

CS 362 – Object-Oriented Design

Introduction to Software Design and Architecture

Kamonphop Srisopha

Kamonphop.s@cmu.ac.th



Faculty of Science, Chiang Mai University

คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่

Agenda:

- What is Software Design?
- The Measure of Design Quality
- Career in Software Design and Architecture
- Software Design in the Industry
- Reflection

What is Software Design?

A Common Misconception

**Software
Design** \neq **User
Interface
Design**

Actuality

Software Design

แต่เราจะไม่พูดถึงกระบวนการทั้ง User Interface Design ในวิชานี้

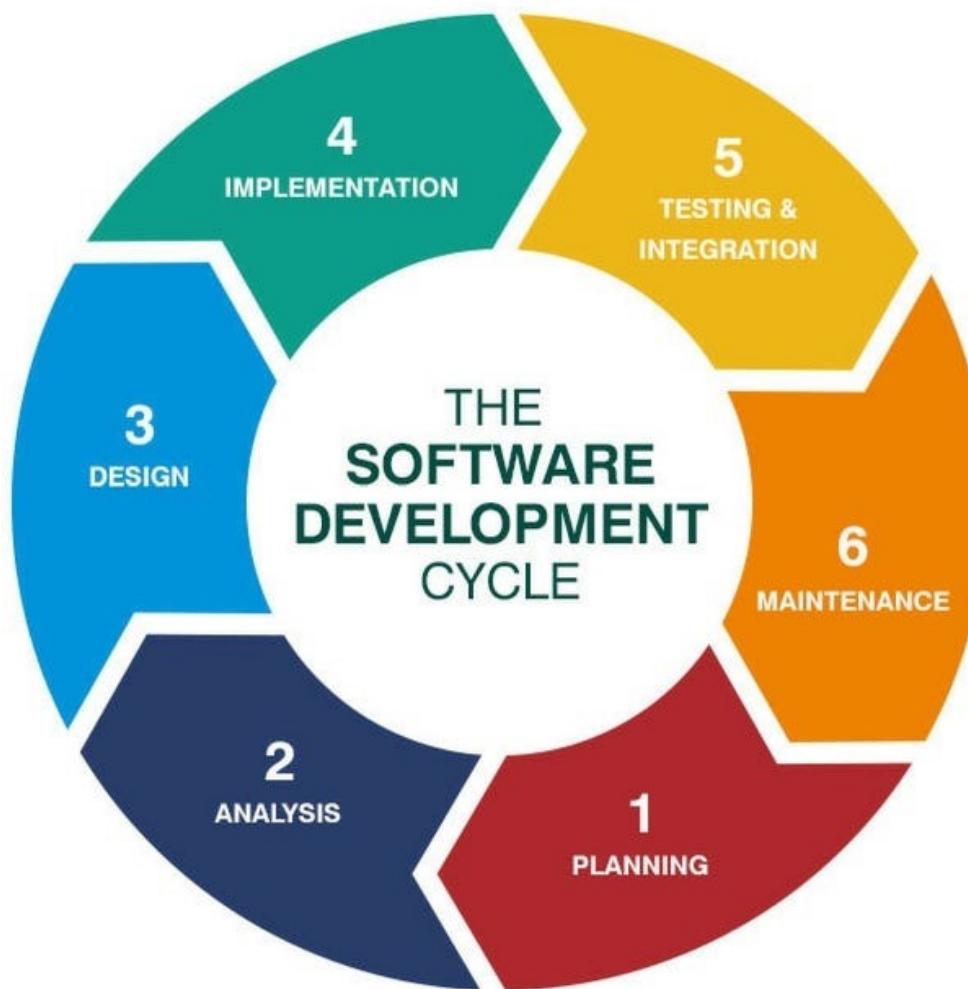
User Interface Design

What is Software Design?

- **การออกแบบซอฟต์แวร์** คือ

- การนำความต้องการของผู้ใช้ของซอฟต์แวร์ (Software requirements) มากำหนดรายละเอียดโครงสร้างภายในของซอฟต์แวร์ เพื่อนำไปพัฒนาระบบ
- การวางแผนโครงการและการจัดระเบียบของซอฟต์แวร์
- กระบวนการกำหนดสถาปัตยกรรม (Architecture) ส่วนประกอบ (component) ส่วนประสาน (interface) และลักษณะด้านอื่นๆ ของระบบ หรือส่วนประกอบของระบบ เพื่อให้ได้มาซึ่งแบบจำลองของการออกแบบ (Design Model)
- การนำหลักการออกแบบต่างๆ มาตีกรอบเพื่อทำให้เราสามารถเปลี่ยน software concept เป็น reality ได้ หรือจะพัฒนาไปในทางไหน

Software Design in SDLC



Software Design

- Good design isn't just about code. It is about being able to **express ideas for your software** with other developers, other teams, and your clients.
- Having a well-thought design makes your software **easier to implement, reduces a need for major changes** later or the project and it saves you from headaches down the line.
- Knowledge of Software Design and Architecture will help your software **become flexible, reusable, and maintainable**.

What is Software Design?

คือ กระบวนการจัดระเบียบ code ที่รวม classes, packages, functions, etc. และความสัมพันธ์ระหว่างสิ่งเหล่านั้น รวมถึงกระบวนการกำหนดสถาปัตยกรรม (Architecture) ของระบบ

ไม่ว่าเราจะพัฒนาซอฟต์แวร์อย่างไร ซอฟต์แวร์เราก็จะมี Design หรือ สถาปัตยกรรม (Architecture) อะไรสักอย่าง (เพราะ เราสามารถหา ความสัมพันธ์ของสิ่งข้างต้นได้)

คำถามที่ควรถามคือ
เราจะรู้ได้อย่างไรว่าซอฟต์แวร์เรามี Design ที่ดี
(is the design any good?)

What Makes a “Good” Design?

What is a “*good*” design doesn’t have obvious answer. Physical systems: durability and longevity



Software is ***malleable*** in a way that physical items cannot be. And a good design in software focuses on “**adaptability and maintainability**.” But changes can introduce complexities and new issues.

Software Entropy

กล่าวถึงการความยุ่งเหยิงของซอฟต์แวร์ที่เกิดขึ้นตามเวลา ซึ่งมักเกิดจากผลสะสมของการเปลี่ยนแปลงที่เกิดขึ้น การอัพเดท การปรับปรุง และแพตช์ที่ซอฟต์แวร์ได้รับทีละเล็กละน้อย

เช่น เมื่อเราเพิ่ม features เข้าไปในซอฟต์แวร์, มันก็จะ

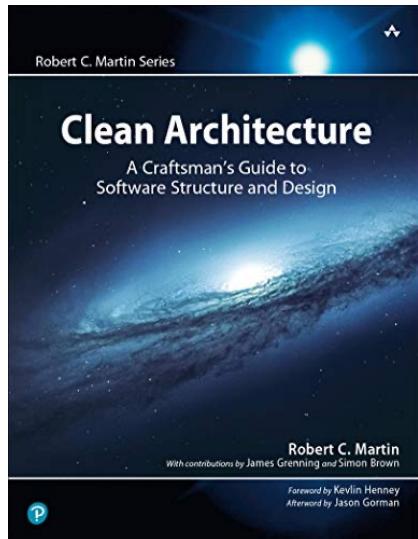
- ซับซ้อนเพิ่มมากขึ้น
- อ่านให้เข้าใจยากมากขึ้น
- โครงสร้างของโค้ดแย่ลง
- หนี้ทางเทคนิคเพิ่มสูงขึ้น (Technical debt)

สิ่งต่างๆเหล่านี้ทำให้ Maintain ซอฟต์แวร์อยากขึ้น และทำให้ใช้เวลาและแรงงาน (Effort) มาตรฐานมากขึ้น → ทำให้ cost เพิ่มขึ้น (cost of development ในที่นี้คือแรงและเวลาที่เสียไปรวมถึงกำลังเงิน)



Software ที่มี Design ที่ดีสามารถลดเวลาของการเกิด Software Entropy ได้และช่วยให้ overall effort ที่ใช้ในการแก้ไขหรือเปลี่ยนแปลง Software ลดลง

The Measure of Design Quality



“The measure of design quality is simply the measure of **the effort required to meet the needs of the customer**. If that effort is low, and stays low throughout the lifetime of the system, the design is good. If that effort grows with each new release, the design is bad. It’s as simple as that”



Robert C. Martin

Clean Architecture: A Craftman's Guide to Software Structure and Design

Planning For Change

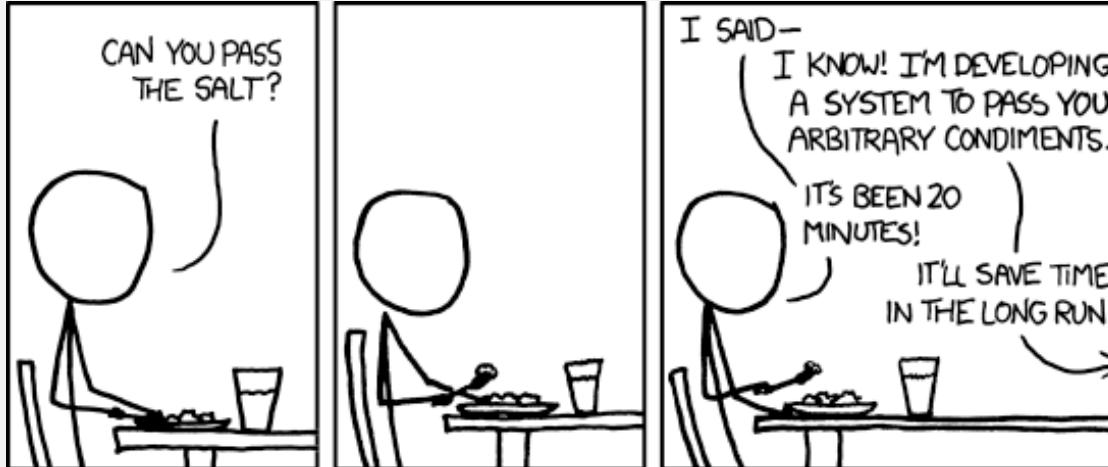
ดังนั้นกุญแจสำคัญของ Design ที่ดีคือการวางแผนสำหรับการเปลี่ยนแปลงที่อาจเกิดขึ้นได้กับซอฟต์แวร์และทำให้ซอฟต์แวร์เรา maintain ได้ง่ายที่สุดเท่าที่จะทำได้

แต่ถึงอย่างนั้นเราก็คาดเดาการเปลี่ยนแปลงที่อาจเกิดขึ้นได้ ได้ยาก



เพราะฉะนั้นเราต้องการที่จะดูระเบียบส่วนประกอบ,
โครงสร้าง, และความสัมพันธ์ต่าง ๆ ของระบบของเรา
ให้ดีที่สุด เพื่อที่ลดผลกระทบจากการเปลี่ยนแปลง
ไม่ให้กระจายออกไปกระทบส่วนต่างๆ ของระบบ

Over-Design Dilemma



ที่กล่าวมาทั้งหมดก่อนหน้า ไม่ได้มายความว่าเราต้อง “over-design”

- เราควรพิจารณาวงจรชีวิตระยะยาวของซอฟต์แวร์, จุดประสงค์ของซอฟต์แวร์, และความสำคัญของซอฟต์แวร์ (ถ้าซอฟต์แวร์ล้มจะมีผลกระทบอย่างไร) เช่น สำหรับระบบควบคุมการจราจรทางอากาศ เราคงต้องการที่จะให้ความสำคัญกับการ design (ไปจนถึง over-design) ระบบ
- แต่หากซอฟต์แวร์ของเรา มีจุดประสงค์เดียวและคาดว่าจะมีอายุการใช้งานที่สั้น, เราอาจจะโอบอุ่นว่าซอฟต์แวร์จะเป็นแบบที่เปลี่ยนแปลงหรือนำมาใช้ในจุดประสงค์อื่นๆ ได้ยาก (ไม่ต้องคำนึงถึง design ที่ดีมากนัก)

Example:

- บน CMU Moodle อาจารย์สามารถโหลดคะແນ Quiz ของนักศึกษามาเป็น excel .xlsx ได้ เพื่อใช้ต่อ. ในโปรแกรมของอาจารย์ต้องการ
 - เปิดและอ่านไฟล์ .xlsx ที่มี student ID อยู่ column ที่ 3, คะແນอยู่ column ที่ 4
 - ค้นหาคะແນสูงสุดของแต่ละคน (quiz นี้สามารถทำได้หลายรอบ)
 - ถ้านักศึกษาได้ 8 ข้อจาก 10 ขึ้นไปให้ปัดขึ้นเป็นได้คະແນเต็ม
 - ถ้าได้ต่ำกว่า 8 ข้อให้ scale เป็นเต็ม 8, เพราะฉะนั้น 6/8 จะได้ 75 คะແນ (แทนที่จะได้ 60 คะແນ)
 - save ไฟล์เป็น csv และขึ้นในระบบส่วนกลางที่ต้องการ student ID อยู่ใน column ที่ 1

Potential Change for the Future?

Things that Could Be Changed

- Export file to different extensions (.csv,.json, etc.)
- The column numbers (take into account different file templates)
- Identifier – accomodate student ID and student Number (look up functionality)
- The score threshold from 8 to 9
- Number of quiz questions
- Upload file format template to other recording systems?
- User-friendliness
- ***What else?***

Should I Plan for Them?

- อาจจะไม่
- โปรแกรมขนาดเล็ก ไม่ใช่ระบบขนาดใหญ่ แก้สิ่งต่างๆได้ไม่ยาก
- ใช้งานโดยอาจารย์ผู้สอนวิชานั้นเท่านั้น (อย่างมากสองคน)
- User friendliness ก็ไม่ต้องคำนึงถึงมาก (เพราะเหตุผลข้างบน)
- CMU Moodles จะปิดตัวลงแล้ว มช จะให้อาจารย์ทุกคนมาใช้ Mango เพราจะนั้น script ที่ใช้อยู่อาจจะไม่ได้ใช้งานอีก หลังจากที่ Moodles ปิดตัวลง
- และอื่นๆ

Agile Design

- **Iterative Refinement in Agile:** Agile methodology supports the idea that every design will eventually require changes, some of which may be unforeseen, promoting the practice of writing code that is understandable, changeable, and resistant to unexpected changes.
- **Modular Design Principle:** A key practice in agile development is adopting a modular design, where different components of the software can be independently evolved or altered over time.
- **Lifecycle Considerations in Design:** The design approach should be aligned with the expected lifecycle of the software, considering factors like longevity, user needs evolution, and the extent of anticipated changes.

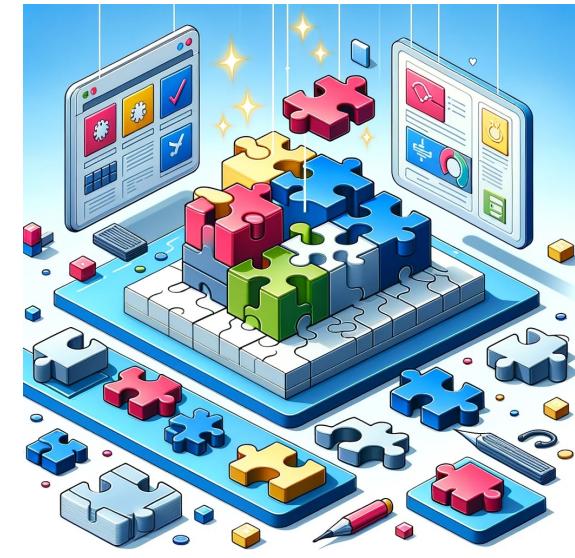
Software Design Aims to Make Software



Flexible



Maintainable



Reusable

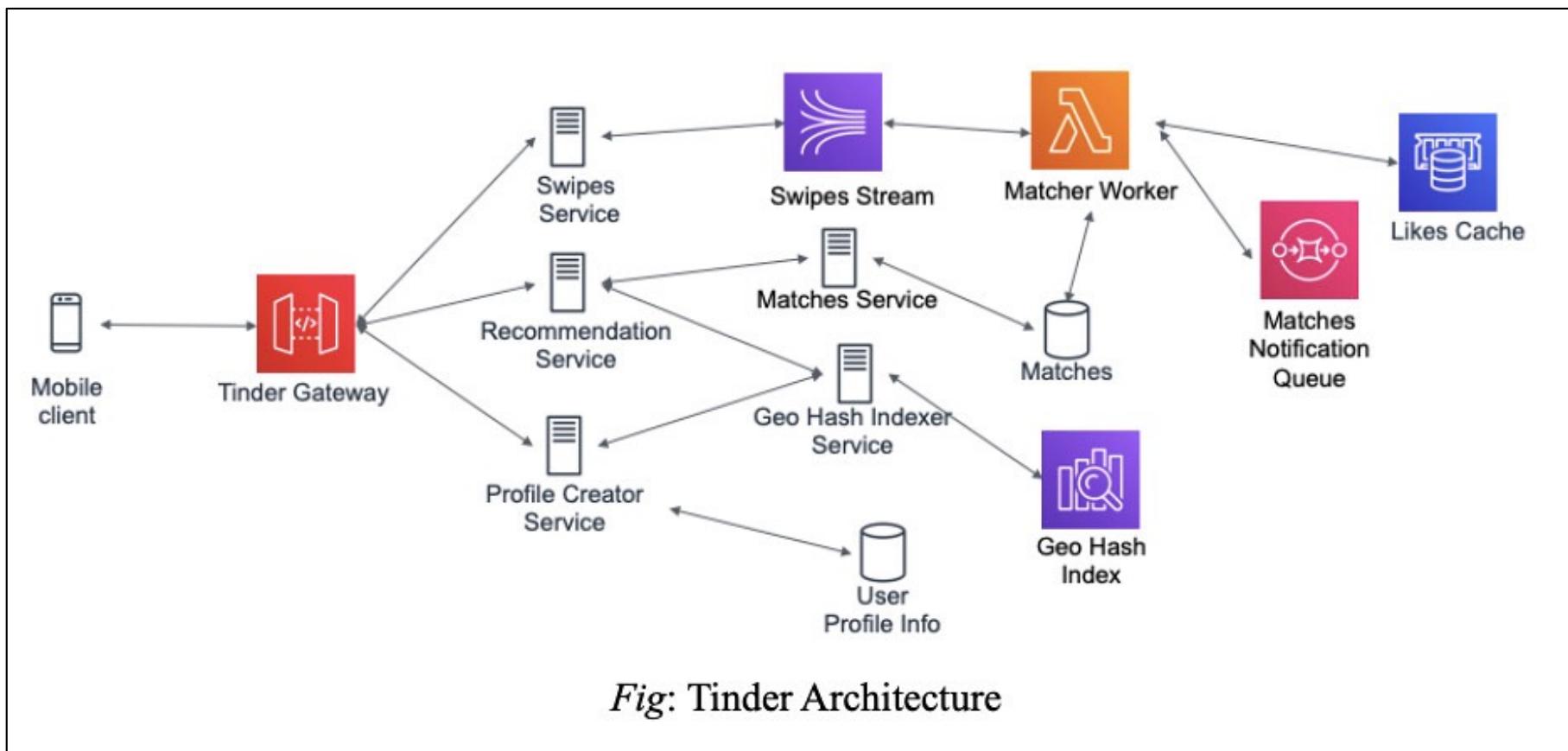
Software Design Process

- การทำงานแบบ iterative (วนซ้ำ) เพื่อวิเคราะห์ข้อมูลในหลายๆ ด้าน เช่น business, functional requirements, non-functional requirements, และจากข้อมูลอื่นๆ ที่เกี่ยวข้อง
- สามารถแบ่งออกเป็นสองระดับใหญ่
 - เชิงสถาปัตยกรรม (Architecture Design)
 - โครงสร้างหลักของซอฟต์แวร์ว่ามีส่วนงานอะไรบ้าง มีโครงสร้างย่อยอะไรบ้าง ให้มองเห็นภาพรวม โดยไม่ลงลึกในรายละเอียดของแต่ละส่วน
 - เชิงส่วนรายละเอียด (Detailed Design)
 - อธิบายรายละเอียดของแต่ละส่วนประกอบของซอฟต์แวร์ เพื่อช่วยให้เขียนโปรแกรมได้ง่ายที่สุด (Algorithm อะไร, flowchart ของขั้นตอนการทำงานของ function, โครงสร้างการจัดเก็บข้อมูล, และอื่นๆ)

Example: Architecture Design

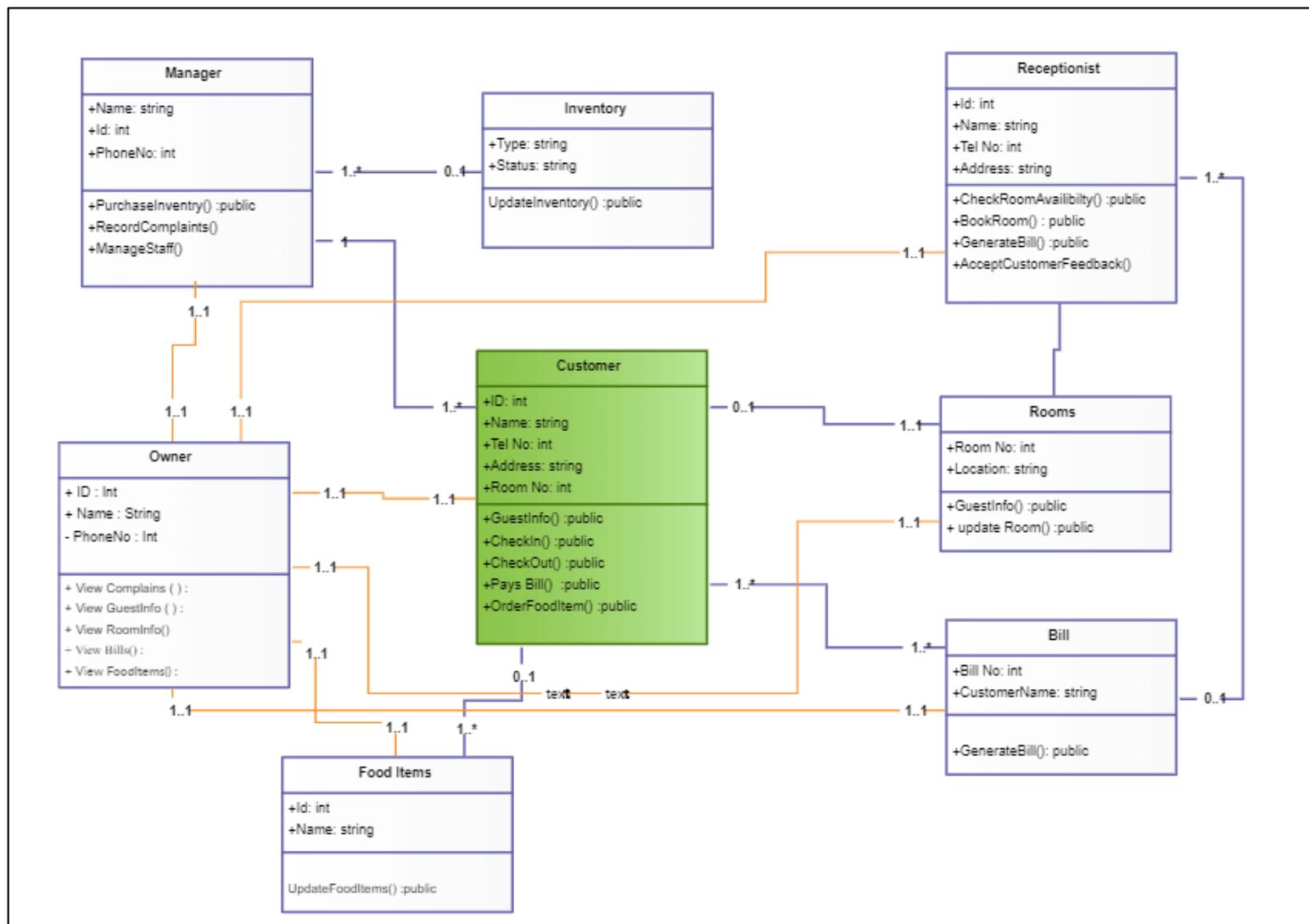
Architecture ของ Tinder

- บอกภาพโดยรวมว่ามีส่วนอะไรบ้าง อันไหนติดต่อกับอันไหน โดยยังไม่ลง detail

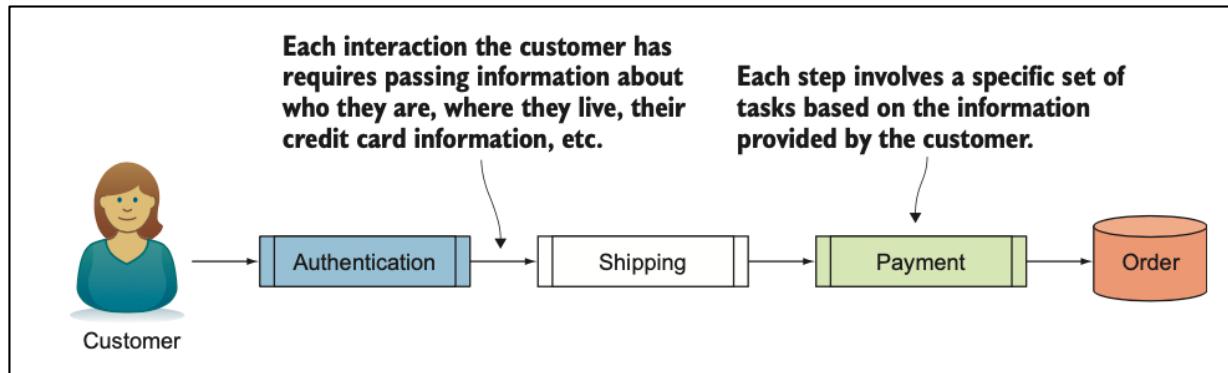


Example: Detailed Design

Class Diagram ຂອງຮະບຸ Hotel Management



Design is a Process



Software design involves planning and sketching systems for effective execution. Software design focuses on data flow and system components.

You've likely experienced a design process, often without realizing it.

Moments like these:

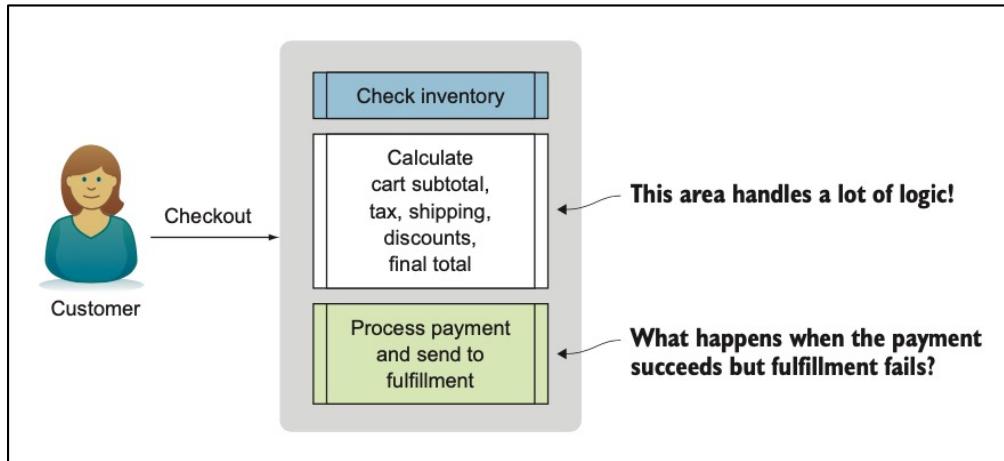
- “Pausing code writing to reassess goals”
- “Using the assessment to guide you to code”
- “Discovering new directions or more efficient methods”

Software is not only for “the user”

- Serves multiple audiences (including the developers themselves)
- Some **trade-offs** you have to make
- Users may want:
 - **Speed** – does its job as quickly as it can
 - **Integrity** – protected from corruption
 - **Resources** – use network bandwidth efficiently
 - **Security** - can read and write data only where authorized
- Developers may want:
 - **Loose coupling** – components are not intricately dependent on one another
 - **Flexibility** – can adapt the software to related tasks
 - **Extensibility** – can change something without affecting other aspects.

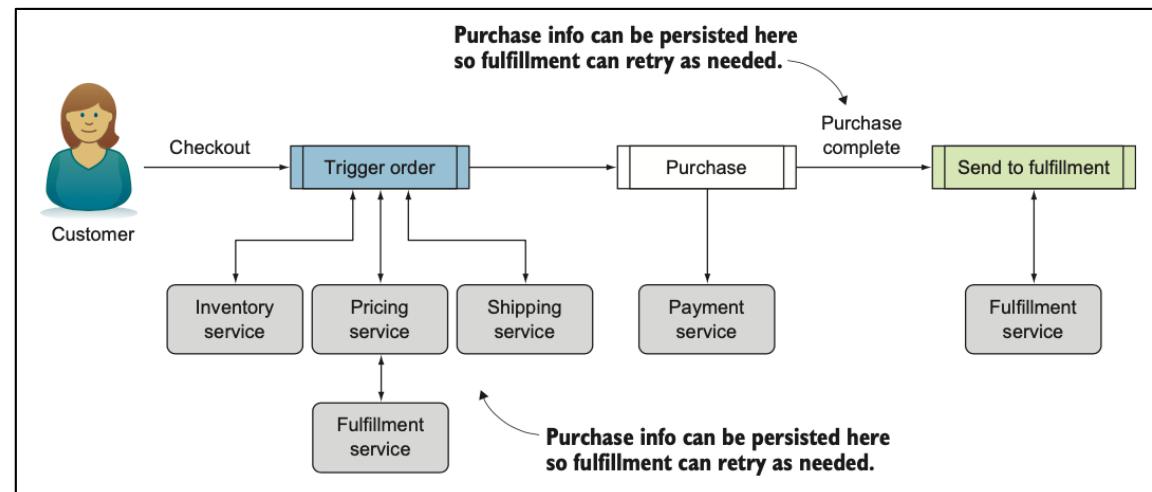
This is where careful design and thinking
about the system as a whole will help.

Example: An E-commerce Website



An e-commerce system created without much thought

Improvement through
Decomposition!



What a thoughtfully planned e-commerce system might look like

From Requirements to Design - 1

เปรียบเทียบกับพัฒนาซอฟต์แวร์กับการสร้างบ้าน

- **House Analogy:**

- นักศึกษาถูกจ้างให้สร้างบ้าน ก่อนที่จะสร้างมันขึ้นมา นักศึกษาต้องเข้าใจก่อนว่าผู้ว่าจ้างต้องการบ้านแบบไหน
- เจ้าของบ้านบอกว่าอยากรื้อบ้านชั้นเดียว ที่มี gym, 3 ห้องน้ำ, 2 ห้องนอน, และ ห้องนั่งเล่น
- การหาความต้องการไม่ใช่แค่ฟังว่า เจ้าของบ้านบอกว่าอะไร แต่ต้องคิดถึงสิ่งที่เจ้าของบ้านไม่บอกด้วย เช่น
 - ????
 - ????
- Process นี้เรียกว่า Requirements Elicitation (การหาความต้องการ)

From Requirements to Design - 2

- House Analogy:

- เมื่อได้ความต้องการ เราจะสามารถ outline วิธีการแก้ปัญหา (การสร้างซอฟต์แวร์นั้นได้)
- Architecture (**conceptual designs**) → สร้างเป็น Diagram (Boxes and Lines)
- เช่น components มีอะไรบ้าง เชื่อมต่อ (connect) กันอย่างไร แต่ละ component มีบทบาทหน้าที่ (responsibility) อะไร
- หลังจากที่ได้ **conceptual design** (floor plan) แล้วก็มาคิดว่าจะวาง plumbing, electrical ของทั้งบ้านอะไรยังไง



From Requirements to Design -3

- House Analogy:

- **Technical Design** – ดูแต่ละ component ว่า ต้องอะไรยังไง เช่น Gym อาจจะต้องมีการ reinforce พื้นให้แข็งแรงสามารถหันหน้าหันหลังได้โดยไม่ทำให้พื้นพังหรือบ้านทรุด
- Conflict อาจเกิดขึ้นได้
- Iterative Approach – คุยกับ client ตรวจสอบว่าตรงตามที่ต้องการมั้ย หรือ กลับไปเริ่มคิด conceptual design ใหม่ถ้า technical design บอกว่า conceptual design มีปัญหา
- *Adjustments are easier on paper than after implementation*



What is Career in Software Design and Architecture?

Career in Software Design and Architecture

- แต่ละบริษัทอาจจะมี Role ไม่เหมือนกันขึ้นอยู่กับสิ่งต่าง ๆ เช่น ขนาดของทีม, scope ของ project, ประสบการณ์ของ dev, แบบแผนที่ทีมใช้ เป็นต้น
- ในบางบริษัทอาจจะมี Role แยกต่างหากสำหรับ Software Designer หรือ Architect (เหมือนที่พี่ๆ เบทาโน่ได้กล่าวไว้) แต่ในบางบริษัท Software Designer/Architect อาจเป็นคนในทีม dev เอง
- Software Designer focuses on **detailed design** aspects of software components (แต่ละ individual component ว่ามีรายละเอียดอะไรบ้างหน้าที่คืออะไร)
- ส่วน Software Architect จะมองระบบ **ทั้งหมด (high-level)** และเลือกสิ่งต่างๆ ที่เหมาะสม เช่น framework, วิธีการจัดเก็บข้อมูล, การกำหนดส่วนประกอบของ component, และ relationship ระหว่าง component เหล่านั้น

Characteristics of a Software Designer and a Software Architect

- **Creative problem solver** who can come up with a quality solution for the problem at hand.
- Able to **communicate or express ideas** effectively with product manager and development team.
- **Understand at both low and high levels** what the client wants
- **Detailed**-oriented.

Building a House Analogy

- A Software Architect = *A Building Architecture*
- A Software Designer = *A Interior Designer*



Focuses on **major structures**, the materials, the integrity, + services like electrical and plumbing)



Focuses on the **flow** between rooms and the **details** that make each room beautiful, comfortable, and usable (the color schemes, materials, finishes, and furnishings)

Software Design in the Industry

Reflection

Software Projects You Worked On

ลองนึกถึงโปรเจคซอฟต์แวร์ที่นักศึกษาเคยได้ทำ และตอบคำถามต่อไปนี้

1. การออกแบบของตัวซอฟต์แวร์นั้นหมายถึงอะไร?
2. การออกแบบตัวซอฟต์แวร์เป็นอย่างไร ?
3. สามารถทำให้ออกแบบดีขึ้นได้มั้ย ?
4. นักศึกษารู้ได้อย่างไรว่าซอฟต์แวร์ถูกออกแบบมาดีหรือไม่ดี ?
5. ปรับเปลี่ยนอะไรในโค้ดของซอฟต์แวร์นั้นง่ายมั้ย ? (มีเหตุการณ์เช่น แก้ไขอะไรเล็กๆ ทำให้โค้ดส่วนอื่นพังหรือเปล่า)
6. นำบางส่วนหรือทุกส่วนของโค้ดกลับมาใช้ใหม่ได้ - reuse (เช่น กับโปรเจคอื่น) ได้มั้ย?
7. การทำ maintenance task หลังจากเราทำมันเสร็จง่ายหรือเปล่า ?

Class Activity: Find Out and Report

- ให้ระบุสองสิ่ง (หรือกิจกรรม) ที่ต้องทำถ้าจะกล่าวถึง Software Design?
- ให้ระบุบางสิ่งที่เป็นส่วนประกอบของ Software Design?
- ให้ระบุว่าใครใช้ผลลัพธ์ของการทำ Software Design และเพื่ออะไร?

Questions?

