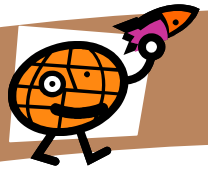


วิธีปฏิบัติของวิศวกรซอฟต์แวร์

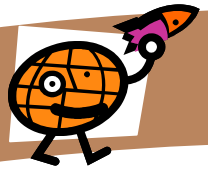
วิธีปฏิบัติด้านวิศวกรรมซอฟต์แวร์ (Software Engineering Practice)

เราทราบว่ากิจกรรมของการพัฒนาซอฟต์แวร์พื้นฐานประกอบไปด้วยอะไรบ้าง ดังนั้นนักวิศวกรซอฟต์แวร์จึงต้องเตรียมตัวและนำวิธีปฏิบัติทางด้านวิศวกรรมซอฟต์แวร์มาประยุกต์ใช้ในกิจกรรมดังกล่าว ดังรายละเอียดต่อไปนี้

ความสำคัญของการปฏิบัติ (The Essence of Practice)

George Polya ได้กล่าวถึงสาระสำคัญของการแก้ปัญหา ซึ่งต่อมาได้เป็นความสำคัญของการปฏิบัติด้านวิศวกรรมซอฟต์แวร์ ไว้ดังนี้

1. เข้าใจปัญหา (สื่อสารและการวิเคราะห์)
2. วางแผนการแก้ปัญหา (สร้างแบบจำลองและการออกแบบซอฟต์แวร์)
3. ปฏิบัติตามแผน (การเขียนโปรแกรม)
4. ตรวจดูความถูกต้องของผลที่ได้ (การทดสอบและประเมินคุณภาพ)



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านวิศวกรรมซอฟต์แวร์ (Software Engineering Practice)

หลักการสำคัญ (Core Principles)

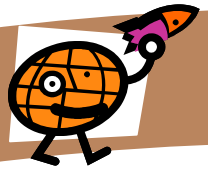
Devid Hooker ได้เสนอหลักการสำคัญ 7 ประการของวิธีปฏิบัติด้านวิศวกรรมซอฟต์แวร์ ดังนี้

หลักที่ 1 **เหตุผลที่ต้องการ (The reason it all exists)**
ก่อนเริ่มโครงการซอฟต์แวร์ใด ๆ ต้องแน่ใจว่าซอฟต์แวร์นั้นมีวัตถุประสงค์ทางธุรกิจที่ชัดเจนและผู้ใช้สามารถรับรู้ถึงคุณประโยชน์ของมันได้

หลักที่ 2 **เน้นความเรียบง่าย (Keep IT Simple, Stupid! หรือ KISS)**
ควรทำการออกแบบซอฟต์แวร์แบบง่ายๆ ธรรมดา ๆ ที่สุด ใช้ระบบที่เข้าใจง่าย ดูแลรักษาง่าย เน้นความเรียบง่ายเป็นสำคัญ

หลักที่ 3 **รักษาวิสัยทัศน์ (Maintain the Vision)**
วิสัยทัศน์ที่ชัดเจนเป็นส่วนสำคัญต่อความสำเร็จของโครงการซอฟต์แวร์

หลักที่ 4 **สิ่งที่คุณผลิตจะมีคนใช้ (What you produce, Others will consume)**
ต้องระลึกไว้อยู่เสมอว่าระบบซอฟต์แวร์ที่คุณสร้างขึ้นมาจะต้องมีคนนำไปใช้ดูแลรักษาหรือจัดทำเอกสารประกอบ ดังนั้น จะต้องทำให้ผู้ใช้เหล่านั้นสามารถเข้าใจสิ่งที่คุณสร้างเพื่อนำไปใช้ประโยชน์ต่อไป



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านวิศวกรรมซอฟต์แวร์ (Software Engineering Practice)

หลักการสำคัญ (Core Principles) ต่อ.

หลักที่ 5 **เปิดรับอนาคต (Be Open to the Future)**

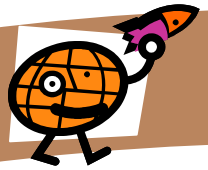
ในสภาพแวดล้อมทางเทคโนโลยีคอมพิวเตอร์ทุกวันนี้ การเปลี่ยนแปลงในเรื่องคุณลักษณะ (Specification) และแพลตฟอร์มเพราะฮาร์ดแวร์เปลี่ยนแปลงเร็วมาก และซอฟต์แวร์อาจมีอายุสั้นลง ล้าสมัยเร็วขึ้น อย่างไรก็ตามซอฟต์แวร์ในระดับอุตสาหกรรมต้องสามารถอยู่ได้นาน ซึ่งต้องขึ้นอยู่กับการออกแบบตั้งแต่ต้น ให้ครอบคลุมปัญหาต่าง ๆ ที่อาจเกิดขึ้นได้ในอนาคต และหาทางแก้ไขเท่าที่เป็นไปได้

หลักที่ 6 **วางแผนการนำมาใช้ใหม่ล่วงหน้า (Plan Ahead for Reuse)**

การนำมาใช้ใหม่ช่วยประหยัดเวลาและแรงงาน และประสบผลความสำเร็จในระดับที่สูง แต่ก็เป็นเรื่องยากสำหรับการพัฒนาระบบซอฟต์แวร์

หลักที่ 7 **คิด! (Think!)**

คิดให้รอบคอบก่อนลงมือทำสิ่งใด ๆ ซึ่งการหยุดคิดสักนิดอาจจะช่วยให้ได้ผลลัพธ์ที่ดีกว่า



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

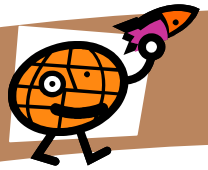
วิธีปฏิบัติด้านการวางแผน (Planning Practice)

กิจกรรมการวางแผนรวมวิธีปฏิบัติทั้งทางด้านการจัดการและด้านเทคนิคเข้าไว้ด้วยกัน โดยเป็นตัวช่วยให้ทีมงานซอฟต์แวร์สามารถกำหนดแผนการเดินทาง (Road Map) ให้มุ่งไปสู่เป้าหมายเชิงกลยุทธ์และวัตถุประสงค์ระดับกลวิธี เนื่องจากการวางแผนเป็นสิ่งจำเป็น เพื่อให้แผนเป็นเสมือนตัวนำทางให้กับทุกคนในทีม โดยใช้หลักต่อไปนี้นำไปประยุกต์ใช้กับการวางแผนทุกระดับ

เน้นหลักการสำคัญ (Core Principles) ในการวางแผน

หลักที่ 1 **เข้าใจขอบข่ายของโครงการ (Understand the scope of the project)**
มันเป็นไปไม่ได้เลยที่จะใช้แผนการเดินทาง (Road Map) หากคุณไม่รู้ว่ากำลังเดินทางไปไหน ขอบข่าย (scope) เป็นตัวบ่งบอกว่าจุดหมาย (Destination) คืออะไร

หลักที่ 2 **นำลูกค้าเข้ามามีส่วนร่วมในการวางแผน (Involve the customer in the planning activity)**
ลูกค้าเป็นผู้กำหนดลำดับความสำคัญก่อนหลังของงาน ซึ่งเป็นข้อจำกัดของโครงการจึงต้องให้ลูกค้าเข้ามามีส่วนร่วมในการวางแผน



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการวางแผน (Planning Practice)

เน้นหลักการสำคัญ (Core Principles) ในการวางแผน (ต่อ)

หลักที่ 3 **ระลึกอยู่เสมอว่าการวางแผนต้องมีการทบทวน (Recognize that planning is iterative)**

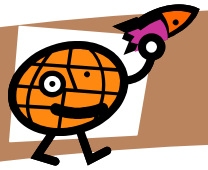
เนื่องจากหลายสิ่งหลายอย่างเปลี่ยนแปลงได้ เมื่อเริ่มลงมือปฏิบัติจริง ดังนั้น เราจะต้องมีการทบทวนและปรับแผนให้สอดคล้องกับสถานการณ์ใหม่

หลักที่ 4 **คาดการณ์จากสิ่งที่คุณทราบ (Estimate based on what you know)**

จุดมุ่งหมายของการประมาณ เพื่อเป็นสิ่งสะท้อนถึงความพยายาม ต้นทุน และเวลาที่ใช้ในแต่ละงานย่อย โดยดูจากความเข้าใจงานของทีมเป็นหลัก หากข้อมูลที่มีไม่ชัดเจนหรือเชื่อถือไม่ได้ ผลการประมาณต่าง ๆ ก็จะเชื่อถือไม่ได้เช่นกัน

หลักที่ 5 **ในระหว่างการวางแผน จงพิจารณาความเสี่ยงของโครงการ (Consider risk as you define the plan)**

หากพบว่ามีความเสี่ยงที่อาจก่อนให้เกิดผลกระทบอย่างมากต่อโครงการ จะต้องเผื่อการวางแผนสำรองเอาไว้ด้วย โดยแผนโครงการและตารางการทำงานจะต้องได้รับการปรับเปลี่ยนให้สอดคล้องกับโอกาสที่ความเสี่ยงจะเกิดขึ้นด้วย



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการวางแผน (Planning Practice)

เน้นหลักการสำคัญ (Core Principles) ในการวางแผน (ต่อ)

หลักที่ 6 **ยอมรับความจริง! (Be realistic!)**

ไม่มีใครสามารถให้เต็มร้อยกับงานได้ทุกวัน นักปราชญ์ก็ยังสามารถทำพลาดได้ นอกจากนี้ยังอาจมีการละเลยงาน ความกำกวม และสิ่งรบกวนอื่น ๆ ที่อาจส่งผลต่อโครงการได้ ดังนั้น จึงควรคิดถึงสิ่งเหล่านี้ไว้ด้วยในขณะทำการวางแผน

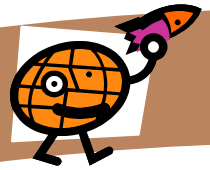
หลักที่ 7 **ปรับสภาพหยาบ (Adjust granularity)**

สภาพหยาบ (Granularity) หมายถึงระดับรายละเอียดของการวางแผนโครงการ

แผนละเอียด (Fine Granularity Plan) จะทำให้รายละเอียดของงานย่อยค่อนข้างมากในช่วงเวลาสั้น (จึงทำให้สามารถตรวจสอบและควบคุมได้)

แผนหยาบ (Coarse Granularity Plan) กล่าวถึงงานย่อยที่กว้างขึ้นครอบคลุมช่วงเวลาที่นานขึ้น

โดยทั่วไป สภาพหยาบจะปรับเปลี่ยนจากละเอียดไปสู่หยาบขึ้น เมื่อเวลาเคลื่อนที่ไปจากปัจจุบัน แต่อย่างไรก็ตาม เราอาจเขียนแผนโดยละเอียดได้สำหรับสองสามสัปดาห์หรือสองสามเดือนข้างหน้า แต่กิจกรรมหลายเดือนข้างหน้านั้นไม่มีความจำเป็นที่ต้องวางแผนไว้ละเอียด



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

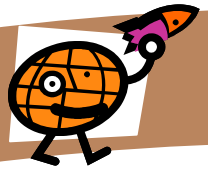
วิธีปฏิบัติด้านการวางแผน (Planning Practice)

เน้นหลักการสำคัญ (Core Principles) ในการวางแผน (ต่อ)

หลักที่ 8 **กำหนดว่าจะประกันคุณภาพอย่างไร (Define how to ensure quality)**
ควรกำหนดไว้ในแผนด้วยว่าจะดูแลในเรื่องของคุณภาพอย่างไร ถ้าจะต้องมีการพิจารณาในเชิงเทคนิค ก็ควรกำหนดเวลาให้แน่นอนลงไป

หลักที่ 9 **ควรมีแนวทางในการรับมือกับการเปลี่ยนแปลง (Describe how you intend to accommodate change)**
ในระหว่างที่งานวิศวกรรมซอฟต์แวร์กำลังดำเนินไป ทีมงานควรกำหนดแนวทางในการรับมือกับความเปลี่ยนแปลงที่อาจเกิดขึ้น เช่น กำหนดไว้ว่าลูกค้าปรับเปลี่ยนเรื่องต่าง ๆ ได้ทุกเวลาหรือไม่ หากยอมให้ลูกค้าเปลี่ยนแปลง เราต้องปฏิบัติทันทีหรือไม่ ผลกระทบและค่าใช้จ่ายที่เกิดขึ้นจากการเปลี่ยนแปลงนั้นเป็นอย่างไร

หลักที่ 10 **เฝ้าติดตามแผนและทำการปรับเปลี่ยนตามความเหมาะสม (Track to plan frequently and make adjustments as required)**
ต้องคอยเฝ้าดูความคืบหน้าของโครงการทุกวัน เพื่อเฝ้าระวังปัญหาที่อาจเกิดขึ้นและส่งผลต่อตารางเวลา



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการวางแผน (Planning Practice)

Barry Boehm เสนอแนวทางที่จะจัดการกับวัตถุประสงค์ของโครงการ หลักไมล์ และตารางเวลา ความรับผิดชอบ แนวทางการบริหารด้านการจัดการและด้านเทคนิค และทรัพยากรที่จำเป็นต่อโครงการ โดยให้ชื่อว่า “กฎ W^5HH” (W^5HH Principle) จากชุดคำถามต่อไปนี้

ทำไมถึงต้องพัฒนาระบบนี้ (Why is the system being developed?)

ควรมีเหตุผลที่เหมาะสมสำหรับผู้เกี่ยวข้องทุกคน ในงานซอฟต์แวร์ที่จะทำ อีกนัยหนึ่งคือการถามว่า มันคุ้มกับแรงงาน เวลา และเงินที่จะลงไปหรือไม่

จะทำอะไร (What will be done?)

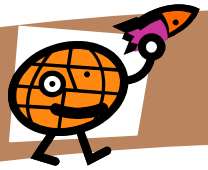
บอกว่าหน้าที่หลัก ๆ ของซอฟต์แวร์ที่ต้องการคืออะไร

จะทำเสร็จเมื่อไร (When will it be accomplished?)

เขียนลำดับงานและเส้นเวลาของแต่ละงานย่อยที่สำคัญ ระบุหลักชัยต่าง ๆ ที่ลูกค้าต้องการ

ใครรับผิดชอบหน้าที่อะไร (Who is responsible for a function?)

ระบุบทบาทและหน้าที่ของสมาชิกในทีมซอฟต์แวร์



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการวางแผน (Planning Practice)

กฎ W^5HH (ต่อ)

พวกเขาอยู่ที่ใดกันบ้างในองค์กร (Where are they organizationally located?)

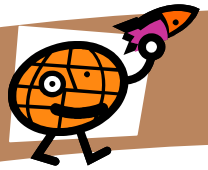
บทบาทและหน้าที่บางอย่างอาจจไม่ได้อยู่กับทีมงานผู้ผลิตซอฟต์แวร์ ลูกค้า ผู้ใช้ และผู้มีส่วนใดส่วนเสียอื่น ๆ ต่าง ก็มีความรับผิดชอบของตัวเองเช่นกัน

งานในเชิงเทคนิคและเชิงบริหารจะบรรลุได้อย่างไร (How will the job be done technically and managerially?)

เมื่อได้กำหนดขอบข่ายของโครงการแล้ว จะต้องทำการระบุกลยุทธ์เชิงเทคนิคและเชิงบริหารไว้ด้วย

จะต้องใช้ทรัพยากรต่าง ๆ มากน้อยเท่าใด (How much of each resource is needed?)

คำตอบของปัญหานี้ได้จากการพยายามประมาณทรัพยากรต่าง ๆ ที่ต้องใช้ในโครงการ โดยใช้ข้อมูลจากคำตอบของคำถามก่อนหน้านี้เป็นฐาน



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการสื่อสาร (Communication Practice)

กิจกรรมการติดต่อสื่อสารเป็นสิ่งที่ช่วยให้การรวบรวมความต้องการของลูกค้าเป็นไปได้ การติดต่อสื่อสารที่มีประสิทธิภาพ นับเป็นสิ่งที่ท้าทายมากที่สุด

เน้นหลักการสำคัญ (Core Principles) ในการสื่อสารกับลูกค้า

หลักที่ 1 ฟัง! (Listen!)

จงให้ความสำคัญกับทุกคำของลูกค้า ถ้ามีข้อสงสัยให้สอบถามเพื่อความกระจ่าง

หลักที่ 2 เตรียมตัวให้พร้อมก่อนเริ่มการติดต่อสื่อสาร (Prepare before you communicate)

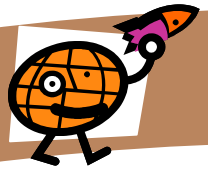
พยายามที่จะเข้าใจปัญหาด้วยตนเองก่อน โดยอาจต้องมีการหาข้อมูลเพิ่มเติม

หลักที่ 3 ควรมีผู้ประสานงาน (Someone should facilitate the activity)

ในทุก ๆ การติดต่อสื่อสาร ควรจะมีผู้นำที่ทำหน้าที่ในการควบคุมการสนทนาให้มุ่งไปในทิศทางที่เกิดผล และเป็นผู้ประนีประนอมความขัดแย้งที่อาจเกิดขึ้น

หลักที่ 4 การติดต่อสื่อสารแบบพบหน้ากันดีที่สุด (Face-to-face communication is best)

โดยเฉพาะถ้ามีการเตรียมเอกสารประกอบอย่างดี เช่น โครงร่างของเรื่องที่จะพูด



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการสื่อสาร (Communication Practice)

เน้นหลักการสำคัญ (Core Principles) ในการสื่อสารกับลูกค้า (ต่อ)

หลักที่ 5 จดรายละเอียดการพูดคุยรวมถึงข้อตกลงในเรื่องต่าง ๆ (Take notes and document decision)

หลักที่ 6 พยายามให้เกิดความร่วมมือ (Strive for collaboration)

ความร่วมมือกันระหว่างสมาชิกในทีม เป็นช่วยก่อให้เกิดความไว้วางใจซึ่งกันและกัน อันจะนำไปสู่เป้าหมายร่วมกันของทีมได้

หลักที่ 7 เน้นหัวข้อ ไม่แตกประเด็น (Stay focused, modularize your discussion)

การติดต่อสื่อสารใด ๆ ก็ตาม ยิ่งมีผู้เกี่ยวข้องมาก ก็ยิ่งมีโอกาสขัดแย้งมากหรือวนไปวนมา ต้องพยายามอย่าให้การสนทนาออกนอกประเด็น

หลักที่ 8 ถ้ามีอะไรไม่ชัดเจน ให้วาดภาพประกอบ (If something is unclear, draw a picture)

กรณีที่การสื่อสารด้วยคำพูดทำให้เกิดความไม่ชัดเจน การวาดภาพ ตาราง หรือใช้รูปประกอบ อาจช่วยได้

หลักที่ 9 พยายามให้การติดต่อสื่อสาร “เดินหน้า” (Move on)

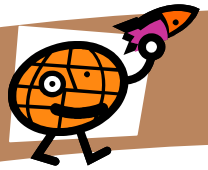
ไม่ว่าจะได้ข้อสรุปหรือไม่ หรือว่ามีบางเรื่องที่ยังขาดความชัดเจน ก็ให้การสื่อสารเดินหน้าไปก่อน อย่าพูดซ้ำไปซ้ำมากหาข้อยุติไม่ได้ให้เปลืองเวลา

วิธีปฏิบัติด้านการสื่อสาร (Communication Practice)

เน้นหลักการสำคัญ (Core Principles) ในการสื่อสารกับลูกค้า (ต่อ)

หลักที่ 10 **การต่อรองไม่ใช่การแข่งขันหรือเกม จะเป็นการดีที่สุด หากทั้งสองฝ่ายเป็นผู้ชนะ (Negotiation is not a contest or a game, It works best when both parties win)**

มีหลาย ๆ เรื่องที่นักวิศวกรซอฟต์แวร์ต้องต่อรองกับลูกค้า เช่น งานที่ซอฟต์แวร์ทำได้ ลำดับงานก่อนหลัง กำหนดวันส่งมอบ ทั้งนี้ จะต้องมีการร่วมมือร่วมใจสร้างเป้าหมายร่วมกัน การต่อรองต้องอาศัยการประนีประนอมจากทั้งสองฝ่าย



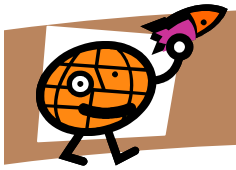
วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการสร้างแบบจำลอง (Modeling Practice)

ด้านวิศวกรรมซอฟต์แวร์ เราแบ่งแบบจำลองออกเป็น 2 ประเภท คือ

1) **แบบจำลองการวิเคราะห์ (Analysis Model)**
   เป็นแบบจำลองที่เกิดจากการวิเคราะห์และสรุปความต้องการของลูกค้า เพื่อกำหนดทิศทางในการพัฒนาซอฟต์แวร์ กระบวนการสร้างแบบจำลองเพื่อใช้ในการวิเคราะห์ต้องทำอย่างถูกต้องและชัดเจน
2) **แบบจำลองการออกแบบ (Design Model)**
   เป็นกระบวนการที่นำความต้องการของลูกค้าที่สรุปชัดเจนแล้วมาสร้างเป็นแบบจำลอง โดยบ่งบอกถึงลักษณะทางสถาปัตยกรรมส่วนต่อประสานผู้ใช้ (User Interface) และรายละเอียดในระดับองค์ประกอบย่อย (Component Level)

โดยการออกแบบต้องคำนึงถึงปัจจัยภายในและภายนอกระบบ เนื่องจากเป็นตัวแปรสำคัญที่จะส่งผลถึงระดับคุณภาพของระบบ ผู้พัฒนาจึงควรเข้าใจพื้นฐานของการออกแบบอย่างลึกซึ้งก่อนจะทำให้ระบบที่ได้มีคุณภาพในระดับสูง



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการสร้างแบบจำลอง (Modeling Practice)

หลักการสำคัญในการสร้างแบบจำลองการวิเคราะห์ (Analysis Modeling Principles)

หลักที่ 1 **ต้องแสดงและเข้าในโดเมนสารสนเทศของปัญหา (The information domain of a problem must be represented and understood)**

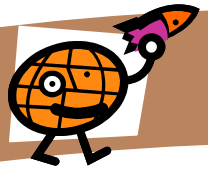
เนื่องจากหลายสิ่งหลายอย่างเปลี่ยนแปลงได้ เมื่อเริ่มลงมือปฏิบัติจริง ดังนั้น เราจะต้องมีการทบทวนและปรับแผนให้สอดคล้องกับสถานการณ์ใหม่

หลักที่ 2 **ต้องระบุฟังก์ชันที่ซอฟต์แวร์ทำงานได้ (The Function that the software performs must be defined)**

ฟังก์ชันที่ซอฟต์แวร์ทำงานได้นับเป็นประโยชน์โดยตรงต่อผู้ใช้สุดท้าย และเป็นตัวสนับสนุนลักษณะอื่น ๆ ซึ่งผู้ใช้สามารถมองเห็นได้ บางฟังก์ชันทำการแปลงข้อมูลเข้าสู่ระบบ ในขณะที่บางฟังก์ชันส่งผลกระทบต่อระดับการควบคุมการประมวลผลซอฟต์แวร์ภายในหรือองค์ประกอบภายนอก

หลักที่ 3 **บ่งบอกถึงพฤติกรรมของซอฟต์แวร์เมื่อมีเหตุการณ์ภายนอกมากระทบ (The software behavior as a consequence of external events must be represented)**

ตัวอย่างของเหตุการณ์ภายนอกดังกล่าว ได้แก่ ข้อมูลเข้าที่ป้อนโดยผู้ใช้สุดท้าย ข้อมูลควบคุมที่ได้จากระบบภายนอก หรือข้อมูลที่ได้จากการเฝ้าระวังผ่านระบบเครือข่าย



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการสร้างแบบจำลอง (Modeling Practice)

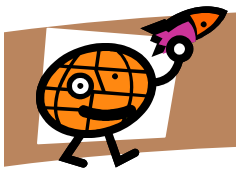
Analysis Modeling Principles (ต่อ)

หลักที่ 4 แบบจำลองที่อธิบายถึงสารสนเทศ ฟังก์ชัน และพฤติกรรม (The Models that depict information, function, and behavior must be partitioned in a manner that uncovers detail in a layered fashion)

แบบจำลองการวิเคราะห์เป็นขั้นแรกในการแก้ปัญหาทางวิศวกรรมซอฟต์แวร์ มันช่วยให้ผู้พัฒนาซอฟต์แวร์เข้าใจปัญหาได้ดีขึ้น และช่วยในการออกแบบต่อไป ปัญหาที่ซ้ำซ้อนต้องใช้วิธีแบ่งแยกและปกครอง (Divide and Conquer) เข้าช่วยในการจัดการ โดยอาจแบ่งออกเป็นปัญหาที่เล็กลง ๆ จนกว่าจะกลายเป็นปัญหาย่อยที่เข้าใจง่ายขึ้น

หลักที่ 5 ง่ายย่อยของการวิเคราะห์ควรเริ่มจากสารสนเทศที่สำคัญ ไปสู่รายละเอียดของการนำไปใช้จริง (The analysis task should move from essential information toward implementation detail)

แบบจำลองการวิเคราะห์เริ่มจากการอธิบายปัญหาในมุมมองของผู้ใช้สุดท้าย โดยจะอธิบาย “สาระสำคัญของปัญหา” ดังนั้นต้องเข้าถึงส่วนสำคัญของปัญหาและมีรายละเอียดของเครื่องมือที่ใช้อย่างสมบูรณ์



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการสร้างแบบจำลอง (Modeling Practice)

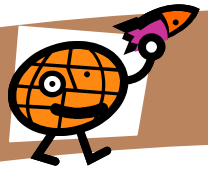
หลักการสำคัญในการสร้างแบบจำลองการออกแบบ (Design Modeling Principles)

หลักที่ 1 **เราต้องมองการออกแบบย่อยกลับไปหาแบบจำลองการวิเคราะห์ได้ (Design should be traceable to the analysis model)**

แบบจำลองการออกแบบต้องแปลงสารสนเทศในแบบจำลองการวิเคราะห์ไปเป็นสถาปัตยกรรม ซึ่งหมายถึง ชุดของระบบย่อย ซึ่งนำฟังก์ชันหลัก ๆ ไปปฏิบัติจริง และหมายถึงชุดการออกแบบระดับองค์ประกอบย่อย ซึ่งการนำการวิเคราะห์ไปทำให้เป็นจริง องค์ประกอบต่าง ๆ ของการออกแบบนี้ควรจะตรวจสอบกลับไปหาแบบจำลองการวิเคราะห์ได้ ยกเว้นในส่วนการออกแบบที่เกี่ยวข้องกับโครงการพื้นฐานของซอฟต์แวร์

หลักที่ 2 **พิจารณาสถาปัตยกรรมของระบบที่สร้าง (Always consider the architecture of the system to be built)**

สถาปัตยกรรมซอฟต์แวร์เป็นเหมือนโครงร่างของระบบ มันมีผลกับส่วนต่อประสานต่าง ๆ โครงสร้างข้อมูล พฤติกรรม และการไหลของการควบคุม โปรแกรม ดังนั้นการออกแบบควรเริ่มด้วยการพิจารณาด้านสถาปัตยกรรมก่อน จากนั้นค่อยไปทำในระดับองค์ประกอบย่อยต่อไป



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการสร้างแบบจำลอง (Modeling Practice)

Design Modeling Principles (ต่อ)

หลักที่ 3 **ในระหว่างการวางแผน จงพิจารณาความเสี่ยงของโครงการ (Consider risk as you define the plan)**

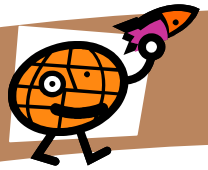
การออกแบบข้อมูลเป็นส่วนสำคัญของการออกแบบสถาปัตยกรรม การออกแบบข้อมูลอย่างมีระบบจะทำให้การไหลของโปรแกรมง่ายขึ้น ทำให้การออกแบบและการอิมพลีเมนต์องค์ประกอบย่อยต่าง ๆ ของซอฟต์แวร์สะดวกขึ้น จนถึงการประมวลผลโดยรวมมีประสิทธิภาพมากขึ้น

หลักที่ 4 **ส่วนต่อประสานต้องได้รับการออกแบบอย่างถี่ถ้วน (Interfaces must be designed with care)**

ลักษณะการไหลของข้อมูลระหว่างองค์ประกอบย่อยของระบบมีผลต่อประสิทธิภาพการประมวลผล ส่วนต่อประสานที่ได้รับการออกแบบที่ดีจะช่วยให้สะดวกต่อการบูรณาการส่วนต่าง ๆ เข้าด้วยกัน และยังช่วยผู้ทดสอบในการตรวจสอบความถูกต้องของฟังก์ชันองค์ประกอบด้วย

หลักที่ 5 **ควรออกแบบส่วนต่อประสานกับผู้ใช้ให้สอดคล้องกับความต้องการ (User interface design should be tuned to the need of the end user)**

การออกแบบควรให้ความสำคัญกับความสะดวกในการใช้งานของผู้ใช้สุดท้าย เนื่องจากจัดเป็นหน้าตาของซอฟต์แวร์



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการสร้างแบบจำลอง (Modeling Practice)

Design Modeling Principles (ต่อ)

หลักที่ 6 **การออกแบบในระดับองค์ประกอบย่อยควรเป็นอิสระกันในเชิงหน้าที่ (Component-level design should be functionally independent)**

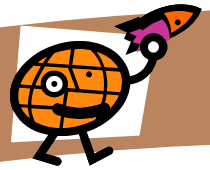
แต่ละองค์ประกอบย่อยควรเน้นหน้าที่เพียงหนึ่งเดียว

หลักที่ 7 **องค์ประกอบย่อยต่างๆ ควรขึ้นต่อกันเองให้น้อยที่สุด (Component should be loosely coupled to one another and to the external environment)**

ในขณะที่ระดับการขึ้นต่อกันหรือการเข้าคู่กัน (Coupling) ขององค์ประกอบย่อยเพิ่มขึ้นนั้น โอกาสที่เกิดความผิดพลาดก็จะเพิ่มขึ้นตาม ความสามารถในการดูแลรักษาระบบโดยรวมก้จะลดลงด้วย ดังนั้น ควรให้มีการขึ้นต่อกันน้อยที่สุด

หลักที่ 8 **แบบจำลองการออกแบบควรจะเป็นเข้าใจได้โดยง่าย (Design representations should be easily understandable)**

เป้าหมายของการออกแบบ เพื่อสื่อสารกับผู้เกี่ยวข้องกับฝ่ายต่าง ๆ ดังนั้น หากแบบนั้นยากต่อการเข้าใจ ก็อาจทำให้มันเป็นตัวกลางการสื่อสารที่ไม่มีประสิทธิภาพ



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติด้านการสร้างแบบจำลอง (Modeling Practice)

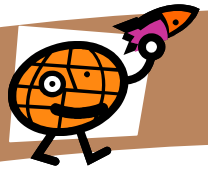
Design Modeling Principles (ต่อ)

หลักที่ 9 **ควรทำการออกแบบหลาย ๆ รอบ โดยแต่ละรอบ ต้องพยายามให้แบบเรียบง่ายขึ้น ไม่ซับซ้อน (The design should be developed iteratively)**

คล้ายกับงานสร้างสรรค์แบบอื่น ๆ ซึ่งต้องมีการพิจารณาทบทวนทำซ้ำ จนให้ได้มาซึ่งงานที่มีข้อบกพร่องน้อยที่สุด และในขณะเดียวกันก็ควรจะทำให้ไม่ซับซ้อนเกินไปด้วย

จากหลักสำคัญดังกล่าว จะช่วยให้วิศวกรซอฟต์แวร์ได้แบบที่มีคุณภาพทั้งในเรื่องปัจจัยภายในและภายนอก

- **ปัจจัยคุณภาพภายนอก (External Quality Factors)**
  หมายถึง คุณลักษณะของซอฟต์แวร์ที่เป็นที่สังเกตได้โดยผู้ใช้ เช่นความเร็ว ความเชื่อถือได้ ความถูกต้อง และการใช้งาน
- **ปัจจัยคุณภาพภายใน (Internal Quality Factors)**
  หมายถึง คุณลักษณะที่เป็นประโยชน์ต่อวิศวกรซอฟต์แวร์ ซึ่งช่วยนำไปสู่แบบที่ดีในเชิงเทคนิค ซึ่งผู้ออกแบบจะต้องเข้าหลักพื้นฐานในการออกแบบให้ถ่องแท้เสียก่อน



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

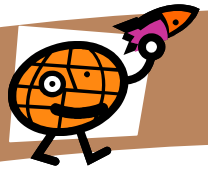
วิธีปฏิบัติในการสร้างซอฟต์แวร์ (Construction Practice)

กิจกรรมการสร้างซอฟต์แวร์จะรวมถึงการเขียนและทดสอบซอฟต์แวร์ที่ทำงานได้และพร้อมจะส่งมอบให้กับลูกค้า ในด้านวิศวกรรมสมัยใหม่นั้นการเขียนโปรแกรมอาจหมายถึง

1) การเขียนโค้ดโปรแกรมเองโดยตรง
2) การผลิตโค้ดโปรแกรมอัตโนมัติจากตัวกลางแทนแบบขององค์ประกอบย่อย
3) การผลิตโค้ดโปรแกรมอัตโนมัติ ที่สามารถเลือกเอ็กซิคิวได้โดยใช้ภาษาโปรแกรมยุคที่ 4 เช่น Rational Rose

สำหรับการทดสอบจะเน้นไปที่ระดับองค์ประกอบย่อย ซึ่งมักเรียกกันว่า “การทดสอบระดับหน่วย” (Unit Testing) นอกจากนี้ ยังมีการทดสอบระดับอื่น ได้แก่

1) การทดสอบรวม (Integration Testing)
2) การทดสอบการตรวจรับ (Validation Testing)
3) การทดสอบการยอมรับ (Acceptance Testing)



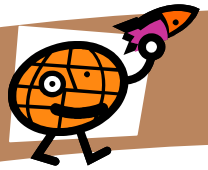
วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติในการสร้างซอฟต์แวร์ (Construction Practice)

แนวคิดและหลักในการเขียนโปรแกรม (Coding Principles and Concepts)

หลักการเตรียมพร้อม (Preparation Principles) : ก่อนเริ่มเขียนโปรแกรม เราต้อง

1) เข้าใจปัญหาที่กำลังพยายามหาทางแก้
2) เข้าใจหลักสำคัญในการออกแบบ
3) เลือกภาษาโปรแกรมที่สอดคล้องกับความต้องการของซอฟต์แวร์ที่จะสร้าง และสอดคล้องกับสภาพแวดล้อมของซอฟต์แวร์จะทำงาน
4) เลือกสภาพแวดล้อมด้านการโปรแกรม ซึ่งจะช่วยให้เราทำงานได้ง่าย
5) สร้างชุดของตัวทดสอบในระดับหน่วย (A Set of Unit Test) ซึ่งจะใช้กับองค์ประกอบย่อย หลังจากเขียนโปรแกรมเสร็จ



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติในการสร้างซอฟต์แวร์ (Construction Practice)

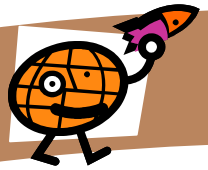
แนวคิดและหลักในการเขียนโปรแกรม (Coding Principles and Concepts)

หลักการเขียนโปรแกรม (Coding Principles) : ในระหว่างการเขียนโปรแกรม เราต้อง

1) เขียนอัลกอริทึมตามโครงสร้างของภาษาโปรแกรม
2) เลือกโครงสร้างข้อมูลซึ่งสอดคล้องกับความต้องการของระบบ (Design)
3) เข้าใจสถาปัตยกรรมซอฟต์แวร์และสร้างส่วนต่อประสานที่สอดคล้อง
4) ใช้คำสั่งตรรกะแบบมีเงื่อนไข (Conditional Logic) ให้ง่ายที่สุดเท่าที่จะทำได้
5) สร้างลูปซ้อนลูป (Nested Loop) ในแบบที่ง่ายต่อการทดสอบ
6) เลือกชื่อตัวแปรที่สื่อความและเป็นไปตามมาตรฐานการเขียนโปรแกรมที่ใช้กันอยู่ทั่วไป
7) เขียนโปรแกรมที่มีหลายเหตุช่วยในการอ่านโปรแกรมให้เข้าใจได้ในตัวเอง
8) จัดรูปแบบหน้าตา เช่น การย่อหน้า และการเว้นบรรทัด เพื่อช่วยให้เข้าใจง่าย

หลักการตรวจสอบความถูกต้อง (Validation Principles) : หลังเขียนโปรแกรมเสร็จ เราต้อง

1) ทำการอ่านโค้ดโปรแกรมตามความเหมาะสม
2) ทำการทดสอบระดับหน่วยและการแก้ไขข้อผิดพลาดที่พบ
3) จัดเรียงโค้ดโปรแกรมใหม่



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติในการสร้างซอฟต์แวร์ (Construction Practice)

หลักการทดสอบ (Testing Principles)

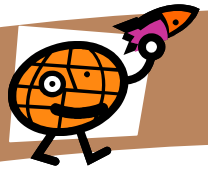
กฎโดยทั่วไปที่อาจยึดถือเป็นเป้าหมายของการทดสอบ คือ

- การทดสอบคือการเอ็กซิคิวโปรแกรมเพื่อหาข้อผิดพลาด
- กรณีทดสอบ (Test Case) ที่ดีจะต้องมีโอกาสสูงในการหาข้อผิดพลาดที่ยังไม่ทราบ
- การทดสอบที่ประสบความสำเร็จคือการทดสอบที่สามารถค้นพบข้อผิดพลาดที่ยังไม่ทราบมาก่อนได้

หลักในการทดสอบที่สำคัญ มีดังต่อไปนี้

หลักการที่ 1 **ทุกการทดสอบ ควรสามารถติดตามกลับไปยืนยันกับความต้องการของลูกค้าได้ (All tests should be traceable to customer requirements)**

วัตถุประสงค์ของการทดสอบซอฟต์แวร์ก็คือเพื่อค้นหาข้อผิดพลาด ดังนั้น ความบกพร่องที่ร้ายแรงที่สุดในสายตาของลูกค้า ก็คือการที่โปรแกรมไม่อาจทำงานได้ตามที่ระบุไว้ในความต้องการ



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติในการสร้างซอฟต์แวร์ (Construction Practice)

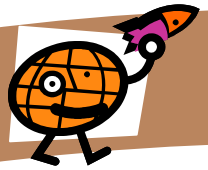
หลักในการทดสอบที่สำคัญ (ต่อ)

หลักการที่ 2 **ควรมีการวางแผนการทดสอบก่อนเริ่มการทดสอบจริง (Tests should be planned long before testing begins)**

การวางแผนการทดสอบอาจเริ่มได้ในทันทีที่แบบจำลองการวิเคราะห์เสร็จสมบูรณ์ การกำหนดรายละเอียดของแต่ละกรณีการทดสอบ (Test Case) อาจเริ่มได้ในทันทีที่แบบจำลองการออกแบบเป็นรูปเป็นร่าง ดังนั้นก่อนที่จะลงมือเขียนโปรแกรม ควรจะวางแผนและออกแบบการทดสอบไว้ล่วงหน้า

หลักการที่ 3 **ใช้กฎของพาเรโตในการทดสอบซอฟต์แวร์ (The Pareto principle applies to software testing)**

นำกฎ 80-20 ของพาเรโตมาประยุกต์ใช้ว่า 80% ของข้อผิดพลาดที่อาจในระหว่างการทำสอบจะถูกตรวจสอบกลับไปได้เพียง 20% ขององค์ประกอบย่อยทั้งหมด ปัญหาก็คือ การแยกองค์ประกอบย่อยที่น่าสงสัยเหล่านั้นออกมาเพื่อทดสอบอีกครั้งได้อย่างไร



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติในการสร้างซอฟต์แวร์ (Construction Practice)

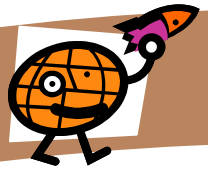
หลักในการทดสอบที่สำคัญ (ต่อ)

หลักการที่ 4 **การทดสอบควรตั้งต้นจาก “เล็ก” ไปหา “ใหญ่” (Testing should begin “in the small” and progress toward testing “in the large”)**

การทดสอบครั้งแรก ๆ โดยทั่วไปจะเน้นไปที่องค์ประกอบย่อย (Component) แต่ละส่วน จากนั้นจึงเพิ่มเป็นกลุ่มขององค์ประกอบย่อย แล้วจึงไปทดสอบทั้งระบบ

หลักการที่ 5 **เป็นไปไม่ได้ที่จะทดสอบทั้งหมดโดยละเอียด (Exhaustive testing is not possible)**

เป็นไปไม่ได้ที่จะทดสอบทั้งโปรแกรมในทุกเส้นทางการทำงาน แต่สิ่งที่เป็นไปได้ก็คือ การทดสอบที่ต้องครอบคลุมทุกตรรกะของโปรแกรม และทุกเงื่อนไขในระดับองค์ประกอบย่อย



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติในการนำไปใช้งาน (Deployment Practice)

กิจกรรมการนำไปใช้งานประกอบด้วย การกระทำ 3 ประการคือ

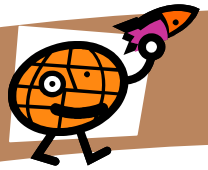
1) การส่งมอบ (Delivery)
2) การสนับสนุน (Support)
3) การป้อนกลับ (Feedback)

เนื่องจากแบบจำลองกระบวนการซอฟต์แวร์มีลักษณะปรับเปลี่ยนไปเรื่อย ๆ กิจกรรมการนำไปใช้งานจึงเกิดหลาย ๆ ครั้งในช่วงอายุซอฟต์แวร์แต่ละ**รอบการส่งมอบ** ลูกค้าจะได้ซอฟต์แวร์ที่ทำงานได้ และในแต่ละ**รอบสนับสนุน** เอกสารประกอบและบุคลากรฝ่ายสนับสนุนจะเข้าช่วยเหลือลูกค้าระหว่างรอบการนำไปใช้งานทั้งหมด ในขณะที่**รอบการป้อนกลับ** จะเป็นแนวทางในการปรับปรุงหน้าที่ และลักษณะของซอฟต์แวร์ให้ดีขึ้น

หลักการที่พึงปฏิบัติในช่วงการส่งมอบซอฟต์แวร์มีดังนี้

หลักการที่ 1 **ต้องบริหารความคาดหวังของลูกค้าที่มีต่อซอฟต์แวร์ได้ (Customer expectations for the software must be managed)**

บ่อยครั้งที่ลูกค้ามักคาดหวังมากกว่าที่ตกลงกันไว้ ทำให้มักเกิดความผิดหวังตามมา ผลก็คือ ได้การป้อนกลับที่ไม่เป็นประโยชน์ ดังนั้น วิศวกรซอฟต์แวร์ควรต้องระวังในเรื่องการสื่อสารกับลูกค้า



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติในการนำไปใช้งาน (Deployment Practice)

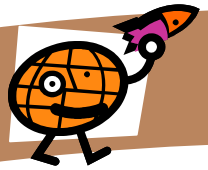
หลักการที่พึงปฏิบัติในช่วงการส่งมอบซอฟต์แวร์ (ต่อ)

หลักการที่ 2 **ต้องรวบรวมและทดสอบสินค้าที่จะส่งมอบได้ครบถ้วน (A complete delivery package should be assembled and tested)**

โปรแกรมที่ทำงานได้ ไฟล์ข้อมูลสนับสนุน เอกสารประกอบ และข้อมูลอื่น ๆ ที่เกี่ยวข้อง ต้องได้รับการรวบรวมและทดสอบในเบื้องต้นกับผู้ใช้จริง และภายใต้โครงแบบ (Configurations) ที่เป็นไปได้ทุกแบบ เช่น ฮาร์ดแวร์ ระบบปฏิบัติการ หรือ อุปกรณ์ต่อพ่วงอื่น ๆ

หลักการที่ 3 **ต้องมีข้อกำหนดที่ชัดเจนในงานการสนับสนุน ก่อนการส่งมอบ (A support regime must be established before the software is delivered)**

เมื่อผู้ใช้ประสบปัญหา เขาต้องการคำอธิบายที่ถูกต้องและรวดเร็ว ดังนั้น ควรมีการวางแผนงานสนับสนุนไว้ล่วงหน้า อุปกรณ์ช่วยเหลือต่าง ๆ และควรมีการจดบันทึกปัญหาที่ลูกค้าประสบไว้ด้วย



วิธีปฏิบัติของวิศวกรซอฟต์แวร์

วิธีปฏิบัติในการนำไปใช้งาน (Deployment Practice)

หลักการที่พึงปฏิบัติในช่วงการส่งมอบซอฟต์แวร์ (ต่อ)

หลักการที่ 4 **ต้องจัดเตรียมสื่อการใช้ซอฟต์แวร์ ให้กับผู้ใช้**
(Appropriate instructional materials must be provided to end-user)
ทีมงานซอฟต์แวร์ต้องส่งมอบมากกว่าตัวซอฟต์แวร์ เราต้องพัฒนาสื่อช่วยสอนต่าง ๆ ให้แนวทางการแก้ปัญหาที่อาจเกิดขึ้น และข้อมูลเกี่ยวกับซอฟต์แวร์ที่แตกต่างไปจากการส่งมอบซอฟต์แวร์ครั้งที่ผ่านมา

หลักการที่ 5 **หากยังพบจุดบกพร่องในซอฟต์แวร์อยู่ ต้องแก้ไขก่อน ห้ามส่งมอบ**
(Buggy software should be fixed first, deliver later)
ภายใต้เงื่อนไขด้านเวลาที่จำกัด บางบริษัทซอฟต์แวร์อาจส่งมอบสินค้าคุณภาพต่ำ โดยทำการแจ้งเตือนลูกค้าถึงจุดบกพร่อง (Bugs) ที่มีว่า “จะแก้ไขในครั้งถัดไป” นี่เป็นการกระทำที่ไม่ถูกต้อง มีคำกล่าวกันว่า “ลูกค้าจะไม่จำว่าคุณส่งสินค้าที่ดีช้าไปเล็กน้อย แต่ลูกค้าจะไม่มีวันลืมปัญหาที่สินค้าคุณภาพต่ำสร้างเอาไว้ ทั้งนี้เพราะเขาต้องเผชิญกับซอฟต์แวร์แย่ ๆ นั้นทุกวัน”