บทที่ 4
วิศวกรรมซอฟต์แวร์และโมเดลการพัฒนาซอฟต์แวร์

การพัฒนาระบบสารสนเทศ คือการนำเอาคอมพิวเตอร์เข้ามาช่วยในการจัดเก็บและประมวลผลข้อมูล เพื่อได้สารสนเทศที่ตรงกับความต้องการในการใช้งาน ซึ่งเป็นงานที่มีความยุ่งยากและมีจำนวนขั้นตอนมากมาย จำเป็นต้องใช้หลักทางวิศวกรรมซอฟต์แวร์เข้ามากำหนดกระบวนการพัฒนา เพื่อทำให้กระบวนการในการพัฒนามีความชัดเจน สามารถตรวจสอบและควบคุมการปฏิบัติงานได้ง่าย ทำให้การพัฒนาระบบสารสนเทศสำเร็จลุล่วงตามกำหนด การทำงานมีมาตรฐานและมีคุณภาพ ระบบสารสนเทศหรือซอฟต์แวร์ที่ได้มีคุณภาพ ช่วยทำให้การดำเนินงานขององค์กรมีประสิทธิภาพมากยิ่งขึ้น ช่วยลดภาระค่าใช้จ่ายขององค์กร ทำให้องค์กรมีความน่าเชื่อถือและมีภาพลักษณ์ที่ดี

1. วิศวกรรมซอฟต์แวร์

วิศวกรรมซอฟต์แวร์ (Software Engineering) หมายถึงการประยุกต์ความรู้ทางวิศวกรรมศาสตร์มาใช้ในกระบวนการพัฒนาระบบสารสนเทศหรือซอฟต์แวร์คอมพิวเตอร์ โดยมีการกำหนดขั้นตอนในการปฏิบัติงานที่ชัดเจน เพื่อทำให้สามารถตรวจสอบและควบคุมคุณภาพของงานได้ง่าย ทำให้สามารถตรวจสอบคุณภาพและปรับปรุงแก้ไขซอฟต์แวร์ได้ง่าย และทำให้ได้ซอฟต์แวร์ที่สามารถนำไปใช้งานเพื่อช่วยแก้ปัญหาการทำงานขององค์กรได้จริง ในกระบวนการทางวิศวกรรมซอฟต์แวร์จะมีบุคคลที่ทำหน้าที่เป็นนักวิศวกรรมซอฟต์แวร์ ทำหน้าที่ควบคุมดูแลโครงการพัฒนาซอฟต์แวร์หรือโครงการพัฒนาระบบสารสนเทศ ตั้งแต่กิจกรรมแรกจนกระทั่งสิ้นสุดโครงการ เพื่อให้กระบวนการทำงานของโครงการพัฒนาระบบมีมาตรฐานและสามารถวัดผลได้ โดยเป้าหมายหลักของวิศวกรรมซอฟต์แวร์ก็คือ ลดการพึ่งพาความสามารถของบุคคลใดบุคคลหนึ่งโดยเฉพาะ และต้องการเพิ่มผลผลิตที่ดี (Productivity)

1.1 กิจกรรมพื้นฐานทางวิศวกรรมซอฟต์แวร์ กระบวนการพัฒนาระบบสารสนเทศโดยใช้หลักการทางวิศวกรรมซอฟต์แวร์ ประกอบด้วยกิจกรรม 4 กิจกรรม ดังนี้

1.1.1 วิศวกรรมความต้องการ (Requirements Engineering) คือการระบุข้อกำหนดซอฟต์แวร์ (Software Specification) ความสามารถในการทำงานของซอฟต์แวร์และเงื่อนไขในการปฏิบัติงาน โดยประกอบด้วยกิจกรรมย่อย 4 กิจกรรม คือ

1) การศึกษาความเป็นไปได้ (Feasibility Study)

2) การวิเคราะห์ความต้องการ (Requirements Analysis)

3) การสรุปเป็นข้อกำหนดลงในเอกสาร (Requirements Specification)

4) การตรวจสอบความต้องการ (Requirements Validation)

1.1.2 การพัฒนาซอฟต์แวร์ (Software Development) คือการพัฒนาหรือสร้างผลิตภัณฑ์ให้ตรงตามข้อกำหนด ด้วยการนำระเบียบวิธีการพัฒนาซอฟต์แวร์ (Methodology) มาใช้กับการพัฒนาซอฟต์แวร์ เพื่อให้กระบวนการพัฒนาซอฟต์แวร์มีมาตรฐาน และได้ซอฟต์แวร์ที่มีคุณภาพ สามารถนำซอฟต์แวร์ไปใช้งานได้จริง

1.1.3 การตรวจสอบความถูกต้องของซอฟต์แวร์ (Software Validation) คือการตรวจสอบความสามารถในการทำงานของซอฟต์แวร์ สามารถทำงานได้อย่างถูกต้องตรงกับข้อกำหนดที่ระบุไว้ในเอกสารหรือไม่ ตรวจสอบความถูกต้องของรายงานหรือสารสนเทศที่ได้จากการทำงานของซอฟต์แวร์

1.1.4 วิวัฒนาการของซอฟต์แวร์ (Software Evolution)โครงการซอฟต์แวร์ขนาดใหญ่จะมีระยะเวลาในการพัฒนาระบบนานและระบบที่ได้จะมีความซับซ้อนสูง ระบบที่พัฒนาจะมีวิวัฒนาการของการตรวจสอบข้อผิดพลาดตามข้อกำหนดและอาจมีข้อกำหนดหรือความต้องการใหม่ที่เพิ่มเข้ามา ดังนั้นซอฟต์แวร์อาจมีการเปลี่ยนแปลงระหว่างการพัฒนา และมีความเป็นไปได้ที่ระบบย่อยบางระบบอาจไม่มีความเป็นอิสระ หากระบบย่อยใดระบบหนึ่งมีการเปลี่ยนแปลงย่อมส่งผลกระทบต่อระบบย่อยอื่นๆ ทำให้ระบบย่อยที่ได้รับผลกระทบนั้น จำเป็นต้องได้รับการเปลี่ยนแปลงไปตามข้อกำหนดใหม่ ดังนั้นซอฟต์แวร์ที่ดี ควรได้รับออกแบบเพื่อรองรับวิวัฒนาการที่สามารถเปลี่ยนแปลงไปตามความต้องการของผู้ใช้ได้อย่างเหมาะสม

กระบวนการทางวิศวกรรมซอฟต์แวร์จะเน้นการนำเทคนิคต่างๆ มาใช้ให้ก่อเกิดประโยชน์สูงสุด เพื่อให้ได้มาซึ่งความสมบูรณ์และคุณภาพของตัวผลิตภัณฑ์ซอฟต์แวร์และสามารถส่งมอบผลิตภัณฑ์ให้กับลูกค้าตรงตามเวลาที่กำหนดไว้ก่อนเริ่มต้นโครงการ คุณภาพของซอฟต์แวร์นั้นจะขึ้นอยู่กับกระบวนการในการพัฒนาซอฟต์แวร์ หากกระบวนการพัฒนาซอฟต์แวร์มีมาตรฐานที่ดีย่อมส่งผลให้ได้ผลิตภัณฑ์ที่มีคุณภาพด้วย และกระบวนการพัฒนาซอฟต์แวร์ยังขึ้นอยู่กับขนาดของโครงการ หากโครงการมีขนาดเล็กกระบวนการที่ใช้ในการพัฒนาซอฟต์แวร์ย่อมมีความซับซ้อนน้อยกว่าโครงการซอฟต์แวร์ขนาดใหญ่ ซึ่งจำเป็นต้องใช้ประสบการณ์และความสามารถของทีมงานค่อนข้างสูงกว่าโครงการขนาดเล็ก หากนำเทคนิคในการพัฒนาซอฟต์แวร์ที่เหมาะสมมาใช้จะทำให้การพัฒนาซอฟต์แวร์มีประสิทธิภาพได้มาซึ่งซอฟต์แวร์ที่มีคุณภาพ และสามารถนำซอฟต์แวร์นั้นไปใช้งานจริงตามระยะเวลาที่ต้องการได้

1.2 คุณสมบัติของซอฟต์แวร์ที่มีคุณภาพ ซอฟต์แวร์ที่ได้จากโครงการพัฒนาซอฟต์แวร์จะต้องมีคุณภาพ สามารถนำไปใช้กับการทำงานขององค์กรได้จริง และก่อให้เกิดประโยชน์ต่อองค์กร ช่วยทำให้การทำงานขององค์กรมีประสิทธิภาพมากยิ่งขึ้น โดยคุณสมบัติของซอฟต์แวร์ที่มีคุณภาพประกอบด้วย

(1) ความถูกต้อง (Correctness) คือซอฟต์แวร์สามารถประมวลผลได้ถูกต้องและตรงตามความต้องการใช้งานของผู้ใช้

(2) ความน่าเชื่อถือ (Reliability) คือผลลัพธ์ที่ได้จากการทำงานของซอฟต์แวร์จะต้องมีความน่าเชื่อถือ สามารถนำไปใช้อ้างอิงในการทำงานขององค์กรหรือช่วยในการตัดสินใจของผู้บริหารระดับต่างๆ ขององค์กรได้

(3) ใช้งานง่าย (User Friendliness) คือซอฟต์แวร์ถูกออกแบบให้สามารถใช้งานได้ง่ายและง่ายต่อการเข้าใจ มีเอกสารหรือข้อความอธิบายในการใช้งานครบทุกส่วน

(4) การบำรุงรักษาง่าย (Maintainability) คือการเขียนคำสั่งควบคุมการทำงานของซอฟต์แวร์จะต้องมีความเป็นระเบียบ จัดย่อหน้าคำสั่งให้สามารถอ่านและเข้าใจได้ง่าย มีข้อความอธิบายส่วนของคำสั่งต่างๆ เพื่อให้สามารถปรับปรุงแก้ไขได้ง่าย

(5) สามารถนำกลับมาใช้งานใหม่ได้ (Reusability) คือในการออกแบบและเขียนคำสั่งควบคุมการทำงาน ของส่วนต่างๆ ในซอฟต์แวร์ ควรมีความเป็นอิสระต่อกันให้มากที่สุด เพื่อให้สามารถนำส่วนงานนั้นกลับมาใช้ใหม่ได้ง่าย ทำให้ช่วยลดต้นทุนและเวลาในการพัฒนา เช่น การเขียนโปรแกรมเชิงวัตถุ (Object Oriented Program)

(6) มีความคงทน (Robustness) คือการทำงานของซอฟต์แวร์มีความถูกต้องทุกครั้งที่เรียกใช้งาน แม้จะมีปัจจัยที่ก่อให้เกิดการเปลี่ยนแปลง เช่นการใช้งานซอฟต์แวร์โดยใช้อุปกรณ์การเข้าถึงที่แตกต่างกัน แต่ผลลัพธ์ที่ได้จากการทำงานก็ยังคงมีความถูกต้องอยู่เสมอให้ผลลัพธ์ที่ไม่แตกต่างกัน แม้ต่างอุปกรณ์กันก็ตาม

(7) มีประสิทธิภาพ (Efficiency) คือผลลัพธ์ที่ได้จากการทำงานของซอฟต์แวร์ช่วยทำให้การทำงานขององค์กรสะดวกรวดเร็วขึ้นกว่าระบบงานเดิม สามารถเพิ่มปริมาณการทำงานหรือการให้บริการขององค์กรขึ้นจากเดิม และช่วยลดค่าใช้จ่ายขององค์กร ทำให้องค์กรมีรายได้เพิ่มขึ้น

(8) ความสะดวกในการเคลื่อนย้าย (Portability) คือสามารถเคลื่อนย้ายสะดวก สามารถนำไปติดตั้งได้บนทุกอุปกรณ์ และทุกระบบที่มีแพลตฟอร์มแตกต่างกัน

(9) มีความปลอดภัย (Security Safety) มีระบบรักษาความปลอดภัยในการใช้งานซอฟต์แวร์ เพื่อป้องกันบุคคลที่ไม่หวังดีต่อระบบ มีการกำหนดสิทธิ์ในการใช้งานของผู้ใช้แต่ละ

ระดับ มีระบบสำรองข้อมูลที่มีประสิทธิภาพเพื่อป้องกันข้อมูลสูญหาย มีระบบกู้คืนข้อมูลเมื่อเกิดความเสียหายต่อข้อมูล

วิศวกรรมซอฟต์แวร์เป็นการนำกฎเกณฑ์ แบบแผน และระเบียบวินัยมาใช้ควบคุมการพัฒนาซอฟต์แวร์ให้มีคุณภาพ หากพัฒนาระบบโดยไม่มีการวางแผนจะทำให้เสียเวลาในการทำงาน เพราะอาจจะต้องเสียเวลาไปกับการทดสอบระบบและการปรับปรุงระบบเพื่อให้ได้ระบบตรงตามที่ต้องการ โดยในการพัฒนาระบบจะนิยมนำวงจรการพัฒนาระบบมาใช้เป็นแผนงานในการทำงาน วงจรการพัฒนาระบบเป็นขั้นตอนการพัฒนาระบบที่มีลำดับขั้นตอนการทำงานในรูปเชิงเส้น (Linear) นั่นคือ เมื่อเสร็จสิ้นระยะของการวิเคราะห์แล้ว ระยะถัดไปก็คือการออกแบบ ซึ่งจะไม่สามารถย้อนกลับไปยังขั้นตอนก่อนหน้าได้ ในขณะที่ระยะการทดสอบและการนำไปใช้จะถูกจัดอยู่ในลำดับท้ายๆ จึงมีความเสี่ยงสูง หากโครงการได้รับการจัดการไม่ดีพอ ระบบไม่ตรงตามความต้องการของผู้ใช้ มีการปรับปรุงแก้ไขอย่างต่อเนื่อง ก็จะทำให้ต้องสูญเสียเวลาไปกับการบำรุงรักษา ซึ่งเป็นระยะที่ยาวนานและมีค่าใช้จ่ายสูง ดังนั้น กระบวนการพัฒนาซอฟต์แวร์ตามแนวคิดวิศวกรรมซอฟต์แวร์ จึงผนวกขั้นตอนในลักษณะการกลับไปทวนซ้ำ (Iteration) การวัดความก้าวหน้า (Incremental) และการทำต้นแบบ (Prototyping) ซึ่งกระบวนการดังกล่าว จะสามารถช่วยให้มีการตรวจทานซอฟต์แวร์ได้ดียิ่งขึ้น ลดความเสี่ยงและควบคุมการเปลี่ยนแปลงของซอฟต์แวร์ได้

จึงสรุปได้ว่า วิศวกรรมซอฟต์แวร์ คือระเบียบแบบแผนทางวิศวกรรมที่นำมาใช้ควบคุมและดำเนินการผลิตซอฟต์แวร์ให้มีประสิทธิภาพ สามารถตรวจสอบข้อผิดพลาดและแก้ไขซอฟต์แวร์ในระหว่างการผลิตซอฟต์แวร์ได้ อีกทั้งสามารถระบุสาเหตุ และความสามารถในการวัดผล โดยมีการนำเครื่องมือสนับสนุนการพัฒนาระบบมาใช้ เพื่อให้ทำงานสะดวก ตรวจสอบข้อผิดพลาดได้ตรงจุด และเป็นไปตามมาตรฐานเดียวกัน เพื่อก่อให้เกิดซอฟต์แวร์ที่ดี มีคุณภาพสูง

2. ระเบียบวิธีการพัฒนาซอฟต์แวร์

ระเบียบวิธีการพัฒนาซอฟต์แวร์ (Software Development Methodology) หรือโมเดลการพัฒนาซอฟต์แวร์ เป็นแบบจำลองที่ใช้สำหรับระบุกิจกรรมหลัก (Key Activities) ในการพัฒนาซอฟต์แวร์ เพื่อให้การพัฒนาซอฟต์แวร์เกิดปัญหาน้อยที่สุด โมเดลการพัฒนาซอฟต์แวร์มีให้เลือกใช้หลายโมเดล การเลือกใช้โมเดลจะขึ้นอยู่กับปัจจัยหลายๆ ปัจจัย เช่น ขนาดของโครงการการพัฒนาระบบสารสนเทศ ความซับซ้อนของโครงการการพัฒนาระบบสารสนเทศ ความเหมาะสมและระดับความเสี่ยงในการพัฒนาซอฟต์แวร์

สาเหตุจำเป็นต้องใช้โมเดลการพัฒนาซอฟต์แวร์ เนื่องจาก

(1) โมเดลการพัฒนาซอฟต์แวร์ มีการแตกกระบวนการการพัฒนาซอฟต์แวร์ออกเป็นระยะ เพื่อให้การดำเนินงานเป็นไปได้ด้วยความสะดวก และง่ายต่อการดำเนินงาน

(2) ซอฟต์แวร์ที่พัฒนามีความซับซ้อน จึงต้องมีการวางแผนการทำงานให้เป็นขั้นตอน เพื่อให้ซอฟต์แวร์สำเร็จตามระยะเวลาที่กำหนดไว้

(3) การแบ่งกระบวนการพัฒนาเป็นระยะ (Phase) ทำให้ง่ายต่อการบริหารจัดการ

(4) แต่ละระยะมีแนวทางต่างๆ ให้เลือกปฏิบัติ

โมเดลการพัฒนาซอฟต์แวร์มีให้เลือกใช้มากมายตามแนวทางในการพัฒนา ซึ่งมีขั้นตอนและกระบวนการที่แตกต่างกันไปในแต่ละโมเดล เอกสารฉบับนี้ขอยกตัวอย่างโมเดลการพัฒนาซอฟต์แวร์ 8 โมเดล ดังนี้

1) Built-and-Fix Model
2) Waterfall Model
3) Incremental Model
4) Spiral Model
5) Joint Application Development (JAD)
6) Rapid Application Development (RAD)
7) Unified Process (UP)
8) Agile Methodologies

โดยแต่ละโมเดลมีรายละเอียดดังนี้

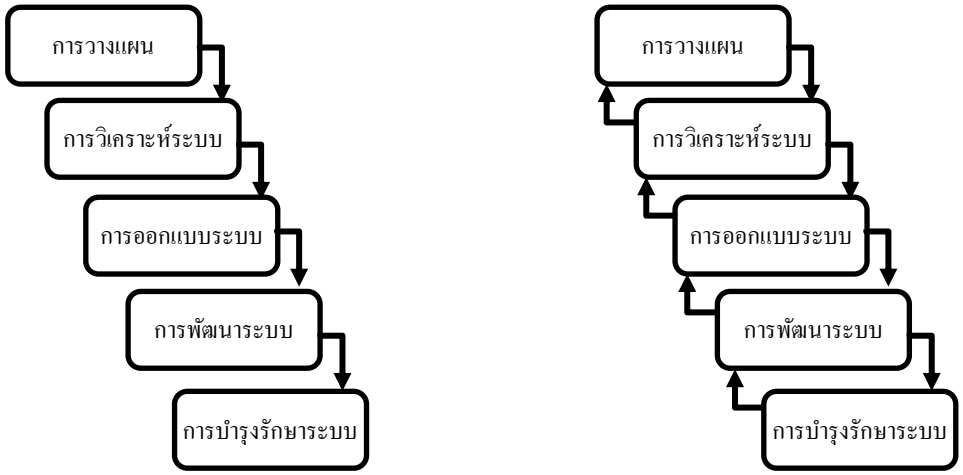
2.1 Built-and-Fix Model เป็นโมเดลการพัฒนาซอฟต์แวร์ที่มีการเขียนโปรแกรม และแก้ไขโปรแกรมไปเรื่อยๆ จากการลองผิดลองถูก จนกระทั่งพอใจหรือผลลัพธ์ที่ได้ตรงตามความต้องการของผู้ใช้ กระบวนการนี้จะทำให้เสียเวลาไปกับการแก้ไขโปรแกรม และการบำรุงรักษาระบบ เหมาะสำหรับการพัฒนาซอฟต์แวร์ขนาดเล็กไม่ซับซ้อน หรืองานที่เกิดข้อผิดพลาดแล้ว ไม่ส่งผลกระทบต่อระบบมากนัก แต่ไม่เหมาะสำหรับการพัฒนาซอฟต์แวร์ขนาดใหญ่ เนื่องจากซอฟต์แวร์ขนาดใหญ่จะมีระบบย่อยและรายละเอียดค่อนข้างมากจะทำให้เสียเวลาและสิ้นเปลืองต้นทุนและบุคลากรในการพัฒนาระบบกว่าจะได้ซอฟต์แวร์ที่มีความสมบูรณ์พร้อมใช้งาน

ขั้นตอนของโมเดล Built-and-Fix Model ประกอบด้วย

(1) เขียนโปรแกรมบางส่วนที่สามารถแก้ไขปัญหาได้
(2) คอมไพล์ และรันโปรแกรมเพื่อทดสอบ
(3) หากพบข้อผิดพลาดในโปรแกรม ก็ดำเนินการแก้ไขปรับปรุง

(4) กลับไปทำซ้ำตาม ขั้นตอนที่ 1-4 จนกระทั่งได้ผลลัพธ์ตรงตามความต้องการ

2.2 Waterfall Model หรือโมเดลน้ำตก เป็นโมเดลที่ได้รับความนิยมนำมาในในการพัฒนาระบบสารสนเทศเป็นอย่างมาก เนื่องจากมีขั้นตอนการดำเนินงานที่ชัดเจนและง่ายต่อการนำไปใช้จริง การดำเนินงานของโมเดลน้ำตกในยุคแรกจะดำเนินงานทีละขั้นตอนให้เสร็จสิ้น จึงจะดำเนินงานในขั้นตอนต่อไป นั่นหมายความว่าต้องดำเนินงานในขั้นตอนที่หนึ่งให้เสร็จสิ้นก่อน จึงจะดำเนินงานในขั้นตอนที่สองได้ ไม่สามารถจะข้ามไปดำเนินงานในขั้นตอนใดก่อนก็ได้ และเมื่อดำเนินการขั้นตอนนั้นเสร็จสิ้นแล้วจะไม่สามารถการย้อนกลับมาดำเนินงานในขั้นตอนนั้นได้อีก เปรียบเสมือนน้ำตกที่ไม่มีการไหลย้อนกลับ แสดงดังรูป 4.1(a) โมเดลน้ำตกแบบดั้งเดิม ซึ่งในการดำเนินงานจริงๆ พบว่าปัญหาส่วนใหญ่ที่เกิดขึ้นมักจะไม่ใช่ปัญหาในขั้นตอนการทำงานปัจจุบัน แต่เป็นปัญหาจากการดำเนินงานขั้นตอนก่อนหน้า แต่ไม่สามารถย้อนกลับไปตรวจสอบได้ จึงทำให้โครงการพัฒนาระบบสารสนเทศล้มเหลว นั่นคือระบบที่พัฒนาอาจมีคุณสมบัติไม่ตรงตามความต้องการของผู้ใช้ หรือไม่สามารถตอบสนองการทำงานได้อย่างแท้จริง จึงมีการปรับปรุงโมเดลน้ำตกให้สามารถย้อนกลับไปตรวจสอบการทำงานของขั้นตอนก่อนหน้าได้ เพื่อให้เกิดความถูกต้องในการทำงาน จึงเกิดโมเดลน้ำตกรูปแบบใหม่แบบวนซ้ำ แสดงดังรูปที่ 4.1(b)



(a) โมเดลน้ำตกแบบดั้งเดิม (b) โมเดลน้ำตกรูปแบบใหม่แบบทวนซ้ำ

รูปที่ 4.1 โมเดลน้ำตกแบบดั้งเดิม และโมเดลน้ำตกรูปแบบใหม่แบบทวนซ้ำ

จากรูปที่ 4.1(a) โมเดลน้ำตกแบบดั้งเดิม ที่ไม่สามารถย้อนกลับไปดำเนินงานในขั้นตอนก่อนหน้าได้ ส่วนรูปที่ 4.1(b) โมเดลน้ำตกรูปแบบใหม่แบบทวนซ้ำ เป็นโมเดลที่ทำการปรับปรุงมา

จากโมเดลน้ำตกแบบดั้งเดิม ในการดำเนินงานสามารถย้อนกลับไปตรวจสอบการทำงานในขั้นตอนก่อนหน้าได้ ขั้นตอนการดำเนินงานโมเดลน้ำตกประกอบด้วย 5 ขั้นตอน คือ

(1) การวางแผน เป็นขั้นตอนวางแผนในการดำเนินงานพัฒนาระบบสารสนเทศ โดยในขั้นตอนนี้ผู้พัฒนาระบบจะทำการรวบรวมความต้องการต่างๆ จากผู้ใช้ หรือเจ้าของระบบ เมื่อสิ้นสุดขั้นตอนจะได้คุณสมบัติของระบบที่ผู้ใช้ต้องการ จากนั้นนำคุณสมบัตินั้นมาทำข้อตกลงร่วมกันทั้งสองฝ่าย เพื่อเป็นหลักฐานยืนยันในการพัฒนาระบบ

(2) การวิเคราะห์ระบบ เป็นขั้นตอนที่นำข้อมูลคุณสมบัติระบบ จากขั้นตอนการวางแผนมาทำการวิเคราะห์ เพื่อสรุปหาข้อมูลที่เกี่ยวข้องกับระบบ ศึกษาขั้นตอนในการดำเนินงานของระบบงานเดิม เพื่อวิเคราะห์หาข้อมูลที่ระบบจะต้องจัดเก็บ สร้างแผนผังแสดงความสัมพันธ์ระหว่างข้อมูลและระบบ สร้างแผนผังระบบที่จะพัฒนาเพื่อนำข้อมูลไปใช้ในขั้นตอนการออกแบบระบบ

(3) การออกแบบระบบ คือการออกแบบรายงานต่างๆ ที่ระบบจะต้องสร้างขึ้น ออกแบบส่วนต่อประสานระหว่างผู้ใช้และระบบ ออกแบบข้อมูลที่จะจัดเก็บในระบบ ออกแบบกระบวนการทำงานของระบบ ซึ่งในขั้นตอนนี้ หากระบบที่พัฒนามีขนาดใหญ่จะทำให้เสียเวลาในการทำงานค่อนข้างมาก เพราะจะต้องรอให้ออกแบบระบบทั้งหมดให้เสร็จสิ้นก่อนจึงจะเข้าสู่กระบวนการของการพัฒนาระบบ

(4) การพัฒนาระบบ เป็นขั้นตอนที่นำข้อมูลที่ได้จากขั้นตอนการออกแบบระบบมาพัฒนาเป็นระบบงานสารสนเทศที่สามารถใช้งานได้จริง เป็นการเปลี่ยนข้อมูลที่อยู่ในรูปแบบเอกสารให้เป็นระบบงานที่สามารถจับต้องได้ และสามารถใช้งานได้จริง จากนั้นนำระบบงานที่พัฒนาไปติดตั้งเพื่อใช้งานได้จริง

(5) การบำรุงรักษาระบบ เมื่อนำระบบงานที่พัฒนาไปใช้งานได้สักระยะเวลาหนึ่ง อาจพบข้อผิดพลาดจากการทำงานของระบบ หรือรายงานที่ได้จากการทำงานไม่สามารถตอบสนองความต้องการของหน่วยงานหรือองค์กร จำเป็นต้องมีการปรับปรุงแก้ไขหรือเพิ่มเติมให้สามารถตอบสนองความต้องการที่เพิ่มขึ้นได้

ข้อดีของโมเดลน้ำตกคือได้ระบบสารสนเทศที่ตรงตามความต้องการของผู้ใช้ เนื่องจากมีการสอบถามความต้องการจากผู้ใช้และจัดทำเอกสารข้อตกลงความต้องการ เพื่อความถูกต้องตรงกันก่อนดำเนินการพัฒนาระบบ ข้อเสียคือใช้เวลาในการดำเนินงานบางกิจกรรม เช่นกิจกรรมการวิเคราะห์และออกแบบระบบค่อนข้างมาก เพราะต้องออกแบบระบบให้เสร็จสิ้นก่อน จึงทำการพัฒนาโปรแกรม ดังนั้นจะได้ระบบสารสนเทศที่สมบูรณ์ก็ต่อเมื่อสิ้นสุดขั้นตอนสุดท้ายของโมเดลซึ่งใช้เวลานาน และหากระบบสารสนเทศที่ได้มีคุณสมบัติไม่ตรงตามความต้องการของผู้ใช้ จะต้อง

แก้ไขปรับปรุง ทำให้เสียเวลาเพิ่มขึ้นและยังสิ้นเปลืองค่าใช้จ่าย เพราะต้องกลับไปกิจกรรมการวิเคราะห์และออกแบบระบบใหม่อีกรอบ

2.3 Incremental Model หรือโมเดลแบบก้าวหน้า เป็นโมเดลที่วิวัฒนาการมาจากโมเดลน้ำตก เนื่องจากโมเดลน้ำตกมีข้อเสียตรงที่จะต้องดำเนินการขั้นตอนให้เสร็จสิ้นก่อนจึงจะดำเนินการขั้นตอนต่อไป ซึ่งหากเป็นโครงการพัฒนาซอฟต์แวร์ที่มีขนาดใหญ่อาจต้องใช้เวลามาก ทำให้มีความเสี่ยงสูงกับโอกาสที่จะต้องย้อนกลับไปเริ่มต้นโครงการใหม่ทั้งหมด หากมีการวางแผนจัดการที่ไม่ดีพอ หลักการของ Incremental Model คือ การแบ่งระบบงานออกเป็นระบบย่อยต่างๆ โดยระบบย่อยเรียกว่า Increment ซึ่งเปรียบเสมือนกับโครงการขนาดเล็ก (Mini-Project) โดยจะทำการพัฒนาระบบงานที่เป็นงานหลักของระบบก่อน จากนั้นจึงพัฒนาต่อเติมในแต่ละ Increment ตามลำดับ จนกระทั่งได้ระบบงานที่เสร็จสมบูรณ์ จากการแบ่งงานออกเป็นระบบย่อยนี้ หากเกิดผลกระทบใดๆขึ้นมา จะส่งผลงานในระบบย่อยเท่านั้น และระบบย่อยจะมีการพัฒนาแบบทวนซ้ำเป็นรอบ มีกระบวนการตรวจสอบความถูกต้อง เพื่อให้งานที่ได้ตรงกับความต้องการ การพัฒนาโปรแกรมโดยใช้โมเดลนี้จะมีความก้าวหน้าของระบบขึ้นเรื่อยๆ แต่ละระยะหรือแต่ละ Increment จะได้ส่วนย่อยของระบบซอฟต์แวร์ และระบบจะสมบูรณ์ขึ้นเรื่อยๆ จนได้ระบบที่สมบูรณ์ในที่สุด

ขั้นตอนการทำงานของโมเดลแบบก้าวหน้า ประกอบด้วย

(1) การศึกษาความเป็นไปได้ของระบบ จากนั้นจะทำการตรวจสอบความถูกต้องของการศึกษาความเป็นไปได้ เมื่อผลของการศึกษาความเป็นไปได้ของการพัฒนาระบบมีความเหมาะสมในการพัฒนาระบบก็จะดำเนินการขั้นตอนต่อไป

(2) การวางแผนและการกำหนดความต้องการ ในขั้นตอนนี้จะทำการวางแผนในการพัฒนาระบบและกำหนดความต้องการต่างๆ ของระบบ จากนั้นจะทำการตรวจสอบความถูกต้องของข้อกำหนดความต้องการ

(3) ขั้นตอนการออกแบบระบบ (Product Design) โดยแตกระบบเป็นระบบย่อย พัฒนาและตรวจสอบระบบย่อยทีละระบบ ในขั้นตอนนี้จะเกิดความก้าวหน้าของระบบ (Increment) โดยแต่ละรอบของการพัฒนาระบบย่อยประกอบด้วยขั้นตอนการทำงาน 5 ขั้นตอน และมีทวนซ้ำในแต่ละความก้าวหน้าของระบบย่อย ซึ่งขั้นตอนการทำงานของแต่ละรอบประกอบด้วย

(3.1) การออกแบบรายละเอียดของระบบย่อย พร้อมทั้งตรวจสอบความถูกต้อง

(3.2) เขียนโปรแกรม และทดสอบโปรแกรมหน่วยย่อยต่างๆ (Unit Testing)

(3.3) นำโปรแกรมย่อยต่างๆ มาประกอบรวมกัน (Integration) และตรวจสอบความถูกต้องของผลิตภัณฑ์ (Product Verification) ว่าทำงานได้อย่างถูกต้องหรือไม่

(3.4) การนำระบบไปใช้งาน จะมีการทดสอบระบบ (System Testing) ว่าระบบทำงานได้อย่างถูกต้อง และเป็นไปตามความต้องการของผู้ใช้หรือไม่

(3.5) ขั้นตอนการดำเนินงานและบำรุงรักษา จะเป็นการทบทวนเพื่อตรวจสอบความถูกต้อง ว่าระบบตรงตามความต้องการของผู้ใช้หรือไม่ (Revalidation)

จากขั้นตอนการทำงาน ทั้ง 5 ขั้นตอนจะเกิดขึ้นในแต่ละรอบของการพัฒนาระบบย่อย และนำระบบย่อยมารวมกันจนเป็นระบบที่สมบูรณ์พร้อมใช้งานในที่สุด จากการพัฒนาซอฟต์แวร์โดยใช้โมเดลความก้าวหน้านี้ จะได้ซอฟต์แวร์ที่สามารถใช้งานได้ทีละส่วนงานย่อยโดยไม่ต้องรอให้เสร็จสิ้นกระบวนการของการพัฒนาซอฟต์แวร์

ในการดำเนินการวนรอบ Increment แรกเสร็จสิ้น จะได้ระบบงานย่อยชิ้นแรก จากนั้นจะเริ่มวนรอบ Increment ถัดไป โดยนำระบบย่อยก่อนหน้ามารวมกับระบบย่อยที่กำลังพัฒนา จากนั้นทำการปรับปรุงและทดสอบระบบ จนระบบย่อยสามารถทำงานร่วมกันได้ ผลลัพธ์จากการทำงานของระบบย่อยหนึ่งอาจเป็นข้อมูลนำเข้าของระบบงานย่อยอื่นก็ได้ โดยแต่ละรอบของ Increment ก็จะทำให้ได้ระบบที่ใหญ่ขึ้นเรื่อยๆ จนกระทั่งได้ระบบงานที่สมบูรณ์ในที่สุด ในการวนรอบแต่ละรอบจะมีการตรวจสอบความถูกต้องในการทำงานของระบบที่พัฒนา ประกอบด้วยการตรวจสอบความถูกต้องของระบบเป็นไปตามข้อกำหนดความต้องการที่กำหนดขึ้นก่อนเริ่มต้นการพัฒนาระบบหรือไม่ ซึ่งเรียกว่า Verification และตรวจสอบความถูกต้องว่าระบบที่ได้ตรงตามความต้องการของผู้ใช้ (User's Requirements) หรือไม่ เรียกว่า Validation เพื่อเป็นการยืนยันถึงความถูกต้องในการทำงานของระบบที่พัฒนาตรงตามความต้องการของผู้ใช้ โดยที่ การทวนซ้ำในแต่ละรอบของโมเดล ถือว่าเป็นการตรวจสอบความถูกต้องในการทำงานของระบบ เพราะการทวนซ้ำคือการทวนซ้ำของงาน อาจจะมากกว่าหนึ่งรอบ เพื่อทำการตรวจสอบเพิ่มเติม หรือแก้ไขข้อผิดพลาดที่เกิดขึ้น ทำให้ช่วยลดความเสี่ยงในการปฏิบัติงานลงได้ จำนวนรอบที่ทวนซ้ำจะขึ้นอยู่กับความซับซ้อนของระบบ และการเพิ่มความก้าวหน้า คือการแบ่งระบบงานออกเป็นระบบย่อย และในการพัฒนาระบบย่อยมีการทวนซ้ำของงาน เพื่อให้ระบบมีความถูกต้องตรงตามความต้องการ ทำให้โครงการมีความก้าวหน้าขึ้นเรื่อยๆ จนได้ระบบที่สมบูรณ์ในที่สุด ในการส่งมอบระบบสามารถทยอยส่งมอบเป็นระบบย่อยได้ ทำให้ผู้ใช้สามารถใช้ซอฟต์แวร์ได้ในเวลาอันรวดเร็ว โดยไม่ต้องรอให้ระบบงานเสร็จสมบูรณ์ แต่ข้อเสีย คือมีความเสี่ยงสูง เนื่องจากมีการนำระบบย่อยมาประกอบรวมกันจนได้ระบบงานที่สมบูรณ์ ระบบงานย่อยที่ใช้งานอยู่อาจจะทำให้การทำงานสะดุดลงได้

2.4 Spiral Model มีหลักการทำงานแบบวนเป็นรอบคล้ายก้นหอยวนตามเข็มนาฬิกา เป็นวิธีการพัฒนาแบบค่อยเป็นค่อยไปทีละรอบ โดยเมื่อจบการทำงานในแต่ละรอบ จะได้ระบบงานที่สามารถใช้งานได้ โดยระบบงานที่ได้แต่ละรอบจะเรียกเป็นเวอร์ชั่น และในแต่ละรอบจะมีการ

วิเคราะห์ความเสี่ยง เพื่อประเมินและวางแผนการทำงานในรอบถัดไป วงจรการทำงานของ Spiral แบ่งออกเป็น 4 ส่วน ดังนี้

(1) การวางแผน (Planning) เป็นการกำหนดจุดมุ่งหมายของโครงการ กำหนดเงื่อนไขในการดำเนินงาน ระบุข้อกำหนดของระบบงานที่ต้องการ ระบุข้อจำกัดในด้านต่างๆ เช่นทีมงานในการดำเนินงาน สภาพแวดล้อมของการพัฒนาระบบและศึกษาหาแนวทางต่างๆที่นำมาใช้แก้ไขปัญหา รวมถึงการพิจารณาถึงต้นทุนที่ใช้ ผลประโยชน์ที่จะได้รับจากการใช้ระบบสารสนเทศที่พัฒนาเรียบร้อยแล้ว

(2) การวิเคราะห์ความเสี่ยง (Risk analysis) เป็นการนำแนวทางในการแก้ไขปัญหาต่างๆ มาประเมินหาความเสี่ยง จากนั้นคัดเลือกแนวทางที่ดีที่สุดและมีความเป็นไปได้สูงสุดมาใช้ในการพัฒนาระบบ เพื่อจัดการความเสี่ยงหรือหลีกเลี่ยงความเสี่ยงที่จะเกิดขึ้น เช่น ความเสี่ยงของโครงการจะล้มเหลว ความเสี่ยงของคุณภาพระบบที่พัฒนาแล้วเสร็จ ความเสี่ยงทางธุรกิจที่เกิดจากการใช้งานระบบที่พัฒนาเรียบร้อยแล้ว ความเสี่ยงต่างๆ นี้ สามารถแก้ปัญหาได้ด้วยการพัฒนาต้นแบบ (Prototype) หรือการจำลองสถานการณ์เพื่อวิเคราะห์หาความเสี่ยง

(3) การพัฒนาและทดสอบระบบ (Engineering) เป็นการพัฒนาตัวต้นแบบตามข้อกำหนดที่กำหนดไว้ในขั้นตอนการวางแผน และในขั้นตอนนี้จะเป็นการพัฒนาระบบต่อยอดจากของเดิมที่เคยพัฒนาในรอบก่อนหน้า ระบบที่พัฒนาจะมีความสามารถเพิ่มเติมจากระบบเดิม ระบบงานที่ได้แต่ละรอบจะเรียกเป็นเวอร์ชั่น จากนั้นทำการทดสอบการทำงานของระบบให้ตรงตามข้อกำหนดที่กำหนดไว้ในขั้นตอนการวางแผน

(4) การประเมิน (Evaluation) เป็นการทบทวนผลลัพธ์ของการทำงานในขั้นตอนที่ผ่านมาร่วมกับเจ้าของระบบ แล้วทำการวางแผนเพื่อเตรียมดำเนินการในรอบถัดไป โดยระบบที่ได้จะมีเวอร์ชั่นที่มีความก้าวหน้าและสมบูรณ์มากขึ้นเรื่อยๆ จนได้ระบบที่สมบูรณ์ในที่สุด

2.5 Joint Application Development (JAD) คือวิธีการพัฒนาระบบร่วมกัน โดยนำบุคคลที่เกี่ยวข้องกับการพัฒนาระบบมาประชุมร่วมกัน เพื่อร่วมกันกำหนดความต้องการของระบบ ขอบเขตการทำงานของระบบ การวิเคราะห์และออกแบบระบบ บุคคลที่เกี่ยวข้องระบบประกอบด้วยเจ้าของระบบ ผู้ใช้งานระบบ นักวิเคราะห์และออกแบบระบบ โปรแกรมเมอร์ วิธีการนี้จะทำให้ช่วยลดเวลาและค่าใช้จ่ายในการดำเนินงานลง เนื่องจากนำบุคคลที่เกี่ยวข้องมาประชุมร่วมกันทำงาน ทำให้ได้ระบบที่พร้อมใช้งานในเวลาอันรวดเร็ว คุณภาพการทำงานของระบบตรงตามความต้องการของผู้ใช้ ผู้เข้าร่วมการดำเนินการวิธีการพัฒนาระบบร่วมกัน ประกอบด้วย

(1) JAD Session Leader เป็นผู้ดำเนินการประชุม ต้องผ่านการอบรมการทำงานเป็นกลุ่มและเป็นผู้ที่คอยอำนวยความสะดวกระหว่างการประชุม จัดตั้งระเบียบวาระการประชุมควบคุม

การประชุมให้อยู่ในวาระ เพื่อให้ได้ข้อมูลตรงจุด และเป็นผู้ชี้ขาดกรณีมีความขัดแย้งกันในระหว่างการประชุม

(2) Users คือ ผู้ใช้ระบบ เนื่องจากเป็นผู้ที่ใช้ระบบเป็นประจำทุกวัน ดังนั้นจะมีความเข้าใจถึงการทำงานและปัญหาที่เกิดขึ้นเป็นอย่างดี และเป็นบุคคลที่สามารถตอบคำถามเกี่ยวกับความสามารถของระบบที่กำลังจะพัฒนา

(3) Manager ผู้บริหารขององค์กร ซึ่งเป็นผู้ที่ใช้ระบบเช่นเดียวกับ User ผู้บริหารจะคอยเตรียมคำถามที่มุ่งไปที่ระบบที่ต้องการพัฒนาขึ้นมาใหม่ คอยจูงใจและคอยช่วยหาข้อสรุปในแต่ละวาระการประชุม

(4) Sponsor คือ ผู้ที่รับผิดชอบเรื่องค่าใช้จ่ายในการพัฒนาระบบนั้นๆ ซึ่งอาจจะเป็นผู้บริหารระดับสูงสุดขององค์กร

(5) System Analyst นักวิเคราะห์ระบบและทีมของนักวิเคราะห์ระบบ ทำหน้าที่เก็บข้อมูลจากการประชุมในแต่ละครั้ง

(6) Scribe คือ ผู้ที่ทำหน้าที่จดสรุปรายละเอียดระหว่างการประชุม โดยทั่วไปอาจใช้เครื่องคอมพิวเตอร์แบบพกพาช่วยในการบันทึก

(7) IS Staff ทีมของหน่วยบริการสารสนเทศองค์กรเช่น นักวิเคราะห์ระบบ โปรแกรมเมอร์ และผู้เชี่ยวชาญด้านฐานข้อมูล บุคคลเหล่านี้สามารถเสนอความคิดเห็นด้านเทคโนโลยีได้

2.6 Rapid Application Development (RAD) คือ วิธีการพัฒนาระบบแบบรวดเร็ว โดยใช้เครื่องมือสนับสนุน (CASE Tools) ช่วยในการพัฒนาระบบ ทำให้ได้ระบบที่สมบูรณ์ในเวลารวดเร็ว ทำให้ช่วยลดต้นทุนและเวลาในการพัฒนา วิธีการนี้เป็นการประยุกต์โมเดลการพัฒนาระบบแบบดั้งเดิมและวิธีการพัฒนแบบ JAD โดยรวมขั้นตอน การวิเคราะห์ การออกแบบ การสร้างและการทดสอบ ไว้ในการประชุมร่วมกันของผู้เกี่ยวข้อง เพื่อลดระยะเวลาในการพัฒนาระบบ ทีมงานที่ทำงานร่วมกันประกอบด้วย ทีมผู้เชี่ยวชาญด้านเทคโนโลยีสารสนเทศและกลุ่มผู้ใช้ วัตถุประสงค์ของ RAD คือต้องการรวมกระบวนการสำคัญต่างๆ เพื่อพัฒนาระบบในเวลาอันสั้น โดยใช้เครื่องมือ (CASE Tools) เช่น การใช้เครื่องมือสร้างแบบฟอร์มและรายงานแบบอัตโนมัติ รวมถึงการนำภาษายุคที่ 4 เพื่อสร้างต้นแบบระบบขึ้นมาได้อย่างรวดเร็ว ภายในระยะเวลาที่จำกัดมากกว่าที่จะให้ระบบมีความสมบูรณ์แบบ เทคนิคสำคัญของ RAD ประกอบด้วย

(1) พัฒนาต้นแบบได้อย่างรวดเร็ว

(2) เป็นแหล่งรวมเครื่องมือเพื่อการพัฒนาระบบ

(3) มีทีมงานที่เชี่ยวชาญการใช้เครื่องมือเหล่านั้น

(4) เป็นแนวร่วมปฏิบัติการกับ JAD

(5) มีกรอบระยะเวลาการพัฒนาที่จำกัด

2.7 Unified Process (UP) คือวิธีการพัฒนาระบบเชิงวัตถุ ที่ถูกพัฒนาขึ้นโดย Rational Software จุดประสงค์ของ Unified Process คือการพัฒนาซอฟต์แวร์ที่มีคุณภาพสูง ตรงตามความต้องการของผู้ใช้ ภายใต้งบประมาณและระยะเวลาที่กำหนดไว้ในโครงการ โดยพื้นฐานสำคัญของกระบวนการ Unified Process คือการสร้างโมเดลและจัดการโมเดลด้วยภาษา UML (Unified Modeling Language) นอกจากนี้ ในปัจจุบัน โมเดลนี้ยังได้รับการยอมรับอย่างกว้างขวาง ด้วยการกำหนดให้เป็นระเบียบวิธีมาตรฐานสำหรับการพัฒนาระบบเชิงวัตถุ ทั้งนี้ระเบียบวิธีของ Unified Process ถูกออกแบบมาเพื่อนำมาใช้กับโครงการผลิตซอฟต์แวร์ขนาดใหญ่ที่สะท้อนให้เห็นถึงวิวัฒนาการของกระบวนการผลิตซอฟต์แวร์สำหรับยุคปัจจุบันได้เป็นอย่างดี ขั้นตอนการพัฒนาระบบด้วย Unified Process ประกอบด้วย 4 ระยะดังนี้

(1) ระยะเริ่มต้น (Inception Phase) เป็นระยะเริ่มต้นของการดำเนินงาน ที่ผู้จัดการโครงการจะกำหนดขอบเขตของระบบ หน้าที่การทำงานหลักๆ ของโครงการที่ต้องทำสำเร็จ และวิสัยทัศน์สำหรับระบบใหม่ โดยการศึกษาถึงประโยชน์ที่ได้รับจากระบบใหม่ หากผลการศึกษาระบบพบว่า โครงการมีส่วนช่วยธุรกิจได้น้อยมาก โครงการพัฒนาซอฟต์แวร์นี้จะถูกยกเลิกโดยทันทีในระยะนี้

(2) ระยะเพิ่มเติมรายละเอียด (Elaboration Phase) การดำเนินงานในระยะนี้ ปกติจะต้องทำงานทวนซ้ำหลายรอบ ด้วยการทำความเข้าใจถึงปัญหาของระบบว่า ระบบจะทำงานได้อย่างไร การทำงานของระยะ Elaboration จะประกอบด้วย การวิเคราะห์ การออกแบบ และการสร้างสถาปัตยกรรมหลักของระบบ ซึ่งเกี่ยวข้องกับการรวบรวมแนวความคิดสำคัญต่างๆ ของผลิตภัณฑ์ โดยเมื่อถึงจุดสิ้นสุดของระยะนี้ ผู้จัดการโครงการจะสามารถประมาณต้นทุนโครงการและเวลาในการทำงานได้ชัดเจน หรือใกล้เคียงความจริงมากขึ้น แบบจำลองที่ใช้ประกอบด้วยไดอะแกรมต่างๆ คือ Use-Case Diagram, Class Diagrams, Sequence Diagrams และไดอะแกรมอื่นๆ ของ UML และการคาดการณ์ต้นทุน ผลกำไร และความเสี่ยง จะกระทำระยะนี้

(3) ระยะการสร้าง (Construction Phase) ระยะนี้จะทำงานทวนซ้ำหลายรอบเช่นกัน เกี่ยวข้องกับการออกแบบและการสร้างระบบ โดยส่วนประกอบสำคัญและคุณสมบัติต่างๆที่จำเป็นต้องมีในระบบทั้งหมด จะได้รับการพัฒนาและนำมาผนวกรวมเข้าด้วยกัน จากนั้น ระบบงานก็จะถูกนำมาทดสอบว่าทำงานถูกต้องหรือไม่ ตรงตามความต้องการของผู้ใช้หรือไม่ และผู้ใช้พึงพอใจหรือไม่ เพื่อพร้อมเข้าสู่การส่งมอบซอฟต์แวร์และการติดตั้งใช้งานจริงต่อไป

(4) ระยะการเปลี่ยนผ่าน (Transition Phase) เป็นระยะการส่งมอบระบบให้แก่ลูกค้า ซึ่งถือเป็นระยะสุดท้าย โดยจะดำเนินการเพียงรอบเดียวหรือหลายรอบก็ได้ ระบบจะถูกติดตั้งและพร้อมสำหรับการปฏิบัติงานจริง มีการฝึกอบรมผู้ใช้ จัดทำเอกสารระบบ คู่มือการใช้ระบบ

2.8 Agile Methodologies เป็นเทคนิคมุ่งตอบสนองความเปลี่ยนแปลงมากกว่าการปฏิบัติงานตามแผน รวมถึงไม่มุ่งเน้นการจัดทำเอกสารที่ไม่จำเป็น ด้วยการเน้นความเป็นเรียบง่าย ตรงไปตรงมา และต้องทำให้ตรงตามความประสงค์ การพัฒนาระบบตามแนวทางของ Agile ประกอบด้วย

(1) มุ่งตอบสนองการเปลี่ยนแปลงมากกว่าการดำเนินงานตามแผน หมายความว่า Agile ได้มองเห็นความจริงว่า แผนการต่างๆ ที่กำหนดขึ้นในโครงการ เมื่อนำมาปฏิบัติจริงแล้ว อาจไม่สามารถดำเนินงานตรงตามแผนได้ทุกครั้งไป

(2) มุ่งความสำคัญที่ตัวบุคคลและการปฏิสัมพันธ์มากกว่ากระบวนการและเครื่องมือ เทคนิค Agile จะเน้นการปฏิสัมพันธ์ด้วยการสื่อสารระหว่างทีมงานกับผู้ใช้ เพื่อให้ได้มาซึ่งสิ่งที่ต้องการจริงๆ มากกว่าการมุ่งเน้นที่ทฤษฎี กระบวนการ และเครื่องมือมากมาย

(3) เน้นผลผลิตของซอฟต์แวร์มากกว่าเอกสาร Agile จะเน้นชิ้นงานหรือผลผลิตซอฟต์แวร์ที่สามารถนำไปใช้งานได้จริง ซึ่งปกติ Agile จะส่งมอบชิ้นงานทางซอฟต์แวร์เป็นระยะๆ เพื่อให้ลูกค้าเห็นความคืบหน้าของชิ้นงานและเกิดความพึงพอใจ

(4) เน้นการทำงานร่วมกันกับลูกค้า มากกว่าการต่อรองเจราจาเรื่องสัญญา Agile จะมุ่งเน้นให้ลูกค้าเข้ามามีส่วนร่วมในการกำหนดความต้องการกับทีมงานอย่างต่อเนื่อง

3. บทสรุป

กระบวนการทางวิศวกรรมซอฟต์แวร์มุ่งเน้นการเพิ่มผลผลิตที่ดี กิจกรรมพื้นฐานของกระบวนการทางซอฟต์แวร์ประกอบด้วย 4 ส่วนหลักๆ คือ ข้อกำหนด การพัฒนาซอฟต์แวร์ การตรวจสอบความถูกต้องของซอฟต์แวร์ และวิวัฒนาการของซอฟต์แวร์

ระเบียบวิธีการพัฒนาซอฟต์แวร์ หรือเรียกว่า โมเดลการพัฒนาซอฟต์แวร์ คือแบบจำลองที่ใช้สำหรับเป็นตัวชี้นำถึงกิจกรรมหลัก การพัฒนาซอฟต์แวร์ปัจจุบันมีโมเดลการพัฒนาซอฟต์แวร์ให้ใช้หลายโมเดล โดยเฉพาะโมเดลสมัยใหม่ตามหลักวิศวกรรมซอฟต์แวร์

4. คำถามท้ายบท

1. อธิบายกิจกรรมพื้นฐานของกระบวนการทางวิศวกรรมซอฟต์แวร์
2. คุณสมบัติของซอฟต์แวร์ที่มีคุณภาพประกอบด้วยอะไรบ้าง
3. อธิบายความหมาย วิศวกรรมซอฟต์แวร์
4. อธิบายขั้นตอนของ Built-and-Fix Model ประกอบด้วยขั้นตอนอะไรบ้าง
5. อธิบายความแตกต่าง Waterfall Model รูปแบบเดิม กับ Waterfall Model แบบทวนซ้ำ
6. อธิบายขั้นตอนของ Incremental Model ประกอบด้วยขั้นตอนอะไรบ้าง
7. อธิบายส่วนของรอบการทำงานของ Spiral Model ประกอบด้วยอะไรบ้าง
8. รายละเอียดสำคัญของโครงการ RAD ประกอบด้วยอะไรบ้าง
9. ระเบียบวิธีของ Agile ประกอบด้วยอะไรบ้าง
10. อธิบายสาเหตุสำคัญที่จำเป็นต้องใช้โมเดลการพัฒนาซอฟต์แวร์

5. เอกสารอ้างอิง

กิตติ ภักดีวัฒนะกุลและพนิดา พานิชกุล.(2546).**คัมภีร์การวิเคราะห์และออกแบบระบบ**.กรุงเทพฯ : เคทีพี คอมพ์ แอนด์ คอนซัลท์.

ฝ่ายผลิตหนังสือตำราวิชาการคอมพิวเตอร์.(2551).**การวิเคราะห์และออกแบบระบบ**.กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

อำไพ พรประเสริฐสกุล.(2544).**การวิเคราะห์และออกแบบระบบ System Analysis and Design**. กรุงเทพฯ:ซีเอ็ดยูเคชั่น.

โอภาส เอี่ยมสิริวงศ์. (2555).**การวิเคราะห์และออกแบบระบบ(ฉบับปรับปรุงเพิ่มเติม)**.กรุงเทพฯ:ซีเอ็ดยูเคชั่น.

Gary B. Shelly and Harry J. Rosenblatt.(2012).**System Analysis and Design,Ninth Edition**.Course Technology.