waterfall (linear sequential model) ⇒ feasibility – requirements – design – implementation – testing

Types of SW life cycle activities

- Feasibility — market analysis
- Requirements — requirement elicitation — domain analysis
- Project planning — cost analysis — scheduling — software quality assurance — Work-breakdown Structure
- Design (Architectural — Interface — Detailed)
- Implementation
- Testing. (unit — integration — system — alpha — beta — acceptance — regression)
- Delivery — installation — training — help desk
- Maintenance

PROTOTYPING	prototype → linear seq model
INCREMENTAL	minimal subset
BOEHM'S SPIRAL	communication. plan. risk. engineering. construction, release, evaluation

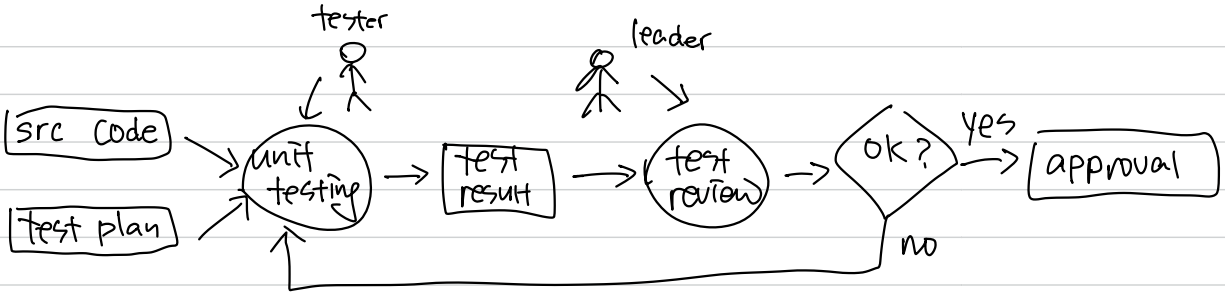
SW = (computer program + associated docs.) ← generic / bespoke

good SW = maintainability
dependability & security
efficiency
acceptability

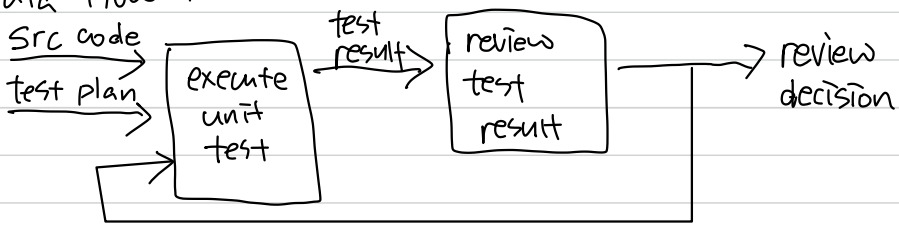
SE = [SW production ~ maintain system] concerns aspects.

agile =
- customer involvement
- incremental delivery
- people not process
- embrace change
- maintain simplicity

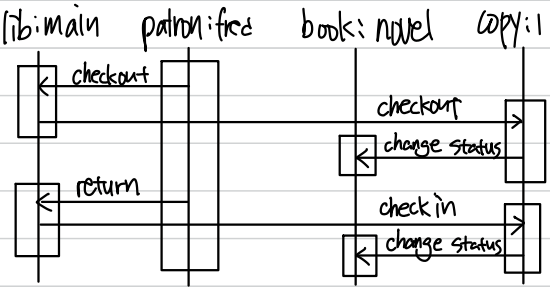
process model : 시작 노드 → 모든 노드 → 종료 노드.



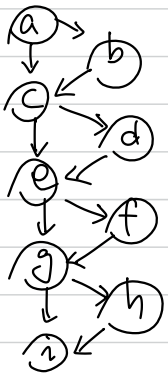
Data flow :



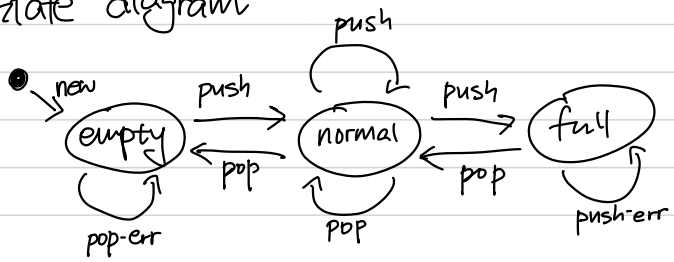
Sequence diagram



control flow diagram (if 분기)



State diagram



SW life cycle vs. process model : 주요 phase, deliverables vs. 각 phase 내 세부 task, artifact, actor

descriptive vs. prescriptive : 무엇이 일어났는가 (의사결정 x) vs. 어떻게 해야 하는지 지침. (의사결정 O)

artifact : 한 작업의 결과가 다른 작업의 입력. 상호의존 X ⇒ 독립 프로세스.

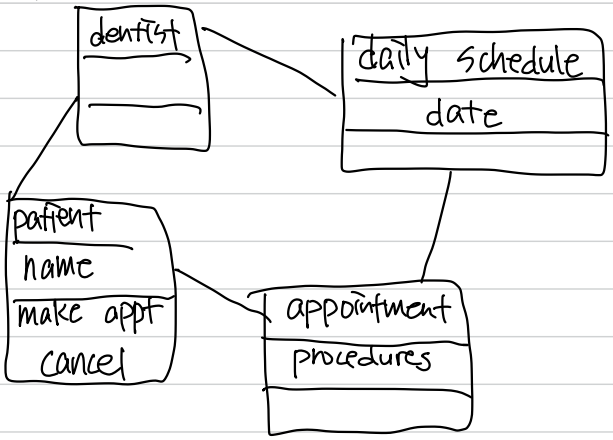
입력 준비 X ⇒ 시작X

Inheritance & Aggregation & general association

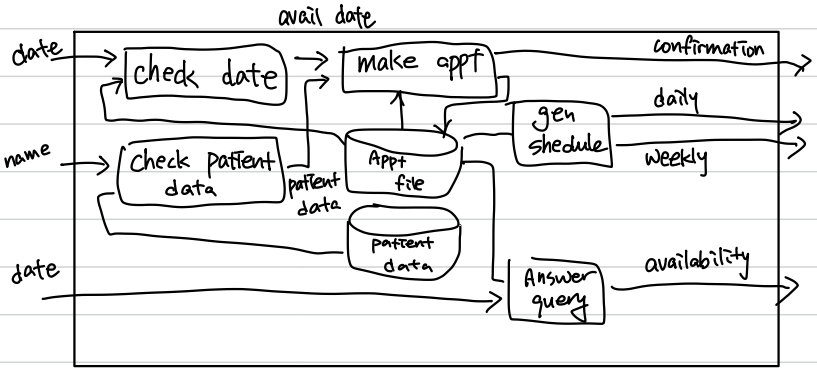
1 * generalization

aggregation

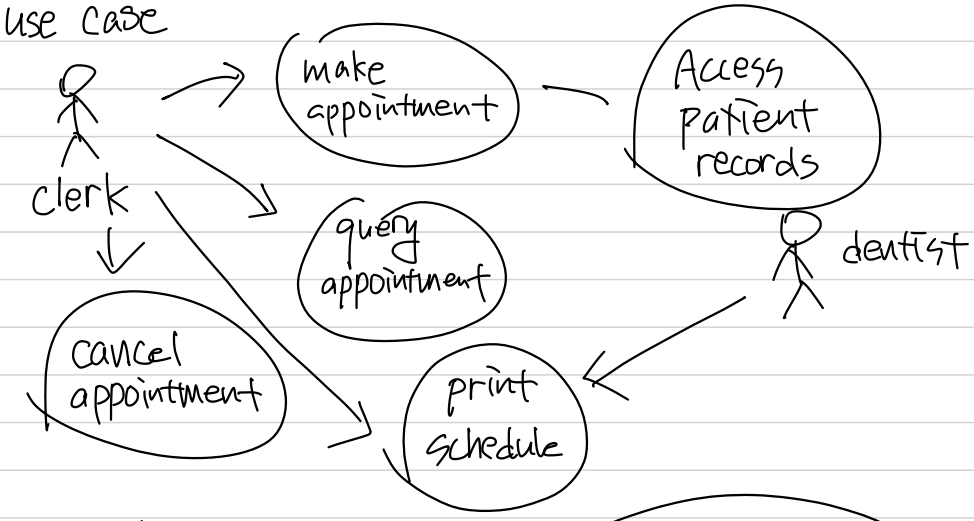
object model



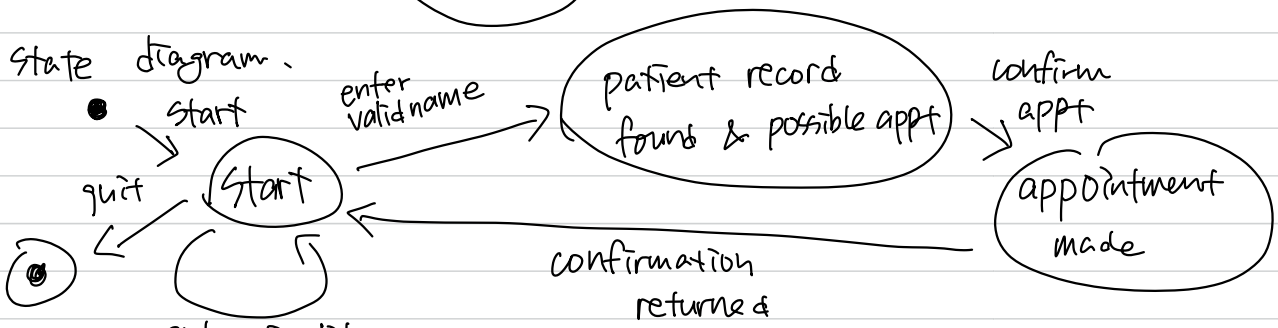
data flow



use case

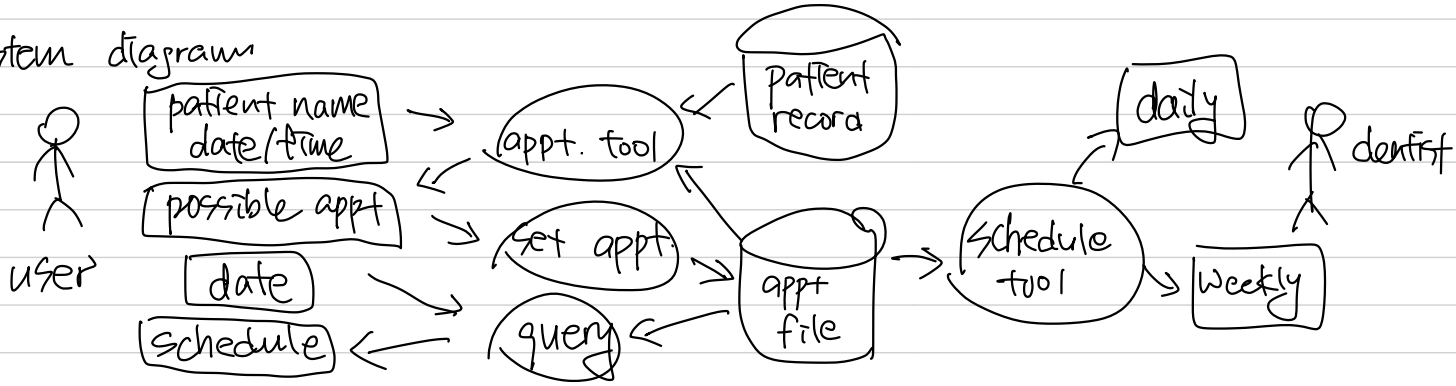


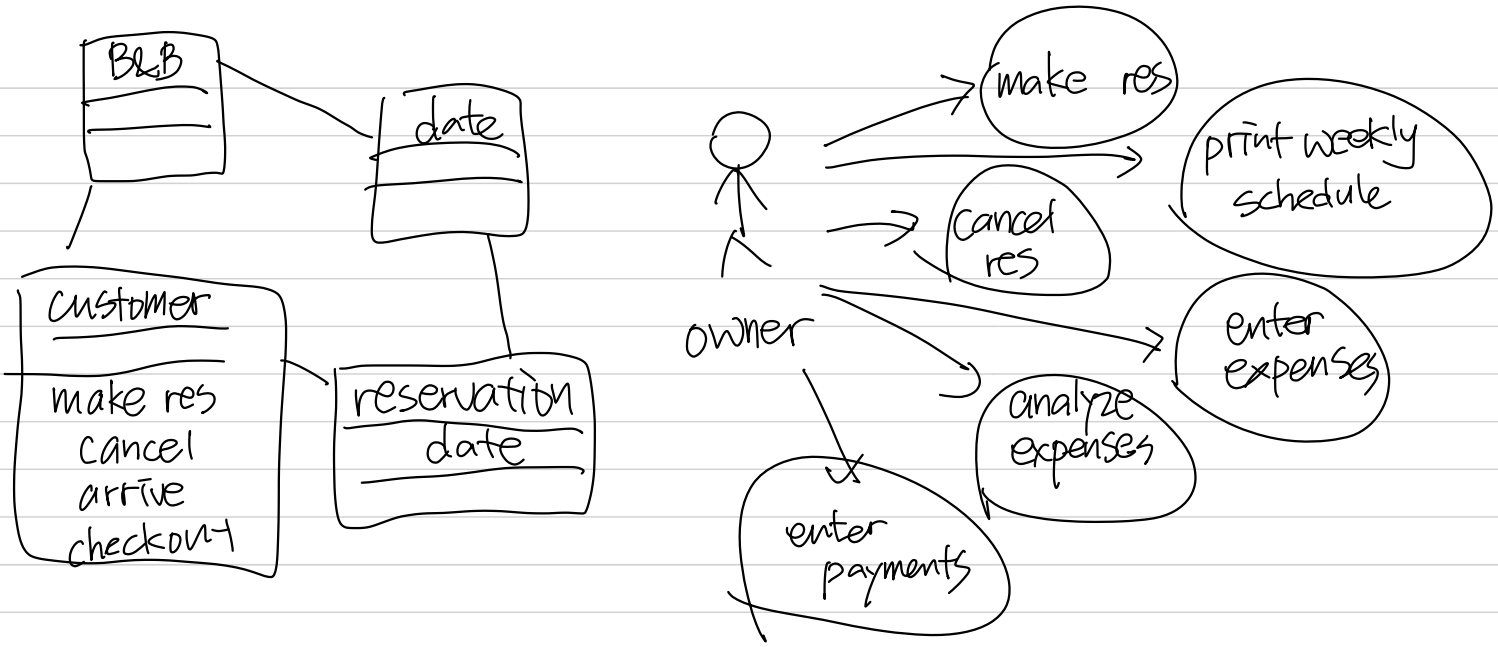
State diagram.



enter invalid name
query date, name
print schedule
delete appt.

System diagram





FR : services system should provide. ⇒ what should be done / or not . functionality .
SW Req Spec ⇒ specific req : all function description

NFR : system properties. constraints . entire system.
product / organizational / external

Domain Req

Req Engineering : Req elicitation, Req analysis , Req validation, Req management

Req elicitation & analysis : discovery classification, organization prioritisation, negotiation, specification

Req validation : demonstrating req define system customer wants.
⇒ review, prototyping, test case gen

validity / consistency / completeness / realism / verifiability

Req management : changing req during RE process.

Cohesion : measure of strength of association of elements within a module (high)

coincidental : no meaningful relationship

logical : elements similar activities chosen by outside. logical, but no primary logical

temporal : function related to time. series of action related to time (initialize or shutdown)

procedural : elem involved in diff activities, but sequential. ordering steps. related to ordering

communicational : elem perform diff functions. refer same input/output. actions are related, not separated

sequential : output is input of other function. elem of module form sequence.

functional : all elem contribute to single, well-defined task. module for single action, goal.

Coupling : measure of interdependence (low)

Data : communicate through a parameter (elementary data type)

Stamp : communicate using composite data

control : data in one module ⇒ direct order of instruction execution in another

common : share data through global data

content : share code

Abstraction : unnecessary details are removed. focus on essential issues.

Data - Architectural - Interface - procedural

refinement | successively refining levels of detail top-down

modularity | divide SW into module. module integrate ⇒ achieve req

dynamic testing ⇒ test data / static testing ⇒ analyze source code

verification: conform specification / validation: satisfy user requirement

Inspection: static verification / Testing: dynamic verification

Development Testing

unit	individual functions/class
component	component (several interacting objects) interface
system	integrating components interaction in system.

User testing: alpha beta acceptance

domain

Scalene: $a<b<c$, $a>b>c$, $a<b>c$

Isosceles: $a=b>c$, $a=c>b$, $b=c>a$, $a=b<c$, $a=c<b$, $b=c<a$

Equilateral: $a=b=c$

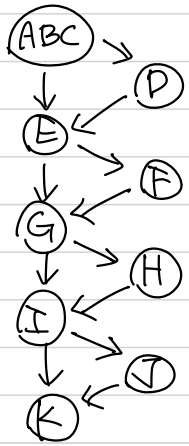
Not a triangle: ($b+c \leq a$) a is largest — largest at 1st, largest at 2nd, largest at 3rd

Bat input:
- 1 bad
- 2 bad
- all bad

conditions	1	2	3	4	5	6	7	8
$a=b$ / $a=c$ / $b=c$	T	T	T	T	T	F	F	F
$a=b=c$	T	T	F	F	F	F	F	F
$a \leqslant b+c$	T	F	T	T	F	T	T	F
$a,b,c \geqslant 0$	T	F	T	F	F	T	F	F

Node CO: every statement coverage

A	read a,b,c
B	type="scalene"
C	if (a=b || b=c || c=a)
D	type="isosceles"
E	if (a=b && b=c)
F	type="equilateral"
G	if (a>=b+c || b>=a+c || c>=a+b)
H	type="not a triangle"
I	if (a<=0 || b<=0 || c<=0)
J	type="bad inputs"
K	print type



C1: every branch testing

ABC-D	G->H
ABC-E	G-I
D-E	I-J
E-F	I-K
E-G	J-K
F-G	

Every path testing.

FFFF	ABCEGIK	scalene
FFTF	ABCEGHIK	not
FFTT	ABCEGHIJK	bad
TFFF	ABCDEGIK	isosceles
TFTF	ABCDEGHIK	not
TFTT	ABCDEGHIJK	bad
TTFF	ABCDEFGIK	equilateral
TTTT	ABCDEFGHIJK	bad

Multi-condition coverage

if (a==b || b==c || a==c)

TXX	ABC-D
FTX	ABC-D
FFT	ABC-D
FFF	ABC-E

if (a==b && b==c)

TT	E-F
TF	E-G
FX	E-G

if (a ≥ b+c || b ≥ a+c || c ≥ a+b)

TXX	G-H
FTX	G-H
FFT	G-H
FFF	G-I

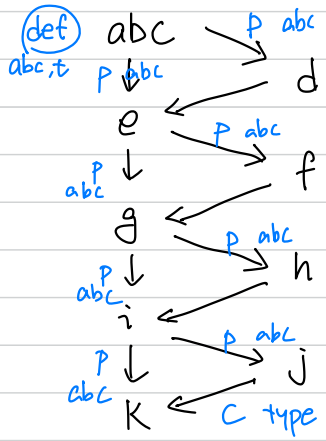
if (a≤0 || b≤0 || c≤0)

TXX	I-J
FTX	I-J
FFT	I-J
FFF	I-K

Sub domain testing.

equilateral	3.3.3
isosceles -1	8.9.9
isosceles -2	9.8.9
isosceles-3	9.9.8
scalene -1	9.4.3
scalene-2	4.9.3
scalene-3	3.4.9
not -1	8.3.4
not -2	3.8.4
not-3	3.4.8
bad -1	0.3.4
bad-2	3.0.4
bad -3	3.4.0

control flow



Def-C use (dcu)

abc	abc-e-g-i-k	(scalene)
d	d-e-g-i-k	(isosceles)
f	f-g-i-k	(equilateral)
h	h-i-k	(not)
j	j-k	(bad)

Def-p use

abc	abc-d
	abc-e
	e-f
	e-g
	g-h
	g-i
	i-j
	i-k

def free ⇒ path without redefinition