บทที่ 2
สรุปสาระสำคัญจากเอกสารที่เกี่ยวข้อง

จากการศึกษาเอกสารและงานวิจัยที่เกี่ยวข้องกับการพัฒนาระบบพัฒนาคุณภาพของวิศวกรซอฟต์แวร์โดยกระบวนการซอฟต์แวร์ระดับบุคคลบนโปรแกรมประยุกต์เว็บ มีส่วนสำคัญที่ควรจะพิจารณาดังต่อไปนี้

2.1 การพัฒนากระบวนการพัฒนาซอฟต์แวร์ (Software Process Improvement)

2.2 กระบวนการซอฟต์แวร์ระดับบุคคล (Personal Software Process : PSP)

2.3 มาตรฐานอุตสาหกรรมซอฟต์แวร์ ISO 29110 VSE

2.4 เทคโนโลยีโปรแกรมประยุกต์เว็บ (Web Application)

2.5 แนวคิด Software as a Service (SaaS)

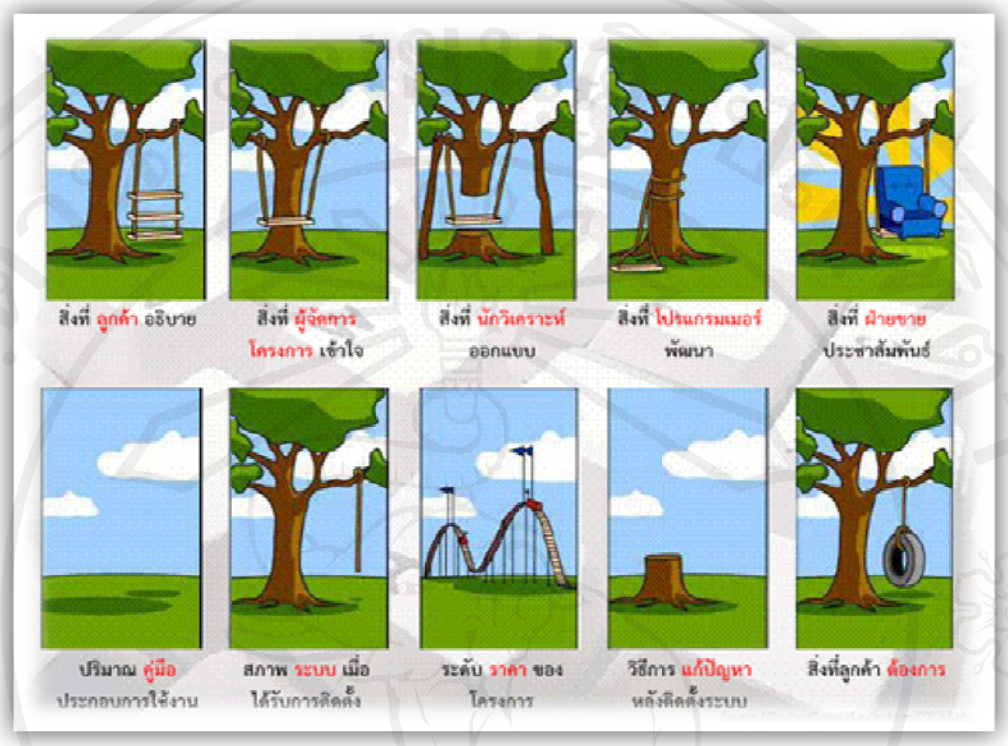
2.6 กระบวนการพัฒนาซอฟต์แวร์แบบจำลองน้ำตก (Waterfall Model)

2.1 การพัฒนากระบวนการพัฒนาซอฟต์แวร์ (Software Process Improvement)

การพัฒนากระบวนการพัฒนาซอฟต์แวร์ (Software Process Improvement) คือ กระบวนการต่าง ๆ ที่ทำให้ความต้องการของลูกค้า (Requirement) กลายเป็นผลิตภัณฑ์ (Product)

แต่ก็มีอุปสรรคหลายด้าน เช่น ด้านการสื่อสารที่ไม่เข้าใจกันของทุกฝ่าย ทั้งตัวลูกค้าเอง ผู้จัดการโครงการ นักวิเคราะห์ โปรแกรมเมอร์ รวมไปถึงฝ่ายขาย ทำให้งานออกมาผิดไปไม่ตรงกับความต้องการและทำให้งานไม่มีคุณภาพ ไม่เป็นที่ยอมรับและเกิดความเสียหาย ตามตัวอย่างดังรูป

ตัวอย่างปัญหาในการพัฒนาซอฟต์แวร์ของข้อมูลที่ผู้พัฒนาทราบและไม่ทราบ ซึ่งข้อมูลที่ผู้พัฒนาไม่ทราบนั้นอาจจะทำให้เกิดผลกระทบหลาย ๆ ด้าน ตัวอย่างเช่น ระยะเวลาในการพัฒนาบางช่วงอาจจะใช้เวลานานเกินไปทำให้งานเสร็จช้ากว่ากำหนด หรือ Bug ที่เกิดทั้งหลังและขณะกำลังพัฒนาทำให้ซึ่งจะทำให้โปรแกรมไม่มีคุณภาพ



รูปที่ 2.1 แสดงความคิดในแต่ละมุมมองที่แตกต่างของบุคคลต่าง ๆ

รูปที่ 2.2 ขั้นตอนกระบวนการพัฒนาซอฟต์แวร์

"โปรแกรมเมอร์" เป็นสิ่งมีชีวิตที่ใช้ชีวิตอยู่ระหว่างวิทยาศาสตร์และศิลปะ การควบคุมให้สิ่งมีชีวิตชนิดนี้ปฏิบัติตาม กฎ ระเบียบ หรือข้อบังคับ อาจส่งผลให้ได้งานที่คุณภาพด้อยลง หรือสร้างความไม่พอใจสะสม

อย่างไรก็ตาม โปรแกรมเมอร์แต่ละคน มักมีแบบแผนกระบวนการทำงานประจำตัว และเมื่อโปรแกรมเมอร์มีแบบแผนการทำงานประจำตัว การพัฒนากระบวนการจึงสามารถทำได้ในระดับ บุคคล

ในระดับสากล มีการใช้ PSP เป็นเครื่องมือในการพัฒนากระบวนการในระดับดังกล่าว

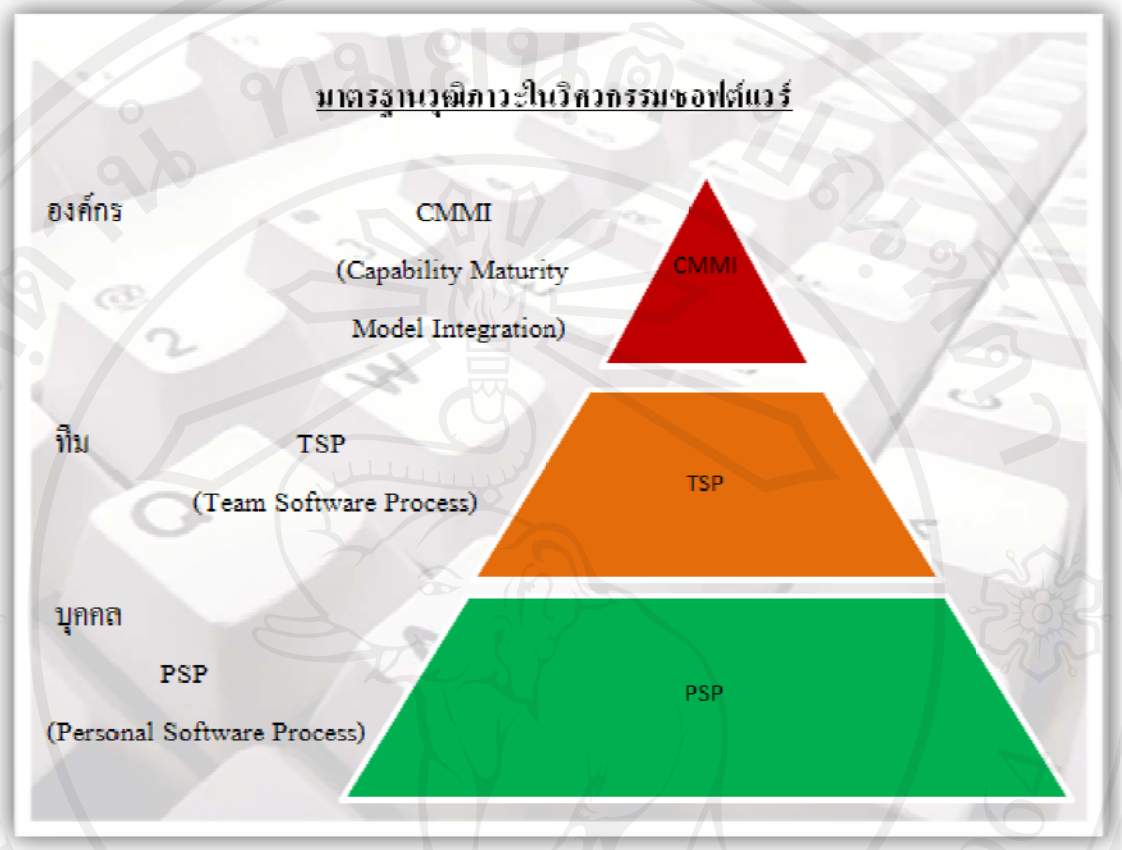
2.2 กระบวนการซอฟต์แวร์ระดับบุคคล (Personal Software Process : PSP)

เพื่อการพัฒนา Software Engineering ตามหัวข้องานวิจัยของนักศึกษาได้ทำการศึกษาทฤษฎีและหลักการต่าง ๆ ที่สามารถนำมาประยุกต์ใช้ โดยแบ่งออกเป็นหัวข้อต่าง ๆ ดังต่อไปนี้

- PSP เป็นแนวคิดในการจัดเก็บและสรุปข้อมูลของกระบวนการพัฒนาซอฟต์แวร์เพื่อนำมาปรับปรุงเพิ่มประสิทธิภาพในภาพรวม
- สร้างและควบคุมมาตรฐานโดย Software Engineering Institute (SEI), Carnegie Melon University (CMU)
- เทียบเท่ากับ Capability Maturity Model Integration (CMMI) Level 5 ในระดับบุคคล

โดย PSP จะเน้นกระบวนการวิศวกรรมซอฟต์แวร์ การประยุกต์ใช้ PSP กับวิศวกรรมซอฟต์แวร์ การปรับปรุงกระบวนการพัฒนาซอฟต์แวร์ส่วนบุคคล และ มาตรฐานเชิงคุณภาพ

PSP เป็นพื้นฐานของมาตรฐาน TSP หรือ CMMI ที่บริษัทพัฒนาซอฟต์แวร์ชั้นนำหลายแห่งมีกันซึ่งทำให้บริษัทชั้นนำทั้งหลายเหล่านั้นมีความน่าเชื่อถือในคุณภาพซอฟต์แวร์รวมไปถึงวิธีการและกระบวนการในการผลิตที่เป็นแบบแผนทำให้ง่ายต่อการควบคุม ซึ่งบุคลากรในบริษัทนั้น ๆ ควรจะต้องมีพื้นฐานของ PSP มาก่อนจึงจะทำให้สามารถปฏิบัติตามมาตรฐานการพัฒนาซอฟต์แวร์ในระดับสูงขึ้นไปได้เป็นอย่างดี

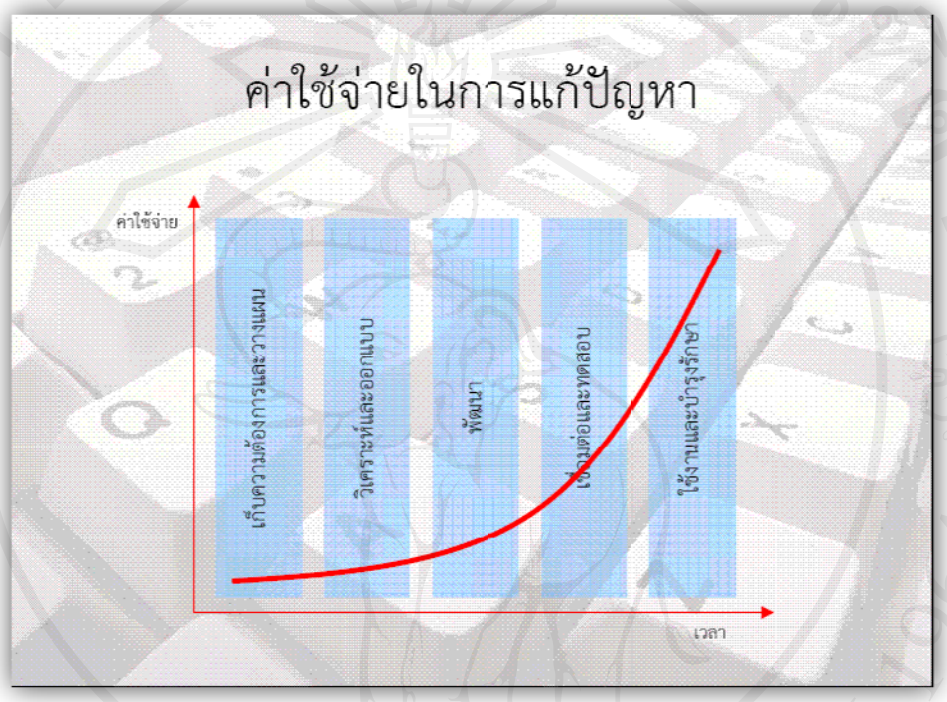


รูปที่ 2.3 มาตรฐานวุฒิภาวะในวิศวกรรมซอฟต์แวร์

สาเหตุที่ต้องใช้ทฤษฎีของ PSP ประกอบการพัฒนา

- กระบวนการพัฒนาของโปรแกรมเมอร์แต่ละบุคคลแตกต่างกัน
  - เมื่อกระบวนการพัฒนาโปรแกรมของแต่ละบุคคลไม่สอดคล้องกัน จะสามารถทำให้เกิดปัญหาขึ้นภายในทีมได้
- โปรแกรมเมอร์ต้องบันทึกข้อมูลของตนเพื่อใช้ในการปรับปรุงกระบวนการในภายหลัง
  - โปรแกรมเมอร์หรือนักพัฒนาซอฟต์แวร์ส่วนใหญ่มักไม่คำนึงถึงข้อผิดพลาดที่เกิดขึ้น เมื่อเกิดปัญหาขึ้นมาแล้ว ก็แก้ไขและปล่อยผ่านไปโดยไม่คำนึงถึงการเก็บข้อมูลไว้เป็นสถิติเพื่อวิเคราะห์และพัฒนาตนเองในภายหลัง
- การป้องกันเป็นการจัดการปัญหาได้ดีที่สุด
  - การพัฒนาตนเองโดยใช้ทฤษฎีของ PSP ทำให้สามารถรู้ได้ถึงข้อบกพร่องของตนเองและทำให้ป้องกันและจัดการปัญหากับก่อนที่จะเกิดขึ้นได้

- กระบวนการที่มีคุณภาพที่ดีที่สุดคือกระบวนการที่ “ไม่สร้างปัญหาเลย (Zero Defect)”
  - เพื่อให้วิศวกรซอฟต์แวร์สามารถเข้าถึงจุดสูงสุดในการพัฒนาซอฟต์แวร์นั่นคือการไม่พบปัญหาใดเลยในขณะพัฒนาซอฟต์แวร์ หรือ (Zero Defect)



รูปที่ 2.4 แสดงค่าช้าจ่ายในการแก้ปัญหา

ค่าใช้จ่ายในการแก้ปัญหาจะสูงหรือต่ำขึ้นอยู่กับระยะเวลา ณ ที่ทำการแก้ปัญหาโดยจะแบ่งและเรียงตามลำดับค่าใช้จ่ายจากต่ำไปสูงดังนี้

1. เก็บความต้องการและวางแผน
2. วิเคราะห์และออกแบบ
3. พัฒนา
4. เชื่อมต่อและทดสอบ
5. ใช้งานและบำรุงรักษา

ดังนั้นจึงควรที่ศึกษาทฤษฎี PSP เพื่อลดข้อผิดพลาดตั้งแต่การวางแผนเพื่อลดปัญหาและข้อผิดพลาดซึ่งจะไม่ทำให้เกิดค่าใช้จ่ายที่สูงจนเกินไป

ตัวชี้วัดหลักใน PSP

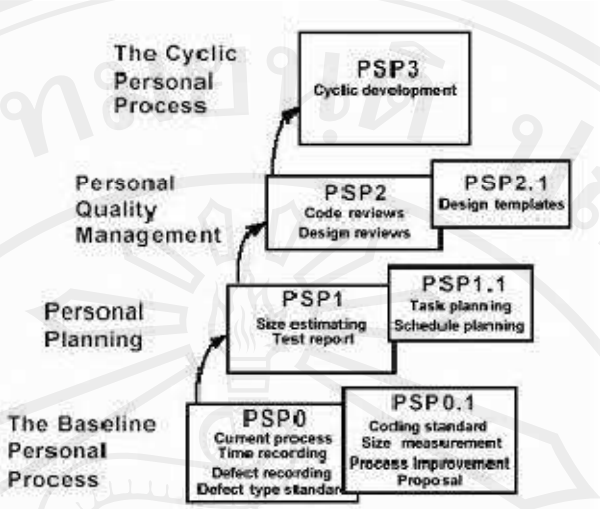
- ขนาดของโปรแกรม
  - บรรทัด (Lines of Code)
- เวลาทั้งหมดที่ใช้ในการพัฒนา
  - นาที (Minutes)
- คุณภาพ
  - ปัญหา/ข้อผิดพลาด (Defects)

ระดับของกระบวนการ PSP

วิศวกรซอฟต์แวร์จะต้องทำตามวิธีการที่กำหนดไว้ (Prescribed Methods) โดยขั้นตอนจะถูกนำเสนอตั้งแต่ระดับ PSP0 จนถึง PSP3 ในบางระดับอาจมีระดับย่อยเช่น PSP2.1 และมีแบบฝึกหัดการเขียนโปรแกรมและต้องบันทึกเป็นรายงาน ซึ่งสำหรับในแต่ละแบบฝึกหัดวิศวกรซอฟต์แวร์จะได้ทราบถึงวิธีการทางวิศวกรรมซอฟต์แวร์ที่แตกต่างกัน และถูกปรับยกระดับคุณภาพให้ค่อย ๆ สูงขึ้น ผลของวิธีการที่แนะนำจะแสดงให้เห็นจากงานที่ทำ และวัดประสิทธิภาพการทำงานด้วยตัวของพวกเขาเอง ขั้นตอนของกระบวนการ PSP0 ถึง PSP3 เป็นดังรูปที่ 2.5

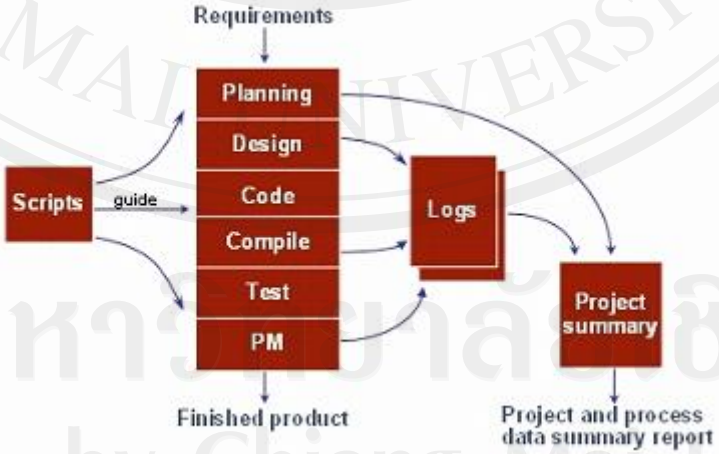
PSP0: The Baseline Personal Process

เป็นกระบวนการส่วนบุคคลสำหรับใช้อ้างอิงเปรียบเทียบกับกระบวนการที่ได้รับการปรับปรุงฝึกปฏิบัติแล้วในภายหลัง ในขั้นตอน PSP0 และ PSP0.1 นี้ได้จัดเตรียมการแนะนำ PSP ฐานเริ่มต้นโดยฝึกวัดข้อมูล ได้แก่ ขนาดของโปรแกรม เวลา และข้อบกพร่องซึ่งเป็นข้อมูลที่ใช้พัฒนาซอฟต์แวร์ของวิศวกรในแบบฉบับของตัวเอง ผ่านแบบฝึกหัดสามบท และมีการบังคับวัฏจักรในการพัฒนาซอฟต์แวร์ประกอบด้วย 6 ขั้นตอนตามที่กำหนดไว้ดังรูปที่ 2.6 กล่าวคือ วางแผน ออกแบบ เขียนโค้ด คอมไพล์ ทดสอบ และชันสูตร (PM: Post Mortem)



รูปที่ 2.5 ระดับของกระบวนการ PSP

PSP0 เป็นการแนะนำการวัดในกระบวนการขั้นพื้นฐาน และการวางแผน เวลาที่ใช้ในการพัฒนา ข้อบกพร่องที่ปรากฏ และขนาดของโปรแกรมซึ่งถูกวัดและบันทึกบนฟอร์มของเอกสารที่กำหนดไว้ให้ แผนงานที่วิศวกรวางไว้ก่อนการพัฒนา และผลสรุปสุดท้ายจะต้องใส่ไว้ในเอกสาร Project Summary และในขั้นตอนของ PSP0.1 ได้แนะนำฟอร์มเอกสารที่ให้บันทึกชื่อ Process Improvement Proposals (PIPS) ไว้สำหรับบันทึกแนวความคิดที่ต้องการใช้เพื่อพัฒนากระบวนการด้วย



รูปที่ 2.6 PSP Process Flow

PSP1: Personal Project Management

PSP1 และ PSP1.1 เน้นเทคนิคการจัดการโครงการส่วนบุคคล ได้แนะนำการประมาณขนาดของโปรแกรมและแรงงาน (Effort) ที่คาดว่าจะใช้ล่วงหน้า การวางแผนตารางเวลา และวิธีการติดตามผลในตารางเวลาที่วางไว้ (Schedule Tracking) การประมาณขนาดของโปรแกรมและแรงงานโดยใช้วิธี PROBE (Proxy-Based Estimating) เป็นการอาศัยหลักความสัมพันธ์จากการประมาณค่าตัวแทน (Proxy) เช่น จำนวนของวัตถุ (Object) จำนวนฟังก์ชัน จำนวนกระบวนคำสั่ง (Procedures) เพื่อใช้ในการประมาณขนาดของโปรแกรมเบื้องต้น จากนั้นใช้ค่าข้อมูลที่เคยเก็บไว้ (Historical Data) ในการแปลงค่าจากขนาดของ Proxy ไปเป็นขนาดของ LOC (Line of Code)

PSP2: Personal Quality Management

ในขั้นตอนของ PSP2 ได้เพิ่มการออกแบบส่วนบุคคล (Personal Design) และการทบทวนรหัสคำสั่ง (Code Review) การทบทวนนี้ช่วยให้วิศวกรซอฟต์แวร์พบข้อบกพร่องได้เร็วขึ้นในกระบวนการและเห็นคุณค่าของการค้นพบข้อบกพร่องแต่เนิ่น ๆ พวกเขาจะต้องวิเคราะห์ข้อบกพร่องที่พบและใช้เป็นข้อมูลในการสร้างรายการตรวจสอบที่ใช้เพื่อการทบทวน (Review Checklists) ข้อดีของรายการการตรวจสอบที่สร้างจากประสบการณ์ที่ผ่านมาของแต่ละคน ทำให้ค้นพบข้อบกพร่องของแต่ละคนที่มักจะเกิดขึ้นซ้ำ ๆ กัน

การทบทวนมีประสิทธิผลสำหรับการกำจัดข้อบกพร่องที่มักพบในขั้นตอนการคอมไพล์โปรแกรม และข้อบกพร่องที่ส่วนมากพบที่ขั้นตอนการทดสอบ แต่การลดข้อบกพร่องในขั้นตอนการทดสอบที่สำคัญเกิดจากการออกแบบที่มีคุณภาพ PSP2.1 จึงชี้ให้เห็นถึงความจำเป็นของกรออกแบบ โดยเพิ่มส่วนของ Design Notation ให้มีแม่แบบ (Template) ของการออกแบบสี่แบบ และวิธีการทบทวนการออกแบบ (Design Review Methods) โดยความตั้งใจนี้ไม่ใช่เพื่อแนะนำวิธีการออกแบบแบบใหม่แต่เพื่อให้แน่ใจว่านักออกแบบได้ไตร่ตรอง และทำเอกสารการออกแบบในมุมมองที่แตกต่างกันสี่ด้าน ได้แก่ ข้อกำหนดทางการปฏิบัติการ (Operational Specification) ข้อกำหนดทางฟังก์ชัน (Functional Specification) ข้อกำหนดทางสถานะ (State Specification) และข้อกำหนดทางตรรกะ (Logic Specification)

PSP3: Cyclic Personal Process

PSP3 ชี้ให้เห็นถึงขนาดของโครงการที่เหมาะสมสำหรับผู้พัฒนาคนหนึ่งที่สามารถทำได้โดยไม่สูญเสียคุณภาพของงาน หรือแม้แต่ผลิตภาพ (Productivity) วิศวกรซอฟต์แวร์จะเรียนรู้ถึงผลิตภาพที่ตัวเองทำได้ในช่วงของขนาดโครงการที่เล็กที่สุดจนถึงขนาดของโครงการที่ใหญ่ที่สุดจุด

ที่ผลิตภาพเริ่มลดลงก็คือจุดที่บอกถึงขนาดโครงการที่สูงสุดที่ผู้พัฒนาจะรับได้ PSP3 ได้แนะนำกลยุทธ์ในการพัฒนาเป็นวงรอบ (A Cyclic Development Strategy) ซึ่งโปรแกรมขนาดใหญ่จะถูกแบ่งเป็นส่วน ๆ ในแต่ละวงรอบในการพัฒนาตามความสามารถของนักพัฒนาเอง แลทำส่วนที่เหลือในวงรอบถัดไป และเพื่อเป็นการสนับสนุนแนวคิดนี้ PSP3 ยังได้แนะนำ High-Level Design, High-Level Design Review, Cycle Planning และ Development Cycles บนพื้นฐานของกระบวนการ PSP2.1 ฟอร์มเอกสารใหม่สองฟอร์มได้ถูกแนะนำได้แก่ เอกสารสรุปในแต่ละวงรอบ ซึ่งบันทึกผลสรุปของขนาด เวลาที่ใช้ และข้อบกพร่องที่ค้นพบในแต่ละวงรอบ และเอกสารที่ติดตามประเด็นปัญหาสำคัญ (Issue Tracking Log) ที่อาจส่งผลต่อวงรอบในอนาคต หรือเมื่อพัฒนาทุกวงรอบอย่างสมบูรณ์แล้ว

ในตารางที่ 2.1 เป็นการสรุปสิ่งที่ผู้เรียน (วิศวกรซอฟต์แวร์ หรือนักเรียนที่ทำการฝึก PSP) จะเรียนรู้และปรับปรุงกระบวนการของตัวเองผ่านแบบฝึกหัดเหล่านี้ ภายหลังจากที่ได้รับมอบหมายให้ทำแต่ละ Assignment ของแต่ละขั้นตอนของ PSP

ตารางที่ 2.1 PSP Training Modules

Process level	**Assignments**	**สิ่งที่ผู้เรียนจะเรียนรู้**
PSP0	Assignment#1	เรียนรู้การบันทึก เวลา และข้อบกพร่อง (Time and Defect Recording) ของแต่ละกระบวนการ
PSP0.1	Assignment#2 Assignment#3	เรียนรู้การบันทึกขนาดของโปรแกรม (Size Recording)
PSP1	Assignment#4	การประมาณขนาดของโปรแกรม และเวลาที่ใช้ (Size and Time Estimating)
PSP1.1	Assignment#5 Assignment#6	การวางแผนงาน และการจัดกำหนดการ (Task and Schedule Planning)
	Midterm Report	พัฒนากระบวนการสำหรับการวิเคราะห์ข้อมูลในกระบวนการ เพื่อใช้วิเคราะห์ข้อมูลในกระบวนการที่ได้จาก Assignments ทั้งหกข้างต้น
PSP2	Assignment#7	เรียนรู้การทบทวนการทำงานเป็นรายบุคคล (Personal Reviews)

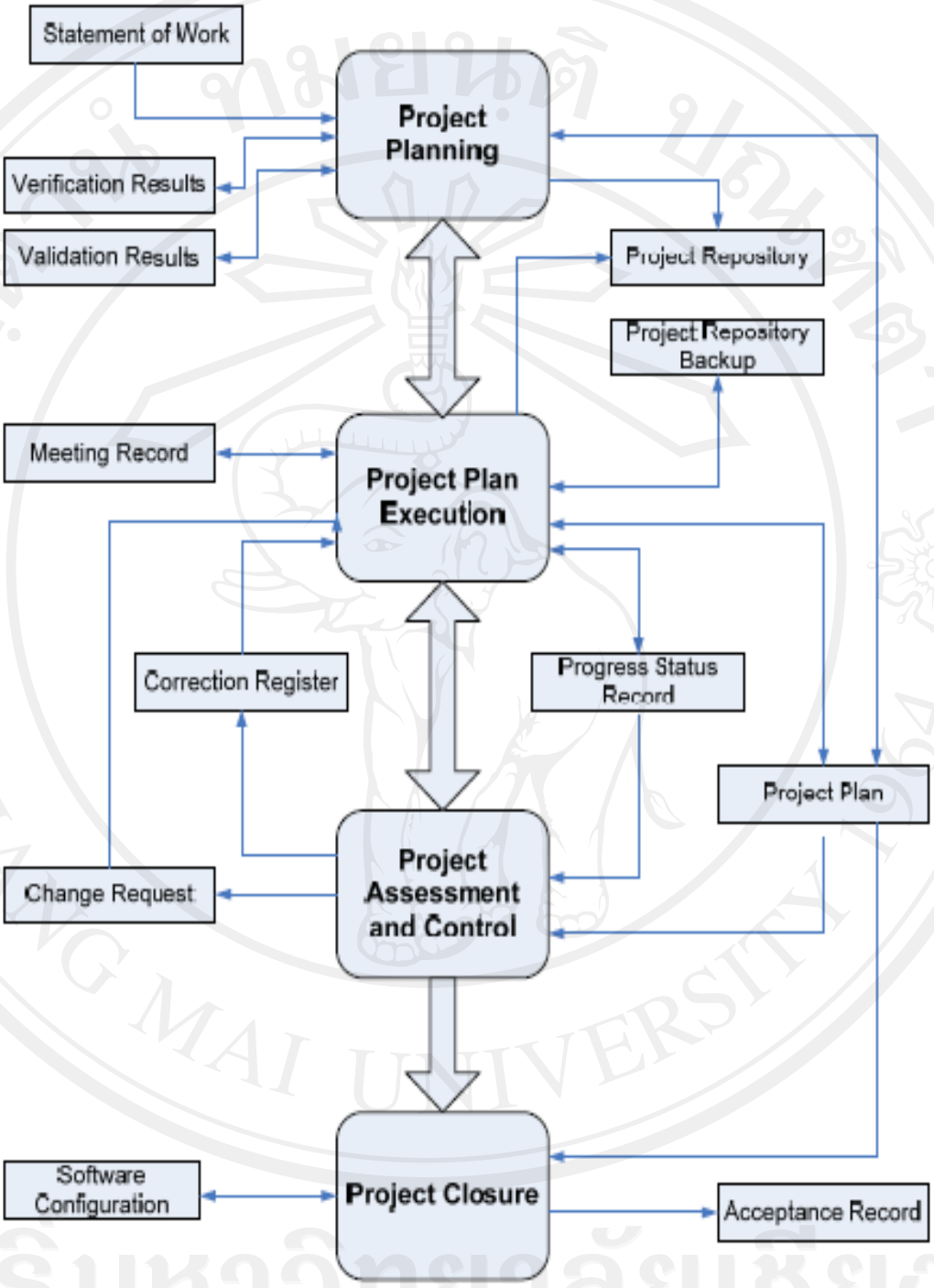
PSP2.1	Assignment#8 Assignment#9	เรียนรู้กระบวนการออกแบบอย่างเป็นทางการ (Formal Design Process)
PSP3	Assignment#10	เรียนรู้การพัฒนาเป็นวงรอบ (Cyclic Development)
	Final Report	พัฒนากระบวนการสำหรับการวิเคราะห์ข้อมูลในกระบวนการและใช้เพื่อวิเคราะห์ข้อมูลที่ได้มาทั้งหมด

2.3 มาตรฐานอุตสาหกรรมซอฟต์แวร์ ISO 29110 VSE

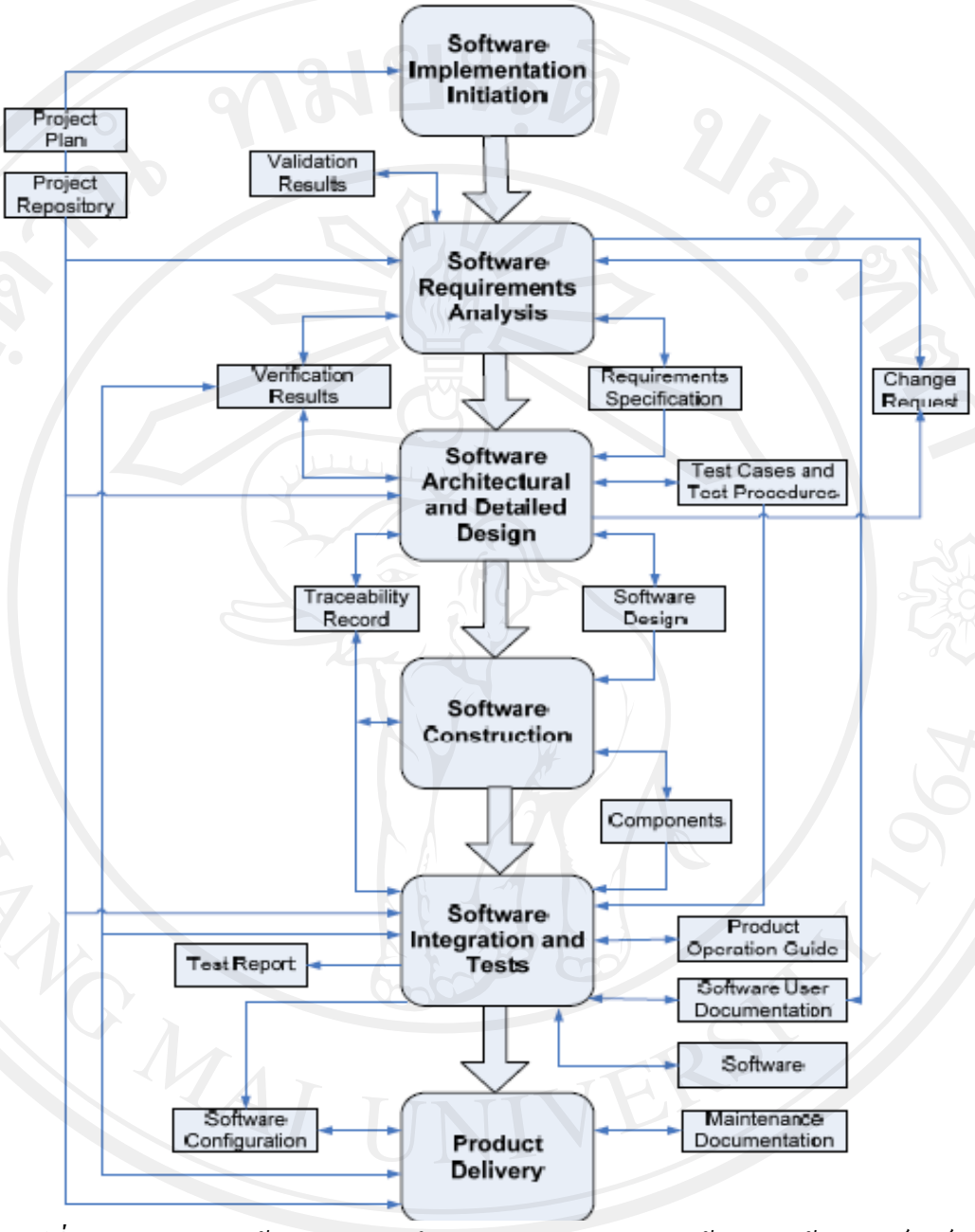
ISO 29110 เป็นแนวคิดยุคใหม่ของ ISO ที่จะเน้นการเติบโตของอุตสาหกรรมขนาดกลางและเล็ก รวมทั้งผู้ประกอบใหม่ที่เข้ามาให้มีโอกาสในการแข่งขันตามแนวทางการพัฒนา Outsourcing ซึ่งในอดีตที่ผ่านมา มาตรฐานวิศวกรรมซอฟต์แวร์ได้ถูกทำให้เป็นเรื่องที่เข้าใจยากและมีความสลับซับซ้อนยุ่งยากในการปฏิบัติตาม ประกอบกับมาตรฐานซอฟต์แวร์ระดับสากลที่มีอยู่ในปัจจุบันจะเหมาะสมกับการปฏิบัติงานขององค์กรขนาดใหญ่ ISO 29110 จึ่งถูกพัฒนาด้วยแนวคิดพื้นฐานเพื่อสนับสนุนองค์กรขนาดเล็กให้มีโอกาสในการปรับปรุงกระบวนการและรับรองคุณภาพในระดับสากล

กระบวนการของ ISO 29110 จะเน้นให้ผู้ประกอบการซึ่งอาจจะเป็นผู้ประกอบการอิสระ ผู้ประกอบการขนาดเล็กที่มีบุคคลกรไม่เกิน 25 คน หรือหน่วยงานทางด้านSoftwareที่อยู่ในองค์กรขนาดใหญ่ให้มีกระบวนการในการพัฒนาซอฟต์แวร์ที่เป็นระบบ และเข้าสู่กระบวนการสากล โดยจะเป็นการเริ่มต้นในเชิงกิจกรรมของการปรับปรุงกระบวนการ หรือ SPI (Software Process Improvement)

ทาง ISO 29110 ได้ให้ความสำคัญในกระบวนการที่จะต้องทำการปรับปรุงให้เป็นระบบและเป็นสากลสองกระบวนการหลัก คือ กระบวนการด้านการบริหารโครงการ (Project Management) และกระบวนการด้านการสร้างซอฟต์แวร์ (Software Implementation) ซึ่งจะประกอบด้วยกระบวนการย่อยๆภายในอีก ทั้งสองกระบวนได้ถูกออกแบบให้เหมาะสมกับผู้ประกอบการขนาดเล็กจึงมีความเหมาะสมในการประยุกต์ใช้ได้ทันที โดยได้กำหนดขนาดของกระบวนการให้เหมาะสมกับองค์กรขนาดเล็ก จึงไม่สร้างปัญหาในการปรับใช้งานให้เข้ากับองค์กร



รูปที่ 2.7 กระบวนการด้านการบริหารโครงการและกระบวนการด้านการสร้างซอฟต์แวร์



รูปที่ 2.8 กระบวนการด้านการบริหารโครงการและกระบวนการด้านการสร้างซอฟต์แวร์

2.4 เทคโนโลยีโปรแกรมประยุกต์เว็บ (Web Application)

ส่วนมากคนมักจะคุ้นเคยกับ Desktop Application หรือโปรแกรมคอมพิวเตอร์ที่ติดตั้งบนคอมพิวเตอร์ส่วนบุคคล เช่น โปรแกรมพวก Microsoft Office เช่นโปรแกรมพิมพ์งาน หรือ Word Processor ที่ใช้พิมพ์งาน ซึ่งจะติดตั้งบนเครื่องคอมพิวเตอร์และใช้ได้ทีละคน

หากทำงานที่บริษัทจะคุ้นเคยกับโปรแกรมที่บริษัทใช้ เช่น ERP หรือ MRP หรือโปรแกรมห้องสมุด โปรแกรมพวกนี้มักจะเป็นโปรแกรมแบบ Client - Server คือโปรแกรมที่ใช้งานโดยคน

หลาย ๆ คนพร้อม ๆ กัน โดยมีการเก็บข้อมูลไว้ที่ฐานข้อมูลกลาง ทำให้ทุกคนใช้ข้อมูลเดียวกันร่วมกันได้

โดยโปรแกรมจะถูกแบ่งออกเป็นสองส่วน คือส่วนหนึ่งถูกติดตั้งที่ Server ส่วนกลาง และอีกส่วนติดตั้งที่คอมพิวเตอร์ของผู้ใช้ หรือที่เรียกว่า Client ซึ่งทั้งสองส่วนจะทำงานร่วมกัน โดยโปรแกรมบน Server มักจะทำงานหลัก ๆ ที่จำเป็นเช่นการคำนวณ การค้นหาข้อมูล การเก็บข้อมูล ส่วนโปรแกรมที่คอมพิวเตอร์ของเรา หรือที่เรียกว่า Client นั้นจะทำหน้าที่นำเสนอข้อมูล และรับข้อมูลจากผู้ใช้ หรือที่เรียกว่าเป็น User Interface โปรแกรมแบบนี้ซับซ้อนและดูแลยาก เพราะหาก Upgrade โปรแกรมที่ Server ก็ต้อง Upgrade โปรแกรมที่ Client ด้วย ซึ่งเป็นเรื่องที่ลำบากเนื่องจาก Client มีหลายเครื่องทำให้ยากต่อการ Upgrade ได้ครบ

ในระยะหลังนี้จึงได้เกิดโปรแกรมอีกประเภทหนึ่งซึ่งได้รับความ นิยมมากขึ้น โปรแกรมนั้นก็คือ Web Application เป็นโปรแกรมที่ติดตั้งบน Server ซึ่ง Web Application สามารถใช้งานแทนโปรแกรมทั้งแบบ Desktop และแบบ Client - Server เช่น โปรแกรม Google Application ซึ่งใช้แทน Microsoft Office เช่นมีทั้ง Word Processor และหรือ Spread Sheet ที่ใช้แทน Excel

โดยเฉพาะโปรแกรมแบบ Client-Server หลายตัวก็กำลังแปลงตัวเป็น Web Application เพื่อตอบสนองความต้องการของลูกค้า เช่น SAP, Lotus Notes ฯลฯ

ข้อดี ของ Web Application ตรงที่ Web Application ไม่ต้องใช้ Client Program ทำให้ไม่ต้อง Upgrade Client Program และสามารถใช้ผ่าน Internet Connection ที่มีความเร็วต่ำกว่า ทำให้ใช้โปรแกรมได้จากทุกแห่งในโลก

2.5 แนวคิด Software as a Service (SaaS)

SaaS (Software as a Service) เป็นแนวทางพัฒนาซอฟต์แวร์รูปแบบใหม่สำหรับการให้บริการซอฟต์แวร์โดยที่แต่ก่อนอาจจะต้องซื้อซอฟต์แวร์เป็นลิขสิทธิ์ (License) และลงโปรแกรมบนคอมพิวเตอร์ของตนเองเพื่อใช้งาน เมื่อถึงเวลาที่ผู้ผลิตทำการอัพเดทเป็นเวอร์ชั่นใหม่ผู้ใช้งานต้องดาวน์โหลดหรือซื้อซอฟต์แวร์ในเวอร์ชั่นใหม่ และถ้าหากมีผู้ใช้ซอฟต์แวร์เป็นจำนวนมากจะทำให้เสียเวลาและเงินอย่างมากในการอัพเดทซอฟต์แวร์ในแต่ละครั้ง ซึ่ง SaaS จะสามารถแก้ปัญหาในจุดนี้ได้โดยมองซอฟต์แวร์ เป็นเหมือนบริการอย่างหนึ่ง โดยผู้ใช้บริการเพียงแค่จ่ายเงินค่าบริการแล้วก็สามารถใช้งานซอฟต์แวร์ผ่านทางเว็บบราวน์เซอร์ได้ทันที เมื่อมีการอัพเดทซอฟต์แวร์ก็จะทำเองอัตโนมัติโดยผู้ผลิต SaaS มีข้อดีคือผู้ใช้จะสามารถวางแผนงบประมาณสำหรับการซื้อซอฟต์แวร์ได้ง่ายยิ่งขึ้นและใช้เวลาน้อยในการอัพเดทเวอร์ชั่นซอฟต์แวร์แต่ละครั้ง

ดังนั้น SaaS ก็คือ Web-Based Software ที่ผู้ใช้สามารถเรียกใช้บริการต่าง ๆ ของซอฟต์แวร์ผ่านทางเว็บ โดยผู้ใช้มิต้องคำนึงถึงว่าซอฟต์แวร์ที่ใช้อยู่จะมี Host Server ตั้งอยู่ที่ใด,ใช้ OS อะไร หรือว่าถูกเขียนขึ้นมาโดยใช้ภาษาคอมพิวเตอร์อะไร และไม่ต้อง Install โปรแกรมไว้บนเครื่องคอมพิวเตอร์ที่ใช้อยู่เลย ตัวอย่างของ SaaS ที่พบเห็นกันได้ในชีวิตประจำวันก็อย่างเช่น Web-Based Email Service ต่าง ๆ อย่าง Hotmail, Gmail, Yahoo ที่จะมีการเก็บโปรแกรมและข้อมูลต่าง ๆ ไว้ที่ Host แล้วให้ผู้ใช้สามารถเรียกใช้ Application ต่าง ๆ ผ่านทางเว็บไซต์ได้

วัตถุประสงค์ของ SaaS มีดังนี้

1. เพื่อให้การบริหารจัดการและควบคุมซอฟต์แวร์เป็นไปอย่างมีประสิทธิภาพ ตัวอย่างเช่น บริษัทที่ขายซอฟต์แวร์และประสบปัญหากับการถูกก็อปปี้โปรแกรมไปขาย ก็จะสามารถควบคุมปัญหาส่วนนี้ได้ง่ายขึ้น
2. เพื่อให้ผู้ใช้มีความสะดวกสบายมากขึ้น เพราะบางคนมักจะเจอปัญหา ตัวอย่างเช่น ซื้อโปรแกรมมาแต่ลงไม่เป็น หรือ ต้องคอยตามอัพเกรดโปรแกรมที่ใช้ ซึ่งบางครั้งไม่มีเวลา การใช้ SaaS จะตัดปัญหาเหล่านี้ไป เพราะบริษัทที่ให้บริการอยู่จะดูแลให้หมด
3. เพื่อให้การบริการซอฟต์แวร์เป็นไปได้อย่างทั่วถึง ทุกที่ ทุกเวลา บนโลกใบนี้
4. เพื่อให้สามารถใช้ซอฟต์แวร์เพียงชุดเดียวในการให้บริการแก่ผู้ใช้หลาย ๆ คน ซึ่งนี่เป็นสิ่งที่ผู้ผลิตซอฟต์แวร์ต้องการเนื่องจากจะช่วยประหยัดต้นทุนลงไปได้มาก
5. เพื่อให้การจ่ายเงินมีความยืดหยุ่นตามบริการที่ใช้ ใครใช้มากก็จ่ายมาก ใครใช้น้อยก็จ่ายน้อยลงมาตามลำดับ

SaaS สามารถแบ่งออกได้เป็น 2 ประเภท คือ

1. **Business-Orientated Services** คือ การให้บริการแก่องค์กรหรือหน่วยงานต่าง ๆ
2. **Customer-Orientated Services** คือ การให้บริการแก่บุคคลทั่ว ๆไป ส่วนใหญ่แล้วมักจะเป็นการให้บริการแบบฟรีๆ

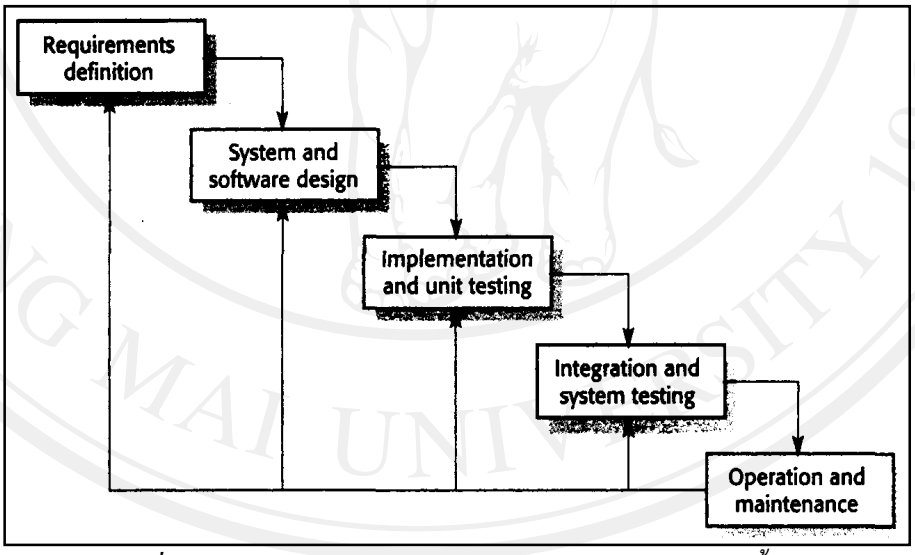
สรุปโดยทั่วไป SaaS จะมีลักษณะต่าง ๆ ดังต่อไปนี้

1. ให้บริการผ่านทาง Web Browser โดยผู้ใช้ไม่ต้องติดตั้งซอฟต์แวร์ลงบนเครื่องของตนเองและสามารถเรียกใช้งานผ่านเว็บไซต์ได้ทันที
2. มีลักษณะการใช้งานแบบ On-Demand คือ เมื่อผู้ใช้ Access เข้าไปใช้ในโปรแกรมครั้งหนึ่งแล้ว ต่อไปไม่ว่าจะอยู่ที่ไหนหรือเวลาใด ก็สามารถกลับไปใช้ซอฟต์แวร์นั้นได้ทุกเมื่อ

3. ใช้เท่าไหร่ก็จ่ายเท่านั้น หมายถึง จ่ายเงินตามจริงที่ใช้งาน ไม่ต้องเหมาจ่ายทั้งหมด
4. ไม่จำเป็นต้องมี IT Support คอยดูแล เพราะ SaaS ใช้งานง่ายมาก และโครงสร้างรวมทั้งระบบภายใน ทางบริษัทผู้ให้บริการจะเป็นผู้ดูแลให้ทั้งหมด

2.6 กระบวนการพัฒนาซอฟต์แวร์แบบจำลองน้ำตก (Waterfall Model)

แบบจำลองน้ำตกประกอบไปด้วยขั้นตอนการดำเนินงานที่เรียงต่อเนื่องกันเป็นลำดับขั้นตอนต่อไปจะเริ่มต้นดำเนินการได้จะต้องรอให้ขั้นตอนก่อนหน้าเสร็จสิ้นก่อน โดยขั้นตอนพื้นฐานในการดำเนินงานพัฒนาซอฟต์แวร์ในแบบจำลองน้ำตกประกอบด้วย 5 ขั้นตอนได้แก่ การกำหนดความต้องการ (Requirement Definition) การออกแบบซอฟต์แวร์และระบบ (System and Software Design) การลงมือพัฒนาและทดสอบในระดับหน่วย (Implementation and Unit Testing) การประสานระบบและทดสอบระบบ (Integration and System Testing) การนำไปใช้และบำรุงรักษา (Operation and Maintenance) ดังรูปที่



รูปที่ 2.9 แสดงกระบวนการพัฒนาซอฟต์แวร์แบบจำลองน้ำตก

แบบจำลองน้ำตก ประกอบไปด้วยขั้นตอนพื้นฐานในการดำเนินงานพัฒนาซอฟต์แวร์ ซึ่งมี 5 กระบวนการได้แก่

1. **การกำหนดความต้องการ (Requirement Definition)** เป็นกระบวนการในการกำหนดวัตถุประสงค์ การทำงานและขอบเขตของระบบ จากการประชุมกับผู้ใช้ระบบ แล้วนำมาอธิบายในรายละเอียด เพื่อสร้างเป็นเอกสารข้อกำหนดความต้องการของระบบ

2. **การออกแบบซอฟต์แวร์และระบบ (System and Software Design)** เป็นกระบวนการในการนำความต้องการของระบบ มาอธิบายรูปแบบสถาปัตยกรรมและละเอียดต่างๆ เพื่อระบุส่วนประกอบของระบบ การอธิบายการทำงานรวมถึงความสัมพันธ์ของส่วนต่างๆ แล้วสร้างเป็นเอกสารการออกแบบระบบ เพื่อสื่อสารให้ผู้พัฒนา เข้าใจตรงกัน
3. **การลงมือพัฒนาและทดสอบในระดับหน่วย (Implementation and Unit Testing)** ในกระบวนการนี้ ซอฟต์แวร์ที่ถูกออกแบบไว้ จะถูกสร้างให้ทำงานได้จริงในแต่ละส่วนตามความต้องการ พร้อมทั้งทดสอบในแต่ละส่วนแยกกัน เพื่อให้แน่ใจว่าการทำงานในแต่ละส่วนนั้น ตรงกับความต้องการมากที่สุด
4. **การประสานระบบและทดสอบระบบ (Integration and System Testing)** หลังจากที่พัฒนาในแต่ละส่วน ให้สามารถทำงานได้ตามความต้องการแล้ว ก็จะต้องนำแต่ละส่วนมาทำการประสานกันเป็นระบบ และทดสอบเพื่อให้แน่ว่าระบบโดยรวมทั้งหมดสามารถทำงานร่วมกันอย่างราบรื่น และตรงกับความต้องการมากที่สุด หลังจากกระบวนการนี้แล้ว ซอฟต์แวร์ก็พร้อมจะถูกส่งมอบให้ผู้ใช้ต่อไป
5. **การนำไปใช้และบำรุงรักษา (Operation and Maintenance)** เป็นกระบวนการที่มีช่วงระยะเวลานานที่สุดของวงจรชีวิตซอฟต์แวร์ ตั้งแต่การติดตั้งซอฟต์แวร์ การใช้งานและการบำรุงรักษาระบบ ให้สามารถทำงานได้อย่างราบรื่น ตลอดช่วงเวลาที่ซอฟต์แวร์ถูกใช้งาน รวมถึงการปรับเปลี่ยนซอฟต์แวร์ ให้ตอบสนองกับความต้องการใหม่ ตามรูปแบบธุรกิจหรือการใช้งานของผู้ใช้ที่เปลี่ยนแปลงไป

โดยทุกๆ กระบวนการจะต้องได้ผลลัพธ์ เพื่อนำไปใช้ในกระบวนการต่อไป และมีผลตอบกลับ เพื่อส่งกลับไปยังกระบวนการที่ต้องการต่อไป

ในบทนี้ได้แสดงข้อมูลสาระสำคัญ จากเอกสารต่างๆ ที่เกี่ยวข้องกับการวิจัยครั้งนี้ ซึ่งได้ทำการศึกษาและอ้างอิง เพื่อให้บรรลุตามวัตถุประสงค์และขอบเขตของการศึกษา ที่นำเสนอในบทก่อนหน้านี้ ซึ่งหลังจากได้ข้อมูลที่เกี่ยวข้องแล้ว ก็จะออกแบบวิธีการศึกษาวิจัยโดยอาศัยกระบวนการการพัฒนาซอฟต์แวร์แบบน้ำตก ซึ่งรายละเอียดการออกแบบวิธีการศึกษาวิจัยในกระบวนการต่างๆ จะได้ถูกกล่าวในบทถัดไป