


## บทที่ 6 Key Methodologies for IT Project Management

### Methodology

 แปลว่า รูปแบบหรือวิธีการในการดำเนินการหรือรูปแบบวิธีปฏิบัติในการทำสิ่งใด ๆ

 ในการพัฒนาซอฟต์แวร์ หมายถึง รูปแบบที่ทีมพัฒนาซอฟต์แวร์ใช้ในการบริหารจัดการทรัพยากรต่าง ๆ ไม่ว่าจะเป็นทรัพยากรวัสดุ สิ่งของ คนรวมไปถึงขั้นตอนการทำงานต่าง ๆ เพื่อให้การพัฒนา ซอฟต์แวร์ นั้นสำเร็จลุล่วงไปด้วยดีเรียกอีกอย่างว่า ซอฟต์แวร์ Development Model หรือ ซอฟต์แวร์ Development life cycle (SDLC)

### Waterfall Model

 เป็นโมเดลเชิงเส้นที่ตรงไปตรงมามีการทำงานที่ไม่ซับซ้อนเป็นโมเดลดั้งเดิมที่มีการใช้งานมายาวนานตั้งแต่การพัฒนา ระบบ IT ในยุคก่อนหน้าโดยโมเดลประกอบไปด้วยขั้นตอนที่เรียงกันตามลำดับ ได้แก่

 Requirements


 Design

 Implementation

 Verifications

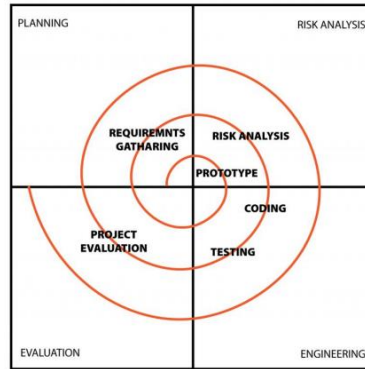
 Maintenance

 แต่ละขั้นตอนจะต้องเสร็จสิ้น 100% ก่อนที่จะไปดำเนินการในขั้นตอนต่อไปได้ซึ่งหากใช้โมเดลนี้ในการพัฒนา ซอฟต์แวร์ จะไม่มีขั้นตอนหรือแนวทางในการย้อนกลับไปแก้ไข

 ข้อดี → เนื่องจากโมเดลมีขั้นตอนการทำงานเป็นเส้นตรงส่งผลให้ง่ายต่อการบริหารจัดการเหมาะสมกับโปรเจกต์ที่มีเป้าหมายและความต้องการที่ชัดเจน ผู้คุมโครงการไม่จำเป็นที่จะต้องมีการตรวจสอบการคืบคลานมากนักก็สามารถใช้เทคนิคนี้ได้เนื่องจากมีความตรงไปตรงมาไม่ซับซ้อนเหมาะสำหรับงาน เช่น งานราชการ งานที่บุคลากรมีตำแหน่งหน้าที่ชัดเจน งาน freelance ที่ทำงานคนเดียว

 ข้อสังเกต → การดำเนินงานตามโมเดลนี้จะยึดแผนการทำงานและ deadline เป็นหลักซึ่งในหลายครั้งส่งผลให้เกิดการล่าช้าเนื่องจากธรรมชาติของการทำโครงการมักที่จะเกิดการล่าช้าจากการบริหารความเสี่ยงที่ไม่ครอบคลุม อีกทั้งไม่มีวิธีการรองรับการบริหารบุคลากร หากมีความจำเป็นจะต้องมีการสลับตำแหน่งจะทำให้การทำงานในส่วนอื่นเกิดข้อบกพร่องได้

## Spiral Model (meta model)



🧸 เป็นโมเดลที่มีลักษณะเป็นรูปก้นหอยลากเส้นโค้งวนจากด้านในออกมาด้านนอกซึ่งจะมีลักษณะการทำงานเป็นรูป คือ สามารถวนลูปได้เรื่อย ๆ จนกว่าการพัฒนาซอฟต์แวร์จะเสร็จสิ้นโดยแต่ละลูปจะเรียกว่า เฟส (phase) แบ่งเป็น 4 เฟส

- 🌸 Requirements การได้มาซึ่งความต้องการของระบบ
- 🌸 Risk management การวิเคราะห์ปัญหาหรือความเสี่ยง
- 🌸 Implementation การพัฒนาระบบ
- 🌸 Testing & Evaluation การทดสอบและประเมินผล

🧸 โมเดลนี้สามารถใช้ในการบริหารความเสี่ยงได้ดี เนื่องจากมีการบริหารความเสี่ยงในทุกขั้นของการทำงาน อีกทั้งยังมีช่วงให้สามารถปรับแก้ไขความต้องการได้ การพัฒนาซอฟต์แวร์ที่แบ่งเป็นเฟสโดยการพัฒนาไปทีละส่วนจะเหมาะสมกับโมเดลนี้

🧸 ข้อดี → การรับมือกับความเสี่ยงถือเป็นจุดเด่นของโมเดลนี้ซึ่งจัดว่าเป็นหนึ่งในโมเดลที่ดีที่สุดในด้านการวิเคราะห์รับมือและจัดการกับความเสี่ยงซึ่งจะทำในทุก ๆ เฟสของการพัฒนาซอฟต์แวร์ อีกทั้งยังมีความยืดหยุ่นในการจัดการกับความต้องการของระบบ เนื่องจากจะมีการตรวจปรับความต้องการอยู่ในทุก ๆ เฟสเช่นเดียวกันเป็นโมเดลที่เหมาะสมสำหรับโครงการขนาดใหญ่ที่มีความซับซ้อน

🧸 ข้อเสีย → ไม่เหมาะสมสำหรับโครงการที่มีงบประมาณน้อย เนื่องจากการบริหารความเสี่ยงหรือการเก็บความต้องการในทุก ๆ เฟสของการพัฒนาระบบจะมีค่าใช้จ่ายที่มากเป็นโมเดลที่มีความซับซ้อนกว่าโมเดลแบบอื่น ๆ เนื่องจากมีการทำซ้ำกิจกรรมเดิม ๆ ในทุก ๆ เฟสส่งผลให้เปลืองทรัพยากรต้องใช้ผู้เชี่ยวชาญในการทำงาน เนื่องจากการวิเคราะห์ความเสี่ยงอยู่เป็นประจำบริหารจัดการการทำงานด้วยโมเดลนี้ได้ยาก

## Agile Model



🧸 เป็นรูปแบบการพัฒนาซอฟต์แวร์ที่นิยมใช้ในยุคปัจจุบัน เนื่องจากรูปแบบของการพัฒนาซอฟต์แวร์ในปัจจุบันมีความแตกต่างจากในอดีตเป็นอย่างมากในอดีตระบบ IT จะเป็นระบบที่ไม่มีระบบย่อยมาก เช่น พัฒนาโปรแกรมจำนวน 1 โปรแกรม, พัฒนาเว็บไซต์จำนวน 1 เว็บไซต์ซึ่งสามารถบริหารจัดการให้เสร็จสิ้นโดยใช้โมเดลแบบ waterfall ให้เสร็จได้อย่างราบรื่นโดยที่ไม่มีการเปลี่ยนแปลงเนื้อหาหรือความต้องการ





🧸 แตกต่างจากระบบ IT ในปัจจุบันที่มีการทำงานแบบ micro-service มีการเชื่อมต่อกับระบบภายนอกมีจำนวนผู้ใช้หลายกลุ่มมีการใช้ API จากผู้ให้บริการที่หลากหลายอีกทั้งแรงกดดันจากคู่แข่งที่มีการพัฒนาเช่นเดียวกัน ส่งผลให้ความต้องการของระบบหรือความต้องการอาจต้องมีการเปลี่ยนแปลงในระหว่างทางของการพัฒนาซึ่งการพัฒนาซอฟต์แวร์ตามแนวคิดแบบ Agile จะรองรับและสนับสนุนการทำงานในลักษณะนี้

🧸 ยังถูกใช้เป็นวิธีการทำงานขององค์กรในภาพรวมหรือในบางองค์กรจะมีวัฒนธรรมทำงานตามโมเดลนี้ โดยโมเดลนี้จะยึดผู้นำของทีมหรือผู้บริหารองค์กรเป็นคนสำคัญซึ่งมีหน้าที่นำพาทิศทางหรือทีมไปสู่จุดหมาย ความต้องการของลูกค้าหรือผู้ถือเป็นคนสำคัญอันดับสูงสุดของโมเดลนี้อีกทั้งยังเป็นการทำงานกันเป็นรายวัน


🧸 ขั้นตอนใน agile มี 6 ขั้นตอน ได้แก่

- 🌸 Requirements รวบรวมความต้องการของระบบ
- 🌸 Design ออกแบบระบบ
- 🌸 Development พัฒนาระบบ
- 🌸 Testing ทดสอบระบบ
- 🌸 Deployment ติดตั้งระบบหรือนำไปใช้
- 🌸 Review รับ feedback จากลูกค้าหรือผู้ใช้


## ข้อดี


-  ทุกคนในทีมได้มีส่วนร่วมและได้มีโอกาสได้พูดคุยสื่อสารกัน
-  สนับสนุนการได้มาซึ่ง feedback จากลูกค้า
-  มีความยืดหยุ่นรองรับการปรับเปลี่ยนความต้องการ
-  มีการติดต่อสื่อสารกับผู้ใช้หรือลูกค้าอยู่เสมอ

## ข้อจำกัด





 การระบุ deadline ที่แน่นอนอาจทำได้ยาก เนื่องจากการพูดคุยและการพูดคุยนำมาซึ่งการเปลี่ยนแปลง ดังนั้นยังมีการพูดคุยกับลูกค้ามากขึ้นอาจส่งผลให้มีการปรับเปลี่ยนกำหนดการบ่อยขึ้น เช่น งานเสร็จเร็วกว่ากำหนดหรือมีการเพิ่มงาน

 อาจเป็นการยากที่จะให้ทุกคนในทีมงานหรือองค์กรมีปฏิสัมพันธ์กันเพื่อให้การพัฒนาซอฟต์แวร์ไปต่อให้

 จากคำกล่าวที่ว่า agile นั้นคาดการณ์ deadline ได้ยากเนื่องจากเมื่อได้ความต้องการแล้วจะเข้าสู่ของการพัฒนาซอฟต์แวร์และเมื่อถึงขั้นตอนสุดท้ายที่มี review เพื่อรับ feedback จากผู้ใช้ ผู้ใช้อาจยังไม่พอใจกับงานที่ได้จึงเกิดเป็นความต้องการเพิ่มเติมซึ่งก็จะวนกลับเข้าสู่อีกรอบแต่ถ้าหากลูกค้ามีความพึงพอใจแล้วก็จะถือว่าเสร็จสิ้นกระบวนการพัฒนาซอฟต์แวร์ของ agile

 จากภาพโมเดลจะเห็นว่าลักษณะการทำงานคล้าย spiral model แตกต่างที่ agile จะต้องมีส่วนได้ส่วนเสียหรือลูกค้าหรือผู้ใช้เข้ามามีส่วนร่วมในขั้นตอนต่างๆ อยู่เสมอมากที่สุดเท่าที่จะเป็นไปได้ อีกทั้งโมเดลนี้ยังเน้นไปที่การพัฒนาระบบ ปรับปรุงและพัฒนาไปจนกว่าลูกค้าจะพอใจต่างกับสองโมเดลก่อนหน้านี้ที่จะพัฒนาระบบให้เสร็จสิ้นในการทำงานครั้งเดียวซึ่งโมเดลนี้จะสามารถประกอบไปด้วยทีมมากกว่าหนึ่งทีมก็ได้ที่มีส่วนร่วมในการพัฒนาซอฟต์แวร์ตัวเดียวกันแต่มีผู้นำเพียงหนึ่งคนที่คอยบริหารจัดการ

 แบ่งกลุ่มบุคคลในการทำงานแบบ agile ได้ดังนี้

-  Stakeholders = ผู้มีส่วนได้ส่วนเสีย เช่น ลูกค้า ผู้ใช้งาน
-  Developer = นักพัฒนาระบบ
-  Product Owner = เจ้าของระบบ
-  Project Manager = ผู้นำทีมพัฒนาระบบ