บทที่ 10

วิศวกรรมซอฟท์แวร์

วัตถุประสงค์

หลังจากที่เรียนจบบทที่ 10 แล้ว นักศึกษาต้องสามารถ:

- Understand the software life cycle.
- Describe the development process models.
- Understand the concept of modularity in software engineering.
- Understand the importance of quality in software engineering.
- Understand the role of documentation in software engineering.

บทนำ

- ในบทนี้จะแนะนำแนวคิดของวิศวกรรมซอฟท์แวร์โดยเริ่มต้นจากวงจรชีวิตของซอฟท์แวร์ (software life cycle) จากนั้นจะอธิบายสองแบบจำลองหลักที่ใช้เป็นกระบวนการสำหรับพัฒนาซอฟท์แวร์คือ แบบจำลองน้ำตก (waterfall models) กับ แบบจำลองที่เพิ่มรายละเอียด (incremental models) สุดท้ายจะอธิบายคุณลักษณะที่สำคัญที่เกี่ยวข้องกับวิศวกรรมซอฟท์แวร์เช่น modularity, quality, และการจัดทำเอกสาร (documentation)
- วิศวกรรมซอฟท์แวร์คือการสร้างและการใช้วิธีการและหลักการอันเป็นที่ยอมรับทางวิศวกรรมเพื่อให้ได้มาซึ่งซอฟท์แวร์ที่มีความน่าเชื่อถือเมื่อ

บทนำ

ทำงานกับเครื่องคอมพิวเตอร์ คำนิยามนี้กำหนดขึ้น 30 ปีหลังจากที่เครื่องคอมพิวเตอร์เครื่องแรกได้ถูกสร้างขึ้นบนโลกนี้ ในช่วงเวลานั้นซอฟท์แวร์ถูกมองว่าเป็นศิลป์มากกว่าเป็นศาสตร์ ดังชุดหนังสือ 3 เล่มที่เขียนโดยศาสตราจารย์ Donald E. Knuth ชื่อ “The Art of Computer Programming” เมื่อปี ค.ศ. 1960 และต้นปี 1970 ในหนังสือชุดนี้ย้ำเน้นว่าในสมัยนั้นการเขียนโปรแกรมถือว่าเป็นศิลป์อย่างหนึ่ง อย่างไรก็ตามหนังสือชุดนี้ถือเป็นหนังสือที่อธิบายความรู้และแนวคิดของวิทยาการคอมพิวเตอร์ไว้ได้อย่างค่อนสมบูรณ์ครบถ้วนมากที่สุดชุดหนึ่ง ผู้ใฝ่เรียนควรได้หาเวลาและโอกาสอ่านเสีย

10.1

วงจรชีวิตของซอฟท์แวร์

วงจรชีวิตของซอฟท์แวร์

- แนวคิดพื้นฐานของวิศวกรรมซอฟท์แวร์คือ**วงจรชีวิตซอฟท์แวร์ (software life cycle)** ซอฟท์แวร์ก็เหมือนกับผลิตผลทั่วๆไปที่จะต้องผ่านวงจรของขั้นตอนที่ต้องทำซ้ำๆ (ดังรูปที่ 10.1)
- ซอฟท์แวร์ถูกพัฒนาขึ้นครั้งแรกโดยกลุ่มของนักพัฒนา/โปรแกรมเมอร์ โดยปกติซอฟท์แวร์ที่สร้างขึ้นจะถูกใช้งานไประยะหนึ่งก่อนที่จะมีการปรับปรุงหรือแก้ไข การปรับปรุงแก้ไขมักจะเกิดขึ้นเสมออาจเนื่องจากความผิดพลาดที่มีอยู่ในตัวโปรแกรม กฎระเบียบหรือเงื่อนไขมีการเปลี่ยนแปลง หรือแม้แต่นโยบายขององค์กรเปลี่ยนไป

วงจรชีวิตของซอฟท์แวร์ (ต่อ)

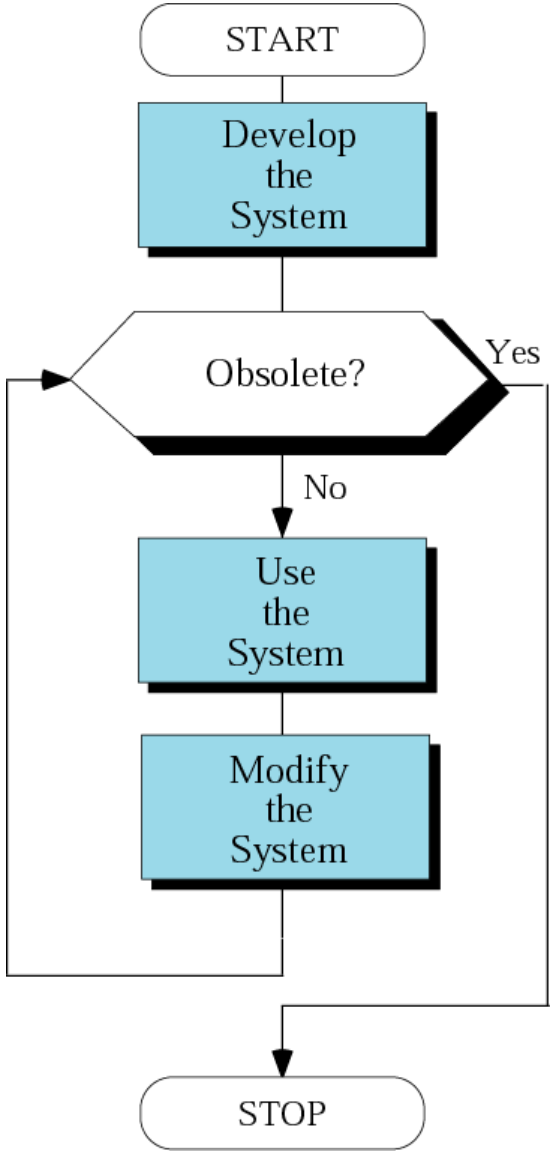
- การใช้งานและการปรับปรุงแก้ไขซอฟท์แวร์จะดำเนินต่อไปจนกระทั่งซอฟท์แวร์ล้าสมัย (obsolete) หมายความว่าโปรแกรมที่ใช้หมดประสิทธิภาพ ภาษาที่ใช้เขียนไม่มีการใช้อีกต่อไป องค์กรมีการเปลี่ยนแปลงอย่างใหญ่หลวง เป็นต้น
- กระบวนการพัฒนาในวงจรชีวิตของซอฟท์แวร์ประกอบด้วย 4 ระยะ:

  1. การวิเคราะห์ (analysis)
  2. การออกแบบ (Design)
  3. การดำเนินการ (Implementation)
  4. การทดสอบ (Testing)

  ดังรูปที่ 10.2

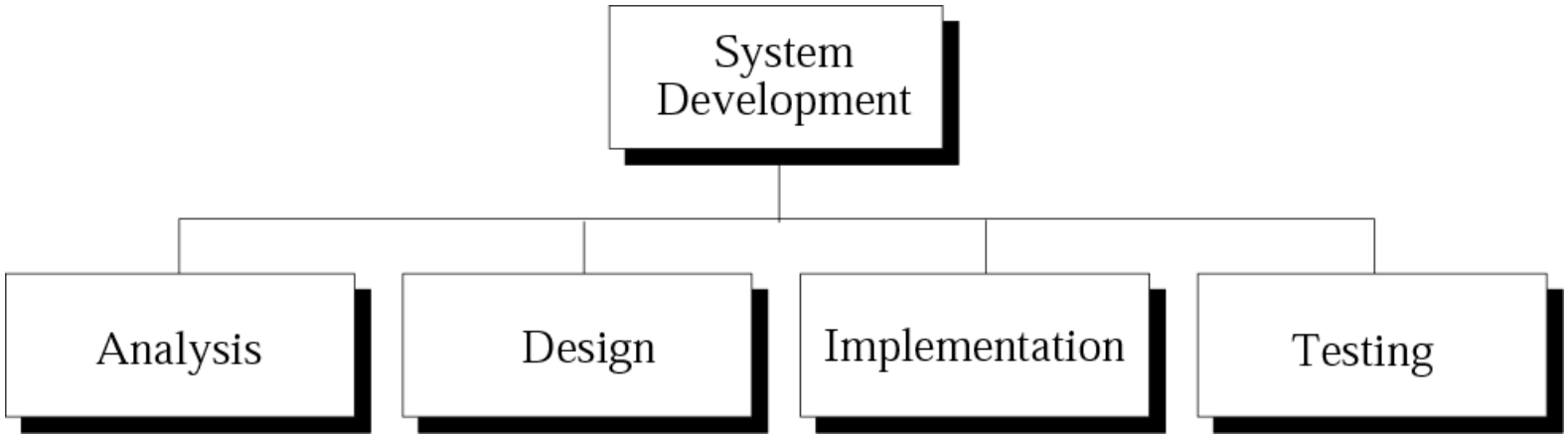
รูปที่ 10-1

วงจรชีวิตของระบบ



รูปที่ 10-2

ขั้นตอนการพัฒนาระบบ



วงจรชีวิตของซอฟท์แวร์

- **ระยะที่ 1**: **การวิเคราะห์** เป็นการกำหนดความต้องการของระบบ **(What the software should do?)** มี 4 ขั้นตอนคือ

    1. กำหนดผู้ใช้ (Define the User)
    2. กำหนดความต้องการของผู้ใช้ (Define the Needs)
    3. กำหนดความต้องการของระบบ (Define the Requirements)
    4. กำหนดวิธีการ (Define the Methods)

- **ระยะที่ 2**: **การออกแบบ** เป็นการระบุรายละเอียดว่าความต้องการจากระยะที่ 1 จะสำเร็จได้อย่างไร ต้องทำอย่างไร **(How the software should be built?)**

วงจรชีวิตของซอฟท์แวร์

- **ระยะที่ 3**: **การดำเนินการ** เป็นการลงมือเขียนโปรแกรม (coding) โดยใช้ภาษาที่กำหนดไว้ บางครั้งอาจมีการใช้เครื่องมือช่วย (tools) เช่น code generator, debugger, etc
- ระยะที่ 4: การทดสอบ เป็นกิจกรรมที่ทำการตรวจสอบความถูกต้องของโปแรกมก่อนการนำไปใช้งาน การทดสอบโปรแกรมมี 2 วิธีหลักๆคือ

    1. Black Box Testing
    2. White Box Testing

10.2

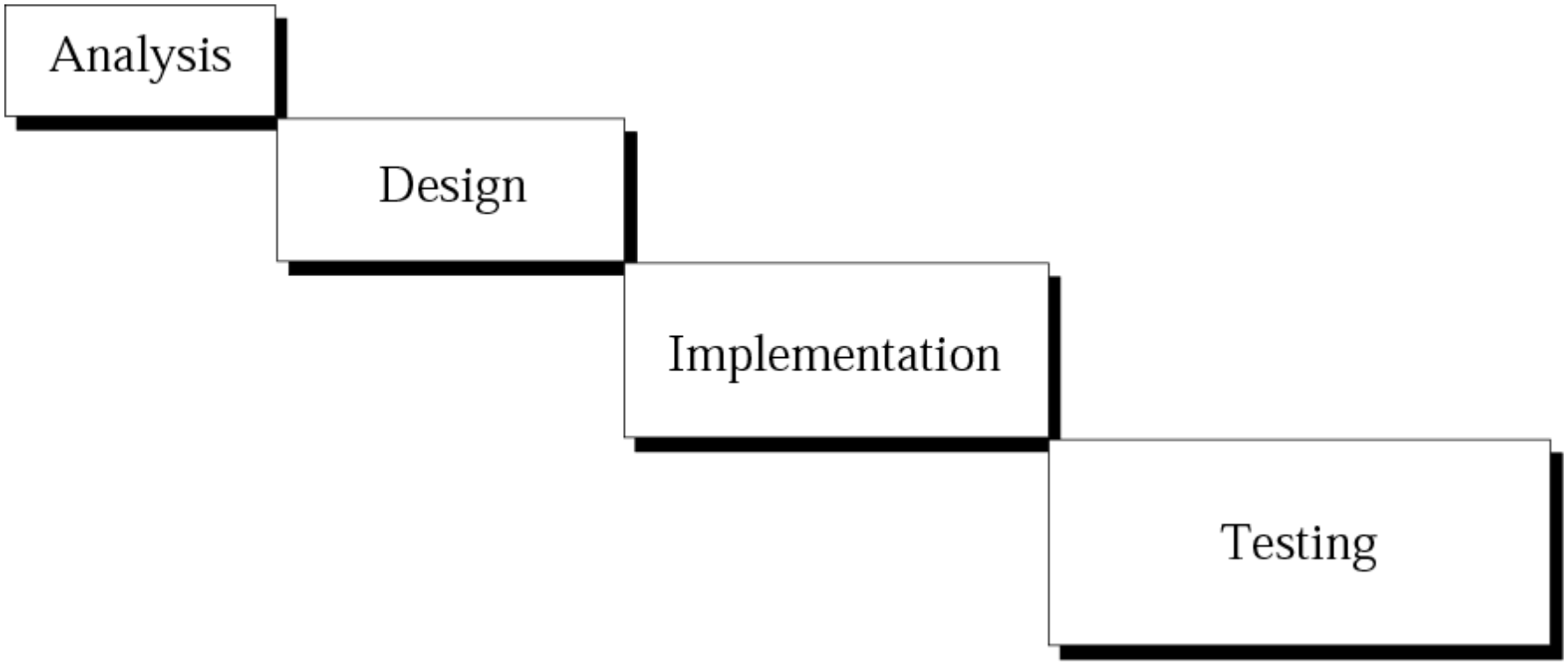
แบบจำลองกระบวนการพัฒนาระบบ

แบบจำลองการพัฒนาโปรแกรม

- **แบบจำลองน้ำตก** (Waterfall Model) เป็นการพัฒนาที่ระยะก่อนหน้าจะต้องเสร็จครบถ้วนสมบูรณ์ก่อนที่ระยะต่อไปจะดำเนินการได้ (ดังรูปที่ 10.3)
- **แบบจำลองที่เพิ่มรายระเอียด** (Incremental Model) เป็นการพัฒนาแบบวนซ้ำหลายๆรอบ ในแต่ละรอบจะเพิ่มเติมรายละเอียดเข้าไปเพื่อให้โปรแกรมสมบูรณ์มากขึ้นๆ ดังรูปที่ (10.4)

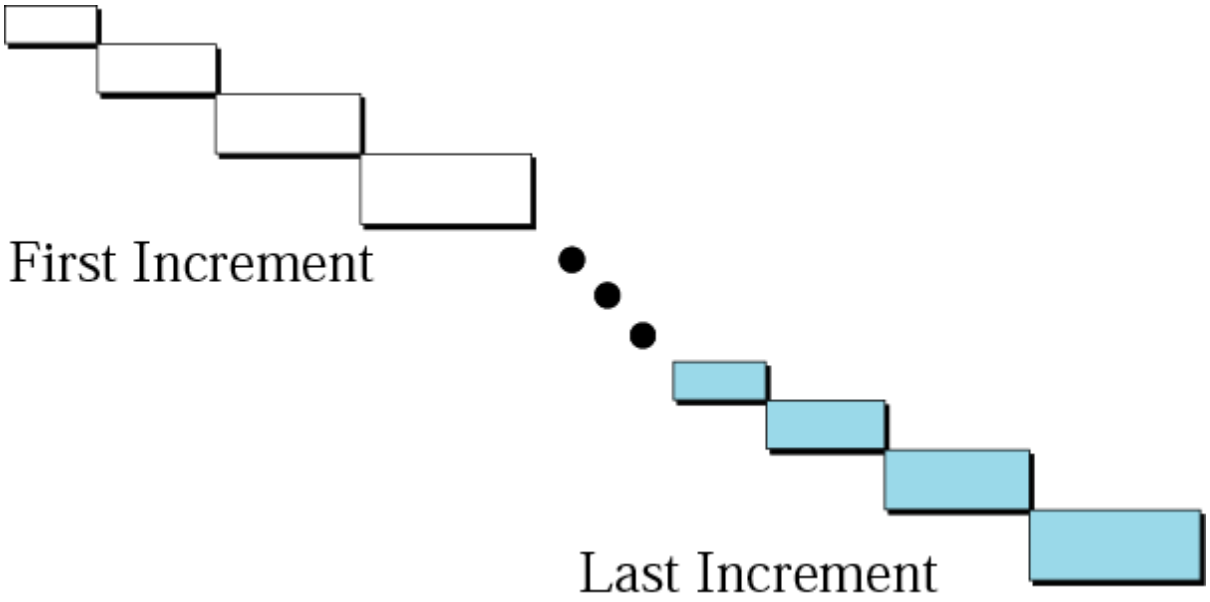
รูปที่ 10-3

Waterfall model



รูปที่ 10-4

Incremental model



10.3

MODULARITY

Modularity

- **Modularity** หมายถึงการแบ่งระบบงานที่มีขนาดใหญ่และซับซ้อนออกเป็นส่วนย่อยๆ ซึ่งจะทำให้สามารถเข้าใจแต่ละส่วนได้ง่ายขึ้น
- **Tools**: เครื่องมือที่สำคัญที่ใช้เช่น Structure chart, Class diagram, UML
- คุณสมบัติที่พึงประสงค์ของ Modularity

  1. Coupling: เป็นตัววัดหรือบ่งบอกระดับของความสัมพันธ์ระหว่าง 2 โมดุลใดๆ เราต้องการให้ค่า coupling ต่ำๆ coupling มี 5 ระดับคือ

  * Data coupling
  * Control coupling
  * Content coupling
  * Stamp coupling
  * Global coupling

Modularity

- **2. Cohesion**: เป็นตัววัดหรือบ่งบอกระดับการเกาะตัวกันหรือยึดเหนี่ยวกันระหว่างคำสั่งภายในโมดุลเดียวกัน ความต้องการคือในแต่ละโมดุลเราอยากให้ค่า cohesion สูงๆ cohesion มี 7 ระดับคือ

    * Functional cohesion
    * Sequential cohesion
    * Communicational cohesion
    * Procedural cohesion
    * Temporal cohesion
    * Logical cohesion
    * Coincidental cohesion

10.4

QUALITY

คุณภาพของซอฟท์แวร์

- **การกำหนดคุณภาพซอฟท์แวร์: คุณภาพซอฟท์แวร์กำหนดไว้ดังนี้**

  “ Software that satisfies the user’s explicit and implicit requirements, is well documented, meet the oprating standards of the organization, and runs efficiently on the hardware for which it was developed.”

องค์ประกอบของคุณภาพของซอฟท์แวร์

- **Quality Factors: องค์ประกอบของคุณภาพของซอฟท์แวร์สามารถแบ่งออกเป็น 3 ส่วนคือ**

  *** Operability --> ความสามารถที่จะปฏิบัติการได้**

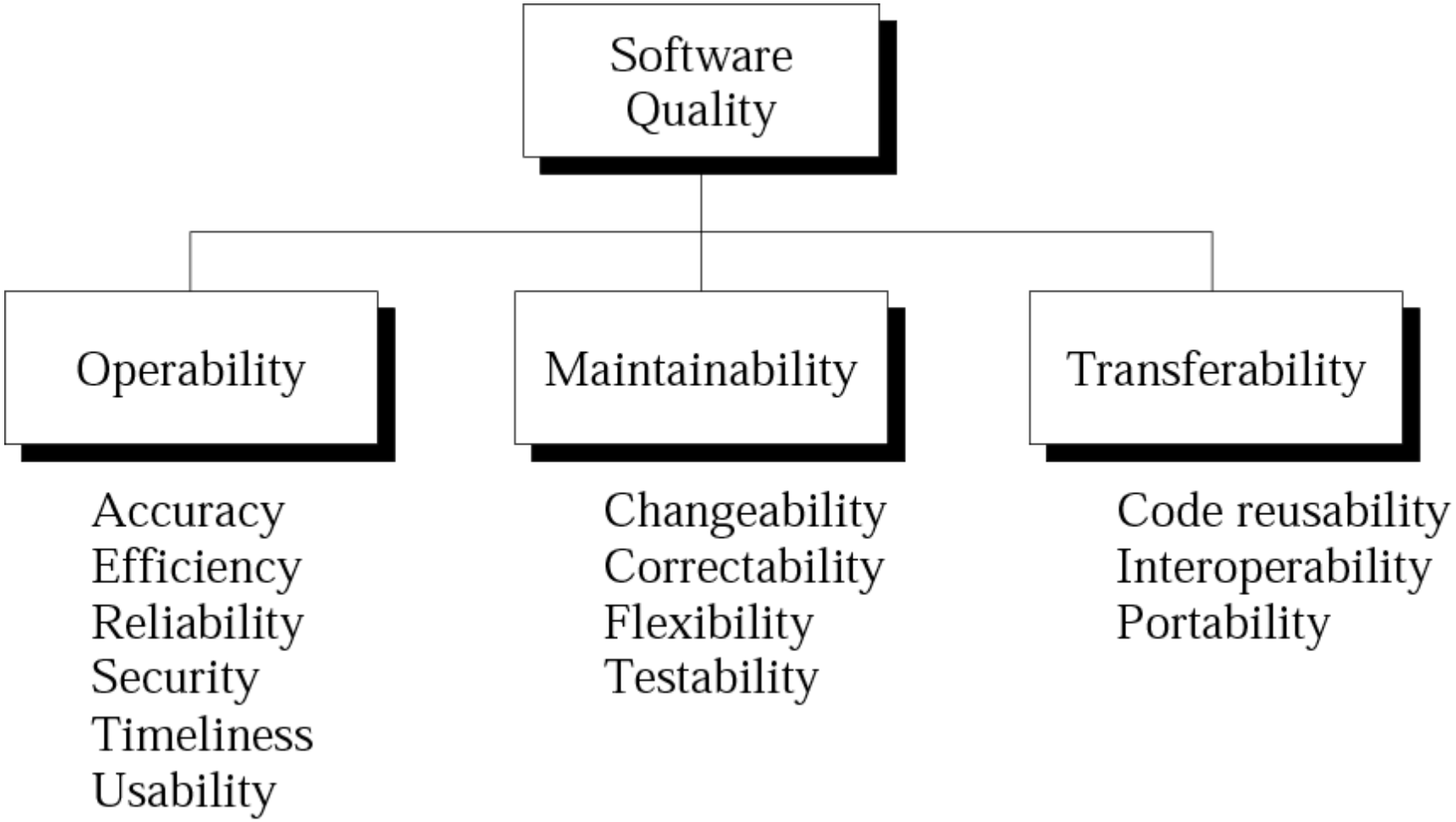
  *** Maintainability --> ความสามารถที่จะแก้ไขเปลี่ยนแปลงได้ง่าย**

  *** Transferability --> ความสามารถที่เคลื่อนย้ายจากเครื่องหนึ่งให้ไปทำงานในอีกเครื่องหนึ่งได้**

รายละเอียดดังรูปที่ 10.5

รูปที่ 10-5

องค์ประกอบของคุณภาพ



วงจรคุณภาพ (Quality Circle)

- **Quality Circle: เป็นการผสมผสานคุณภาพของซอฟท์แวร์เข้าไปในตัวซอฟท์แวร์ตั้งแต่ระยะที่ 1-4 หรือทุกระยะของการพัฒนา และต้องกระทำอย่างต่อเนื่อง คุณภาพของซอฟท์แวร์ไม่สามารถเพิ่มเข้าไปก่อนหรือหลังการพัฒนา**
- การทำให้ซอฟท์แวร์มีคุณภาพตามที่ประสงค์มี 6 ขั้นตอนคือ

1. Quality tools

2. Technical reviews

3. Formal testing

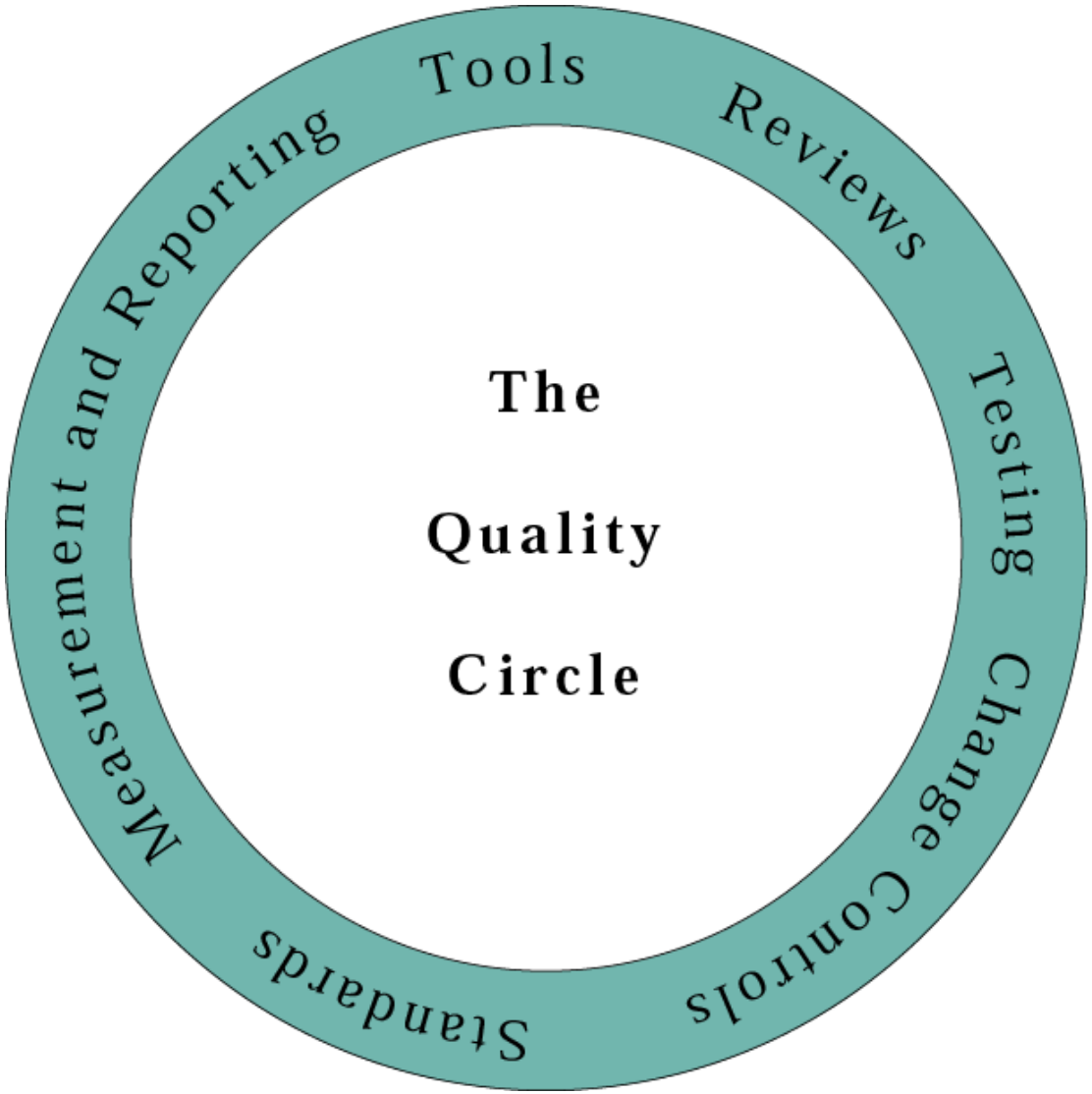
4. Change controls

5. Standards

6. Measurement and reporting

รูปที่ 10-6

วงจรคุณภาพ



10.5

การจัดทำเอกสาร

การจัดทำเอกสาร (Documentation)

- ซอฟท์แวร์ที่ดีต้องมีเอกสารประกอบ เอกสารอาจอยู่ในรูปสิ่งพิมพ์หรืออิเล็กทรอนิกส์ วัตถุประสงค์ของเอกสารคือให้ผู้ใช้สามารถใช้งานได้ง่ายและช่วยให้การเปลี่ยนแปลงแก้ไขสะดวกและรวดเร็วขึ้น โดยปกติเอกสารแบ่งออกเป็น 2 ประเภทใหญ่ๆคือ

  1. User documentation สำหรับผู้ใช้งาน

  2. System documentation เป็นเอกสารที่ทำขึ้นในแต่ละระยะของการพัฒนาซอฟท์แวร์โดยมีรายละเอียดดังนี้

     * เอกสารในระยะการวิเคราะห์ -- บันทึกความต้องการ

การจัดทำเอกสาร (ต่อ)

* เอกสารในระยะการออกแบบ — บันทึกพิมพ์เขียวของซอฟท์วร์
* เอกสารในระยะการดำเนินการ — บันทึกรายละเอียดของโปรแกรม ข้อยกเว้นต่างๆ ตลอดจนสิ่งแวดล้อมในการทำงานของโปรแกรม
* เอกสารในระยะการทดสอบ — บันทึกผลการทดสอบอย่างละเอียดเพื่อเก็บไว้เป็นหลักฐาน
* เอกสารบันทึกกระบวนการทำงานของโปรแกรม — บันทึกประวัติการทำงานของโปรแกรม

คำสำคัญ

- Software Life Cycle
- Software Quality
- Modularity