บทที่ 1
ความรู้เบื้องต้นเกี่ยวกับวิศวกรรมซอฟต์แวร์
(Introduction to Software Engineering)

วิศวกรรมซอฟต์แวร์ (Software Engineering) เป็นศาสตร์ที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์โดยตรง (Pressman, 2010) ในบทนี้กล่าวถึงความรู้เบื้องต้นเกี่ยวกับวิศวกรรมซอฟต์แวร์ โดยเริ่มจากนิยามและประเภทของซอฟต์แวร์ในส่วนต้น จากนั้นจะกล่าวถึงความหมายของวิศวกรรมซอฟต์แวร์ รวมทั้งสาขาย่อยและสายงานที่เกี่ยวข้อง และลำดับชั้นของวิศวกรรมซอฟต์แวร์ในส่วนท้าย

1.1 ซอฟต์แวร์ (Software)

คำนิยามของซอฟต์แวร์ (Pressman, 2010) มีดังต่อไปนี้

1) ซอฟต์แวร์เป็นชุดคำสั่งที่เมื่อมีการทำงานในระบบคอมพิวเตอร์แล้วสามารถให้คุณสมบัติ (Feature) ฟังก์ชัน (Function) และสมรรถนะ (Performance) ตามที่ผู้ใช้ต้องการ
2) ซอฟต์แวร์เป็นโครงสร้างข้อมูลซึ่งมีความเพียงพอที่จะจัดการสารสนเทศ (Information) ได้
3) ซอฟต์แวร์เป็นข้อมูลในลักษณะพรรณนาที่อาจอยู่ในรูปสิ่งของที่จับต้องได้หรืออยู่ในรูปแบบเสมือนที่ใช้อธิบายกระบวนการหรือการทำงานของโปรแกรม

ซอฟต์แวร์เป็นสิ่งที่สร้างขึ้นในลักษณะที่ต่างกับวัตถุจริง โดยมีคุณลักษณะต่างจากฮาร์ดแวร์ (Hardware) ดังนี้

1) ใช้คำว่าพัฒนาซอฟต์แวร์แทนการผลิตซอฟต์แวร์
2) ซอฟต์แวร์ไม่มีการเสื่อมสภาพ ซึ่งต่างกับฮาร์ดแวร์ที่มีการเสื่อมสภาพตามกาลเวลา
3) ซอฟต์แวร์มักสร้างแบบสั่งทำ แม้ว่าจะมีลักษณะที่เป็นส่วนประกอบหรือคอมโพเนนท์ (Component) ก็ตาม
4) ซอฟต์แวร์ในยุคปัจจุบันมีการพัฒนาให้อยู่ในรูปแบบของการบริการมากขึ้น

ซอฟต์แวร์มีความสำคัญในลักษณะที่ตัวซอฟต์แวร์เองเป็นเทคโนโลยี รวมทั้งเป็นองค์ประกอบสำคัญของเทคโนโลยีอื่นที่เกี่ยวข้องกับธุรกิจ วิทยาศาสตร์ วิศวกรรมศาสตร์ อีกทั้งยังสามารถทำให้เกิดการสร้างเทคโนโลยีอื่นๆ ได้ เช่น พันธุวิศวกรรม นาโนเทคโนโลยี เป็นต้น เมื่อความสำคัญของซอฟต์แวร์เพิ่มขึ้น การพัฒนาซอฟต์แวร์ขนาดใหญ่ที่มีโครงสร้างซับซ้อนก็เกิดขึ้น กลไกการพัฒนาซอฟต์แวร์จึงมีความจำเป็นเพื่อที่จะช่วยให้สามารถพัฒนาซอฟต์แวร์ได้ง่ายขึ้น รวดเร็วขึ้น ราคาถูกลง และยังคงคุณภาพสูง รวมทั้งสามารถบำรุงรักษาซอฟต์แวร์ได้สะดวก ซึ่งกลไกดังกล่าวเรียกว่า “วิศวกรรมซอฟต์แวร์” (Pressman, 2010; Sommerville, 2010)

1.2 ประเภทของซอฟต์แวร์

ในปัจจุบันมีการพัฒนาซอฟต์แวร์กันอย่างกว้างขวาง ทำให้เกิดซอฟต์แวร์หลายประเภทซึ่งพัฒนาขึ้นเพื่อตอบสนองความต้องการของผู้ใช้ที่แตกต่างกัน โดยประเภทของซอฟต์แวร์อาจแบ่งได้ดังต่อไปนี้

1.2.1 ซอฟต์แวร์ระบบ (System Software)

ซอฟต์แวร์ระบบเป็นซอฟต์แวร์ที่พัฒนาขึ้นเพื่อใช้ในการรองรับการทำงานและจัดการทรัพยากรพื้นฐานของคอมพิวเตอร์ ได้แก่

1.2.1.1 ระบบปฏิบัติการ (Operating System)

ระบบปฏิบัติการเป็นซอฟต์แวร์ที่มีหน้าที่จัดการทรัพยากรของคอมพิวเตอร์และให้บริการแก่ซอฟต์แวร์ประยุกต์ (Application Software) โดยจะเป็นตัวกลางระหว่างฮาร์ดแวร์และซอฟต์แวร์ประยุกต์ (Tanenbaum, 2007) ระบบปฏิบัติการบนคอมพิวเตอร์ที่ใช้กันมากในปัจจุบัน ได้แก่ Microsoft Windows (Microsoft Corp., 2012), Linux (Torvalds & Diamond, 2002; Bovet & Cesati, 2005) และ Mac OS X (โอเอส เท็น) (Apple Inc., 2012) เป็นต้น ส่วนระบบปฏิบัติการบนอุปกรณ์เคลื่อนที่ส่วนบุคคล เช่น Android (Google Inc., 2008), iOS สำหรับอุปกรณ์กลุ่ม iPhone และ iPad (Apple Inc., 2007), BlackBerry OS (Research In Motion, 1999), Windows Phone (Microsoft Corp., 2010) และ MeeGo (Linux Foundation, 2011) เป็นต้น แนวโน้มทางการพัฒนาระบบปฏิบัติการในอนาคตอาจมีการรวมเอาเว็บเบราเซอร์เข้าเป็นส่วนหนึ่งของซอฟต์แวร์ระบบ ซึ่งปัจจุบันสามารถพบได้ในระบบ ปฏิบัติการ Google Chrome OS (Google Inc., 2011)

1.2.1.2 ดีไวซ์ไดรเวอร์ (Device Driver)

ดีไวซ์ไดรเวอร์เป็นซอฟต์แวร์ที่ช่วยให้โปรแกรมคอมพิวเตอร์อื่นๆ สามารถติดต่อกับอุปกรณ์ภายนอกได้ การพัฒนาดีไวซ์ไดรเวอร์ขึ้นกับแต่ละสถาปัตยกรรมของระบบปฏิบัติการ เช่น ดีไวซ์ไดรเวอร์สำหรับอุปกรณ์ USB จะทำให้ระบบปฏิบัติการสามารถติดต่อกับแฟลชไดรว์ (Flash Drive) ที่ผู้ใช้เสียบเข้ากับพอร์ต USB ได้ โดยดีไวซ์ไดรเวอร์บน Linux (Corbet, Rubini & Kroah-Hartman, 2005) จะแตกต่างกับดีไวซ์ไดรเวอร์บน Windows (Baker & Lazano, 2000; Viscarola & Mason, 2006) สำหรับอุปกรณ์เดียวกัน เป็นต้น

1.2.1.3 ซอฟต์แวร์แม่ข่าย (Server Software)

ซอฟต์แวร์แม่ข่ายเป็นซอฟต์แวร์ที่ช่วยให้เครือข่ายสามารถให้บริการต่างๆ ได้ เช่น Apache HTTP Server (Apache Software Foundation, 2008) เป็นเว็บเซิร์ฟเวอร์ (Web Server) ที่ให้บริการเมื่อลูกข่ายร้องขอ (Request) เพื่อที่จะเข้าถึงหน้าเว็บผ่านทางเว็บเบราเซอร์ เป็นต้น

1.2.1.4 ระบบจัดการหน้าต่าง (Window System หรือ Window Manager)

ระบบจัดการหน้าต่างเป็นซอฟต์แวร์ที่ใช้สำหรับติดต่อกับผู้ใช้ โดยปกติจะเป็นส่วนหนึ่งของสภาวะแวดล้อมเดสทอป (Desktop Environment) โปรแกรมกลุ่มนี้ทำหน้าที่รับการโต้ตอบจากผู้ใช้และทำการแสดงผลในรูปแบบของหน้าต่าง ส่วนเมนูและชิ้นส่วนอื่นๆ ที่มองเห็นได้ ในระบบปฏิบัติการ Windows นั้น ระบบจัดการหน้าต่างจะไม่สามารถแยกออกมาได้อย่างชัดเจน แต่สำหรับระบบปฏิบัติการตระกูล Linux ได้มีการแยกสถาปัตยกรรมส่วนนี้ออกมาอย่างชัดเจน เช่น ระบบจัดการหน้าต่าง GNOME (GNOME Foundation, 1999) หรือ KDE (KDE e.V., 1998) เป็นต้น

1.2.1.5 ซอฟต์แวร์อรรถประโยชน์ (Utility Software)

ซอฟต์แวร์อรรถประโยชน์เป็นซอฟต์แวร์ที่ออกแบบมาเพื่อช่วยในการวิเคราะห์ ปรับแต่ง และบำรุงรักษาคอมพิวเตอร์ เช่นกลุ่มคำสั่ง GNU Coreutils (Free Software Foundation, 2001) ที่ใช้กันอย่างแพร่หลายบนระบบปฏิบัติการ Linux

1.2.2 ซอฟต์แวร์ประยุกต์ (Application Software)

ซอฟต์แวร์ประยุกต์เป็นซอฟต์แวร์ที่ให้บริการผู้ใช้ตามกลุ่มงานที่ซอฟต์แวร์นั้นๆ ถูกออกแบบมารองรับ ซอฟต์แวร์กลุ่มนี้แตกต่างจากซอฟต์แวร์ระบบ เนื่องจากสามารถทำงานเฉพาะเจาะจงลงไปตามประเภทของงานที่ต้องการ เช่น

- ซอฟต์แวร์ธุรกิจ รวมถึงซอฟต์แวร์กลุ่มบริหารทรัพยากรองค์กร เช่น Apache OFBiz (Apache Software Foundation, 2006) หรือซอฟต์แวร์กลุ่มบริหารลูกค้าสัมพันธ์ (Customer Relationship Management – CRM) เช่น Salesforce (Salesforce.com Inc., 2009) เป็นต้น
- ซอฟต์แวร์เกมส์
- ซอฟต์แวร์สำหรับใช้งานอินเตอร์เนต เช่น เว็บเบราเซอร์, Instant Messenger เป็นต้น
- ซอฟต์แวร์ฐานข้อมูล
- ซอฟต์แวร์ชุดสำนักงานอัตโนมัติ เช่น Word Processing, Spreadsheet เป็นต้น

1.2.3 เว็บแอพพลิเคชัน (Web Application)

เว็บแอพพลิเคชันเป็นซอฟต์แวร์ที่ทำงานในเว็บเบราเซอร์ ซึ่งแตกต่างจากแอพพลิเคชันทั่วๆ ไปที่ทำงานบนสภาพแวดล้อมเดสทอป เว็บแอพพลิเคชันนี้จะแตกต่างกับเว็บไซต์ธรรมดาที่เน้นการนำเสนอเนื้อหาเป็นหลัก ในขณะที่ซอฟต์แวร์ในกลุ่มเว็บแอพพลิเคชันมักสร้างขึ้นเพื่อย้ายการทำงานที่เคยทำบนเดสทอปมาสู่เว็บ เช่น

- ซอฟต์แวร์ธุรกิจ เช่น Salesforce.com เป็นซอฟต์แวร์ธุรกิจประเภท CRM ที่เป็นเว็บแอพพลิเคชัน
- ซอฟต์แวร์เกมส์ เช่น เกมส์ที่สร้างด้วย Adobe Flash
- ซอฟต์แวร์ชุดสำนักงานอัตโนมัติ เช่น Google Docs

1.2.4 แอพพลิเคชันบนอุปกรณ์มือถือ (Mobile Applications)

แอพพลิเคชันที่ทำงานบนมือถือหรืออุปกรณ์ประเภทแท็บเล็ต (Tablet) จะมีการโต้ตอบกับผู้ใช้ในลักษณะเฉพาะ รวมถึงการใช้ความสามารถบางอย่างของตัวอุปกรณ์นั้นๆ เพื่อเพิ่มประสิทธิภาพในการใช้งาน เช่น การเปลี่ยนทิศทางการแสดงผลเมื่อผู้ใช้หมุนตัวอุปกรณ์ การระบุตำแหน่งของผู้ใช้ผ่านโมดูลจีพีเอส (GPS – Global Positioning System) ที่มีอยู่ในตัวอุปกรณ์ หรือตอบโต้กับผู้ใช้ในลักษณะต่างๆ กันผ่านจอชนิดสัมผัส เป็นต้น

1.2.5 ซอฟต์แวร์เชิงสายการผลิต (Product-line Software)

ซอฟต์แวร์เชิงสายการผลิตเป็นกลุ่มของซอฟต์แวร์ที่ถูกออกแบบมาให้มีความสามารถเฉพาะอย่างสำหรับลูกค้าที่มีความต้องการต่างๆ กัน ซอฟต์แวร์ประเภทนี้มักมีตลาดอยู่ในกลุ่มเฉพาะ เช่น โปรแกรมคุมคลังสินค้า โปรแกรมจัดการการผลิต เป็นต้น โดยลักษณะเฉพาะของซอฟต์แวร์ประเภทนี้ คือ ในซอฟต์แวร์ตัวหนึ่งๆ จะมีฟังก์ชันการทำงานร่วมกันและมีส่วนที่ต่างกันออกไปเฉพาะสำหรับลูกค้าแต่ละราย (Clements & Northrop, 2001)

1.2.6 ซอฟต์แวร์ประเภทฝังตัว (Embedded Software)

ซอฟต์แวร์ประเภทฝังตัวเป็นซอฟต์แวร์ที่อยู่ในอุปกรณ์ประเภทฝังตัวชนิดต่างๆ เช่น ไมโครคอนโทรลเลอร์ (Micro Controller) (Barr & Massa, 2006) เพื่อสร้างแอพพลิเคชันที่ส่วนใหญ่ใช้ติดต่อกับอุปกรณ์ภายนอก โดยทั่วไปซอฟต์แวร์ประเภทนี้มักใช้รับข้อมูลจากเซนเซอร์ตรวจจับประเภทต่างๆ ซึ่งอาจเป็นการตรวจจับอุณหภูมิ ความชื้น หรือรับสัญญาณ GPS จากนั้นทำการประมวลผลเพื่อทำงานเฉพาะอย่าง ในทางกลับกันหากอุปกรณ์ฝังตัวนั้นมีสมรรถนะสูงมากๆ ซอฟต์แวร์ในอุปกรณ์ดังกล่าวอาจจะซับซ้อนเกือบเทียบเท่ากับซอฟต์แวร์ระบบที่ใช้ในคอมพิวเตอร์ทั่วไป เนื่องจากข้อจำกัดของทรัพยากรที่มีในอุปกรณ์ประเภทฝังตัวรวมไปถึงอายุการใช้งานของแบตเตอรี่ในอุปกรณ์เหล่านั้นทำให้การพัฒนาซอฟต์แวร์ประเภทฝังตัวมีลักษณะเฉพาะแตกต่างจากซอฟต์แวร์ประเภทอื่นๆ อย่างชัดเจน เช่น ความสามารถในการประหยัดพลังงานเมื่อระบบว่าง เป็นต้น

1.2.7 ซอฟต์แวร์เชิงวิศวกรรมศาสตร์และวิทยาศาสตร์ (Engineering and Scientific Software)

ซอฟต์แวร์เชิงวิศวกรรมศาสตร์และวิทยาศาสตร์เป็นซอฟต์แวร์ที่เน้นสนับสนุนการทำงานและการคำนวณเชิงวิศวกรรมหรือวิทยาศาสตร์ โดยจะเน้นถึงความถูกต้องในการคำนวณ การสนับสนุนการแก้สมการประเภทต่างๆ การประมวลผลเชิงสัญลักษณ์ การคำนวณสมรรถนะสูง รวมถึงการคำนวณเชิงขนาน เป็นต้น ในงานบางประเภทอาจเป็นการประมวลผลข้อมูลขนาดมหาศาลที่ต้องการการจัดข้อมูลชนิดพิเศษ ซึ่งทำให้บางส่วนของซอฟต์แวร์ประเภทนี้เกี่ยวข้องกับซอฟต์แวร์ระบบ เช่น ระบบการเก็บข้อมูลของโครงการ ALICE (European Organization for Nuclear Research, 2008) โครงการย่อยของ CERN เป็นต้น

1.2.8 ซอฟต์แวร์ปัญญาประดิษฐ์ (Artificial Intelligence Software)

ซอฟต์แวร์ปัญญาประดิษฐ์เป็นซอฟต์แวร์ที่พัฒนาเพื่อให้คอมพิวเตอร์มีความสามารถในการคิดเองได้ (Russell & Norvig, 2009) โดยซอฟต์แวร์กลุ่มนี้อาจใช้เป็นส่วนประกอบในซอฟต์แวร์แอพพลิเคชันเชิงธุรกิจอื่นเพื่อช่วยสนับสนุนการตัดสินใจของผู้บริหาร เช่น ซอฟต์แวร์วิเคราะห์ตัวหนังสือเพื่ออ่านออกเสียง ซอฟต์แวร์ควบคุมการทำงานของหุ่นยนต์ เป็นต้น ในบางลักษณะซอฟต์แวร์ปัญญาประดิษฐ์มักจะเกี่ยวข้องกับโครงสร้างข้อมูลเฉพาะแบบและอาจต้องการการคำนวณสมรรถนะสูง ซึ่งซอฟต์แวร์ในลักษณะดังกล่าวจะคาบเกี่ยวกับซอฟต์แวร์ทางวิทยาศาสตร์

จากประเภทซอฟต์แวร์ที่กล่าวมาอาจสรุปได้ว่าซอฟต์แวร์แต่ละประเภทแม้จะมีลักษณะที่ต่างกันแต่ก็มีความเกี่ยวข้องกันในหลายแง่มุม อีกทั้งซอฟต์แวร์บางกลุ่มยังมีลักษณะที่คาบเกี่ยวกันและเป็นที่น่าสนใจว่าความคาบเกี่ยวดังกล่าวอาจเพิ่มขึ้นหรือลดลงได้ ซึ่งขึ้นกับการเปลี่ยนแปลงของเทคโนโลยีด้านซอฟต์แวร์ที่ดำเนินต่อไปอย่างไม่หยุดนิ่ง การปรับใช้วิศวกรรมซอฟต์แวร์กับซอฟต์แวร์บางประเภทในบางยุคอาจนำมาใช้ได้กับซอฟต์แวร์อีกประเภทหนึ่งในอีกยุคหนึ่งก็เป็นได้

1.3 วิศวกรรมซอฟต์แวร์ (Software Engineering)

วิศวกรรมซอฟต์แวร์เป็นศาสตร์เชิงประยุกต์เกี่ยวกับแนวทางที่เป็นระบบ มีระเบียบแบบแผน และสามารถวัดได้ในเชิงปริมาณต่อการพัฒนาและการบำรุงรักษาซอฟต์แวร์ นั่นคือการประยุกต์หลักวิศวกรรมศาสตร์เข้ากับการพัฒนาซอฟต์แวร์ (Pressman, 2010; Sommerville, 2010) โดยสาขาย่อยในวิศวกรรมซอฟต์แวร์สามารถจำแนกได้ดังนี้

1.3.1 ความต้องการเชิงซอฟต์แวร์ (Software Requirements)

ความต้องการเชิงซอฟต์แวรเป็นการศึกษาเกี่ยวกับความต้องการเชิงซอฟต์แวร์ รวมถึงแนวทางการสื่อสาร รวบรวม และจัดเก็บความต้องการเชิงซอฟต์แวร์จากกลุ่มเป้าหมาย

1.3.2 การออกแบบซอฟต์แวร์ (Software Design)

การออกแบบซอฟต์แวร์เป็นการศึกษาเกี่ยวกับการวิเคราะห์และออกแบบความต้องการและข้อกำหนดทางซอฟต์แวร์ให้สามารถนำไปพัฒนาเป็นซอฟต์แวร์ที่สามารถทำงานได้

1.3.3 การพัฒนาซอฟต์แวร์ (Software Development)

การพัฒนาซอฟต์แวร์เป็นการศึกษากระบวนการการพัฒนาซอฟต์แวร์ โดยเกี่ยวข้องกับเทคนิคการสร้างซอฟต์แวร์อย่างเป็นระบบ มีการเลือกใช้ภาษาโปรแกรม (Programming Language) เฟรมเวิร์ค (Framework) และดีไซน์แพตเทิร์น (Design Pattern) (Gamma et al., 1994) ที่เหมาะสม

1.3.4 การทดสอบซอฟต์แวร์ (Software Testing)

การทดสอบซอฟต์แวร์เป็นการศึกษาเกี่ยวกับกลไกการทดสอบซอฟต์แวร์ทั้งในระดับที่ใกล้การพัฒนามากที่สุด ซึ่งสามารถเข้าถึงต้นรหัส (Source Code) ของตัวซอฟต์แวร์ได้ และในระดับที่ไกลที่สุด คือ การทดสอบซอฟต์แวร์ที่กำลังทำงานอยู่ก่อนนำไปใช้จริง รวมทั้งแง่มุมการทดสอบต่างๆ เช่น ประสิทธิภาพ และการรับโหลด เป็นต้น (Paton, 2005; Pressman, 2010)

1.3.5 การบำรุงรักษาซอฟต์แวร์ (Software Maintenance)

การบำรุงรักษาซอฟต์แวร์เป็นการศึกษาเกี่ยวกับการบำรุงรักษาซอฟต์แวร์ การใช้เครื่องมือหรือภาษาในการแก้ไขบำรุงรักษาหลังจากครบระยะการใช้งานซอฟต์แวร์ (April & Abran, 2008; Pressman, 2010) กลไกในการบำรุงรักษาอาจรวมถึงเทคโนโลยีการนำซอฟต์แวร์รุ่นใหม่ไปใช้แทนรุ่นเก่าโดยที่ไม่จำเป็นต้องหยุดการทำงานของระบบ

1.3.6 การจัดการการปรับแต่งซอฟต์แวร์ (Software Configuration Management)

การจัดการปรับแต่งซอฟต์แวร์เป็นการศึกษาเกี่ยวกับการจัดการรุ่นของต้นรหัสซอฟต์แวร์ติดตามการเปลี่ยนแปลงของต้นรหัสและการบำรุงรักษาส่วนต่อยอดจากต้นรหัสที่มีอยู่แล้ว (Aiello & Sachs, 2010)

1.3.7 การจัดการเชิงวิศวกรรมซอฟต์แวร์ (Software Engineering Management)

การจัดการเชิงวิศวกรรมซอฟต์แวร์เป็นการศึกษาการจัดการระบบซอฟต์แวร์ในลักษณะเดียวกันกับการจัดการโครงการ โดยอาจจะมีการนำโปรแกรมสำหรับควบคุมจัดการโครงการมาใช้ดูแลการพัฒนาซอฟต์แวร์

1.3.8 กระบวนการพัฒนาซอฟต์แวร์ (Software Development Process)

กระบวนการพัฒนาซอฟต์แวร์เป็นการศึกษากระบวนการในการพัฒนาซอฟต์แวร์ที่ครอบคลุมตั้งแต่การรวบรวมความต้องการไปจนถึงการดูแลรักษาซอฟต์แวร์ การศึกษากระบวนการพัฒนาซอฟต์แวร์มักทำในลักษณะการสร้างแบบจำลองของกระบวนการจากการใช้งานจริง ตัวอย่างแบบจำลองกระบวนการพัฒนาซอฟต์แวร์ เช่น แบบจำลองน้ำตก (Waterfall) แบบจำลองสไปรัล (Spiral) และการพัฒนาแบบอาไจล์ (Agile) เป็นต้น

1.3.9 เครื่องมือทางวิศวกรรมซอฟต์แวร์ (Software Engineering Tools)

เครื่องมือทางวิศวกรรมซอฟต์แวร์เป็นการศึกษาการสร้างและพัฒนาเครื่องมือเพื่อช่วยในการพัฒนาซอฟต์แวร์ กลุ่มเครื่องมือดังกล่าวจะเรียกว่าเครื่องมือช่วยเหลือทางวิศวกรรมซอฟต์แวร์ (Computer-Aided Software Engineering Tools - CASE Tools)

1.3.10 คุณภาพซอฟต์แวร์ (Software Quality)

คุณภาพซอฟต์แวร์เป็นการศึกษาเกี่ยวกับคุณภาพของซอฟต์แวร์และเทคนิคการวัดคุณภาพทั้งในแง่มุมของคุณภาพในการออกแบบ คุณภาพของการพัฒนาซอฟต์แวร์ให้ได้ตามการออกแบบ คุณภาพของต้นรหัส และคุณภาพของผลิตภัณฑ์เมื่อเสร็จสิ้นการพัฒนาในขั้นตอนสุดท้าย เป็นต้น

1.4 สายงานด้านวิศวกรรมซอฟต์แวร์

สายงานทางด้านวิศวกรรมซอฟต์แวร์เป็นกลุ่มอาชีพที่กว้าง หลากหลาย และเป็นที่ต้องการในอุตสาหกรรมมากที่สุดแขนงหนึ่ง งานทางด้านวิศวกรรมซอฟต์แวร์ครอบคลุมงานในลักษณะนักวิเคราะห์ นักพัฒนา ผู้ดูแลและสนับสนุน รวมไปถึงสถาปนิกซอฟต์แวร์และผู้ที่มีบทบาทเฉพาะในแต่ละระยะของกระบวนการพัฒนาซอฟต์แวร์ เป็นต้น (Pressman, 2010) สายงานทางด้านวิศวกรรมซอฟต์แวร์ที่น่าสนใจมีดังต่อไปนี้

1.4.1 นักวิเคราะห์ระบบ (System Analyst)

นักวิเคราะห์ระบบเป็นผู้ที่มีบทบาทสำคัญในการวิเคราะห์ปัญหา วางแผนการแก้ปัญหา รวมทั้งแนะนำซอฟต์แวร์ให้แก่ผู้ใช้ นักวิเคราะห์ระบบมีบทบาทที่สำคัญอีกอย่างหนึ่งก็คือ ประสานงานให้การพัฒนาซอฟต์แวร์เป็นไปตามความต้องการเชิงธุรกิจและความต้องการอื่นๆ นักวิเคราะห์ระบบมักเกี่ยวข้องกับการแก้ปัญหาในหลายแง่มุมจึงจำเป็นต้องมีความรู้ในภาษาโปรแกรมหลายภาษา ระบบปฏิบัติการหลายประเภท และแพล็ตฟอร์มต่างๆ ของฮาร์ดแวร์ นอกจากนี้นักวิเคราะห์ระบบมักจะต้องเป็นผู้ที่เขียนความต้องการเชิงซอฟต์แวร์ให้อยู่ในรูปของข้อกำหนดเชิงเทคนิค เช่น เอกสารการวิเคราะห์และการออกแบบต่างๆ ซึ่งอาจจะอยู่ในรูปแบบของผังงาน (Flow Chart) หรือยูสเคส (Use Case) เป็นต้น

1.4.2 สถาปนิกซอฟต์แวร์ (Software Architect)

สถาปนิกซอฟต์แวร์เป็นผู้ที่รับผิดชอบเกี่ยวกับการกำหนดขอบเขตของการเลือกเทคโนโลยีและเฟรมเวิร์คระหว่างการพัฒนาซอฟต์แวร์ และเลือกวิธีการที่ใช้เป็นมาตรฐานในการพัฒนา ทั้งนี้อาจรวมถึงเป็นผู้สร้างหรือกำหนดเฟรมเวิร์คสำหรับแอพพลิเคชันที่กำลังพัฒนาอยู่ สถาปนิกซอฟต์แวร์จำเป็นที่จะต้องรู้ว่ามีสิ่งใดสามารถใช้ซ้ำได้บ้างสำหรับในองค์กรหรือในแอพพลิเคชันที่กำลังพัฒนา โดยต้องสังเกตและเข้าใจถึงภาพรวมสภาพแวดล้อมของระบบ สามารถออกแบบชิ้นส่วนซอฟต์แวร์สำหรับระบบ รวมทั้งมีความรู้ครอบคลุมไปถึงแอพพลิเคชันอื่นๆ ในองค์กรที่กำลังพัฒนาแอพพลิเคชันต้องติดต่อด้วย ทั้งนี้สถาปนิกซอฟต์แวร์ยังต้องสามารถแบ่งแอพพลิเคชันที่ซับซ้อนออกเป็นส่วนย่อยที่สามารถจัดการได้ง่าย มีความเข้าใจฟังก์ชันการทำงานของแต่ละคอมโพเนนท์ในตัวแอพพลิเคชันเป็นอย่างดี รวมถึงการติดต่อและการขึ้นต่อกันระหว่างคอมโพเนนท์เหล่านั้น ที่สำคัญที่สุดคือสามารถสื่อสารแนวคิดข้างต้นให้นักพัฒนาสามารถเข้าใจได้เพื่อให้การทำงานเป็นไปอย่างมีประสิทธิภาพ สถาปนิกซอฟต์แวร์มักใช้ UML หรือ OOP เป็นเครื่องมือช่วยสื่อสารสิ่งเหล่านี้ไปยังนักพัฒนาซอฟต์แวร์

ประเภทของสถาปนิกซอฟต์แวร์ อาจแบ่งได้ดังนี้

1) สถาปนิกวิสาหกิจ (Enterprise Architect) รับผิดชอบสถาปัตยกรรมระหว่างโครงการทั้งหมด
2) สถาปนิกโซลูชัน (Solution Architect) เน้นสถาปัตยกรรมโซลูชัน (Solution) ในระดับรายละเอียด
3) สถาปนิกแอพพลิเคชัน (Application Architect) รับผิดชอบการใช้ซ้ำคอมโพเนนท์และการดูแลโครงการเดี่ยว

1.4.3 โปรแกรมเมอร์ (Programmer)

โปรแกรมเมอร์เป็นผู้ที่ทำหน้าที่เขียนซอฟต์แวร์คอมพิวเตอร์ ในบางครั้งอาจเรียกว่านักพัฒนาซอฟต์แวร์ (Software Developer) หรือวิศวกรซอฟต์แวร์ (Software Engineer) โดยทั่วไปโปรแกรมเมอร์จะมีหน้าที่ทั้งการเขียนโปรแกรม การทดสอบและการแก้ไขบั๊ก (Bug) รวมถึงการคิดออกแบบและทดสอบตรรกะเพื่อแก้ปัญหาด้วยคอมพิวเตอร์

1.4.4 ผู้ดูแลระบบ (System Administrator)

ผู้ดูแลระบบเป็นผู้ที่มีหน้าที่รับผิดชอบในการดูแลและบำรุงรักษาระบบ ซึ่งอาจรวมทั้งระบบเครือข่ายในส่วนที่เกี่ยวข้องกับการพัฒนาโปรแกรม ผู้ดูแลระบบมักเขียนโปรแกรมในรูปแบบสคริปต์ เพื่อจัดการระบบในเบื้องต้น โดยใช้ภาษาโปรแกรม เช่น Perl หรือ Bash Script เป็นต้น ภาษาสคริปต์ (Scripting Language) ดังกล่าวจะเรียกใช้ซอฟต์แวร์อรรถประโยชน์เพื่อทำงานระดับล่างต่อไป สำหรับงานบำรุงรักษาระบบของผู้ดูแลระบบจะครอบคลุมถึงการปรับปรุงการติดตั้งโปรแกรมรุ่นใหม่ โดยเฉพาะอย่างยิ่งเมื่อซอฟต์แวร์บางตัวในระบบที่เป็นลักษณะแม่ข่ายเกิดข้อผิดพลาดที่อาจนำไปสู่ความไม่ปลอดภัยของ

องค์กร ผู้ดูแลระบบจำเป็นต้องมีความสามารถในการนำแพตช์ (Patch) มาแก้ไขโปรแกรมรุ่นที่มีบั๊กให้เป็นรุ่นที่ปลอดภัยมากขึ้น ในกรณีของการใช้โปรแกรมประเภทเปิดต้นรหัสผู้ดูแลระบบมักต้องมีความสามารถในการแก้ไขข้อผิดพลาดและคอมไพล์โปรแกรมรุ่นใหม่ได้เอง

1.4.5 ผู้ดูแลเครือข่าย (Network Administrator)

ผู้ดูแลเครือข่ายเป็นผู้ที่มีหน้าที่รับผิดชอบในการดูแลคอมพิวเตอร์ในส่วนที่เกี่ยวข้องกับเครือข่ายทั้งที่เป็นฮาร์ดแวร์และซอฟต์แวร์ รวมไปถึงการปรับแต่ง ดูแล และติดตั้งอุปกรณ์เครือข่ายใหม่ ผู้ดูแลเครือข่ายจะต้องมีความรู้ทางเทคนิคเชิงลึก มีความสามารถในการเรียนรู้อุปกรณ์เครือข่ายใหม่ๆ และซอฟต์แวร์ที่เกี่ยวข้องได้อย่างรวดเร็ว สำหรับในหลายองค์กรอาจจะรวมถึงงานทางด้านการออกแบบเครือข่ายด้วย

1.4.6 ผู้ดูแลฐานข้อมูล (Database Administrator)

ผู้ดูแลฐานข้อมูลเป็นผู้ที่มีหน้าที่ออกแบบ อิมพลีเมนต์ (Implement) ดูแลรักษาและซ่อมบำรุงฐานข้อมูลขององค์กร โดยมีบทบาทที่อาจรวมถึงการกำหนดกลยุทธเพื่อการพัฒนาและออกแบบฐานข้อมูล เฝ้าระวังและปรับปรุงสมรรถนะและความจุของฐานข้อมูล และประเมินความต้องการในการขยายการใช้งานของระบบฐานข้อมูล ผู้ที่ทำงานในตำแหน่งนี้จำเป็นต้องมีทักษะอื่นๆ ด้วย เช่น ความรู้ความเข้าใจเกี่ยวกับการย้ายโอนข้อมูล การทำซ้ำข้อมูล การสำรองและการกู้คืนข้อมูล เป็นต้น

1.4.7 วิศวกรสนับสนุนลูกค้า (Customer Support Engineer)

วิศวกรสนับสนุนลูกค้าเป็นวิศวกรที่ทำหน้าที่สนับสนุนภายในหน่วยงานหรือสนับสนุนสินค้าของบริษัทในทางเทคนิค โดยอาจรับแก้ไขปัญหาเกี่ยวกับระบบหรือซอฟต์แวร์ เพื่อให้แน่ใจว่าระบบหรือซอฟต์แวร์ดังกล่าวสามารถทำงานได้อย่างที่ควรจะเป็น ในระหว่างช่วงการพัฒนาหรือการออกแบบระบบใหม่นั้น การสนับสนุนการสร้างระบบมีความจำเป็นเพื่อทำให้การพัฒนาเป็นไปอย่างราบรื่น ในกรณีของซอฟต์แวร์วิศวกรสนับสนุนจะมีบทบาทในการปฏิบัติงานตลอดระยะเวลาการดำเนินการ เพื่อให้ระบบสามารถใช้งานได้อย่างมีประสิทธิภาพ

1.4.8 ผู้ดูแลเว็บ (Web Master)

ผู้ดูแลเว็บเป็นผู้ที่มีบทบาทในการพัฒนาและดูแลรักษาเว็บไซต์ทั้งในแง่มุมของระบบและเนื้อหา และอาจครอบคลุมไปถึงการสนับสนุนผู้ที่เข้ามาใช้งานเว็บไซต์ การจัดการกลั่นกรองการนำเสนอความเห็น และการปรับปรุงประสิทธิภาพการตอบสนองของเว็บไซต์ต่อผู้ใช้

1.4.9 วิศวกรความมั่นคง (Security Engineer)

วิศวกรความมั่นคงเป็นวิศวกรที่มีความรับผิดชอบในการตรวจสอบดูแลซอฟต์แวร์เพื่อป้องกันการเกิดข้อผิดพลาดที่เกินกว่าซอฟต์แวร์ได้เตรียมรองรับไว้ รวมถึงช่องโหว่ของระบบ โดยวิศวกรเครือข่ายจะต้องมีความเข้าใจในการบ่งชี้อาการพื้นฐานและปรับความปลอดภัยให้กับเครือข่ายและซอฟต์แวร์ รวมถึงสามารถนำแพตช์มาติดตั้งเพื่อทำให้ระบบมีความปลอดภัยได้

1.4.10 ผู้จัดการโครงการ (Project Manager)

ผู้จัดการโครงการเป็นผู้ที่มีความรับผิดชอบในการจัดการโครงการ ซึ่งมีความจำเป็นอย่างมากที่จะต้องมีประสบการณ์ทางการพัฒนาซอฟต์แวร์สูง ผู้จัดการโครงการจำเป็นต้องมีความเข้าใจในรูปแบบวิธีการพัฒนาซอฟต์แวร์ที่นำมาใช้เป็นอย่างดี รวมถึงเข้าใจและคุ้นเคยกับวงจรการพัฒนาซอฟต์แวร์ (Software Development Life Cycle) ผู้จัดการโครงการมักจะมีหน้าที่รับผิดชอบดังนี้

- พัฒนาแผนสำหรับโครงการ
- จัดการติดต่อประสานงานกับลูกค้า
- จัดการประสานงานกับทีม
- จัดการความเสี่ยงที่เกิดขึ้น
- จัดการตารางเวลาการพัฒนา
- จัดการงบประมาณ
- จัดการและแก้ไขความขัดแย้ง

1.4.11 ผู้จัดการการปรับแต่งซอฟต์แวร์ (Software Configuration Manager)

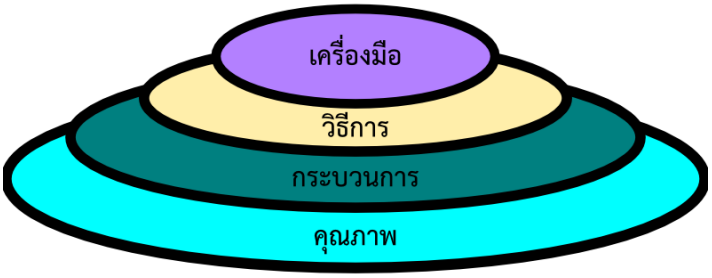
ผู้จัดการการปรับแต่งซอฟต์แวร์เป็นผู้ที่มีหน้าที่ดูแลและจัดการติดตามควบคุมซอฟต์แวร์โดยเฉพาะอย่างยิ่งการจัดการรุ่นของต้นรหัส และการวางเบสไลน์ (Baseline) ซึ่งเกี่ยวข้องกับการแยกสาขาของต้นรหัส และการจัดการการเปลี่ยนแปลงของต้นรหัสที่เกิดขึ้นเพื่อปรับลงสู่เบสไลน์ที่มี ทั้งนี้เพื่อปรับปรุงคุณภาพซอฟต์แวร์หรือลดข้อผิดพลาดในซอฟต์แวร์

1.4.12 ผู้จัดการคุณภาพซอฟต์แวร์ (Software Quality Manager)

ผู้จัดการคุณภาพซอฟต์แวร์เป็นผู้ที่มีหน้าที่รับผิดชอบในการจัดการคุณภาพของกระบวนการพัฒนาซอฟต์แวร์และตัวผลงานที่ได้ โดยผลงานที่มีคุณภาพจะต้องเป็นซอฟต์แวร์ที่ตรงตามความต้องการที่ผู้ใช้กำหนดไว้และผู้ใช้พึงพอใจ ในมุมมองของการปรับปรุงคุณภาพของกระบวนการพัฒนาซอฟต์แวร์ ผู้จัดการคุณภาพซอฟต์แวร์จะต้องมีบทบาทในการผลักดันให้บุคคลากรในองค์กรมีทัศนคติที่เป็นบวกต่อการปรับปรุงคุณภาพ โดยให้เป็นหน้าที่ของทุกคนในองค์กรหรือในทีมพัฒนาซอฟต์แวร์ ผู้จัดการคุณภาพซอฟต์แวร์อาจรับหน้าที่ตรวจสอบคุณภาพด้วยการทดลองระบบงานก่อนนำไปให้ลูกค้าใช้งานจริง

1.5 ลำดับชั้นของวิศวกรรมซอฟต์แวร์ (Software Engineering Layers)

วิศวกรรมซอฟต์แวร์เป็นเทคโนโลยีซึ่งสามารถแยกออกเป็นลำดับชั้นได้ แนวทางของวิศวกรรมซอฟต์แวร์อยู่บนพื้นฐานของความต้องการที่จะนำไปสู่คุณภาพซึ่งมาจากความพยายามในการปรับปรุงคุณภาพการพัฒนาอย่างต่อเนื่อง และนำไปสู่การพัฒนาที่มีประสิทธิภาพสูงขึ้น (Pressman, 2010)



รูปที่ 1 *ลำดับชั้นของวิศวกรรมซอฟต์แวร์ (Pressman, 2010)*

พื้นฐานสำคัญที่สนับสนุนวิศวกรรมซอฟต์แวร์ คือ การเน้น “คุณภาพ” (Quality) ในชั้น “กระบวนการ” (Process) จะเป็นตัวประสานชั้นที่เป็นเทคโนโลยีต่างๆ เข้าด้วยกัน เพื่อให้การพัฒนาซอฟต์แวร์เป็นไปได้อย่างราบรื่นและอยู่ในขอบเขตของเวลานำส่ง กระบวนการจะเป็นทั้งตัวกำหนดเฟรม-เวิร์คที่ต้องเตรียมขึ้นเพื่อให้สามารถนำเทคโนโลยีทางวิศวกรรมซอฟต์แวร์ไปใช้ในการพัฒนาซอฟต์แวร์ได้อย่างมีประสิทธิภาพ โดยกระบวนการพัฒนาซอฟต์แวร์จะต้องเตรียมพื้นฐานของการควบคุมโครงการนั้นๆ และเตรียมข้อมูลแวดล้อมเพื่อที่จะนำ “วิธีการ” (Methods) พัฒนามาประยุกต์ อีกทั้งระบุวิธีการสร้างผลิตภัณฑ์ บอกถึงการเตรียมไมล์สโตน (Milestone) เพื่อให้ควบคุมคุณภาพได้เป็นระยะๆ รวมทั้งสามารถจัดการการเปลี่ยนแปลงได้อย่างเหมาะสม วิธีการด้านวิศวกรรมซอฟต์แวร์จึงเป็นกลไกในเชิงเทคนิคที่แสดงว่าจะต้องสร้างซอฟต์แวร์อย่างไร วิธีการดังกล่าวรวมไปถึงงานต่างๆ ตั้งแต่การติดต่อ การวิเคราะห์ความต้องการ การออกแบบ การสร้างโปรแกรม การทดสอบ ตลอดจนการสนับสนุนและการดูแลรักษา วิธีการทางวิศวกรรมซอฟต์แวร์จะอยู่บนพื้นฐานของแต่ละกลุ่มเทคโนโลยีและรวมถึงเทคนิคการโมเดลและการอธิบายต่างๆ

สำหรับในชั้น “เครื่องมือ” (Tools) นั้น วิศวกรรมซอฟต์แวร์จะให้การสนับสนุนทั้งในลักษณะอัตโนมัติหรือกึ่งอัตโนมัติแก่กระบวนและวิธีการเพื่อให้ทำงานร่วมกันได้อย่างมีประสิทธิภาพสูงสุด ซึ่งกลุ่มเครื่องมือดังกล่าวเรียกว่า เครื่องมือช่วยเหลือทางวิศวกรรมซอฟต์แวร์ อย่างไรก็ตามองค์ประกอบสำคัญที่สุดของการพัฒนาซอฟต์แวร์คือบุคลากร (People)

1.6 บทสรุป

วิศกรรมซอฟต์แวร์เป็นศาสตร์ที่ประยุกต์วิธีการทางวิศวกรรมมาปรับใช้กับการพัฒนาซอฟต์แวร์ ในบทนี้ได้กล่าวถึงนิยามและประเภทของซอฟต์แวร์ จากนั้นได้กล่าวถึงความหมายและสาขาย่อยของวิศวกรรมซอฟต์แวร์รวมทั้งสายงานที่เกี่ยวข้อง ในส่วนท้ายของบทได้กล่าวถึงลำดับชั้นของวิศวกรรมซอฟต์แวร์ซึ่งเป็นแนวคิดพื้นฐานในการเพิ่มคุณภาพซอฟต์แวร์โดยใช้เครื่องมือ วิธีการ และกระบวนการร่วมกัน ในบทที่ 2 จะกล่าวถึงวงจรการพัฒนาซอฟต์แวร์ประเภทต่างๆ ซึ่งเป็นกระบวนการหลักในการพัฒนาซอฟต์แวร์อย่างมีระบบเพื่อให้ได้ซอฟต์แวร์คุณภาพสูงต่อไป

1.7 คำถามท้ายบทที่ 1

1) ซอฟต์แวร์คืออะไร

2) ซอฟต์แวร์ต่างกับฮาร์ดแวร์อย่างไร

3) วิศวกรรมซอฟต์แวร์คืออะไร

4) ประเภทของซอฟต์แวร์ระบบมีอะไรบ้าง

5) ระบบจัดการหน้าต่างในระบบปฏิบัติการแต่ละแบบ มีลักษณะแตกต่างกันอย่างไร

6) ซอฟต์แวร์ประยุกต์คืออะไร ซอฟต์แวร์ประยุกต์ประเภทซอฟต์แวร์ธุรกิจในปัจจุบันมีอะไรบ้าง

7) ซอฟต์แวร์ชุดสำนักงานอัตโนมัติในปัจจุบันมีอะไรบ้าง แนวโน้มการใช้งานมีลักษณะอย่างไร

8) สาขาย่อยในวิศวกรรมซอฟต์แวร์มีอะไรบ้าง

9) อธิบายลำดับชั้นของวิศวกรรมซอฟต์แวร์และความเกี่ยวข้องกันของแต่ละลำดับชั้น