D1

Systems Development Life Cycle (SDLC)

$$\begin{tikzcd} & \text{Planing} \arrow[dr] & \\ \text{Implementation} \arrow[ur] & & \text{Analysis} \arrow[dl] \\ & \text{Design} \arrow[ul] & \end{tikzcd}$$

- design within budget
- deliver on time
- find error easier. prevent it happen again

Planing (why / how)

{ creative ⇒ plan anything
fix
replace } → better one

- monitor "time" & "task" in the same time
- assign the person to the right place

Analysis

Analysis < technique / financial　　who / what / when

產出 documentation (feasible、deliverable)

Design

what programs to write
what each program will do

Implementation

{ Construct (Build、test)
Install
Support }

SDLC: Methodologies

一套開發流程
combine process with data

Structured

{ Waterfall
Parallel }

- static
- need clearly describe request
- hard to go back

Rapid Application

{ Phased
Prototyping
Throwaway Prototyping }

- interation
- Incremented } Double "I"

Agile

{ XP
SCRUM }

- short time

Object-Oriented System Analysis & Design

{ Unified Modeling Language
Unified Process }

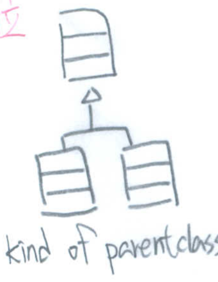
* Use-case driven
↳ 以function為主
* Architecture Centric
* Iterative & Incremental
combine phase what work for and data with process

O-O system 特色

{ Class & Object 最小運行單位
Methods & Messages }

- redefined same attributes
- don't reveal ex: password

combine class with class



{ Encapsulation & information hiding
Inheritance }

* 通用屬性往上提，特別屬性往下放

-Polymorphism & dynamic binding
* become unique instance

Architecture centric 依描述對象不同

{ Function view (external)
Structural view (static): 類別、物件圖
Behavioral view (Dynamic): 通訊、循序圖 }
↳ 如何交互，誰呼叫誰 instance/object relationshi

O-O 將系統分割 (Vertical / Horizontal

Break and conquer until finding simple one, throw a rough one slim、stupid、lean

Unified Process : keep it simple, stupid

二維: phases & work flows

* 選一個 Unified 變成 Unique 然後 use it
compose many pieces, put it into project
Focus one business modeling & requirements gathering

Inception → Elaboration → Construction → Transition

UML: 工程師 & 工程師溝通的語言

* 文字表達不清楚，含義模糊

	Structured		RAD			Agile	
	Waterfall	Parallel	Phased	Prototyping	Throwaway Prototyping	XP	scrum
Unclear 需求	P	P	G	E	E	E	E
Unfamiliar 技術	P	P	G	P	E	G	G
Complex	G	G	G	P	E	G	G
Reliable	G	G	G	P	E	E	E
Short time	P	G	E	E	G	E	E
Schedule visibility	P	P	E	E	G	E	E

中小型是 good
大型是 poor

Software Process

*強大的 test

divide and link

divide into small pieces to manage.

* documentation

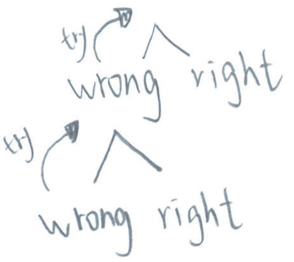
↳ 更了解 code

Tame Problem

- specification
- solution
- stopping rules
- correct / wrong
- 有 definite 因果

* 一定有答案

testing until best



ex: math

Chess playing

Wicked Problem

- specification
- solution

分不開

- always can do it better

Subjective

↳ just don't like it not it is wrong

* no right to be wrong

ex: 軟体開發

經濟改革、政策

* 水能載舟 亦能覆舟

角度不同，分類不同

✡ learning define what & need

say no but not offend the user

Software Paradigm

△ Procedural paradigm

↳ refind by lower process

✡ process ex: DFD

△ O-O Paradigm

↳ 相互關聯的 Objects

✡ Object

△ Data - oriented ex: DRD

✡ data entities

✡ relationship

Agile Methods

△ Dynamic System Development Method (DSDM)

↳ framework work with XP & Rational Unified Process

↳ 80-20 principle

↳ agile / plan-driven project

* all things are important

時間有限，最重要先做

△ Feature Driven Development (FDD)

↳ feature / model driven

↳ 配置管理、

review and inspection、

regular builds

↳ agile / plan-driven project

✡ 5個階段

△ Scrum (一堆 meeting)

↳ Scrum Master

Product owner

Team

↳ 15 mins meeting / 日

↳ team retrospect

△ Scrum (cont.)

Release planing meeting

- product backlog : 確定、優先考慮需求
- sprint : 確定這次 increment 要交付的
- 確認 sprint 的 activites

Sprint iteration

- sprint planning meeting
  - ↳ what & how to build next
- daily scrum meeting
  - ↳ exchange status

Sprint review meeting

- increment demo
- team retrospection

Deployment

* Sprint 2~4 週

△ Extreme Programming (XP)

· Anyone can change any code anywhere at any time.

· Integration & build many times a day whenever a task is completed.

· 工作時數 < 40hr 一週

Exploration : 資訊蒐集、可行性評估

Planning : 下個版本的 stories、plan

Iteration: 架構、implement、test

productionizing : 評估性能、test

Maintenance : 改善當前版本 (維護)

Death : documentation

DFD 注重 process 拆解 process

not a code but helpful
easy to find for and
design it correct.

* Leveling . Balancing
功能. data flow

What the systen does,
not how it does it.

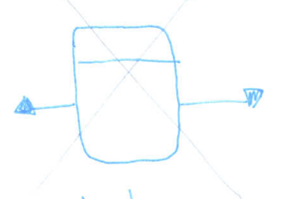
* graphical . logical

process → V.

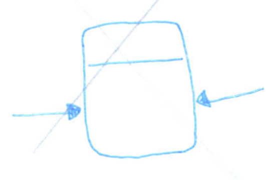
Data store → 複數 n

External entity → 單數 n.
↳ outside, not part of system
(人、非人,其他系統、時間)

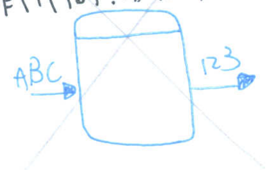
Spontaneous generation



Black holes



Gray holes
*答非所問.最難找



Wrong

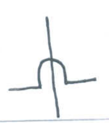
problem domain
系統開發領域
ex: 選課系統(教育)
↳ student, teacher

* graphical technique
let other understand easier
is useful information

* base on fact-finding results

Context diagram 內有 process 0
系統關聯(環境)圖
就是系統本身,非子系統
與 daigram 0 不一定要平衡
一直往下拆解 process
直到 Functional primitive
即開始撰寫 code
無需其他解釋即可理解

DFD is not a what
show what, how
dependent on system



data flow
交叉用跳線

Data Dictionary
{ data store、flow 的內容
diagram element (name、id...)
及所有用到的名詞.其頻率

ensures data consistency
提供組成 system 的全面資訊
all data be sorted
easier be used

Process Description < table / tree
幫助理解內容
確認是否考慮所有情況
{ Sequence
selection
Iteration (Looping)

ex: 3个條件.每个 Yes/No
会有 2^3 个結果(排列組合)
same results can be combined
each condition is unique
簡化規則,減少 code 判斷式

table ⇒ tree 邏輯要一樣
graphical ↳ 橫向圖

Pseudocode 偽 code DFD
⇒ structure english
(演算法)
注意縮排
用簡單的英文即可

PLLP (four-model approach)
1. Physical current
2. Logical current
3. Logical new
4. Physical new

but added time and cost

D2

Project Management
- △ business needs
- △ feasibility
- △ request
- △ select project
  - ↳ different purpose
    different choose
- △ breakdown 架構
- △ 甘特圖
- △ network diagrams
- △ use-case
- △ 互动 workplan

✱ planing & controlling
正確、時間、預算
task < monitor
control
人 - 協調

business needs → project
{ 顧客面
技術面

Business Value
{ Tangible：有形
(可量化)
Intangible：無形
(不可量化)

✱想辦法將無形
轉成有形價值
Show to boss.

System Request (文件)
- △ project sponsor
- △ business need
  - ↳ reason
- △ business requirement
  - ↳ what system will do
- △ business value
  - ↳ organization benefit
- △ special issues

Feasibility
- △ 技術
- △ 經濟：NPV = PV Benefits - PV Costs (Cumulative NPV)
  ROI = Total benefits - Total costs / Total Costs
  Break-even point = 第一个由負轉正的 (yearly NPV - Cumulative NPV) / yealy NPV
- △ 組織 (stakeholder) ↳ 再加上已过年數才会是答案

Selection
Value added vs. risk
Maximize cost / benefit ratio
✱ 資源有限

Project Management Tools
- △ Work breakdown structures (WBS)
  - ↳ 時間、狀態、前置作業
    甘特圖
- △ Network diagrams
  - ↳ PERT、CPM

Project Effort Estimation
✱用於安排 time & effort
{ Technical complexity factors
Environmental factors

UAW
UUCW
UUCP = UAW + UUCW
TFC = 0.6 + (0.01 × TFactor)
EF = 1.4 + (-0.03 × EFactor)
UCP = UUCP × TFC × EF
Effort in person-hours
= UCP × $\boxed{PHM}$

(E1~E6 < 3) + (E7~E8 > 3)
個數 ≤ 2 , PHM = 20
= 3 or 4 , PHM = 28
(風險太高) else , Rethink project

Staffing the Project
決定需求人數
降低 staff 衝突

Creating & Managing the workplan
- △ Workplan → dynamic / sequential list of all tasks
- △ Approaches
  - ↳ 修改 / 完成 project
    從用到的 Methodology 推導 tasks
- △ Unified Process
  - ↳ Iterative & incremental
    Tasks / 時間間格 follow the Phases
    每个工作流執行不同任務

Scope Management (Scope "creep")
因 requirements 增加導致 (有害影響)
只允許絕對必要的變化
✱ Time box 時間固定，(捨去部份功能)
D $\boxed{A\ \cancel{B}\ C}$ 想增加要先減少

Jelled Team
強大精英團隊，注重專案目標
凝聚力高、氣氛好
members enjoy their work

Staff Plan
人數 = $\frac{\text{person-months}}{\text{time to complete}}$ (沒有考慮個人能力)
✱ 人多溝通難，不一定好
✱ technical & interpersonal skills

Motivating People
- △ 20% time rule → 20%時間放手去做
- △ 金錢以外的 P2P 表彰
- △ 允許 focus on interests

Management
Environment：生產力集中、圖示、建立標準
更容易懂
Infrastructure：存 deliverables、communication
use Unified Process standard document
別把 documentation 放最後

Evolutionary Work Breakdown Structures
✱ standard manner、incremental & iterative
✱ Unified Process