Software Engineering

วิศวกรรมซอฟต์แวร์ (Software Engineering) :

เป็นศาสตร์เกี่ยวกับวิศวกรรมด้านซอฟต์แวร์ มีเนื้อหาเกี่ยวข้องกับการใช้กระบวนการทางวิศวกรรมในการดูแลการผลิต ตั้งแต่การเริ่มเก็บความต้องการ การตั้งเป้าหมายของระบบ การออกแบบกระบวนการพัฒนา การตรวจสอบ การประเมินผล การติดตามโครงการ การประเมินต้นทุน การรักษาความปลอดภัย ไปจนถึงการคิดราคาซอฟต์แวร์เป็นต้น

วิศวกรรมซอฟต์แวร์ประยุกต์ความรู้และเทคโนโลยีทางด้านวิศวกรรมศาสตร์ วิศวกรรมคอมพิวเตอร์ วิทยาการคอมพิวเตอร์ เทคโนโลยีสารสนเทศ การบริหารจัดการโครงการ และสาขาอื่น ๆ ที่เกี่ยวข้องเข้าด้วยกัน เพื่อสร้างซอฟต์แวร์ที่สามารถปฏิบัติงานตามเป้าหมาย ภายใต้เงื่อนไขที่กำหนด

Software Engineering

มุมมองทางการศึกษาในแง่ของสาขาวิชา

ในปี ค.ศ. 1968 คำว่า"*วิศวกรรมซอฟต์แวร์* (software engineering)" ถูกใช้อย่างแพร่หลายเพื่อแสดงถึงกิจกรรมต่างๆที่รวมถึงการเขียนโปรแกรม (programming) และการรหัส (coding) [Macro, 1987]. ก่อนปี ค.ศ. 1974 สาขาวิชาวิศวกรรมซอฟต์แวร์ยังไม่ปรากฏ [Barnes, 1998]. สถาบันเทคโนโลยีโรเชสเตอร์ (The Rochester Institute of Technology (RIT)) ในประเทศสหรัฐอเมริกาได้อ้างว่าเป็นสถาบันแรกที่แนะนำหลักสูตรปริญญาตรีสาขาวิศวกรรมซอฟต์แวร์ [Lutz, 1999].

Software Engineering

ความหมายของวิศวกรรมซอฟต์แวร์

- วิศวกรรมซอฟต์แวร์ คือกระบวนการสร้างสรรค์โปรแกรมโดยใช้หลักทางวิศวกรรมเข้ามาช่วยในการดำเนินการสร้าง (อ.สมหมาย สุขคำ)
- "Software Engineering is systematic approach to the development operation , maintenance , retirement of software" (IEEE)
- "วิชาการว่าด้วยการออกแบบโปรแกรมคอมพิวเตอร์ ตลอดจนการบริหารงานการพัฒนาเพื่อที่จะได้มาซึ่ง ผลิตผลซอฟต์แวร์ที่มีคุณภาพสูง ราคาถูก และภายในเวลาที่กำหนดให้" (สุชาย ธนวเสถียร)

ความแตกต่างของวิทยาการคอมพิวเตอร์และวิศวกรรมซอฟต์แวร์

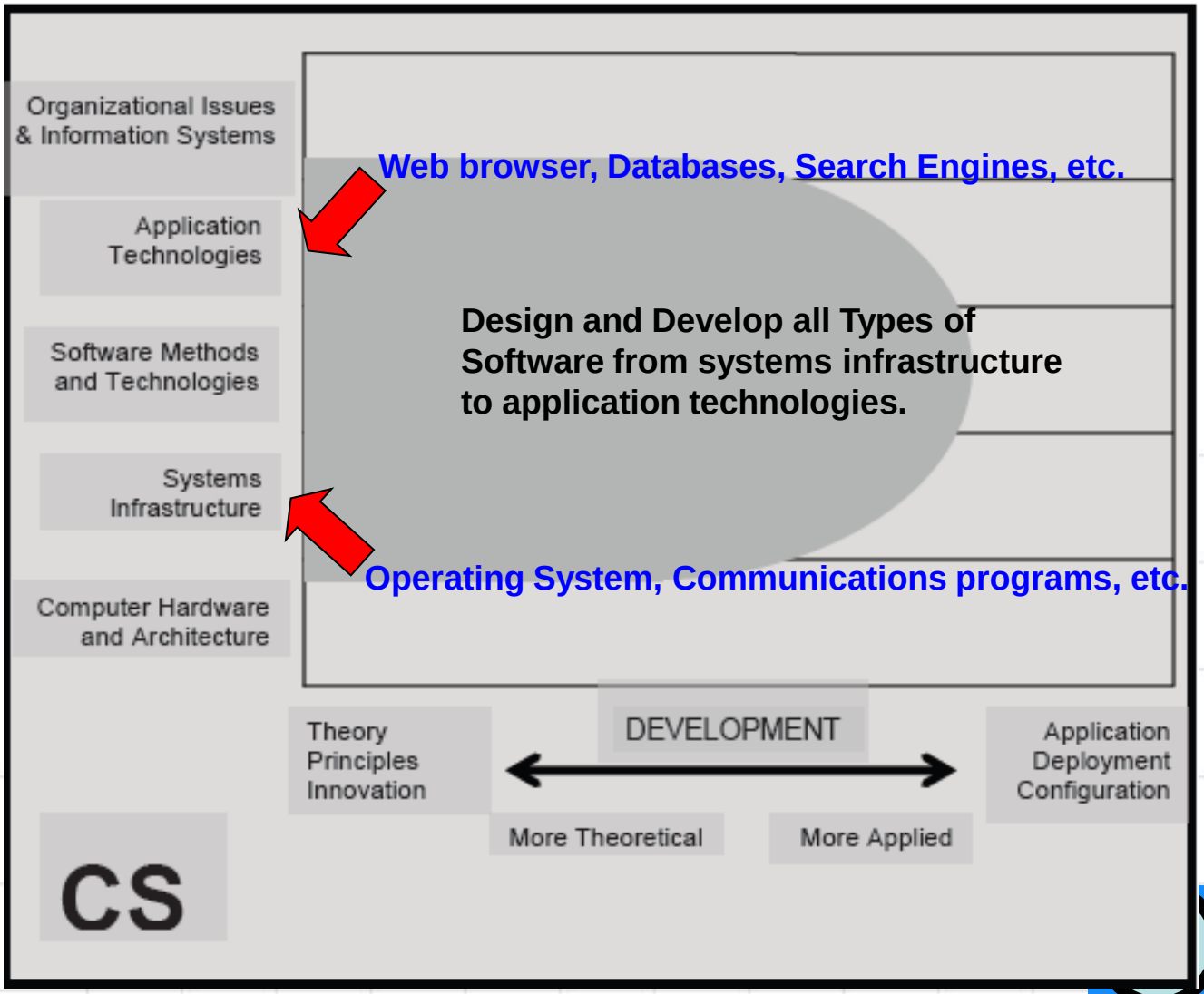
วิทยาการคอมพิวเตอร์ (Computer Science)

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

* ผลงานถูกพิจารณา หรือ ตัดสินโดยกลุ่มนักวิทยาศาสตร์

วิทยาการคอมพิวเตอร์
Computer Science
(CS)

(เรียนรู้ทุกสิ่งที่เกี่ยวข้องกับวงการนี้)



ความแตกต่างของวิทยาการคอมพิวเตอร์และวิศวกรรมซอฟต์แวร์

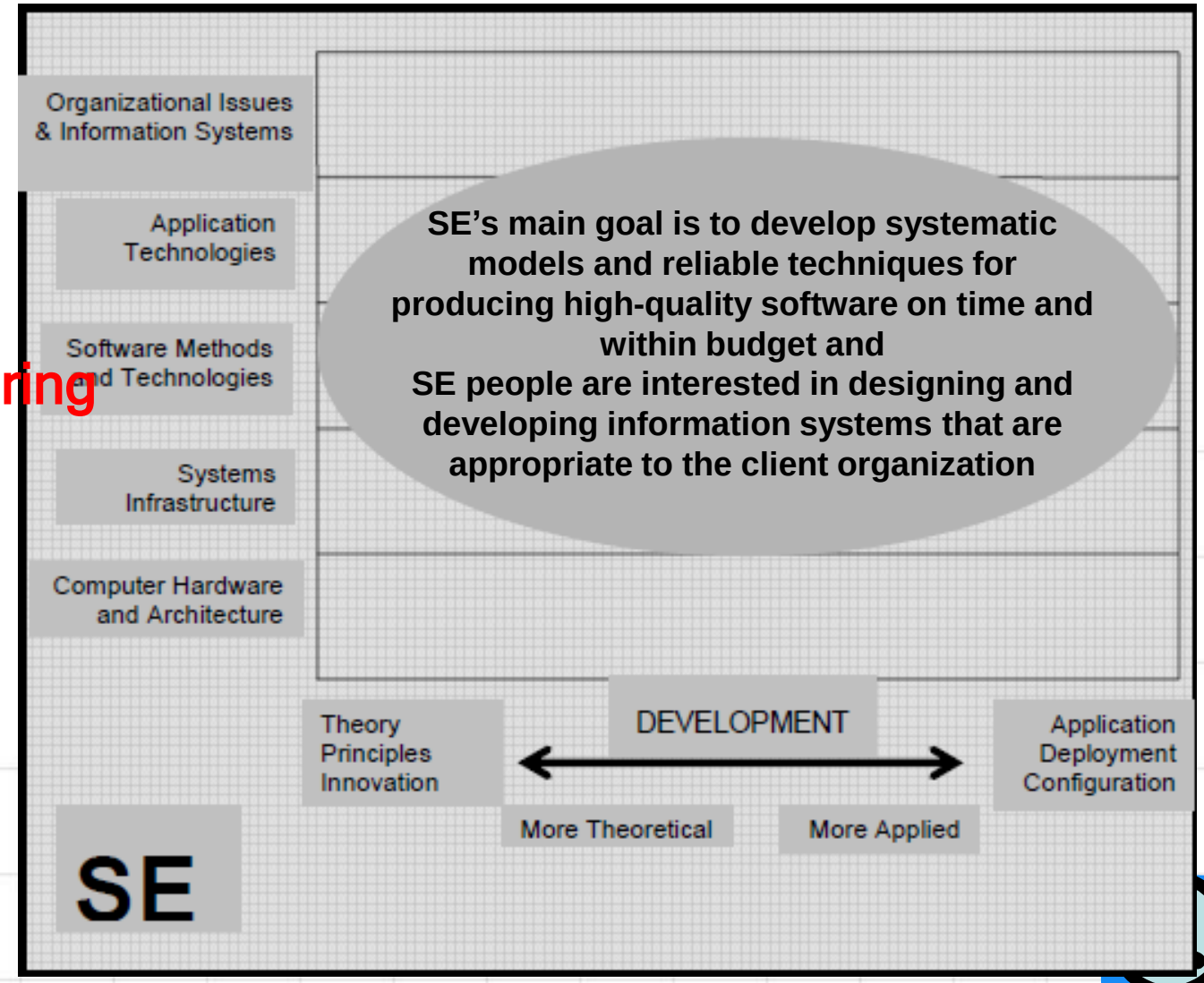
วิศวกรรมซอฟต์แวร์ (Software Engineering)

อยู่บนรากฐานของวิธีการทางวิศวกรรมศาสตร์ ซึ่งประยุกต์แนวคิด/ทฤษฎีทางวิทยาศาสตร์ คณิตศาสตร์และเทคโนโลยีขณะนั้นในการสร้างผลิตภัณฑ์ที่เป็นประโยชน์และปลอดภัยต่อสาธารณะ

* ผลงานถูกพิจารณา หรือ ตัดสินโดยกลุ่มผู้ใช้

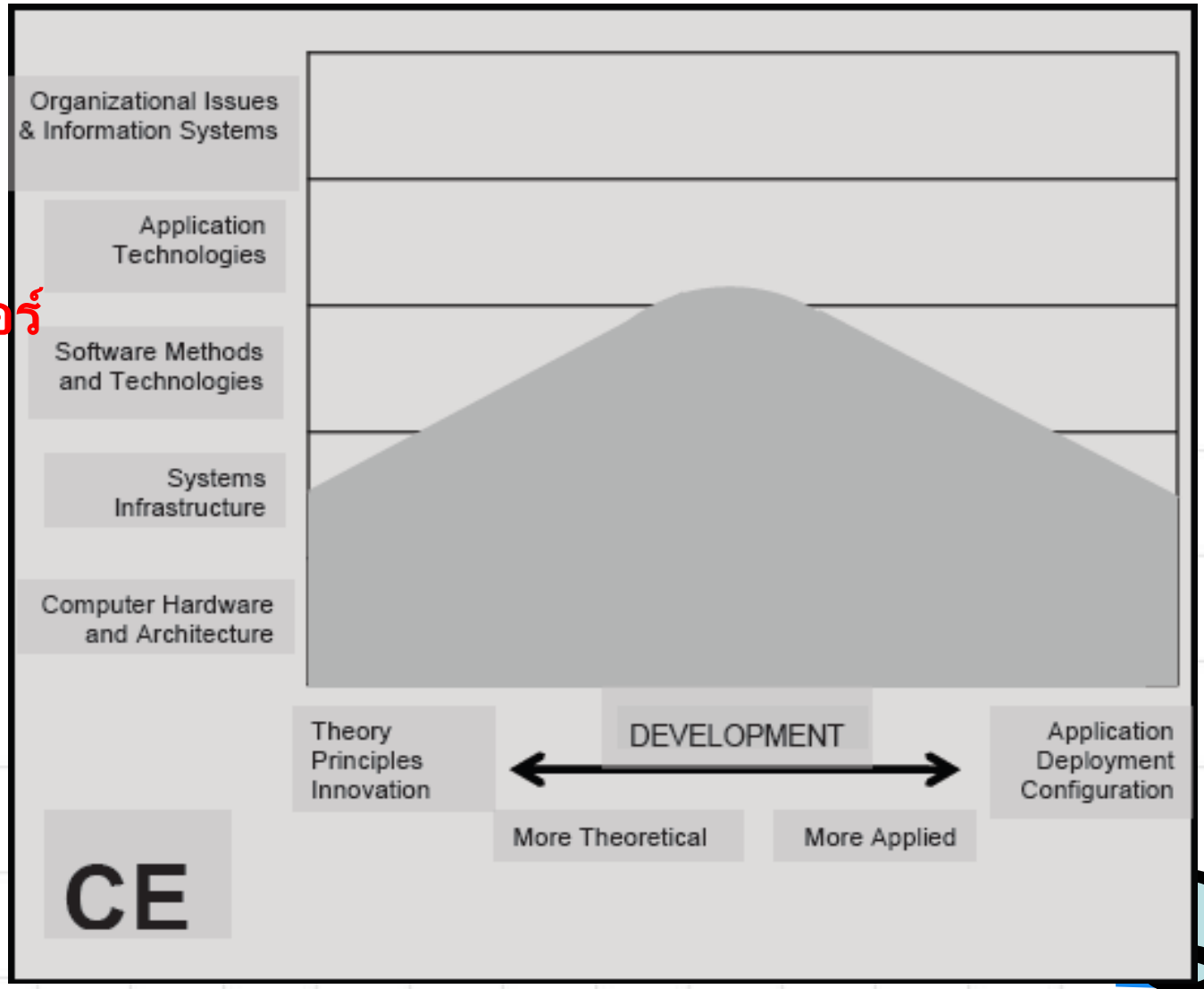
Software Engineering

วิศวกรรมซอฟต์แวร์
Software Engineering
(SE)

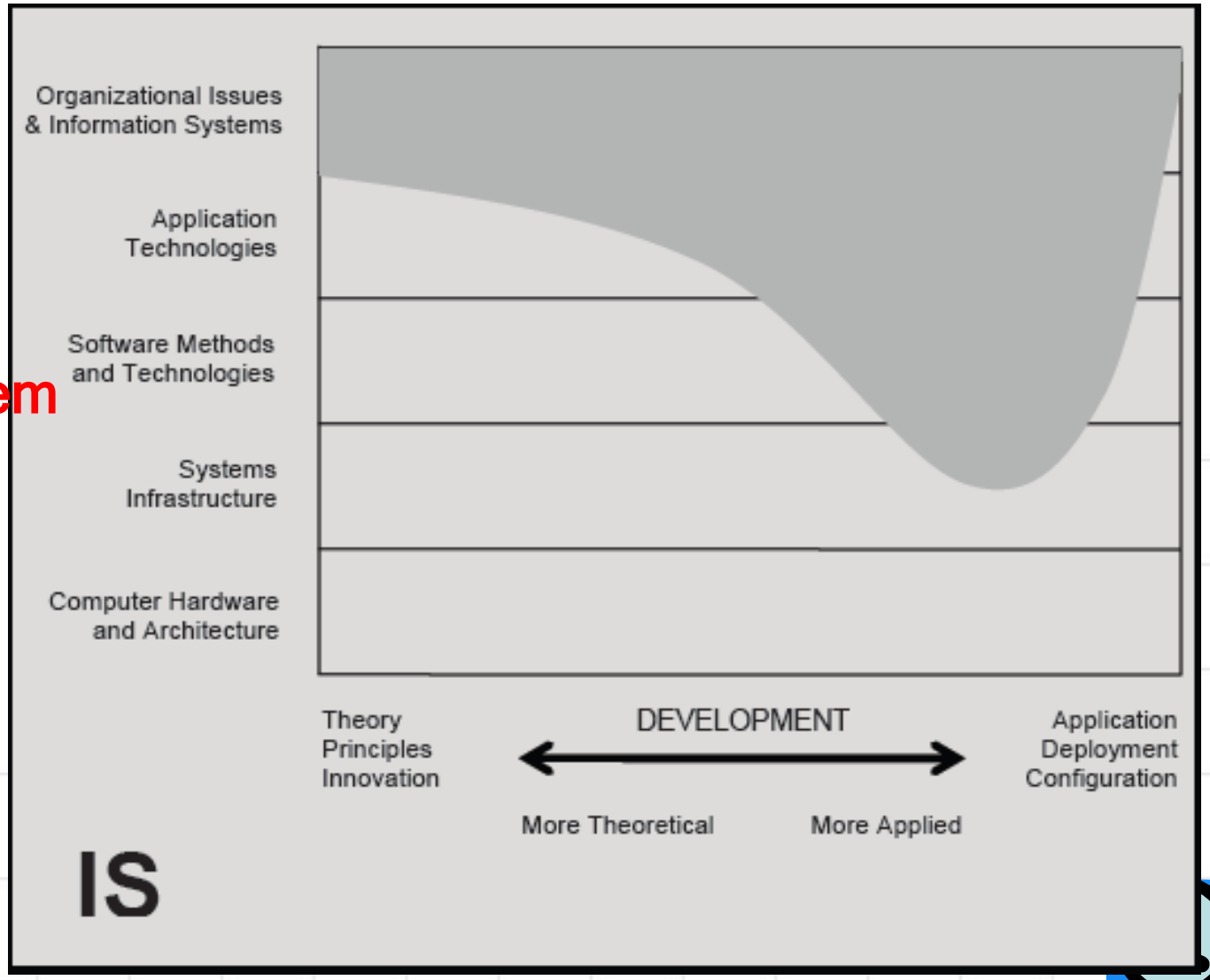


Software Engineering

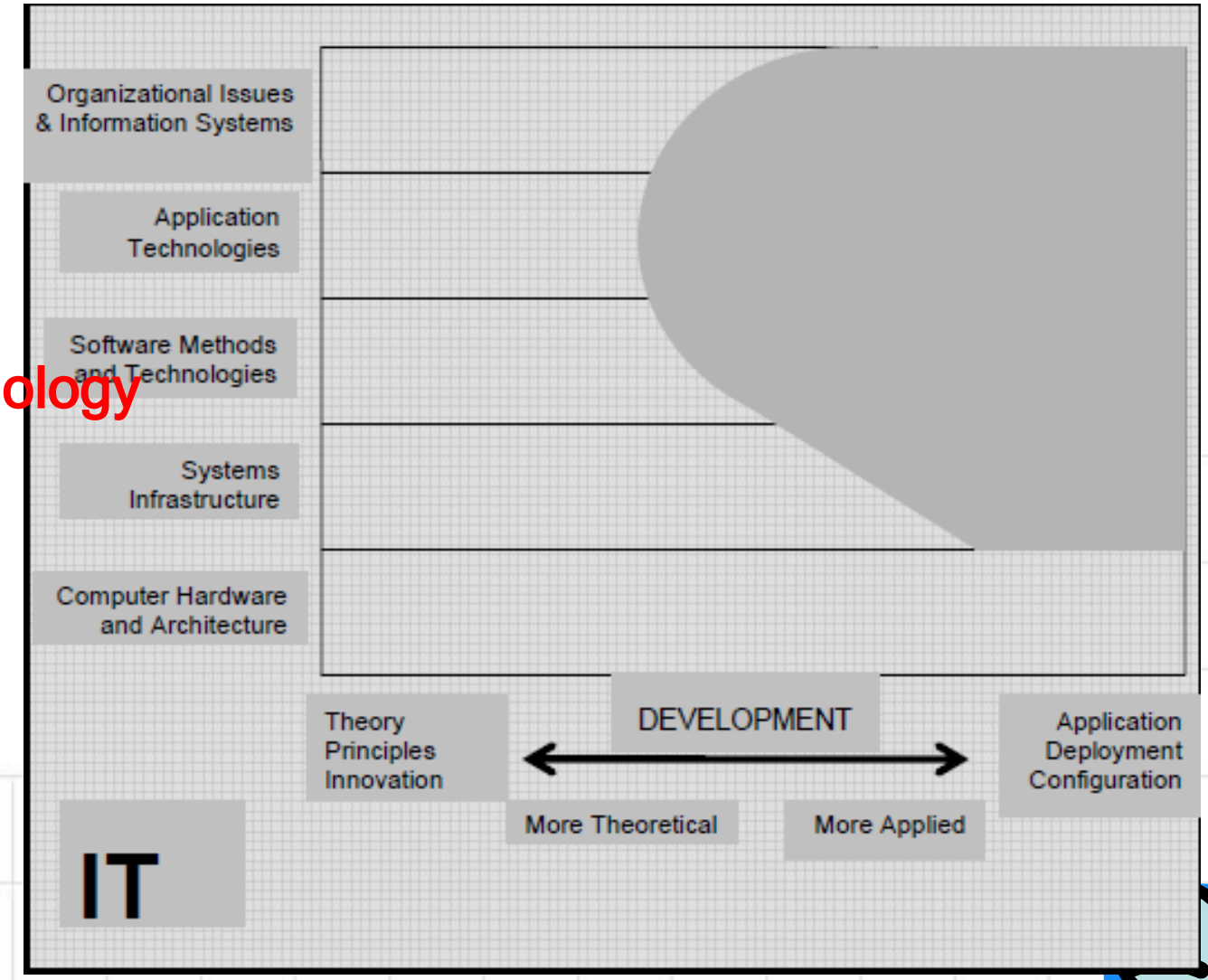
วิศวกรรมคอมพิวเตอร์
Computer Engineering (CE)



ระบบสารสนเทศ
Information System
(IS)



เทคโนโลยีสารสนเทศ
Information Technology
(IT)

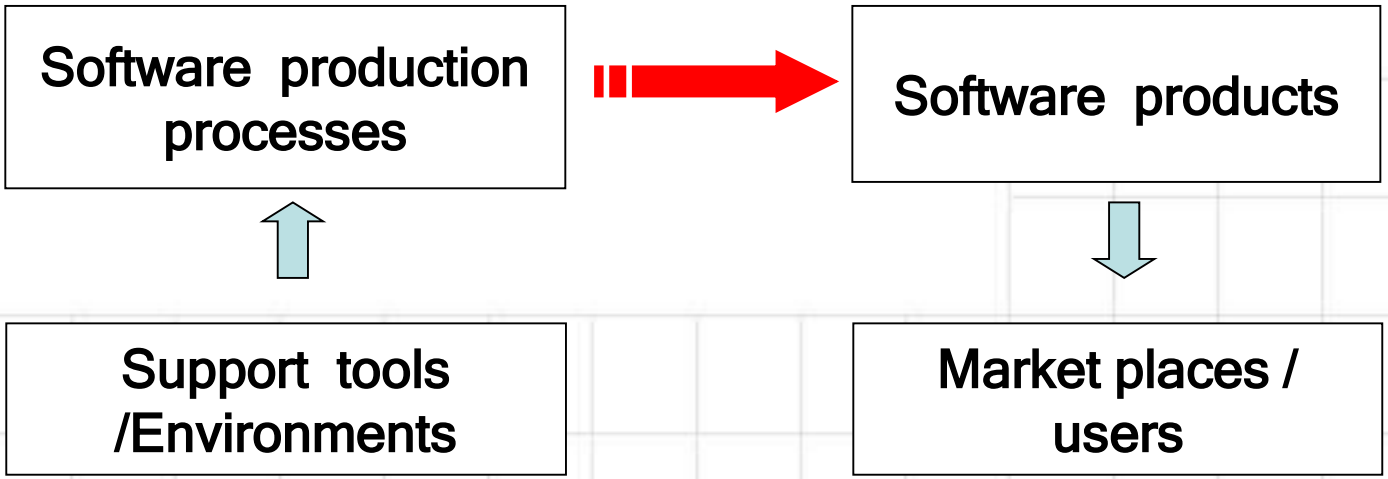


ลักษณะของวิศวกรรมซอฟต์แวร์

- เกี่ยวข้องกับการสร้างโปรแกรมขนาดใหญ่
- สามารถจัดการเกี่ยวกับความซ้ำซ้อนได้
- เน้นการทำงานร่วมกันของบุคลากร
- สามารถเปลี่ยนแปลงได้ง่ายเมื่อจำเป็น
- เน้นการพัฒนาให้มีประสิทธิภาพ
- สนองความต้องการของผู้ใช้

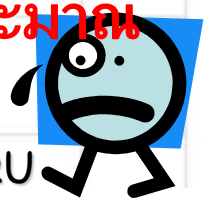
องค์ประกอบของการวิศวกรรมซอฟต์แวร์

การวิศวกรรมซอฟต์แวร์เป็นกระบวนการผลิต (production)ที่ประกอบด้วยกิจกรรมช่วงต่างๆ เพื่อสร้างผลิตภัณฑ์ซอฟต์แวร์ (software products) การทำกิจกรรมในแต่ละช่วงอาศัยเทคนิคและเครื่องมือช่วยต่างๆ (support tools) ที่นักวิชาการคอมพิวเตอร์และนักวิจัยได้เสนอไว้



Software Engineering

- กระบวนการ หมายถึง ขั้นตอนต่าง ๆ ที่เกี่ยวข้องกัน และนำไปปฏิบัติเพื่อให้บรรลุวัตถุประสงค์ที่กำหนดไว้
- กระบวนการมีความหมายรวมถึง ทรัพยากร เช่น คน , วิธีการ และเครื่องมือที่จำเป็นสำหรับนำไปใช้ปฏิบัติตามขั้นตอนวิธีการที่กำหนดไว้ในกระบวนการด้วย
- กระบวนการที่ดีย่อมสามารถปฏิบัติซ้ำ และได้ผลลัพธ์แบบเดียวกันเสมอ
- กระบวนการวิศวกรรมซอฟต์แวร์ ช่วยให้เห็นกิจกรรมต่าง ๆที่จำเป็นสำหรับการพัฒนา SW ที่กำหนดให้ ตั้งแต่ต้นจนจบและประสบผลสำเร็จดี
- กระบวนการดังกล่าวไม่ใช่ SDLC (System Development Life Cycle) ที่นักพัฒนารู้จัก เพราะ SDLC มีแต่เพียงเฟส (phase) สำคัญเท่านั้น และ SDLC ก็เน้นแต่เพียงกิจกรรมที่เกี่ยวข้องกับการพัฒนาเท่านั้น ไม่ได้กล่าวถึงกิจกรรมอื่น ๆ เช่น การวางแผน , การประมาณการ , การสอบทานผลงาน เป็นต้น



Software Engineering

- ช่างตัดสูทต้องรู้จักกระบวนการทุกอย่างตั้งแต่
  - รู้จักลักษณะของผ้า
  - รู้จักลักษณะของสูทที่ลูกค้าต้องการ
  - รู้จักการประมาณราคาและเวลาที่จะทำสำเร็จ
  - รู้จักการวัดตัว
  - รู้จักการสร้างแบบสูทที่พอดีกับขนาดรูปร่างของลูกค้า
  - รู้จักการตัดผ้าออกเป็นชั้นส่วนต่าง ๆ
  - รู้จักการเย็บชิ้นส่วนของผ้าให้เป็นชุด
  - รู้จักการประมาณเวลาที่ลูกค้ามาลองสวมใส่ และ สามารถแก้ไขได้
  - รู้จักการเย็บทุกขั้นตอนจนสำเร็จ

Software Engineering

- วิศวกรโยธา ต้องรู้จักกระบวนการทุกอย่างตั้งแต่
  - รู้จักลักษณะของดิน หิน กรวด ทราย และวัสดุที่เกี่ยวข้อง
  - รู้จักลักษณะของเทคโนโลยีสนับสนุน และมั่นคงความปลอดภัย
  - รู้จักการประมาณราคาและเวลาที่จะทำสำเร็จ
  - รู้จักรูปแบบการสร้างที่เหมาะกับความต้องการลูกค้า
  - รู้จักการวิธีการออกแบบ/โครงแบบ/โครงสร้าง
  - รู้จักการวางแผนการสร้างเป็นเฟสต่าง ๆ
  - รู้จักการควบคุมงาน เชื่อมต่อแต่ละเฟสงานอย่างเป็นระบบ
  - รู้จักการตรวจสอบ ทวนสอบ สามารถแก้ไขได้
  - รู้จักการตรวจสอบทุกกระบวนการจนสำเร็จ

ลักษณะของกระบวนการวิศวกรรมซอฟต์แวร์

- **Understandability** : มีการนิยามขอบเขตของกระบวนการที่ชัดแจ้งและง่ายต่อการเข้าใจ
- **Visibility** : ทำให้กิจกรรมกระบวนการชัดเจนที่สุดเพื่อสามารถมองเห็นจากภายนอกได้ชัดเจน
- **Supportability** : เครื่องมือช่วยการวิศวกรรมซอฟต์แวร์ (CASE) สามารถช่วยสนับสนุนกิจกรรมกระบวนการในขอบเขตใด
- **Acceptability** : กระบวนการที่กำหนดสามารถยอมรับและใช้โดยวิศวกรซอฟต์แวร์ในการผลิตผลิตภัณฑ์ซอฟต์แวร์
- **Reliability** : กระบวนการถูกออกแบบในแนวทางซึ่งความผิดพลาดของกระบวนการถูกหลีกเลี่ยงก่อนที่จะส่งผลต่อความผิดพลาดของผลิตภัณฑ์ซอฟต์แวร์

ลักษณะของกระบวนการวิศวกรรมซอฟต์แวร์

- **Robustness** : กระบวนการสามารถทำงานต่อได้แม้ว่ามีปัญหาที่ไม่คาดการณ์เกิดขึ้น
- **Maintainability** : กระบวนการสามารถวิวัฒนาการเพื่อตอบสนองการเปลี่ยนแปลงความต้องการขององค์กร
- **Rapidity** : กระบวนการสามารถทำให้ส่งมอบผลิตภัณฑ์ได้เร็วนับจากที่รูปแบบคุณลักษณะของซอฟต์แวร์ (Software specifications) ถูกกำหนด

Introduction

นิยามของซอฟต์แวร์

(1) ซอฟต์แวร์ หมายรวมถึงโปรแกรมคอมพิวเตอร์ทุกประเภทที่ทำให้คอมพิวเตอร์ทำงานได้

(2) ซอฟต์แวร์ (Software) หมายถึง ชุดคำสั่ง หรือโปรแกรมที่บอกให้ส่วนต่าง ๆ ของคอมพิวเตอร์ทำงาน (Telling the machine what to do)

(3) ซอฟต์แวร์ หมายความรวมไปถึงการควบคุมการทำงานของอุปกรณ์แวดล้อมต่างๆ เช่น Modem, CD ROM, Drive เป็นต้น

(4) ซอฟต์แวร์ เป็นสิ่งที่มองไม่เห็นจับต้องไม่ได้ แต่รับรู้การทำงานของมันได้ ซึ่งต่างกับ ฮาร์ดแวร์ (Hardware) ที่สามารถจับต้องได้

Introduction

บทบาทของซอฟต์แวร์ :

1. **ซอฟต์แวร์คือตัวผลิตภัณฑ์**
   ลักษณะที่ดึงความสามารถทางการประมวลผลของฮาร์ดแวร์คอมพิวเตอร์ออกมา เพื่อให้สามารถดำเนินงานหนึ่ง ๆ โดยตัวมันจะผลิต จัดการ จัดหา ปรับเปลี่ยน แสดงผล หรือส่งต่อข้อมูล

2. **ซอฟต์แวร์คือเครื่องมือที่ก่อให้เกิดผลิตภัณฑ์**
   ลักษณะที่ก่อให้เกิดผลิตภัณฑ์ โดยจะทำหน้าที่เหมือนกับเป็นตัวควบคุมคอมพิวเตอร์ ระบบสารสนเทศ หรือเป็นสิ่งที่ช่วยในการสร้างและควบคุมโปรแกรมอื่น ๆ ซึ่งผลิตภัณฑ์ที่มีคุณค่ามากที่สุดที่มาจากซอฟต์แวร์เหล่านี้คือ ข้อมูลสารสนเทศ (information)

Introduction

บทบาทของซอฟต์แวร์ :

ในปัจจุบันอุตสาหกรรมด้านซอฟต์แวร์มีขนาดใหญ่ การสร้างผลิตภัณฑ์หนึ่งๆ อาจเปลี่ยนจากโปรแกรมเมอร์คนเดียว -> ทีมผู้เชี่ยวชาญด้านซอฟต์แวร์โดยมุ่งเน้นไปที่เทคโนโลยี อย่างไรก็ตามยังคงคำถามที่มักเกิดขึ้น ได้แก่

- ทำไมการสร้างซอฟต์แวร์จึงใช้เวลานาน
- ทำไมต้นทุนในการพัฒนาซอฟต์แวร์จึงสูง
- ทำไมไม่สามารถหาข้อผิดพลาดเจอทั้งหมด ก่อนส่งมอบลูกค้า
- ทำไมต้องใช้เวลาและความพยายามในการบำรุงรักษาซอฟต์แวร์
- ทำไมการประเมินความก้าวหน้าในการสร้างและบำรุงรักษาจึงเป็นเรื่องยาก

Introduction

คุณสมบัติของซอฟต์แวร์ :

(1) ซอฟต์แวร์ถูกพัฒนาหรือจัดการให้เกิดขึ้น ไม่ได้เกิดจากการผลิตในโรงงานหรือสินค้าทั่วไป

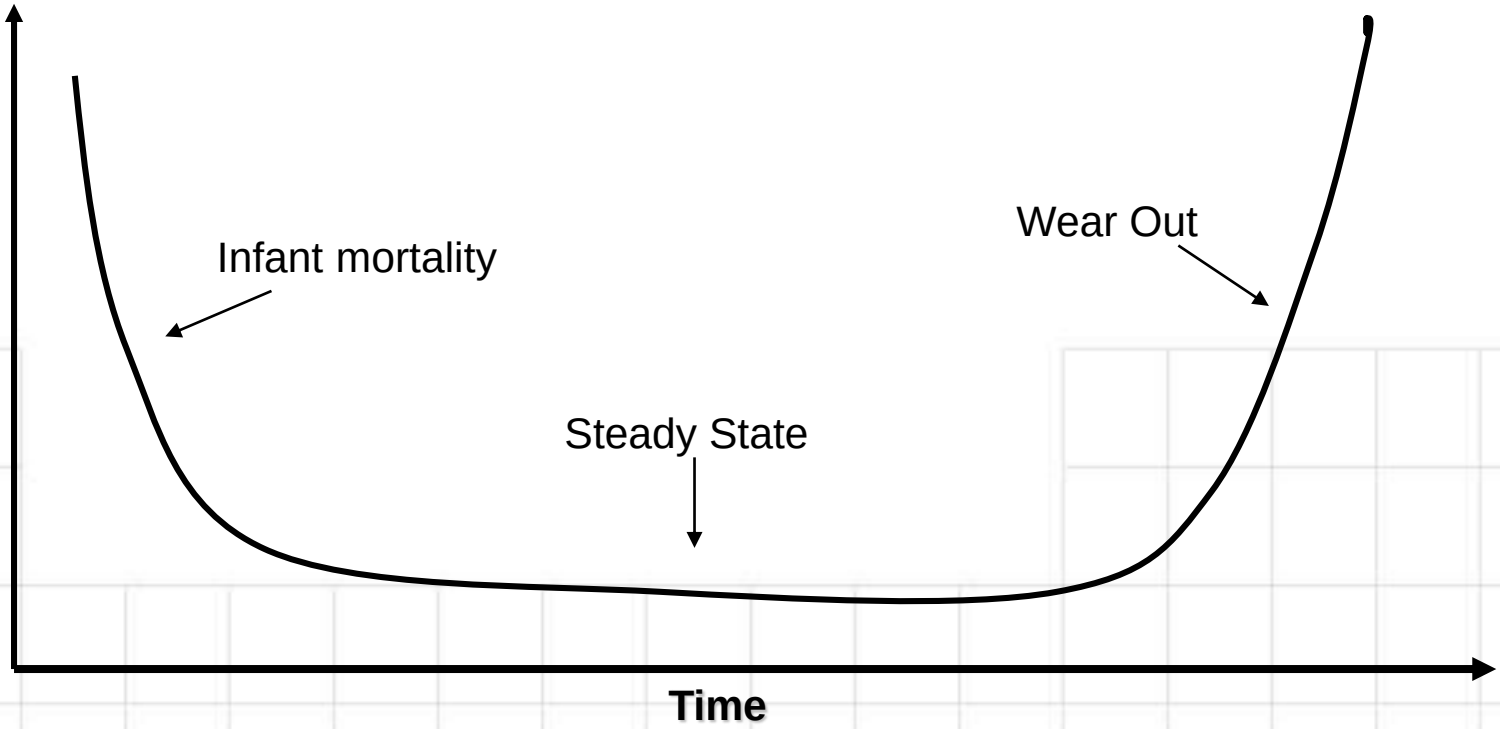
การพัฒนาซอฟต์แวร์อาจคล้ายกับการพัฒนาฮาร์ดแวร์ แต่ความสัมพันธ์ระหว่างคนและงานที่ต้องทำต่างกัน จึงไม่สามารถจัดการด้านซอฟต์แวร์ในแบบเดียวกันกับผลิตฮาร์ดแวร์ได้

: ซอฟต์แวร์ถูกจัดการให้เกิดขึ้น (engineered) ไม่ได้ถูกผลิตขึ้น (manufactured)

คุณสมบัติของซอฟต์แวร์ :

(2) ซอฟต์แวร์ไม่สึกหรอ

ฮาร์ดแวร์มีความสัมพันธ์แบบอ่างอาบน้ำ (Bathtub Curve)

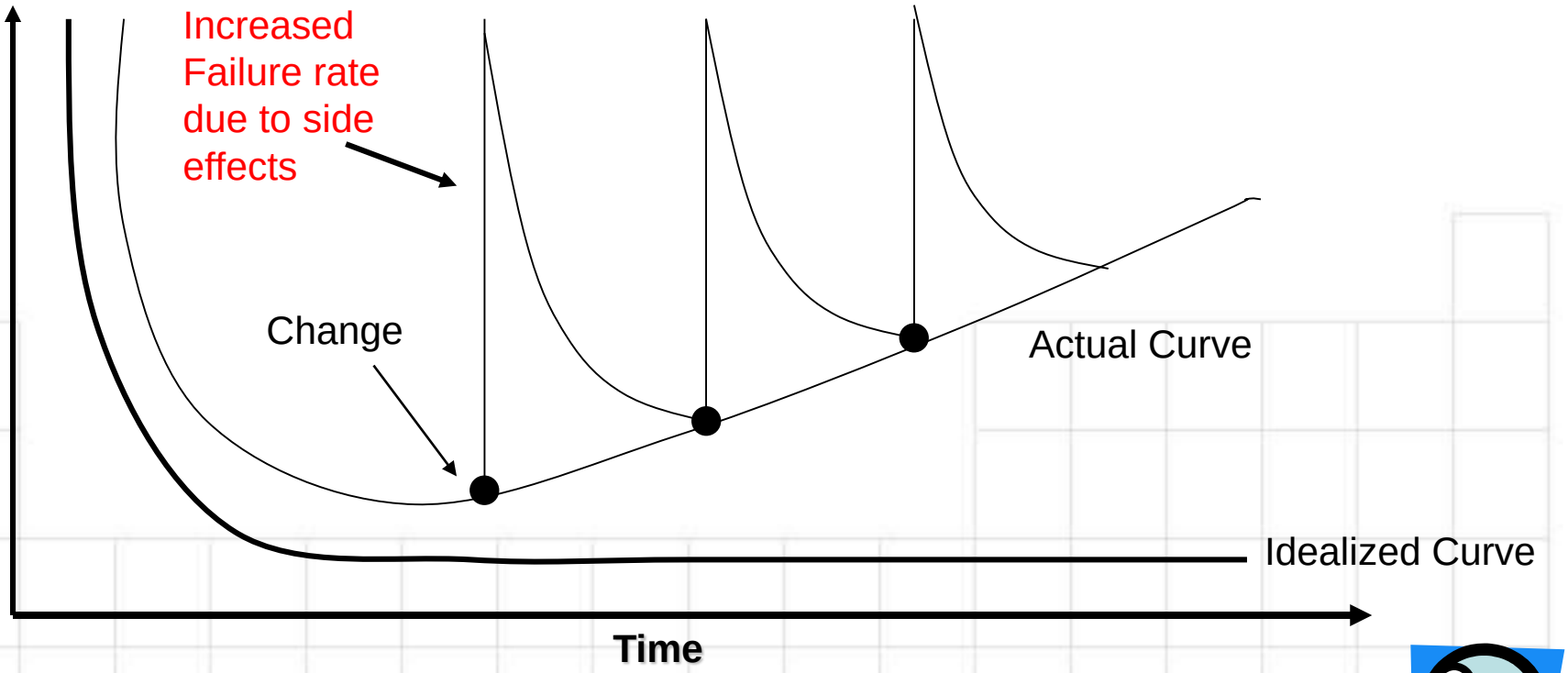


กราฟแสดงอัตราความล้มเหลวของฮาร์ดแวร์

คุณสมบัติของซอฟต์แวร์ :

(2) ซอฟต์แวร์ไม่สึกหรอ

ซอฟต์แวร์มีความสัมพันธ์แบบอุดมคติ (Idealized Curve)



กราฟแสดงอัตราความล้มเหลวของซอฟต์แวร์

Introduction

คุณสมบัติของซอฟต์แวร์ :

(3) อุตสาหกรรมซอฟต์แวร์กำลังมุ่งไปสู่การสร้างจากองค์ประกอบย่อย (Component-based Construction) แต่การผลิตยังออกแบบตามความต้องการของลูกค้าเป็นหลัก (Custom build)
ซึ่งรูปแบบของการผลิตฮาร์ดแวร์ วิศวกรเริ่มร่างแบบวรจรดิจิตอล และใช้ส่วนประกอบดิจิตอล อินเตอร์เฟสมาตรฐานและคู่มือบอกการเชื่อมต่อที่สามารถใช้งานได้ทันที โดยส่วนประกอบเหล่านั้นเมื่อถูกทำเป็นมาตรฐานก็สามารถนำมาใช้ในการออกแบบระบบใหม่ ๆ ได้ ซึ่งเป็นเรื่องปกติของกระบวนการวิศวกรรม
แต่**ซอฟต์แวร์** การออกแบบและพัฒนาส่วนประกอบของซอฟต์แวร์เพื่อให้สามารถนำไปใช้กับโปรแกรมอื่น ๆ เพิ่งเริ่มต้นในระดับหนึ่ง เช่น หน้าต่างกราฟิก เมนูป๊อบอั๊ฟ หรือการติดต่อสื่อสารอื่น ๆ

Introduction

ชนิดของซอฟต์แวร์ที่เปลี่ยนไป :

(1) ซอฟต์แวร์ระบบ (System Software)

คือ ชุดโปรแกรมที่เขียนขึ้นเพื่อบริการโปรแกรมอื่น ๆ ซึ่ง

- อาจมีโครงสร้างสารสนเทศที่ซับซ้อน แต่กำหนดได้
- ต้องติดต่อกับฮาร์ดแวร์อย่างหนัก
- ต้องใช้ได้กับผู้ใช้หลาย ๆ คน
- ต้องทำงานหลาย ๆ อย่างพร้อม ๆ กันอย่างราบรื่น
- ต้องมีการจัดตารางเวลา (Scheduling) จัดแบ่งทรัพยากร (Resource Sharing) และการจัดการกระบวนการที่ล้ำสมัย
- ต้องใช้อินเตอร์เฟสภายนอกหลายชั้น

(2) ซอฟต์แวร์ประยุกต์ (Application Software)

คือ โปรแกรมเฉพาะทางที่ใช้แก้ปัญหาเฉพาะกิจ มีข้อมูลเชิงเทคนิคซึ่งทำให้การตัดสินใจทางการบริหารจัดการต่าง ๆ เป็นไปโดยสะดวก

Introduction

ชนิดของซอฟต์แวร์ที่เปลี่ยนไป :

(3) ซอฟต์แวร์เชิงวิศวกรรม/วิทยาศาสตร์ (Engineering /Scientific Software)

คือ ซอฟต์แวร์ที่มักเป็นขั้นตอนวิธีที่ช่วยในการคำนวณ มีการใช้งานก้าวตั้งแต่ด้านดาราศาสตร์ ชีววิทยา ฟิสิกส์ และอื่น ๆ รวมถึงการจำลองระบบ และงานอินเตอร์แอคทีฟอื่น ๆ

(4) ซอฟต์แวร์ฝังตัว (Embedded Software)

คือ ซอฟต์แวร์ที่ฝังตัวอยู่ในตัวผลิตภัณฑ์ หรือตัวระบบ มีไว้เพื่อให้สามารถใช้และควบคุมลักษณะและหน้าที่ต่าง ๆ เพื่ออำนวยความสะดวกแก่ผู้ใช้งานหรือตัวระบบเอง

(5) ซอฟต์แวร์สายการผลิต (Product-line Software)

คือ ซอฟต์แวร์ที่มีความสามารถเฉพาะเพื่อให้ใช้ได้กับลูกค้าที่แตกต่างกัน และมักเน้นไปทางด้านการตลาด ได้แก่ โปรแกรมประมวลผลคำ โปรแกรมสเปรดชีท โปรแกรมกราฟิก โปรแกรมจัดการฐานข้อมูล เป็นต้น

Introduction

ชนิดของซอฟต์แวร์ที่เปลี่ยนไป :

(6) เว็บแอพพลิเคชั่น (Web Application)

คือ ซอฟต์แวร์ที่ครอบคลุมงานประยุกต์ที่มีรูปแบบที่ง่ายที่สุด แต่นำเสนอสารสนเทศโดยใช้ข้อความและกราฟิกที่จำกัด

(7) ซอฟต์แวร์ปัญญาประดิษฐ์ (Artificial Intelligence Software)

คือ ซอฟต์แวร์ที่ใช้ประโยชน์จากอัลกอริทึมที่ไม่ใช่เชิงการค้า เพื่อแก้ปัญหาที่ยากและซับซ้อนกว่าการวิเคราะห์คำนวณแบบตรง ๆ งานประยุกต์ในซอฟต์แวร์หมวดนี้รวมถึง

- วิทยาการหุ่นยนต์ (Robotics)
- ระบบผู้เชี่ยวชาญ (Expert System)
- การรู้จำแบบ (Pattern Recognition)
- โครงข่ายประสาทเทียม (Artificial Neural Network)
- การพิสูจน์ทฤษฎี (Theorem Proving)
- การเล่นเกม (Game Playing)

Introduction

ความท้าทายที่เกิดขึ้นในปัจจุบัน :

(1) การประมวลผลได้ทุกที่ (Ubiquitous Computing)

การพัฒนาของเครือข่ายไร้สาย (Wireless Network) นำมาสู่คอมพิวเตอร์แบบกระจาย (Distributed Computing) อย่างแท้จริง นักวิศวกรซอฟต์แวร์จึงต้องสร้างซอฟต์แวร์ที่ใช้งานได้บนเครื่องมือเล็ก ๆ ติดต่อสื่อสารกันได้กับคอมพิวเตอร์ทุกรูปแบบทั่วทั้งเครือข่ายที่ไร้ขีดจำกัด

(2) อินเทอร์เน็ต (Internet/ Net Sourcing)

โลกของ WWW เป็นตัวขับเคลื่อนสำคัญสำหรับงานคอมพิวเตอร์ นักวิศวกรซอฟต์แวร์จะต้องสร้างแอพพลิเคชั่นที่ง่ายและทันสมัย และทำให้ผู้ใช้ทั่วโลกใช้งานได้เท่าเทียมกัน

ความท้าทายที่เกิดขึ้นในปัจจุบัน :

(3) โอเพนซอร์ส (Open Source)

ในปัจจุบันมีการเปิดเผยรหัสต้นฉบับ (Source Code) ทำให้ผู้ใช้สามารถปรับเปลี่ยนระบบให้ตรงกับความต้องการของตนเอง ซึ่งนักวิศวกรที่เป็นผู้ผลิตซอฟต์แวร์ลักษณะนี้จะต้องหาทางที่จะเขียนรหัสต้นฉบับซึ่งมีคำอธิบายโดยละเอียดในตัวเอง และสื่อสารให้ผู้ใช้ทราบว่ามีการเปลี่ยนแปลงอะไรในโปรแกรมได้อย่างชัดเจน

(4) เศรษฐกิจยุคใหม่ (The New Economy)

ธุรกิจดอทคอมซึ่งควบคุมตลาดการเงินในช่วงปลายทศวรรษที่ 1990 และตามด้วยความพินาศในช่วยต้นทศวรรษที่ 2000 (YTK) ทำให้ผู้คนเริ่มตระหนักและเชื่อว่าเศรษฐกิจยุคใหม่ได้ตายไปแล้ว แต่ความเป็นจริงเป็นเพียงจุดเริ่มต้นของโลกยุคใหม่ และจะค่อย ๆ วิวัฒนาการอย่างช้า ๆ และกระจายตัวในระดับกว้าง

Introduction

ความท้าทายของนักวิศวกรซอฟต์แวร์ คือ

> **“การหาวิธีที่จะสร้างแอพพลิเคชั่น ที่จะอำนวยความสะดวกในการติดต่อสื่อสาร และการกระจายสินค้า/บริการในระดับมวลชนโดยใช้แนวคิดใหม่”**

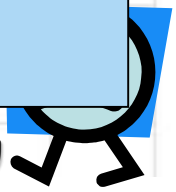
ซึ่งความท้าทายใหม่ ๆ ดังที่กล่าวมา สอดคล้องกับกฎแห่งผลที่ไม่ได้คาดหวัง เพราะเรายังคงคาดถึงผลกระทบบางอย่างไม่ได้ในวันนี้ แต่นักวิศวกรซอฟต์แวร์สามารถจะเตรียมพร้อมได้โดยการเริ่มกระบวนการที่สามารถปรับเปลี่ยนให้เข้ากับเทคโนโลยีที่เปลี่ยนแปลงได้ในอนาคตข้างหน้า

Introduction

5 ปัจจัยที่ต้องคำนึงในเชิงวิศวกรรมซอฟต์แวร์ (Five Different Aspects)

บางปัญหาด้านวิศวกรรมซอฟต์แวร์สามารถแก้ไขโดยหลักการทางคณิตศาสตร์และวิทยาการคอมพิวเตอร์ แต่ก็มีหลายที่ไม่สามารถดำเนินการได้ อาทิ เศรษฐศาสตร์ (Economics) การจัดการ (Management) หรือปรัชญา (Psychology) จึงต้องแสดงให้เห็นถึงขอบเขตที่วิศวกรรมซอฟต์แวร์จะต้องพิจารณา

1. ด้านประวัติศาสตร์ (Historical Aspects)
2. ด้านเศรษฐกิจ (Economic Aspects)
3. ด้านการบำรุงรักษา (Maintenance Aspects)
4. ด้านการวิเคราะห์และออกแบบ (Specification and Design Aspects)
5. ด้านทีมเขียนโปรแกรม (Team Programming Aspects)



Introduction

1. ปัจจัยด้านประวัติศาสตร์ (Historical Aspects)

- !Fact ระบบไฟฟ้าสามารถล่มได้ แต่ยังไม่บ่อยเท่าระบบเงินเดือนล่ม
- !True สะพานสามารถพังได้ แต่ยังไม่เท่ากับที่ระบบปฏิบัติการพัง แม้กระทั่งการพิพากษาผู้ผลิต
- NATO Study group in 1967 coined the term “Software Engineering” A conclusion : Software engineering should use the philosophies and paradigms of established engineering disciplines to solve “Software Crisis”
- Software Crisis : Quality of software ware generally unacceptably low and deadlines and cost limits were not being met.

- Why then cannot bridge-building techniques be used to build operating System ?????

1. ปัจจัยด้านประวัติศาสตร์ (Historical Aspects)

- **First major difference.**

คุณสมบัติของวิศวกรรมด้านอื่น ๆ กับวิศวกรรมซอฟต์แวร์ ยังมีผลกระทบที่เกิดการการล้มเหลวของสิ่งดำเนินการต่างกัน เช่น

Civil ➡

- การออกแบบจะคำนึงถึงความปลอดภัยของมนุษย์
- ผลกระทบจากการล้มเหลวจะคงอยู่และจะเสมือนบทเรียนที่สอนให้รู้ถึงความเสียหายที่เกิดจากโครงสร้างสะพาน

Software ➡

- ระบบปฏิบัติการล่ม จะไม่ถูกนำไปพิจารณาเรื่องการลงทุนใด ๆ
- เมื่อเกิดการล่ม, ก็ไม่มีหลักฐานใด ๆ บ่งบอกถึงสาเหตุการล่มได้
- ถ้าวิศวกรซอฟต์แวร์เอาจริงเอาจังกัดเพื่อไม่ให้เกิดการล่มของระบบปฏิบัติการเหมือนกับวิศวกรโยธาเอาจริงเอาจังป้องกันไม่ได้เกิดการพังของสะพานได้ ก็จะ สามารถยกระดับความเชี่ยวชาญด้านวิศวกรรมซอฟต์แวร์เทียบเท่ากับวิศวกรรมด้านอื่นๆ ได้

1. ปัจจัยด้านประวัติศาสตร์ (Historical Aspects)

- **Second major difference.**
  ระหว่างการปรับปรุง/บำรุงรักษาเพื่อให้สะพานกับระบบปฏิบัติการมีประสิทธิภาพ

  - การบำรุงรักษาสะพาน มักดำเนินการเพียง ทาสีใหม่, ซ่อมแซมรอยร้าวเล็ก ๆ น้อย ๆ , ปรับพื้นถนน เป็นต้น ซึ่งไม่ทำให้โครงสร้างเดิมเปลี่ยนแปลง

  - แต่ด้านวิศวกรซอฟต์แวร์ต้องปรับเปลี่ยนการประมวลผลระบบปฏิบัติการแบบ time sharing จากหลายระบบที่สถาปัตยกรรมแตกต่างกันได้ ซึ่งอาจมีผลทำให้เกิดการเปลี่ยนแปลงโครงสร้างได้

2. ปัจจัยด้านเศรษฐกิจ (Economic Aspects)

- ความสัมพันธ์ระหว่างวิศวกรรมซอฟต์แวร์ และวิทยาการคอมพิวเตอร์เหมือนกับความสัมพันธ์ระหว่างวิศวกรรมเคมี และเคมี โดยที่
  > วิทยาการคอมพิวเตอร์ และเคมี อยู่ในสายวิทยาศาสตร์ ซึ่งมีทั้งทฤษฎีและการปฏิบัติ
  > เคมี ปฏิบัติโดยการเข้าแล็ปทำงาน ส่วนวิทยาการคอมพิวเตอร์ ปฏิบัติโดยการเขียนโปรแกรม

ตัวอย่างเช่น **"กระบวนการสกัดน้ำมันเบนซินจากถ่านหิน"**

- **นักเคมี** -> หาวิธีการที่จะสามารถแปลงถ่านหินให้กลายเป็นน้ำมันเบนซิน
- **วิศวกรเคมี**
  -> พินิจและวิเคราะห์เพื่อจะเลือกกลไกเพียง 1 เดียวที่จะสกัดน้ำมันเบนซินจากถ่านหิน
  -> ประเมินผลในทุกทางที่จะสร้างความชัดเจนในทางเลือกและหาต้นทุนค่าใช้จ่ายต่อลิตรที่น้อยที่สุด

2. ปัจจัยด้านเศรษฐกิจ (Economic Aspects)

- เช่นเดียวกับความสัมพันธ์ระหว่างวิศวกรรมซอฟต์แวร์และวิทยาการคอมพิวเตอร์
  > วิทยาการคอมพิวเตอร์ลงทุนในการผลิตซอฟต์แวร์ที่หลากหลาย ซึ่งอาจดีบางไม่ดีบางเพื่อให้ได้ผลลัพธ์ที่ต้องการ
  > วิศวกรรมซอฟต์แวร์จะลงทุนโดยเลือกเทคนิคหนึ่ง ๆ และหาวิธีการให้ประหยัดค่าใช้จ่ายมากที่สุด

Example!! Currently using CT_{old} discovery that CT_{new} produced in nine-tenths (10%) of the time. Common sense seems to dictate that CT_{new} is appropriate techniques to use.
But!! The economics of software engineering may imply the opposites.
- One reason : Cost of introducing new technology
- Second reason : Economics of Maintenance

3. ปัจจัยด้านการบำรุงรักษา (Maintenance Aspects)

- Life cycle : (waterfall model, 1970s) : seven broad phases.
  1) Requirements : clients 's requirement
  2) Specification (analysis) : the clients 's requirement are analyzed and presented in the form of "specification document". proposed software development .
  3) Design : design 2 process
     > architectural design - broken down into component (module)
     > detailed design – designed each module .
     (resulting design document)
  4) Implementation : coded and tested.
  5) Integration : combined and tested as a whole.

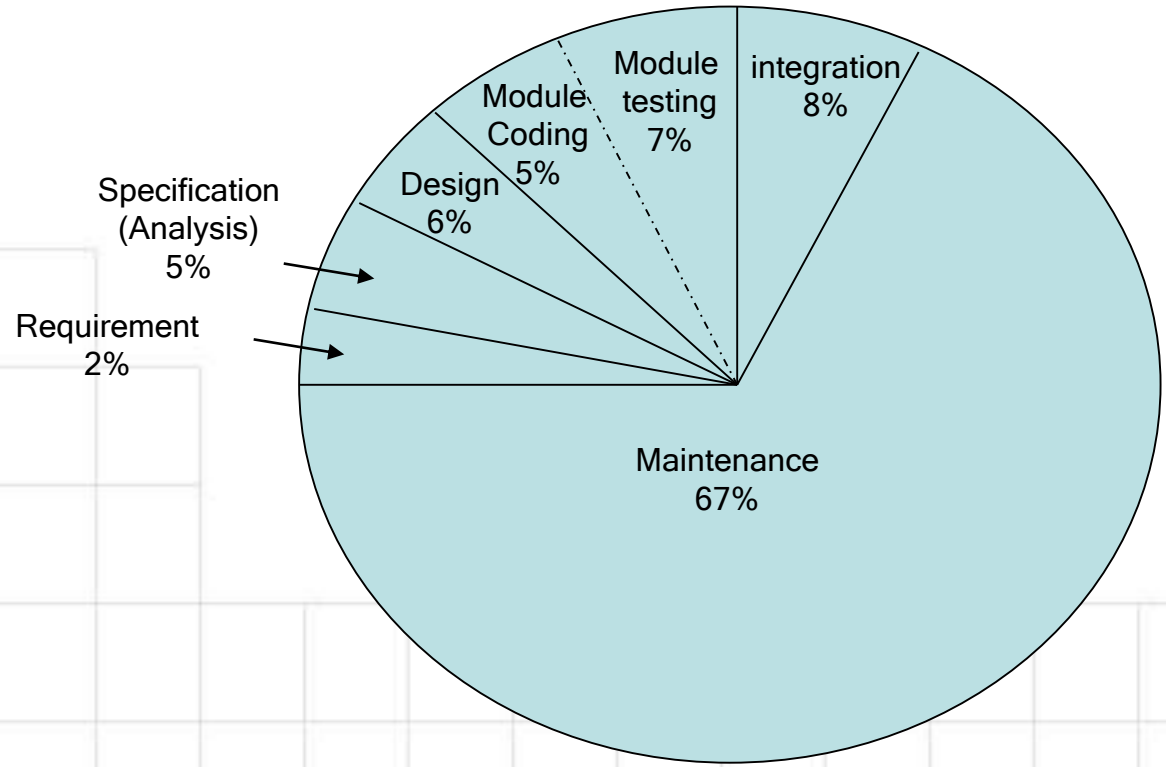
3. ปัจจัยด้านการบำรุงรักษา (Maintenance Aspects)

6) Maintenance : includes all changes to the product
-> specification document. Maintenance includes :
- Corrective maintenance (or software repair)
specification unchanged.
- Enhancement maintenance (or software update)
specification and implementation of those changes.
Two types of enhancement.
(1) perfective maintenance : changes that client thinks will improve the effectiveness of the product.
(2) adaptive maintenance : change that environment in with product operates, new government regulations.

7) Retirement : product is removed from service.

3. ปัจจัยด้านการบำรุงรักษา (Maintenance Aspects)

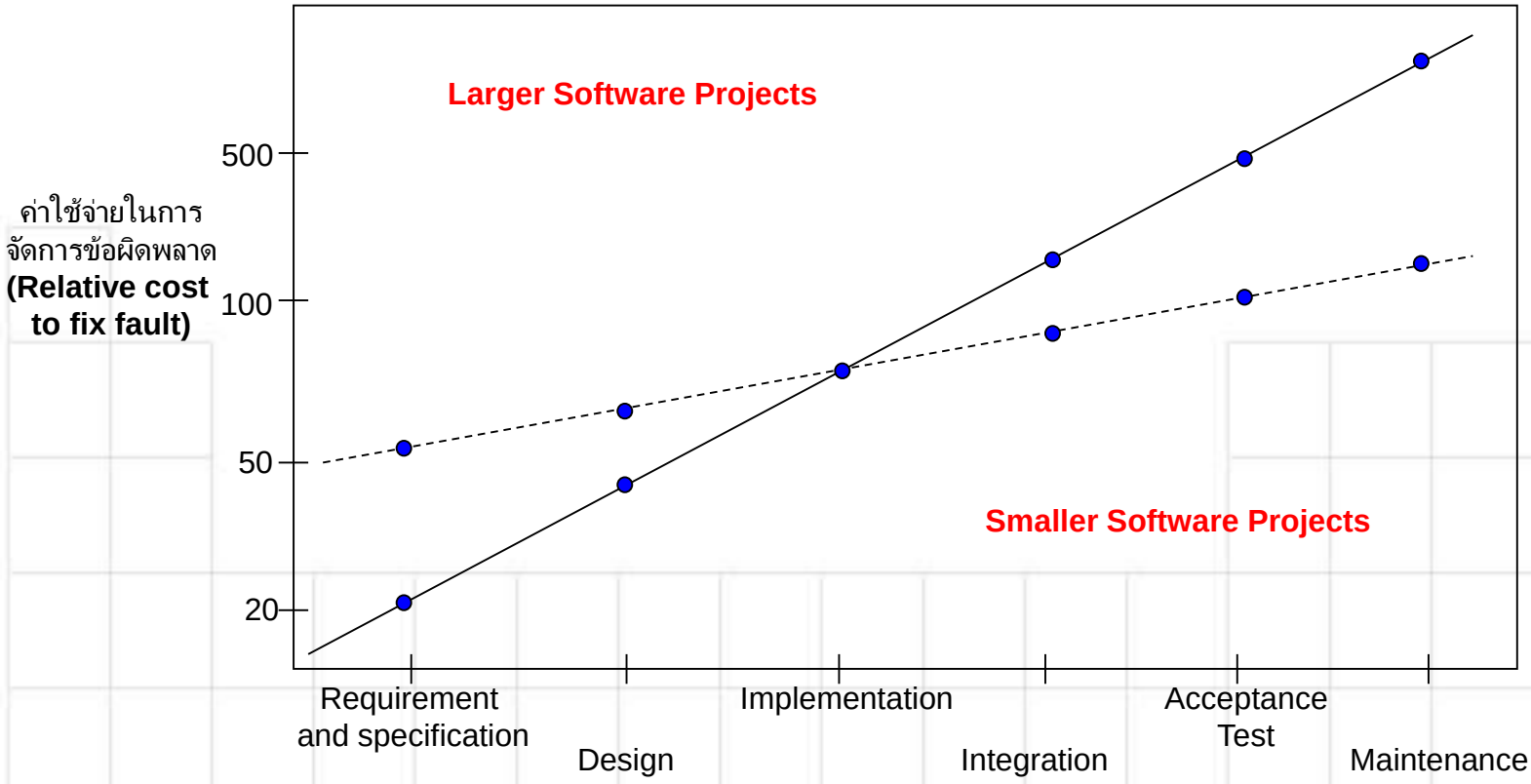
Approximate relative costs of the phases of the software life cycle.
(Stephen R. Schach, 1996, P.11)



Maintenance is so important, a major aspect of software engineering consists of those techniques, tools, and practices the lead to a reduction in maintenance cost.

4. ปัจจัยด้านการวิเคราะห์และออกแบบ (Specification and Design Aspects)

- Software Professional : HUMAN



4. ปัจจัยด้านการวิเคราะห์และออกแบบ (Specification and Design Aspects)

- ถ้าเกิดความผิดพลาดในระหว่าง requirement phase ก็จะมีผลทำให้เกิดข้อผิดพลาดตามมาใน specification, design and the code.
- อย่างไรก็ตาม ไม่ควรให้ความสำคัญเฉพาะการวิเคราะห์และออกแบบ เพราะข้อผิดพลาดอาจเกิดได้ในส่วนงานก่อนและหลังได้
- หากวิเคราะห์และออกแบบได้ดี จะสามารถลดค่าใช้จ่ายของการบำรุงรักษาได้ 10% และสามารถลดค่าใช้จ่ายโดยรวมได้ถึง 7%

  ซึ่งหมายถึง หากลดข้อผิดพลาดในการวิเคราะห์และออกแบบได้ จะสามารถลดจำนวนข้อผิดพลาดโดยรวมได้ 6% -7%

5. ปัจจัยด้านทีมเขียนโปรแกรม (Team Programming Aspects)

- ปัจจัยที่มีผลต่อประสิทธิภาพและค่าใช้จ่าย (Performance-price factor)
  = เวลาที่ใช้ปฏิบัติการในล้านคำสั่ง x ค่าใช้จ่าย CPU และหน่วยความจำหลัก

- หากพิจารณาถึงค่าผลลัพธ์ จะสามารถลดลงได้ในยุคคอมพิวเตอร์ที่เปลี่ยนแปลงไป ซึ่ง ถ้าด้านฮาร์ดแวร์มีประสิทธิภาพดีขึ้น ลงทุนน้อยลง อย่างไรก็ตาม องค์กรสามารถจัดหาฮาร์ดแวร์ในการรันผลิตภัณฑ์ของตนได้ แต่ผลิตภัณฑ์นั้นอย่างไรก็ต้องถูกเขียนโดยบุคคลในเวลาที่กำหนด ดังนั้นปัจจัยที่ต้องคำนึงประการสุดท้ายคือ ทีมเขียนโปรแกรม ว่าควรจะพัฒนาโดยคนเดียว หรือกี่คน จึงจะเหมาะสม

Example: ถ้าผลิตภัณฑ์นั้นจะถูกส่งมอบภายใน 8 เดือน แต่ถ้าพัฒนาโดยนักเขียนโปรแกรมเพียง 1 คน จะต้องใช้เวลาถึง 15 ปีจึงจะสมบูรณ์

Ans ควรต้องพัฒนากันเป็นทีมจึงเหมาะสม.

5. Team Programming Aspects (ต่อ)

- อย่างไรก็ตาม ผู้บริหารโครงการก็ต้องการทีมผู้พัฒนาให้เหมาะสมในแต่ละกระบวนการพัฒนาโดยอาจใช้เทคนิคในการกำหนดความสัมพันธ์ระหว่างคน/งาน/เวลาหรือค่าใช้จ่าย

- รวมถึงหัวหน้าทีมเขียนโปรแกรมต้องสามารถรับปัญหาด้าน interface and communication.

- Example:
  > Somchai and Somsri code modules P and Q respectively, where module P calls module Q.
  > When Somchai codes P he writes a call to Q with five arguments in the argument list. Somsri codes Q with five arguments, but in a difference order from those of Somchai.

  Ans Fault or error result.

5. Team Programming Aspects (ต่อ)

- ปัญหาที่พบ :
  - Design problem
  - Often changed after coding commences
  - Notification of a change is sometimes not distributed to all members of the development team.
  - Poor communication

- **Tip of the iceberg**

However, considerable proportion of today's software engineering must also include techniques for ensuring that teams are properly organized and managed.

แบบฝึกหัดส่งท้ายชั่วโมงเรียน วันนี้!!

จงเขียน Flowchart

1. เขียนอัลกอริทึมการคิดอัตราภาษีนิติบุคคล (กระดาษ)

กำไรสุทธิ	อัตราภาษีร้อยละ
ไม่เกิน 300,000 บาท	ยกเว้น
เกิน 300,000 บาท แต่ไม่เกิน 3,000,000 บาท	15
เกิน 3,000,000 บาท ขึ้นไป	20

ตัวอย่าง

รับค่า 300,000 บาท -> เสียภาษี 0 บาท
รับค่า 500,000 บาท -> เสียภาษี 30,000 บาท
รับค่า 5,000,000 บาท -> เสียภาษี 405,000+400,000=805,000 บาท

2. เขียนโปรแกรมด้วย Delphi ตามอัลกอริทึมข้อที่ 1 (งานส่งในชั้นเรียนสัปดาห์หน้า)