



[SF220-W01]

# บทนำสู่วิศวกรรมซอฟต์แวร์

## Introduction to Software Engineering

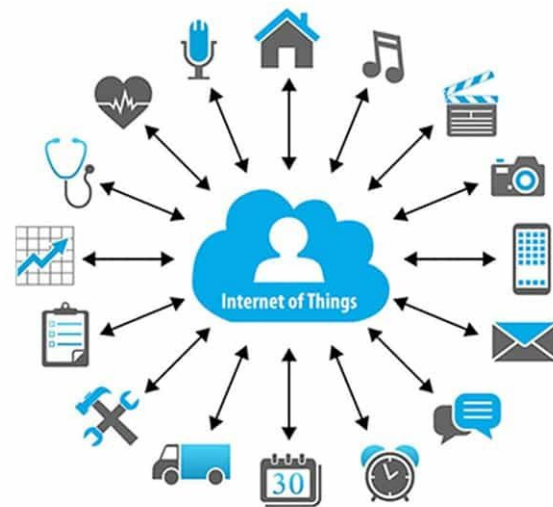
- ❖ **ผู้สอน:** อ.ดร.อัศวรุตติ ตาตาม
- ❖ **วัน:** ศุกร์ที่ 13 มกราคม พ.ศ. 2565
- ❖ **เวลา:** 9.30-12.30
- ❖ **สถานที่:** วศ.606/1

# Topics

1. นิยามของซอฟต์แวร์และวิศวกรรมซอฟต์แวร์  
Definitions of Software and Software Engineering
2. บุคคลที่เกี่ยวข้องกับวิศวกรรมซอฟต์แวร์  
Software Engineering Key People
3. วิวัฒนาการและความสำเร็จของวิศวกรรมซอฟต์แวร์
4. องค์ประกอบของวิศวกรรมซอฟต์แวร์
5. แนวทางปฏิบัติด้านวิศวกรรมซอฟต์แวร์

# บทนำ

- คอมพิวเตอร์เป็นสิ่งที่มีความประโยชน์และโทษต่อมนุษย์
- ชีวิตประจำวันเราใช้คอมพิวเตอร์ในทุกด้าน
- ผู้ใช้ต้องป้อนคำสั่งให้กับคอมพิวเตอร์
- คำสั่งที่ใช้ควบคุม เรียกว่า **ซอฟต์แวร์**



ในปัจจุบันนี้ **ซอฟต์แวร์**คอมพิวเตอร์ ถือได้ว่าเป็นเทคโนโลยีที่เข้ามามีบทบาทในชีวิตประจำวันของคนเรามากขึ้น ทั้งในชีวิตส่วนตัว ทั้งทางด้านการศึกษา ธุรกิจ อุตสาหกรรม ความบันเทิง และอื่นๆ อีกมากมาย

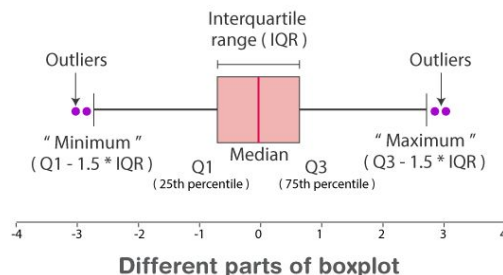
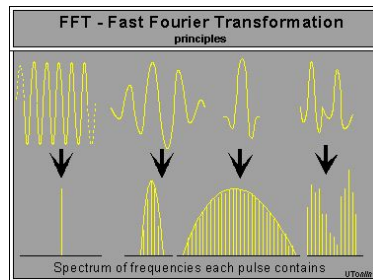
# เป้าหมายหลักของการพัฒนาซอฟต์แวร์

- เพื่อให้ได้ซอฟต์แวร์ที่มี
  - ประสิทธิภาพ
  - ถูกต้อง
  - เชื่อถือได้
  - รวดเร็ว
  - ปลอดภัย
  - ทนสมัย
  - ง่ายต่อการใช้งาน
  - ง่ายต่อการปรับเปลี่ยน
  - เคลื่อนย้ายง่าย
  - มีส่วนต่อประสานกับระบบอื่น
  - สามารถนำกลับมาใช้ใหม่ได้

# จอห์น ไวลด์เลอร์ ทูเคย์ | John Wilder Tukey

**John Wilder Tukey** (June 16, 1915 – July 26, 2000)

- An American mathematician and statistician
- Best known for the development of
  - **The fast Fourier Transform (FFT)** algorithm  
(วิธีทางคณิตศาสตร์สำหรับอธิบายการแปลงความสัมพันธ์  
จากโดเมนเวลาเป็นโดเมนความถี่ในการวิเคราะห์การสั่นสะเทือน)
  - **กราฟรูปกล่อง (Box Plot)**  
การนำเสนอข้อมูลเชิงปริมาณในรูปกราฟ  
เพื่อแสดงลักษณะการแจกแจงตำแหน่งและการกระจาย  
ของข้อมูล
- Coining
  - The term '**bit**' and
  - The first published use of the word '**software**'.



Computer Bit



Computer Byte



ComputerHope.com

# จอห์น ไวลด์เนอร์ ทูเคย์ | John Wilder Tukey

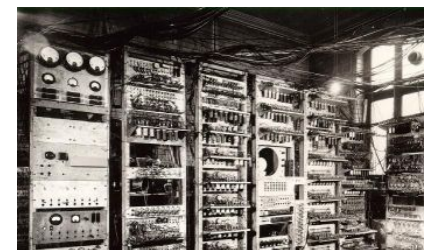
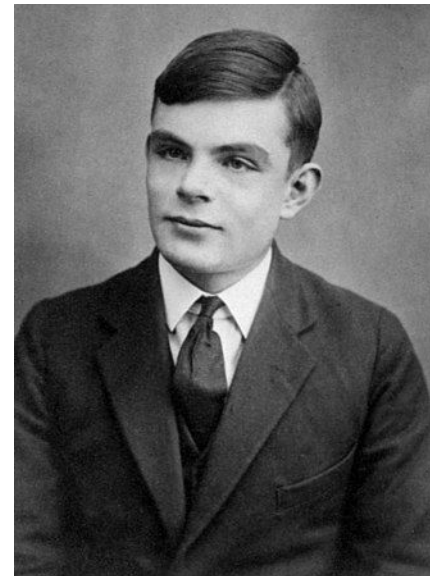
- The first published use of the term "software" in a computing context is often credited to American statistician John W. Tukey, who published the term in "The Teaching of Concrete Mathematics," American Mathematical Monthly, January 9, 1958.
- Tukey wrote:  
 "Today the '**software**' comprising the carefully planned interpretive routines, compilers, and other aspects of automative programming are at least as important to the modern electronic calculator as its 'hardware' of tubes, transistors, wires, tapes and the like"



"ทุกวันนี้ '**ซอฟต์แวร์**' ที่ประกอบด้วย ขั้นตอนการตีความและวางแผนไว้อย่างรอบคอบ ในการแปลโปรแกรม เป็นโปรแกรมคอมพิวเตอร์(คอมไพเลอร์) และแง่มุมอื่น ๆ การเขียนโปรแกรมอัตโนมัติมีความสำคัญต่อเครื่องคิดเลขอิเล็กทรอนิกส์สมัยใหม่เป็นอย่างน้อย เช่นเดียวกับ '**ฮาร์ดแวร์**' ของหลอด ทรานซิสเตอร์ สายไฟ เทป และอื่น ๆ"

# แอลัน แมธิสัน ทัวริง | Alan Mathison Turing

- **แอลัน แมธิสัน ทัวริง** (แนวคิดของซอฟต์แวร์ปรากฏครั้งแรก)  
– นักคณิตศาสตร์, นักตรรกศาสตร์, นักรหัสวิทยาและวีรบุรุษสงครามชาวอังกฤษ และเป็นที่ยอมรับว่าเป็นบิดาของวิทยาการคอมพิวเตอร์  
English mathematician, computer scientist, logician, cryptanalyst, philosopher, and theoretical biologist.
- **What Did Alan Turing Invent?**  
Turing, often with the help of others, invented or helped develop:
  - The concept of **Artificial intelligence**. In his famous 1950 paper.
  - The concept of **a programable digital computer** with his Turing Machines (which build on Babbage's theoretical machines). Which he introduced in 1936.
  - The **Manchester Mark 1**. One of the first computers. Created in 1948.
  - And much more.



The Manchester Mark 1

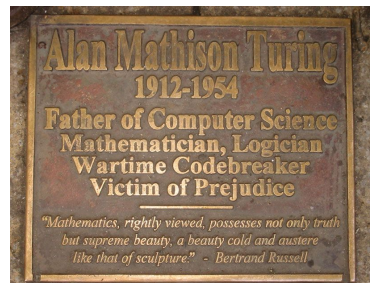
## บุคคลสำคัญในวงการคอมพิวเตอร์

1. Atanasoff, John V.(จอห์น วี. อะทานาซอฟฟ์)
2. Lovelace, Ada (เอดา เลิฟเลซ)
3. Babbage, Charles (ชาร์ลส แบบเบจ)
4. Minsky, Marvin (มาร์วิน มินสกี)
5. Berners-Lee, Tim (ทิม เบอร์เนอรส์-ลี)
6. Neumann, John von (จอห์น วอน นอยแมน)
7. Bricklin, Daniel (แดเนียล บริคลิน)
8. Papert, Seymour (เซย์มัวร์ ปาเปิร์ต)
9. Cerf, Vinton (วินตัน เซิร์ฟ)
10. Postel, Jonathan (โจนาธาน โพสเทล)
11. Cray, Seymour (เซมัวร์ เครย์)
12. Ritchie, Dennis (เดนนิส ริทชี)
13. Dijkstra, Edsger
14. Sinclair, Clive (ไคลฟ์ ซินแคลร์)
15. Gates, Bill (บิล เกตส์)
16. Stallman, Richard (ริชาร์ด สตอลแมน)
17. Hopper, Grace Murray (เกรซ มัวร์เรย์ ฮอปเปอร์)
18. Torvalds, Linus (ลินุส ทอร์วัลด์ส)
19. Jobs, Steve (สตีฟ จ๊อบส์)      Turing,
20. Alan Mathison (อลัน มาทิสัน ทัวริง)
21. Kildall, Gary (แกรี่ คิลดัลล์)
22. Wall, Larry (แลร์รี วอลล์)
23. Knuth, Donald (โด널ด์ คุก)
24. Wozniak, Stephen (สตีเฟน วอซเนียค)

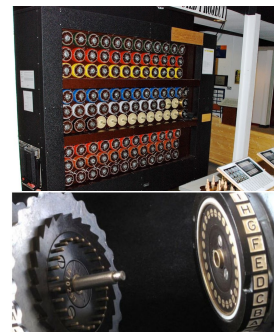
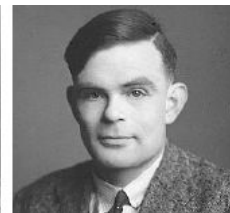


# Alan Mathison Turing

- **Alan Turing** gets the title  
“**the father of computer science and Artificial Intelligence**”  
from his early work where he illustrated the concept of AI and programmable digital computers before either existed.



- **Turing** made significant contributions to computing, codebreaking, and even helped the Allies win WWII.
- What Did **Alan Turing** Invent?  
The **Bombe**, an early computer that cracked Germany's Enigma codes during WWII (based on a Polish device, the Bomba).  
**Breaking the Enigma code** is thought to have shortened the war with Germany by years and saved millions of lives.  
Developed in 1939.



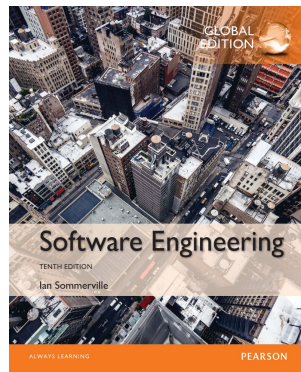
# เียน ซอมเมอร์วิลล์ | Ian Sommerville (Software Engineer)

Ian F. Sommerville, (born 23 February 1951) is a British academic.

- The author of a popular student textbook on **systems engineering**, as well as a number of other books and papers.
- Worked as a professor of **systems engineering** at the University of St Andrews in Scotland until 2014 and is
- A prominent researcher in the field of **systems engineering**, system dependability and social informatics, being an early advocate of an interdisciplinary approach to system dependability.

## Publications

- Sommerville, I. (2016) Software Engineering. 10th Edition, Pearson Education Limited, Boston.



## Some Basic Definitions

- **n. Software** -- Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.  
โปรแกรมคอมพิวเตอร์ ขั้นตอน และเอกสารและข้อมูลที่เกี่ยวข้องที่เกี่ยวข้องกับการทำงานของระบบคอมพิวเตอร์
  - ชุดคำสั่ง ที่เรียงกันเป็นโปรแกรมคอมพิวเตอร์, สั่งงานให้คอมพิวเตอร์ทำงาน
  - โครงสร้างข้อมูล เช่น ระบบการจัดการฐานข้อมูล มีการจัดทำเอกสาร
  - ใช้งานบนคอมพิวเตอร์และอุปกรณ์อิเล็กทรอนิกส์อื่น
- **n Engineering**  
-- Application of systematic, disciplined, quantifiable approach to some process.  
การประยุกต์ใช้แนวทางเชิงปริมาณอย่างเป็นระบบ มีระเบียบวินัย กับกระบวนการบางอย่าง

# ธรรมชาติของซอฟต์แวร์ | Nature of Software

## 1. มองไม่เห็น จับต้องไม่ได้ (Intangible Product)

- การพัฒนาซอฟต์แวร์ขึ้นอยู่กับหลายปัจจัย จะต้องทำการทดสอบประสิทธิภาพซอฟต์แวร์จึงเป็นไปได้ลำบาก

## 2. ทำสำเนา นำกลับมาใช้ใหม่ได้ (Reuse)

- การนำกลับมาใช้ใหม่ของซอฟต์แวร์ มีส่วนช่วยให้นักพัฒนาทำงานได้สะดวกขึ้น

## 3. ไม่ใช่เครื่องจักรในการผลิต ในมุมมองของอุตสาหกรรมผลิตซอฟต์แวร์

- ควรจัดตั้งทีมงานที่มีความรู้ความสามารถ เพื่อพัฒนาซอฟต์แวร์ให้ได้ประสิทธิภาพมากที่สุด

## 4. ยากต่อการเปลี่ยนแปลง เพราะระบบซับซ้อนสูง

- ซอฟต์แวร์ที่ดีต้องไม่เปลี่ยนไปตามกาลเวลา หากจะต้องคงอยู่และใช้งานได้อย่างคงทน

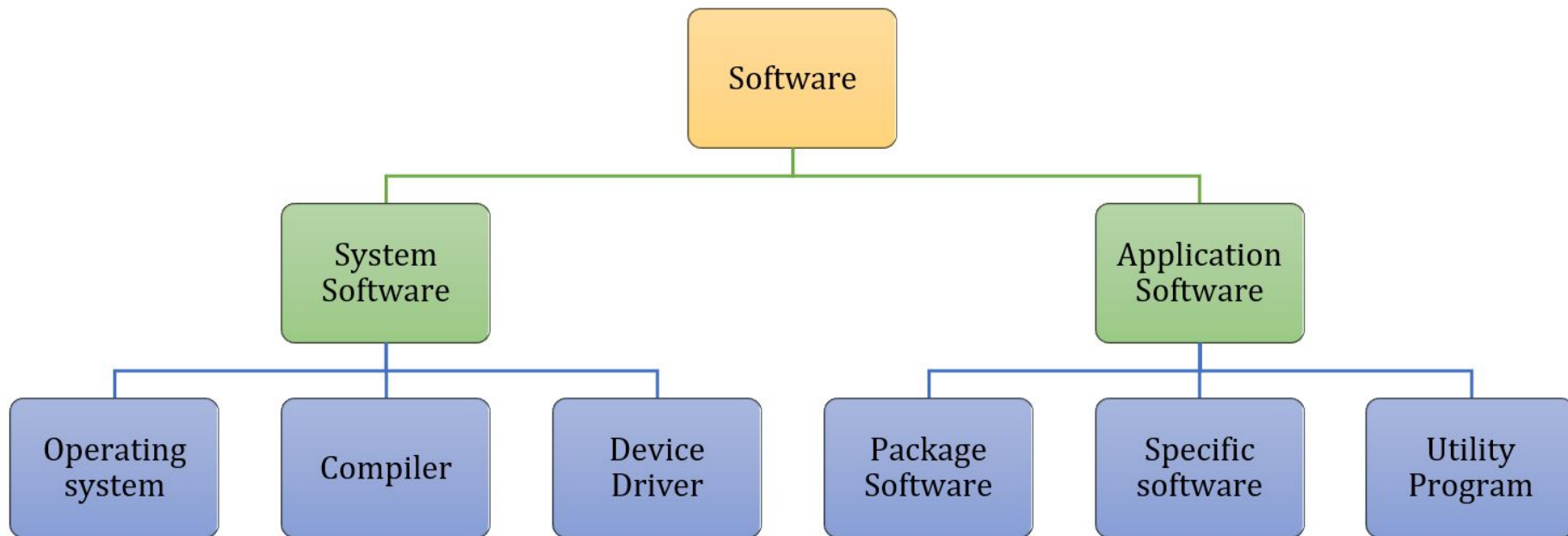
## 5. สร้างตามข้อกำหนดที่ลูกค้าและผู้ใช้ต้องการ

- แม้ว่าการพัฒนาซอฟต์แวร์สำเร็จรูปจะเป็นที่แพร่หลายเป็นอย่างมาก แต่ความต้องการของลูกค้าที่มีความต้องการซอฟต์แวร์เฉพาะงานนั้นก็ยังคงมีอยู่มากเช่นกัน

## 6. ซอฟต์แวร์ที่ไม่สืบทอดหรือไม่ล้าสมัย

- มักพบความล้มเหลวบ้าง ในขั้นตอนของการพัฒนาช่วงแรกๆ เพราะอาจจะยังไม่เข้าใจกันอยู่บ้าง
- เมื่อเวลาผ่านไปพบว่าอัตราความล้มเหลวจะลดลง และซอฟต์แวร์ที่พัฒนามาก็จะกลายเป็นซอฟต์แวร์ที่คงทนถาวร

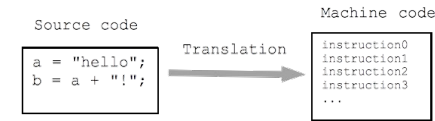
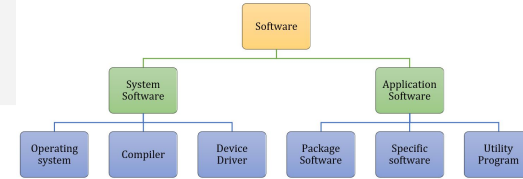
# ประเภทของซอฟต์แวร์ | Types of Software



# ประเภทของซอฟต์แวร์ | Types of Software

## ซอฟต์แวร์ระบบ (System Software)

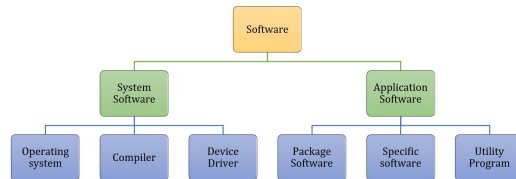
- ซอฟต์แวร์ที่ทำงานโดยตรงกับฮาร์ดแวร์คอมพิวเตอร์และให้ฟังก์ชันการทำงานพื้นฐานแก่ผู้ใช้งานรวมถึงซอฟต์แวร์อื่นๆ เพื่อให้ทำงานได้อย่างราบรื่น
- โดยทั่วไปจะควบคุมการทำงานของฮาร์ดแวร์ และยังควบคุมอุปกรณ์ฮาร์ดแวร์ เช่น จอภาพ เครื่องพิมพ์ และ อุปกรณ์เก็บข้อมูล เป็นต้น เปรียบเสมือนส่วนต่อประสานระหว่างฮาร์ดแวร์และแอปพลิเคชันของผู้ใช้
- ช่วยให้พวกเราสื่อสารกับ **ฮาร์ดแวร์** เข้าใจภาษาเครื่อง (เช่น 1 หรือ 0)  
 ในขณะที่ซอฟต์แวร์ทำงานในภาษาที่มนุษย์อ่านได้ เช่น อังกฤษ ฮินดี เยอรมัน เป็นต้น  
 ดังนั้นซอฟต์แวร์ระบบจะแปลงภาษาที่มนุษย์อ่านได้เป็นภาษาเครื่องทั้งไปและกลับ



## ซอฟต์แวร์ประยุกต์ (Application Software)

- ซอฟต์แวร์ที่ทำหน้าที่พิเศษหรือมีฟังก์ชันที่มากกว่าการทำงานของคอมพิวเตอร์ เรียกว่า **ซอฟต์แวร์ประยุกต์**
  - ออกแบบเพื่อทำงานเฉพาะสำหรับผู้ใช้
  - เป็นผลิตภัณฑ์หรือโปรแกรมที่ออกแบบมาเพื่อตอบสนองความต้องการของผู้ใช้
  - เช่น โปรแกรมประมวลผลคำ สเปรดชีต การจัดการฐานข้อมูล สินค้าคงคลัง โปรแกรมเงินเดือน ฯลฯ

# ซอฟต์แวร์ระบบ ( System Software)



## 1. ระบบปฏิบัติการ (Operating System)

- เป็นโปรแกรมหลักของระบบคอมพิวเตอร์
- เมื่อระบบคอมพิวเตอร์เปิด จะเป็นซอฟต์แวร์ตัวแรกที่โหลดเข้าสู่หน่วยความจำของคอมพิวเตอร์
- โดยพื้นฐานแล้วจะจัดการทรัพยากรทั้งหมด  
เช่น หน่วยความจำ CPU เครื่องพิมพ์ ฮาร์ดดิสก์ ฯลฯ และมีส่วนต่อประสานกับผู้ใช้
  - i. ช่วยให้ผู้ใช้สามารถโต้ตอบกับระบบคอมพิวเตอร์ได้
- นอกจากนี้ยังให้บริการต่าง ๆ กับซอฟต์แวร์คอมพิวเตอร์อื่น ๆ

**ตัวอย่างระบบปฏิบัติการ** Linux, Apple macOS, Microsoft Windows



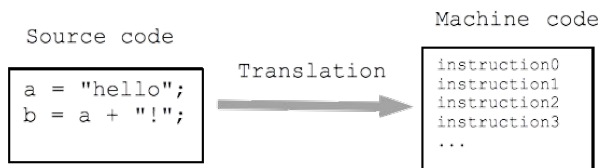
## 2. ตัวประมวลผลภาษา (Language Processor)

- เราทราบกันดีว่าซอฟต์แวร์ระบบจะแปลงภาษาที่มนุษย์อ่านได้ให้เป็นภาษาเครื่องทั้งไปและกลับ
- โดยจะแปลงโปรแกรมที่เขียนด้วยภาษาโปรแกรมระดับสูง เช่น

**จาก** Java, C, C++, Python เป็นต้น (เรียกว่าซอร์สโค้ด - **source code**)

**ไปเป็น** ชุดคำสั่งที่เครื่องสามารถอ่านได้ง่าย

(เรียกว่าออบเจกต์โค้ดหรือรหัสเครื่อง - **object code** or **machine code**)

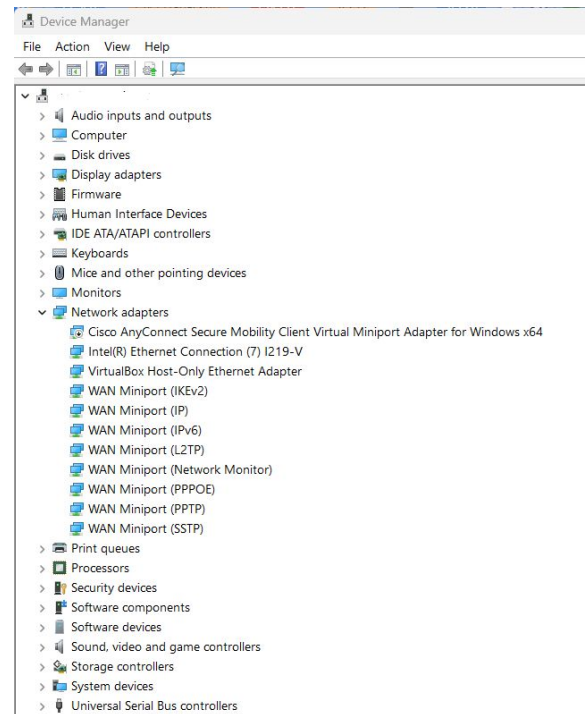
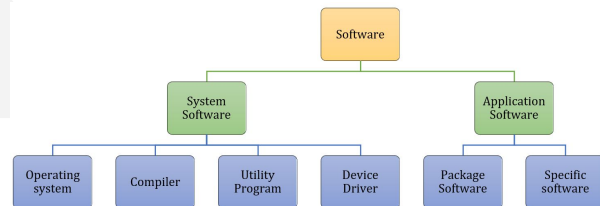




# ซอฟต์แวร์ระบบ ( System Software)

## 3. โปรแกรมควบคุมอุปกรณ์ (Device Driver)

- โปรแกรมหรือซอฟต์แวร์ที่ควบคุมอุปกรณ์และช่วยให้อุปกรณ์นั้นทำงานตามหน้าที่ได้ อุปกรณ์ทุกชนิด
- เช่น เครื่องพิมพ์ เมาส์ โมเด็ม ฯลฯ
- จำเป็นต้องมีไดรเวอร์เพื่อเชื่อมต่อกับระบบคอมพิวเตอร์ตลอดไป
- ดังนั้น เมื่อเราเชื่อมต่ออุปกรณ์ใหม่กับระบบคอมพิวเตอร์ก่อนอื่นเราต้องติดตั้งไดรเวอร์ของอุปกรณ์นั้นเพื่อให้ระบบปฏิบัติการของเรารู้วิธีควบคุมหรือจัดการอุปกรณ์นั้น





# ประเภทของซอฟต์แวร์ประยุกต์) | Types of Application Software

## 1. ซอฟต์แวร์วัตถุประสงค์ทั่วไป (General Purpose Software)

- ซอฟต์แวร์ประยุกต์สำหรับงานที่หลากหลายและ  
ไม่จำกัดเฉพาะการทำงานเฉพาะด้านเท่านั้น
- ตัวอย่าง** MS-Word, MS-Excel, PowerPoint



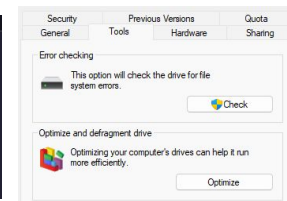
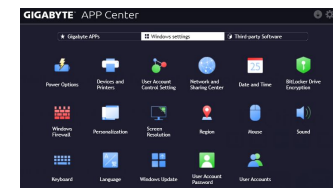
## 2. ซอฟต์แวร์ที่กำหนดเอง (Customized Software):

- ซอฟต์แวร์ประยุกต์ออกแบบมาเพื่อทำงานหรือหน้าที่เฉพาะ
- ตัวอย่าง** ระบบจองรถไฟ ระบบจองสายการบิน ระบบจัดการใบแจ้งหนี้ เป็นต้น



## 3. ซอฟต์แวร์อรรถประโยชน์ (Utility Software):

- ซอฟต์แวร์ประยุกต์เพื่อสนับสนุนโครงสร้างพื้นฐานของคอมพิวเตอร์  
เพื่อวิเคราะห์ กำหนดค่า เพิ่มประสิทธิภาพและบำรุงรักษาระบบ
- ตัวอย่าง** โปรแกรมป้องกันไวรัส ตัวแยกส่วนดิสก์  
ตัวทดสอบหน่วยความจำ การซ่อมแซมดิสก์ ตัวล้างดิสก์  
ตัวล้างรีจิสทรี ตัววิเคราะห์พื้นที่ดิสก์



# Characteristics and Features of Software (1)

## 1. ซอฟต์แวร์ระบบ (System Software)

- ซอฟต์แวร์ระบบเป็นชุดโปรแกรมที่เขียนขึ้นเพื่อให้บริการติดต่อ ควบคุมการทำงานพื้นฐานของฮาร์ดแวร์ อีกทั้งการแปลความหมายของคอมพิวเตอร์เพื่อไปแสดงผล

## 2. ซอฟต์แวร์ประยุกต์ (Application Software)

- ซอฟต์แวร์ประยุกต์ที่พัฒนาขึ้นมาเพื่อวัตถุประสงค์การใช้งานเฉพาะอย่าง
- ตัวอย่าง** การประมวล ณ์ จุดขาย การควบคุมกระบวนการผลิตตามเวลาจริง

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

- มักเป็นขั้นตอนวิธีที่ช่วยในการคำนวณ มีการใช้งานกว้าง ตั้งแต่ทางด้านดาราศาสตร์ ชีววิทยา ฟิสิกส์ เคมีและอื่นๆ ในงานบางลักษณะ
- มีการจำลองวิธีการทำงานการคำนวณในเชิงวิศวกรรม เพื่อให้เห็นผลการทำงานเหล่านั้นเพื่อประกอบการตัดสินใจ

## Characteristics and Features of Software (2)

### 4. ซอฟต์แวร์ฝังตัว (Embedded Software)

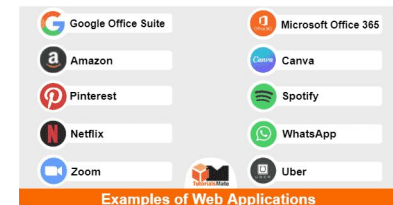
- เป็นซอฟต์แวร์ที่ฝังอยู่ในตัวผลิตภัณฑ์
- ตัวระบบที่มีไว้เพื่อให้สามารถใช้และควบคุมลักษณะและหน้าที่ต่างๆ
- ปัจจุบันเราพบอยู่ในอุปกรณ์ไฟฟ้า เครื่องครัว หรือ แม้กระทั่งอุปกรณ์ในชีวิตประจำวัน เช่น นาฬิกา

### 5. ซอฟต์แวร์สายการผลิต (Product-line Software)

- เป็นซอฟต์แวร์ที่มีความสามารถเฉพาะด้านการผลิตเพื่อให้ใช้ได้กับลูกค้าที่แตกต่างกันซอฟต์แวร์ประเภทนี้เน้นเฉพาะกลุ่มบุคคล

### 6. เว็บแอปพลิเคชัน (Web Application)

- เป็นซอฟต์แวร์ที่ใช้งานง่าย ความสามารถในการใช้งานสูง
- มักจะใช้เว็บแอปพลิเคชันในการแสดงผลข้อความเชื่อมโยงและกราฟฟิกต่างๆ



# Characteristics and Features of Software (3)

## 7. ซอฟต์แวร์ปัญญาประดิษฐ์ (Artificial Intelligence Software)

- เป็นซอฟต์แวร์เพื่อแก้ปัญหาที่มีความซับซ้อนกว่าการวิเคราะห์คำนวณแบบตรงๆ
- **ตัวอย่าง** วิทยาการหุ่นยนต์ (Robotics) ระบบผู้เชี่ยวชาญ (Expert System) การเรียนรู้จำแบบ (Pattern Recognition) โครงข่ายประสาทเทียม (Artificial Neural Network) การพิสูจน์ทฤษฎี (Theorem Proving) และการเล่นเกม (Game Playing)

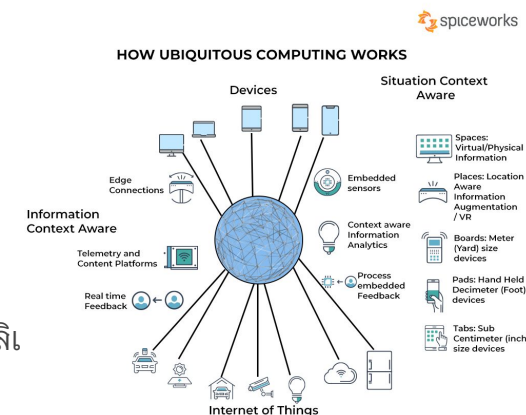


## 8. การประมวลผลได้ทุกที่ (Ubiquitous Computing)

- เป็นซอฟต์แวร์ที่พัฒนาไปอย่างรวดเร็วของเครือข่ายไร้สาย (Wireless Network) นำมาสู่การประมวลผลคอมพิวเตอร์แบบกระจาย (Distributed Computing) ที่สามารถทำงานได้อย่างรวดเร็ว

## 9. อินเทอร์เน็ต (Internet)

- เป็นการพัฒนาซอฟต์แวร์เพื่อตอบสนองโลกของเวปไซด์ไวต์เว็บ
- ขั้วเคลื่อนสำคัญสำหรับงานคอมพิวเตอร์ นักวิศวกรซอฟต์แวร์จะต้องสร้างแอปพลิเคชันให้ทั่วโลกใช้ประโยชน์ได้เท่าเทียมกัน



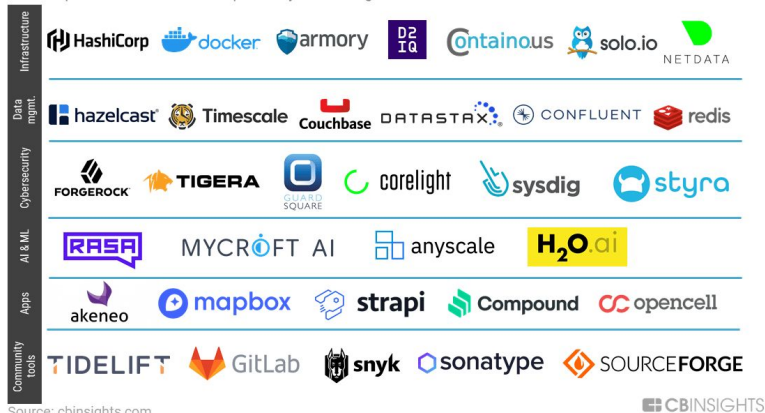
# Characteristics and Features of Software (4)

## 10. โอเพนซอร์ส (Open Source)

- เป็นพัฒนาซอฟต์แวร์โดยการเปิดเผยรูปแบบวิธีการพัฒนารูปแบบการเขียนโค้ดโปรแกรม เพื่อให้ผู้ที่มีความสนใจมาพัฒนาร่วมกันได้ต่อไป

### The open-source software ecosystem expands

Open-source software companies by market segment



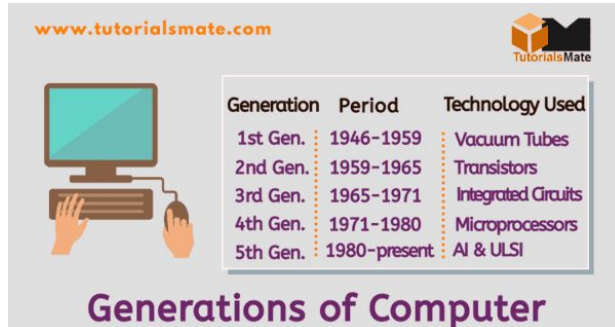
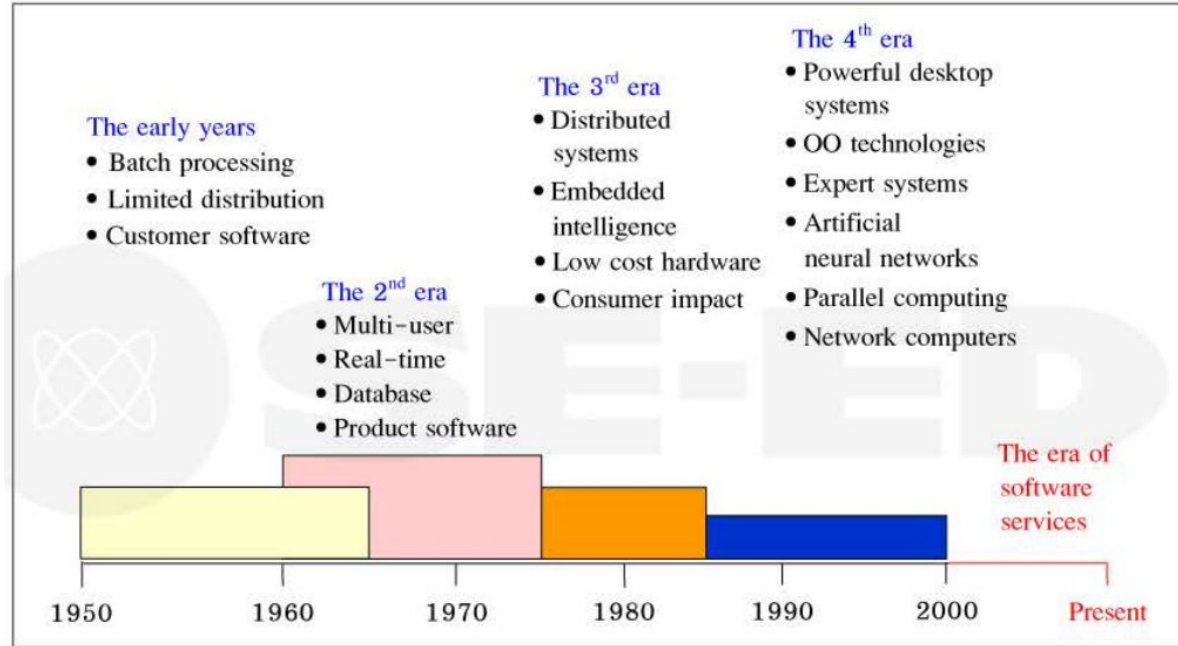
### Example of popular open source software



infopulse

# วิวัฒนาการของวิศวกรรมซอฟต์แวร์ | Evolution of Software Engineering

- ยุคแรก (ปี ค.ศ. 1950-1965)
- ยุคที่สอง (ปี ค.ศ. 1960-1975)
- ยุคที่สาม (ปี ค.ศ. 1975-1985)
- ยุคที่สี่ (ปี ค.ศ. 1985-2000)
- ยุคบริการซอฟต์แวร์ (ปี ค.ศ. 2000 ถึง ปัจจุบัน)



# วิวัฒนาการของการพัฒนาซอฟต์แวร์ (1/3)

## 1. ช่วงเริ่มต้นของวิศวกรรมซอฟต์แวร์ (1945 – 1965)

- วิศวกรรมซอฟต์แวร์จะเป็นที่รู้จักกันเฉพาะกลุ่มของผู้พัฒนาซอฟต์แวร์เท่านั้น
- มุ่งเน้นผลิตซอฟต์แวร์เพื่อการค้า
- จนกระทั่งองค์การนาโต้ (NATO) ได้มีการจัดสัมมนาวิศวกรรมซอฟต์แวร์ขึ้น ครั้งแรกเมื่อปี ค.ศ. 1968 ณ ประเทศเยอรมัน  
→ จึงเป็นการจุดประกายให้ศาสตร์ทางด้านวิศวกรรมซอฟต์แวร์และซอฟต์แวร์คอมพิวเตอร์เป็นที่รู้จักมากยิ่งขึ้น

## 2. ช่วงกลางของวิศวกรรมซอฟต์แวร์ (1965 – 1985)

- ช่วงเริ่มต้นมุ่งเน้นการผลิตซอฟต์แวร์เป็นจำนวนมาก โดยไม่ได้คำนึงถึง
  1. ต้นทุนการผลิต
  2. งบประมาณที่ใช้
  3. หรือแม้กระทั่งข้อจำกัดเรื่องของเวลาในการพัฒนาซอฟต์แวร์
- **โดยในช่วงเวลาที่ยากลำบากนี้ทำให้เกิดวิกฤตการณ์**
  - จึงเป็นเหตุให้ต้องจ้างนักวิศวกรซอฟต์แวร์ที่มีความรู้ มีอาชีพที่สามารถผลิตซอฟต์แวร์ที่มีคุณภาพสูง
- **ตัวอย่าง** ซอฟต์แวร์ระบบปฏิบัติการ OS/360
  - ระยะเวลาในการผลิตที่นานถึง 10 ปี
  - มูลค่าการลงทุนที่สูงและ
  - มีจำนวนนักพัฒนาซอฟต์แวร์จำนวนมาก

แต่เมื่อผลิตเสร็จสิ้นแล้วกลับใช้งานซอฟต์แวร์ได้ไม่ดีเท่าที่ควรจะเป็น



# วัฒนาการของการพัฒนาซอฟต์แวร์ (2/4)

## The IBM System/360 (S/360)

- A family of mainframe computer systems
- Announced by IBM on April 7, 1964,  
Delivered between 1965 and 1978.
- The first family of computers **designed**
  - To cover both commercial and scientific applications and
  - To cover a complete range of applications from small to large.
- The design distinguished between architecture and implementation,
  - Allowing IBM to release a suite of compatible designs at different prices.
  - All but the only partially compatible Model 44 and the most expensive systems use microcode to implement the instruction set, which features 8-bit byte addressing and binary, decimal, and hexadecimal floating-point calculations.



IBM System/360 Model 30 central processor unit (CPU)

Also known as	S/360
Developer	IBM
Manufacturer	IBM
Product family	See <a href="#">table of models</a>
Type	Mainframe computer
Release date	April 7, 1964
Discontinued	1978
Media	7-track tape · 9-track tape · DASDs · paper tape · printed paper · punched cards
Operating system	BOS/360 · TOS/360 · DOS/360 · OS/360 · TSS/360
Memory	8 KB – 9 MB (core memory)
Predecessor	700/7000 series
Successor	System/370
Related	System/360 architecture



## วิวัฒนาการของการพัฒนาซอฟต์แวร์ (3/4)

### ประวัติวิศวกรรมซอฟต์แวร์ | History of software engineering

- The notion of **software engineering** was first proposed in 1968 at a conference held to discuss what was then called the **software crisis** (Naur and Randell 1969).
  - The **“Software Crisis”** – First identified at NATO Conference, Garmisch, Germany, 1968
  - –Characterized by software which is of
    - (1) Poor Quality (unreliable), (2) Over Budget, (3) Delivered late
- Throughout the 1970s and 1980s, a variety of **new software engineering techniques** and **methods** were developed, such as
  - structured programming, information hiding, and object-oriented development.
  - Tools and standard notations were developed which are the basis of today’s **software engineering**.



# วิวัฒนาการของการพัฒนาซอฟต์แวร์ (4/4)

## 3. ช่วงปี ค.ศ.1985 – ปัจจุบัน

- การพัฒนาทางด้านซอฟต์แวร์ยังคงดำเนินมาอย่างต่อเนื่อง
- มีงานวิจัยเป็นเครื่องมือผลักดันให้การผลิตซอฟต์แวร์นั้นเติบโตขึ้น
- มีการนำเทคโนโลยีสมัยใหม่เข้ามาช่วยในการผลิต รวมถึงวิธีการพัฒนาซอฟต์แวร์รูปแบบใหม่ กล่าวได้ว่าเป็นยุคของการแก้ปัญหาวิกฤตซอฟต์แวร์อย่างแท้จริง

## ปัจจัยที่เป็นแรงขับเคลื่อนของการพัฒนาซอฟต์แวร์ในช่วงนี้

### 1. เครื่องมือ

มีการคิดค้นเครื่องมือและพัฒนาเครื่องมือเพื่ออำนวยความสะดวกในการพัฒนาซอฟต์แวร์

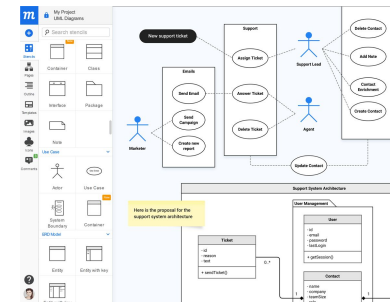
**ตัวอย่าง** เคสทูล (Case Tool) ยูเอ็มแอล (UML)

รวมถึงมาตรฐานต่างๆที่กำหนดคุณภาพของซอฟต์แวร์

### 2. กระบวนการ

มีการกำหนดระเบียบวิธีในการพัฒนาซอฟต์แวร์ให้มีคุณภาพ

มีกระบวนการในการปรับปรุงคุณภาพซอฟต์แวร์



# วัฏจักรชีวิตของการพัฒนาซอฟต์แวร์ | Software Development Life Cycle

- **วัฏจักรชีวิตของการพัฒนาซอฟต์แวร์** คือกระบวนการเชิงตรรกะ (Logical Process) ในการพัฒนาซอฟต์แวร์หรือระบบ เพื่อแก้ไขปัญหาและพัฒนาการทำงาน
- **Software Development Life Cycle : SDLC**
  - A process followed for a software project, within a software organization.
  - Consists of a detailed plan describing how to
    - develop,
    - maintain,
    - replace and alter or enhance specific software.
  - The life cycle defines a methodology for improving the quality of software and the overall development process.

# วัฏจักรชีวิตของการพัฒนาซอฟต์แวร์ | Software Development Life Cycle

วัฏจักรชีวิตของการพัฒนาซอฟต์แวร์ คือ  
กระบวนการเชิงตรรกะ (Logical Process)  
ในการพัฒนาซอฟต์แวร์หรือระบบ เพื่อแก้ไขปัญหา  
และ พัฒนางาน

## A typical SDLC consists of the 6 stages


- Stage 1: Planning and Requirement Analysis
- Stage 2: Defining Requirements
- Stage 3: Designing the Product Architecture
- Stage 4: Building or Developing the Product
- Stage 5: Testing the Product
- Stage 6: Deployment in the Market and Maintenance



## 5 ปัจจัยแห่งความสำเร็จ | Key Success Factors in Software Development Projects

1. **มีเป้าหมายที่ชัดเจน (Have Clear Objectives)** หนึ่งในปัญหาที่พบบ่อยที่สุด
  - การไม่มีวิสัยทัศน์ที่ชัดเจนสำหรับผลิตภัณฑ์และวัตถุประสงค์ที่ชัดเจนสำหรับวงจรการพัฒนา
2. **มีทีมที่มีวุฒิภาวะ (Have a Mature Team)**
  - วุฒิภาวะของทีมของคุณมีบทบาทสำคัญในความสำเร็จโดยรวมของโครงการ ทีมต้องมีทีมที่สอดคล้องกันจึงจะจัดการโครงการของคุณตั้งแต่ต้นจนจบ
3. **ใช้แนวทางปฏิบัติในการจัดการโครงการ (Apply Relevant Project Management Practices)**
  - ส่วนที่สำคัญที่สุด คือ การเลือกกลยุทธ์การจัดการโครงการที่เหมาะสม
4. **สร้างการสื่อสารที่มีประสิทธิภาพ (Establish Effective Communication)**
  - การสื่อสารเป็นปัจจัยที่สำคัญที่กำหนดประสิทธิภาพของความร่วมมือกับลูกค้า
  - ไม่ว่าคุณจะทำงานกับทีมภายในองค์กรหรือทีมนักพัฒนาจากระยะไกลที่อยู่คนละซีกโลก เราจำเป็นต้องมีช่องทางการสื่อสารที่ชัดเจนและครอบคลุม
5. **มั่นใจในคุณภาพ (Assure the Quality)**
  - สิ่งนี้ชัดเจนแต่หลายบริษัทยังคงจัดการกับปัญหานี้ในทางที่ผิด
  - คุณภาพของซอฟต์แวร์ของคุณเป็นสิ่งสำคัญยิ่งสำหรับประสบการณ์การใช้งานที่เหมาะสม

# สิ่งที่ทำให้โครงการซอฟต์แวร์ล้มเหลว

- 
1. ความต้องการของลูกค้าไม่ชัดเจน
  2. ลูกค้าบอกความต้องการไม่ได้
  3. วิเคราะห์ ออกแบบระบบ และ ทดสอบซอฟต์แวร์ไม่ดี
  4. เทคโนโลยีมีการเปลี่ยนแปลง
  5. การลาออก เปลี่ยนงาน
  6. ส่งมอบช้ากว่ากำหนด
  7. ใช้งบประมาณเกิน
  8. ซอฟต์แวร์ไม่เข้ากับสภาพแวดล้อม
  9. บำรุงรักษายาก ค่าใช้จ่ายสูง

# 10 Reasons Why Software Projects Fail

## โครงการซอฟต์แวร์ใหม่ (หรือสตาร์ทอัพ) เพลอร์เซ็นต์จำนวนมากล้มเหลว

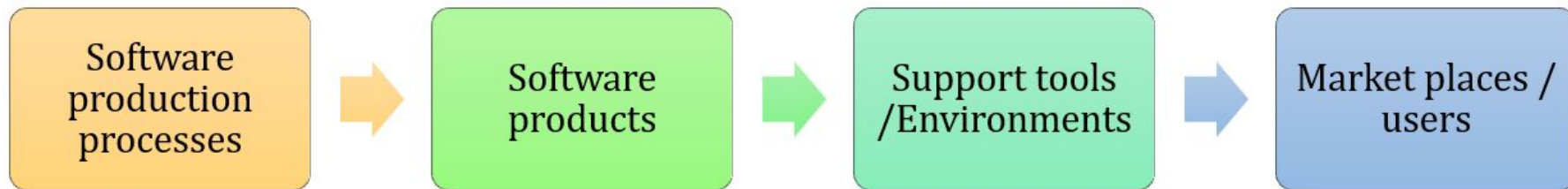
- \*จากประสบการณ์ 25 ปี c-sharpcorner ในการสร้างผลิตภัณฑ์ในอุตสาหกรรมซอฟต์แวร์
- 70% ถึง 80% หรือ 4 ใน 5 โครงการใหม่มักจะโครงการล้มเหลว

## 10 อันดับแรกที่ทำให้โครงการซอฟต์แวร์ล้มเหลว

1. Not having clear understanding of business requirements (**ไม่มีความเข้าใจข้อกำหนดทางธุรกิจ**)
2. Not involving end users in the early stage of development (**พัฒนาในช่วงแรกยังไม่เกี่ยวข้องกับผู้ใช้**)
3. Selecting a wrong team and outdated technology (**เลือกทีมที่ไม่ถูกต้องและเทคโนโลยีที่ล้าสมัย**)
4. Not delivering on time (**ส่งไม่ตรงเวลา**)
5. Underestimating cost of development (**ประเมินต้นทุนการพัฒนาต่ำเกินไป**)
6. Moving targets (**ย้ายเป้าหมาย**)
7. Poor planning and poor project management (**วางแผน/การจัดการโครงการที่ไม่ดี**)
8. Missing accountability (**ขาดความรับผิดชอบ**)
9. Bad user experience (UX/UI)  
**UX (User Experience) และ UI (User Interface)** จึงเป็นการออกแบบเพื่อให้ผู้ใช้งานเว็บไซต์และแอปพลิเคชันใช้งานได้ง่าย ไม่สับสนเส้นทาง สัมผัสได้ถึงประสบการณ์ที่ดีแบบไม่สะดุดตลอดการใช้งาน
10. Team not being vested in the project (**ทีมงานไม่ได้มีส่วนได้ส่วนเสียกับโครงการ**)

## องค์ประกอบของวิศวกรรมซอฟต์แวร์

- การวิศวกรรมซอฟต์แวร์เป็นกระบวนการผลิต (production) ที่ประกอบด้วย
  - กิจกรรมช่วงต่างๆ เพื่อสร้างผลิตภัณฑ์ซอฟต์แวร์ (software products)
  - การทำกิจกรรมในแต่ละช่วงอาศัยเทคนิคและเครื่องมือช่วยต่างๆ (support tools)
  - ที่นักวิชาการคอมพิวเตอร์และนักวิจัยได้เสนอไว้
  - การทำธุรกิจกับลูกค้า/ผู้ใช้งาน/แหล่งธุรกิจ





## 6 benefits of learning software engineering for project managers

1. Learn the basics of programming  
(เรียนรู้พื้นฐานการเขียนโปรแกรม)
2. Understand the Software Lifecycle  
(ทำความเข้าใจเกี่ยวกับวงจรชีวิตของซอฟต์แวร์)
3. Become a better leader (เป็นผู้นำทีมที่ดีขึ้น)
4. Get an understanding of emerging tech (ทำความเข้าใจเกี่ยวกับเทคโนโลยีที่เกิดขึ้นใหม่)
5. Understand ethical and security concerns (ทำความเข้าใจข้อกังวลด้านจริยธรรมและความปลอดภัย)
6. Learn how to be more innovative as a business (เรียนรู้วิธีสร้างนวัตกรรมให้มากขึ้นในฐานะธุรกิจ)



# คุณลักษณะของกระบวนการทางวิศวกรรมซอฟต์แวร์

1. **Understandability** : มีการนิยามขอบเขตของกระบวนการที่ชัดเจนและง่ายต่อการเข้าใจ
2. **Visibility** : ทำให้กิจกรรมกระบวนการชัดเจนที่สุดเพื่อสามารถมองเห็นจากภายนอกได้ชัดเจน
3. **Supportability** :  
เครื่องมือช่วยการวิศวกรรมซอฟต์แวร์ (CASE) สามารถ ช่วยสนับสนุนกิจกรรมกระบวนการในขอบเขตใด
4. **Acceptability** :  
กระบวนการที่กำหนดสามารถยอมรับและใช้โดยวิศวกร ซอฟต์แวร์ในการผลิตผลิตภัณฑ์ซอฟต์แวร์
5. **Reliability** : กระบวนการถูกออกแบบในแนวทางซึ่งความผิดพลาดของ กระบวนการถูกหลีกเลี่ยงก่อนที่จะส่งผลกระทบต่อความผิดพลาดของผลิตภัณฑ์ ซอฟต์แวร์
6. **Robustness** : กระบวนการสามารถทำงานต่อไปได้แม้ว่ามีปัญหาที่ไม่ คาดการณ์เกิดขึ้น
7. **Maintainability** : กระบวนการสามารถวิวัฒนาการเพื่อตอบสนองการ เปลี่ยนแปลงความต้องการขององค์กร
8. **Rapidity** :  
กระบวนการสามารถทำให้ส่งมอบผลิตภัณฑ์ได้เร็วขึ้นจากที่ รูปแบบคุณลักษณะของซอฟต์แวร์ (Software specifications) ถูกกำหนด

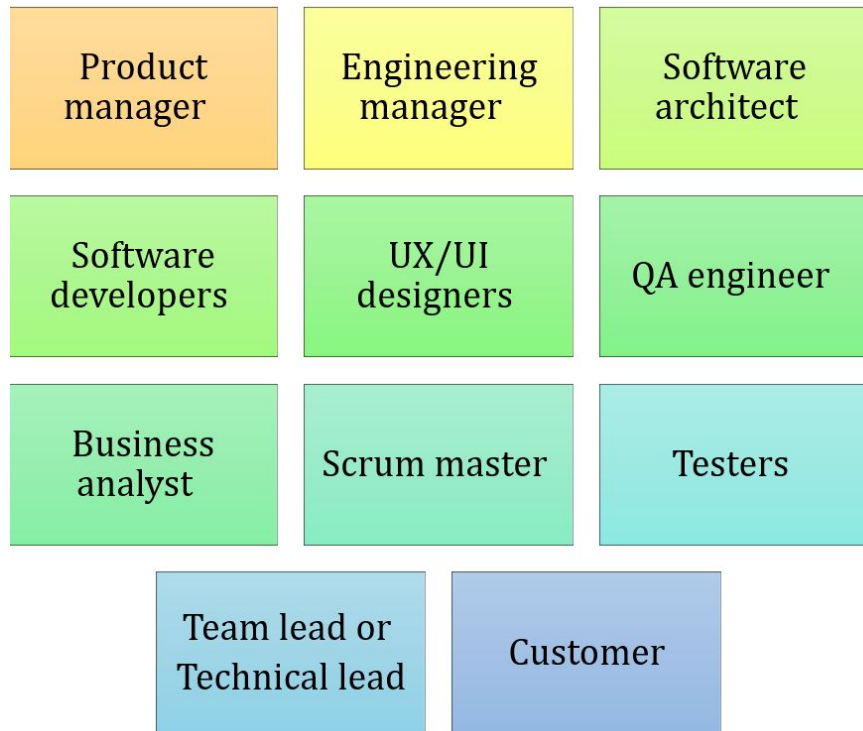
## Software Engineering is related to both Computer Science and Systems Engineering.

วิทยาการคอมพิวเตอร์ (Computer Science)	วิศวกรรมระบบ (System Engineering)
<ul style="list-style-type: none"> <li>• <b>วิทยาการคอมพิวเตอร์</b> เกี่ยวข้องกับทฤษฎีและวิธีการที่รองรับคอมพิวเตอร์และระบบซอฟต์แวร์ ในขณะที่วิศวกรรมซอฟต์แวร์เกี่ยวข้องกับปัญหาในทางปฏิบัติของการผลิตซอฟต์แวร์</li> <li>• <b>ความรู้บางอย่างเกี่ยวกับวิทยาการคอมพิวเตอร์</b> <ul style="list-style-type: none"> <li>- จำเป็นสำหรับวิศวกรซอฟต์แวร์ในลักษณะเดียวกัน</li> <li>- เช่น ความรู้ทางฟิสิกส์บางอย่างเป็นสิ่งจำเป็นสำหรับวิศวกรไฟฟ้า</li> <li>- ทฤษฎีวิทยาการคอมพิวเตอร์มักใช้ได้กับโปรแกรมที่มีขนาดค่อนข้างเล็ก</li> </ul> </li> <li>• <b>ทฤษฎีที่สวຍงามของวิทยาการคอมพิวเตอร์</b> <ul style="list-style-type: none"> <li>- มักไม่ค่อยเกี่ยวข้องกับปัญหาขนาดใหญ่และซับซ้อนที่ต้องใช้กระบวนการแก้ไขทางซอฟต์แวร์</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• <b>วิศวกรรมระบบ</b> เกี่ยวข้องกับทุกแง่มุมของการพัฒนาและวิวัฒนาการของระบบที่ซับซ้อน           <ul style="list-style-type: none"> <li>- ซึ่งซอฟต์แวร์มีบทบาทสำคัญ</li> </ul> </li> <li>• <b>วิศวกรรมระบบ</b> เกี่ยวข้องกับ           <ul style="list-style-type: none"> <li>- การพัฒนาฮาร์ดแวร์</li> <li>- การออกแบบนโยบายและกระบวนการ และ</li> <li>- การปรับใช้ระบบ ตลอดจนวิศวกรรมซอฟต์แวร์</li> </ul> </li> <li>• <b>วิศวกรรมระบบมีส่วนร่วมใน</b> <ul style="list-style-type: none"> <li>- การระบุระบบ</li> <li>- กำหนดสถาปัตยกรรมโดยรวม</li> <li>- จากนั้นจึงรวมส่วนต่าง ๆ เพื่อสร้างระบบที่เสร็จสมบูรณ์</li> </ul> </li> </ul>

## ความแตกต่าง CS & SE

เกณฑ์	CS	SE
ความซับซ้อนของระบบ	ต่ำ	สูง
จำนวนของนักพัฒนาซอฟต์แวร์	น้อย	มาก
เวลาในการพัฒนาซอฟต์แวร์	สั้น	นาน
ค่าใช้จ่ายในการบำรุงรักษาซอฟต์แวร์	ต่ำ	สูง
ความสามารถนำกลับมาใช้ใหม่	น้อย	มาก

# Typical Team Roles in Software Development



Ref' :

<https://alcor-bpo.com/recruitment-news/10-key-roles-in-a-software-development-team-who-is-responsible-for-what/>

# คุณภาพของซอฟต์แวร์ | Software Quality

**คุณภาพของซอฟต์แวร์ (Software Quality)** หมายถึง

- การวัดคุณภาพของซอฟต์แวร์ที่ได้ทำการผลิตให้มีมาตรฐานการผลิตตรงตามหลักสากล

โดยใช้เกณฑ์ CMMI ในการวัดคุณภาพซอฟต์แวร์

**CMMI : Capability Maturity Model Integration**

- เป็นมาตรฐานกระบวนการในการพัฒนางาน ถูกสร้างขึ้นที่ Software Engineering Institute, Carnegie Mellon University, USA
- เป็นมาตรฐานในการปรับปรุงคุณภาพซอฟต์แวร์ให้มีประสิทธิภาพ เป็นที่รู้จัก และยอมรับของสากล
- หากองค์กรใดได้รับ CMMI ถือว่าองค์กรนั้นมี product (โปรดัก) และกระบวนการพัฒนา product (โปรดัก) ที่มีประสิทธิภาพ เป็นที่น่าเชื่อถือของลูกค้า และเป็นตัวการันตีชิ้นงาน



## 8 หลักจริยธรรมที่ หน่วยงาน ACM/IEEE

### สมาคมคอมพิวเตอร์เอซีเอ็ม (Association for Computing Machinery: ACM)

คือ สมาคมระหว่างประเทศทางด้านคอมพิวเตอร์

1. **นักวิศวกรซอฟต์แวร์** ต้องปฏิบัติหน้าที่ โดยคำนึงถึงประโยชน์ส่วนรวมเป็นหลัก
2. **นักวิศวกรซอฟต์แวร์** ต้องคำนึงถึงความต้องการของลูกค้าและนายจ้าง โดยสอดคล้องกับประโยชน์ส่วนรวม
3. **นักวิศวกรซอฟต์แวร์** ต้องผลิตซอฟต์แวร์ตามหลักการมาตรฐานวิชาชีพ
4. **นักวิศวกรซอฟต์แวร์** ต้องตัดสินใจอย่างอิสระและรวมเป็นอันหนึ่งอันเดียวกัน โดยประเมินตามหลักการของมาตรฐานวิชาชีพ
5. ผู้จัดการโครงการและผู้นำสนับสนุนและเผยแพร่แนวคิดด้านจริยธรรม เพื่อบริหารโครงการซอฟต์แวร์และบำรุงรักษาซอฟต์แวร์
6. **นักวิศวกรซอฟต์แวร์** ต้องยึดมั่นในคุณธรรมและรักษาชื่อเสียงในวิชาชีพ โดยคำนึงถึงประโยชน์ส่วนรวม
7. **นักวิศวกรซอฟต์แวร์** ต้องมีความเป็นธรรมและให้การสนับสนุนเพื่อนร่วมงาน
8. **นักวิศวกรซอฟต์แวร์** ต้องมีการเรียนรู้เกี่ยวกับวิชาชีพตลอดชีวิต และให้การสนับสนุนแนวคิดด้านจริยธรรม เพื่อนำไปสู่การเป็นมืออาชีพ



**EDUCATING  
THE MIND**

WITHOUT

**EDUCATING  
THE HEART**

IS NO EDUCATION AT ALL.

---

**ARISTOTLE**

**Reference:**  
<https://www.redbubble.com/i/art-board-print/Educating-the-mind-without-educating-the-heart-is-no-education-at-all-by-proofreading/36956096.JUXJO>

Thank you  
for  
your attention.

“การให้ความรู้แก่สมองโดยที่หัวใจปราศจากการเรียนรู้ มีค่าเท่ากับการไม่ได้เรียนรู้อะไรเลย.”

-อริสโตเติล-

<https://www.goodreads.com/author/quotes/2192.Aristotle>