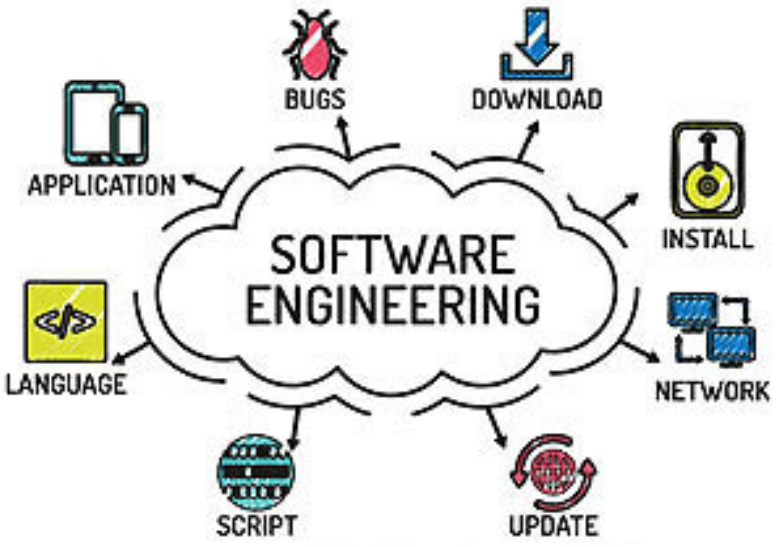
Chapter 1 : Introduction to Software Engineering

ASSIT.PROF. JUTHAWUT CHANTHARAMALEE

CURRICULUM OF COMPUTER SCIENCE

FACULTY OF SCIENCE AND TECHNOLOGY, SUAN DUSIT UNIVERSITY



Outline of this presentation

1. ซอฟต์แวร์ การเปลี่ยนแปลง และปัญหาที่พบ
2. วิศวกรรมซอฟแวร์และความสำคัญ
3. องค์ประกอบของวิศวกรรมซอฟต์แวร์
4. วิวัฒนาการของวิศวกรรมซอฟต์แวร์
5. คุณลักษณะของซอฟต์แวร์ที่มีคุณภาพ
6. ระเบียบวิธีปฏิบัติของวิศวกรรมซอฟต์แวร์

ซอฟต์แวร์ การเปลี่ยนแปลง และปัญหาที่พบ

วิศวกรรมซอฟต์แวร์ (Software Engineering)

มุมมองทางการศึกษาในแง่ของสาขาวิชา

ในปี ค.ศ. 1968 คำว่า *"วิศวกรรมซอฟต์แวร์ (software engineering)"* ถูกใช้อย่างแพร่หลาย เพื่อแสดงถึงกิจกรรมต่าง ๆ ที่รวมถึงการเขียนโปรแกรม (programming) และการรหัส (coding) [Macro, 1987].

ก่อนปี ค.ศ. 1974 สาขาวิชาวิศวกรรมซอฟต์แวร์ยังไม่ปรากฏ [Barnes, 1998].

สถาบันเทคโนโลยีโรเชสเตอร์ (The Rochester Institute of Technology (RIT)) ในประเทศสหรัฐอเมริกาได้อ้างว่าเป็นสถาบันแรกที่แนะนำหลักสูตรปริญญาตรีสาขาวิศวกรรมซอฟต์แวร์ [Lutz, 1999].

ความหมายวิศวกรรมซอฟต์แวร์ (Software Engineering Definition)

วิศวกรรมซอฟต์แวร์ คือกระบวนการสร้างสรรค์โปรแกรมโดยใช้หลักทางวิศวกรรมเข้ามาช่วยในการดำเนินการสร้าง (อ.สมหมาย สุขคำ)

"Software Engineering is systematic approach to the development operation , maintenance , retirement of software" (IEEE 83b)

ความหมายที่ 1 หมายถึง วิชาการว่าด้วยการออกแบบโปรแกรมคอมพิวเตอร์ ตลอดจนการบริหารงานการพัฒนาเพื่อที่จะได้มาซึ่ง ผลิตผลซอฟต์แวร์ที่มีคุณภาพสูง ราคาถูก และภายในเวลาที่กำหนดให้ (สุชาย ธนวเสถียร)

ความหมายที่ 2 หมายถึง การพัฒนาซอฟท์แวร์ให้ได้ผลลัพธ์ใกล้เคียงเป้าหมายหรือบรรลุเป้าหมายของการพัฒนา อันได้แก่

1. ซอฟท์แวร์ที่มีคุณภาพ
2. สามารถส่งมอบได้ตรงเวลา
3. อยู่ภายใต้งบประมาณที่คาดการณ์
4. มีคุณสมบัติตรงตามความต้องการของผู้ใช้ (ผศ.ดร.สมนึก คีรีโต)

ความหมายที่ 3 เป็นศาสตร์เกี่ยวกับวิศวกรรมด้านซอฟต์แวร์มีเนื้อหาเกี่ยวข้องกับการใช้กระบวนการทางวิศวกรรมในการดูแลการผลิต ตั้งแต่การเริ่มเก็บความต้องการ การตั้งเป้าหมายของระบบ การออกแบบ กระบวนการพัฒนา การตรวจสอบ การประเมินผล การติดตามโครงการ การประเมินต้นทุน การรักษาความปลอดภัย ไปจนถึงการคิดราคาซอฟต์แวร์ เป็นต้น

ความหมายที่ 4 หมายถึง การนำแนวทางที่เป็นระบบ มีระเบียบ กฎเกณฑ์ และสามารถวัดผลในเชิงปริมาณได้ มาประยุกต์ใช้ในการ พัฒนา ปฏิบัติการ และบำรุงรักษาซอฟต์แวร์ ซึ่งก็คือ เพื่องานด้าน วิศวกรรมการผลิตซอฟต์แวร์ หรือกล่าวอีกนัยหนึ่งคือ เป็นการศึกษาวิธีการผลิตซอฟต์แวร์นั่นเอง [IEEE, 2004]

ความหมายที่ 5 หมายถึง การผสมผสานระหว่างศาสตร์และศิลป์ เพื่อการผลิตซอฟต์แวร์เชิงพาณิชย์ โดยเริ่มต้นตั้งแต่การจัดทำข้อกำหนดคุณสมบัติของระบบ ตลอดจนการบำรุงรักษาระบบให้เป็นปกติ [Sommerville, 2007] โดยแบ่งนัยสำคัญออกเป็น 2 ประเด็น คือ

1. สหวิทยาการจัดการด้านวิศวกรรม
2. ผู้เชี่ยวชาญด้านการผลิตซอฟต์แวร์

ความหมายที่ 6 หมายถึง การนำหลักวิชาการด้านวิศวกรรมมาดูแลกระบวนการผลิตซอฟต์แวร์ ตั้งแต่ขั้นตอนแรกจนถึงขั้นตอนบำรุงรักษาหลังการใช้งาน เพื่อให้ซอฟต์แวร์ที่ได้มีคุณภาพสูงสุดภายใต้ข้อจำกัดด้านเวลาและต้นทุน (กิตติ ภักดีวัฒนะกุล)

บทบาทที่เปลี่ยนแปลงไปของซอฟต์แวร์

1. โปรแกรม (Program)
2. ซอฟต์แวร์ (Software)
3. แอปพลิเคชันซอฟต์แวร์ (Application Software)
4. ซอฟต์แวร์สำหรับแก้ปัญหา (Software Solution)

ซอฟต์แวร์ (Software) ชุดคำสั่งที่เป็นตัวสั่งการทำงานของคอมพิวเตอร์

ประเภทของซอฟต์แวร์

แบ่งตามวัตถุประสงค์การใช้งาน ออกเป็น 7 กลุ่ม ดังนี้

1. ซอฟต์แวร์ระบบ (System Software)

เป็นซอฟต์แวร์ที่ประกอบไปด้วยกลุ่มของโปรแกรมย่อยที่ถูกเขียนขึ้นมาเพื่อให้บริการโปรแกรมอื่น

2. ซอฟต์แวร์แอปพลิเคชั่น (Application Software)

เป็นโปรแกรมแก้ปัญหางานทางธุรกิจโดยเฉพาะ ทำงานบนเครื่องคอมพิวเตอร์แบบ Standalone และบางครั้งสามารถทำงานแบบเวลาจริง (Real-time)

3. ซอฟต์แวร์ด้านวิทยาศาสตร์และวิศวกรรม (Scientific Software/Engineering

เป็นซอฟต์แวร์ที่ใช้เฉพาะงานด้านวิทยาศาสตร์และวิศวกรรมศาสตร์

4. ซอฟต์แวร์แบบฝัง (Embedded Software)

เป็นซอฟต์แวร์ที่ถูกติดตั้งไว้ภายในอุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ หรือภายในระบบงาน

5. ซอฟต์แวร์แบบสายการผลิต (Product-Line Software)
   - เป็นซอฟต์แวร์เฉพาะด้านที่ลูกค้าหลายกลุ่มสามารถนำไปใช้งานได้เหมือนกันหรืออาจเป็นกลุ่มลูกค้าเฉพาะ และลูกค้าตลาดใหญ่ที่เป็นผู้ใช้ทั่วไป
6. เว็บแอปพลิเคชั่น (Web Application)
   - กรณีที่ซอฟต์แวร์แอปพลิเคชั่นสามารถทำงานบนเว็บไซต์ เพื่อจัดการข้อมูลในฐานข้อมูลบนเว็บได้
7. ซอฟต์แวร์ปัญญาประดิษฐ์ (Artificial Intelligence Software)
   - เป็นซอฟต์แวร์ที่ถูกออกแบบให้มีอัลกอริทึมในการทำงานที่ซับซ้อนเลียนแบบสมองมนุษย์ เพื่อแก้ปัญหาที่มีความซับซ้อนสูงด้วยการวิเคราะห์ตามหลักของเหตุและผล

แบ่งตามอุตสาหกรรมการผลิตซอฟต์แวร์ ได้ 2 ประเภท ดังนี้

1. Generic Product
   - เป็นผลิตภัณฑ์ซอฟต์แวร์หรือระบบที่ถูกผลิตภัณฑ์ซอฟต์แวร์หรือระบบที่ถูกผลิตขึ้นโดยผู้ผลิตซอฟต์แวร์รายใหญ่ (Software Vendor) เพื่อจำหน่ายให้กับลูกค้าในตลาดซอฟต์แวร์ทั่วไปที่ต้องการซื้อไปใช้งานตามความสามารถของซอฟต์แวร์
2. Customized Product
   - เป็นผลิตภัณฑ์ซอฟต์แวร์หรือระบบ สำหรับลูกค้าเฉพาะรายที่ได้ตกลง ทำสัญญาว่าจ้าง

ข้อแตกต่าง Generic Product และ Customized Product

Generic Product	Customized Product
ผลิตขึ้นมาโดยไม่ยึดความต้องการของลูกค้า นับเป็นการควบคุมความต้องการของลูกค้า	ผลิตขึ้นมาตามความต้องการ กำหนด และควบคุมโดยลูกค้า

ที่มาของวิศวกรรมซอฟต์แวร์

1. ซอฟต์แวร์ได้มีการเปลี่ยนแปลงบทบาทหน้าที่
2. ฮาร์ดแวร์คอมพิวเตอร์มีประสิทธิภาพมากขึ้นและราคาถูกลง
3. ความซับซ้อนที่เพิ่มมากขึ้นของซอฟต์แวร์
4. ซอฟต์แวร์ล้าสมัยกลายเป็นซอฟต์แวร์เก่า (Legacy Software)

วิศวกรรมซอฟแวร์และความสำคัญ

ประโยชน์ของวิศวกรรมซอฟต์แวร์

1. กระบวนการผลิตซอฟต์แวร์ที่มีประสิทธิภาพ เป็นระบบ
2. มีมาตรฐานกำหนดวิธีการทำงานอย่างชัดเจน
3. มีการตรวจสอบคุณภาพของซอฟต์แวร์
4. มีเอกสารควบคุมกำกับการทำงานตลอดทั้งกระบวนการ
5. มีการตรวจสอบและประกันคุณภาพของซอฟต์แวร์ที่ผลิตก่อนส่งถึงมือผู้บริโภค
6. สามารถทำงานได้ ถึงแม้ว่าจะเปลี่ยนทีมงาน

อำนาจในการจัดหาเทคโนโลยีเชิงวัตถุเครื่องคอมพิวเตอร์มาใช้

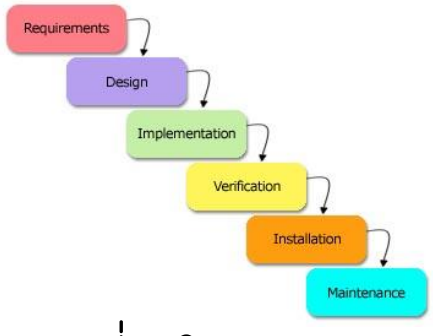
เครือข่ายคอมพิวเตอร์

ระยะเวลาเปิดตัวผลิตภัณฑ์

การเปลี่ยนแปลงในวิศวกรรมซอฟต์แวร์

เศรษฐกิจที่เจริญเติบโตขึ้น

Graphic User Interface



ปัญหาที่พบใน WaterFall Model

รูปแสดงปัจจัยที่ทำให้เกิดการเปลี่ยนแปลงไปสู่วิศวกรรมซอฟต์แวร์

ความสำคัญของวิศวกรรมซอฟต์แวร์

1. ปัจจัยการเปลี่ยนแปลงที่ทำให้งานด้านวิศวกรรมซอฟต์แวร์มีความสำคัญมากขึ้น ดังนี้
2. การเปลี่ยนแปลงของระยะเวลาการเปิดตัวผลิตภัณฑ์ที่รวดเร็วขึ้น
3. การเปลี่ยนแปลงในอุตสาหกรรมผลิตคอมพิวเตอร์
4. บุคคลทั่วไปหรือบริษัทขนาดเล็กมีอำนาจซื้อเครื่องคอมพิวเตอร์มากขึ้น
5. การแพร่หลายของการเชื่อมต่อเครือข่ายคอมพิวเตอร์ทั้งแบบท้องถิ่นและแบบระยะไกล
6. ความสามารถในการดัดแปลงใช้เทคโนโลยีเชิงวัตถุเข้ากับระบบงานได้
7. การเปลี่ยนแปลงของส่วนประสานกับผู้ใช้ที่มีแบบเป็นกราฟิกมากขึ้น
8. แบบจำลองของกระบวนการผลิตซอฟต์แวร์แบบ Waterfall ไม่สามารถคาดการณ์ความต้องการของผู้ใช้ได้อีกต่อไป

ความแตกต่างของวิศวกรรมซอฟต์แวร์และวิทยาการคอมพิวเตอร์

วิทยาการคอมพิวเตอร์ (Computer Science)

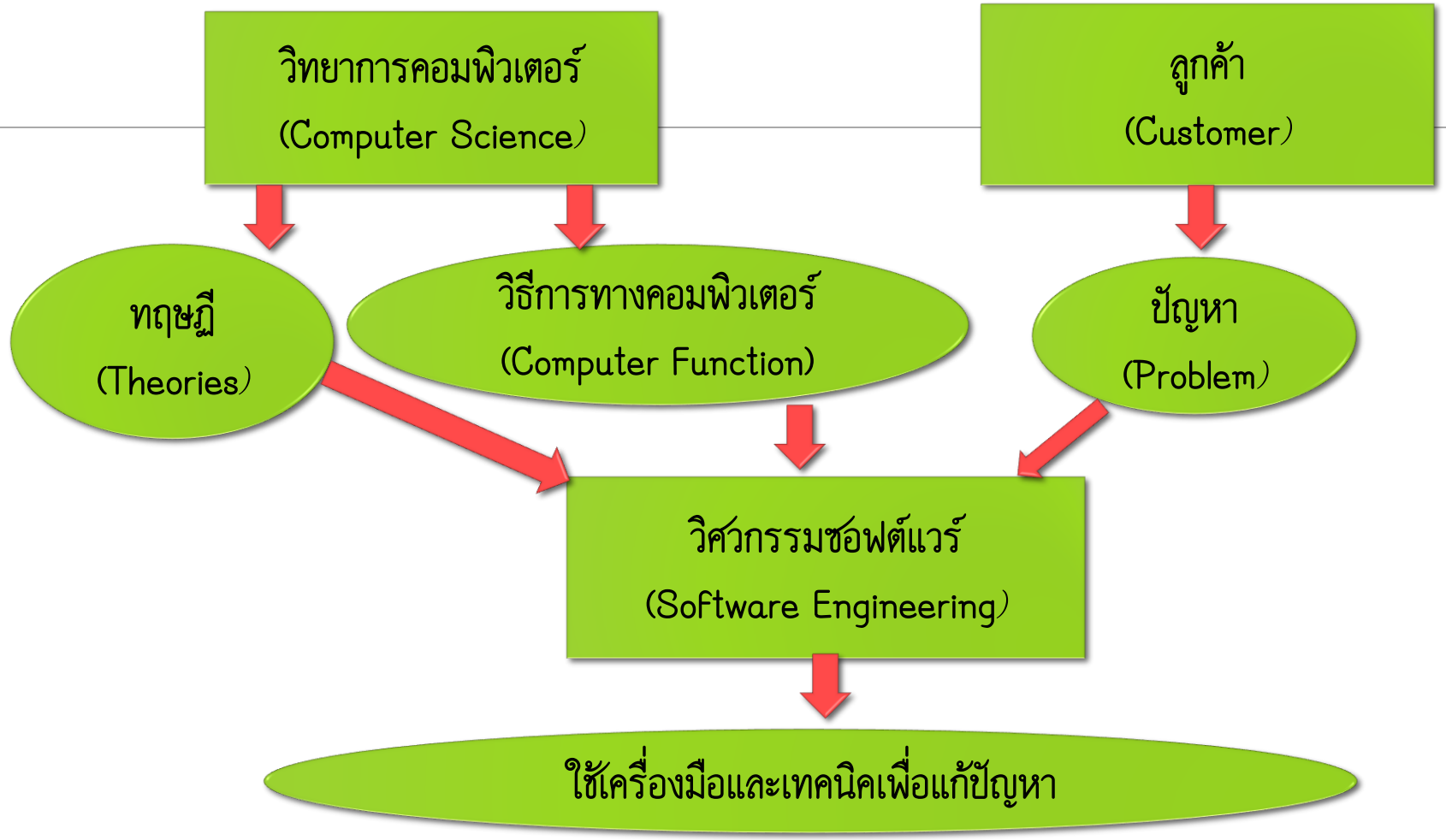
อยู่บนรากฐานของวิทยาศาสตร์ ซึ่งเน้นการทำความเข้าใจและค้นหาความจริงเกี่ยวกับความรู้ทางคอมพิวเตอร์ เพื่อสร้างแนวคิด/ทฤษฎีใหม่ หรือ ปฏิเสธแนวคิด/ทฤษฎีเดิม และขยายวงความรู้ให้กว้างขึ้นจากแนวคิด/ทฤษฎีที่มีอยู่

* ผลงานถูกพิจารณา หรือ ตัดสินโดยกลุ่มนักวิทยาศาสตร์

วิศวกรรมซอฟต์แวร์ (Software Engineering)

อยู่บนรากฐานของวิธีการทางวิศวกรรมศาสตร์ ซึ่งประยุกต์แนวคิด/ทฤษฎีทางวิทยาศาสตร์ คณิตศาสตร์และเทคโนโลยีขณะนั้นในการสร้างผลิตภัณฑ์ที่เป็นประโยชน์และปลอดภัยต่อสาธารณะ

* ผลงานถูกพิจารณา หรือ ตัดสินโดยกลุ่มผู้ใช้



ความสัมพันธ์ระหว่างวิศวกรรมซอฟต์แวร์และวิทยาการคอมพิวเตอร์

ความแตกต่างของวิศวกรรมซอฟต์แวร์และวิศวกรรมระบบ

วิศวกรรมระบบ (System Engineering)

เกี่ยวข้องกับทุก ๆ ด้านของการพัฒนาและการเปลี่ยนแปลงของระบบที่มีความซับซ้อน โดยมีซอฟต์แวร์เป็นแกนหลักในการทำงานของระบบ การวิศวกรรมระบบเป็นการกระทำที่ก่อให้เกิดการกำหนดระบบ ระบุถึงสถาปัตยกรรมทั้งระบบ แล้วนำส่วนประกอบที่แตกต่างกันมาประสานเข้าด้วยกันจนกลายเป็นระบบ 1 ระบบ

- การวิศวกรรมระบบเป็นศาสตร์ที่เก่าแก่กว่าวิศวกรรมซอฟต์แวร์
- ซอฟต์แวร์เป็นแกนหลักในการทำงานของระบบ

ความแตกต่างของวิศวกรรมซอฟต์แวร์กับการวิเคราะห์และออกแบบระบบ

การวิเคราะห์และออกแบบระบบ (System Analysis and Design)

เป็นการศึกษา วิเคราะห์ และแยกแยะปัญหาที่เกิดขึ้นในระบบ แล้วทำการออกแบบ และกำหนดคุณสมบัติทางเทคนิค โดยนำระบบคอมพิวเตอร์มาประยุกต์ใช้เพื่อแก้ปัญหาที่ได้ทำการวิเคราะห์มาแล้ว

ระบบที่ถูกนำมาวิเคราะห์และออกแบบส่วนใหญ่*เป็นระบบสารสนเทศ*ที่จะนำมาใช้ภายในองค์กร โดยมี *"นักวิเคราะห์ระบบ (System Analyst)"* เป็นผู้รับผิดชอบงานวิเคราะห์และออกแบบโดยตรง

ความแตกต่างของวิศวกรรมซอฟต์แวร์กับการวิเคราะห์และออกแบบระบบ

วิศวกรรมซอฟต์แวร์ (Software Engineering) ทำหน้าที่เกี่ยวกับการผลิตซอฟต์แวร์เพื่อการค้า กระบวนการที่นำมาใช้พัฒนาซอฟต์แวร์หรือระบบจะคล้ายคลึงกัน แต่ขั้นตอนของวิศวกรรมซอฟต์แวร์มีมากกว่าขั้นตอนของการวิเคราะห์และออกแบบ ขั้นตอนสำคัญของการวิเคราะห์และออกแบบมีเพียงการจัดทำความต้องการ (Requirement) การวิเคราะห์ (Analysis) และออกแบบ (Design) เท่านั้น

บุคคลที่เกี่ยวข้องกับงานวิศวกรรมซอฟต์แวร์

ผู้ใช้ (User)

ลูกค้า (Customer)

นักพัฒนา (Developer)

ความสัมพันธ์ระหว่างกลุ่มบุคคลที่เกี่ยวข้องกับงานวิศวกรรมซอฟต์แวร์

ลูกค้า (Customer)

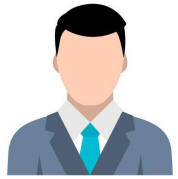
ให้เงินสนับสนุนการผลิตซอฟต์แวร์

ความต้องการ

ใช้งานซอฟต์แวร์ (ระบบ)

สร้างซอฟต์แวร์ (ระบบ)

ความต้องการ



ซอฟต์แวร์ (ระบบ)

ผู้ใช้ (User)

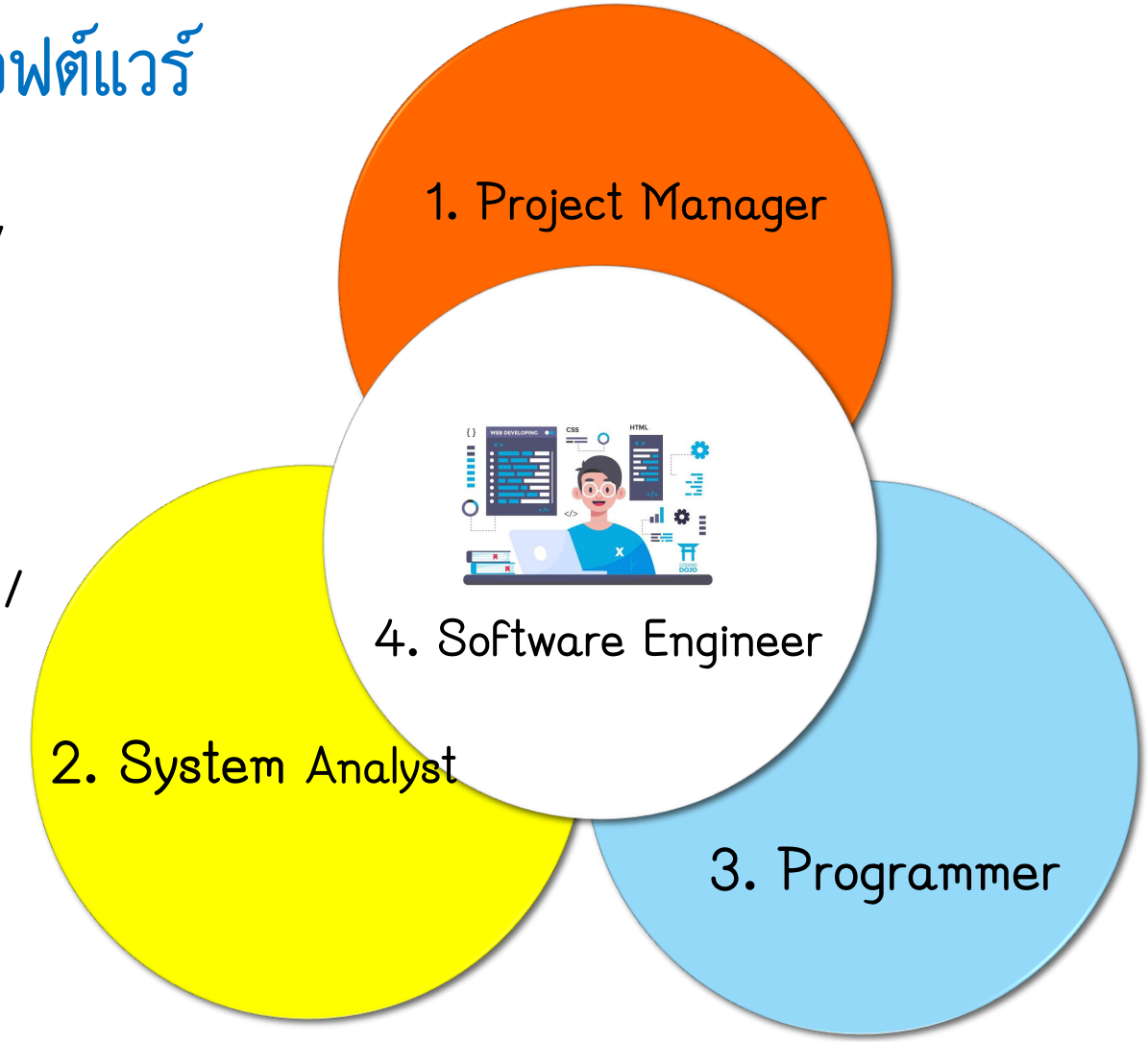
นักพัฒนา (Developer) วิศวกรซอฟต์แวร์ (Software Engineer)

คุณสมบัติของวิศวกรซอฟต์แวร์

1. มีความรู้ด้านการผลิตซอฟต์แวร์
2. มีความรู้ด้านการบริหารโครงการ
3. มีความรู้ด้านการจัดการ
4. มีความรู้ด้านธุรกิจ
5. มีความรู้ด้านประชาสัมพันธ์
6. มีความน่าเชื่อถือ
7. มีความรู้สึกไว
8. มีความเป็นผู้นำ
9. มีความอดทนต่อภาวะความกดดัน
10. มีความยืดหยุ่นสูง
11. มีความรับผิดชอบ
12 มีความยุติธรรม

บุคลากรที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์

1. Project Manager หน้าที่ วางแผนโครงงาน/ จัดการบุคลากร/ ควบคุม ตรวจสอบ
2. System Analyst หน้าที่ วิเคราะห์ความต้องการ/ออกแบบระบบตามความต้องการ
3. Programmer หน้าที่ ออกแบบ/เขียนโปรแกรม/ ทดสอบ แก้ไข

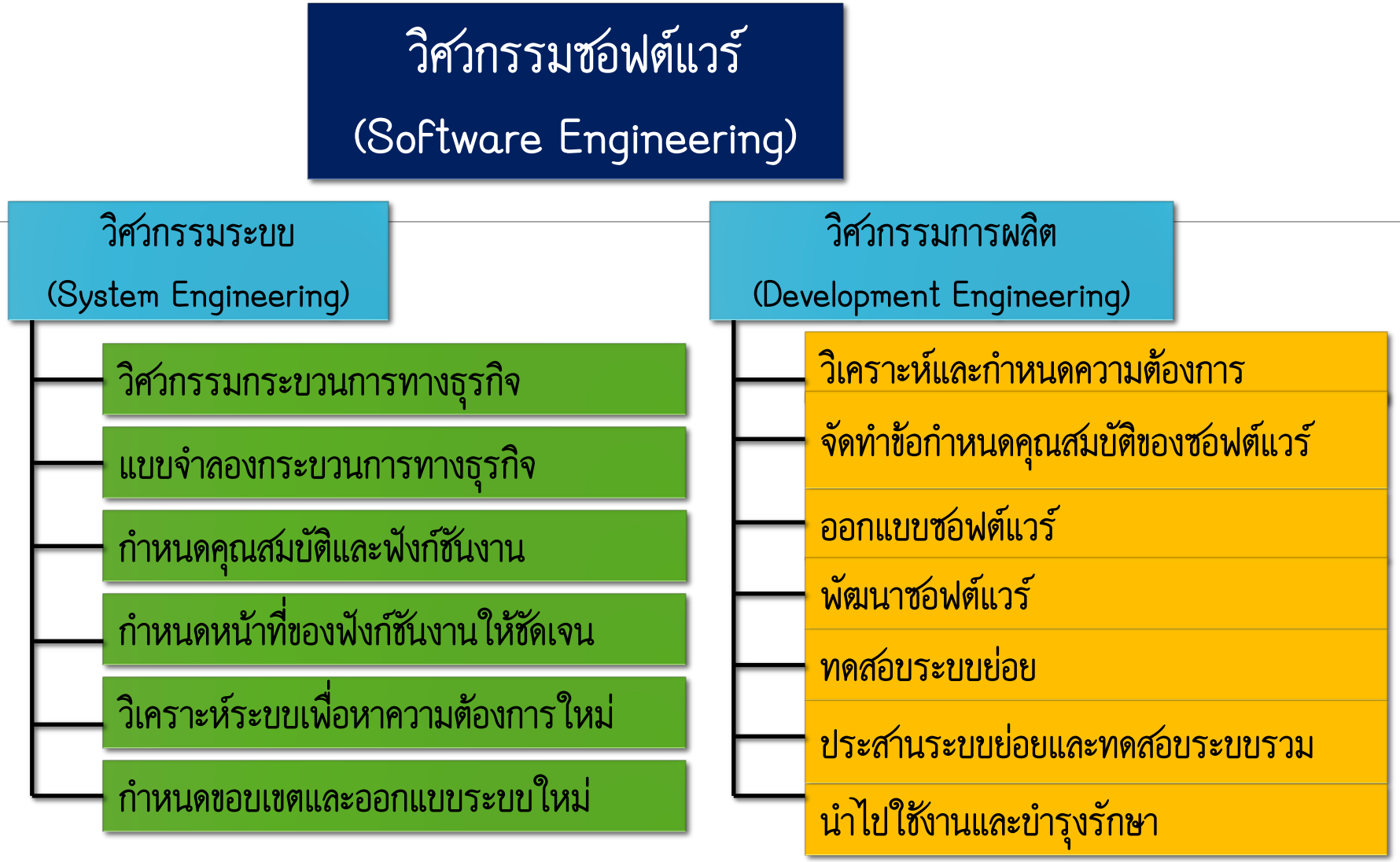


องค์ประกอบของวิศวกรรมซอฟต์แวร์

องค์ประกอบวิศวกรรมซอฟต์แวร์แบ่งออกเป็น 2 ส่วน ดังนี้

ส่วนที่ 1 วิศวกรรมระบบ (System Engineering) หมายถึง กระบวนการศึกษาและวิเคราะห์ของระบบที่มีความสลับซับซ้อน เพื่อสนับสนุนการทำงานใน

ส่วนที่ 2 วิศวกรรมการผลิต (Development Engineering) ซึ่งเป็นกระบวนการแปรสภาพความต้องการของระบบ (System Requirements) ให้กลายเป็นซอฟต์แวร์อันเป็นเป้าหมายสำคัญทางด้านวิศวกรรมซอฟต์แวร์



• รูปแสดงองค์ประกอบของวิศวกรรมซอฟต์แวร์

วิศวกรรมระบบ มีหน้าที่ ดังนี้

1. กำหนดวัตถุประสงค์ของระบบ
2. กำหนดขอบเขตของระบบ
3. แบ่งระบบออกเป็นส่วน ๆ ตามฟังก์ชันงานหรือคุณสมบัติของระบบ
4. พิจารณาความสัมพันธ์ของส่วนประกอบต่าง ๆ ที่เกี่ยวข้องทั้งหมด
5. กำหนดความสัมพันธ์ของปัจจัยนำเข้า ประมวลผล และผลลัพธ์
6. พิจารณาปัจจัยที่มีส่วนเกี่ยวข้องในระบบ
7. กำหนดความต้องการในส่วนของปฏิบัติการ (Operation) และฟังก์ชันงาน (Function) ทั้งระบบ

8. สร้างแบบจำลองระบบ (System Model) เพื่อใช้วิเคราะห์และพัฒนาให้สอดคล้องกับแบบจำลองซอฟต์แวร์ (Software Model) ที่สร้างขึ้น

9. นำเสนอและแลกเปลี่ยนข้อคิดเห็นกับผู้ที่เกี่ยวข้องกับระบบ
   - ผู้ใช้ระบบ
   - เจ้าของระบบ
   - ผู้ที่เกี่ยวข้องกับผลประโยชน์ที่มีต่อระบบ

วิศวกรรมการผลิต มีหน้าที่ ดังนี้

1. กำหนดความต้องการและจัดทำข้อกำหนดคุณสมบัติ
2. ออกแบบแนวทางแก้ปัญหาให้สอดคล้องกับความต้องการ
3. พิจารณาสถาปัตยกรรมให้สอดคล้องกับแนวทางแก้ปัญหา
4. วางแผนโครงการผลิตซอฟต์แวร์
5. ทดสอบซอฟต์แวร์ในแต่ละคอมโพเน้นท์
6. ผนวกคอมโพเน้นต่าง ๆ รวมเป็นระบบเดียวกัน

7. ทดสอบการผนวกรวมระบบ พร้อมตรวจสอบความถูกต้องและความสอดคล้องกับความต้องการที่ได้กำหนดไว้

8. พิจารณากลยุทธ์ในการนำไปใช้งาน

9. นำไปใช้งาน

10. ปรับเปลี่ยนกระบวนการจัดการ

11. บำรุงรักษาและติดตั้งซอฟต์แวร์

วิวัฒนาการของวิศวกรรมซอฟต์แวร์

วิวัฒนาการแบ่งระยะเวลาออกเป็น 3 ช่วง ดังนี้

1. ช่วงระหว่างปี ค.ศ. 1945 ถึง 1965 : จุดเริ่มต้นของวิศวกรรมซอฟต์แวร์
   - นำไปใช้งานครั้งแรกราวปี ค.ศ. 1950 ถึงต้นปี ค.ศ. 1960 การผลิตมุ่งเน้นที่ซอฟต์แวร์เป็นสำคัญ
   - North Atlantic Treaty Organization (นาโต / NATO) จัดสัมมนาวิศวกรรมซอฟต์แวร์สองครั้ง ครั้งแรกปี ค.ศ. 1968 และปี ค.ศ. 1969 ณ ประเทศเยอรมัน ได้รับการยอมรับอย่างเป็นทางการ

2. ช่วงระหว่างปี ค.ศ. 1965 ถึง 1985 : วิกฤติซอฟต์แวร์

- จุดวิกฤติในช่วงปี ค.ศ. 1960 ถึงปี ค.ศ. 1980
- กรณีตัวอย่าง
  - ซอฟต์แวร์ระบบปฏิบัติการ OS/360
  - ซอฟต์แวร์ระบบรักษาความปลอดภัยของฐานจรวดนำวิถีแอร์เรียน (Ariane)
  - ซอฟต์แวร์ระบบควบคุมการแผ่รังสีสำหรับเครื่องรักษาผู้ป่วยด้วยรังสีวิทยา

ปีเตอร์ จี นูมัน (Peter G. Neumann) รายงานความเสียหายที่เกิดจากซอฟต์แวร์ต่อคณะกรรมาธิการความเสี่ยง (Risk Committee) โดยสรุปได้ดังนี้

- อุตสาหกรรมซอฟต์แวร์เกิดฟองสบู่แตก
- การผลิตซอฟต์แวร์เกิดการชะลอตัว
- แนวทางการพัฒนาซอฟต์แวร์ตลอดระยะเวลา 20 ปีที่ผ่านมา ยังไม่สามารถนำมาใช้งานได้จริงกับการทำงาน
- วิศวกรซอฟต์แวร์ต่างตระหนักและให้การยอมรับ

3. ช่วงระหว่างปี ค.ศ. 1985 ถึงปัจจุบัน : ฟองสบู่แตก

- ยุคของการแก้ปัญหาวิกฤติซอฟต์แวร์อย่างแท้จริง โดยมีปัจจัยที่เป็นแรงขับเคลื่อนดังนี้
  - เครื่องมือ (Tool)
  - สหวิทยาการ (Discipline)
  - วิธีการที่ถูกแบบแผน (Formal Method)
  - กระบวนการ (Process)
  - ความเป็นมืออาชีพ

ปี ค.ศ. 1987 เฟรด บรูคส์ (Fred Brooks) ได้เผยแพร่ผลงานในบทความเรื่อง “No Silver Bullet” โดยมีใจความว่า ยังไม่มีเทคโนโลยีหรือแนวทางปฏิบัติใดตลอดเวลา 10 ปี ที่เป็นเครื่องบ่งชี้ถึงวิธีการปรับปรุงเพื่อเพิ่มผลผลิตและคุณภาพอย่างมีประสิทธิผลได้อย่างแท้จริง

คุณลักษณะของซอฟต์แวร์ที่มีคุณภาพ

คุณลักษณะของซอฟแวร์

1. ความสามารถในการบำรุงรักษา (Maintainability)
   - ง่ายต่อการบำรุงรักษา
   - สามารถเปลี่ยนแปลง (Change)
   - ปรับเปลี่ยนให้เหมาะสม (Adaptive)
   - ตอบสนอง (Response) ได้อย่างรวดเร็วและทันท่วงที
2. ความสามารถในการพึ่งพา (Dependability)
   - ความน่าเชื่อถือ (Reliability)
   - ผ่านการทวนสอบและตรวจรับ (Verification and Validation)

3. ประสิทธิภาพ (Efficiency)
   - ประหยัด หรือสิ้นเปลืองน้อยที่สุด
   - ใช้ทรัพยากรต่าง ๆ ได้อย่างคุ้มค่า
4. ความสามารถในการใช้งาน (Usability)
   - สะดวก และง่ายต่อการใช้งาน
   - เสริมสร้างการเรียนรู้ได้อย่างรวดเร็ว
5. วิธีการวัดผลหรือประเมินผลจากปัจจัยด้านอื่น ๆ เช่น
   - การประเมินผลความพึงพอใจของลูกค้า (Customer Satisfaction)
   - การคำนวณต้นทุน และงบประมาณการดำเนินการ (Cost and Budget)
   - กระตรวจสอบและประกันคุณภาพของซอฟต์แวร์ทางวิศวกรรม

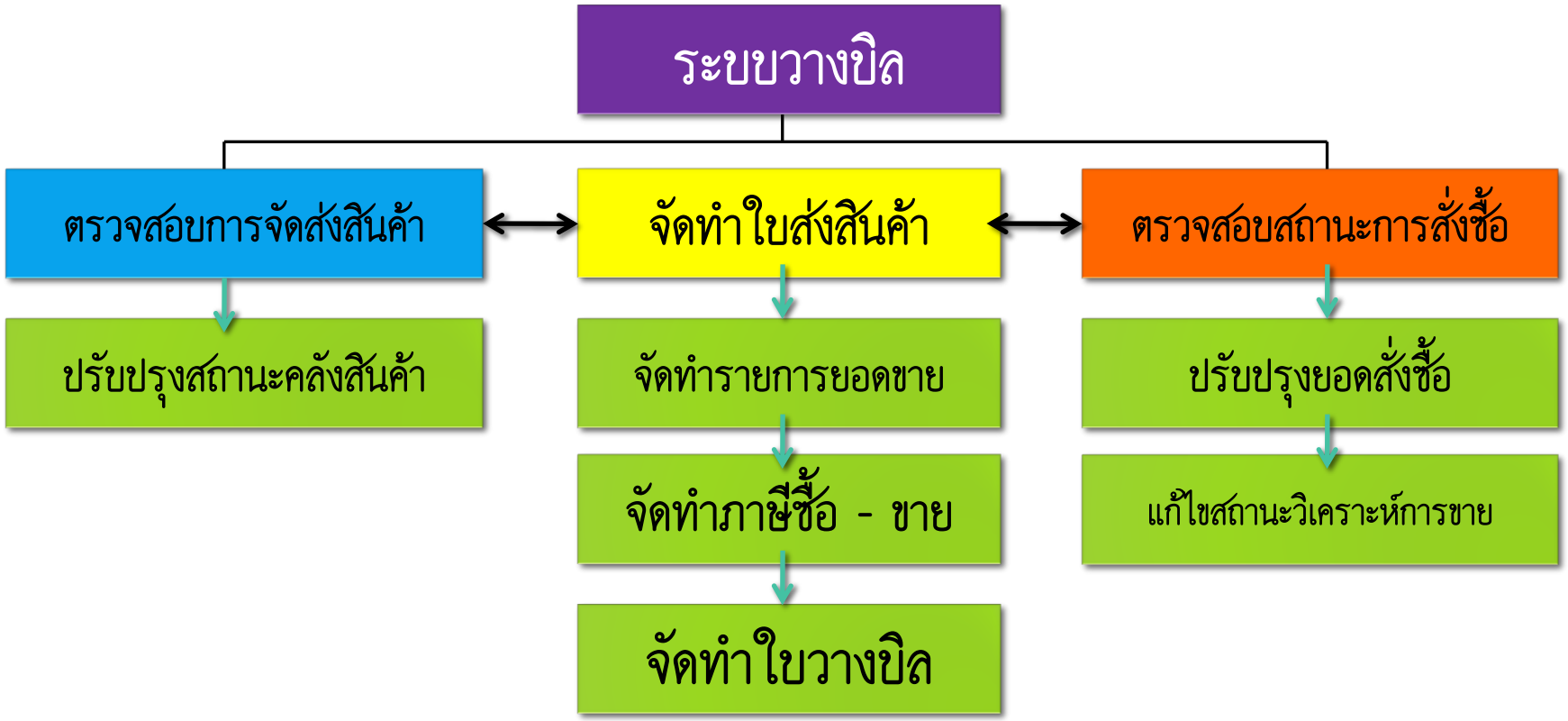
ระเบียบวิธีปฏิบัติของวิศวกรรมซอฟต์แวร์

- วิศวกรรมซอฟต์แวร์ เป็นงานที่แทรกซึมอยู่ในทุกขั้นตอนของกระบวนการผลิตซอฟต์แวร์
- ระเบียบวิธีปฏิบัติของวิศวกรรมซอฟต์แวร์ (Software Engineering Methodology) จึงเป็นไปตามแนวทางการพัฒนาซอฟต์แวร์ (Software Development Approach) มีสองแนวทาง คือ
  - แนวทางเชิงโครงสร้าง
  - แนวทางเชิงวัตถุ

1. แนวทางเชิงโครงสร้าง (Structured Approach)

- แบ่งระบบและความต้องการออกเป็นระบบย่อย (Sub-System)
- ลักษณะของระบบจึงเป็นโครงสร้างแบบลำดับชั้น
- ระเบียบวิธีปฏิบัติชนิดหนึ่งที่นิยมนำมาใช้ในขั้นตอนการวิเคราะห์และออกแบบระบบ คือ “การวิเคราะห์และออกแบบระบบเชิงโครงสร้าง (Structured System Analysis and Design: SSAD)” คิดค้นโดย Yourdan & Demarco ปี ค.ศ. 1978
- ข้อเสีย
  - ต้องวิเคราะห์และออกแบบข้อมูลรวมถึงพฤติกรรมของระบบแยกกันคนละส่วน ทำให้ต้องใช้เวลานาน
  - ใช้ต้นทุนมากเกินไป
  - เสี่ยงต่อการเปลี่ยนแปลงความต้องการของผู้ใช้

ตัวอย่าง การวิเคราะห์และออกแบบระบบเชิงโครงสร้าง



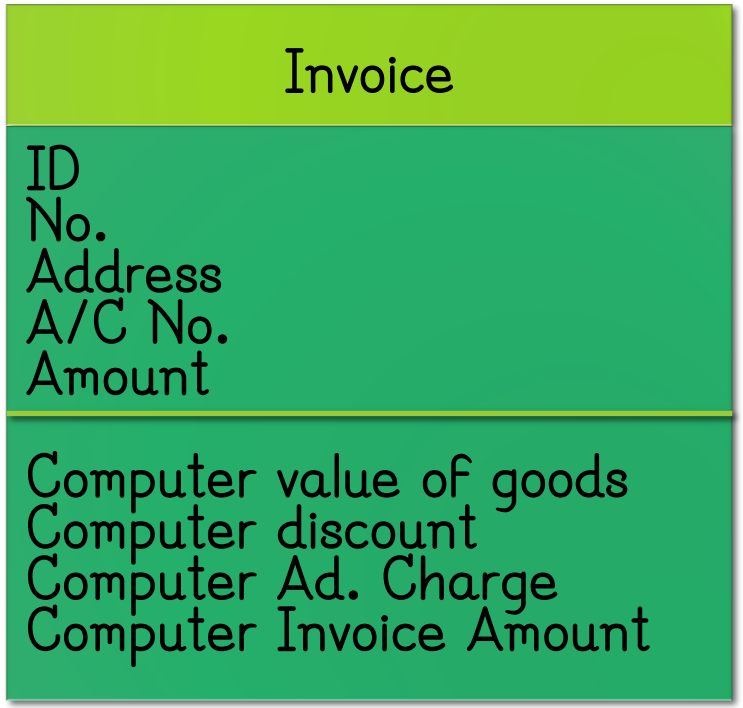
2. แนวทางเชิงวัตถุ (Object - Oriented Approach)

- คิดค้นโดย Grady Booch, James Rumbaugh และ Ivar Jacobson
- การวิเคราะห์และออกแบบระบบเชิงวัตถุ (Object-Oriented System Analysis and Design)
- เป็นการวิเคราะห์ระบบโดยการมองทุกอย่างในระบบเป็นอ็อบเจ็กต์ (Object)
- ภายในอ็อบเจ็กต์ จะมีส่วนข้อมูลและพฤติกรรมของระบบ

ข้อดี

1. การวิเคราะห์และออกแบบระบบรวดเร็ว
2. รองรับกับระบบงานที่มีความซับซ้อนสูง
3. ทันต่อการเปลี่ยนแปลงความต้องการของผู้ใช้

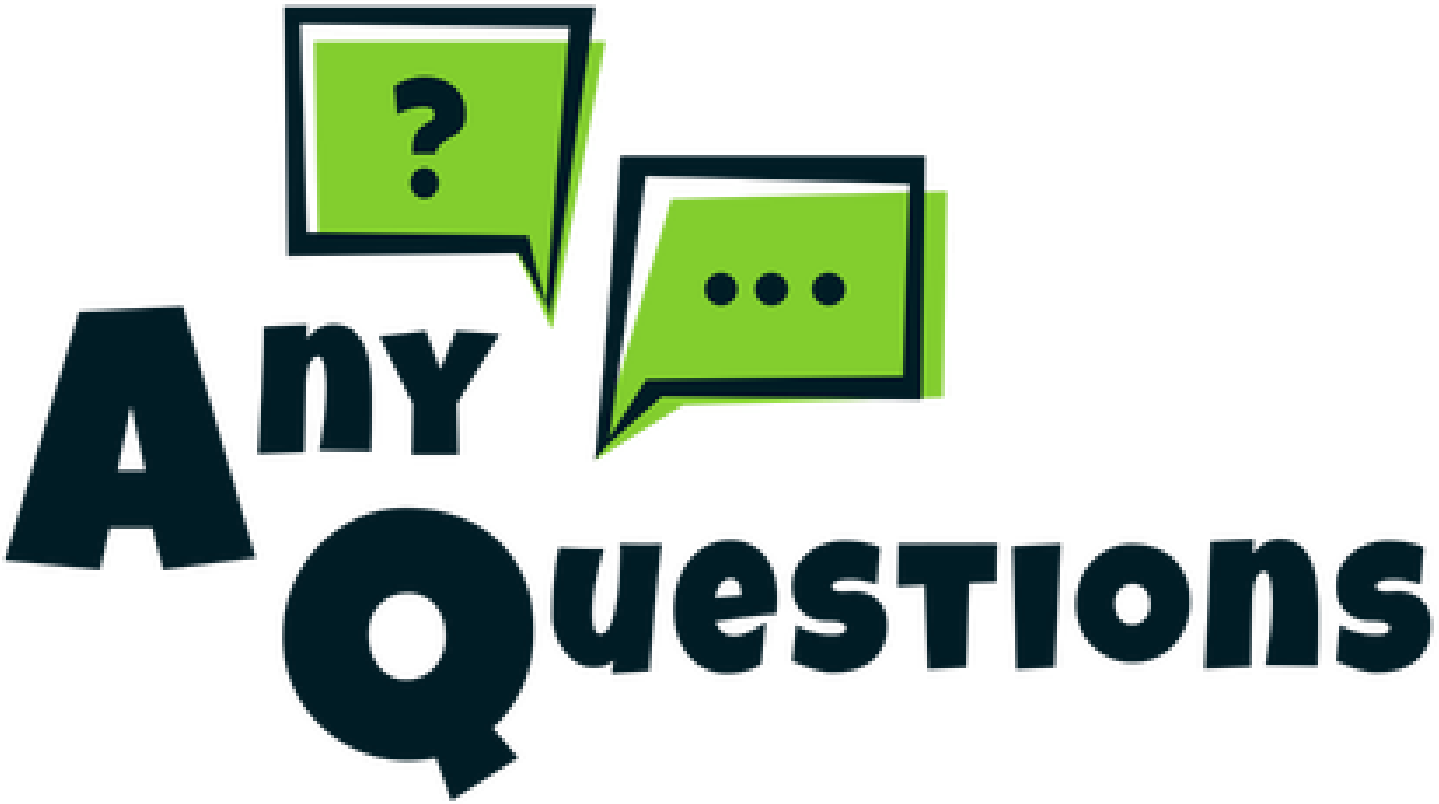
ตัวอย่างอ็อบเจ็กต์



object

Attributes

Methods


Any Questions