บทที่ 3
ยูนิฟายด์โพรเซส

เกริ่นนำ

ที่ผ่านมาผู้เรียนดได้ศึกษาเกี่ยวกับความรู้พื้นฐานและแนวคิดต่าง ๆ เกี่ยวกับเชิงวัตถุไปแล้ว ในบทนี้ผู้ศึกษาจะได้เข้าใจเกี่ยวกับกระบวนการพัฒนาระบบเชิงวัตถุและความรู้เกี่ยวกับของยูนิฟายด์โพรเซส และแนะนำหลักการของยูนิฟายด์โพรเซส (Unified Process) เพื่อนำมาประยุกต์ใช้งานในการพัฒนาระบบเชิงวัตถุ เนื่องจากเป็นกระบวนการที่เน้นใช้หลักการเชิงวัตถุมาพัฒนาระบบซอฟต์แวร์ได้เป็นอย่างดี โดยมีการกำหนดกรอบแนวทางที่ชัดเชน ว่าใครทำอะไร มีความรับผิดชอบหน้าที่ใดในกระบวนการ นอกจากนี้ได้มีการนำเสนอแนวคิดและโมเดลที่ได้มาตรฐานสากลระดับโลก เพื่อให้การพัฒนาระบบเป็นไปอย่างมีประสิทธิภาพและตรงกับความต้องการของผู้ใช้มากที่สุด

ความหมายของกระบวนการพัฒนาระบบ

กระบวนการพัฒนาระบบ หมายถึง การสร้างระบบงานคอมพิวเตอร์เพื่อการดำเนินให้บรรลุเป้าหมายของโปรแกรม เครื่องมือ ซอฟแวร์ หรือแม้แต่ระบบงานขนาดใหญ่ก็ได้ กระบวนการพัฒนาระบบแบบดั้งเดิมหรือเชิงโครงสร้างจะมีความยุ่งยากและซับซ้อน การพัฒนาระบบเชิงวัตถุจึงถูกพัฒนาขึ้นเพื่อให้มีความยืดหยุ่นในการปรับเปลี่ยนตามความต้องการของผู้ใช้ได้อย่างสะดวกยิ่งขึ้น (กิตติ ภักดีวัฒนะกุล และพนิดา พานิชกุล, 2546) ความแตกต่างระหว่างกระบวนการพัฒนาระบบเชิงโครงสร้างและระบบเชิงวัตถุมีทั้งทางด้านการวิเคราะห์ การออกแบบ รูปแบบโมเดล และการดำเนินงานต่าง ๆ โดยความแตกต่างระหว่างกระบวนการพัฒนาระบบเชิงโครงสร้างและระบบเชิงวัตถุ คือ การวิเคราะห์และออกแบบระบบเชิงโครงสร้างจะอาศัยการวิเคราะห์ความต้องการและทำแผนภาพการไหลเวียนของข้อมูล(DFD) คำอธิบายกระบวนการ (Process Description) การจัดทำพจนานุกรมข้อมูล(Data dictionary) และการออกแบบแผนภาพอีอาร์(Entity Relationship Diagram) เพื่อจัดทำฐานข้อมูลต่อไป (กิตติพงษ์ กลมกล่อม, 2552) แต่สำหรับการวิเคราะห์และออกแบบระบบเชิงวัตถุ จะอาศัยการวิเคราะห์ความต้องการโดยใช้ยูสเคสต่าง ๆ แล้วระบุคลาสของวัตถุ ออกแบบแผนภาพคลาส และพัฒนาส่วนของระบบการจัดการฐานข้อมูลจากการแปลงแผนภาพคลาสต่าง ๆ หลังจากนั้นจัดทำแผนภาพความสัมพันธ์เพื่อการพัฒนาฐานข้อมูลเชิงวัตถุและพัฒนาโปรแกรมหรือระบบ (อำไพ พรประเสริฐสกุล, 2544)

แนวคิดของการพัฒนาระบบ

การพัฒนาระบบงานทางคอมพิวเตอร์เพื่อใช้ประโยชน์ในการจัดการข้อมูล ประมวลผล และดำเนินการต่าง ๆ ไม่ว่าจะเป็นทางธุรกิจเอกสารและภาครัฐ ผู้จัดการโครงการ (Project manager หรือ PM)

มีบทบาทสำคัญอย่างมากในการบริหารจัดการทีมงานและพัฒนาระบบ โดยทั่วไปมีอยู่ 2 แนวทางหลักในการพัฒนาระบบงาน ได้แก่ การพัฒนาระบบแบบดังเดิม (Traditional Development Process) หรือแบบโครงสร้าง (Structural Development Process) และการพัฒนาระบบเชิงวัตถุ (Object Oriented Development Process)

กระบวนการพัฒนาระบบเชิงโครงสร้าง

กระบวนการพัฒนาระบบเชิงโครงสร้าง (Structural Development Process) กระบวนการพัฒนาระบบแบบดังเดิม (Traditional) นั้น เริ่มต้นจากรูปแบบในการวิเคราะห์และออกแบบระบบเชิงโครงสร้าง (Structural Analysis and Design, **SAD**) ซึ่งเรานิยามการออกแบบและดำเนินงาน ตามลำดับดังนี้

1). กำหนดปัญหาและโครงการ (Problem and Project Identification)

2). วิเคราะห์และระบุความต้องการ (Requirement Analysis and Specification)

3). ออกแบบแผนภาพการไหลเวียนข้อมูล (Data Flow Diagram (DFD) Design)

4). ให้คำอธิบายกระบวนการ (Process description)

5). จัดทำพจนานุกรมข้อมูล (Data Dictionary)

6). ออกแบบแผนภาพความสัมพันธ์ (ER Design)

7). จัดทำฐานข้อมูล (Database Development)

8). ออกแบบหน้าจอ (Interface Design)

9). เขียนโปรแกรมและระบบ (Implementation)

10). ทดสอบและสอบทานระบบ (System Testing and Auditing)

11). ถ่ายโอนงาน (Data Transfer System)

12). จัดสร้างแผนภาพดีพอยท์เมน (Deployment Diagram Design)

ระบบงานและโปรแกรมที่มีขนาดใหญ่จะมีซับซ้อนมากขึ้น กระบวนการพัฒนาระบบงานแบบดั้งเดิมจะใช้เวลานานในการออกแบบและพัฒนาระบบค่อนข้างมาก และเพื่อให้ตอบสนองต่อความต้องการของผู้ใช้ได้อย่างง่ายและประหยัดทั้งทางด้านเวลาและค่าใช้จ่ายในการพัฒนาระบบ แนวคิดกระบวนการพัฒนาระบบจึงเปลี่ยนไปเป็นตามหลักการแบ่งแยกส่วนต่าง ๆ (Decomposition) ให้เป็นวัตถุ (Object) ที่ประกอบด้วยข้อมูลหรือคุณลักษณะ (Data or Attribute) กับ การปฏิบัติหรือพฤติกรรม (Action or Behavior) โดยเป็นแนวทางการวิเคราะห์และสังเคราะห์วิธีการแก้ปัญหาที่มุ่งเน้นที่ข้อมูล (Data) และการปฏิบัติ (Actions) โดยรวมเป็นหน่วยเดียวกันคือ วัตถุ โดยพิจารณาขอบเขตของปัญหา (Problem Domain) ในโลกแห่งความจริง (Real World) ดังนั้นการพัฒนากระบวนพัฒนาระบบแบบใหม่ที่อาศัยแนวคิดเชิงวัตถุจะทำให้การดำเนินงานมีความสอดคล้องกับสภาพจริงและมีความยืดหยุ่นมากยิ่งขึ้น

กระบวนการพัฒนาระบบเชิงวัตถุ

กระบวนการพัฒนาระบบเชิงวัตถุ (Object Oriented Development Process) จะเริ่มจากรูปแบบการวิเคราะห์และออกแบบระบบเชิงวัตถุ (Object Oriented Analysis and Design, **OOAD**) จะดำเนินการเป็นกระบวนการ ดังนี้

1). กำหนดปัญหาและโครงการ (Problem and Project Identification)

2). วิเคราะห์และระบุความต้องการ (Requirement Analysis and Specification) โดยใช้ยูสเคส

3). ระบุคลาสและองค์ประกอบของคลาส (Class Identification)

4). ออกแบบแผนภาพคลาส (Class Diagram Design)

5). แปลงแผนภาพคลาสให้เป็นระบบการจัดการฐานข้อมูลแบบความสัมพันธ์ (RDBMS)

6). จัดทำแผนภาพความสัมพันธ์ต่างๆ (Sequence Diagram, Activity Diagram, etc.)

7). จัดทำฐานข้อมูล (Database Development)

8). ออกแบบหน้าจอ (Interface Design)

9). เขียนโปรแกรมและระบบ (Implementation)

10). ทดสอบและสอบทานระบบ (System Testing and Auditing)

11). ถ่ายโอนงาน (Data Transfer System)

12). จัดสร้างแผนภาพดีพอยท์เมน (Deployment Diagram Design)

จะเห็นว่าขั้นตอนทั้ง 12 ขั้นตอนมีลักษณะคล้ายคลึงกับขั้นตอนการพัฒนาระบบแบบดั้งเดิมอยู่บ้าง เพื่อให้เห็นขั้นตอนที่แตกต่างกันอย่างชัดเจน เราสามารถเปรียบเทียบแต่ละขั้นตอนได้ดังแสดงในตารางที่ 4.1 ตารางเปรียบเทียบขั้นตอนการวิเคราะห์และออกแบบระบบเชิงโครงสร้าง (Structural Analysis and Design (**SAD**)) กับ ระบบเชิงวัตถุ (Object Oriented Analysis and Design (**OOAD**)) พบว่าส่วนหลักที่มีความแตกต่างกัน ได้แก่ ขั้นตอนที่เกี่ยวข้องกับการวิเคราะห์และออกแบบระบบ โดยการวิเคราะห์และออกแบบระบบเชิงโครงสร้างจะอาศัยการวิเคราะห์ความต้องการและทำแผนภาพการไหลเวียนของข้อมูล คำอธิบายกระบวนการ การจัดทำพจนานุกรมข้อมูลและการออกแบบแผนภาพ ER เพื่อจัดทำฐานข้อมูลต่อไป แต่สำหรับการวิเคราะห์และออกแบบระบบเชิงวัตถุ จะอาศัยการวิเคราะห์ความต้องการโดยใช้ยูสเคสต่าง ๆ แล้วระบุคลาสของวัตถุ ออกแบบแผนภาพคลาส และพัฒนาส่วนของระบบการจัดการฐานข้อมูลแบบรีเลชันเนล Relational Database Management System (RDBMS) จากการแปลงแผนภาพคลาสหลังจากนั้นจัดทำแผนภาพความสัมพันธ์แบบต่าง ๆ เพื่อการพัฒนาฐานข้อมูลเชิงวัตถุและพัฒนาโปรแกรมหรือซอฟแวร์ต่อไป ขั้นตอนอื่น ๆ ในทางปฏิบัติย่อมมีแตกต่างทางด้านการใช้เครื่องมือหรือรูปแบบในการพัฒนาให้สอดคล้องกับการวิเคราะห์โดยเฉพาะการโปรแกรมมิ่งเชิงวัตถุ อย่างไรก็ตามขั้นตอนเหล่านี้ให้ความหมายในการจัดทำในลักษณะเดียวกันของทั้งสองวิธีการวิเคราะห์และออกแบบระบบหรือซอฟต์แวร์ (ฝ่ายผลิตหนังสือตำราวิชาการคอมพิวเตอร์, 2551)

ตารางที่ 3.1 การเปรียบเทียบลำดับขั้นตอนการวิเคราะห์และออกแบบระบบเชิงโครงสร้าง (SAD) และเชิงวัตถุ (OOAD)

SAD	OOAD
1. Problem and Project Identification	1. Problem and Project Identification
2. Requirement Analysis and Specification	2. Requirement Analysis and Specification (Use case)
3. Data Flow Diagram (DFD) Design	3. Class Identification
4. Process description	4. Class Diagram Design
5. Data Dictionary	5. Development of RDBMS
6. ER Design	6. Sequence Diagram, Activity Diagram, etc.
7. Database Development	7. Database Development
8. Interface Design	8. Interface Design
9. Implementation	9. Implementation
10. System Testing and Auditing	10. System Testing and Auditing
11. Data Transfer System	11. Data Transfer System
12. Deployment Diagram Design	12. Deployment Diagram Design

วัฏจักรการพัฒนาระบบงาน

วัฏจักรการพัฒนาระบบงาน (Systems Development Life Cycle (**SDLC**)) มีขั้นตอนทั้งหมด 7 หลัก ได้แก่ การวางแผน (Planning) การวิเคราะห์ (Analysis) การออกแบบเชิงตรรกะ (Design) การพัฒนาโปรแกรม (Implementation) การทดสอบระบบ (Testing) การดำเนินงานและการดีพอยเมน (Operation and deployment)และการบำรุงรักษา (Maintenance)

จะเห็นว่าจากขั้นตอนทั้งเจ็ดอย่างนี้ เป็นเสมือนการสรุปรวบยอดจากขบวนการพัฒนาระบบทั้งหมด 12 ขั้นตอนที่กล่าวในข้างต้น ซึ่งรูปแบบในการดำเนินงานนั้น ๆ จะสอดคล้องกันไปไม่ว่าจะเป็นการพัฒนาระบบแบบใดก็ตาม นั่นคือ

1) การวางแผน (Planning) ครอบคลุมกระบวนการการกำหนดปัญหาและโครงการ
2) การวิเคราะห์ (Analysis) ครอบคลุมกระบวนการการวิเคราะห์และระบุความต้องการ
3) การออกแบบเชิงตรรกะ (Design) ถ้าเป็นระบบเชิงโครงสร้างจะครอบคลุมขั้นตอนการออกแบบแผนภาพไหลเวียนข้อมูล การให้คำอธิบายกระบวนการ การจัดทำพจนานุกรมข้อมูล และการ

ออกแบบแผนภาพความสัมพันธ์ ส่วนระบบเชิงวัตถุจะครอบคลุมกระบวนการการระบุคลาส การออกแบบแผนภาพคลาส การแปลงแผนภาพคลาสและการจัดทำแผนภาพความสัมพันธ์

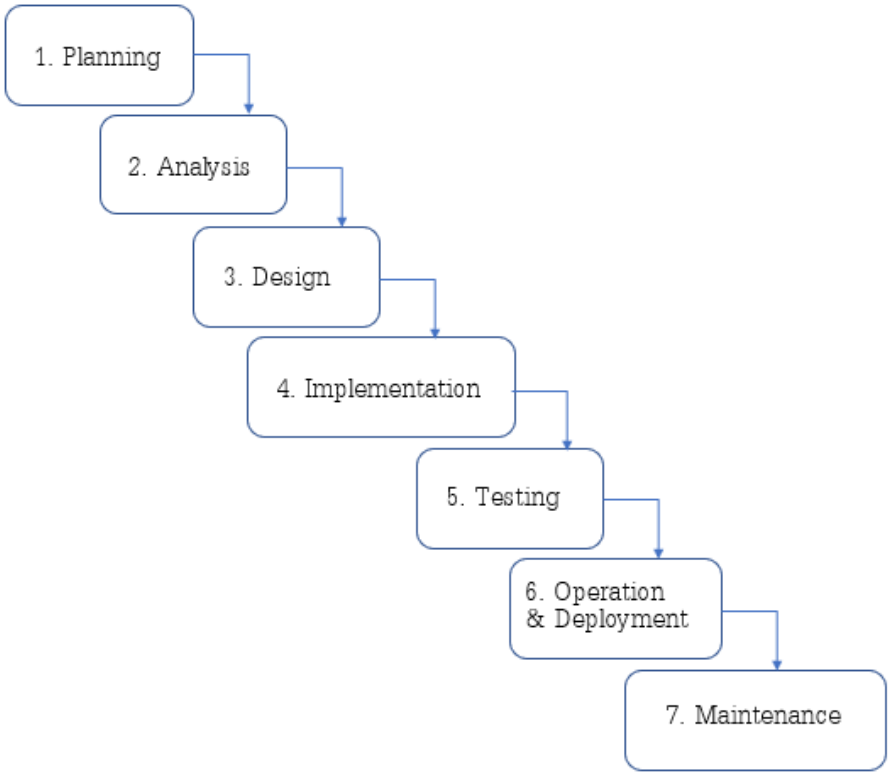
4) การพัฒนาโปรแกรม (Implementation) ครอบคลุมกระบวนการการจัดทำฐานข้อมูล การออกแบบหน้าจอ การเขียนโปรแกรมและระบบ
5) การทดสอบระบบ (Testing) ครอบคลุมกระบวนการการทดสอบและสอบทานระบบ
6) การดำเนินงานและการดีพอยเมน (Operation and deployment) ครอบคลุมกระบวนการการถ่ายโอนงานและการจัดสร้างแผนภาพดีพอยท์เมน
7) การบำรุงรักษา (Maintenance) เป็นขั้นตอนหลังจากการพัฒนาระบบสมบูรณ์แล้วเพื่อรักษาประสิทธิภาพการทำงานของระบบและปรับปรุงแก้ไขระบบต่อไป

ในการพัฒนาระบบนั้น จากวัฏจักรของการพัฒนาระบบนี้ ยังมีรูปแบบการดำเนินงานในแบบต่าง ๆ ที่แตกต่างกันออกไปอีก รูปแบบต่าง ๆ ของการดำเนินงานเหล่านี้ เรียกอีกอย่างหนึ่งว่า โมเดลการพัฒนาระบบ โดยในที่นี้จะอธิบายถึง 2 โมเดลที่สำคัญ คือ โมเดลวอเตอร์โฟลล์และโมเดลวนวนรอบเพิ่มพูน

โมเดลวอเตอร์โฟลล์

โมเดลวอเตอร์โฟลล์ (Waterfall model) จะมีลักษณะการดำเนินแต่ละขั้นตอนของวัฏจักรการพัฒนาระบบงาน ที่เป็นการกระทำที่ต่อเนื่องกันไปเหมือนน้ำตกที่ไหลลงจากที่สูงสู่ที่ต่ำ และจะเว้นพักน้ำในบางขั้นของชั้นหินของน้ำตก และสุดท้ายก็จะไหลตกลงมายังพื้นล่าง ดังแสดงให้เห็นในภาพที่ 4.2 ลักษณะของโมเดลวอเตอร์โฟลล์

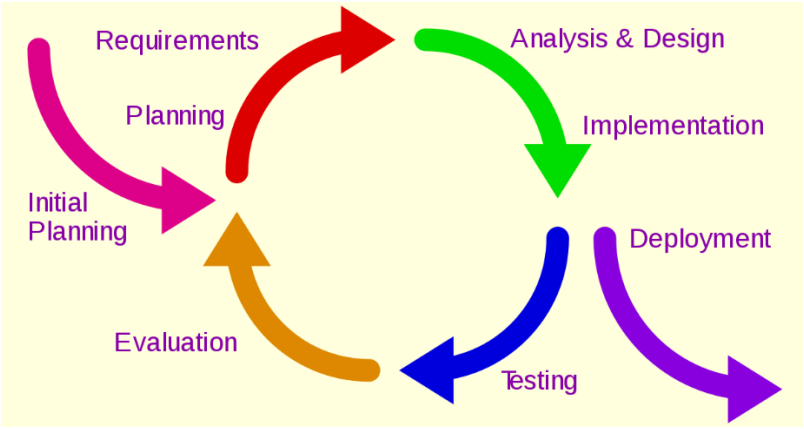
การพัฒนาระบบดั้งเดิมนั้น (หรือแบบเชิงโครงสร้างนั่นเอง) จะใช้วัฏจักรของการพัฒนาระบบ (SDLC) ในรูปแบบของวอเตอร์โฟลล์ กล่าวคือ เริ่มจากการดำเนินการวางแผนงานให้เรียบร้อยรัดกุม แล้วจึงค่อยดำเนินการวิเคราะห์และออกแบบระบบให้สมบูรณ์ แล้วจึงดำเนินการพัฒนาระบบ ทดสอบระบบ และดำเนินการให้ระบบใช้งานได้ (Shelly, G. and Rosenblatt, H., 2012) เมื่อระบบทำงานได้ดีแล้ว เป็นอันเข้าสู่กระบวนการบำรุงรักษาระบบ จะเป็นการพัฒนาระบบแบบดั้งเดิมนี้ ถ้าต้องการแก้ไขระบบให้สอดคล้องกับความต้องการของผู้ใช้ใหม่ ไม่สามารถดำเนินการได้โดยง่าย เพราะจะต้องย้อนกลับไปเริ่มตั้งแต่แรก และระบบแบบโครงสร้างไม่อำนวนความสะดวกให้กับการแก้ไขในบางส่วนของระบบได้อย่างง่าย เพราะทุกส่วนงานมักจะเกี่ยวโยงกันและส่งผลกระทบถึงกันได้ถ้ามีการเปลี่ยนแปลง



ภาพที่ 3.1 Waterfall model ของ SDLC

โมเดลวนรอบเพิ่มพูน

โมเดลรอบเพิ่มพูน (Iterative and Incremental model) จะมีลักษณะการดำเนินแต่ละขั้นตอนของวัฏจักรการพัฒนาระบบงานเป็นการกระทำที่ต่อเนื่องสำหรับบางส่วนงานที่กระทำได้ก่อน มีข้อมูลก่อน มีความชัดเจนในการดำเนินงานได้ แล้ววนกลับมาพิจารณาหรือเพิ่มเติมส่วนที่ต้องการเพิ่มเติมใหม่ได้ สามารถปรับเปลี่ยนและตรวจสอบระบบได้งาน และทำงานเป็นรอบ ๆ งาน ดังแสดงในภาพที่ 4.3



ภาพที่ 3.2 โมเดลวนรอบเพิ่มพูน (Iterative and Incremental Model)

(**ที่มา:** https://en.wikipedia.org/wiki/Iterative_and_incremental_development)

โดยปกติโครงการพัฒนาระบบจะถูกแบ่งออกเป็นหลายโครงการย่อย แต่ละโครงการย่อยจะมีกำหนดเวลา แผนงานของตัวเอง ซึ่งอาจเรียกแต่ละโครงการย่อยว่า รอบงาน (Iteration) และแต่ละรอบงานกระทำการพัฒนาระบบตามขั้นตอนวิเคราะห์ ออกแบบ และพัฒนาระบบของตนเองจนได้ระบบที่นำไปประมวลผลได้ ระบบจะถูกเพิ่มพูนให้โตขึ้น (Incremental) ตลอดเวลาตามรอบงานแต่ละรอบงานที่ทำเสร็จจนกระทั่งได้ระบบที่สมบูรณ์นั่นเอง

การพัฒนาระบบแบบวนวนรอบเพิ่มพูน มีประโยชน์เพื่อจัดการกับความเสี่ยงที่มีผลกระทบสูงตั้งแต่ช่วงต้น เพื่อสร้างสถาปัตยกรรมระบบไว้ก่อน เป็นแนวทางในการพัฒนาระบบและเป็นกรอบงานที่สามารถจัดการกับการเปลี่ยนความต้องการที่ไม่สามารถเลี่ยงได้เป็นอย่างดี เป็นการสร้างระบบโดยการเพิ่มพูนผลตลอดเวลาในแต่ละรอบงาน แทนที่จะทำระบบในครั้งเดียวให้เสร็จสมบูรณ์ตอนใกล้จบโครงการ ซึ่งถ้ามีการเปลี่ยนแปลงเกิดขึ้นจะมีค่าใช้จ่ายมาก และเป็นกรรมวิธีที่ทำให้ทีมงานทำงานได้อย่างมีประสิทธิภาพ

กระบวนการพัฒนาระบบเชิงวัตถุมีรูปแบบที่สอดคล้องและสามารถดำเนินการแบบวนรอบเพิ่มพูนได้เป็นอย่างดี เนื่องจากการปรับแต่งระบบจะกระทำได้โดยง่าย ยกตัวอย่างเช่นจากการเพิ่มคลาสของวัตถุใหม่เข้าไปในระบบและเชื่อมโยงปฏิบัติสัมพันธ์และกิจกรรมระหว่างกัน ก็จะสามารถปรับปรุงระบบได้ง่ายขึ้น และสอดคล้องกับความต้องการเปลี่ยนแปลงไปของผู้ใช้ได้อย่างง่าย

ความหมายของยูนิฟายด์โพรเซส

ความหมายของยูนิฟายด์โพรเซส (Unified Process) คือวิธีการพัฒนาระบบเชิงวัตถุ ที่ถูกพัฒนาขึ้นโดย Rational Software จุดประสงค์ของยูนิฟายด์โพรเซสคือการพัฒนาซอฟต์แวร์ที่มีคุณภาพสูง ตรงตามความ ต้องการของผู้ใช้ ภายใต้งบประมาณและระยะเวลาที่กำหนดไว้ในโครงการ โดยพื้นฐานสำคัญของกระบวนการยูนิฟายด์โพรเซส คือการสร้างโมเดลและจัดการโมเดลด้วยภาษา UML(Unified Modeling Language) (Clariso, R., Gonzalez, C. and Cabot, J., 2017) นอกจากนี้ ในปัจจุบัน โมเดลนี้ยังได้รับการยอมรับอย่างกว้างขวาง ด้วยการกำหนดให้เป็นระเบียบวิธีมาตรฐานสำหรับการพัฒนาระบบเชิงวัตถุ ทั้งนี้ระเบียบวิธีของยูนิฟายด์โพรเซสถูกออกแบบมาเพื่อนำมาใช้กับโครงการผลิตซอฟต์แวร์ขนาดใหญ่ที่สะท้อนให้เห็นถึงวิวัฒนาการของกระบวนการผลิตซอฟต์แวร์สำหรับยุคปัจจุบันได้เป็นอย่างดี โดยการพัฒนาระบบด้วยยูนิฟายด์โพรเซสจะแบ่งออกเป็น 4 ระยะดังนี้

(1) ระยะเริ่มต้น (Inception Phase) เป็นระยะเริ่มต้นของการดำเนินงาน ที่ผู้จัดการ โครงการจะกำหนดขอบเขตของระบบ หน้าที่การทำงานหลัก ๆ ของโครงการที่ต้องทำสำเร็จ และวิสัยทัศน์สำหรับระบบใหม่ โดยการศึกษาถึงประโยชน์ที่ได้รับจากระบบใหม่ หากผลการศึกษาระบบพบว่า โครงการมีส่วนช่วยธุรกิจได้น้อยมาก โครงการพัฒนาซอฟต์แวร์นี้จะถูกยกเลิกโดยทันทีในระยะนี้

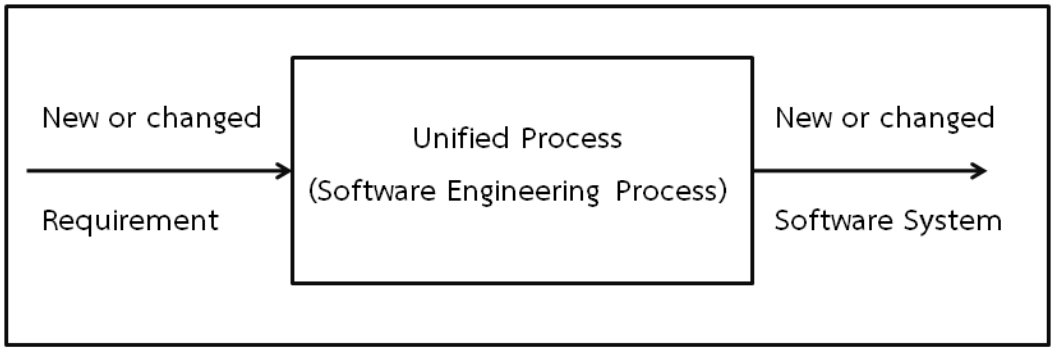
(2) ระยะเพิ่มเติมรายละเอียด (Elaboration Phase) การดำเนินงานในระยะนี้ ปกติ จะต้องทำงานทวนซ้ำหลายรอบ ด้วยการทำความเข้าใจถึงปัญหาของระบบว่า ระบบจะทำงานได้ อย่างไร การทำงานของระยะ Elaboration จะประกอบด้วย การวิเคราะห์ การออกแบบ และการสร้างสถาปัตยกรรมหลักของระบบ

ซึ่งเกี่ยวข้องกับการรวบรวมแนวความคิดสำคัญต่าง ๆ ของผลิตภัณฑ์ โดยเมื่อถึงจุดสิ้นสุดของระยะนี้ ผู้จัดการโครงการจะสามารถประมาณต้นทุนโครงการและเวลาในการทำงานได้ชัดเจน หรือใกล้เคียงความจริงมากขึ้น แบบจำลองที่ใช้ประกอบด้วย แผนภาพต่าง ๆ คือ Use - Case Diagram, Class Diagrams, Sequence Diagrams และไดอะแกรม อื่น ๆ ของ UML และ การคาดการณ์ต้นทุน ผลกำไร และความเสี่ยง จะกระทำระยะนี้

(3) ระยะการสร้าง (Construction Phase) ระยะนี้จะทำงานทวนซ้ำหลายรอบเช่นกัน เกี่ยวข้องกับการออกแบบและการสร้างระบบ โดยส่วนประกอบสำคัญและคุณสมบัติต่าง ๆ ที่จำเป็นต้องมีในระบบทั้งหมดจะได้รับการพัฒนา และนำมาผนวกรวมเข้าด้วยกัน จากนั้น ระบบงานก็จะถูกนำมาทดสอบว่าทำงานถูกต้องหรือไม่ ตรงตามความต้องการของผู้ใช้หรือไม่ และผู้ใช้พึงพอใจหรือไม่ เพื่อพร้อมเข้าสู่การส่งมอบซอฟต์แวร์และการติดตั้งใช้งานจริงต่อไป

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

เนื่องจากยูนิฟายด์โพรเซส นั้นเป็นกรรมวิธีในการพัฒนาซอฟต์แวร์ (Software Development Process) โดยจะอธิบายแนวทางสำหรับการวิเคราะห์ การออกแบบ การจัดสร้างการติดตั้ง และอาจรวมถึงการบำรุงรักษาซอฟต์แวร์ ซึ่งเป็นกรรมวิธีการพัฒนาซอฟต์แวร์ที่ได้รับความนิยมสำหรับระบบซอฟต์แวร์เชิงวัตถุ โดยยูนิฟายด์โพรเซสจะอธิบายการจัดสรรงานและความรับผิดชอบให้กับทีมงานไว้อย่างชัดเจนว่าใคร จะทำอะไร เมื่อไร และทำอย่างไร เพื่อให้เกิดความมั่นใจว่าจะได้ซอฟต์แวร์ที่ดีมีคุณภาพ ตรงกับความต้องการของลูกค้า และการพัฒนาซอฟต์แวร์อยู่ภายใต้งบประมาณและค่าใช้จ่ายที่ได้ประมาณการไว้ดังแสดงในภาพที่ 3.1 **(ที่มา : ออนไลน์** http://202.28.94.55/members/473020449-5/up1.htm**)**



ภาพที่ 3.3 แสดงแนวคิดการทำงานของยูนิฟายด์โพรเซส

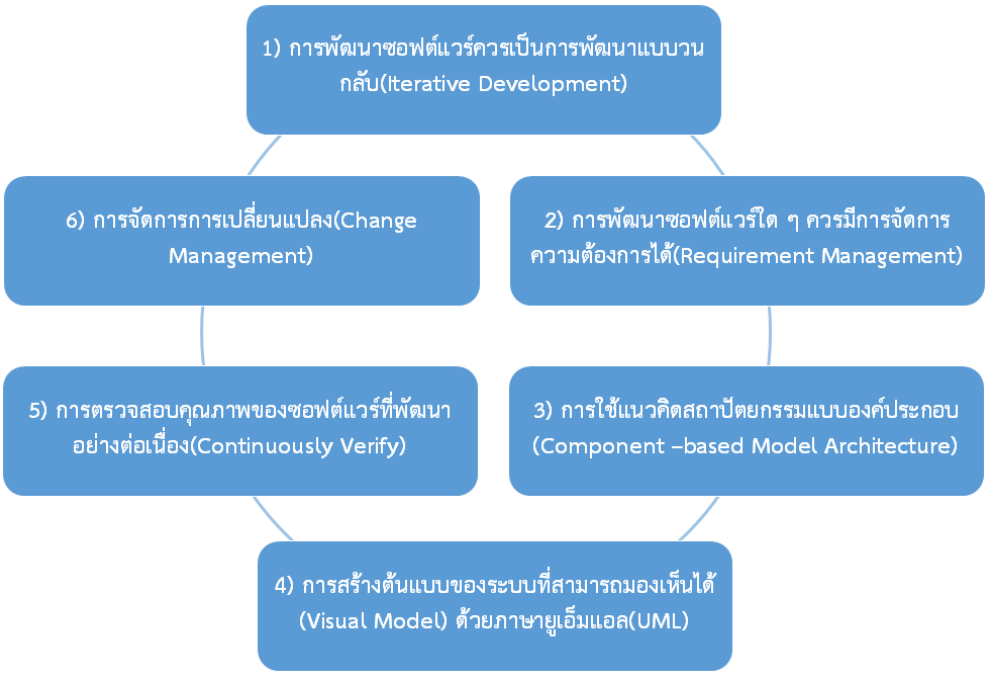
แนวคิดเรื่องยูนิฟายด์โพรเซสโดยความหมายแล้วคือกระบวนการทางวิศวกรรมซอฟต์แวร์ที่เกิดจากการรวมเอาสิ่งที่ผู้เชี่ยวชาญทางการพัฒนาซอฟต์แวร์ (Software Engineer) ที่เคยกำหนดไว้และใช้แล้วได้ผลดีในการพัฒนาซอฟต์แวร์นั้นมารวมกันโดยเลือกแต่เทคนิคที่ดีและขั้นตอนหลักที่เหมือน ๆ กันและใช้ได้อย่างมีประสิทธิภาพของแต่ละผู้เชี่ยวชาญดังกล่าวมารวมกันและกำหนดให้มีชื่อใหม่ว่า "กระบวนการพัฒนา

ซอฟต์แวร์แบบรวมเป็นหนึ่งเดียว"บางครั้งจะพบว่ามีกระบวนการคล้ายกันนี้ในแวดวงทางวิศวกรรมซอฟต์แวร์ เช่น Rational Unified Process ซึ่งเป็นของบริษัทยักษ์ใหญ่ในวงการพัฒนาซอฟต์แวร์ที่ชื่อว่า “Rational Rose Corporation” และถูกจดลิขสิทธิ์ด้วยหลักการหรือแนวคิดจะคล้ายกันแต่จะแตกต่างที่รายละเอียดของกระบวนการมากกว่าสำหรับแนวคิดที่มีลักษณะร่วมกันหรือเหมือนกันของยูนิฟายด์โพรเซส ได้แก่ การพัฒนาแบบวนกลับ (Iterative Development) การจัดการกับความต้องการ (Requirement Management) และการใช้เครื่องมือช่วยทางวิศวกรรมซอฟต์แวร์ (CASE Tools) เป็นต้น (โอภาส เอี่ยมสิริวงศ์, 2555)

วัตถุประสงค์ของยูนิฟายด์โพรเซส

วัตถุประสงค์ของยูนิฟายด์โพรเซสคือ การทำให้ได้ซอฟต์แวร์ที่มีคุณภาพและสอดคล้องกับความต้องการ (Requirements) ของผู้ใช้โดยอยู่ภายใต้งบประมาณและเวลาที่สามารถคาดเดาได้ (Predictable Budget and Time) ยูนิฟายด์โพรเซสนั้นจะเน้นการกำหนดบทบาทไปที่ทีมพัฒนางานมากกว่าแต่ละบุคคล กล่าวคือจะมีการกำหนดว่าในแต่ละช่วง (Phase) ของการพัฒนานั้นว่าควรประกอบไปด้วยใคร(Who) แต่ละคนมีหน้าที่รับผิดชอบอะไร (What) จะทำงานที่รับผิดชอบนั้นเมื่อไหร่ (When) และปฏิบัติอย่างไร (How) ที่กล่าวมาเป็นลักษณะที่เป็นนามธรรม (Abstract) หรืออาจกล่าวได้ว่ามันเป็นภาพมุมสูง (Bird eye view) ของกระบวนการยูนิฟายด์โพรเซสซึ่งอาจจะทำให้เข้าใจภาพยังไม่ชัดเจนกลยุทธ์ที่ใช้ในยูนิฟายด์โพรเซสรวมแล้วจะเรียกว่า “Best Practice Model” หรือ “Best Practice” ก็ได้ นั้นคือโดยลักษณะของยูนิฟายด์โพรเซสจะมีการปฏิบัติ 6 อย่าง ดังนี้

1) การพัฒนาซอฟต์แวร์ควรเป็นการพัฒนาแบบวนกลับ (Iterative Development)
2) การพัฒนาซอฟต์แวร์ใด ๆ ควรมีการจัดการความต้องการได้ (Requirement Management)
3) การใช้แนวคิดสถาปัตยกรรมแบบองค์ประกอบ (Component–based Model Architecture)
4) การสร้างต้นแบบของระบบที่สามารถมองเห็นได้ (Visual Model) โดยใช้ยูเอ็มแอล (UML)
5) การตรวจสอบคุณภาพของซอฟต์แวร์ที่พัฒนาอย่างต่อเนื่อง (Continuously Verify)
6) การจัดการการเปลี่ยนแปลง (Change Management)



ภาพที่ 3.4 แสดงลักษณะของยูนิฟายด์โพรเซส

การพัฒนาซอฟต์แวร์ควรเป็นการพัฒนาแบบวนกลับ

1. การพัฒนาซอฟต์แวร์ควรเป็นการพัฒนาแบบวนกลับ (Iterative Development) กล่าวคือในแต่ละรอบการทำงาน (Iteration) หนึ่ง ๆ จะประกอบด้วยกิจกรรมเหล่านี้ คือการกำหนดปัญหาการวิเคราะห์ปัญหาการออกแบบการสร้างและสุดท้ายการทดสอบความต้องการของผู้ใช้นั้นผลที่ได้คือซอฟต์แวร์ที่สามารถทำงานได้ (Executable Product) ซึ่งแตกต่างจากกระบวนการพัฒนาแบบดั้งเดิม (Traditional Development) ที่กว่าจะได้ผลลัพธ์ในรูปของซอฟต์แวร์ที่สามารถทำงานได้ นั้นจะต้องรอไปจนกว่าจะมีการทดสอบระบบทั้งหมดเรียบร้อยซึ่งสิ่งนี้ส่งผลให้โครงการพัฒนาซอฟต์แวร์นั้นมีความเสี่ยงสูงในการล้มเหลวได้เมื่อเวลาผ่านไปดั่งสำนวนที่ว่า “กว่าจะรู้ตัวก็สายเสียแล้ว” นั้นเอง การพัฒนาแบบวนย้อนกลับจะสังเกตง่าย ๆ จากลักษณะดังต่อไปนี้

1) โครงสร้างถูกแบ่งออกเป็นรอบ (Iteration)

2) สิ่งที่ได้จากการพัฒนาระบบในแต่ละรอบจะมีการนำไปพัฒนาเพิ่มสำหรับรอบต่อ ๆ ไป จนกว่าจะกลายเป็นระบบที่สมบูรณ์

3) ในแต่ละรอบทีมจะต้องทำงานซ้ำ (Iterate) ขั้นตอนการวิเคราะห์ ออกแบบ พัฒนาโปรแกรมและทดสอบโปรแกรมจากลักษณะของ Interactive and Incremental Development คือลักษณะของการวนรอบทำซ้ำทำเพิ่มขึ้น โดยในแต่ระรอบจะเริ่มด้วยการวางแผนเก็บรวบรวมความต้องการ วิเคราะห์ พัฒนาโปรแกรม ทดลองโปรแกรมแล้วนำไปใช้งานพร้อมกับเก็บข้อมูลการประเมินผลเพื่อวางแผนดำเนินการในรอบต่อไปและแต่ระรอบจะต้องมีการเพิ่มส่วนอื่น ๆ ของระบบจนกว่าจะครบดังนั้นก่อนดำเนินโครงการพัฒนา

ระบบสิ่งสำคัญคือจะต้องวางแผนว่าทั้งโครงการจะต้องแบ่งออกเป็นกี่รอบในแต่ละรอบจะเพิ่มเติมส่วนใดของระบบ จึงจะทำให้ระบบที่ได้มีความสมบูรณ์และสามารถรองรับความต้องการของผู้ใช้ระบบที่เปลี่ยนแปลงอยู่เสมอได้

2. การพัฒนาซอฟต์แวร์ใด ๆ ควรมีการจัดการความต้องการได้ (Requirement Management)

มีคำถามหนึ่งที่มักจะถามกันบ่อยคือ “ทำไมต้องจัดการความต้องการในเมื่อดูหรือศึกษาจากเอกสารรายงานต่าง ๆ ของระบบงานเดิมก็จบ!” ความคิดนี้เป็นความคิดแบบเก่าแบบเดิมที่มองว่าระบบที่พัฒนานั้นเป็นระบบเอกเทศ (Standalone) ซึ่งสามารถเกิดได้บ่อยในการพัฒนาซอฟต์แวร์บ้านโดยเฉพาะกระทรวงสาธารณสุขที่มักมองว่าแค่ดูหรือศึกษาจากรายงาน (Output) และกระบวนการทำงาน (Workflow) แล้วมาเขียนโปรแกรมก็จบ ไม่เห็นจะยากปัญหาที่ประสบอยู่ก็คือ มันเกิดจากการคิดแบบนี้นั้นแหละซึ่งคงพอจะสังเกตได้ว่าทำไมซอฟต์แวร์ของกระทรวงสาธารณสุขมันเยอะเหลือเกินและแต่ละตัวก็ไม่สามารถทำงานร่วมกันได้เลยข้อมูลใช้ร่วมกันแทบจะไม่ได้ ถ้าจะใช้ก็ต้องมาเล่นแร่แปรธาตุข้อมูลอีกทำให้เสียเวลา เสียแรงงานตลอดจนซอฟต์แวร์นั้นดูแลรักษายาก ตามลำดับสิ่งที่เป็นปัญหานี้ก็มาจากเหตุ คือ การที่ไม่ได้มองสิ่งที่พัฒนาแบบเป็นองค์รวมขาดการศึกษาระบบที่แวดล้อมหรือเกี่ยวข้องด้วย นั่นเองดังนั้นต้องมีการจัดการความต้องการที่ดี เพราะ “ความต้องการ” ถือว่าเป็นสิ่งที่สำคัญที่สุดหากไม่สามารถจัดการกับความต้องการที่เปลี่ยนแปลงได้ย่อมส่งผลกระทบต่อการพัฒนาระบบเป็นอย่างมาก และเสี่ยงต่อความล้มเหลว จะตามมาสำหรับการจัดการความต้องการนั้นจะเน้นไปที่ทำอย่างไรจะจัดการความต้องการได้อย่างเหมาะสมทั้งนี้ระบบสามารถเปลี่ยนไปตามความต้องการได้ โดยใช้ทรัพยากรในการพัฒนาน้อยที่สุดสรุปวัตถุประสงค์ของการจัดการความต้องการก็คือ เพื่อที่จะทำให้มั่นใจได้ว่าแก้ปัญหาได้ถูกต้อง เหมาะสม และสร้างระบบที่สอดคล้องกับความต้องการของผู้ใช้ ทั้งนี้การจัดการความต้องการจะต้องเป็นแนวทางที่เป็นระบบ (Systematic Approach) ซึ่งมันก็จะมีเทคนิคและวิธีการของมัน

3. การใช้แนวคิดสถาปัตยกรรมแบบองค์ประกอบ (Component–Based Model Architecture)

ได้มีผู้เชี่ยวชาญด้านการพัฒนาซอฟต์แวร์ กล่าวไว้ว่าในการพัฒนาซอฟต์แวร์หรือระบบใด ๆ ก็แล้วแต่จะต้องมีการกำหนดหรือออกแบบสถาปัตยกรรมของระบบก่อนเสมอ กล่าวคือหากจะเปรียบเทียบให้เข้าใจชัดเจนขึ้น ก็จะขอเปรียบเทียบกับการสร้างบ้านที่จะต้องมีการกำหนดก่อนว่าลักษณะบ้านที่อยากจะได้นั้นควรมีคุณลักษณะอย่างไร (ตามหลักวิศวกรรม) เช่น มีห้องนอน 4 ห้อง ห้องน้ำ 4 ห้องอยู่ภายในห้องนอน ห้องนั่งเล่น 1 ห้อง ห้องครัว 1 ห้อง โดยที่ห้องนอนทุกห้องมีขนาด 3 คูณ 4 ตารางเมตรห้องน้ำมีขนาด 2 คูณ 2 ตารางเมตร เป็นต้น และมีการกำหนดอีกว่าห้องครัวต้องไม่ติดกับห้องนอนและห้องนั่งเล่นและเผื่อที่ไว้สำหรับทำโรงจอดรถด้วยในอนาคต ตามลำดับเหล่านี้และคือสถาปัตยกรรมของบ้าน ในทางซอฟต์แวร์ก็เหมือนกับการสร้างบ้านที่ต้องมีการกำหนดคุณลักษณะของซอฟต์แวร์ก่อนเสมอเพียงแต่การกำหนดสถาปัตยกรรมของซอฟต์แวร์นั้นไม่สามารถมองเห็นได้จับต้องได้เหมือนบ้าน (Intangible Object) เช่น สามารถนำออกข้อมูลในรูปแบบต่าง ๆ ได้โดยผ่านทางอุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ เช่นดิสก์ (Disk) เป็นต้น

ข้อสรุปที่สำคัญเกี่ยวกับยูนิฟายด์โพรเซส

1) การนำกลับมาใช้ได้อีก (Reusable) ทั้งส่วนที่เป็นองค์ประกอบ (Component) และตัวสถาปัตยกรรม (Architecture) ของระบบ

2) เพื่อเป็นพื้นฐานในการจัดการโครงการ เช่นการวางแผนโครงการการจัดทีมพัฒนาและกำหนดการส่งมอบซอฟต์แวร์ที่พัฒนา

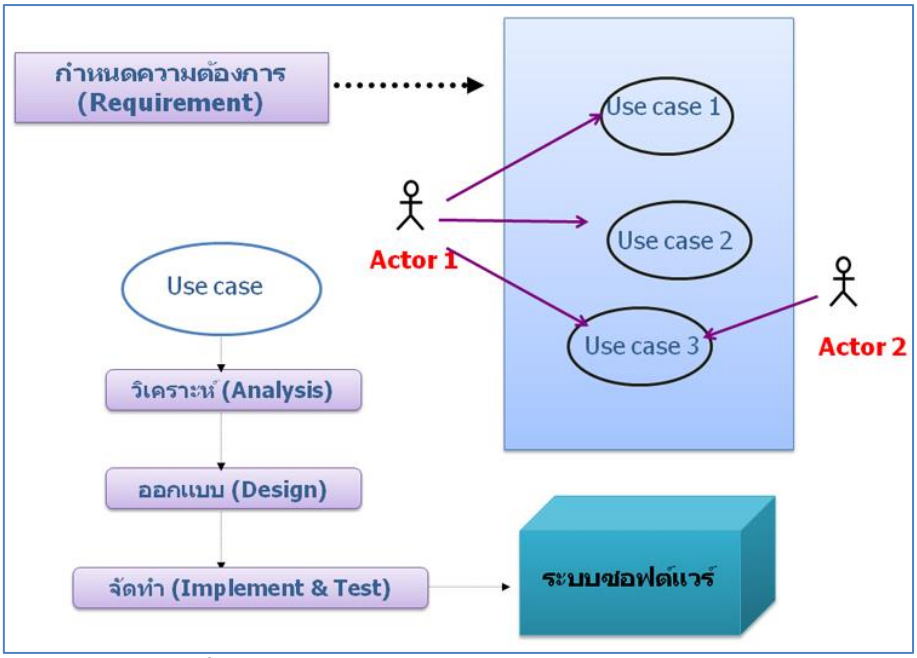
3) เพื่อให้มีการควบคุมการทำงานอย่างชาญฉลาดในความซับซ้อนและการดูแลความเป็นเอกภาพของระบบที่พัฒนา

การสร้างต้นแบบของระบบที่สามารถมองเห็นได้ด้วยยูเอ็มแอล

การสร้างต้นแบบของระบบที่มองเห็นได้คือ การทำให้เห็นภาพของระบบที่จะพัฒนาว่าควรมีหน้าตาเป็นอย่างไรหากเปรียบเทียบก็คงจะขอย้อนกลับในหัว ข้อที่ 3 ว่าหลังจากวาดสถาปัตยกรรมของบ้านที่จะสร้างไว้ในใจแล้วก็ต้องนำเสนอสิ่งที่คิดนึกให้บุคคลที่เกี่ยวข้องในการสร้างหรือบุคคลที่สนใจได้เข้าใจว่าสิ่งที่จะสร้างนั้นมีรูปร่างหน้าตาเป็นอย่างไร โดยการสร้างแบบจำลองของบ้านจริง (Model) ที่จะสร้างในสัดส่วนที่ย่อขนาดจากของจริงและซอฟต์แวร์ก็เช่นเดียวกันที่จะต้องมีการนำเสนอให้ทีมงานพัฒนาได้เห็นภาพของระบบที่จะพัฒนาว่ามีลักษณะเป็นอย่างไรเช่น มีองค์ประกอบอะไรบ้าง แต่ละองค์ประกอบเชื่อมต่อกันอย่างไร เป็นต้นซึ่งหากนำเสนอในรูปที่เป็นรูปธรรมดังเช่น แบบจำลองบ้านก็คงเป็นไปไม่ได้ทั้งนี้เนื่องจากธรรมชาติของตัวซอฟต์แวร์แล้วไม่อาจนำเสนอในรูปแบบดังกล่าวได้ดังนั้นผู้เชี่ยวชาญด้านการพัฒนาซอฟต์แวร์จึงได้เสนอวิธีการนำเสนอแบบจำลองทางซอฟต์แวร์ในรูปของการใช้ภาษาสัญลักษณ์ (Notation Language) มาอธิบายในลักษณะของแผนภาพ (Diagram) โดยภาษาสัญลักษณ์ที่ใช้ก็อาจใช้ภาษาใดก็ได้ เช่น Object Oriented Software Engineering, Object Modeling Technique หรือ Unified Modeling Language เป็นต้นแต่ในที่นี้จะใช้ภาษาสัญลักษณ์ที่มีความสมบูรณ์และเป็นที่นิยมใช้ในการพัฒนาระบบเชิงวัตถุ คือ Unified Modeling Language สำหรับภาษายูเอ็มแอลจะประกอบด้วยแผนภาพทั้งหมด 13 ประเภทให้เลือกใช้ได้ตามความเหมาะสมของแต่ละงาน ซึ่งในที่นี้จะกล่าวถึงเพียง 9 ประเภทหลักโดยจะอธิบายแต่ละแผนภาพอย่างคร่าว ๆ ก่อนและจะอธิบายรายละเอียดของแต่ละแผนภาพในบทที่ 4 ต่อไป

1) แผนภาพยูสเคส

แผนภาพยูสเคส (Use Case Diagram) เป็นแผนภาพที่ทำหน้าที่เก็บรวบรวมความต้องการของระบบโดยมีเทคนิคในการสร้างแบบจำลองเพื่อใช้อธิบายหน้าที่ของระบบใหม่หรือระบบปัจจุบัน ซึ่งเป็นกระบวนการสร้าง ยูสเคสเป็นแบบรอบโดยความต้องการของระบบจะได้จาก ลูกค้า ผู้ใช้ ผู้พัฒนาระบบ องค์ประกอบของยูสเคส ได้แก่ยูสเคส (Use Case) แอกเตอร์ (Actor) ความสัมพันธ์ระหว่างยูสเคส (Use Case Relationship) และระบบ (System)



ภาพที่ 3.5 แสดงการใช้ยูสเคสในการจัดการความต้องการ

2) แผนภาพคลาส

แผนภาพคลาส (Class Diagram) คือสิ่งที่เป็นผลผลิตที่ได้จากขั้นตอนการออกแบบระบบโดยมันจะใช้อธิบายมุมมองเชิงมุมมองแบบโครงสร้างเชิงสถิตย์ (Static View หรือ Static Structure) ของระบบที่กำลังจะพัฒนานั้น ในแง่ของรายละเอียดของการแก้ปัญหาของระบบมากกว่าจะอธิบายสภาพหรือลักษณะปัญหาที่พบของระบบดังกล่าวกล่าวคือ มันจะอธิบายว่าระบบนั้นถูกแก้ปัญหาอย่างไรมีรายละเอียดและหนทางแก้ปัญหาอย่างไรบ้างซึ่งสิ่งดังกล่าวก็ล้วนเกิดจากการกำหนดขึ้นมาโดยผู้พัฒนาระบบนั้นเอง (Artifact) ซึ่งเรียกมันว่าข้อกำหนดขององค์ประกอบภายในระบบหรือเรียกว่า Specification of Software Class

3) แผนภาพวัตถุ

แผนภาพวัตถุ (Object Diagram) ประกอบด้วยวัตถุและความสัมพันธ์ระหว่างวัตถุ โดยแต่ละวัตถุจะแสดงอินสแตนซ์ของแต่ละคลาสที่มีในระบบและความสัมพันธ์ต่าง ๆ ระหว่างคลาส เช่น Dependency, Generalization, Association จะมีลักษณะเช่นเดียวกับในแผนภาพคลาส

4) แผนภาพสถานะ

แผนภาพสถานะ (State Diagram) ประกอบด้วยสถานะและเหตุการณ์ต่าง ๆ ของวัตถุที่ทำให้สถานะของวัตถุเปลี่ยนและการกระทำที่เกิดขึ้นเมื่อสถานะของระบบเปลี่ยนไป สามารถบอกสถานะของวัตถุได้ โดยจะให้ความสนใจว่า ณ เวลาใด ๆ วัตถุนั้นมีสถานะเป็นแบบใด

5) แผนภาพประสาน

แผนภาพประสาน (Collaboration Diagram) ทำหน้าที่เช่นเดียวกับแผนภาพลำดับแต่รูปแบบและลักษณะการเขียนจะต่างกัน

6) แผนภาพลำดับ

แผนภาพลำดับ (Sequence Diagram) คือ บทบรรยายรายละเอียดการทำงานภายในยูสเคสใด ๆ ด้วยภาพนั้นเอง ซึ่งจะเป็นภาพหรือสัญลักษณ์ที่แสดงถึงการโต้ตอบกันระหว่างผู้ใช้กับระบบที่กำลังพัฒนาในลักษณะของการร้องขอบริการของแอกเตอร์และการตอบสนองของระบบตามลำดับของเหตุการณ์ก่อน หลัง โดยที่ตัวแผนภาพลำดับเองจะแสดงถึงกลุ่มของวัตถุต่าง ๆ ที่ต้องทำงานร่วมกัน (Collaboration) เพื่อให้บรรลุเป้าหมายการทำงานของยูสเคสหนึ่ง ๆ ทั้งนี้การโต้ตอบระหว่างวัตถุหรือแอกเตอร์กับวัตถุจะอยู่ในลักษณะของผู้ส่งข่าว (Sender Object) กับผู้รับข่าวสาร (Received Object) ตามลำดับ ส่วนข่าวสาร (Message) ที่จะส่งหากันนั้นจะใช้เส้นลูกศรหัวสีดำทึบแสดงและมีการกำหนดชื่อกำกับเส้นลูกศรดังกล่าวด้วยเพื่อแสดงให้เห็นความหมายของข่าวสาร (Message) ดังกล่าวว่าจะให้วัตถุที่รับข่าวสาร (Message) นั้นทำหรือปฏิบัติอะไร

7) แผนภาพดีพลอยเมนท์

แผนภาพดีพลอยเมนท์ (Deployment Diagram) คือแผนภาพที่ใช้แทนการแสดงสถาปัตยกรรมของระบบ รวมทั้งความสัมพันธ์ระหว่างฮาร์ดแวร์และซอฟต์แวร์ที่ใช้ในระบบ

8) แผนภาพกิจกรรม

แผนภาพกิจกรรม (Activities Diagram) แสดงลำดับกิจกรรมของการทำงาน สามารถแสดงทางเลือกที่เกิดขึ้นได้ แผนภาพกิจกรรมจะแสดงขั้นตอนการทำงานในการปฏิบัติการ โดยประกอบไปด้วยสถานะต่าง ๆ ที่เกิดขึ้นระหว่างการทำงาน และผลจากการทำงานในขั้นตอนต่าง ๆ

9) แผนภาพคอมโพเน้นท์

แผนภาพคอมโพเน้นท์ (Component Diagram) เป็น แผนภาพซึ่งแสดงโครงสร้างทางกายภาพของ Software โดยจะประกอบด้วยองค์ประกอบซึ่งอยู่ในรูปต่าง ๆ เช่น Binary Text และ Executable ภายใน Component Diagram ก็จะมีความสัมพันธ์แสดงอยู่เช่นเดียวกับ Class diagram, Object diagram

การตรวจสอบคุณภาพของซอฟต์แวร์ที่พัฒนาอย่างต่อเนื่อง

การตรวจสอบคุณภาพของซอฟต์แวร์ที่พัฒนาอย่างต่อเนื่อง (Continuously Verify) จะเป็นการกำหนดหลักเกณฑ์และเงื่อนไขว่าซอฟต์แวร์ที่ได้ควรมีคุณภาพอย่างไรนอกเหนือจากความสอดคล้องกับความต้องการของผู้ใช้แล้วซึ่งการทดสอบจะมีการทดสอบซอฟต์แวร์ที่พัฒนาขึ้นซึ่งอยู่รูปของโปรแกรมที่สามารถประมวลผลได้ (Executable Release) ทุกรอบของการพัฒนา (Iteration) สำหรับเกณฑ์หรือเงื่อนไขทั่วไปที่ใช้ในการทดสอบนั้นสามารถแบ่งออกเป็น 4 ประเด็นหลัก ๆ คือ

1) การทดสอบการทำงานของระบบ (Function Testing) คือ การทดสอบในเรื่องของคุณลักษณะของระบบ (Features) ขั้นตอนการทำงานของหน้าที่ระบบ และการรักษาความปลอดภัยของระบบ เป็นต้น

2) การทดสอบการใช้ระบบ (Usability Testing) คือ การทดสอบการใช้ระบบเป็นการประเมินสภาพการใช้งานระบบจากมุมมองของผู้ใช้โดยเน้นไปที่ปัจจัยเกี่ยวกับมนุษย์ที่เป็นผู้ใช้ระบบ (Human Factor) ความเป็นเอกภาพของส่วนติดต่อผู้ใช้ (GUI: Graphic User Interface) คำแนะนำ (On-line Help) เอกสารในการพัฒนาและการใช้ระบบ

3) การทดสอบความน่าเชื่อถือของระบบ (Reliability Testing) คือ การทดสอบว่าซอฟต์แวร์ที่พัฒนาขึ้นนั้นทำงานได้น่าเชื่อถือเพียงใดและไม่ก่อให้เกิดข้อผิดพลาดในขณะทำงานเช่น การ Crash , Hang หรือ Memory Leaks เป็นต้น

4) การทดสอบประสิทธิภาพ (Performance Testing) คือ การทดสอบว่าซอฟต์แวร์ที่พัฒนาทำงานได้ถูกต้องและมีประสิทธิภาพเพียงใดภายใต้สภาวะที่เครื่องคอมพิวเตอร์กำลังประมวลผลซอฟต์แวร์ (Run Software) นั้นทำงานอย่างหนัก (Peak load)

5) การทดสอบการสนับสนุน (Supportability) คือ การทดสอบว่าซอฟต์แวร์ที่พัฒนานั้น สามารถที่จะนำไปติดตั้ง (Deployment) ตามที่ตั้งใจไว้ได้โดยจะรวมเอาทั้งการทดสอบการติดตั้งและการกำหนดโครงร่างของระบบ ตามลำดับ

การจัดการการเปลี่ยนแปลง

การจัดการการเปลี่ยนแปลง (Change Management) “ทำไมต้องมีการจัดการการเปลี่ยนแปลง” และ “จะจัดการ เปลี่ยนแปลงในแง่ใดบ้างภายใต้กระบวนการพัฒนาระบบ” ทำไมต้องมีการจัดการการเปลี่ยนแปลง” สาเหตุก็เนื่องจากว่าภายใต้ระบบที่ถูกพัฒนาในลักษณะ Software – Intensive ที่มีขนาดใหญ่นั้น มักจะมีการจัดแบ่งทีมรับผิดชอบหน้าที่ต่าง ๆ แยกกันออกไปตามความถนัดและกิจกรรมที่มีในกระบวนการพัฒนาซอฟต์แวร์นั้น เช่นทีมหาความต้องการของระบบที่พัฒนา ทีมวิเคราะห์และออกแบบระบบ ทีมเขียนโปรแกรมระบบ (Implementation) ทีมทดสอบระบบ เป็นต้นภายในทีมเหล่านี้ก็อาจแบ่งออกเป็นเป็นทีมย่อย ๆ อีก เช่นในทีมวิเคราะห์และออกแบบระบบ ก็อาจแบ่งเป็นทีมกำหนดสถาปัตยกรรมระบบทีมออกแบบยูสเคส ทีมวิเคราะห์ยูสเคส เป็นต้น ซึ่งงานต่าง ๆ เหล่านี้ล้วนสัมพันธ์ซึ่งกันและกัน (Object Management Group, 1997) นอกจากนี้หากทีมงานอยู่กันคนละสถานที่ทำงานอีกก็ย่อมเป็นอุปสรรคต่อการประสานการทำงานเป็นอย่างมากดังนั้นถ้าหากจัดการเรื่องการเปลี่ยนแปลงของระบบที่กำลังพัฒนาไม่ดีพอก็ย่อมจะส่งผลกระทบต่อการทำงานของทีมงานทั้งหมดได้เพราะอย่าลืมว่าผลที่ได้จากการทำงาน (Work product) ของทีมแต่ละทีมนั้นจะต้องมีการส่งต่อไปยังอีกทีมหนึ่งที่เกี่ยวข้องด้วยและผลที่ได้จากการทำงาน (Work Product) นั้นอาจมีหลายรุ่น (Multiple Version) ก็ได้ซึ่งทำยังไงระหว่างทีมจะรู้ว่ามีการใช้หรืออ้างอิงรุ่นของผลที่ได้จากการทำงานรุ่นใดเพราะฉะนั้นหากจัดการเปลี่ยนแปลงไม่ดีพอก็ส่งผลกระทบต่อกระบวนการพัฒนานั้นด้อยประสิทธิภาพลงได้และส่งผลให้เกิดความสับสนภายในระหว่างทีมพัฒนาเหล่านั้นตามมา

ส่วนคำถามที่สอง คือ “จะจัดการเปลี่ยนแปลงในแง่ใดบ้างภายใต้กระบวน การพัฒนาระบบ ?” ในการจัดการเปลี่ยนแปลงนั้นก็จะมีการนำเอาเทคนิค วิธีการที่เรียกว่า Unified Change Management (UCM) มาจัดการประเด็นต่าง ๆ ที่เกิดขึ้นในขณะพัฒนาระบบ ประกอบด้วยการจัดการเรื่องร้องขอการเปลี่ยนแปลงความต้องการของผู้ใช้การรายงานสถานะหรือสภาวะของระบบที่พัฒนาว่ามีระดับ ประเภท จำนวนการผิดพลาด (Error) ที่เกิดและแก้ไขไปแล้วเท่าไร เป็นต้น

แนวทางแบบปฏิบัติที่ดี

แนวทางแบบปฏิบัติที่ดี (Best Practices) ยูนิฟายด์โพรเซสได้กำหนดแนวทางอย่างกว้าง ๆ เพื่อการบรรลุแนวทางปฏิบัติที่ดี “Best Practices” ไว้ดังนี้

1) ต้องมีการใช้แนวทางการพัฒนาแบบวนกลับ (Iterative Development) เพื่อการระบุคอมโพเน้นท์ (Component) ที่อาจเพิ่มขึ้นได้เมื่อเวลาการพัฒนาระบบผ่านไปซึ่งจะทำให้ผู้พัฒนาตัดสินใจได้ว่าควรจะซื้อสร้างเองหรือนำคอมโพเน้นท์เดิมที่เคยสร้างไว้กลับมาใช้ใหม่

2) ต้องมีการกำหนดหรือจัดทำคำแนะนำ (Guideline) สำหรับกิจกรรมและสิ่งต่าง ๆ ที่เป็น ผลที่เกิดจากการพัฒนา (Artifacts)

3) กระบวนการทั้งหลายในการพัฒนาจะต้องมุ่งไปที่การกำหนดสถาปัตยกรรมของระบบซอฟต์แวร์ที่จะพัฒนาก่อน ทั้งนี้เพื่อให้ผู้พัฒนาสามารถระบุว่าระบบงานที่พัฒนาควรจะประกอบคอมโพเน้นท์ไปด้วยกี่คอมโพเน้นท์และคอมโพเน้นท์เหล่านี้ควรจะถูกเชื่อมต่อหรือเชื่อมโยงการทำงานร่วมกันด้วยวิธีการใดตลอดจนถึงกลไกพื้นฐานและรูปแบบ (Pattern) ต่าง ๆ ที่นำมาใช้ในการโต้ตอบระหว่างคอมโพเน้นท์ต่าง ๆ ภายในระบบ

4) ต้องใช้ยูสเคสเป็นเครื่องมือในการผลักดันให้เกิดการออกแบบและการสร้างรหัสคำสั่งของระบบที่พัฒนาเช่น การใช้ตัวแบบยูสเคสในการพิจารณาหรือกำหนดรายการความเสี่ยงสูงของระบบที่กำลังพัฒนาให้เป็นสิ่งที่ต้องดำเนินการเร่งด่วน

5) ต้องมีการสร้างตัวแบบ (Model) ของระบบที่กำลังพัฒนาเพื่อเป็นการนิยาม (Abstract) ลักษณะของระบบที่กำลังพัฒนา เช่นการใช้ ยูเอ็มแอลเพื่อนำเสนอหรือแสดงต้นแบบที่เห็นได้ (Visual Model) สิ่งหนึ่งที่น่าสังเกตก็คือยูนิฟายด์โพรเซสจะให้ความสำคัญกับคำว่า "ทีม" หรือ "ทีมงาน"กล่าวคือ ภายใต้กระบวนการใด ๆ ก็ตามที่เกิดขึ้นในพัฒนาซอฟต์แวร์นั้นจะต้องมีการกำหนดอย่างชัดเจน ภายในทีมพัฒนาโปรแกรมว่าควรประกอบไปด้วยใครบ้างแต่ละคนควรทำอะไรทำเมื่อไหร่และสุดท้ายคือทำอย่างไรตามลำดับทั้งนี้เพื่อให้บรรลุเป้าหมายที่กำหนดไว้อย่างแน่นอนในแต่ละระยะของการพัฒนา (Phases)

วงจรชีวิตของยูนิฟายด์โพรเซส (Unified Process Lifecycle)

วงจรชีวิตของยูนฟายด์โพรเซสจะมีวงจรชีวิต (Life cycle) หรือระยะ (Phases) ของกระบวนการทั้งหมด 4 ระยะ ดังนี้

ระยะที่ 1. เตรียมงาน (Inception) คือ ระยะของการนิยามขอบเขตของโครงการ

ระยะที่ 2. ทำรายละเอียด (Elaboration) คือ ระยะของการวางแผนโครงการ จัดทำรายละเอียดความต้องการ จัดสร้างสถาปัตยกรรมระบบ

ระยะที่ 3.จัดสร้าง (Construction) คือ ระยะของการสร้างและทดสอบโปรแกรม

ระยะที่ 4.ถ่ายโอน (Transition) คือ ระยะของติดตั้งถ่ายโอนระบบให้กับผู้ใช้

ภาพที่ 3.6 วงจรชีวิตของยูนิฟายด์โพรเซส

ระยะที่ 1.ระยะเตรียมการ (Inception phases) คือระยะเริ่มโครงการที่ต้องมีการกำหนดขอบเขต (Scoping) ของโครงการอย่างชัดเจน โดยมีการกำหนดสิ่งที่เรียกว่า Primary Use Cases และแอกเตอร์ตามลำดับ ซึ่งสิ่งนี้จะเป็นยูสเคสและแอกเตอร์ที่สำคัญของระบบที่กำลังพัฒนาตลอดจนการร่างแผนการพัฒนา (Business plan) ที่ประกอบด้วยกิจกรรมหลัก ระยะเวลาเป้าหมาย และทรัพยากรใดบ้างที่ต้องใช้ในโครงการพัฒนาระบบนั้นนอกจากนี้ยังมีการกำหนดหมุดหลักที่สำคัญ(Major Milestone) ของระยะเอาไว้ด้วยเมื่อสิ้นสุดแต่ละระยะ โดยระยะนี้เขาบอกว่าจะต้องให้ได้สิ่งที่เรียกว่า ข้อตกลงร่วมกันของทีมพัฒนา (Agreement Statement) ตลอดจนความเสี่ยงทั้งหลายที่มีอยู่หรือเกิดขึ้นภายใต้โครงการที่พัฒนาและสามารถจัดความเสี่ยงเหล่านั้นออกเป็นระดับต่าง ๆ ได้เช่น ระดับต่ำ กลาง สูง เป็นต้น อย่างสมเหตุสมผลและเข้าใจร่วมกันภายใต้ทีมพัฒนา โดยผลลัพธ์ที่ควรจะได้เมื่อผ่านระยะที่ 1 ดังแสดงในตารางที่ 3.2

ตารางที่ 3.2 ผลลัพธ์ที่ควรจะได้เมื่อผ่านระยะที่ 1

ลำดับ	รายการ	หมายเหตุ
1	เอกสารวิสัยทัศน์ (Vision document)	
2	โมเดลยูสเคส(Use case model) เบื้องต้น (10-20%)	
3	พจนานุกรมคำศัพท์โครงการ(Project glossary) เบื้องต้น	
4	กรณีการทำธุรกิจ ครอบคลุมบริบทธุรกิจ เงื่อนไข (Business Rules)	

5	การจัดการความเสี่ยงเบื้องต้น(Risk management)	
6	แผนโครงการแสดงระยะงานและรอบงาน 6.1) GanttChart 6.2) PERT Chart	
7	โมเดลธุรกิจ ถ้ามี	
8	ต้นแบบระบบ(Prototypes)	

ระยะที่ 2. ระยะลงรายละเอียด (Elaboration Phases) คือระยะของการกำหนดความต้องการและ Baseline ของสถาปัตยกรรมของระบบหรือโครงการพัฒนานั้นกล่าวคือถ้าหากมีการกำหนดความต้องการและสถาปัตยกรรมของระบบดีแล้ว(เช่น ชัดเจนครบถ้วน ตรงความต้องการของผู้ใช้)มันจะช่วยลดความเสี่ยงโครงการที่จะประสบกับความล้มเหลวน้อยลงในช่วงสุดท้ายของระยะนี้ จะเป็นการกำหนดรายละเอียดของการประมาณการค่าใช้จ่ายปริมาณทรัพยากรที่จะว่าควรมีปริมาณเท่าใด สำหรับหมุดหลักที่สำคัญ(Major Milestone) ของระยะนี้คือการกำจัดความเสี่ยงที่สูงออกไปจากโครงการและต้องมีการกำหนดสถาปัตยกรรมของระบบให้แล้วเสร็จ โดยผลลัพธ์ที่ควรจะได้เมื่อผ่านระยะที่ 2 ดังแสดงในตารางที่ 3.3

ตารางที่ 3.3 ผลลัพธ์ที่ควรจะได้เมื่อผ่านระยะที่ 2

ลำดับ	รายการ	หมายเหตุ
1	ยูสเคสโมเดลสมบูรณ์อย่างน้อย 80%	
2	ความต้องการเพิ่มเติม (Supplementary Requirement nonfunctional requirements)	
3	คำอธิบายสถาปัตยกรรมซอฟต์แวร์(Software Architecture Description)	
4	ต้นแบบที่สามารถนำไปประมวลผลได้(Execute)ได้	
5	ปรับรายการความเสี่ยง	
6	แผนการพัฒนาระบบทั้งโครงการ	
7	คู่มือผู้ใช้งานบางส่วนคู่มือผู้ใช้ (User manual) ซึ่งในขั้นตอนนี้อาจจะมีหรือไม่ก็ได้	

ระยะที่ 3. ระยะจัดสร้าง(Construction Phases) คือระยะที่ทำการสร้างระบบ โดยจะมีการแบ่งรอบการทำงานออกเป็นหลายรอบการทำงาน(Iteration) จนกว่าจะมีการปล่อยระบบออกไปใช้ที่เรียกว่า Beta Version สำหรับหมุดหลักที่สำคัญของระยะนี้คือระบบหรือซอฟต์แวร์ที่พัฒนานี้จะต้องมีความสมบูรณ์และมีคุณภาพอยู่ในระดับที่ยอมรับได้ตลอดจนความพร้อมของกลุ่มผู้ใช้(Stake Holder) เช่น ผู้ใช้ทั่วไป ผู้ดูแลระบบผู้ดูแลฐานข้อมูลของระบบ เป็นต้นต้องมีความพร้อมในการเปลี่ยนถ่ายจากระบบเดิมไปสู่ระบบใหม่และ

รวมทั้งระดับค่าใช้จ่ายของโครงการเป็นที่ยอมรับได้ โดยผลลัพธ์ที่ควรจะได้เมื่อผ่านระยะที่ 3 ดังแสดงในตารางที่ 3.4

ตารางที่ 3.4 ผลลัพธ์ที่ควรจะได้เมื่อผ่านระยะที่ 3

ลำดับ	รายการ	หมายเหตุ
1	ระบบซอฟต์แวร์ที่ตรงกับแพลตฟอร์มเป้าหมาย	
2	คู่มือใช้งาน	
3	คำอธิบายระบบที่ออกวางจำหน่าย	

ระยะที่ 4.ระยะถ่ายโอน(Transition Phases) คือระยะที่ส่งมอบหรือเปลี่ยนจากระบบเดิมไปสู่ระบบใหม่ตลอดจนการฝึกอบรมระบบใหม่ให้กับผู้ใช้การติดตั้งและสนับสนุนการใช้ระบบใหม่ให้กับผู้ใช้ตามลำดับ สำหรับหมุดหลักที่สำคัญของระยะนี้คือการปล่อยระบบงานที่พัฒนาออกสู่การใช้งานของผู้ใช้และยังรวมถึงการกำหนดและเริ่มต้นวงจรการพัฒนาใหม่เพื่อขยายขีดความสามารถของระบบที่ส่งมอบไปแล้วนั้นอีก โดยผลลัพธ์ที่ควรจะได้เมื่อผ่านระยะที่ 4 ดังแสดงในตารางที่ 3.5

ตารางที่ 3.5 ผลลัพธ์ที่ควรจะได้เมื่อผ่านระยะที่ 4

ลำดับ	รายการ	หมายเหตุ
1	ทดสอบเพื่อรับระบบ	
2	ทดสอบการ Integrate ของระบบ ความเข้ากันได้ของระบบย่อยต่าง ๆ	
3	แปลงข้อมูลลงสู่ฐานข้อมูลที่ใช้งาน	
4	อบรมผู้ใช้และผู้บำรุงรักษาระบบ	
5	ออกวางระบบสู่ท้องตลาด	

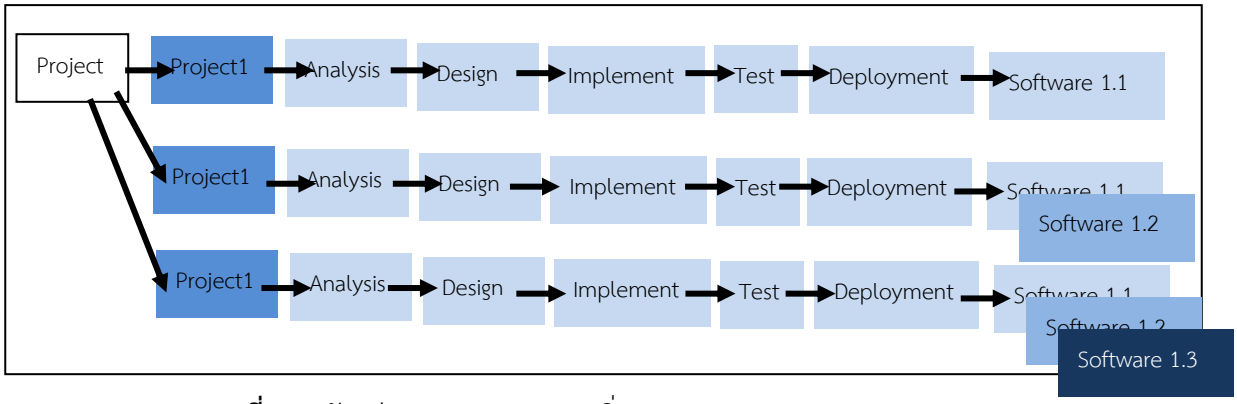
โดยสรุปแล้ว ในระยะต่าง ๆ ของการพัฒนาแบบยูนิฟายด์โพรเซสนี้จะไม่มีการกำหนดระยะเวลาสิ้นสุดไว้อย่างตายตัวทั้งนี้แล้วแต่ลักษณะของโครงการที่ดำเนินการว่ามีขนาดและความซับซ้อนของระบบว่ามีมากน้อยขนาดใดนอกจากนี้ในแต่ละระยะจะมีการกำหนดและแบ่งรอบการทำงานแบบวนกลับออกเป็นหลาย ๆรอบในลักษณะวงจรชีวิตการพัฒนาซอฟต์แวร์แบบน้ำตก(Waterfall Model) สิ่งสำคัญที่ต้องคำนึงถึงก็คือการพัฒนาที่ใช้กระบวนการแบบยูนิฟายด์โพรเซสนี้หากองค์กรใดมีการนำไปใช้และใช้กับโครงการพัฒนาซอฟต์แวร์หลาย ๆ โครงการพร้อมกันต้องระวังการคาบเกี่ยวกันของการใช้ทรัพยากรบุคคลหรือกำลังคนดังนั้นต้องกำหนดความสอดคล้องการใช้กำลังคน (Man power) ระหว่างโครงการพัฒนาเหล่านั้นให้ดี

รูปลักษณ์ของยูนิฟายด์โพรเซส

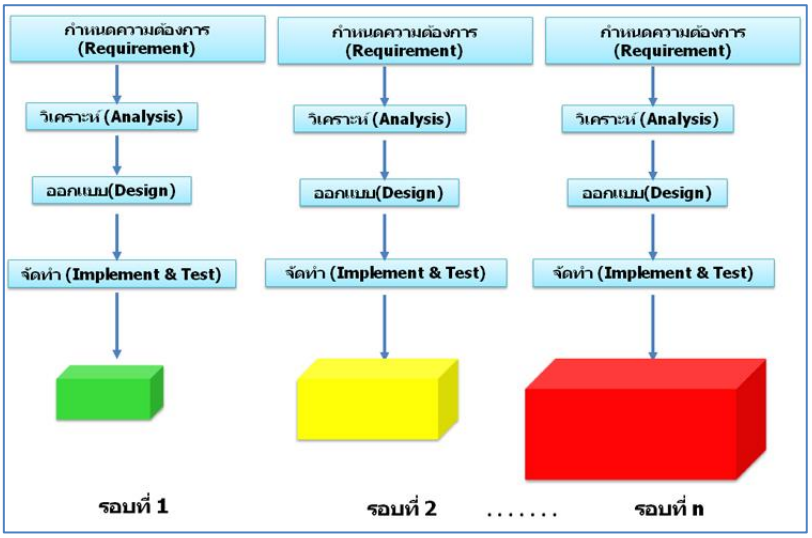
1. เป็นกรรมวิธีวนรอบเพิ่มพูน (An Iterative and Incremental Process)
2. เป็นกรรมวิธีที่มียูสเคสเป็นตัวขับเคลื่อน (An Use-Case Driven Process)
3. เป็นกรรมวิธีที่มีสถาปัตยกรรมเป็นศูนย์กลาง (An Architecture-Centric Process)

1. ยูนิฟายด์โพรเซสทำงานเป็นรอบเพิ่มพูนผล (Iterative and incremental)

โครงการพัฒนาระบบถูกจัดออกเป็นหลายโครงการย่อยโดยแต่ละโครงการย่อยมีกำหนดเวลาแผนงานของตัวเอง เรียกแต่ละโครงการย่อยว่า รอบงาน(Iteration) ซึ่งแต่ละรอบงานกระทำการพัฒนาระบบตามขั้นตอนวิเคราะห์ ออกแบบ จัดสร้างของตัวเอง ได้ผลลัพธ์เป็นระบบที่นำไปประมวลผลได้ระบบจะถูกเพิ่มพูนให้โตขึ้นตลอดเวลาตามรอบงานแต่ละรอบงานที่ทำเสร็จจนกระทั่งได้ระบบที่สมบูรณ์ เป็นการพัฒนาระบบที่เรียกว่าวนรอบเพิ่มพูนผล(Iterative and incremental development) ดังแสดงตัวอย่างในภาพที่ 3.5 และภาพที่ 3.6

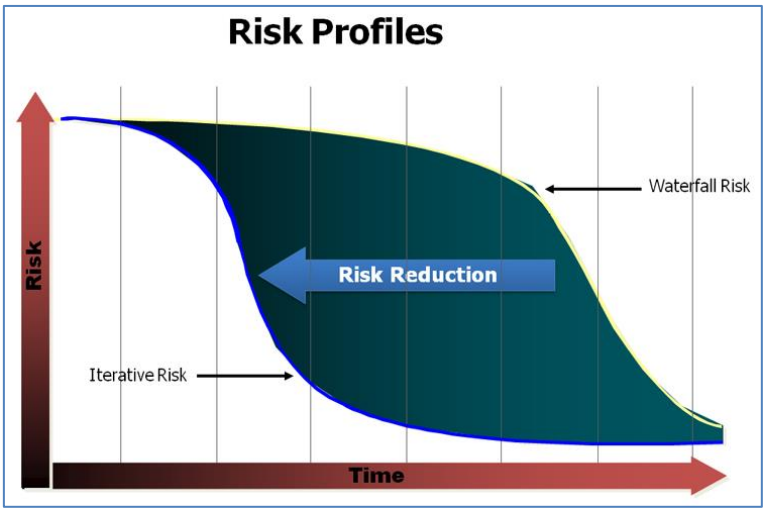


ภาพที่ 3.7 ตัวอย่างการวนรอบแบบเพิ่มพูน (Iterative and Incremental)



ภาพที่ 3.8 แสดงการทำงานแบบรอบเพิ่มพูน

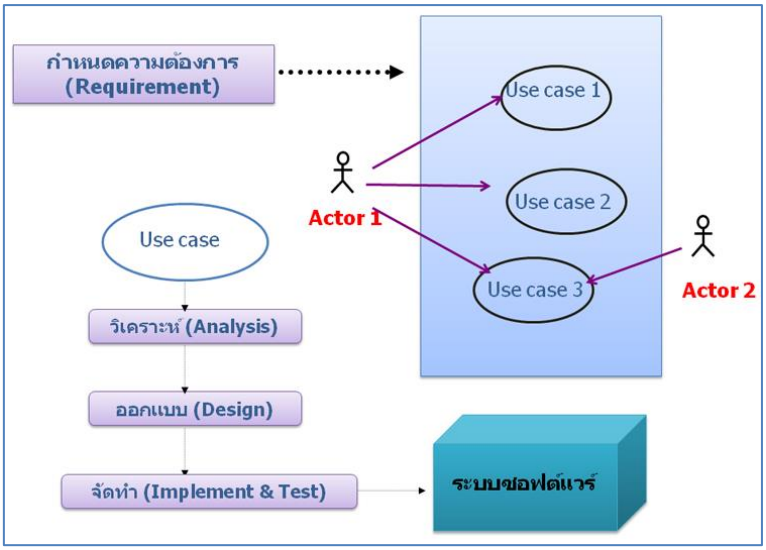
การพัฒนาแบบวนรอบเพิ่มพูนผลเป็นการจัดการกับความเสี่ยงที่มีผลกระทบสูงตั้งแต่ช่วงต้นโดยสร้างสถาปัตยกรรมระบบไว้ก่อนเพื่อใช้เป็นแนวทางในการพัฒนาระบบต่อไปและใช้เป็นกรอบงานที่สามารถจัดการกับการเปลี่ยนความต้องการที่ไม่สามารถเลี่ยงได้เป็นอย่างดี โดยเป็นการสร้างระบบโดยการเพิ่มพูนผลตลอดเวลาในแต่ละรอบงาน แทนที่จะทำระบบในครั้งเดียวให้เสร็จสมบูรณ์ตอนใกล้จบโครงการ ซึ่งถ้ามีการเปลี่ยนแปลงเกิดขึ้นจะมีค่าใช้จ่ายมากเป็นการจัดกรรมวิธีที่ทำให้ทีมงานทำงานได้อย่างมีประสิทธิภาพ



ภาพที่ 3.9 แสดงการเปรียบความเสี่ยงของวิธีวนรอบ(Iterative)และน้ำตก(Waterfall)

2. ยูนิฟายด์โพรเซสถูกขับเคลื่อนด้วยยูสเคส

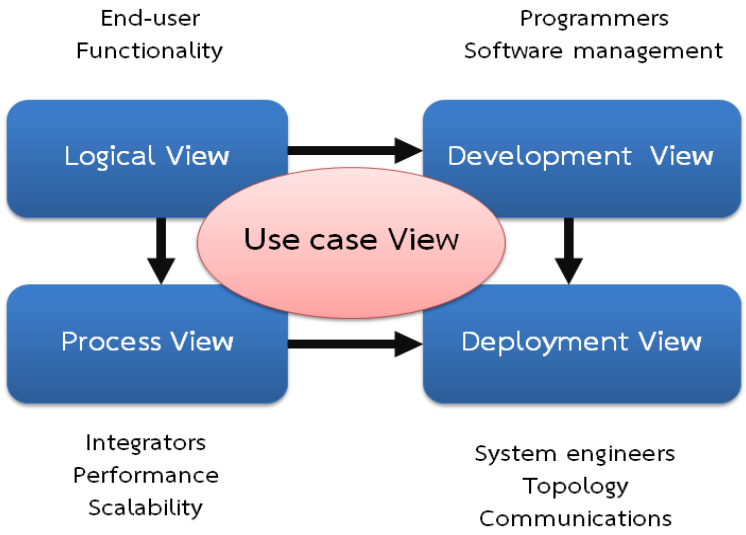
ยูนิฟายด์โพรเซสถูกขับเคลื่อนด้วยยูสเคส (Use case driven) หมายถึงยูสเคสเป็นฟังก์ชันการทำงานในระบบ ซึ่งการทำงานนั้นให้ผลลัพธ์ที่มีคุณค่าต่อผู้ใช้โดยแต่ละยูสเคสเป็นการทำงานที่ผู้ใช้คาดหวังเป็นความต้องการของผู้ใช้โดยใช้แผนภาพยูสเคสอธิบายการทำงานทั้งหมดที่มีในระบบเนื่องจากความต้องการคือยูสเคส การพัฒนาระบบจึงต้องกระทำกับยูสเคส ดังนั้นยูสเคสเป็นตัวขับเคลื่อนการออกแบบ การจัดสร้างและการทดสอบระบบการขับเคลื่นด้วยยูสเคส (Use case driven) หมายถึงการพัฒนาระบบทำตามขั้นตอนโดยใช้ยูสเคสเป็นหลัก เริ่มจากหาความต้องการระบุเป็นยูสเคส จากนั้นทำยูสเคสให้เป็นจริง โดยนำยูสเคสไปวิเคราะห์ ออกแบบ สร้างเป็นรหัสต้นฉบับ(Source code) แล้วทำการทดสอบว่ารหัสต้นฉบับที่สร้างขึ้นทำงานได้ถูกต้องตามที่ระบุไว้ในยูสเคส ดังแสดงในภาพที่ 3.10



ภาพที่ 3.10 แสดงการขับเคลื่อนยูนิฟายด์โพรเซสด้วยยูสเคส

3. ยูนิฟายด์โพรเซสมีสถาปัตยกรรมเป็นศูนย์กลาง

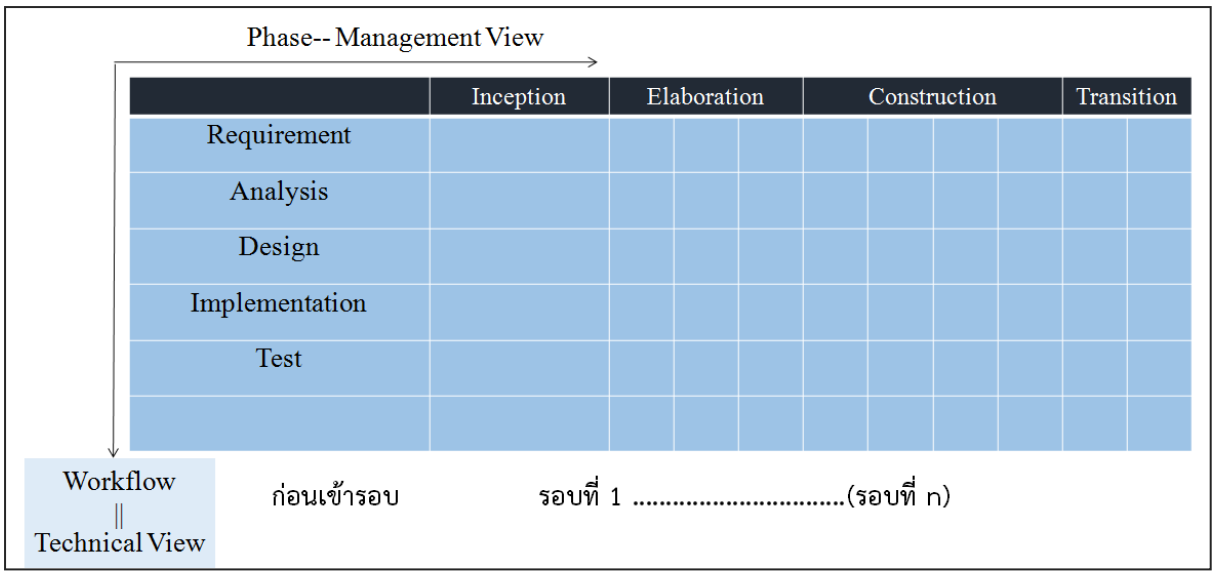
ยูนิฟายด์โพรเซสใช้สถาปัตยกรรมเป็นศูนย์กลาง (Architecture centric) ในการพัฒนาระบบแสดงภาพของระบบในมุมมองต่าง ๆ ดังแสดงในภาพที่ 3.8



ภาพที่ 3.11 แสดงสถาปัตยกรรมที่เป็นศูนย์กลางของยูนิฟายด์โพรเซส

รอบงานและระยะงาน

รอบงานและระยะงาน(Iterations and Phases) โดยทั่วไปโครงสร้างของยูนิฟายด์โพรเซสถูกจัดออกเป็น 2 มิติ คือมิติทางด้านเวลา(Time) หรือระยะของการทำงาน(Phase) และมิติกระแสงาน (Workflows) ดังแสดงในภาพที่ 3.9



ภาพที่ 3.12 วนรอบการทำงานของยูนิฟายด์โพรเซส

รอบงานและเฟส ในแต่ละเฟสแบ่งการทำงานออกเป็นรอบงาน (Iteration) จำนวนรอบงานที่มีในแต่ละเฟสจัดแบ่งตามความเหมาะสมของโครงการ

รอบงาน(Iteration) ในแต่ละรอบงานจะประกอบไปด้วยหลายชุดของของกิจกรรมหรือกระแสงาน ที่ผู้พัฒนาระบบจะต้องกระทำทั้งกระแสงานหลักและกระแสงานสนับสนุน การทำกิจกรรมเสร็จสิ้นในแต่ละรอบงานจะได้ผลลัพธ์เป็นงานที่สามารถนำไปประมวลผลซึ่งอาจจะเป็นประมวลผลภายในองค์กรของผู้พัฒนาระบบ หรือจัดส่งไปให้ลูกค้าเพื่อนำไปติดตั้งใช้งาน

กระแสงานหลัก(Main workflow)

1) Requirements
2) Analysis
3) Design
4) Implement
5) Test

กระแสงานสนับสนุน(Supporting workflow)

1) Deployment
2) Project Management
3) Configuration Management

โครงสร้างการทำงานของยูนิฟายด์โพรเซส (Static structure of the process)

UP กำหนดการปฏิบัติงานของทีมงานไว้อย่างชัดเจนว่าใคร (who) ทำอะไร (what) ทำเมื่อไร (when) และทำอย่างไร (how) เพื่อบรรลุเป้าหมายที่ตั้งไว้

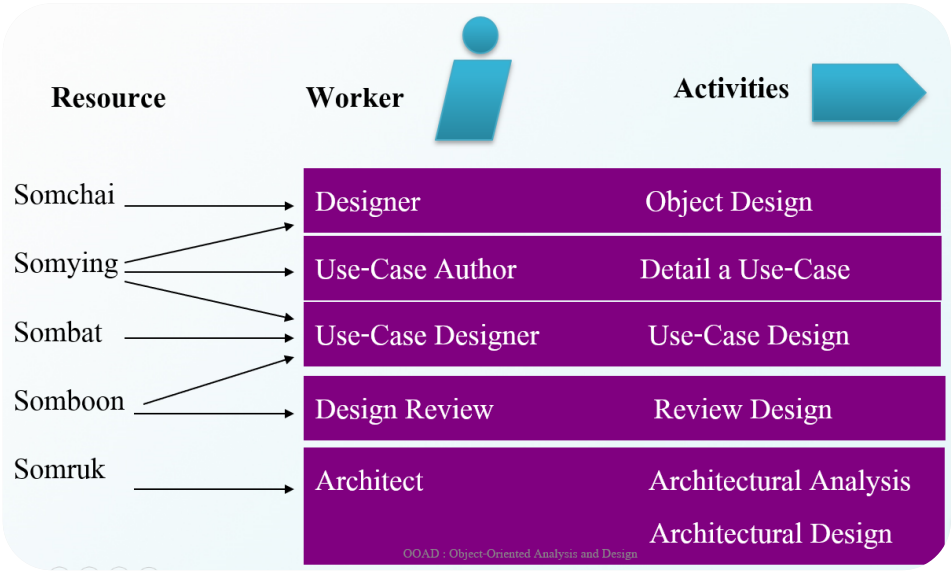
UP แสดงโครงสร้างการปฏิบัติงานเป็นโมเดลที่มี 4 สัญญลักษณ์พื้นฐาน

ตารางที่ 3.6 แสดงโครงสร้างการปฏิบัติงาน

สัญญลักษณ์	ความหมาย
ผู้ปฏิบัติ (worker) 	ผู้ปฏิบัติงาน (worker) – who หมายถึงบทบาทในโครงการที่กระทำโดยบุคคลใดบุคคลหนึ่งหรือทีม
ชิ้นงาน (artifact) 	ชิ้นงาน (artifact) – what หมายถึงงานที่ผลิตโดยผู้ปฏิบัติในรูปของข้อมูลสารสนเทศ โปรแกรม รหัสโปรแกรม หรือเอกสาร(document) กิจกรรม (activity) – how หมายถึงกิจกรรมที่ผู้ปฏิบัติกระทำ
กระแสงาน	กระแสงาน (workflow) – when หมายถึงชุดของกิจกรรมที่เกี่ยวเนื่อง

ผู้ปฏิบัติงาน

ผู้ปฏิบัติงานหรือ worker นิยามถึงพฤติกรรมและความรับผิดชอบ (บทบาทความรับผิดชอบ) ของตัวบุคคลหรือกลุ่มบุคคลที่ทำงานเป็นทีม



ภาพที่ 3.13 ตัวอย่างของการระหน้าที่และกิจกรรมของผู้ปฏิบัติ

จากภาพที่ 3.10 แสดงตัวอย่างของการกำหนดหน้าที่และกิจกรรมให้กับผู้ปฏิบัติงาน โดยแต่ละคนสามารถรับผิดชอบงานได้มากกว่าหนึ่งงาน จากตัวอย่างจะเห็นว่าสมชายมีหน้าที่เป็นดีไซน์เนอร์ ทำหน้าที่ออกแบบวัตถุ ในขณะที่สมหญิงเป็นทั้งผู้ค้นหายูสเคส ออกแบบยูสเคสและกำหนดรายละเอียดของยูสเคสตามลำดับ

กิจกรรม

กิจกรรม (Activity) ของผู้ปฏิบัติงานใด ๆ คือยูนิตงานที่ผู้ปฏิบัติงานนั้นได้รับการมอบหมายให้ทำจากผู้จัดการโครงการ (Project Manager) แต่ละกิจกรรมมีจุดหมายที่ชัดเจน โดยทั่วไปจะเป็นกิจกรรมในการสร้างหรือปรับปรุงชิ้นงาน (Artifact) เช่น การสร้างและปรับปรุงโมเดล หรือคลาส การจัดทำและปรับแผน

ตารางที่ 3.7 ตัวอย่างของกิจกรรม

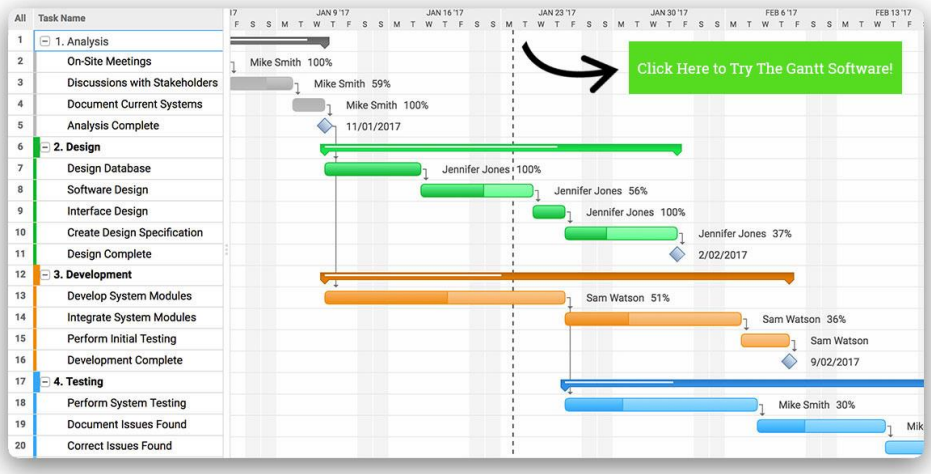
กิจกรรม (Activities)	ผู้ปฏิบัติ (Worker)	ตัวอย่าง
กำหนดรอบงาน	หัวหน้าโครงการ (Project Manager: Plan an iteration, for the Worker: Project Manager)	นัฐพงศ์ ส่งเนียมเป็น Project Manager ทำหน้าที่ มอบหมาย ควบคุม ติดตาม ประเมินผล worker โดยกำหนด รอบให้ ณัฐพล เก็บความต้องการของระบบ (Requirement) ใช้เวลา 2 สัปดาห์
หายูสเคสและแอกเตอร์	นักวิเคราะห์ระบบ (Find use cases and actors, for the Worker: System Analyst)	
ทบทวนการออกแบบ	นักออกแบบระบบ (Review the design, for the Worker: Design Reviewer)	
ตรวจสอบยูสเคส	นักออกแบบยูสเคส	
ทดสอบประสิทธิภาพของระบบ	นักทดสอบระบบ	

ชิ้นงาน

ชิ้นงาน (Artifact) เป็นส่วนของสารสนเทศที่ถูกจัดสร้าง บำรุงรักษา หรือถูกใช้ในการพัฒนาซอฟต์แวร์ ชิ้นงานเป็นผลิตภัณฑ์เกิดขึ้นระหว่างการพัฒนาระบบซึ่งนำไปสู่ผลิตภัณฑ์เป้าหมายของโครงการคือระบบซอฟต์แวร์ในที่สุด ชิ้นงานมีได้หลายรูปแบบ เช่น

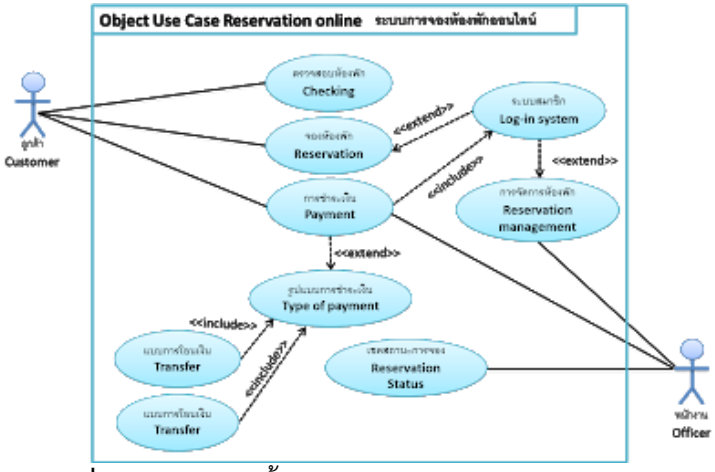
- A model, such as the Use-case Model or the Design Model
- A model element, i.e. an element within a model, such as a class, a use case or a subsystem
- A document, such as business Case or Software Architecture Document
- Source Code
- Executables
- Mockup ,wireframe

ตัวอย่าง ชิ้นงาน Gantt Chart แผนการดำเนินโครงการเป็นหน้าที่ของ PM



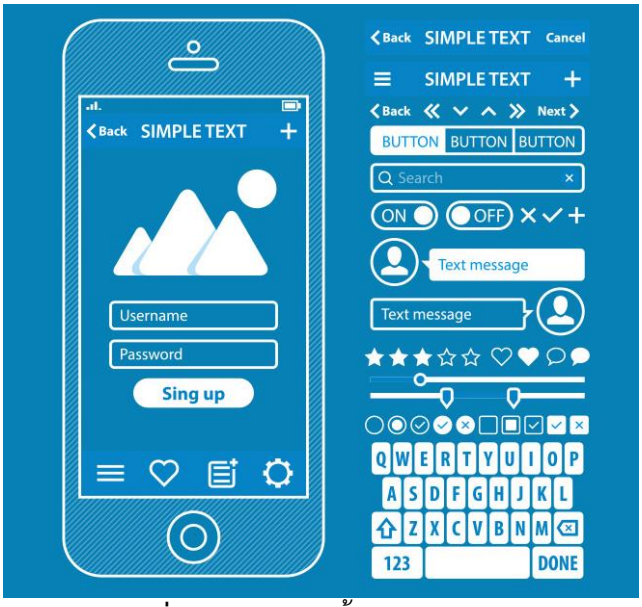
ภาพที่ 3.14 ตัวอย่างชิ้นงาน Gantt Chart

ตัวอย่างชิ้นงานแผนภาพยูสเคส (Use case) หน้าที่ของนักออกแบบยูสเคส (SA use case designer)

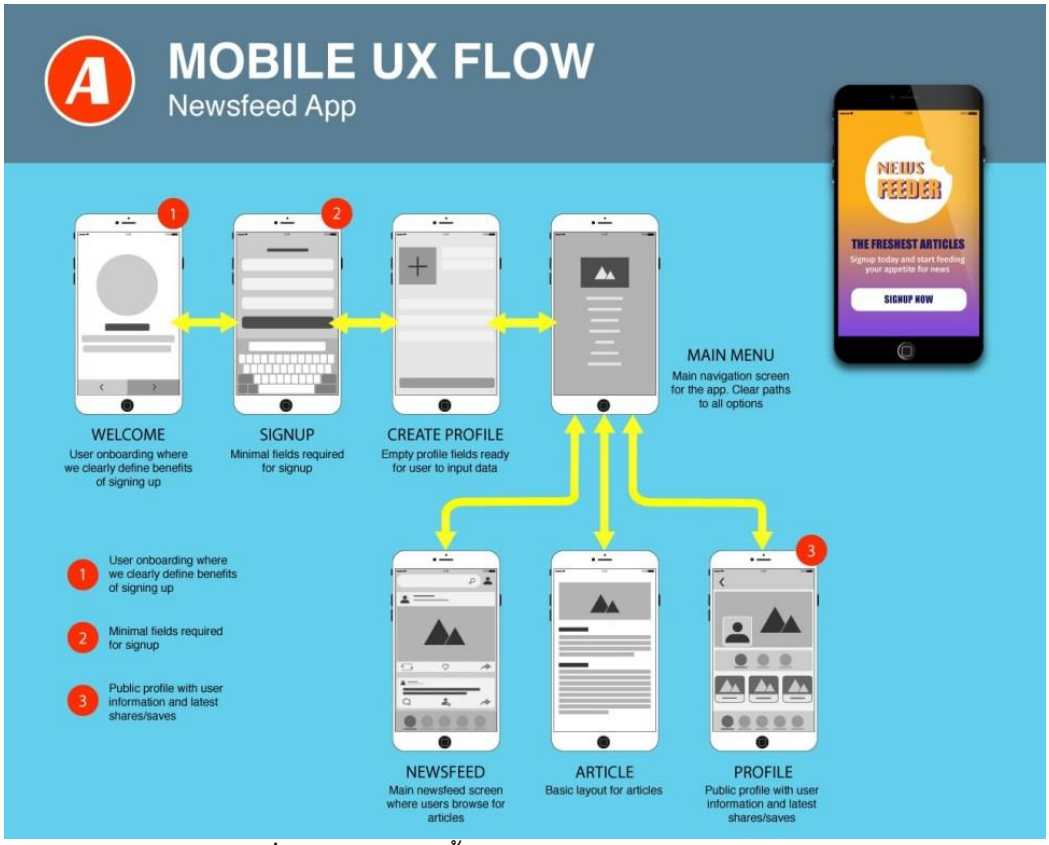


ภาพที่ 3.15 ตัวอย่างชิ้นงานยูสเคสสำหรับนักวิเคราะห์ระบบ

ตัวอย่างชิ้นงานของ UXUI



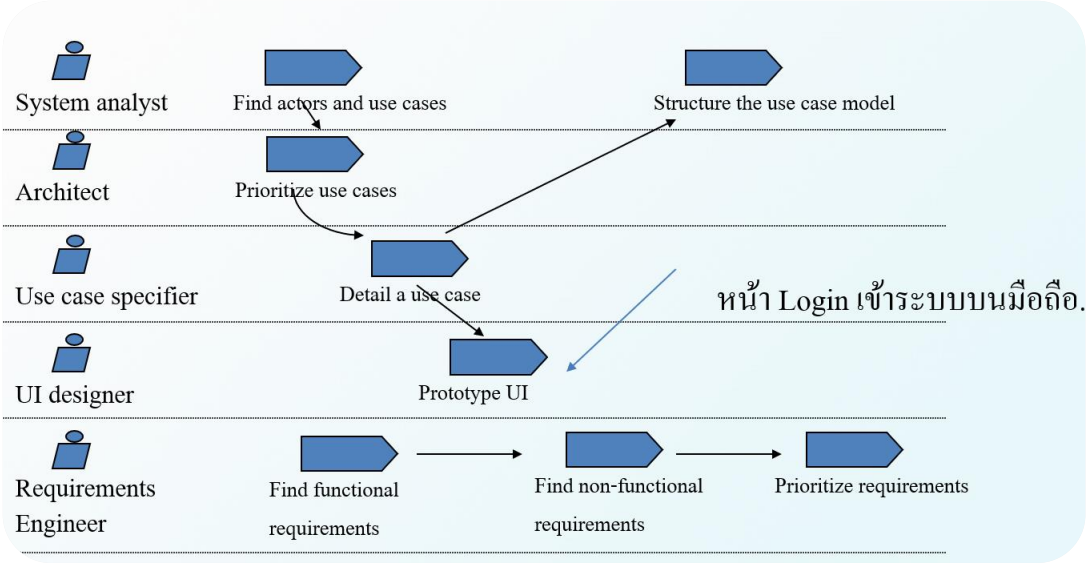
ภาพที่ 3.16 ตัวอย่างชิ้นงานของ UX/UI



ภาพที่ 3.17 ตัวอย่างชิ้นงานกระแสงานของแอปพลิเคชัน

กระแสงาน

กระแสงาน (workflow) เป็นชุดของกิจกรรมที่สร้างผลลัพธ์ที่มีคุณค่าต่อโครงการ โดยผู้ปฏิบัติงานทำงานร่วมกันจนเกิดเป็นกระแสงาน ดังตัวอย่างในภาพที่ 3.18

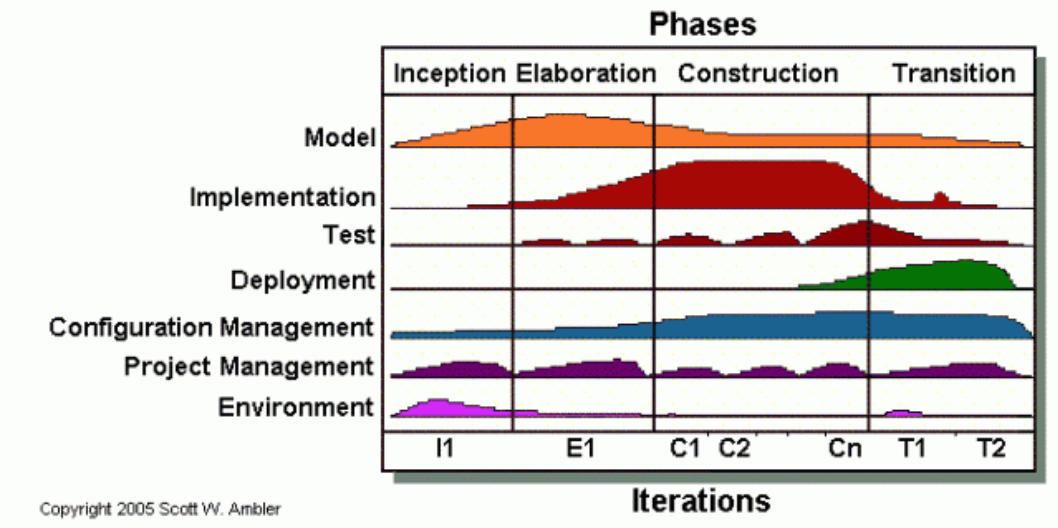


ภาพที่ 3.18 ตัวอย่างกระแสงาน

ความสัมพันธ์ระหว่างยูนิฟายด์โพรเซสกับยูเอ็มแอล

ยูเอ็มแอล (UML) ย่อมาจาก Unified Modeling Language เป็นภาษาที่ใช้อธิบายแบบจำลองต่าง ๆ หรือเป็นภาษาสัญลักษณ์ของรูปภาพมาตรฐาน สำหรับใช้ในการสร้างแบบจำลองเชิงวัตถุ โดยยูเอ็มแอลเป็นภาษามาตรฐานสำหรับสร้างแบบพิมพ์เขียวให้แก่ระบบงาน ซึ่งจะได้อธิบายรายละเอียดของยูเอ็มแอลในบทต่อไป

เนื่องจากยูนิฟายด์โพรเซสเป็นระเบียบวิธีปฏิบัติ (Methodology) ที่จะเป็นตัวกำหนดว่าเวลาใดจะใช้ยูเอ็มแอลตัวไหนในกระบวนการวิเคราะห์และออกแบบระบบด้วยวิธีเชิงวัตถุ(Object-oriented) โดยยูนิฟายด์โพรเซสจะแบ่งออกเป็นสองมิติ คือ กระบวนการพัฒนาระบบซึ่งประกอบด้วยช่วงระยะและกระแสการทำงาน (Workflows) โดยระยะเวลาจะแสดงในแนวนอน และกระแสการทำงานจะแสดงในแนวตั้งดังแสดงในภาพที่ 3.19



ภาพที่ 3.19 แสดงรอบการทำงานและกระแสงาน

จากในภาพข้างบนจะเห็นว่ามีการแบ่งออกเป็น 4 ระยะ ซึ่งแต่ละระยะมีความหมายดังต่อไปนี้

1) Inception ใช้แทนการเริ่มต้นของระบบ ซึ่งจะตรงกับ phase planning ของ UML

2) Elaboration เป็นช่วงของการลงรายละเอียดของระบบ จะตรงกับ phase analysis และ design ของ UML

3) Construction เป็นช่วงของการสร้างระบบ ซึ่งจะตรงกับช่วง implementation ของ UML

4) Transition เป็นช่วงของการส่งมอบระบบ ตรงกับช่วง implementation ของ UML เช่นเดียวกัน

ทั้ง 4 เฟสจะประกอบด้วยกระแสงาน(workflows)ที่ทำงานอยู่ในแต่ละเฟสดังต่อไปนี้

1) Model คืองานการออกแบบ

2) Implementation คืองานการจัดสร้างระบบทั้งหมด

3) Test คืองานการทดสอบ

4) Deployment คืองานการติดตั้งระบบใหม่แทนที่ระบบเดิม

ในกระแสงานสามตัวสุดท้ายเรียกว่ากระแสงานสนับสนุน(Supporting Workflows) ซึ่งเป็นกระบวนการที่จะต้องทำวนซ้ำไปเรื่อยๆเพื่อให้แน่ใจว่าระบบจะมีความถูกต้องสมบูรณ์และพร้อมใช้งานอยู่เสมอประกอบไปด้วย

1) Configuration Management คืองานการปรับแต่งระบบให้เหมาะสมตามความต้องการที่เปลี่ยนแปลงอยู่เสมอ

2) Project Management คืองานบริหารจัดการโครงการ เช่น การจัดคนให้ใช้งานระบบ การฝึกอบรมฯลฯ)

3) Environment คือการจัดเตรียมสภาพแวดล้อมให้พร้อมทำงาน เช่น จัดห้องเซิร์ฟเวอร์ห้องทำงาน ฯลฯ

ยูนิฟายด์โพรเซสคือกระบวนการทำงานซึ่งจะประกอบด้วยกระแสงานย่อย ๆ ในแต่ละระยะทำงานซึ่งในแต่ละกระแสก็จะมีการใช้เครื่องมือในการวิเคราะห์และออกแบบด้วยแผนภาพยูเอ็มแอลที่แตกต่างกันไป หรือยูนิฟายด์โพรเซสเป็นตัวบอกว่างานใดในช่วงระยะเวลาใดจะใช้แผนภาพใดในยูเอ็มแอลนั่นเอง
ที่มา : https://msit5.wordpress.com/2013/10/14/ความสัมพันธ์ระหว่าง-unified-process-ก/

สรุป

กระบวนการทางวิศวกรรมซอฟต์แวร์มุ่งเน้นการเพิ่มผลผลิตที่ดี กิจกรรมพื้นฐานของกระบวนการทางซอฟต์แวร์ประกอบด้วย 4 ส่วนหลัก ๆ คือ ข้อกำหนดการพัฒนาซอฟต์แวร์ การตรวจสอบความถูกต้องของซอฟต์แวร์ และวิวัฒนาการของซอฟต์แวร์ ระเบียบวิธีการพัฒนาซอฟต์แวร์ หรือ เรียกว่า โมเดลการพัฒนาซอฟต์แวร์ คือแบบจำลอง ที่ใช้สำหรับเป็นตัวชี้นำถึงกิจกรรมหลัก การพัฒนาซอฟต์แวร์ปัจจุบันมีโมเดลการพัฒนาซอฟต์แวร์ ให้ใช้หลายโมเดล โดยเฉพาะโมเดลสมัยใหม่ตามหลักวิศวกรรมซอฟต์แวร์

ยูนิฟายด์โพรเซส คือกระบวนการพัฒนาซอฟต์แวร์ตามหลักการเชิงวัตถุ โดยแบ่งออกเป็น 4 ระยะ ได้แก่ ระยะที่ 1 การเตรียมงาน ระยะที่ 2. การลงรายละเอียด ระยะที่ 3 การจัดสร้าง และ ระยะที่ 4 การถ่ายโอนงาน ตามลำดับ ซึ่งจะเน้นที่การแบ่งหน้าที่ความรับผิดชอบโดยวัตถุประสงค์หลักของยูนิฟายด์โพรเซส คือการที่ให้ได้ซอฟต์แวร์ที่มีคุณภาพและสอดคล้องกับความต้องการ (Requirements) ของผู้ใช้โดยอยู่ภายใต้งบประมาณและเวลาที่สามารถคาดเดาได้ (Predictable Budget and Time) ยูนิฟายด์โพรเซสนั้นจะเน้นการกำหนดบทบาทไปที่ทีมพัฒนางานมากกว่าแต่ละบุคคลกล่าวคือจะมีการกำหนดว่าในแต่ละช่วง (Phase) ของการพัฒนานั้นว่าควรประกอบไปด้วยใคร(Who) แต่ละคนมีหน้าที่รับผิดชอบอะไร (What) จะทำงานที่รับผิดชอบนั้นเมื่อไหร่ (When) และปฏิบัติอย่างไร (How) ที่กล่าวมาเป็นลักษณะที่เป็นนามธรรม (Abstract) หรืออาจกล่าวได้ว่ามันเป็นภาพมุมสูง (Bird Eye View) ของกระบวนการยูนิฟายด์โพรเซสซึ่งอาจจะทำให้เข้าใจภาพยังไม่ชัดเจนกลยุทธ์ที่ใช้ในยูนิฟายด์โพรเซสรวมแล้วจะเรียกมันว่า “Best Practice Model” หรือ “Best Practice” ก็ได้

หากผู้ศึกษาต้องการจะเข้าใจกระบวนการพัฒนาซอฟต์แวร์ตามหลักการเชิงวัตถุ จำเป็นต้องมีความรู้พื้นฐานที่เกี่ยวข้องกับหลักการพื้นฐานต่างเป็นอย่างดี เช่น การให้ความคิดรวบยอด โดเมนของปัญหา แอบแทรคชันแบบต่าง ๆ เป็นต้น เพื่อจะใช้เป็นพื้นฐานความรู้ในบทต่อ ๆ ไป

ชื่อ-นามสกุล	รหัส	สาขาวิชา	รุ่น/หมู่	คะแนน	ลายเซ็นต์อาจารย์

แบบฝึกหัดท้ายบทที่ 3

ข้อ 1 จงตอบคำถามต่อไปนี้

1.1 จงอธิบายความหมายของยูนิฟายด์โพรเซสมาพอสังเขป

.....

.....

.....

1.2 ยูนิฟายด์โพรเซสนั้นมีการแบ่งออกเป็นกี่เฟส

.....

.....

.....

1.3 จงอธิบายรูปลักษณ์ของยูนิฟายด์โพรเซสมาพอสังเขป

.....

.....

.....

1.4 อธิบายกิจกรรมพื้นฐานของกระบวนการทางวิศวกรรมซอฟต์แวร์

.....

.....

.....

1.5 คุณสมบัติของซอฟต์แวร์ที่มีคุณภาพประกอบด้วยอะไรบ้าง

.....

.....

.....

1.6 อธิบายความหมาย วิศวกรรมซอฟต์แวร์

.....

.....

.....

1.7 อธิบายขั้นตอนของ Built-and-Fix Model ประกอบด้วยขั้นตอนอะไรบ้าง

.....

.....

.....

1.8 อธิบายความแตกต่าง Waterfall Model รูปแบบเดิมกับ Waterfall Model แบบทวนซ้ำ

.....

.....

.....

1.9 อธิบายขั้นตอนของ Incremental Model ประกอบด้วยขั้นตอนอะไรบ้าง

.....

.....

.....

1.10 อธิบายส่วนของรอบการทำงานของ Spiral Model ประกอบด้วยอะไรบ้าง

.....

.....

.....

ข้อ 2 จงตอบคำถามต่อไปนี้

2.1จงบอกชิ้นงาน ของแต่ละ worker ต่อไป มาอย่างน้อยคนละ 3 ชิ้นงาน

ผู้ปฏิบัติ(Worker)	ชิ้นงาน(Artifact)
นักวิเคราะห์ระบบ (SA)	
นักพัฒนาระบบ (Programmer)	
ผู้บริหารโครงการ (PM)	
นักออกแบบ UX/UI	
นักทดสอบระบบ (Tester)	

2.2 จงบอกชิ้นงาน ของระบบต่อไปนี้

ระบบงาน	ชิ้นงาน(Artifact)
ระบบสั่งซื้อสินค้า(POS)	
ระบบ เวชระเบียนของโรงพยาบาล	
ระบบ ระบบคาร์แคร์(CarCare)	
ระบบคลังสินค้า (warehouse)	

ข้อ 3 จงตอบคำถามต่อไปนี้

3.1 การแบ่งยูนิตงานให้แต่ละคนเป็นผู้รับผิดชอบ ควรคำนึงถึงสิ่งใดบ้าง

.....

.....

.....

.....

.....

.....

3.2 จงอธิบายรูปลักษณ์ของยูนิฟายด์โพรเซส มาพอสังเขป

.....

.....

.....

.....

.....

.....

3.3 ให้อธิบายหรือวิเคราะห์ระบบการสั่งซื้อสินค้าด้วยหลักการวนรอบเพิ่มพูน (Iterative and Incremental)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

เอกสารอ้างอิง

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

กิตติพงษ์ กลมกล่อม. (2552). *การวิเคราะห์และออกแบบระบบเชิงวัตถุด้วย UML.* กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์.

ฝ่ายผลิตหนังสือตำราวิชาการคอมพิวเตอร์.(2551). *การวิเคราะห์และออกแบบระบบ.* กรุงเทพฯ: ซี เอ็ด ยูเคชั่น.

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

โอภาส เอี่ยมสิริวงศ์. (2555). *การวิเคราะห์และออกแบบระบบ(ฉบับปรับปรุงเพิ่มเติม).* กรุงเทพฯ: ซี เอ็ด ยูเคชั่น.

Clariso, R., Gonzalez, C. and Cabot, J. (2017). Smart Bound Selection for the Verification of UML/OCL Class Diagrams. *IEEE Transactions on Software Engineering, 46*(4), 412-426.

Object Management Group. (1997). *WHAT IS UML.* Retrieved 18 September 2020, from https://www.uml.org/

Shelly, G. and Rosenblatt, H. (2012). *System Analysis and Design,* Ninth Edition. Course Technology.

Wikipedia. (2020). Iterative and incremental development. Retrieved 4 August 2020, from https://en.wikipedia.org/wiki/Iterative_and_incremental_development