System Development Life Cycle : SDLC

วงจรการพัฒนาระบบ

วงจรการพัฒนาระบบ

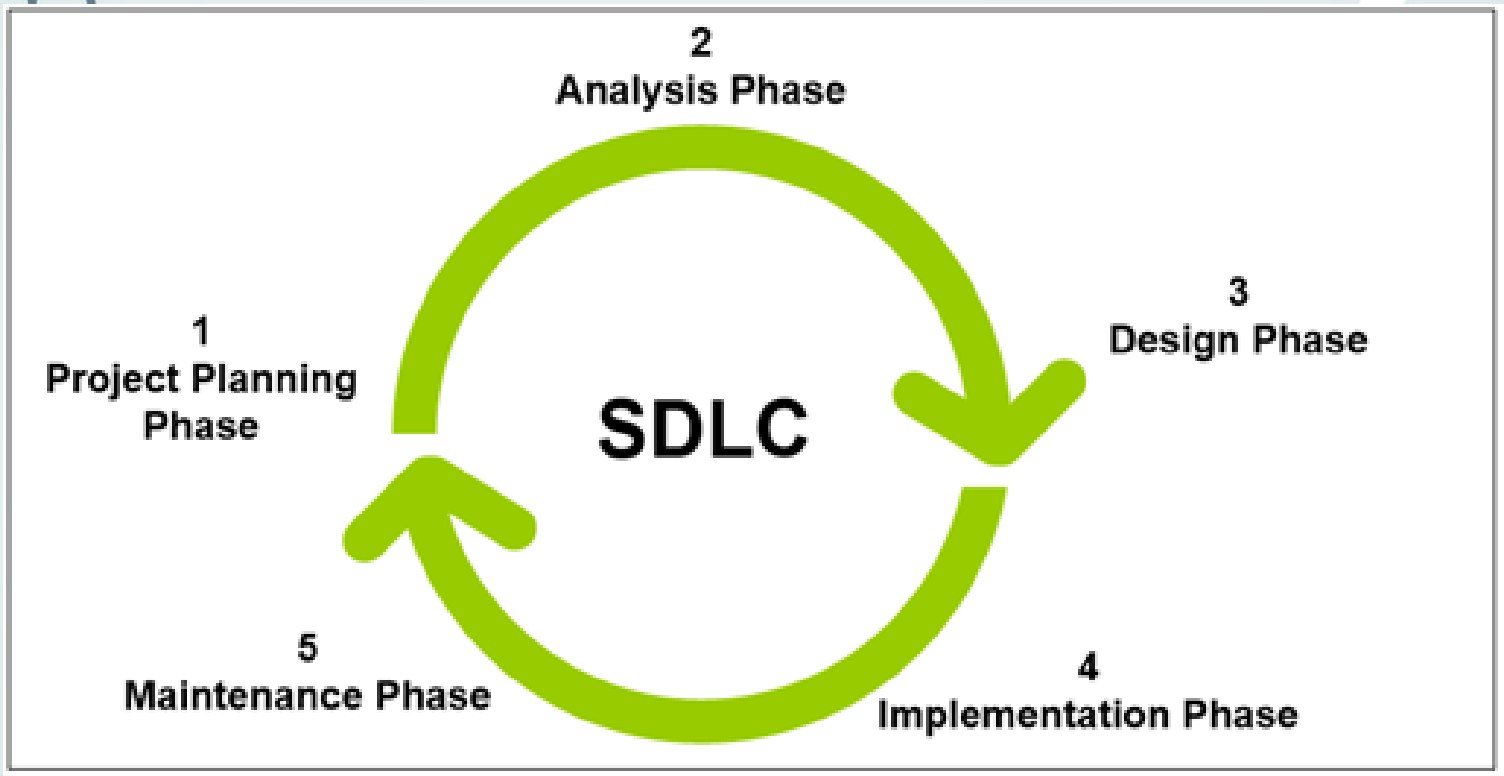
วงจรการพัฒนาระบบหรือที่นิยมเรียกย่อ ๆ ว่า SDLC เป็นวิธีการที่นักวิเคราะห์ระบบใช้ในการพัฒนาระบบงาน

- ใช้สำหรับเรียบเรียงเหตุการณ์หรือกิจกรรม กระทำก่อนหรือกระทำหลัง

- ช่วยให้การพัฒนาระบบงานทำได้ง่ายขึ้น

นักวิเคราะห์ระบบจะต้องทำความเข้าใจให้ชัดเจนถูกต้องว่าในแต่ละขั้นตอนนั้น จะต้องทำอะไร ทำอย่างไร เพื่อได้ผลลัพธ์ตามที่ต้องการ

วงจรการพัฒนาระบบ(ต่อ)



วงจรการพัฒนาระบบ(ต่อ)

โดยปกติแล้ว จะประกอบไปด้วยกลุ่มกิจกรรม 3 ส่วนหลักๆ คือ

1. การวิเคราะห์ (Analysis)
2. การออกแบบ (Design)
3. การนำไปใช้ (Implementation)

ซึ่งกิจกรรมทั้ง 3 ส่วน สามารถใช้งานได้ดีกับโครงการพัฒนาซอฟต์แวร์ขนาดเล็ก

วงจรการพัฒนาระบบ(ต่อ)

ระยะของการพัฒนาระบบ โดยทั่วไป จะประกอบไปด้วย

ระยะที่ 1 : การวางแผนโครงการ (Project Planning Phase)

ระยะที่ 2 : การวิเคราะห์ (Analysis Phase)

ระยะที่ 3 : การออกแบบ (Design Phase)

ระยะที่ 4 : การนำไปใช้ (Implementation Phase)

ระยะที่ 5 : การบำรุงรักษา (Maintenance Phase)

วงจรการพัฒนาระบบ(ต่อ)

ระยะที่ 1 : การวางแผนโครงการ (Project Planning Phase)

จัดเป็นกระบวนการพื้นฐานบนความเข้าใจอย่างถ่องแท้ว่า **ทำไม(Why?)** ต้องสร้างระบบใหม่และกำหนดทีมงานเพื่อสร้างระบบนี้ขึ้นมาได้อย่างไร

- **การเริ่มโครงการ (Project Initiate)** จุดกำเนิดของระบบงาน มักเกิดขึ้นจากผู้ใช้งานระบบซึ่งเป็นผู้คลุกคลีกและปฏิบัติงานกับระบบโดยตรง โดยผู้วิเคราะห์ระบบ จะต้องศึกษาถึงขอบเขตปัญหาที่ผู้ใช้ กำลัง**ประสบปัญหาอยู่** และจะดำเนินการ **แก้ไขอย่างไร** ศึกษาความเป็นไปได้ ว่าระบบที่จะพัฒนาขึ้นมาใหม่ **มีความเป็นไปได้และคุ้มค่าที่จะลงทุนหรือไม่**

วงจรการพัฒนาระบบ(ต่อ)

ระยะของการวางแผนโครงการ จะประกอบไปด้วยกิจกรรมต่าง ๆ ดังนี้

1. กำหนดปัญหา (Problem Definition) ระบบสารสนเทศจะเกิดขึ้นได้ก็ต่อเมื่อผู้บริหารหรือผู้ใช้ตระหนักว่า ต้องการระบบสารสนเทศหรือระบบจัดการเดิม **ไม่มีประสิทธิภาพเพียงพอ**ที่ตอบสนองความต้องการในปัจจุบัน ปัญหาที่สำคัญของระบบสารสนเทศในปัจจุบัน คือ ระบบเขียนมานานแล้ว ส่วนใหญ่เขียนมา**เพื่อติดตามเรื่องการเงิน** ไม่ได้มีจุดประสงค์เพื่อให้ข้อมูล**ข่าวสารในการตัดสินใจ** แต่ปัจจุบันฝ่ายบริหารต้องการดูสถิติการขายเพื่อใช้ในการคาดคะเนในอนาคต หรือความต้องการอื่น ๆ เช่น สินค้าที่มียอดขายสูง หรือสินค้าที่ลูกค้าต้องการสูง หรือการแยกประเภทสินค้าต่าง ๆ ที่ทำได้ไม่ง่ายนัก

วงจรการพัฒนาระบบ(ต่อ)

2. ศึกษาความเป็นไปได้ของโครงการ (Feasibility Study) การกำหนดว่าปัญหาคืออะไรและตัดสินใจว่าการพัฒนาสร้างระบบสารสนเทศ หรือการแก้ไขระบบสารสนเทศเดิมมีความเป็นไปได้หรือไม่ โดยเสียค่าใช้จ่ายและเวลาน้อยที่สุด จากนั้น นักวิเคราะห์ระบบก็จะนำค่าใช้จ่ายและผลประโยชน์(Cost-Benefit) มาดำเนินการดังนี้

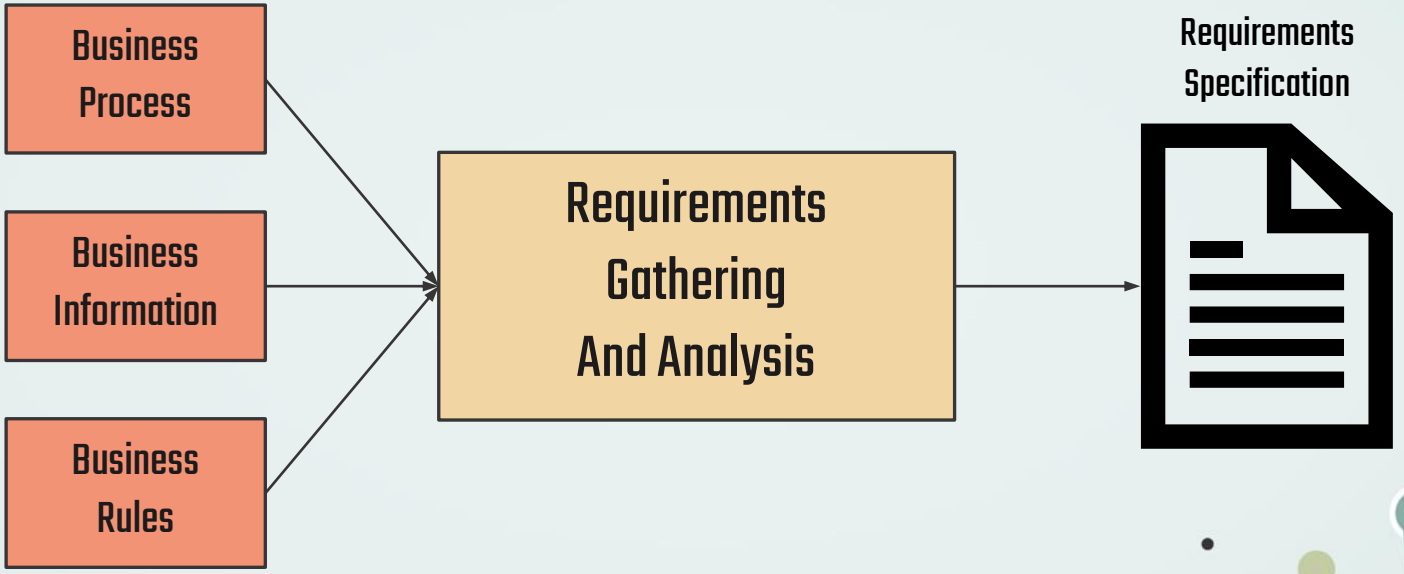
- **จัดทำตารางกำหนดเวลาโครงการ (Project scheduling)**
- **จัดตั้งทีมงานโครงการ (Staff the project)**
- **ดำเนินโครงการ (Launch the project)**

วงจรการพัฒนาระบบ(ต่อ)

ระยะที่ 2 : การวิเคราะห์ (Analysis Phase)

ระยะการวิเคราะห์ จะต้องตอบคำถามให้ได้ก่อนว่า ใครเป็นผู้ใช้ระบบ (Who?) มีอะไรบ้างที่จะต้องทำ (What?) จะต้องทำที่ไหน (Where?) และ เมื่อไร (When?) โดยทีมงานจะศึกษาระบบเดิมว่าทำงานอย่างไร หรือ ธุรกิจดำเนินอย่างไร และกำหนดแนวทางในการปรับปรุงกระบวนการทำงานให้ดียิ่งขึ้น และกำหนดความต้องการของระบบใหม่

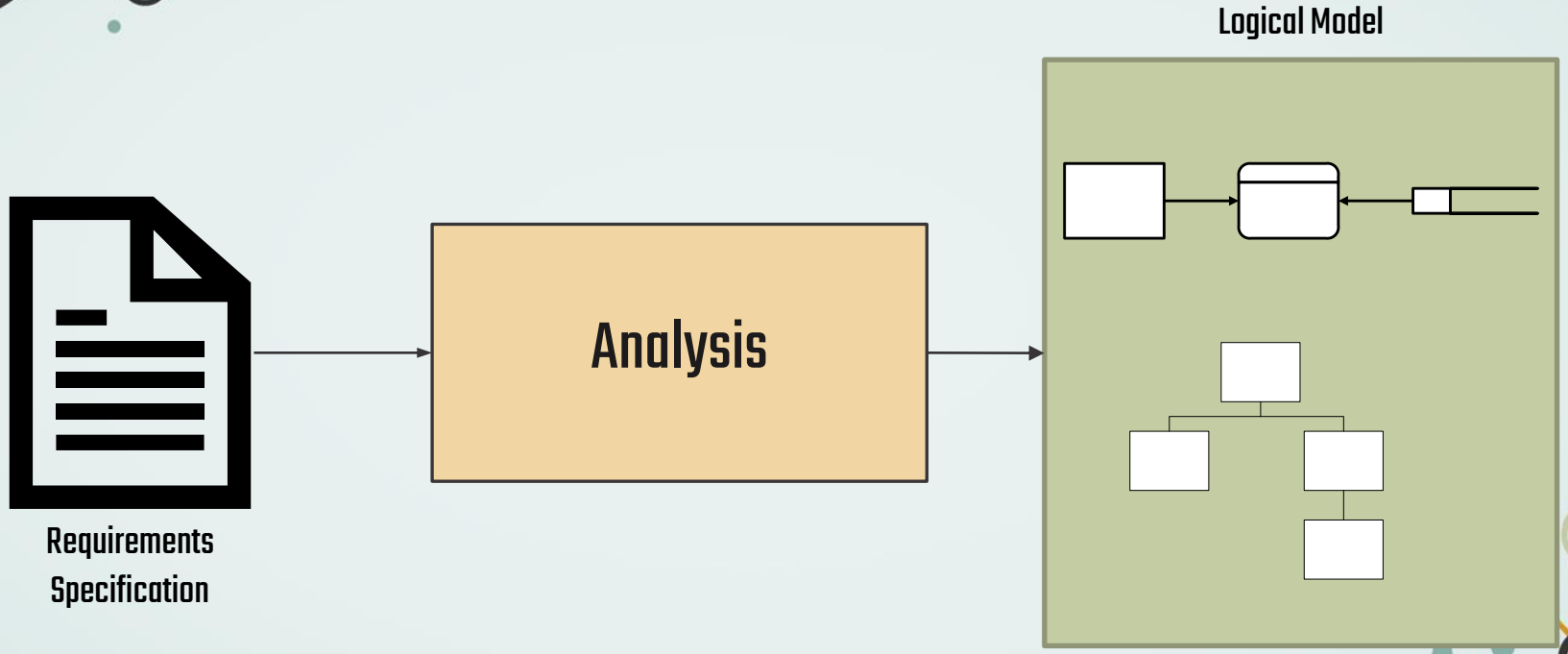
วงจรการพัฒนาระบบ(ต่อ)



วงจรการพัฒนาระบบ(ต่อ)

สิ่งสำคัญของระยะการวิเคราะห์ คือ **การรวบรวมความต้องการ (Requirements Gathering)** นักวิเคราะห์ฯสามารถรวบรวมความต้องการต่างๆ ได้จาก การสังเกต การสัมภาษณ์ การจัดทำแบบสอบถาม อ่านเอกสารเกี่ยวกับการปฏิบัติงานและระเบียบ กฎเกณฑ์ต่างๆของบริษัท รวมถึง ต้องพบปะผู้ใช้ระดับต่างๆ เพื่อให้ทราบกระบวนการทำงาน ปัญหาและแนวทางในการแก้ไขปัญหาจากผู้ใช้งาน จากนั้นนำมาสรุปเป็นข้อกำหนดความต้องการ ที่ผู้เกี่ยวข้องอ่านแล้วต้องเข้าใจความหมายตรงกัน

วงจรการพัฒนาระบบ



วงจรการพัฒนาระบบ(ต่อ)

สรุประยะของการวิเคราะห์ จะประกอบไปด้วยกิจกรรมต่าง ๆ ดังต่อไปนี้

1. วิเคราะห์ระบบงานปัจจุบัน
2. รวบรวมความต้องการในด้านต่าง ๆ และนำมาวิเคราะห์เพื่อสรุปเป็นข้อกำหนดที่ชัดเจน
3. นำข้อกำหนดมาพัฒนาออกมาเป็นความต้องการของระบบใหม่
4. สร้างแบบจำลองกระบวนการของระบบใหม่ด้วยการวาดแผนภาพ แสดงการไหลของข้อมูล (Data Flow Diagram: DFD)
5. สร้างแบบจำลองข้อมูล ด้วยการวาดอีอาร์ไดอะแกรม (Entity Relationship Diagram: ERD)

วงจรการพัฒนาระบบ(ต่อ)

ระยะที่ 3 : การออกแบบ (Design Phase)

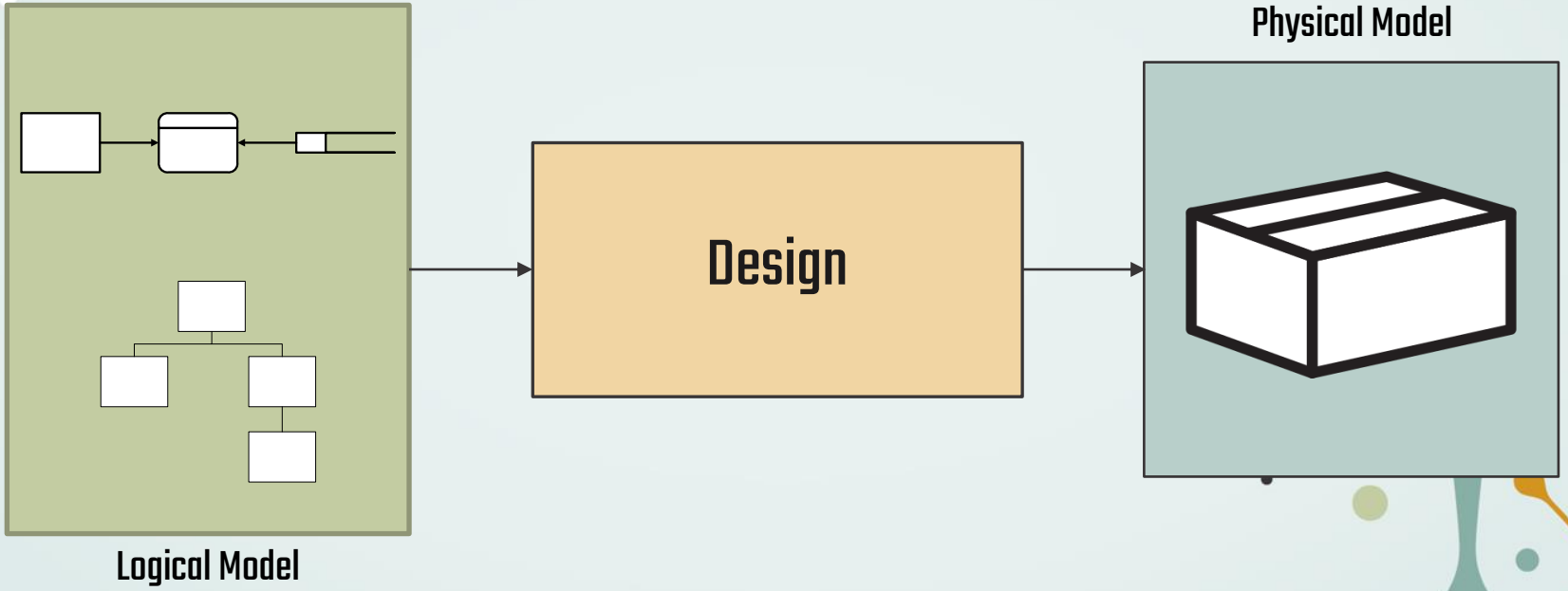
ระยะการออกแบบ เป็นการตัดสินใจว่าระบบจะดำเนินการอย่างไร (How?)

- **กลยุทธ์การจัดหาระบบ** เพื่อได้ความชัดเจนเกี่ยวกับแนวทางการพัฒนาระบบ ว่าจะพัฒนาเองหรือ ซื้อซอฟต์แวร์มาใช้ หรือจ้างหน่วยงานภายนอกมาพัฒนาระบบ
- **ออกแบบสถาปัตยกรรมของระบบ** ที่อธิบายถึง ฮาร์ดแวร์ ซอฟต์แวร์และโครงสร้างพื้นฐาน เช่น ด้านเครือข่าย จะมีการเพิ่มเติมหรือปรับปรุงจากโครงสร้างพื้นฐานเดิมที่มีอยู่
- **การออกแบบอินเตอร์เฟซ** เกี่ยวข้องกับการปฏิสัมพันธ์ระหว่างผู้ใช้กับระบบ รวมถึงฟอร์มและรายงานต่างๆ

วงจรการพัฒนาระบบ(ต่อ)

- **ออกแบบฐานข้อมูล** เพื่อให้ทราบว่าระบบต้องมีข้อมูลอะไรบ้าง ที่ต้องจัดเก็บในฐานข้อมูล
- **การออกแบบโปรแกรม** เพื่อนำไปสู่การเขียนโปรแกรมด้วยคอมพิวเตอร์ในระยะการนำไปใช้งานต่อไป

วงจรการพัฒนาระบบ(ต่อ)



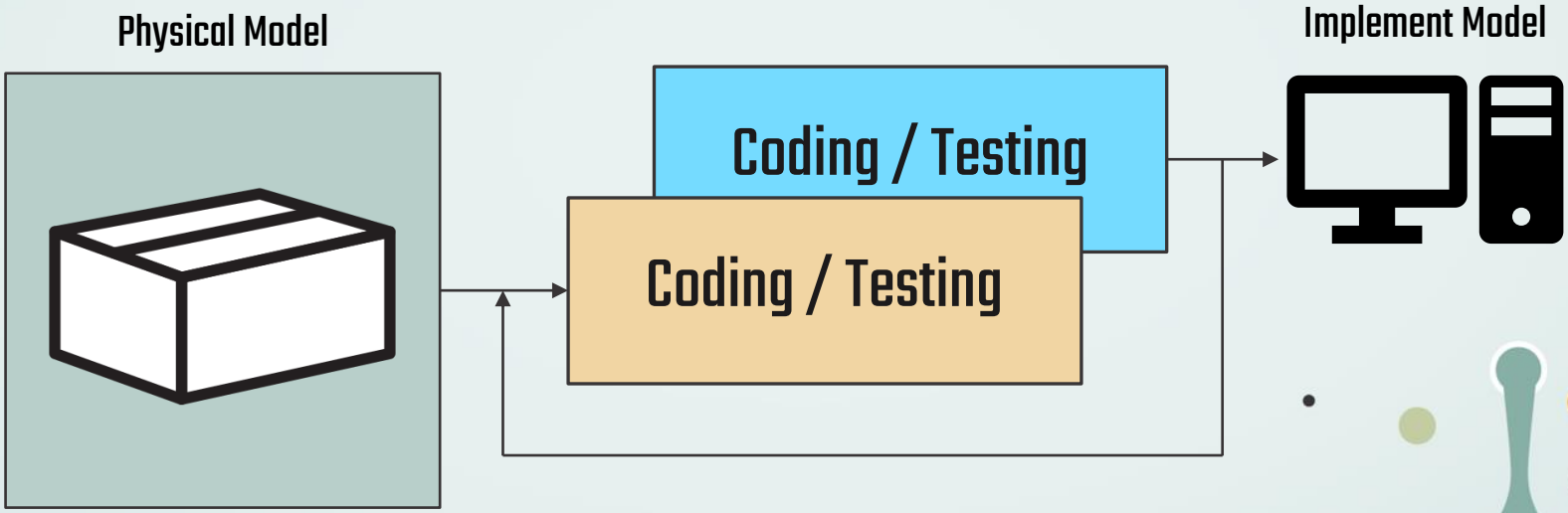
วงจรการพัฒนาระบบ(ต่อ)

สรุประยะของการออกแบบ จะประกอบไปด้วยกิจกรรมต่าง ๆ ดังต่อไปนี้

1. พิจารณาแนวทางในการพัฒนาระบบ
2. ออกแบบสถาปัตยกรรมระบบ (Architecture Design)
3. ออกแบบฐานข้อมูล (Database Design)
4. ออกแบบโปรแกรม (Structure Chart)
5. ออกแบบเอาต์พุต (Output Design)
6. ออกแบบอินพุต (Input Design)
7. ออกแบบยูสเซอร์อินเตอร์เฟซ (User Interface Design)
8. จัดทำต้นแบบ (Prototype)

วงจรการพัฒนาระบบ(ต่อ)

ระยะที่ 4 : การนำไปใช้ (Implementation Phase)



วงจรการพัฒนาระบบ(ต่อ)

สรุประยะของการนำไปใช้ จะประกอบไปด้วยกิจกรรมต่าง ๆ ดังต่อไปนี้

1. สร้างระบบขึ้นมาด้วยการเขียนโปรแกรม
2. ตรวจสอบความถูกต้องทั้งทางด้าน Verification และ Validation และดำเนินการทดสอบระบบ
3. แปลงข้อมูล (Data Convert)
4. ติดตั้งระบบ (System Installation) และจัดทำเอกสารคู่มือ
5. ฝึกอบรมผู้ใช้ และประเมินผลระบบใหม่

วงจรการพัฒนาระบบ(ต่อ)

ระยะที่ 5 : การบำรุงรักษา (Maintenance Phase)

ได้แก่ การแก้ไขโปรแกรมหลังจากการใช้งานแล้ว สาเหตุที่ต้องแก้ไขโปรแกรมหลังจากใช้งานแล้ว สาเหตุที่ต้องแก้ไขระบบส่วนใหญ่มี 2 ข้อ คือ

1. มีปัญหาในโปรแกรม **(Bug)**
2. การดำเนินงานในองค์กรหรือธุรกิจ**เปลี่ยนไป** (Business System)

จากสถิติของระบบที่พัฒนาแล้วทั้งหมดประมาณร้อยละ 40 ของค่าใช้จ่ายในการแก้ไขโปรแกรม เนื่องจากมี "**Bug**" ดังนั้นนักวิเคราะห์ระบบควรให้ความ สำคัญกับการบำรุงรักษา ซึ่งปกติจะคิดว่าไม่มีความสำคัญมากนัก

วงจรการพัฒนาระบบ(ต่อ)

สรุประยะของการบำรุงรักษา จะประกอบไปด้วยกิจกรรมต่าง ๆ ดังต่อไปนี้

1. การบำรุงรักษาระบบ (System Maintenance)
2. การเพิ่มเติมคุณสมบัติใหม่ ๆ เข้าไปในระบบ (Enhance the system)
3. การสนับสนุนงานของผู้ใช้ (Support the Users)

โมเดลการพัฒนาซอฟต์แวร์

แบบจำลองที่แสดงให้เห็นถึงกิจกรรมหลัก (Key Activities) ของกระบวนการพัฒนาซอฟต์แวร์ ด้วยการกำหนดรายละเอียดหรือข้อบัญญัติไว้ในแต่ละกิจกรรมในแต่ละขั้นตอนที่มีลำดับขั้นตอนการพัฒนาที่ชัดเจน เพื่อต้องการให้การพัฒนาซอฟต์แวร์ดำเนินต่อไปให้เกิดปัญหาน้อยที่สุด

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

ข้อควรคำนึงในการเลือกโมเดลการพัฒนาซอฟต์แวร์

ปัจจุบันมีโมเดลในการพัฒนาซอฟต์แวร์ต่างๆ มากมาย โดยโมเดลสมัยใหม่มักจะผนวกขั้นตอนหรือกระบวนการที่สามารถทำงานในลักษณะ **การทวนซ้ำเป็นรอบ (Iteration), การพัฒนาแบบก้าวหน้า (Incremental)** และ**การจัดทำต้นแบบ (Prototyping)** ซึ่งการทำงานในลักษณะดังกล่าว จะช่วยลดความเสี่ยงลงได้มากในกรณีที่โครงการซอฟต์แวร์นั้นมีการเปลี่ยนแปลงในด้านของความต้องการอยู่ตลอดเวลา รวมถึงความต้องการให้ซอฟต์แวร์ที่สร้างขึ้นมานั้นตรงกับความต้องการของผู้ใช้ให้มากที่สุด

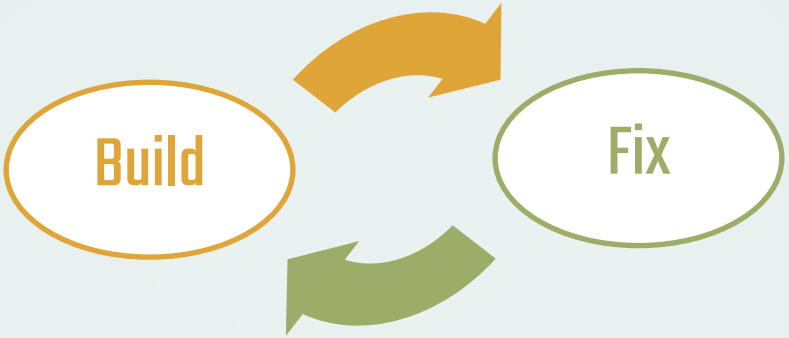
ส่วนการตัดสินใจว่าจะเลือกใช้โมเดลใดมาใช้กับการพัฒนาซอฟต์แวร์ ก็จะขึ้นอยู่กับปัจจัยหลาย ๆ อย่างด้วยกัน ไม่ว่าจะเป็นเรื่องของ **“ขนาดของโครงการ ความซับซ้อน ความเหมาะสม และระดับความเสี่ยง”** เป็นต้น

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

สาเหตุสำคัญที่จำเป็นต้องใช้โมเดลการพัฒนาซอฟต์แวร์

1. โมเดลการพัฒนาซอฟต์แวร์จะมีการแตกขั้นตอนของกระบวนการพัฒนาออกเป็นระยะ (Phase)
2. ซอฟต์แวร์ที่พัฒนามีความซับซ้อนมากขึ้น
3. การแบ่งเป็นกระบวนการพัฒนาเป็นเฟสหรือเป็นระยะ จะทำให้ง่ายต่อการจัดการ
4. แต่ละระยะมีแนวทางต่าง ๆ ให้เลือกปฏิบัติ

Build and Fix Model



โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

Built-and-Fix Model

จัดได้ว่าเป็นโมเดลหนึ่งที่เก่าแก่ที่สุด เป็นโมเดลการพัฒนาซอฟต์แวร์ที่ว่าด้วยการ **“เขียนโปรแกรมและแก้ไขปรับปรุงโปรแกรมไปเรื่อย ๆ ลองผิดลองถูกไปจนกระทั่งคิดว่าพอใจหรือคิดว่าตรงกับความต้องการ”** ซึ่งกระบวนการดังกล่าวจะทำให้สูญเสียเวลาไปกับการดีบั๊กโปรแกรม และบำรุงรักษาระบบ

เหมาะสมกับ **“โปรแกรมขนาดเล็ก”** ที่ **“ไม่มีความซับซ้อน”** หรือ **“เหมาะกับงานที่เมื่อเกิดข้อผิดพลาดแล้วไม่ส่งผลกระทบต่อระบบมากนัก”**

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

สำหรับขั้นตอนของโมเดล Built-and-Fix นี้ประกอบด้วย

1. เขียนโค้ดหรือโปรแกรมบางส่วนที่คาดหวังว่าจะสามารถแก้ไขปัญหาโจทย์เหล่านั้นได้
2. คอมไพล์โปรแกรม และทำการรันโปรแกรม
3. หากพบข้อผิดพลาดในโปรแกรม ก็ดำเนินการแก้ไขปรับปรุง
4. กลับไปทำขั้นตอนที่ 1 - 4 ซ้ำจนกระทั่งมีความรู้สึกว่าดีเพียงพอแล้ว

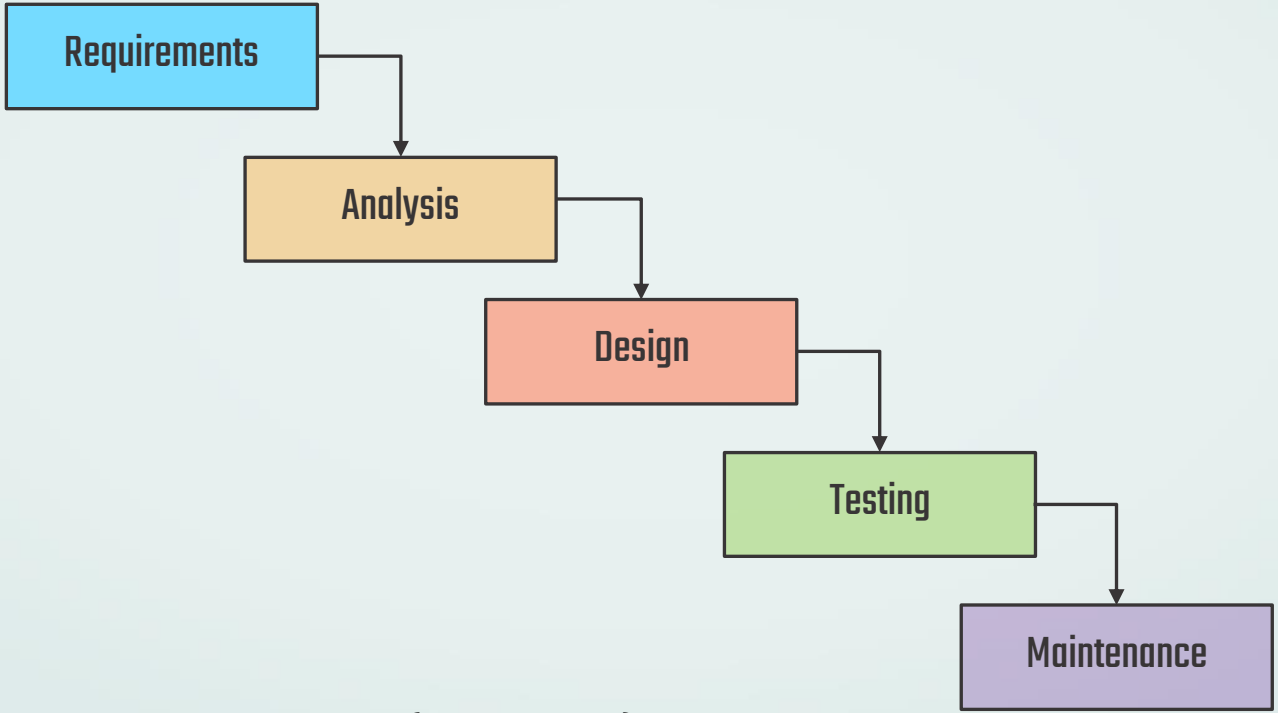
โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

Water Fall Model

หรือเรียกว่าโมเดลน้ำตก เป็นโมเดลที่มีมานานได้เผยแพร่ใช้งานเมื่อราวปี ค.ศ. 1970 และกว่า 30 ปี แล้วที่โมเดลนี้ยังเป็นที่นิยมใช้ในการพัฒนาระบบงานจนถึงปัจจุบันเนื่องจากเป็นโมเดลที่ง่ายต่อการนำไปประยุกต์ใช้ ด้วยการเริ่มต้นจากการรวบรวมความต้องการ การวิเคราะห์ การออกแบบ การเขียนโปรแกรม การทดสอบ และการบำรุงรักษา

มีความคล้ายคลึงกันกับวงจรการพัฒนาระบบตามแนวทางของ SDLC โดยจะเห็นได้ว่ากระบวนการพัฒนาซอฟต์แวร์ตามโมเดลแบบน้ำตกแบบดั้งเดิมนั้น เมื่อมีการเข้าสู่ขั้นตอนใด ๆ แล้วจะไม่มีการย้อนกลับมาทำขั้นตอนก่อนหน้านั้นได้อีก

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)



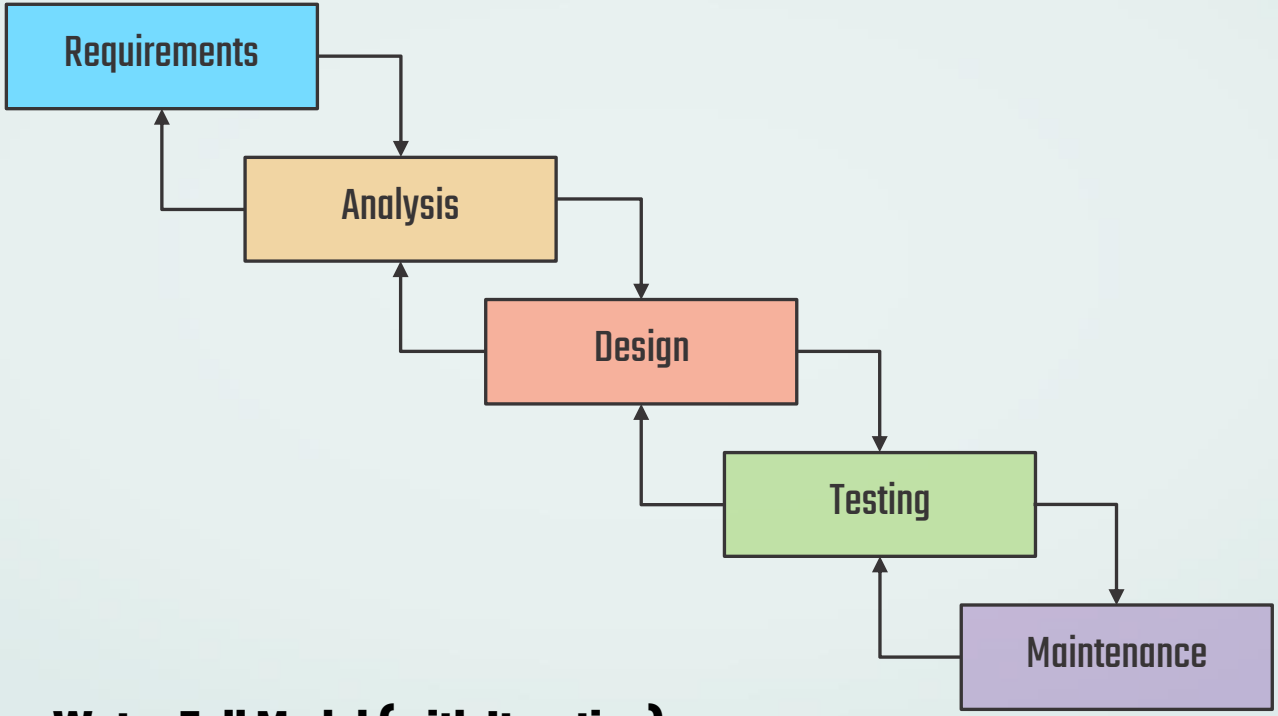
Water Fall Model (Traditional)

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

แต่ในความเป็นจริง เหตุการณ์การย้อนกลับไปทำขั้นตอนก่อนหน้านั้นสามารถเกิดขึ้นได้ เช่น นักวิเคราะห์อาจมองเห็นปัญหาหรือข้อผิดพลาดที่เกิดขึ้น หรือมีบางอย่างจำเป็นต้องได้รับการเปลี่ยนแปลง หรือแก้ไขอย่างเร่งด่วน

จึงได้มีการปรับปรุงโมเดลนี้ใหม่ด้วยการผนวกคุณสมบัติแบบทวนซ้ำเป็นรอบได้ (Iteration)

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)



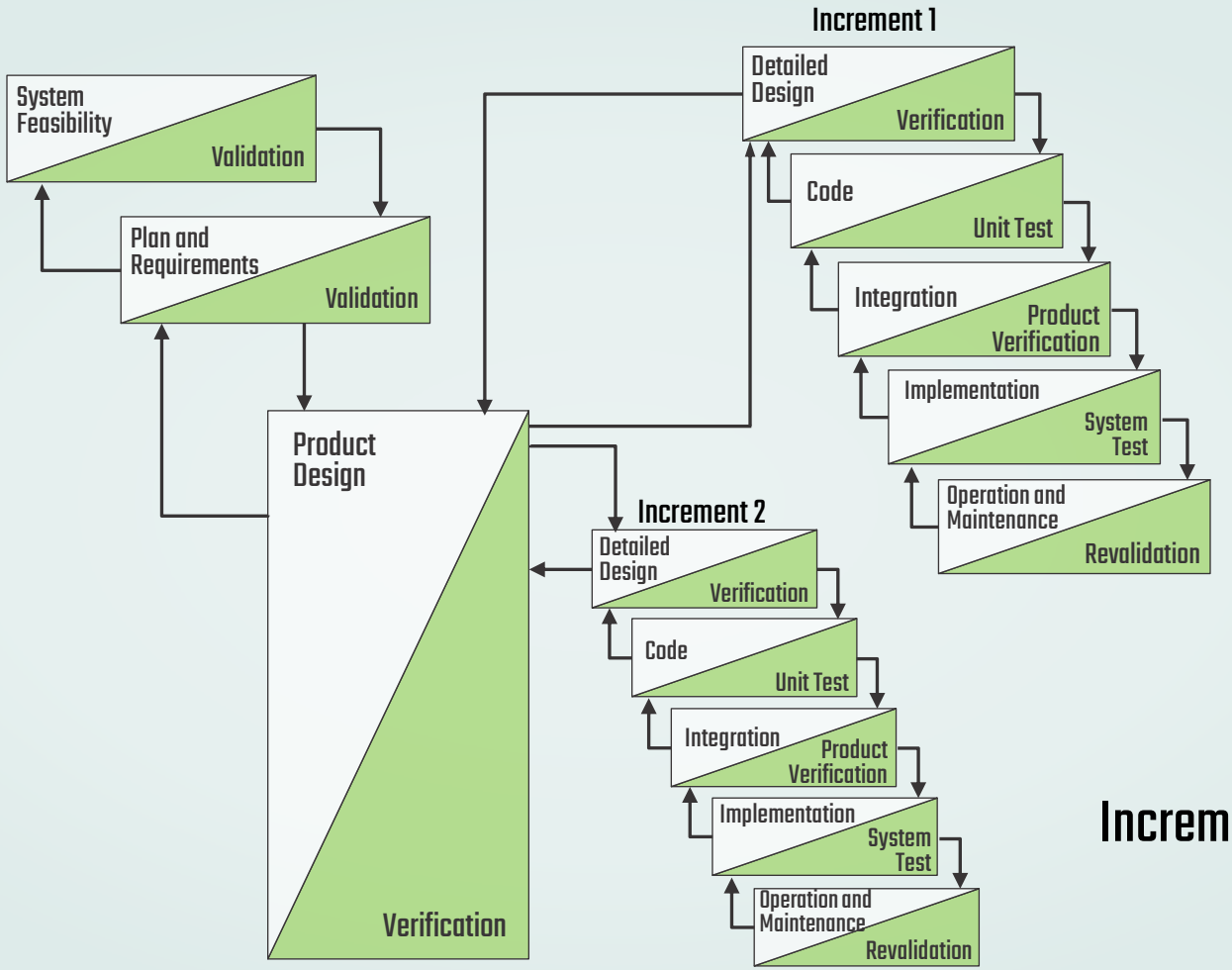
Water Fall Model (with Iteration)

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

Incremental Model

โมเดลแบบก้าวหน้า เป็นโมเดลหนึ่งที่ได้นำหลักการของ Water Fall Model มาปรับปรุงประสิทธิภาพให้ดียิ่งขึ้น ดังที่กล่าวไว้แล้วว่า Water Fall Model จะมี**ข้อเสีย**ตรงที่ **“มีกระบวนการทดสอบอยู่ตอนท้าย ๆ”** ซึ่งมีโอกาสที่จะย้อนกลับไปเริ่มต้นโครงการใหม่ทั้งหมด หากมีการวางแผนจัดการที่ไม่ดีพอ

Incremental Model มีการเพิ่มส่วนของการออกแบบและพัฒนาในรูปแบบของ**ส่วนงานย่อย**ในลักษณะแบบก้าวหน้า (Increment) **“โดยแต่ละส่วนงานย่อยจะมีการทวนซ้ำเป็นรอบ”** ในลักษณะ Iteration พร้อมกับมีระบบการตรวจสอบ



Incremental Model

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

เมื่อโครงสร้างหลักของระบบ ได้รับการออกแบบเป็นที่เรียบร้อยแล้ว ก็จะแบ่งส่วนงานออกเป็นระบบย่อย ๆ โดยการพัฒนาในแต่ละ **Increment** จะประกอบไปด้วยขั้นตอนต่างๆดังนี้ (ซึ่งแต่ละขั้นตอนสามารถกลับไปทวนซ้ำได้)

1. ในขั้นตอนเริ่มต้นของการออกแบบ (Detailed Design) ในแต่ละรอบ จะมีการตรวจสอบความถูกต้องตามรายละเอียดของข้อกำหนด (Verification)

2. ในขั้นตอนของการเขียนโปรแกรม (Code) จะมีการทดสอบหน่วยย่อยของโปรแกรม (Unit Test)

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

3. ในขั้นตอนของการนำโปรแกรมหน่วยย่อยต่าง ๆ มาประกอบรวมกัน (Integration) จะมีการทดสอบความถูกต้องทางด้านรายละเอียดของข้อกำหนดในตัวผลิตภัณฑ์ (Product Verification)

4. ในขั้นตอนของการนำไปใช้ (Implementation) จะมีการทดสอบระบบ (System Test)

5. ในขั้นตอนของการปฏิบัติงานและบำรุงรักษา (Operation and Maintenance) จะมีการทบทวนความต้องการของผู้ใช้อีกครั้ง (Revalidation) ว่าตรงกับวัตถุประสงค์หรือไม่

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

ความหมายของ Verification และ Validation

- **Verification** เป็นการตรวจสอบความถูกต้องตามข้อกำหนด (Specification) หรือความพยายามค้นหาข้อผิดพลาดจากการประมวลผลโปรแกรม
- **Validation** เป็นการตรวจสอบรายละเอียดของผลิตภัณฑ์ว่าตรงตามความต้องการของผู้ใช้หรือไม่

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

ความหมายของ Iteration และ Incremental

- **การทวนซ้ำเป็นรอบ (Iteration)** คือ การทวนซ้ำของงาน ซึ่งมีความเป็นไปได้ว่า การพัฒนางานใด ๆ ก็ตาม อาจมีความจำเป็นต้องมีการทวนซ้ำของงานมากกว่าหนึ่งรอบ เพื่อให้งานนั้นตรงกับความต้องการมากที่สุด การทวนซ้ำมากกว่าหนึ่งรอบนั้นถือเป็นการทบทวนความถูกต้องและลดความเสี่ยง และที่สำคัญ การทวนซ้ำในแต่ละรอบจะหมายถือการแก้ไขปรับปรุงให้ดีขึ้น (Refined)

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

ความหมายของ Iteration และ Incremental

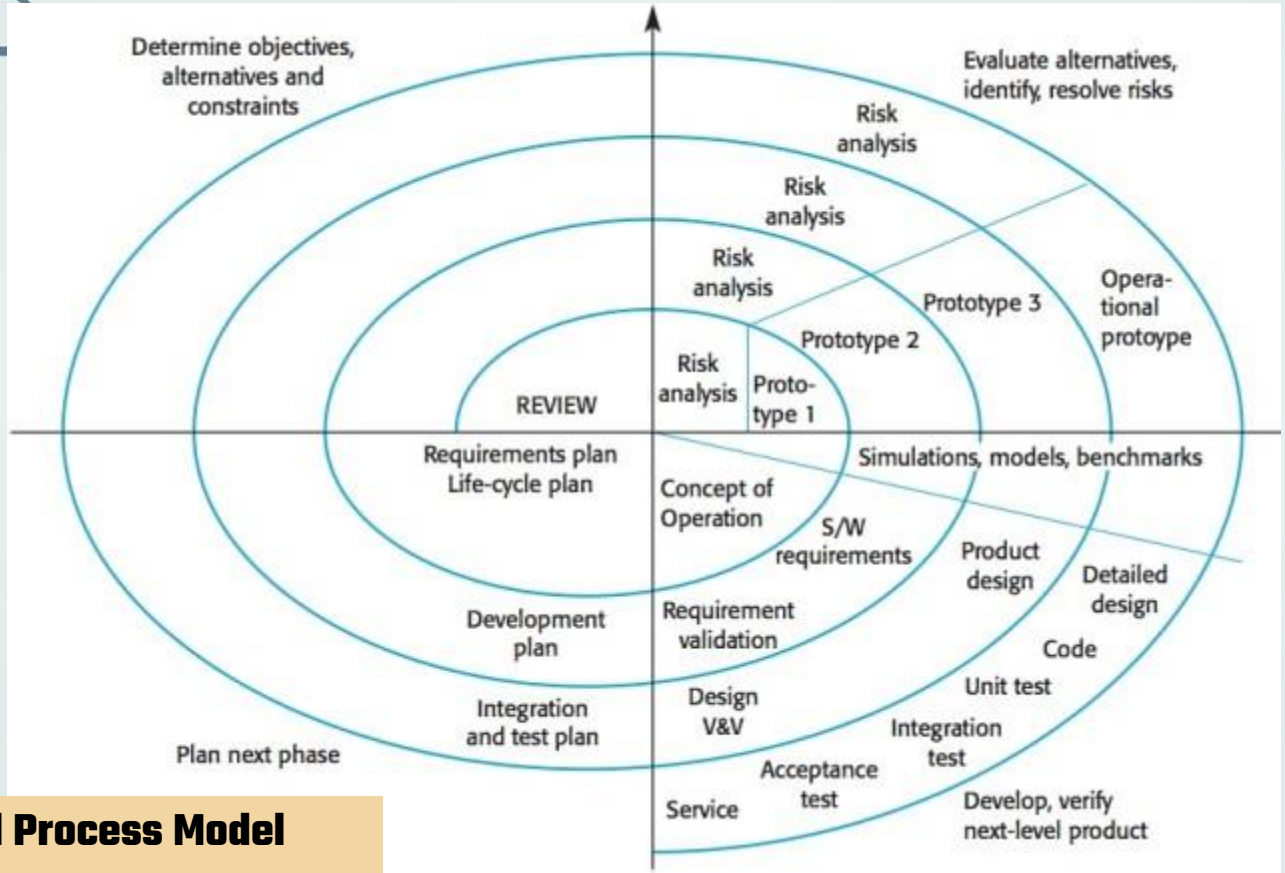
- **การพัฒนาแบบก้าวหน้า (Incremental)** คือ วิธีการพัฒนาส่วนย่อยของระบบให้สมบูรณ์ โดยแต่ละระบบย่อยหรือแต่ละส่วนที่พัฒนานั้นอาจมีจำนวนการทวนซ้ำ (Iteration) หลาย ๆ รอบในส่วนงานย่อยนั้น ๆ จากนั้นก็ทำการเพิ่มระดับความก้าวหน้า (Increment) ในส่วนงานที่ทำสำเร็จ และท้ายสุดก็จะนำส่วนงานย่อย ๆ ที่เสร็จสมบูรณ์เหล่านั้นมาประกอบรวมกัน (Integrated) ให้เป็นหนึ่งระบบที่สมบูรณ์

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

Spiral Model

เป็นโมเดลหนึ่งที่มีหลักการทำงานเป็นลักษณะรอบวงกลม วนเป็นก้นหอย ซึ่งจะวนตามเข็มนาฬิกา ด้วยการวนไปเรื่อย ๆ จากวงในไปสู่วงนอก เป็นวิธีการพัฒนาแบบค่อยเป็นค่อยไปตามจำนวนรอบ การทำงานเสร็จสิ้นในแต่ละรอบ หมายถึงการได้ผลงานเพิ่มมากขึ้น และที่สำคัญ ในแต่ละรอบจะมีการวิเคราะห์ความเสี่ยง เพื่อประเมินและวางแผนการทำงานในรอบถัดไป

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)



The Spiral Process Model

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

โดยแต่ละวงรอบนั้นจะประกอบด้วยขั้นตอนย่อยที่สำคัญดังนี้ คือ

1. การวิเคราะห์ความต้องการ (Requirement Analysis)
2. การวิเคราะห์ความเสี่ยง (Risk Analysis)
3. การออกแบบต้นแบบ (Design Prototype)
4. การพัฒนาต้นแบบและการนำมาประกอบรวมกัน (Develop and Integrate Prototype)

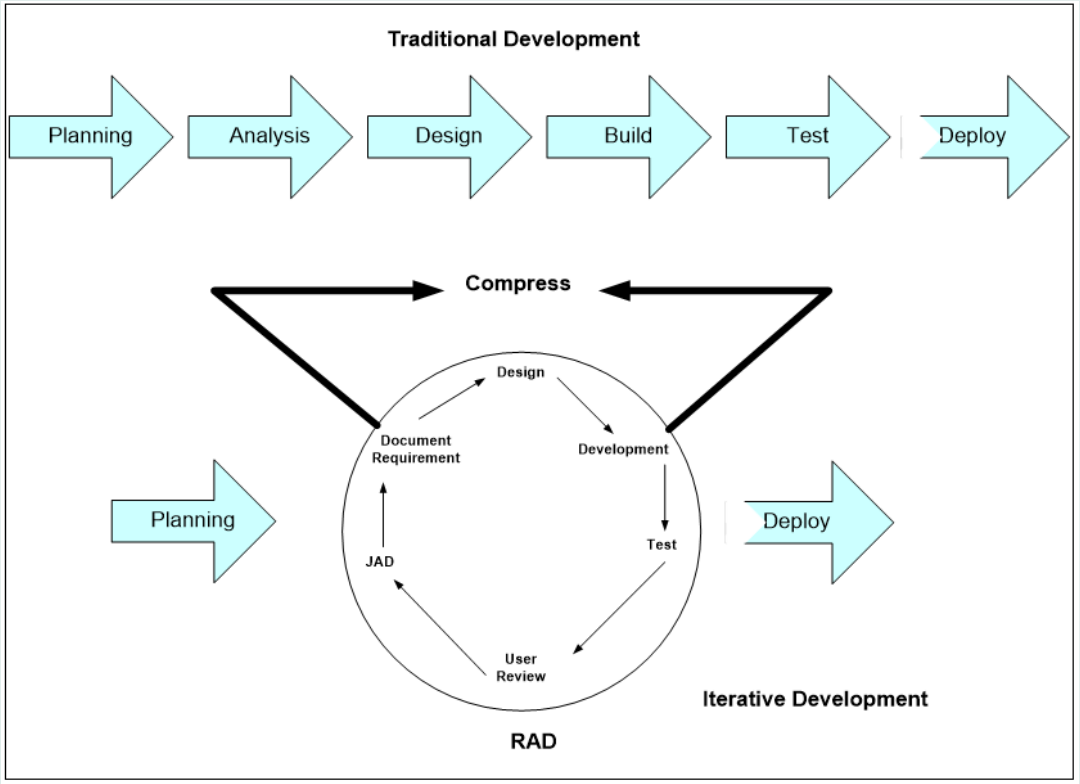
โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

Rapid Application Development (RAD)

RAD คือ วิธีการการพัฒนาระบบแบบรวดเร็ว ซึ่งมักใช้เครื่องมือสนับสนุนอย่าง CASE Tools ช่วยในการพัฒนา จึงส่งผลระบบที่พัฒนาด้วยเทคนิค RAD นั้นใช้เวลาอันสั้น ด้วยการมุ่งเน้นด้านการลดต้นทุนและระยะเวลาในการพัฒนา

RADจัดเป็นกรรมวิธีการพัฒนาซอฟต์แวร์ที่ย่นระยะเวลาของขั้นตอนการวิเคราะห์ (Analysis) การออกแบบ (Design) การสร้าง (Build) และการทดสอบ(Testing) เพื่อจะได้ลดเวลาในการพัฒนาโดยรวมลงได้ ดังนั้น เพื่อความคล่องตัวจึงจำเป็นต้องมี**ทีมงานขนาดเล็ก**ที่มีความรู้ความสามารถเฉพาะซึ่งประกอบด้วย ผู้เชี่ยวชาญด้านเทคโนโลยีสารสนเทศกับกลุ่มผู้ใช้งานร่วมกันพัฒนา โดย RAD นี้ถือเป็นการจัดการโครงการ (Project Management) อย่างง่ายชนิดหนึ่ง

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)



โมเดลแสดงลำดับกระบวนการของ **RAD**

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

เทคนิคการพัฒนาระบบแบบ RAD ที่มีการเร่งรัดเวลาในเรื่องของวิเคราะห์ระบบ จึงอาจทำให้เกิดผลเสียต่อความยืดหยุ่นของระบบที่อาจจำเป็นต้องมีการเปลี่ยนแปลงในอนาคตก็เป็นได้ กล่าวคือ ระบบอาจไม่สามารถรองรับการเปลี่ยนแปลงในอนาคตได้นั่นเอง

เทคนิคสำคัญของการพัฒนาซอฟต์แวร์ตามกรรมวิธีของ RAD

1. พัฒนาต้นแบบได้อย่างรวดเร็ว
2. เป็นแหล่งรวมเครื่องมือเพื่อการพัฒนาระบบมากมาย
3. มีทีมงานที่เชี่ยวชาญการใช้เครื่องมือเหล่านั้น
4. เป็นแนวร่วมปฏิบัติการกับ JAD
5. มีกรอบระยะเวลาการพัฒนาที่จำกัด

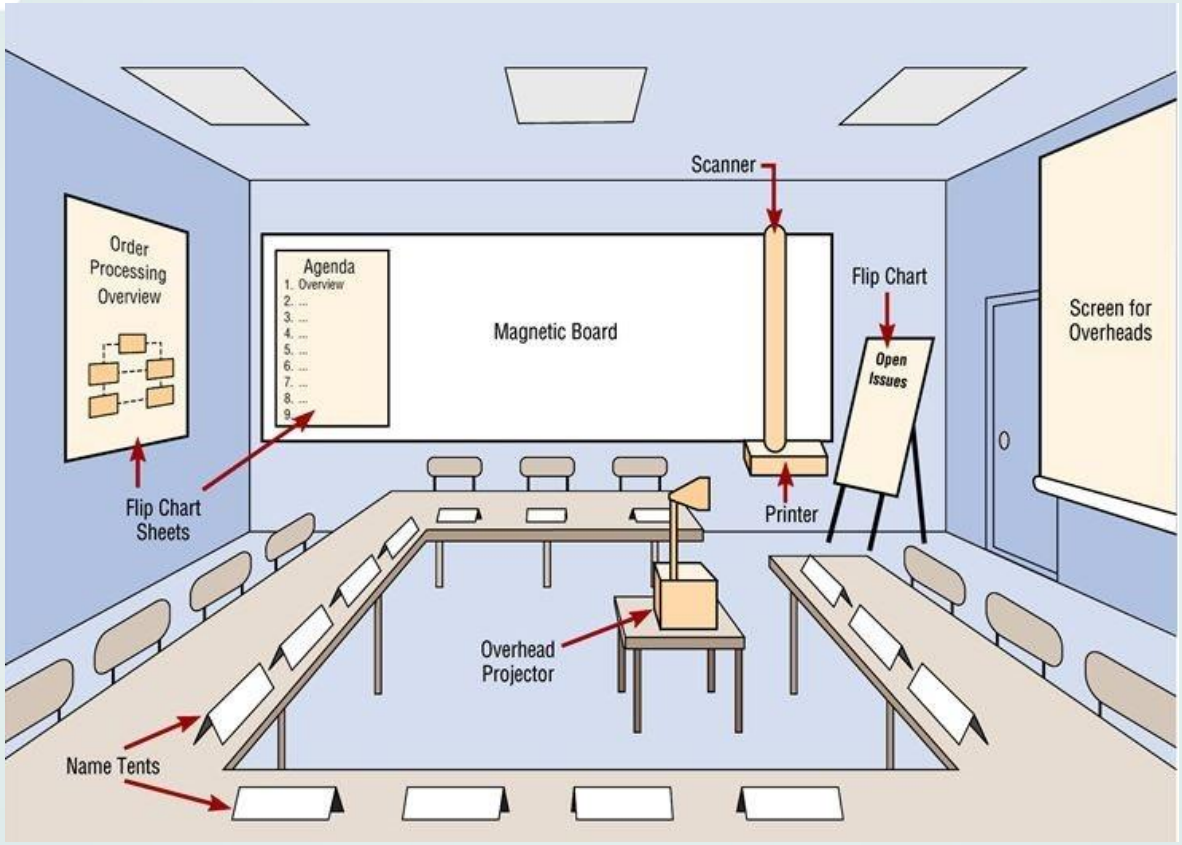
โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

Joint Application Development (JAD)

JAD คือ เทคนิคการ**พัฒนาระบบร่วมกัน** ซึ่งจัดเป็นรากฐานของกระบวนการพัฒนาออกแบบบนพื้นฐานของการพัฒนาระบบคอมพิวเตอร์ที่ประกอบด้วยบุคคลในองค์กร และผู้เชี่ยวชาญทางเทคโนโลยีสารสนเทศเข้าร่วมประชุมเชิงปฏิบัติการ (Workshop)

จุดประสงค์หลักของ JAD นั้นคือการพัฒนาระบบงานที่ใช้เวลาอันสั้นและมีความสมบูรณ์ในโครงการ นั่นหมายถึงคุณภาพของงานที่แล้วเสร็จที่พร้อมจะส่งมอบตรงเวลา

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)



ภายในห้องปฏิบัติการของ JAD ที่เป็นศูนย์รวมการทำงานร่วมกัน

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

ส่วนสำคัญในโครงการ JAD ประกอบด้วย

1.การสนทนาร่วมกัน มุ่งเน้นความคล่องแคล่วและสะดวก ภายใต้กฎเกณฑ์ที่บังคับใช้

2.มีกลุ่มผู้ใช้ประมาณ 3-5 คนที่พร้อมเข้าร่วมประชุมเชิงปฏิบัติการทุกเมื่อ

3.นักพัฒนาที่มีความเชี่ยวชาญด้านเทคโนโลยีสารสนเทศประมาณ 2-3 คน

4.ผู้ที่มีอำนาจซึ่งอาจเป็นผู้จัดการอาวุโสที่สามารถตัดสินและชี้ขาดได้ในกรณีเกิดข้อขัดแย้งระหว่างกันที่ไม่สามารถตกลงกันได้

5.ผู้สังเกตการณ์ประมาณ 2-3 คน ซึ่งมีหน้าที่สังเกตการณ์เท่านั้น ไม่ต้องวิจารณ์สิ่งใด ๆ

6.ผู้เชี่ยวชาญในการสรุปเนื้อหาสาระสำคัญ ที่มีความเข้าใจในระบบงานธุรกิจและเทคโนโลยีสารสนเทศ ที่สามารถสรุปเนื้อหาสาระสำคัญ ๆ ได้สั้น กระชับ และชัดเจน

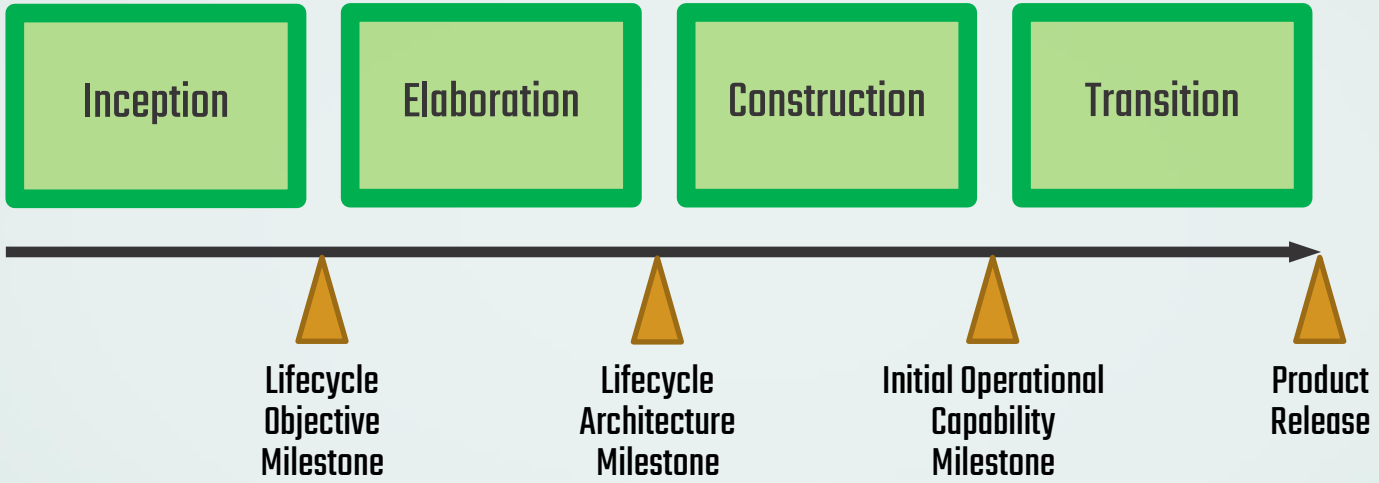
โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

Unified Process (UP)

UP คือ ระเบียบวิธีการพัฒนาระบบเชิงวัตถุ พัฒนาขึ้นโดยบริษัท Rational Software ซึ่งหลายคนรู้จักในนามผลิตภัณฑ์ UML (Unified Modeling Language)

จุดประสงค์ของ UP นั้นต้องการให้ทีมงานพัฒนาซอฟต์แวร์ที่มีคุณภาพสูงตรงตามความต้องการของผู้ใช้ภายใต้งบประมาณและระยะเวลาที่ได้กำหนดไว้ พื้นฐานสำคัญของกระบวนการ UP คือการสร้างโมเดล และการจัดการโมเดลด้วยภาษา UML โดยเฟสหรือระยะต่าง ๆ ใน UP กำหนดไว้ 4 ระยะด้วยกัน คือ

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)



ระยะหรือ Phase ต่างๆของ UP

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

1. Inception Phase (ระยะเริ่มต้น)

ในระยะนี้จะเป็นขั้นตอนของการกำหนดขอบเขตโปรเจกที่แสดงรายละเอียดได้ด้วย Use Case ซึ่งการพัฒนาระบบด้วยวิธีทางเชิงวัตถุนี้ จะกำหนดรูปพรรณของ Use Case นี้ได้ด้วย Use Case Diagram

2. Elaboration Phase (ระยะกำหนดรายละเอียด)

ในระยะนี้จะโฟกัสอยู่สองกิจกรรมด้วยกัน คือ การสร้างข้อกำหนด และการสร้างแผนงานพื้นฐานสำหรับสถาปัตยกรรมระบบ ข้อกำหนดที่ทีมงานสร้างขึ้นประกอบด้วยไดอะแกรมต่าง ๆ เช่น Use Case Diagram, Class Diagrams, Sequence Diagrams และไดอะแกรมอื่น ๆ ของ UML และท้ายสุดการคาดการณ์ในด้านต้นทุน ผลกำไร และความเสี่ยง จะกระทำให้เสร็จสิ้นในเฟสนี้ด้วย

โมเดลการพัฒนาซอฟต์แวร์(ต่อ)

3. Construction Phase (ระยะการสร้างระบบ)

ในระยะนี้จะเป็นการ**พัฒนาระบบ** ด้วยการสร้างซอฟต์แวร์ในลักษณะ Iteration ความเป็นไปได้ในการสร้างหลาย ๆ รีลิสต์ (Release) อย่างต่อเนื่อง จนกระทั่งได้ระบบที่สมบูรณ์

4. Transition Phase (ระยะการเปลี่ยนผ่าน หรือ นำไปใช้)

เป็นระยะของการวางแผนเพื่อ**นำไปใช้** ซึ่งจะทำการฝึกอบรมผู้ใช้ (End-User Training) ติดตั้งระบบ (Installation) และสนับสนุน (Support) ด้วยการปรับปรุงแก้ไขในรายละเอียดในกรณีที่เกิดปัญหา และท้ายสุดก็ดำเนินการส่งมอบระบบงานที่เสร็จสมบูรณ์ที่ผ่านการแก้ไขปรับปรุงแล้วให้กับผู้ใช้งานต่อไป

งานมอบหมาย

จากงานกลุ่ม จงศึกษาลักษณะงานกลุ่มและพิจารณาเลือกโมเดลการพัฒนาซอฟต์แวร์ที่เหมาะสมกับงานกลุ่มของตนเอง พร้อมให้เหตุผลประกอบ