系統分析與設計完全掌握：從零到精通的架構師藍圖 (Mastering Systems Analysis & Design)

費曼學習法 x 考試重點拆解 (Feynman Method x Exam Blueprint)

同學們好，我是帶領這門課的資深教授。今天我們不背死書，我們要用「費曼學習法」——也就是用最簡單的人話，把系統開發的邏輯徹底搞懂。

這份藍圖無縫融入了 Past Paper 的實戰應用，確保你考場無往不利。

系統開發生命週期 (SDLC) 與分析階段的本質

1. Planning
2. Analysis
3. Design
4. Implementation

Feynman Metaphor

蓋房子前必須先畫圖紙。SDLC 就是蓋軟體的標準流程。我們今天的重點在於「分析 (Analysis)」——也就是搞清楚客戶到底要什麼 (What)，以及「設計 (Design)」——決定我們要怎麼做出來 (How)。

Key Exam Points (必背重點)

- **The Analysis Phase (分析階段):** Breaking a whole into its parts to understand nature and functions.
- **Three Steps in Analysis (分析三步驟):**
  1. Understand the existing situation (As-is system).
  2. Identify improvements.
  3. Define requirements for the new system (To-be system).
- **Final Deliverable (產出物):** System Proposal (系統提案書).

找出使用者的真實心聲：五大需求擷取技巧 (Requirements Elicitation)

客戶通常不知道自己要什麼，或者不會表達。我們必須用不同的「偵訊技巧」來找出真相。

1. Interviews (訪談)

One-on-one conversation.
(深入了解個人想法，資訊深度深).

2. JAD (Joint Application Design - 聯合應用開發)

Brainstorming sessions with users and IT staff.
(大家坐下來一起開會討論，整合性極高).

3. Questionnaires (問卷調查)

Surveying a large group.
(發問卷，適合收集廣泛但較淺的資訊).

4. Document Analysis (文件分析)

Reviewing existing reports/forms.
(看現有的舊報表，了解 As-is 系統).

5. Observation (觀察法)

Watching users do their jobs.
(直接看他們怎麼工作，因為他們說的跟做的不一定一樣).

如何選擇最適合的擷取技巧？(Selection Criteria Matrix)

Professor's Note:
考試常考情境題：「請根據情境選擇合適的技巧」。這張表是你拿分的武器！

Comparison of Requirements-Gathering Techniques (Requirements-Gathering Techniques Matrix)

FIGURE 4-13

	Interviews	Joint Application Design	Questionnaires	Document Analysis	Observation
Type of information	As-is, improvements, to be	As-is, improvements, to be	As-is, improvements	As-is	As-is
Depth of information	High	High	Medium	Low	Low
Breadth of information	Low	Medium	High	High	Low
Integration of information	Low	High	Low	Low	Low
User involvement	Medium	High	Low	Low	Low
Cost	Medium	Low–Medium	Low	Low	Low–Medium

Exam Cheat Sheet (考試速記)

- **Need Depth of Info (深度)?**
  -> Choose Interviews or JAD.
- **Need Breadth of Info (廣度)?**
  -> Choose Questionnaires or Document Analysis.
- **Need Integration of Info (資訊整合)?**
  -> Choose JAD (Only JAD is High!).
- **Low Budget (Cost)?**
  -> Choose Questionnaires or Document Analysis.

系統獲取策略 (System Acquisition): Build, Buy, or Outsource?

需求確立後，我們要怎麼生出這個系統? 就像買衣服 : 你可以買布親自裁縫 (Custom)、直接買成衣 (Packaged)，還是花錢請裁縫師幫你做 (Outsourcing)。

1. Custom Development (客製化開發)

Develop entirely in-house. (完全自己寫程式，彈性最大，但最花時間).

2. Packaged Software (套裝軟體)

Buy pre-written software and customize it. (直接買現成的，例如 ERP 系統，快速且經過測試).

3. Outsourcing (外包)

Rely on an external vendor/service provider. (花錢請外面的專業公司做).

System Integration (系統整合): Combining packaged software, legacy systems, and new software. (把新舊系統串接在一起的最大挑戰).

決策矩陣：何時該用哪種獲取策略？(Decision Framework)

Professor's Note:
這是選擇題和申論題的常客。判斷基準在於：業務需求是否獨特？公司有沒有技術能力？

Selecting a System Acquisition Strategy

	When to Use Custom Development	When to Use a Packaged System	When to Use Outsourcing
Business need	Unique (獨特)	Packaged system	Outsourcing
In-house experience	Common (普遍)	Packaged	[illegible]
Project skills	High skills	Packaged	None
Project management	Flexible	Short (短)	Short
Time frame	Time frame	Short/Flexible	Outsourcing.

WARNING: LOSS OF CONTROL

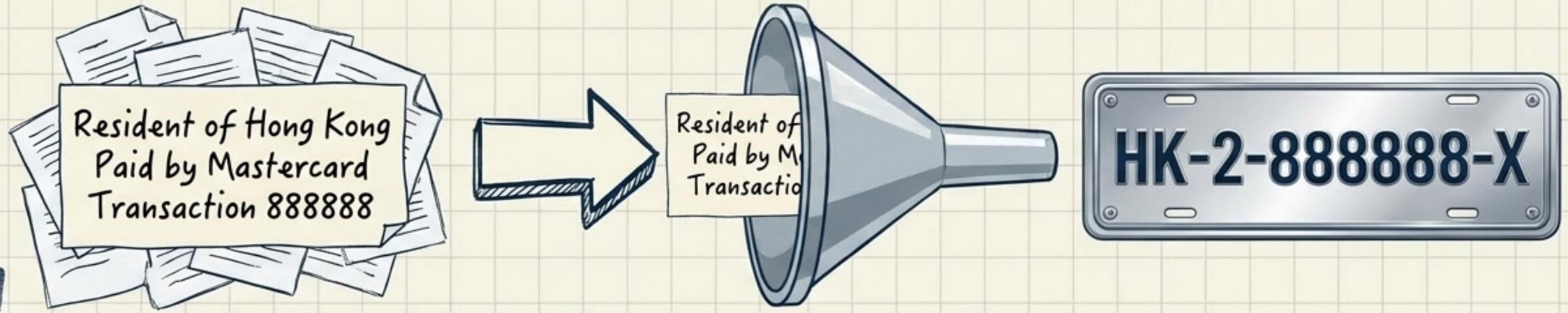
Decision Triggers (決策關鍵字):

- **Business Need:** Unique (獨特) -> Custom. Common (普遍) -> Packaged. Not core (非核心) -> Outsourcing.
- **In-house Exp:** High -> Custom. Exists -> Packaged. None -> Outsourcing.
- **Time Frame:** Flexible (彈性) -> Custom. Short (短) -> Packaged. Short/Flexible -> Outsourcing.

Exam Warning (紅色警戒):
Never outsource what you do not understand! (絕不外包你不懂的東西，否則會失去核心控制權).

賦予資料靈魂：資料代碼設計 (Data Code Design)

想像一下，如果身分證字號不是代碼，而是寫著「住台北的陳小明」，電腦處理起來會有多慢！Data Code（資料代碼）就是把資訊壓縮成一組有意義的數字或字母，用來唯一辨識 (Unique representation) 資料。



Advantages / 優點

- Save storage space (節省儲存空間).
- Easy for record retrieval (易於檢索).
- Help to arrange data in a special order (方便排序).

Design Criteria / 設計原則必背

- ☑ Easy to assemble (容易組合).
- ☑ Easy to remember (容易記憶).
- ☑ Include Check Digit (加入檢查碼) to avoid input error.

實戰演練：改良糟糕的代碼設計 (Past Paper Case Analysis)

Professor's Note: 考試常要求你指出舊代碼的問題，並設計新的代碼。我們來看一個捐款系統的例子。

Flawed Design (原本設計: CCC-P-NNNNNN)

- **Rule**: CCC (Country: 001=HK), P (Payment: 1=Visa, 2=Master).
- **Problem**: The length of CCC is fixed at 3 digits (e.g., 001). It is hard to remember and interpret. (數字無法直覺聯想國家).

Optimized Design (改良設計: CC-P-NNNNNN-C)

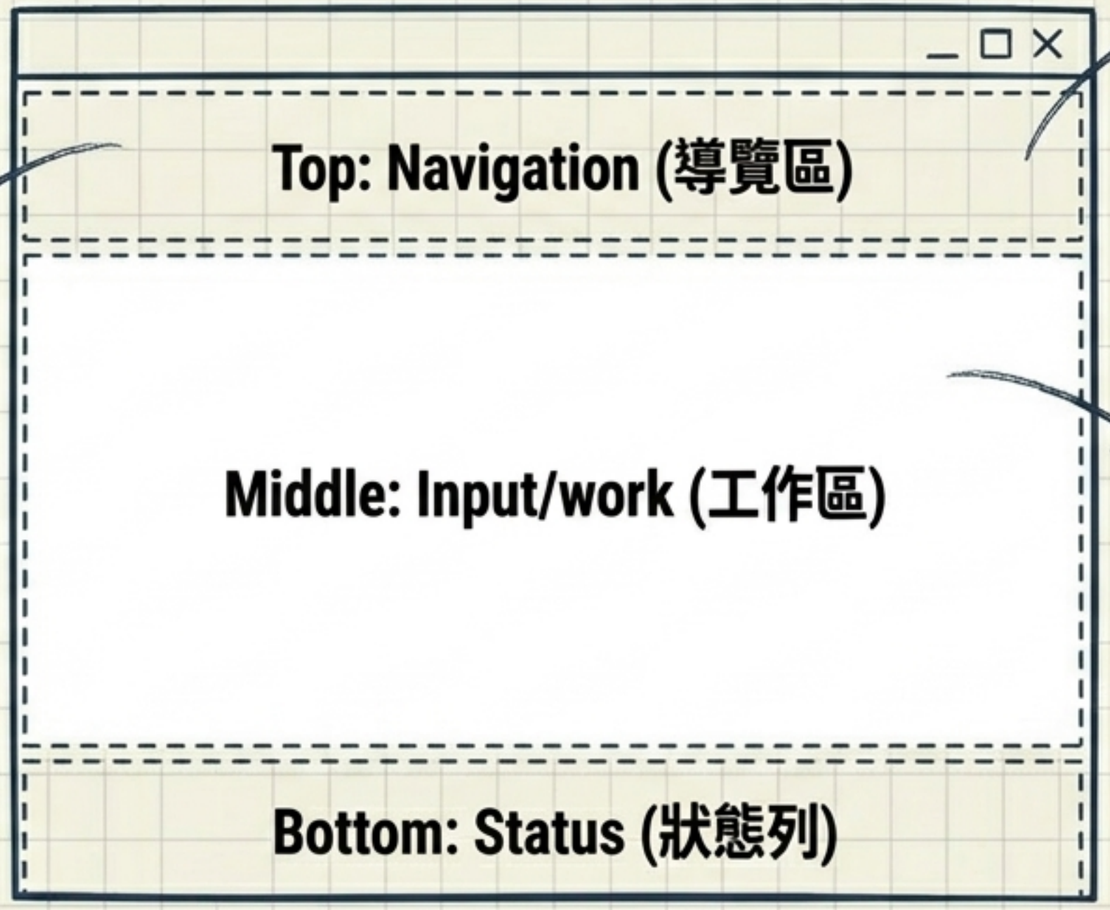
- **CC**: Country Code using Letters (e.g., HK = Hong Kong). Better human interpretation.
- **P**: Payment Method (1=Visa, 2=Master).
- **NNNNNN**: Sequence Number.
- **C**: Check Digit (檢查碼 - 絕對不能忘！).

Result Example: HK-2-888888-X
(香港, Master, 第888888號, 檢查碼 X).

UI 使用者介面設計原則：The Canvas (Part 1)

UI 設計不是只要「漂亮」，而是要讓使用者「不需要思考」。

1. Layout (版面配置):
Organizing areas consistently.
Top for navigation, Middle for input/work, Bottom for status.



2. Content Awareness (內容認知):
Users should always know where they are.
(每個介面都要有 Titles，選單要有麵包屑導覽)。

3. Aesthetics (美學):
Functional and inviting.
Careful use of white space (留白), colors, and fonts.
(高密度的表單會讓人崩潰，必須留白)。

UI 使用者介面設計原則：The User (Part 2)

Professor's Note: 介面是要給「人」用的，所以我們必須考慮使用者的經驗與習慣。

Novice
(新手)

Needs ease of learning
(易學性).

5. Consistency (一致性):
Enable users to predict what will happen.
(名詞、按鈕位置在整個系統中必須統一).

Needs ease of use / shortcuts (易用性).

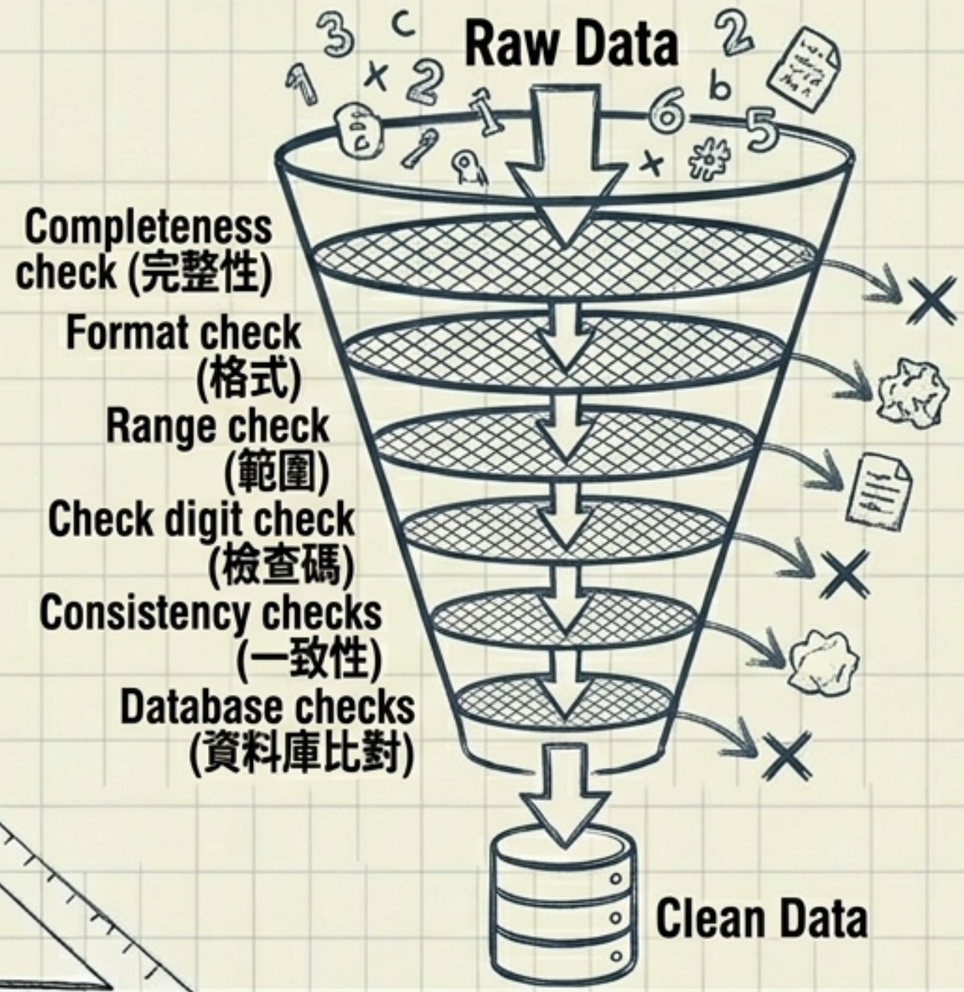
Expert
(專家)

Max 3 Clicks

6. Minimize User Effort (最小化負擔):
Three-clicks rule (三擊法則).
Users should perform work in **no more than 3 mouse clicks** from the main menu.
(把步驟減到最少).

輸入驗證 (Input Validation)：把關資料品質的第一道防線

Garbage in, garbage out. 如果在輸入資料時不馬上檢查，錯誤資料進到資料庫就會引發大災難。

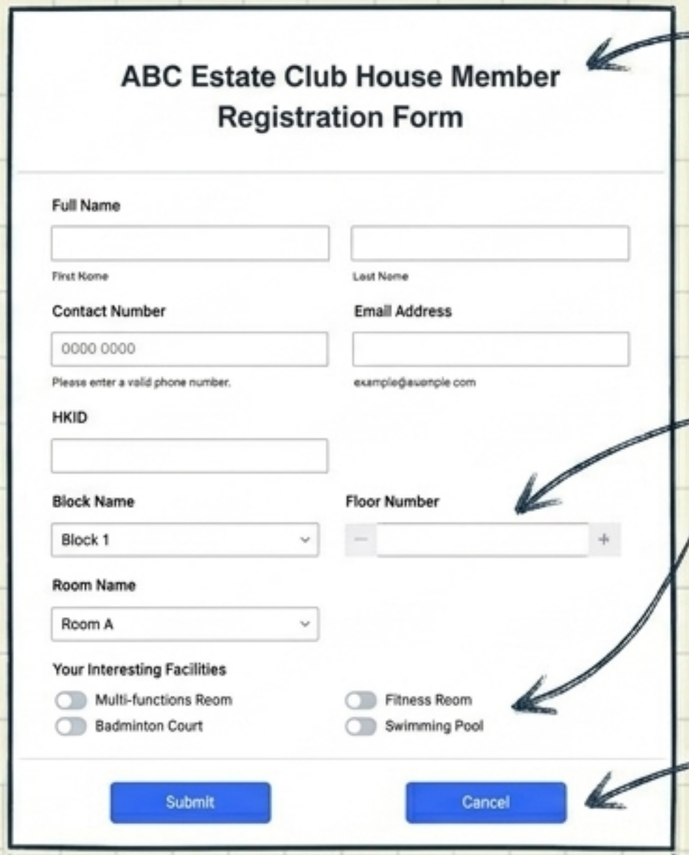


六大輸入驗證法 (必背)：

1. **Completeness check (完整性)**: Ensures all required data entered. (必填不可空白).
2. **Format check (格式)**: Ensures correct data type. (例如日期格式 YYYY/MM/DD).
3. **Range check (範圍)**: Ensures data are within correct min/max. (分數必須 0-100).
4. **Check digit check (檢查碼)**: Formula applied to numeric codes. (防打錯數字).
5. **Consistency checks (一致性)**: Ensure combinations are valid. (入學不能晚於畢業).
6. **Database checks (資料庫比對)**: Compare data against existing records. (帳號是否被註冊).

實戰演練：滿分 UI 表單設計拆解 (Past Paper Case)

Professor's Note: 考試常要求畫資料輸入介面。評分標準在於你是否具備完整的控制元件（Controls）與版面邏輯。



1. **Content Awareness**: Proper Title ('ABC Estate Club House Member Registration Form').

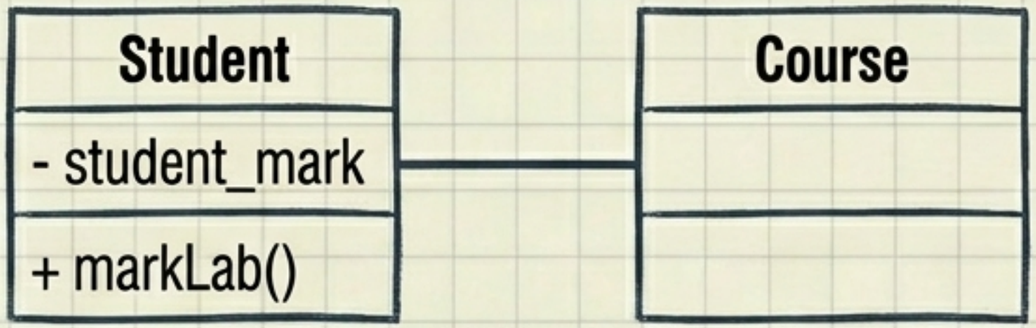
2. **Layout:** Logical Grouping (Personal details grouped together, distinct from facilities).

3. **Error Prevention (Controls):** Drop-down lists for Block/Room to minimize typing error. Toggle buttons for facilities.

4. **Consistency:** Clear, standard buttons at the bottom (Submit in bold blue / Cancel).

教授加碼增潤：物件導向的核心概念 (Object-Oriented Foundations)

Professor's Note: 雖然 PPT 未詳述，但 Past Paper 屢次出現 Class 的程式碼。作為架構師，你必須懂 OO 概念！



- **What is a Class? (類別定義)**
  A template or blueprint used to create objects. It defines **Attributes** (屬性/資料, e.g., student_mark) and **Methods** (方法/行為, e.g., markLab()).

- **Relationships (類別關聯 - 必懂):**
  1. **Inheritance (繼承)**: 'Is-a' relationship (e.g., CarParking extends Vehicle). Subclass inherits.
  2. **Association (關聯)**: 'Uses-a' relationship. Interact but have own lifecycle.
  3. **Aggregation (聚合)**: 'Has-a' relationship. Whole-part relationship (e.g., Department has Teachers).

融會貫通：系統開發大藍圖 (The Architect's Synthesis)

Professor's Note: 考試常要求求入介，產鑰質的問題，值後魡胍、、指制使用確成統產發的開發。|，交給工程師的開發。

Step 1: Elicitation (擷取)

訪談、JAD 找出使用者要什麼。

Step 2: Analysis (分析)

釐清問題，產出 System Proposal (提案書)。

Step 3: Acquisition (獲取策略)

決定 Build, Buy, 還是 Outsource。

Step 4: Design (設計)

設計底層 Data Codes (唯一且省空間) + 表層 UI (防呆設計與輸入驗證)。

Final Output (最終產出)

產出 System Specification (系統規格書)，交給工程師開開發！

考前終極衝刺指南 (Exam Survival Guide)

Professor's Final Words: 這門課考的是「邏輯」而不是死背。遇到情境題，請套用我們今天學的框架：

✅ **選擇/配對題陷阱 (Traps):**
- JAD 是唯一 "Integration" 高的擷取技巧。
- Outsourcing 的死穴是 "失去核心控制權" 。

✅ **設計題得分關鍵 (Design Points):**
- 畫 UI 必加 Title、按鈕，用下拉選單代替手打。
- 設計 Data Code 絕對不能忘記 Check Digit！

✅ **必備專有名詞 (Vocabulary):**
- SDLC, Requirements Elicitation, System Proposal, Input Validation (六種皆需能舉例)。

祝各位順利高分通關！Class Dismissed!