KASETSART UNIVERSITY
FACULTY OF SCIENCE AT SRIRACHA

Software Development Process Models

01418321 System Analysis and Design

Chalothon Chootong (Ph.D.)

Department of Computer Science and Information, Faculty of Science at Sriracha, Kasetsart University Sriracha Campus

chootong.c@ku.th

Software Development Process Models

- คือ แบบจำลองที่ใช้เป็นตัวชี้นำถึงกิจกรรมหลักในการพัฒนาซอฟต์แวร์
- โดยมีการแสดงลำดับขั้นตอนในการพัฒนาที่ชัดเจน และระบุกิจกรรมในแต่ละขั้นตอน เพื่อให้การพัฒนาซอฟต์แวร์ดำเนินต่อไปให้เกิดปัญหาน้อยที่สุด
- โมเดลการพัฒนาซอฟต์แวร์จึงจัดเป็น “กรรมวิธีการพัฒนาซอฟต์แวร์ (Methodology)” หนึ่งที่สามารถนำมาประยุกต์ใช้เพื่อเป็นแนวทางการพัฒนาซอฟต์แวร์ตั้งแต่เริ่มต้นจนกระทั่งสำเร็จ

Software Development Process Models

- โมเดลในการพัฒนาซอฟต์แวร์สมัยใหม่มักจะผนวกขั้นตอนหรือกระบวนการที่สามารถทำงานในลักษณะ
  - การทวนซ้ำเป็นรอบ (Iteration)
  - การพัฒนาแบบก้าวหน้า (Incremental)
  - การจัดทำต้นแบบ (Prototyping)
- การทำงานดังกล่าวจะช่วยลดความเสี่ยงลงได้มากในกรณีที่โครงการมีการเปลี่ยนแปลงความต้องการอยู่ตลอดเวลา

สาเหตุสำคัญที่จำเป็นต้องใช้โมเดลการพัฒนาซอฟต์แวร์

- โมเดลการพัฒนาซอฟต์แวร์จะมีการแตกขั้นตอนของกระบวนการพัฒนาในแต่ละเฟส (Phase)
- ซอฟต์แวร์ที่พัฒนามีความซับซ้อน
- การแบ่งเป็นกระบวนการพัฒนาเป็นเฟสหรือเป็นระยะ จะทำให้ง่ายต่อการจัดการ
- แต่ละเฟสมีแนวทางต่างๆให้เลือกปฏิบัติ

Built-and-Fix Model

- เป็นโมเดลที่มีความเก่าแก่ที่สุด
- เป็นโมเดลการพัฒนาซอฟต์แวร์ที่ว่าด้วยการเขียนโปรแกรม และแก้ไขปรับปรุงโปรแกรมไปเรื่อยๆ ลองผิดลองถูกจนกระทั่งคิดว่าพอใจ ตรงกับความต้องกร
- ทำให้เสียเวลาไปกับการดีบั๊กโปรแกรม และบำรุงรักษาระบบ
- เหมาะสมกับงานขนาดเล็กที่ไม่มีความซับซ้อน
- เหมาะกับงานที่เมื่อเกิดข้อผิดพลาดแล้ว ไม่ส่งผลกระทบต่อระบบมากนัก

Built-and-Fix Model

- ขั้นตอนของโมเดลนี้ ประกอบด้วย
- ✓ เขียนโค้ดหรือโปรแกรมบางส่วนที่คาดหวังว่าจะสามารถ แก้ไขปัญหาโจทย์เหล่านั้นได้
- ✓ คอมไพล์โปรแกรม และทำการรันโปรแกรม
- ✓ หากพบข้อผิดพลาดในโปรแกรม ก็ดำเนินการแก้ไขปรับปรุง
- ✓ กลับไปทำการเขียนโค้ดใหม่ ทำซ้ำจนกระทั่งมีความรู้สึกว่าดีเพียงพอแล้ว

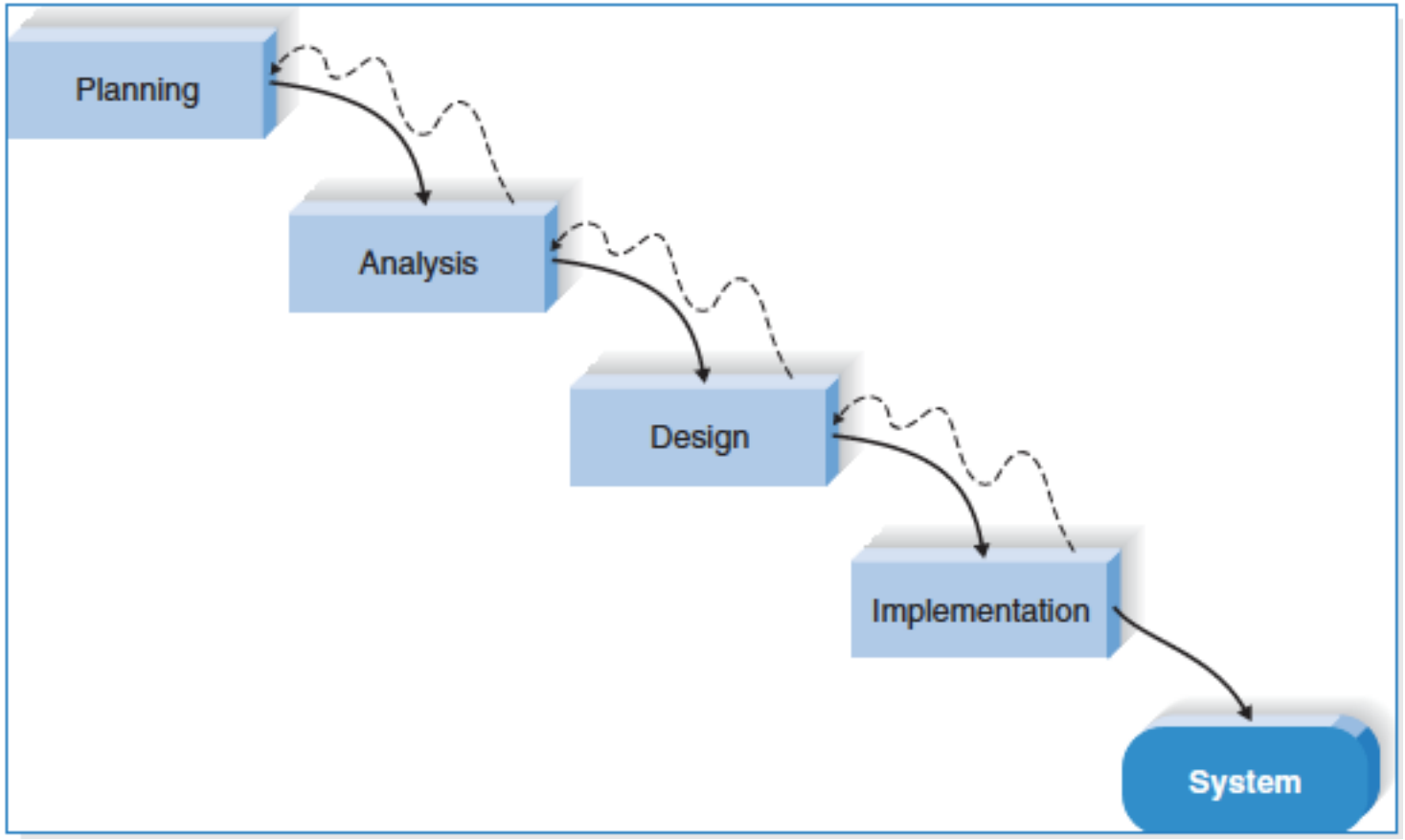
Software Development Process Models

1. Waterfall Model
2. V-Model
3. Incremental Model
4. Iterative Model
5. RAD Model
6. Spiral Model
7. Prototype Model
8. Agile Model

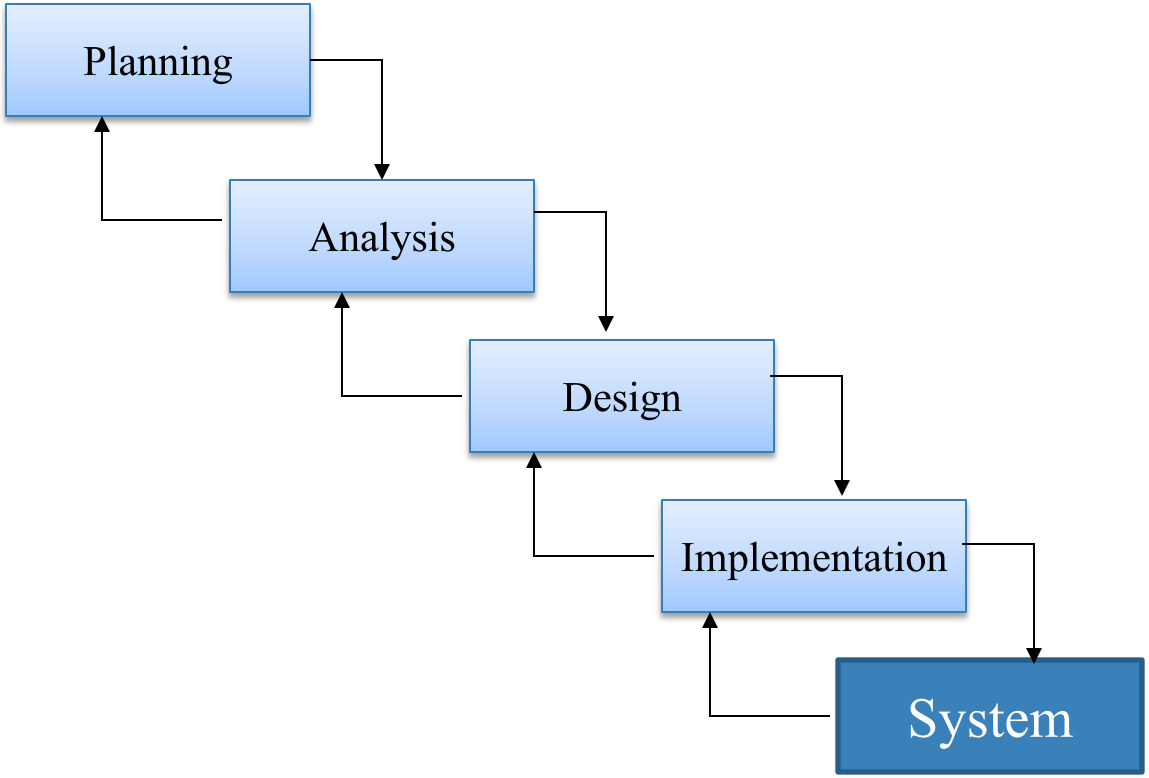
1. Water Fall Model

- หรือโมเดลน้ำตก โมเดลนี้ยังเป็นที่นิยมใช้ในการพัฒนาระบบงานจนถึงปัจจุบัน
- เป็นโมเดลที่ง่ายต่อการประยุกต์ใช้
- มีความคล้ายคลึงกับวงจรการพัฒนาระบบตามแนว SDLC
- ข้อเสียของโมเดลนี้คือ กระบวนการทดสอบจะอยู่ตอนท้ายๆ ดังนั้น หากมีกระบวนการจัดการที่ไม่ดีพอ ก็มีโอกาสที่จะต้องวนกลับไปยังเฟสต้นๆ

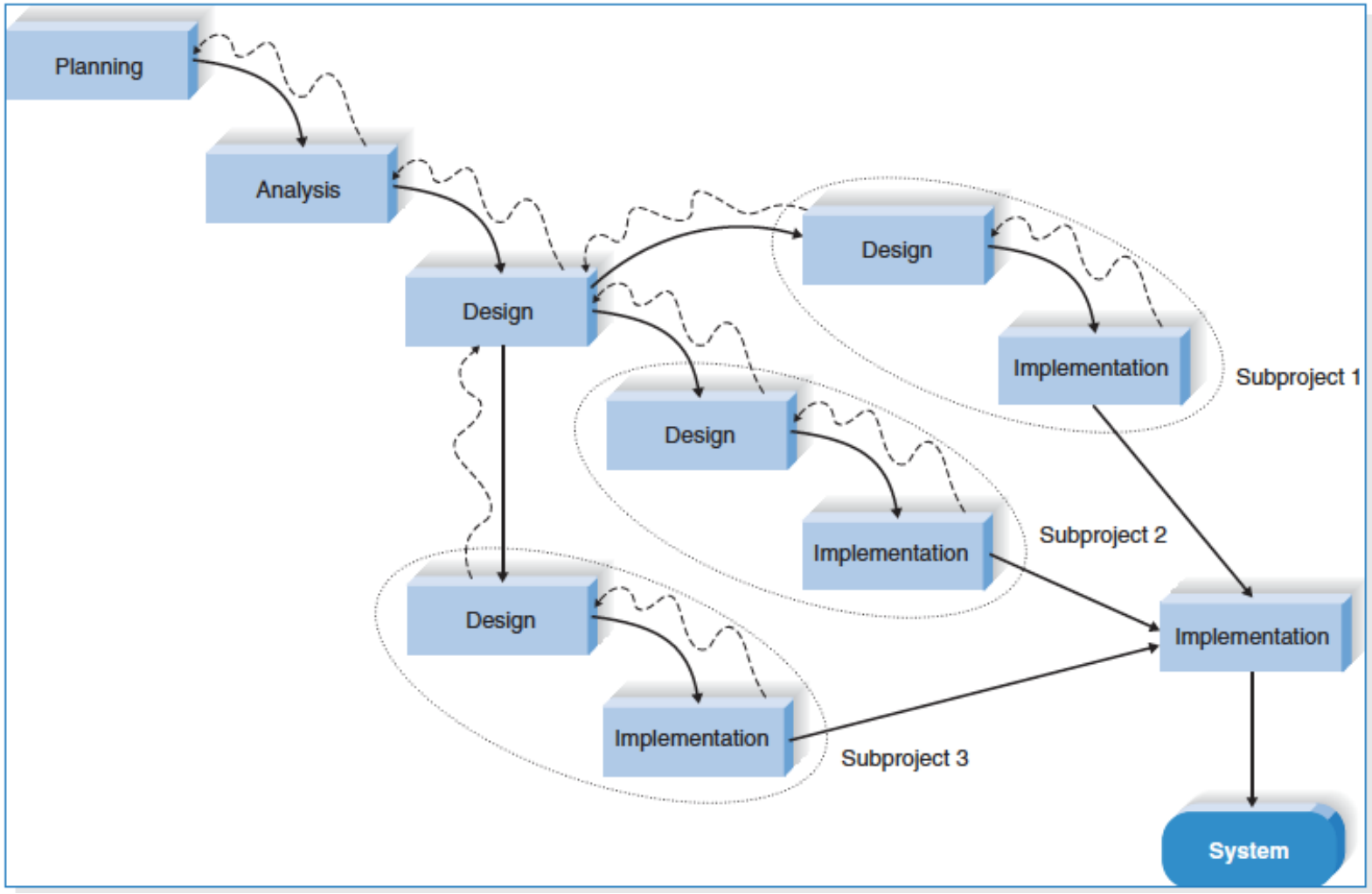
Water Fall Model แบบดั้งเดิม



Water Fall Model ที่มีการวนซ้ำ (Iteration)



Parallel Development



ข้อดี-ข้อเสีย Water Fall Model

- **ข้อดี**

  # เข้าใจได้ง่าย,

  # บริหารจัดการง่าย ขั้นตอนคล้ายกับ SDLC

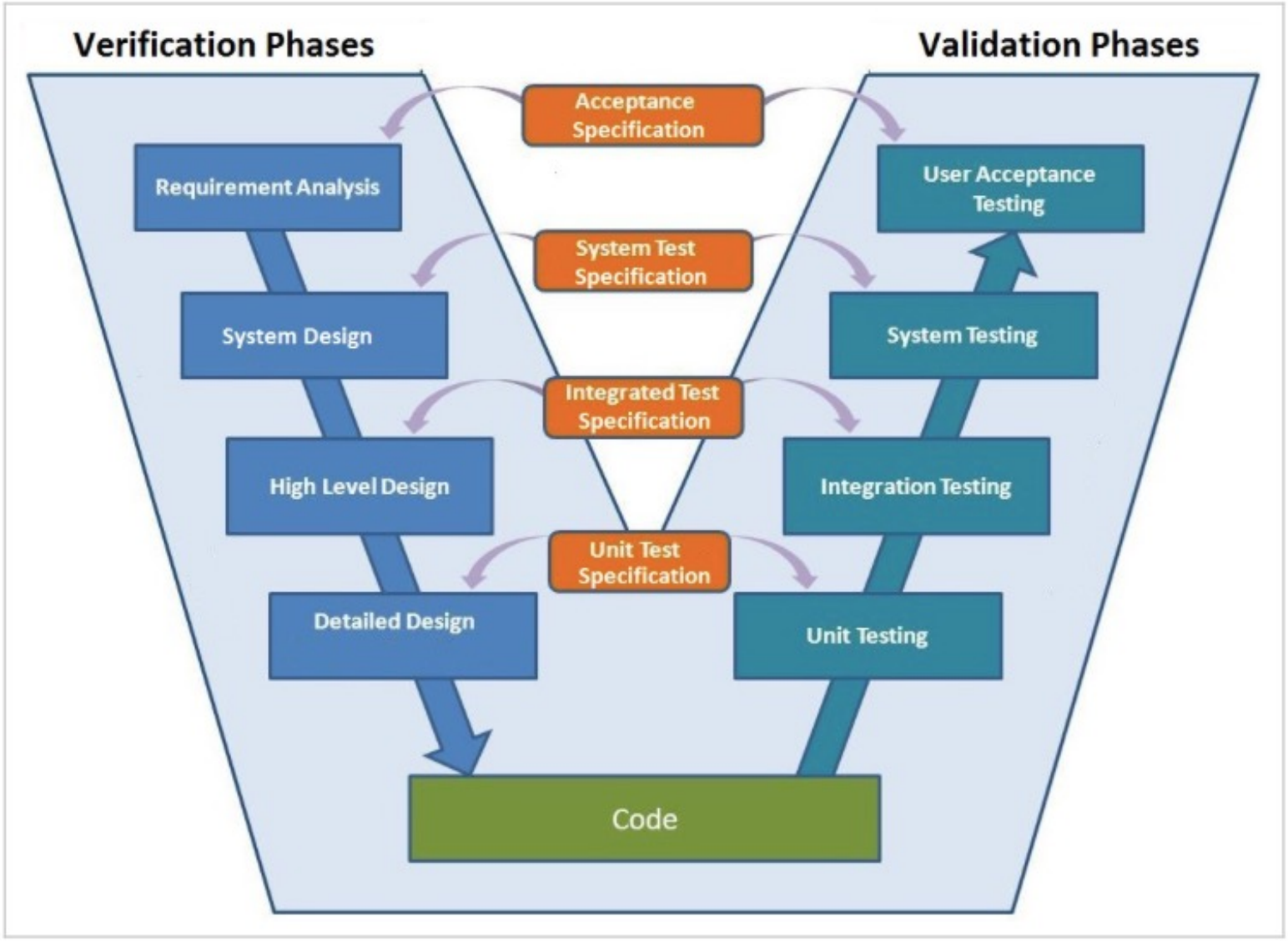
  # เหมาะสำหรับโปรเจคขนาดเล็ก ที่ requirement ไม่เปลี่ยนแปลง

- ข้อเสีย

  # ไม่เหมาะสมสำหรับโปรเจคที่มีความซับซ้อน

  # หรือโปรเจคที่มีการเปลี่ยนแปลงอย่างกระทันหัน

2. V Model


Verification Phases
Validation Phases
Acceptance Specification
Requirement Analysis
User Acceptance Testing
System Test Specification
System Design
System Testing
Integrated Test Specification
High Level Design
Integration Testing
Unit Test Specification
Detailed Design
Unit Testing
Code

V Model

- V model คือส่วนขยายของ Water-Fall Model
- มีขั้นตอนการทดสอบที่สอดคล้องกันสำหรับแต่ละขั้นตอนการพัฒนา ดังนั้น สำหรับทุกขั้นตอนในวงจรการพัฒนาจึงมีขั้นตอนการทดสอบที่เกี่ยวข้อง

จะใช้ V-shaped Model เมื่อไร

- ใช้ในระบบที่ต้องการความเสถียรสูง (high reliability) เช่น ระบบเกี่ยวการจัดการภายในโรงพยาบาล (hospital patient control applications)
- ระบบที่มี Requirement ที่พร้อมและค่อนข้างครบถ้วน

ข้อดีของ V-Shaped model

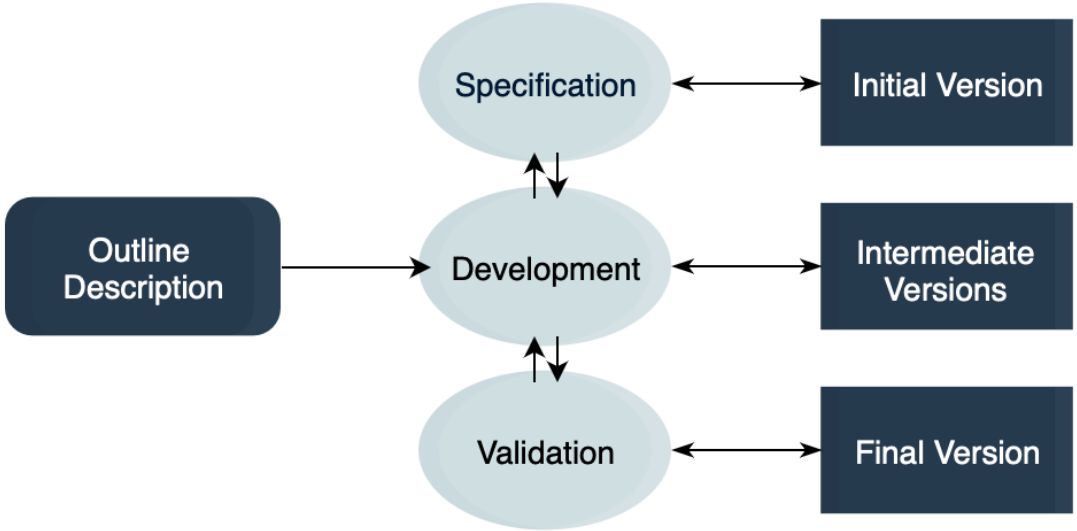
- เน้นการวางแผนสำหรับการ ตรวจสอบ(verification) และการรับรองความถูกต้อง (validation) ของผลิตภัณฑ์ในขั้นเริ่มต้นของการพัฒนาผลิตภัณฑ์
- งานที่ส่งมอบต้องสามารถทดสอบได้ทุกขั้นตอน
- สามารถติดตามความก้าวหน้าได้ทุกขั้นตอน
- เข้าใจง่ายและใช้งานได้ง่าย

ข้อด้อยของ V-Shaped Model

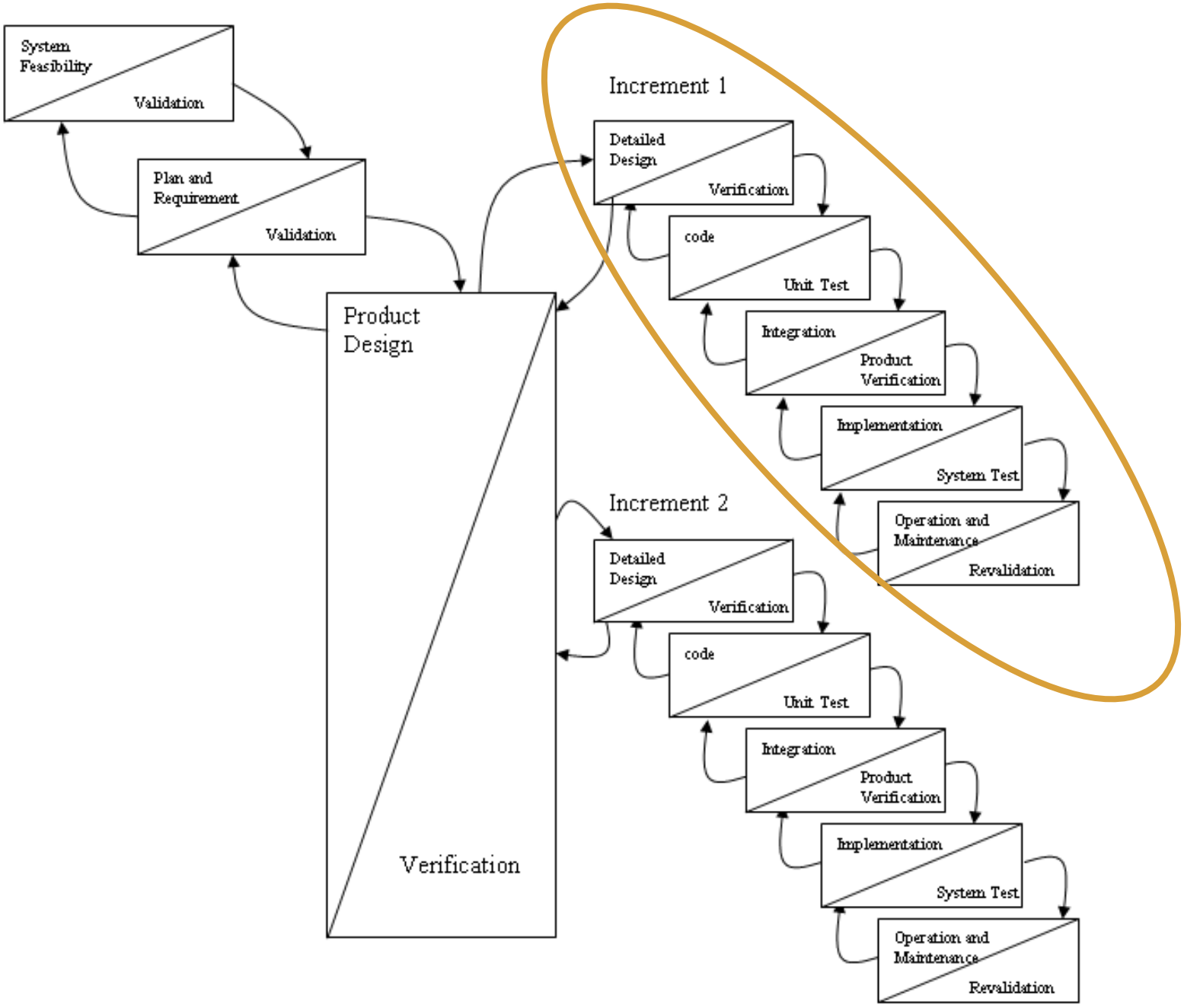
- ยากต่อการจัดการเหตุการณ์ที่เกิดขึ้นพร้อมกัน
- ยากต่อการจัดการกับ Requirement ที่มีการเปลี่ยนแปลงบ่อยๆ
- ไม่มีระบบการจัดการความเสี่ยง (Risk Management)

3. Incremental Model

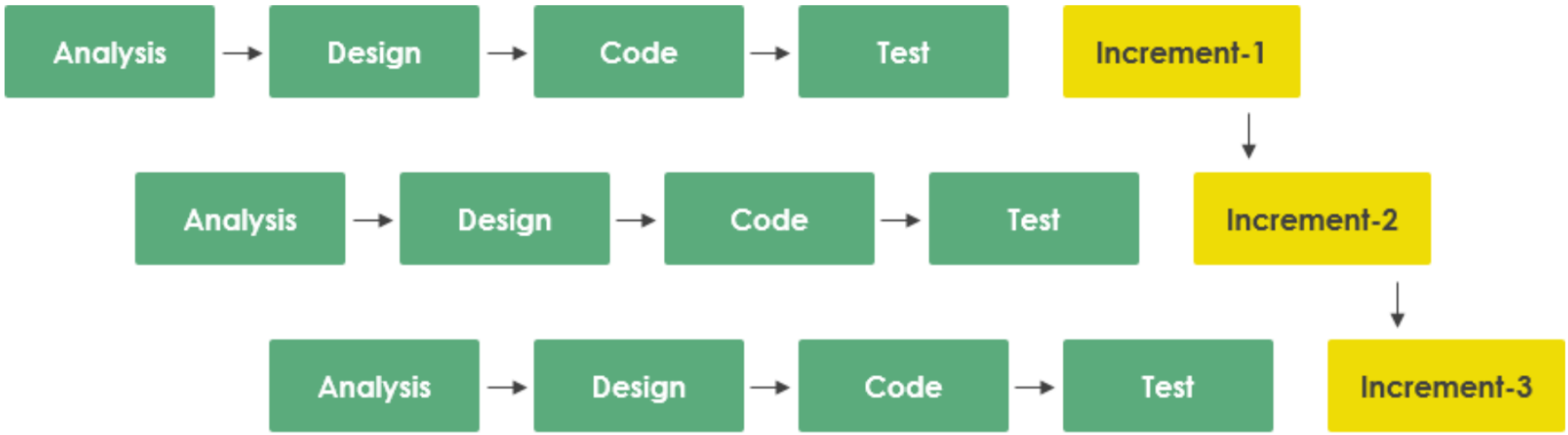
- มีการนำเอาหลักการของ Waterfall Model มาปรับปรุงประสิทธิภาพให้ดียิ่งขึ้น
- มีการเพิ่มส่วนของการออกแบบและพัฒนาในรูปแบบของส่วนงานย่อยในลักษณะแบบก้าวหน้า (Increment) โดยแต่ละส่วนงานย่อยจะมีการทวนซ้ำเป็นรอบในลักษณะ Iteration พร้อมกับมีระบบการตรวจสอบ



https://www.educative.io/blog/software-process-model-types


System
Feasibility
Validation
Plan and
Requirement
Validation
Product
Design
Verification
Increment 1
Detailed
Design
Verification
code
Unit Test
Integration
Product
Verification
Implementation
System Test
Operation and
Maintenance
Revalidation
Increment 2
Detailed
Design
Verification
code
Unit Test
Integration
Product
Verification
Implementation
System Test
Operation and
Maintenance
Revalidation

Incremental Model



Incremental Model

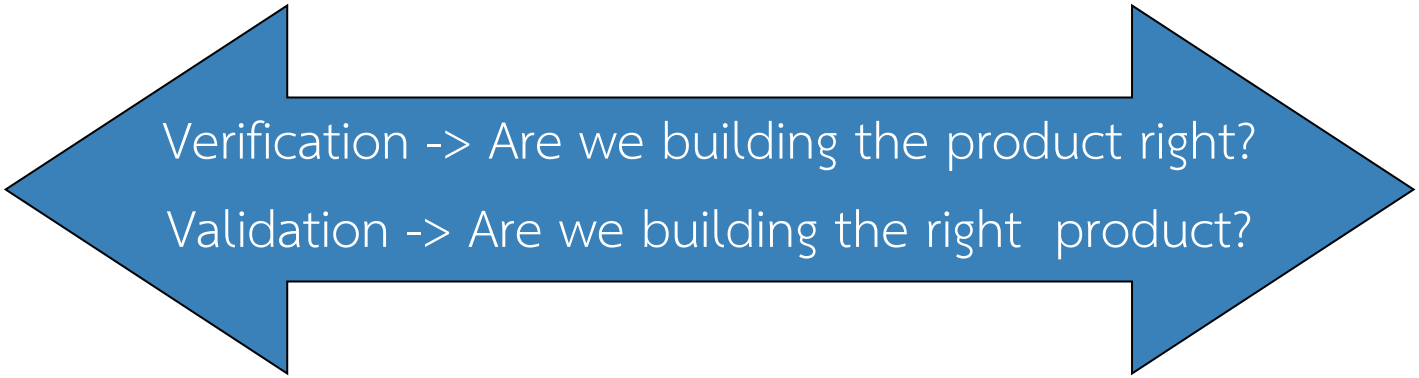
https://www.visual-paradigm.com/guide/software-development-process/what-is-a-software-process-model/

อัตราความก้าวหน้าแต่ละรอบประกอบด้วยการทดสอบดังนี้

- **Detail Design**
  - จะมีการตรวจสอบความถูกต้องตามรายละเอียดของข้อกำหนด (verification)
- **Code**
  - จะมีการทดสอบหน่วยย่อยของโปรแกรม (Unit test)
- **Integration**
  - มีการทดสอบความถูกต้องด้านรายละเอียดของข้อกำหนดในตัวผลิตภัณฑ์ (Product Verification)
- **Implementation**
  - มีการทดสอบระบบโดยรวม (System Test)
- **Operation and Maintenance**
  - มีการทบทวนความต้องการของผู้ใช้อีกครั้ง (Revalidation) ว่าตรงตามวัตถุประสงค์หรือไม่

อัตราความก้าวหน้าแต่ละรอบประกอบด้วยการทดสอบดังนี้

- **Verification** เป็นการตรวจสอบความถูกต้องตามข้อกำหนด (Specification) หรือความพยายามค้นหาข้อผิดพลาดจากการประมวลผลโปรแกรม
- **Validation** เป็นการตรวจสอบรายละเอียดของผลิตภัณฑ์ว่าตรงตามความต้องการของผู้ใช้หรือไม่
- “The incremental model lets stakeholders and developers see results with the first increment. ”



ข้อดี-ข้อเสีย Incremental Model

- **ข้อดี**:

  # เหมาะสำหรับโปรเจคที่มีการเปลี่ยนแปลงระหว่างทาง

  # สามารถตรวจเจอปัญหาได้ตั้งแต่เริ่มต้นโปรเจค เพื่อการวางแผนการจัดการที่ดีขึ้น

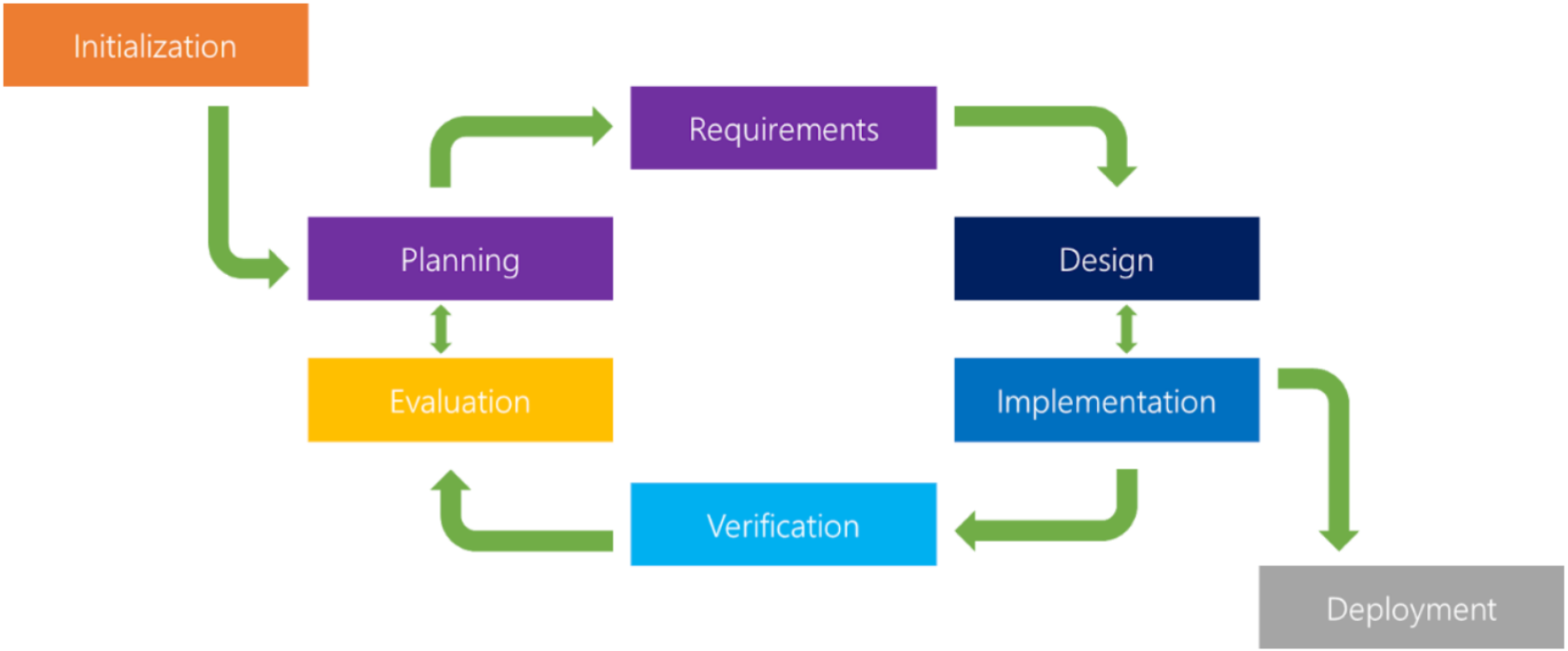
- **ข้อเสีย:**

  # อาจมีค่าใช้จ่ายในการวางแผนโครงการค่อนข้างสูง

  # โมเดลนี้ไม่เหมาะกับโปรเจคที่มีการพัฒนาอย่างต่อเนื่องเพราะลำดับถัดไปไม่สามารถดำเนินการได้ถ้าโครงการก่อนหน้าไม่แล้วเสร็จ

4 Iterative Model

- มีการวางแผนงานแต่ละส่วนของ Project แยกจากกัน
- มีการทบทวนและปรับปรุงในแต่ละ phase ของโปรเจค



Iterative Model

- ข้อดี:
  - # ง่ายต่อการระบุปัญหาที่อาจจะเกิดขึ้น เนื่องจากมีการตรวจสอบแต่แรก
- ข้อเสีย:
  - # เนื่องจากการออกแบบที่ไม่ทับซ้อน ทำให้ต้องใช้เวลานาน
  - # ค่าใช้จ่ายสูง

Iteration & Increment

- **การทวนซ้ำเป็นรอบ (Iteration)**
  - คือการทวนซ้ำของงาน ซึ่งอาจมีการทวนซ้ำมากกว่า 1 รอบ เพื่อให้งานตรงตามความต้องการมากที่สุด
  - การทวนซ้ำถือเป็นการลดความเสี่ยง และจะมีการปรับปรุงให้ดีมากขึ้น
- **การพัฒนาแบบก้าวหน้า (Incremental)**
  - วิธีการพัฒนาส่วนย่อยของระบบให้สมบูรณ์ โดยแต่ละส่วนย่อยอาจมีการทวนซ้ำ
  - มีการเพิ่มระดับความก้าวหน้าในแต่ละระดับในส่วนงานที่ทำเสร็จ และท้ายสุดก็จะนำส่วนงานย่อยๆที่เสร็จสมบูรณ์มาประกอบกันให้เป็นระบบที่สมบูรณ์

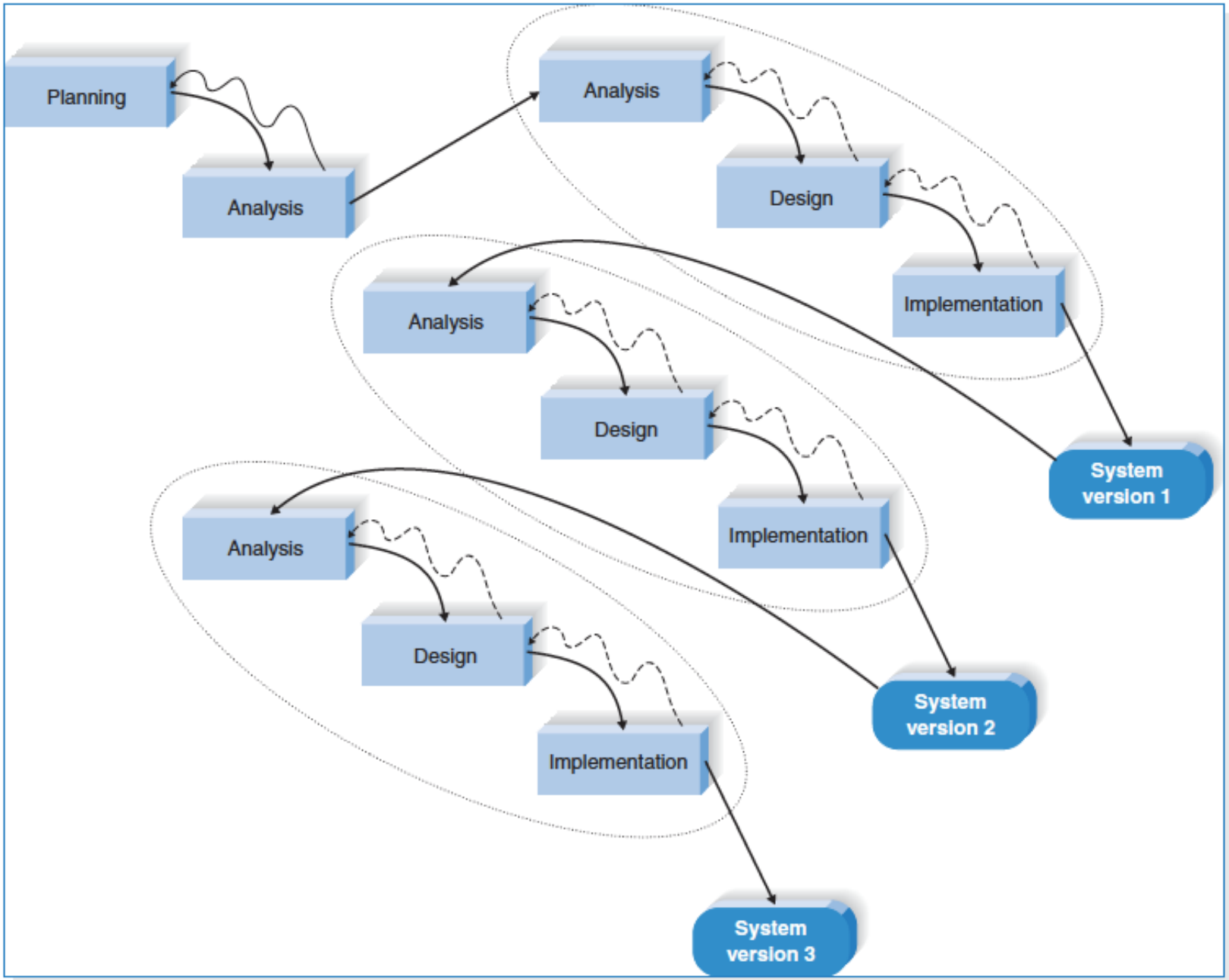
ทำ MVP

5. Rapid Application Development (RAD)

- คือ การพัฒนาแอปพลิเคชันแบบรวดเร็ว และใช้เวลาสั้น
- มักใช้เครื่องมือสนับสนุนอย่าง CASE Tools ช่วยในการพัฒนา
- มุ่งเน้นด้านการลดต้นทุนและระยะเวลาในการพัฒนา
- เป็นการพัฒนาซอฟต์แวร์ที่ย่นระยะเวลาของขั้นตอนการวิเคราะห์ ออกแบบ การสร้าง การทดสอบ เพื่อจะได้ลดเวลาในการพัฒนาโดยรวมลงได้
- การสร้างต้นแบบ (prototype) อย่างรวดเร็ว
- การใช้เทคนิคนี้มุ่งเน้นให้การพัฒนาระบบสำเร็จอย่างรวดเร็ว และใช้งานได้ภายในระยะเวลาที่จำกัดมากกว่าที่จะให้ระบบสมบูรณ์แบบ

5. Rapid Application Development (RAD)

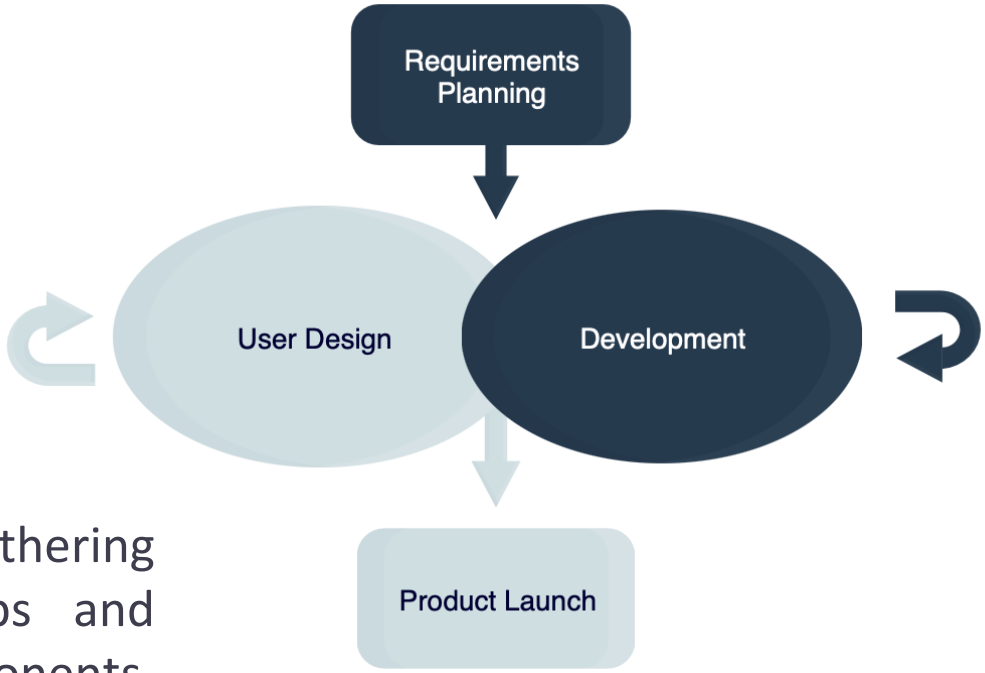
- จำเป็นต้องมีทีมงานขนาดเล็กที่มีความรู้ความสามารถเฉพาะ ซึ่งประกอบด้วยผู้เชี่ยวชาญด้านเทคโนโลยีสารสนเทศกับกลุ่มผู้ใช้งานมาพัฒนาร่วมกัน
- เทคนิคสำคัญของการพัฒนาซอฟต์แวร์ตามกรรมวิธีของ RAD
  - พัฒนาต้นแบบได้อย่างรวดเร็ว
  - เป็นแหล่งรวมเครื่องมือเพื่อการพัฒนาแอปพลิเคชัน
  - มีทีมงานที่เชี่ยวชาญการใช้เครื่องมือเหล่านั้น
  - มีกรอบระยะเวลาการพัฒนาที่จำกัด
- ข้อเสีย อาจไม่รองรับการเปลี่ยนแปลงในอนาคตได้


Planning
Analysis
Analysis
Design
Implementation
Analysis
Design
Implementation
System version 1
Analysis
Design
Implementation
System version 2
Implementation
System version 3

RAD Model

It involves the following phases:

1.Business modeling
2.Data modeling
3.Process modeling
4.Application generation
5.Testing and turnover



The RAD concept focuses on gathering requirements using focus groups and workshops, reusing software components, and informal communication.

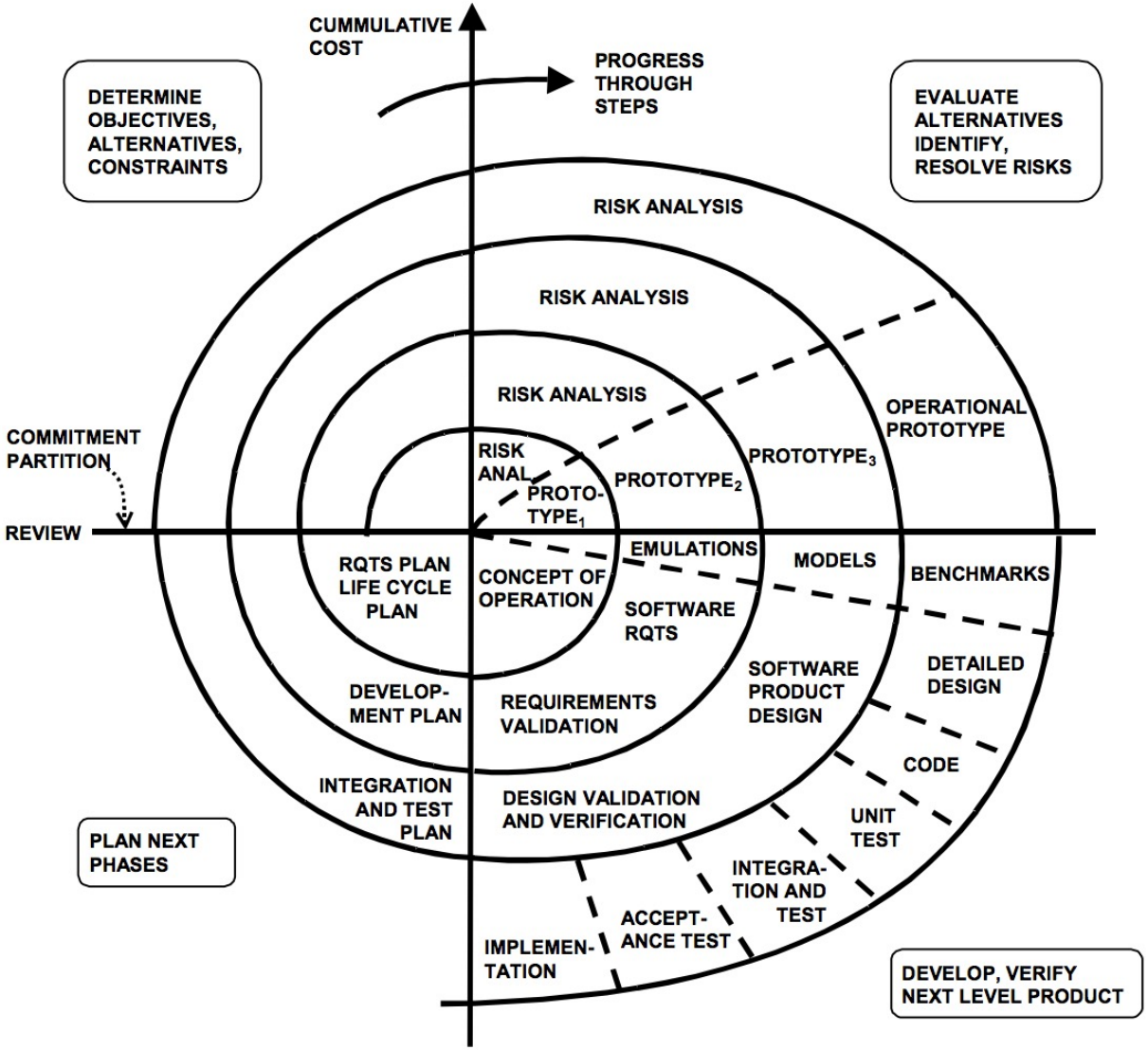
ข้อดี-ข้อเสีย RAD

- **ข้อดี:**
  - # ลดเวลาในการพัฒนา
  - # รับ feedback จาก User ได้ตลอดเวลา
- **ข้อเสีย:**
  - # ต้องอาศัยนักพัฒนาที่มีความชำนาญสูง
  - # มีทักษะการวางแผนที่ดี
  - # และอาจมีปัญหาในการรวมงานตอนสุดท้าย
  - # อาจส่งผลให้เกิดความล้มเหลวที่คาดไม่ถึงและการพัฒนาส่วนประกอบใหม่เพื่อให้พอดีกับส่วนที่เหลือ

6. Spiral Model

- เป็นโมเดลที่มีหลักการทำงานเป็นลักษณะรอบวงกลม วนเป็นก้นหอย ซึ่งจะวนเป็นก้นหอยตามเข็มนาฬิกา ด้วยการวนไปเรื่อยๆ จากวงในไปสู่วงนอก
- โดยในแต่ละวงรอบนั้นจะประกอบด้วยขั้นตอนย่อยที่สำคัญคือ
  - การวิเคราะห์ความต้องการ (Requirement Analysis)
  - การวิเคราะห์ความเสี่ยง (Risk Analysis)
  - การออกแบบต้นแบบ (Design Prototype)
  - การพัฒนาต้นแบบและการนำมาประกอบรวมกัน (Develop and Integrate Prototype)

6. The Spiral Process Model



6. The Spiral Process Model

- ต้นแบบจะมีความสมบูรณ์ในแต่ละรอบวงกลม จากนั้นจะดำเนินการในเฟสต่อไป
- แบบจำลองนี้จะใช้ได้อย่างเหมาะสมกับระบบงานที่มีโอกาสเปลี่ยนแปลงบ่อย เนื่องจากในแต่ละเฟสต้องมีการวิเคราะห์ความต้องการใหม่ และวิเคราะห์ความเสี่ยงว่าจะทำการพัฒนาต่อหรือไม่

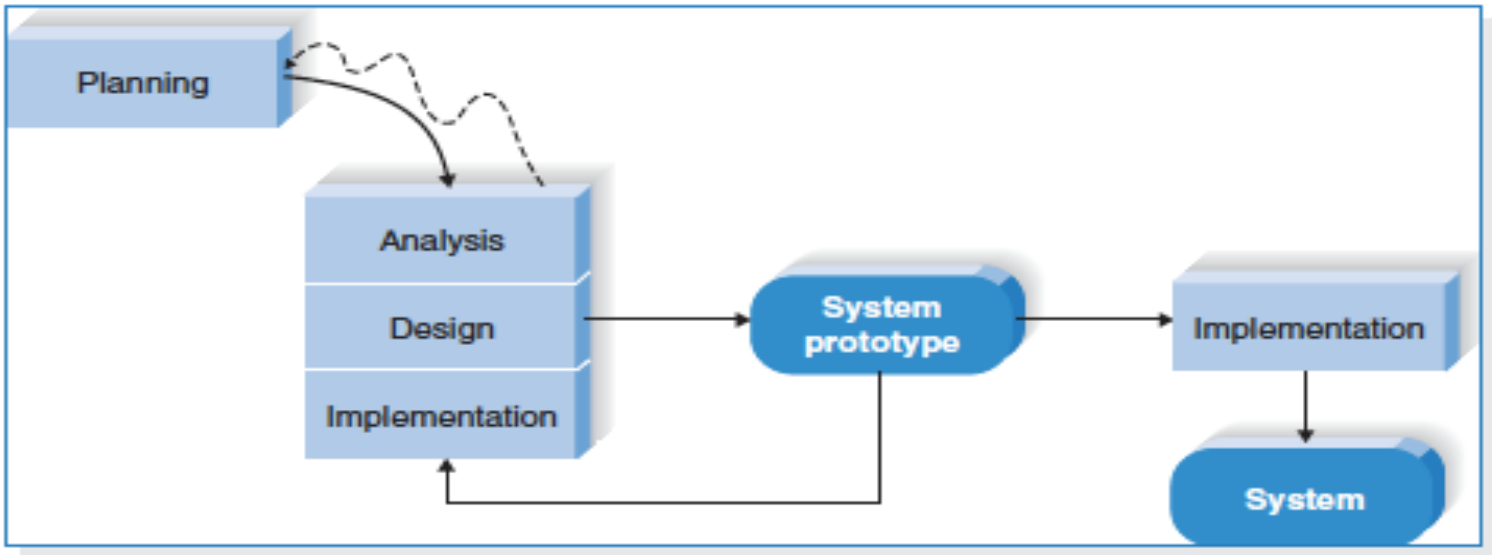
6. The Spiral Process Model

- หลักการของ Spiral Model คือ
  - กระบวนการพัฒนาระบบมีศักยภาพและมีความสมบูรณ์มากขึ้นเรื่อย ๆ
  - วงจรของการเติบโตของระบบแบบค่อยเพิ่มขึ้น ในแต่ละลำดับของการ Implement
  - ความเสี่ยงของการพัฒนาระบบจะค่อย ๆ ลดลง
  - เหมาะสำหรับการพัฒนาระบบขนาดใหญ่
  - เมื่อกระบวนการพัฒนาซอฟต์แวร์ก้าวไป ทั้งลูกค้าและผู้พัฒนา จะมีความเข้าใจในตัวซอฟต์แวร์ดีขึ้น
  - การตอบสนองต่อความเสี่ยง ณ ระดับวิวัฒนาการแต่ละรอบดีขึ้น

ข้อดี-ข้อเสีย Spiral Process Model

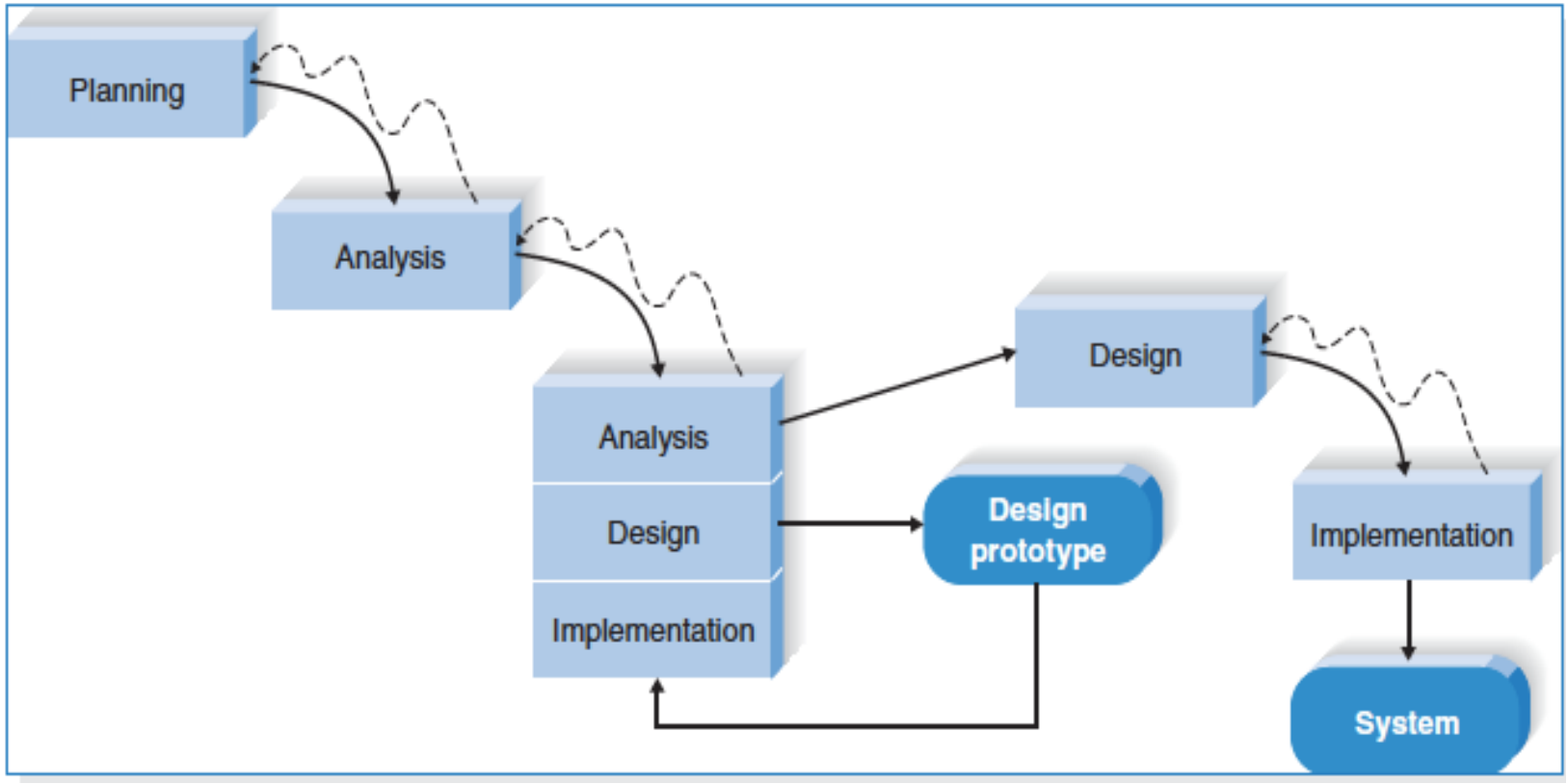
- **ข้อดี:**
  - # แบ่งการพัฒนาออกเป็นขั้นตอน
  - # สามารถจัดการความเสี่ยงในการพัฒนาโปรเจคได้
  - # สามารถประมาณการงบประมาณได้อย่างแม่นยำ
- **ข้อเสีย:**
  - # เกิดการสับสนในการนำไปใช้ เนื่องจากมีการปรับแต่งได้ตลอดเวลา
  - # ต้องการสมาชิกในทีมที่สามารถบริหารความเสี่ยงได้ดี

7. System Prototyping



- วิธีการทำต้นแบบนี้เป็นวิธีที่ผู้ใช้สามารถอธิบายหรือบอกสิ่งที่ต้องการหรือไม่ต้องการ ชอบหรือไม่ชอบเกี่ยวกับระบบที่กำลังจะใช้ต่อไปในอนาคต

Throwaway Prototyping



7. Prototyping

- **ขั้นตอนที่ 1** ระบุสารสนเทศขั้นพื้นฐานที่ผู้ใช้ต้องการ ในขั้นตอนนี้ผู้ใช้บอกความต้องการพื้นฐานเกี่ยวกับผลลัพธ์ที่ต้องการจากระบบ ผู้ออกแบบจะต้องรับผิดชอบที่จะพิจารณาขอบข่ายของระบบที่ผู้ใช้ต้องการ และประมาณการค่าใช้จ่ายในการพัฒนา
- **ขั้นตอนที่ 2** การพัฒนาต้นแบบครั้งแรก วัตถุประสงค์ ก็คือการจัดทำระบบที่จะนำไปใช้ ให้ตรงตามความต้องการพื้นฐานที่ผู้ใช้ ผู้ออกแบบจะต้องรับผิดชอบในการสร้างระบบ โดยการเขียนโปรแกรม หรือใช้ซอฟต์แวร์บางอย่าง เพื่อให้สามารถออกผลลัพท์ตามที่ผู้ใช้ระบุ ซึ่งอาจจะยังไม่ครบถ้วนสมบูรณ์ทั้งหมด เพื่อจัดส่งให้ผู้ใช้ได้ทดลองใช้ ในขั้นตอนนี้ยังไม่คำนึงถึงในเรื่องประสิทธิภาพในการทำงานของระบบ
- **ขั้นตอนที่ 3** ทดลองใช้ต้นแบบ ขั้นตอนนี้เปิดโอกาสให้ผู้ใช้ได้มีประสบการณ์การใช้ระบบเพื่อให้เข้าใจสารสนเทศที่ต้องการ และให้รู้ว่ามีอะไรบ้างที่ระบบสามารถทำได้ และอะไรบ้างที่ทำไม่ได้ เป็นที่คาดหวังว่าผู้ใช้ควรจะค้นพบปัญหาในข้อนี้ ในขั้นตอนนี้ผู้ใช้และผู้ออกแบบจะต้องพิจารณาว่าต้องการให้มีการปรับปรุง เปลี่ยนแปลง แก้ไข หรือเพิ่มเติมในส่วนใดบ้าง
- **ขั้นตอนที่ 4** แก้ไขปรับปรุงต้นแบบ ผู้ออกแบบปรับปรุง เปลี่ยนแปลง แก้ไข เพิ่มเติมระบบตามที่ผู้ใช้ต้องการ ซึ่งควรจะต้องทำการปรับปรุงอย่างรวดเร็ว และส่งให้ผู้ใช้ทดลองใช้งาน

ข้อดี-ข้อเสีย Prototyping

- **ข้อดี:**
  - # สามารถช่วยลดเวลาและค่าใช้จ่ายในการพัฒนา
  - # ผู้ใช้งานมีส่วนร่วมในการออกแบบ
- **ข้อเสีย:**
  - # อาจทำให้ผู้ใช้สับสนระหว่างต้นแบบกับผลิตภัณฑ์สำเร็จรูป
  - # อาจเสียเวลาในการพัฒนาต้นแบบ

8. Agile Development

- Agile คือกลุ่มของ Software methodologies ที่มีแนวคิดเดียวกัน มีลักษณะการพัฒนาแบบ iteration คือ แบ่งส่วนย่อยๆ แล้วพัฒนาต่อเรื่อยๆ แต่ละ iteration จะต้องเกิด working software และต้องพัฒนาต่อ เรื่อยๆ และมีลักษณะให้ความสำคัญกับคนที่ทำ มากกว่า process Agile มี core value ทั้งหมด 4 ข้อดังนี้

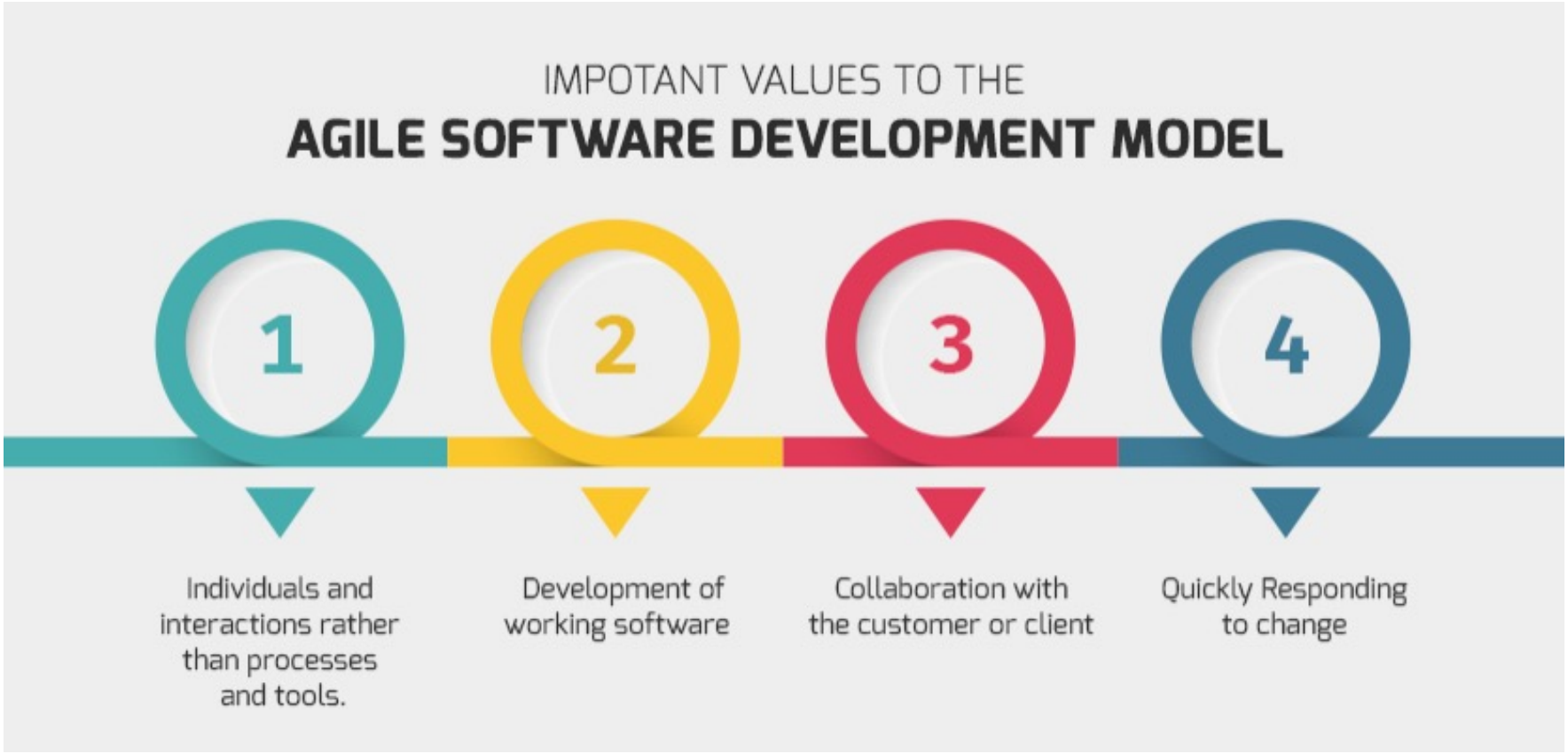
✓ Individuals interactions over process and tool คือการให้ความสำคัญกับบุคคลมากกว่า process และ tool

✓ Working software over comprehensive documentation คือการส่งมอบซอฟต์แวร์ที่นำไปใช้ งานได้จริง มากกว่าการทำเอกสารที่ครบสมบูรณ์

✓ Customer collaboration over contract negotiation คือการเจอลูกค้าในการพัฒนาตลอดระยะ เวลา มากกว่าการทำตามสัญญา

✓ Responding to change over following a plan คือการยอมรับการเปลี่ยนแปลงมากกว่าการทำแผนที่วางไว้

Agile



https://ca.insight.com/en_CA/content-and-resources/2016/07152016-types-of-software-development-models.html#prototype-model

ข้อดี-ข้อเสีย Agile

- **ข้อดี:**
  - # มีการสื่อสารที่ต่อเนื่องของลูกค้า
  - # มีทิศทางที่ชัดเจนในการพัฒนาระบบ
- **ข้อเสีย:**
  - # ถ้าผู้ใช้ไม่ชัดเจนอาจทำให้การพัฒนาสับสน
  - # เอกสารประกอบน้อย
  - # ต้องการทีมที่มีความรอบรู้

Waterfall vs Agile

- Waterfall ตลอดระยะเวลาการพัฒนาจะไม่เห็น product ที่ส่งให้ลูกค้าเลยจะเห็น product จริงๆ ก็ถึงระยะ phase สุดท้ายแล้ว ในขณะที่ Agile ส่ง working software (ซอฟต์แวร์ที่ใช้งานได้จริง) ให้ดูว่า นี้คือสิ่งที่ลูกค้าต้องการหรือไม่ ลูกค้าจะสามารถเปลี่ยน requirement ได้ทันที
- Waterfall มีค่าใช้จ่ายในการรัน project เพิ่มขึ้นเรื่อยๆ เพราะrequirement เปลี่ยนแปลงอยู่ ตลอดเวลา หาก requirement ถูกเปลี่ยนในขณะที่อยู่ใน phase coding หรือ design จะมีค่าใช้จ่ายเพิ่ม ขึ้นกว่า phase แรก ๆ ยิ่งถ้า product นั้นเสร็จแล้ว มีการเปลี่ยนแปลง requirement เพิ่มเติม ค่าใช้จ่าย จะสูงมาก ในขณะที่ Agile จะ fix ค่าใช้จ่ายในการรัน project ไว้คงที่ตลอด
- Waterfall มีความเสี่ยงในระยะ phase สุดท้ายมากเพราะมีโอกาสที่productออกมาแล้วไม่ใช่ สิ่งที่ลูกค้าต้องการ ทำให้มีโอกาส project fail ได้สูง ในขณะที่ Agile รับความเสี่ยงตั้งแต่ต้น เพราะ การส่งมอบ product ครั้งแรกอาจไม่ใช่สิ่งที่ลูกค้าต้องการ ดังนั้นจึงปรับเปลี่ยนเรื่อยๆ จนความเสี่ยงลดลง

Comparison Table

Usefulness in Developing Systems	Waterfall	Parallel	V-Model	Iterative	System Prototyping	Throwaway Prototyping	Agile Development
with unclear user requirements	Poor	Poor	Poor	Good	Excellent	Excellent	Excellent
with unfamiliar technology	Poor	Poor	Poor	Good	Poor	Excellent	Poor
that are complex	Good	Good	Good	Good	Poor	Excellent	Poor
that are reliable	Good	Good	Excellent	Good	Poor	Excellent	Good
with short time schedule	Poor	Good	Poor	Excellent	Excellent	Good	Excellent
with schedule visibility	Poor	Poor	Poor	Excellent	Excellent	Good	Good

Tools to Support System Development

- มีการนำเครื่องมือมาใช้งาน หรือนำมาเพื่อสนับสนุนการพัฒนาระบบ ทำให้การพัฒนาระบบมีความรวดเร็วและมีคุณภาพยิ่งขึ้น เรียกว่า เคสทูลส์ (Computer-Aided Software Engineering : CASE Tools)
- เป็นเครื่องมือที่ออกแบบเพื่อช่วยทำการวิเคราะห์ให้ระบบมีความสมบูรณ์ยิ่งขึ้น
- เคสทูลส์จะมีฐานข้อมูลที่บรรจุไปด้วยสารสนเทศที่เกี่ยวกับโปรเจ็กต์ที่พัฒนา ซึ่งเรียกว่า Repository ซึ่งเป็นแหล่งรวบรวมสารสนเทศที่เกี่ยวกับระบบ ซึ่งอาจประกอบด้วยโมเดล คำอธิบาย รวมถึงการอ้างอิงหรือลิงก์
- สามารถทำการตรวจสอบความถูกต้องของโมเดล

เครื่องมือที่ใช้สนับสนุนการพัฒนาระบบ

เครื่องมือสร้างแผนภาพหรือไดอะแกรม (Diagram Tools)

- ใช้สำหรับสร้างแบบจำลองตามกรรมวิธีของการพัฒนาระบบ เช่น UML Diagram , Data Flow Diagram , ER Diagram

เครื่องมือจัดทำคำอธิบาย (Description Tools)

- ใช้บันทึกคำอธิบายและรายละเอียดของระบบ ทำเพื่อประกอบคำอธิบายของแผนภาพ

เครื่องมือสร้างหรือจัดทำต้นแบบ (Prototyping Tools)

- ช่วยในการสร้างส่วนประกอบของระบบ

เครื่องมือที่ใช้สนับสนุนการพัฒนาระบบ

- **เครื่องมือจัดการด้านคุณภาพ (Quality Management Tools)**
  - ใช้สำหรับวิเคราะห์แบบจำลอง คำอธิบายรายละเอียด และต้นแบบ เพื่อตรวจสอบความถูกต้องครบถ้วน สมบูรณ์ ความถูกต้องตรงกัน
- **เครื่องมือจัดทำเอกสาร (Documentation Tools)**
  - เป็นเครื่องมือที่รวบรวม และจัดทำเอกสารในรูปของรายงาน เพื่อให้สามารถนำเอกสารรายงานเหล่านี้ไปเสนอแก่เจ้าของระบบ ผู้ใช้งาน นักออกแบบ หรือนักพัฒนา
- **เครื่องมือการออกแบบและแปลงรหัส (Design and Code Generator Tools)**
  - เป็นเครื่องมือในการแปลงหรือการ Generate โดยอัตโนมัติ เช่น ทำการแปลงแผนภาพ ER ไดอะแกรม ให้เป็นฐานข้อมูลโดยอัตโนมัติ หรือมีการแปลงโมเดลให้เป็นรหัสโปรแกรม

Case tools types

แบ่งออกเป็น 2 ประเภทคือ

- Upper CASE Tools

■เป็นเครื่องมือที่สนับสนุนการวิเคราะห์ในระหว่างขั้นตอนของการวิเคราะห์และออกแบบ เช่น การสร้างและการตรวจสอบโมเดลที่สร้าง

- Lower CASE Tools)

■เป็นเครื่องมือที่สนับสนุนด้านการนำไปใช้ (Implementation) เช่น โมเดลที่ออกแบบไว้ สามารถทำการแปลงให้เป็นรหัสโปรแกรม

Thanks!

Any questions?