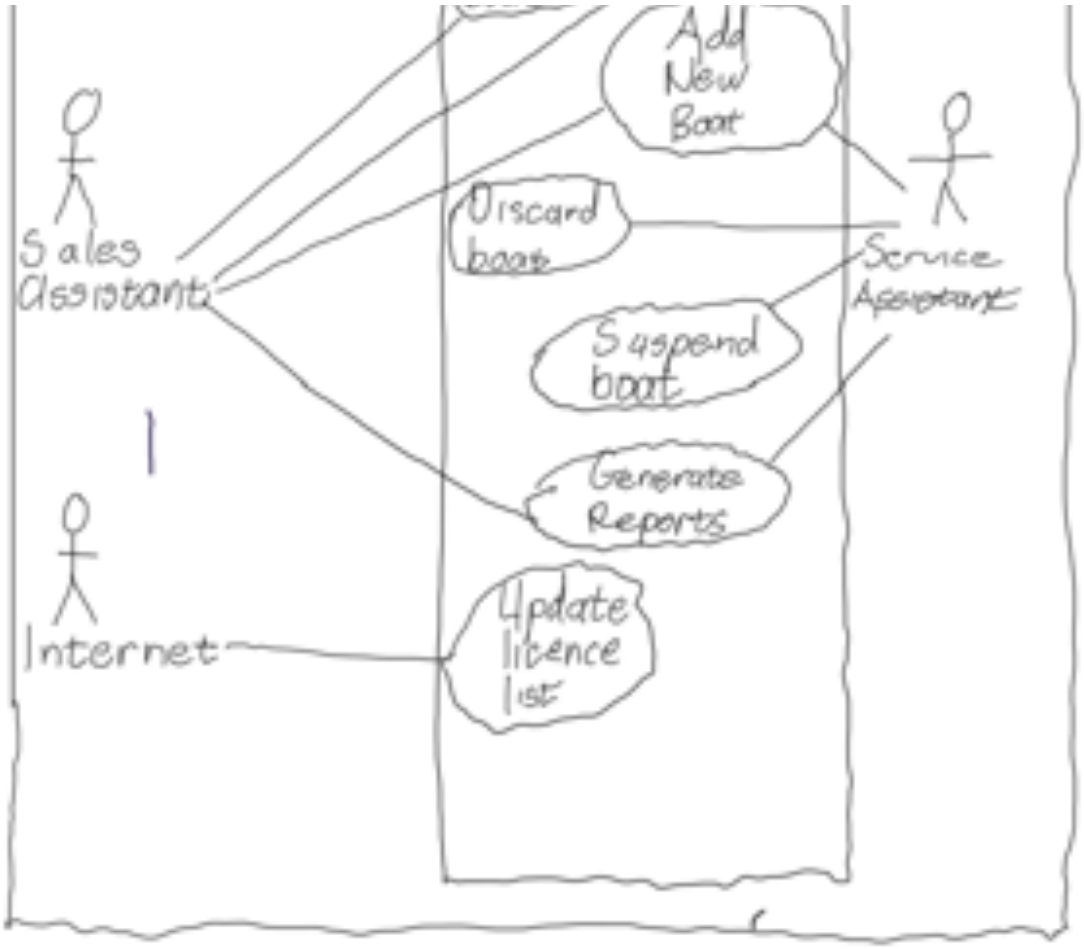
2.9.3 Patterns

Defined as common solutions to common problems

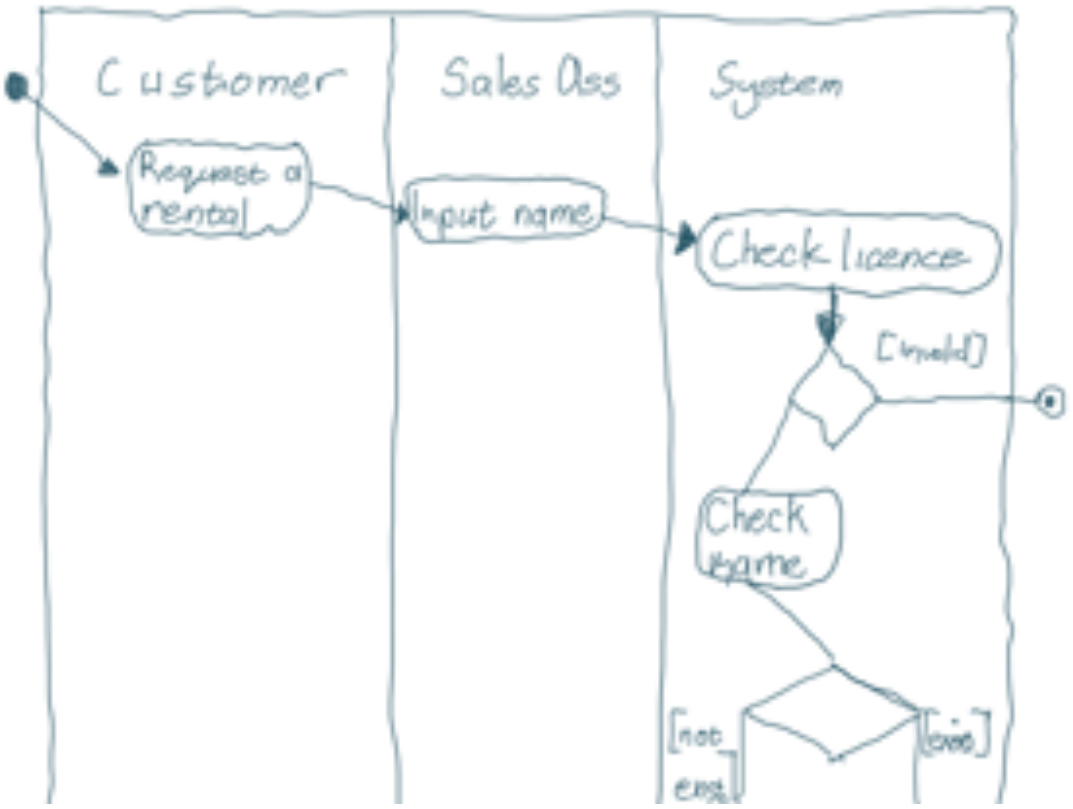
- May reuse (reusability)
- provide clear and concise guide/guide to relationship between classes or objects in sys
- More maintainable systems
- Increase productivity
  - Proven working model exists
    - customizable
      - don't start from scratch
- Easier to document

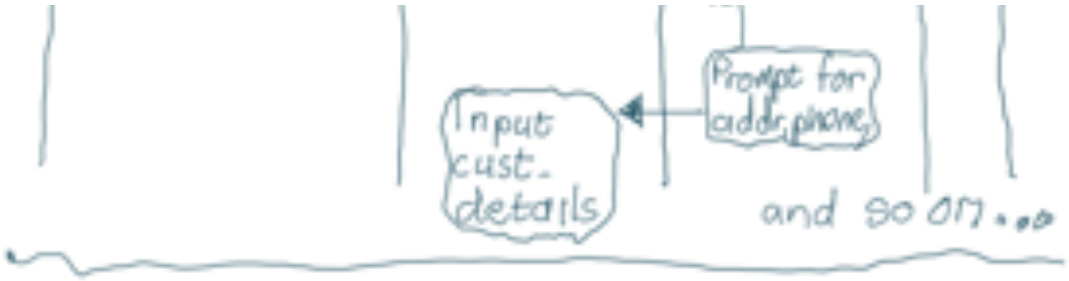
Usecase Diagram



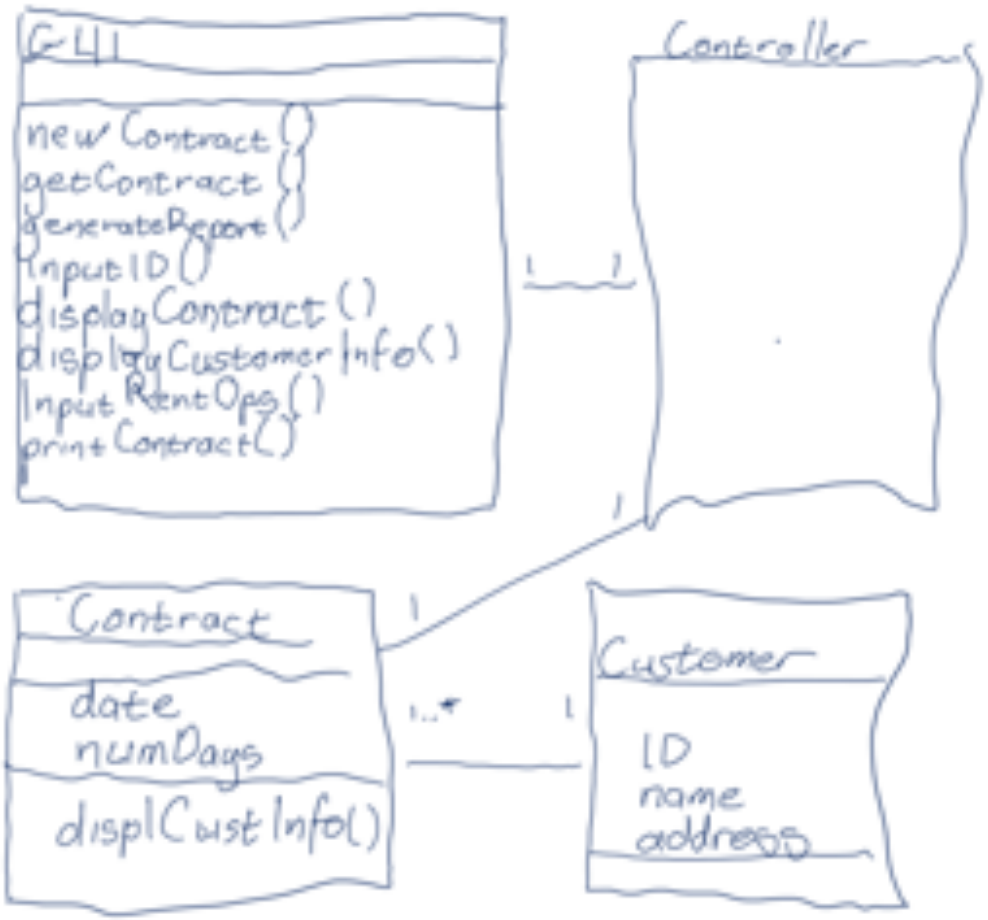


Activity Diagram





Class Diagram



Sequence Diagrams

Collaboration Diagram

State Diagram

- State diag for book object