Software Quality Management

Week 12

ดัดแปลงจาก slides ของหนังสือ Software Engineering [1]

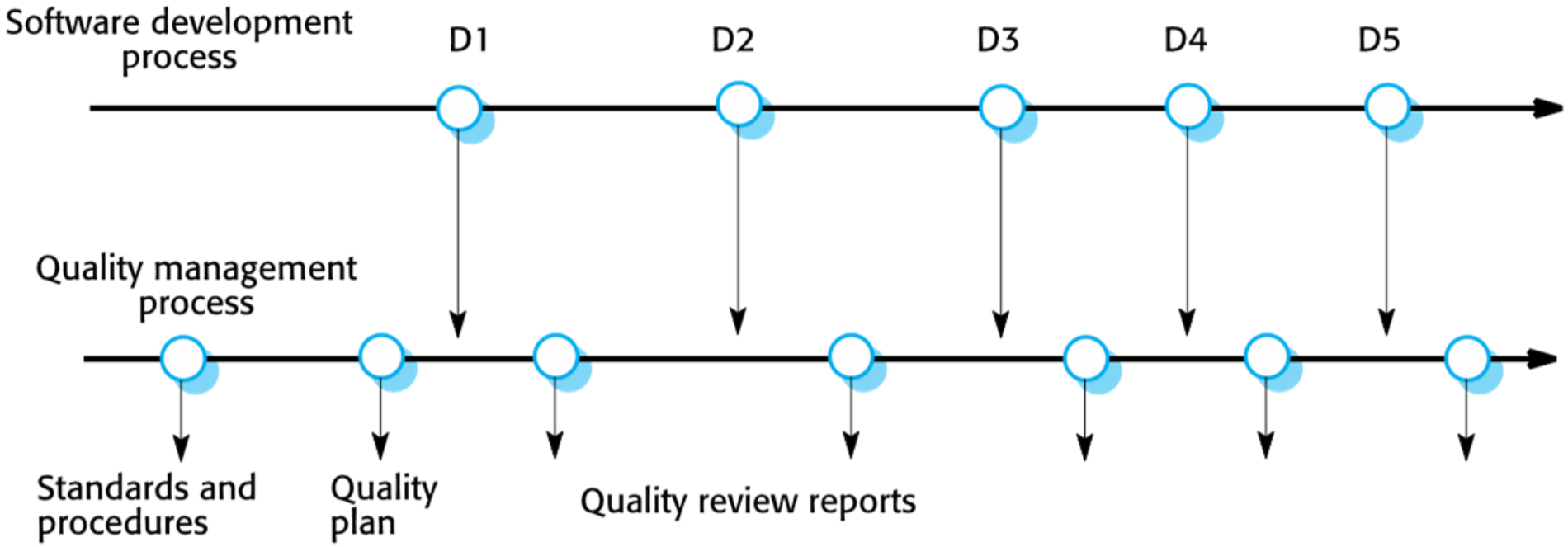
หัวข้อที่จะศึกษา

- Software quality
- Software standards
- Reviews and inspections
- Quality management and agile development
- Software measurement

Software quality management

- มุ่งเน้นที่การสร้างความมั่นใจว่าผลิตภัณฑ์ซอฟต์แวร์มีระดับที่ต้องการ
- ในระดับองค์กร
  - การจัดการคุณภาพจะเกี่ยวข้องกับการคัดเลือกและใช้งาน framework ของกระบวนการ (process) และมาตรฐาน (standard) ต่าง ๆ เพื่อที่จะนำไปสู่การสร้างซอฟต์แวร์ที่มีคุณภาพสูง
- ในระดับโครงการ
  - การจัดการด้านคุณภาพจะเกี่ยวข้องกับการใช้กระบวนการคุณภาพที่เฉพาะเจาะจงและการตรวจสอบว่ากระบวนการวางแผนดังกล่าวได้มีการนำไปปฏิบัติแล้วหรือไม่
  - การจัดการคุณภาพเกี่ยวข้องกับการจัดทำแผนคุณภาพสำหรับโครงการ แผนคุณภาพควรกำหนดเป้าหมายคุณภาพสำหรับโครงการ รวมทั้งกำหนดกระบวนการและมาตรฐานที่จะใช้

Quality management and software development



Quality plans

- โครงสร้างการวางแผนคุณภาพ
  - Product introduction
  - Product plans
  - Process descriptions
  - Quality goals
  - Risks and risk management
- แผนการตรวจสอบคุณภาพควรสั้นและกระชับ
  - ถ้ายาวเกินไป จะไม่มีใครอ่าน

Scope of quality management

- การจัดการด้านคุณภาพมีความสำคัญเป็นพิเศษสำหรับระบบที่มีขนาดใหญ่และซับซ้อน
- เอกสารควบคุมคุณภาพ ใช้บันทึกความคืบหน้าและช่วยรักษาความต่อเนื่องของการพัฒนาในกรณีที่มีการเปลี่ยนแปลงทีมพัฒนา
- สำหรับระบบขนาดเล็ก การจัดการด้านคุณภาพจะมีเอกสารน้อยลงและควรเน้นการสร้างวัฒนธรรมที่เน้นคุณภาพ
- ในการพัฒนาแบบ agile จะต้องมีการจัดทำเอกสารควบคุมคุณภาพ

Software quality

Software quality

- คำว่า “คุณภาพ” โดยปกติหมายความว่าผลิตภัณฑ์ควรเป็นไปตามข้อกำหนด
- ปัญหาสำหรับคุณภาพของระบบซอฟต์แวร์
  - มีความตึงเครียดระหว่างข้อกำหนดด้านคุณภาพของลูกค้า (ประสิทธิภาพ ความน่าเชื่อถือ ฯลฯ) และความต้องการด้านคุณภาพของนักพัฒนา (ความสามารถในการบำรุงรักษา การนำมาใช้ใหม่ เป็นต้น)
- ข้อกำหนดด้านคุณภาพบางเรื่องนั้นอาจจะระบุอย่างชัดเจนได้ยาก
- ข้อกำหนดของซอฟต์แวร์มักไม่สมบูรณ์และมักจะไม่สอดคล้องกัน
- ถ้าจำเป็นต้องเลือก ให้เลือกวิธีการที่พัฒนาซอฟต์แวร์ได้อย่าง “ตรงตามความต้องการ" มากกว่า “สอดคล้องกับข้อกำหนด”

Software fitness for purpose

- ซอฟต์แวร์ได้รับการทดสอบอย่างถูกต้องหรือไม่?
- ซอฟต์แวร์มีความน่าเชื่อถือเพียงพอที่จะนำไปใช้หรือไม่?
- ประสิทธิภาพของซอฟต์แวร์เป็นที่ยอมรับสำหรับการใช้งานตามปกติหรือไม่?
- ซอฟต์แวร์สามารถใช้งานได้หรือไม่?
- ซอฟต์แวร์มีโครงสร้างที่ดีและทำความเข้าใจได้ง่ายหรือไม่?
- ในกระบวนการพัฒนามีมาตรฐานการเขียนโปรแกรมและจัดการจัดทำเอกสารหรือไม่?

Non-functional characteristics

- คุณภาพของระบบซอฟต์แวร์ส่วนใหญ่ มักจะอยู่ในความสามารถที่เป็น non-functional
- ลักษณะดังกล่าวสะท้อนถึงประสบการณ์ของผู้ใช้
  - ถ้าการทำงานของซอฟต์แวร์ไม่เป็นไปตามผู้ใช้ที่คาดหวัง เขามักจะหาวิธีการอะไรสักอย่างเพื่อให้สามารถในการทำสิ่งที่ต้องการจนได้ (เช่น การใช้คีย์ลัด เป็นต้น)

Software quality attributes

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Quality conflicts

- ไม่สามารถปรับแต่งระบบใด ๆ ให้เหมาะกับคุณลักษณะเหล่านี้ทั้งหมดเช่นการปรับปรุงความแข็งแกร่งอาจทำให้ประสิทธิภาพการทำงานลดลง
- แผนคุณภาพจึงควรกำหนดคุณลักษณะคุณภาพที่สำคัญที่สุดสำหรับซอฟต์แวร์ที่กำลังพัฒนาขึ้น
- แผนควรรวมถึงคำนิยามของกระบวนการประเมินคุณภาพซึ่งเป็นวิธีที่ตกลงกันในการประเมินว่ามีคุณภาพบางอย่างเช่นความสามารถในการบำรุงรักษาหรือความทนทานอยู่ในผลิตภัณฑ์

Process and product quality

- คุณภาพของผลิตภัณฑ์มักจะได้รับผลกระทบจากคุณภาพของกระบวนการผลิต
- นี่เป็นสิ่งสำคัญในการพัฒนาซอฟต์แวร์ เนื่องจากคุณลักษณะบางอย่างของคุณภาพของผลิตภัณฑ์นั้นยากที่จะประเมินได้
- มีความสัมพันธ์ที่ซับซ้อนและเข้าใจได้ยาก ระหว่าง software process และ product quality
  - การใช้ทักษะและประสบการณ์ของแต่ละบุคคลมีความสำคัญเป็นพิเศษในการพัฒนาซอฟต์แวร์
  - ปัจจัยภายนอก เช่น ความแปลกใหม่ของ application หรือความจำเป็นในการกระชับตารางเวลาในการพัฒนา อาจส่งผลต่อคุณภาพของผลิตภัณฑ์

Quality culture

- ผู้จัดการคุณภาพ (Quality managers) ควรมุ่งมั่นที่จะพัฒนา 'วัฒนธรรมคุณภาพ' (quality culture) ซึ่งทุกคนรับผิดชอบด้านการพัฒนาซอฟต์แวร์ มุ่งมั่นที่จะบรรลุถึงคุณภาพของผลิตภัณฑ์ในระดับสูง
- ควรส่งเสริมให้ทุกคนในทีม มีความรับผิดชอบต่อคุณภาพของงานและพัฒนาแนวทางใหม่ในการปรับปรุงคุณภาพอย่างสม่ำเสมอ

Software standards

Software standards

- มาตรฐาน (Standards) เป็นตัวกำหนดคุณลักษณะที่จำเป็นสำหรับผลิตภัณฑ์หรือกระบวนการ เป็นสิ่งที่มีบทบาทสำคัญในการจัดการคุณภาพ
- มาตรฐานอาจเป็นได้หลายระดับ ได้แก่
- มาตรฐานระหว่างประเทศ (international standards)
- ระดับชาติ (national standards)
- ระดับองค์กร (organizational standards)
- ระดับโครงการ (project standards)

Product and process standards

- มาตรฐานผลิตภัณฑ์
  - ใช้กับผลิตภัณฑ์ซอฟต์แวร์ที่กำลังพัฒนาและเอกสารมาตรฐานต่าง ๆ เช่น
    - โครงสร้างเอกสาร
    - ข้อกำหนดมาตรฐานเอกสาร
    - ส่วนหัวมาตรฐานสำหรับการระบุความหมายของ class ต่าง ๆ
    - มาตรฐานการเขียนโปรแกรม ซึ่งกำหนดว่าควรใช้ภาษาเขียนโปรแกรมอย่างไร
- มาตรฐานกระบวนการผลิต
  - กำหนดกระบวนการที่ควรปฏิบัติตามในระหว่างการพัฒนาซอฟต์แวร์ เช่น
    - คำจำกัดความของ requirement
    - กระบวนการออกแบบและการตรวจสอบความถูกต้อง
    - เครื่องมือสนับสนุนกระบวนการ
    - คำอธิบายของเอกสารที่ต้องจัดทำในระหว่างกระบวนการผลิต

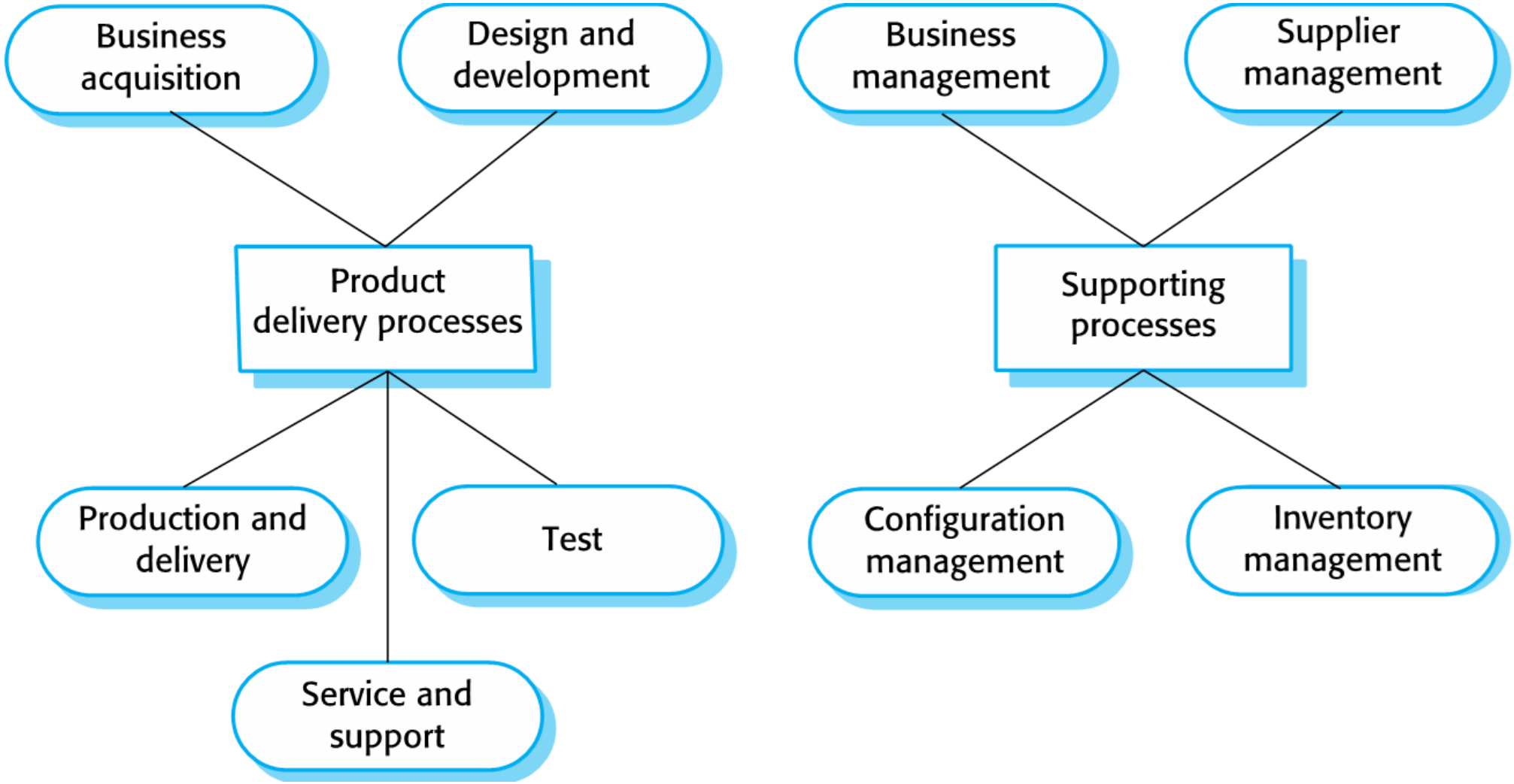
Problems with standards

- วิศวกรซอฟต์แวร์อาจไม่เห็นความทันสมัยของมาตรฐานและความเกี่ยวข้องกับซอฟต์แวร์ที่พัฒนา
- มาตรฐานมักจะเกี่ยวข้องกับการกรอกฟอร์มที่เป็นทางการมากเกินไป
- การเก็บรักษาเอกสารการพัฒนาซอฟต์แวร์เพื่อให้เป็นไปตามมาตรฐาน มักจะเป็นงานที่ยุ่งยาก หากไม่มีเครื่องมือที่ดีมาช่วยจัดการ มักจะทำให้เกิดการต่อต้านโดยคนทำงาน

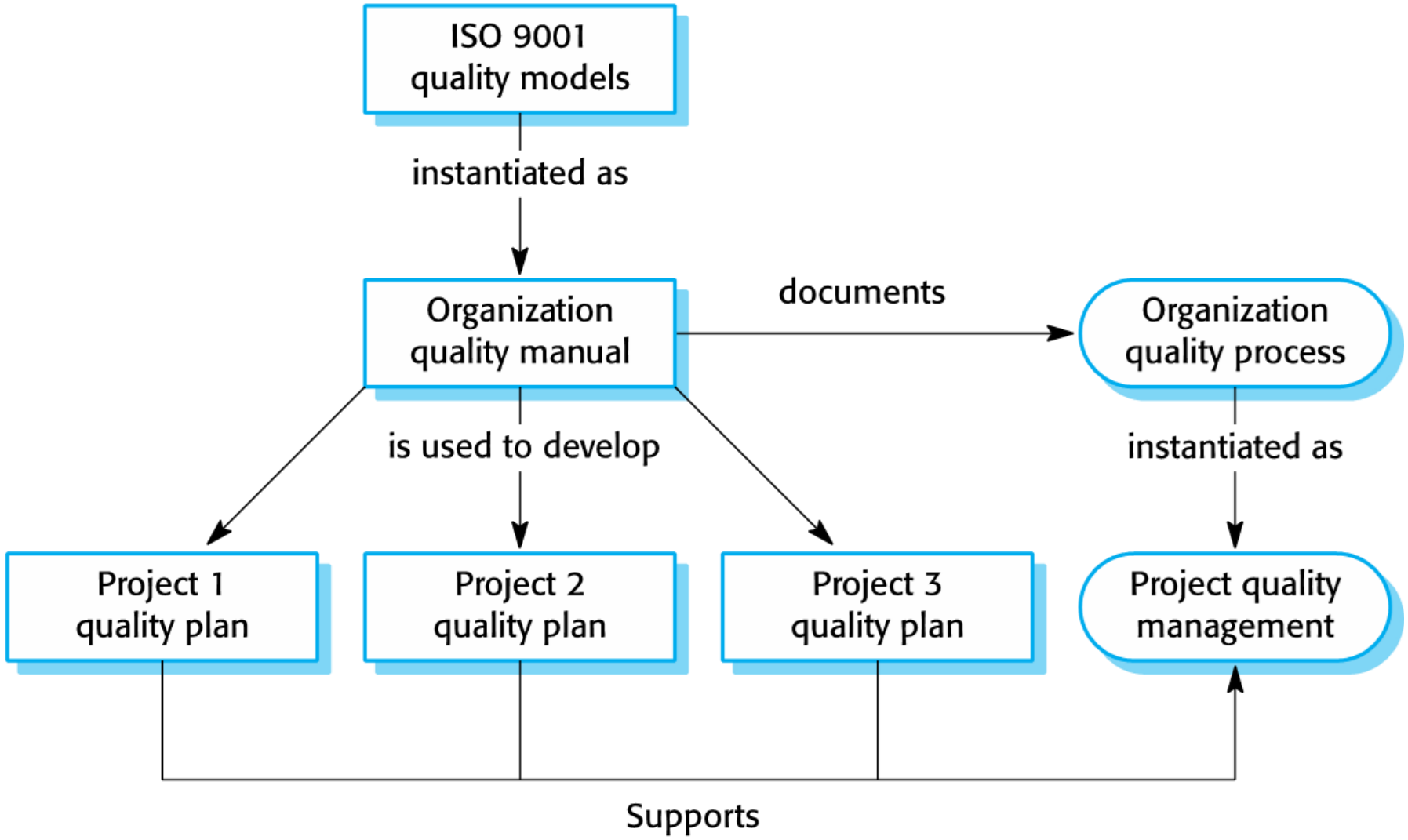
ISO 9001 standards framework

- มาตรฐานสากล (international set of standards) สามารถใช้เป็นพื้นฐานในการพัฒนาระบบการจัดการคุณภาพ
- มาตรฐาน ISO 9001 เป็นมาตรฐานทั่วไปขององค์กรที่ออกแบบ พัฒนา และดูแลรักษาผลิตภัณฑ์ (รวมถึงซอฟต์แวร์)
- มาตรฐาน ISO 9001 สามารถใช้เป็น framework มาตรฐานในการพัฒนาซอฟต์แวร์
  - กำหนดหลักการคุณภาพโดยทั่วไป
  - อธิบายถึงกระบวนการด้านคุณภาพโดยทั่วไป
  - กำหนดมาตรฐานและวิธีการขององค์กร

ISO 9001 core processes



ISO 9001 and quality management



ISO 9001 certification

- มาตรฐานคุณภาพ (Quality standards) และขั้นตอน (procedures) ควรได้รับการจัดทำเป็นคู่มือคุณภาพขององค์กร
- คู่มือด้านคุณภาพขององค์กร ที่เป็นไปตามมาตรฐาน ISO 9000 อาจต้องได้รับการตรวจสอบและรับรองโดยหน่วยงานภายนอก
- ลูกค้าบางรายต้องการ supplier ที่ได้รับการรับรองมาตรฐาน ISO 9000 แม้ว่าจะมีต้นทุนที่สูงขึ้น

Software quality and ISO9001

- การรับรองมาตรฐาน ISO 9001 อย่างเดียวอาจไม่เพียงพอ เนื่องจากการกำหนดคุณภาพเพื่อให้สอดคล้องกับมาตรฐาน ไม่ได้ระบุในมาตรฐาน ISO 9001
- มาตรฐาน ISO 9001 ไม่ได้คำนึงถึงคุณภาพในด้านประสบการณ์จากผู้ใช้ซอฟต์แวร์
  - บริษัทสามารถกำหนดมาตรฐานขึ้นเอง เพื่อให้ครอบคลุมการทดสอบที่ระบุว่า ต้องทดสอบ method ทั้งหมดใน object อย่างน้อยหนึ่งครั้ง

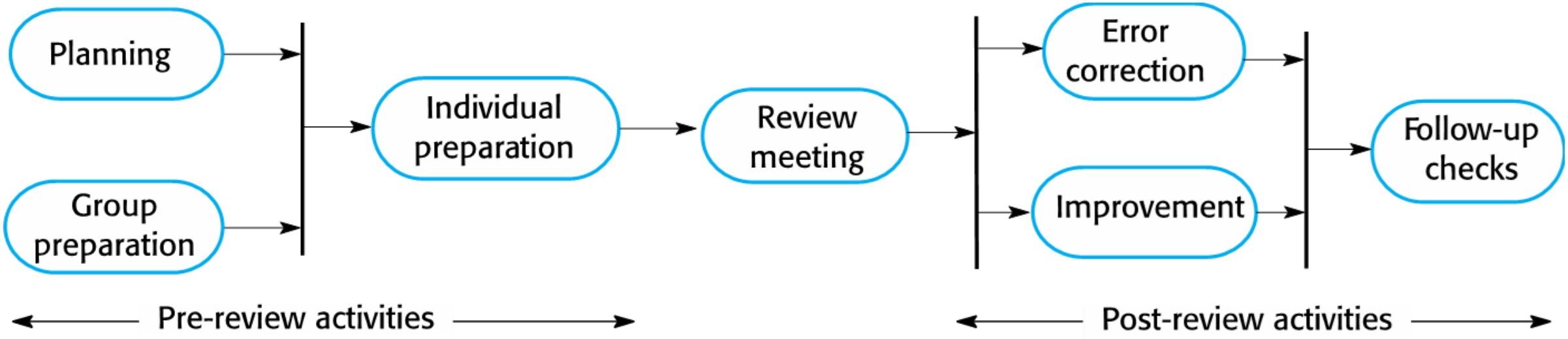
Quality reviews

- ทีมงานดำเนินการตรวจสอบบางส่วนหรือทั้งทั้งหมดของระบบอย่างละเอียด ทั้งในส่วนของซอฟต์แวร์และเอกสารประกอบที่เกี่ยวข้อง
  - ในการ review จะต้องดูให้ละเอียดไม่ว่าจะเป็น Code, designs, specifications, test plans, standards, ฯลฯ
- ซอฟท์แวร์หรือเอกสารอาจถูกรับรองลงนาม (signed-off) ในการตรวจทาน ซึ่งหมายถึงการพัฒนาในขั้นตอนนั้นได้รับการอนุมัติจากฝ่ายบริหาร

Phases in the review process

- กิจกรรมก่อนการตรวจทาน (Pre-review activities)
  - กิจกรรมก่อนการตรวจทานเกี่ยวข้องกับการวางแผนตรวจทานและการเตรียมการตรวจทาน
- การประชุมทบทวน (The review meeting)
  - ในระหว่างการประชุมทบทวน ผู้เขียนเอกสารหรือโปรแกรมที่กำลังได้รับการตรวจทานควรพิจารณาเอกสารทีละขั้นร่วมกับทีมตรวจสอบ
- กิจกรรมหลังการทบทวน (Post-review activities)
  - แก้ไขปัญหาและประเด็นที่ได้รับการแจ้งในระหว่างการประชุมทบทวน

The software review process



Distributed reviews

- กระบวนการที่แนะนำสำหรับการรีวิวถือว่า ทีมรีวิวมีการประชุมแบบตัวต่อตัวเพื่อหารือเกี่ยวกับซอฟต์แวร์หรือเอกสารที่พวกเขากำลังรีวิว
- อย่างไรก็ตามทีมงานพัฒนาซอฟต์แวร์มักกระจายอยู่ในหลายประเทศหรือหลายทวีป ดังนั้นจึงไม่สามารถใช้วิธีการนี้ได้กับสมาชิกในทีมพัฒนา
- สามารถนำวิธีการรีวิวจากระยะไกลมาใช้ โดยใช้เอกสารที่ใช้ร่วมกัน (shared documents) ซึ่งสมาชิกในทีมรีวิวแต่ละคนสามารถรีวิวพร้อมให้ความคิดเห็นได้

Program inspections

- เป็นการรีวิวที่วิศวกรสามารถตรวจสอบแหล่งที่มาของระบบ โดยมีวัตถุประสงค์เพื่อค้นหาความผิดปกติและข้อบกพร่อง
- การตรวจสอบไม่จำเป็นต้องมีการดำเนินการของระบบ (execution) เพื่อให้สามารถใช้งานได้ก่อนการใช้งาน
- สามารถนำไปประยุกต์ใช้กับการแสดงระบบในหลายแง่มุม (เช่น ความต้องการ การออกแบบข้อมูล การกำหนดค่าข้อมูล การทดสอบ ฯลฯ)
- เป็นเทคนิคที่มีประสิทธิภาพสำหรับการค้นพบข้อผิดพลาดของโปรแกรม

Inspection checklists

- ควรใช้รายการตรวจสอบข้อผิดพลาด (Error checklist) เพื่อควบคุมการตรวจสอบ
- Error checklists มีลักษณะเป็นภาษาโปรแกรมและสะท้อนถึงข้อผิดพลาดลักษณะที่อาจเกิดขึ้นในภาษา
- ตัวอย่างรายการ Error checklists
  - การเริ่มต้น (Initialization)
  - การตั้งชื่อค่าคงที่ (Constant naming)
  - การสิ้นสุดการวนรอบ (loop termination)
  - ขอบเขตของอาร์เรย์ (array bounds)
  - ฯลฯ

An inspection checklist (a)

Fault class	Inspection check
Data faults	• Are all program variables initialized before their values are used? • Have all constants been named? • Should the upper bound of arrays be equal to the size of the array or Size -1? • If character strings are used, is a delimiter explicitly assigned? • Is there any possibility of buffer overflow?
Control faults	• For each conditional statement, is the condition correct? • Is each loop certain to terminate? • Are compound statements correctly bracketed? • In case statements, are all possible cases accounted for? • If a break is required after each case in case statements, has it been included?
Input/output faults	• Are all input variables used? • Are all output variables assigned a value before they are output? • Can unexpected inputs cause corruption?

An inspection checklist (b)

Fault class	Inspection check
Interface faults	• Do all function and method calls have the correct number of parameters? • Do formal and actual parameter types match? • Are the parameters in the right order? • If components access shared memory, do they have the same model of the shared memory structure?
Storage management faults	• If a linked structure is modified, have all links been correctly reassigned? • If dynamic storage is used, has space been allocated correctly? • Is space explicitly deallocated after it is no longer required?
Exception management faults	• Have all possible error conditions been taken into account?

Quality management and agile development

Quality management and agile development

- การจัดการด้านคุณภาพ (Quality management) ในการพัฒนาแบบ agile ถือว่าเป็นเรื่องไม่เป็นทางการ
- มันขึ้นอยู่กับการสร้างวัฒนธรรมที่มีคุณภาพของทีม ซึ่งสมาชิกในทีมทุกคนควรรู้สึกรับผิดชอบต่อคุณภาพของซอฟต์แวร์ และดำเนินการทุกวิธีเพื่อให้มั่นใจว่ามีการรักษาคุณภาพไว้ทุกขั้นตอน
- ชุมชน agile มักจะมีพื้นฐานแนวคิดที่สวนทางกับมาตรฐานและกระบวนการคุณภาพตามที่กำหนดไว้ใน ISO 9001

Shared good practice

- ตรวจสอบก่อนเช็คอิน (Check before check-in)
  - โปรแกรมเมอร์มีหน้าที่รับผิดชอบในการจัดทำบทวิจารณ์โค้ดของตนเองกับสมาชิกคนอื่นในทีม ก่อนที่โค้ดจะถูกนำเข้าสู่ระบบเพื่อสร้างซอฟต์แวร์
- อย่าทำลายระบบ (Never break the build)
  - สมาชิกในทีมไม่ควรนำเข้ารหัสที่ทำให้ระบบล้มเหลวเข้าสู่ระบบ
  - นักพัฒนาซอฟต์แวร์ต้องทดสอบการเปลี่ยนแปลงโค้ดกับทั้งระบบ และมั่นใจว่างานเหล่านี้จะได้ผลตามที่คาดไว้
- แก้ไขปัญหาทันทีที่พบ (Fix problems when you see them)
  - หากโปรแกรมเมอร์ค้นพบปัญหาหรือความคลุมเครือในโค้ดที่พัฒนาโดยบุคคลอื่น พวกเขาสามารถแก้ไขปัญหาเหล่านี้ได้โดยตรงแทนที่จะส่งไปให้ผู้พัฒนาเดิมดำเนินการ

Reviews and agile methods

- กระบวนการรีวิวในการพัฒนาซอฟต์แวร์แบบ agile มักไม่เป็นทางการ
- ในการพัฒนาแบบ scrum มีการประชุมทบทวนหลังจากแต่ละ iteration ของซอฟต์แวร์เสร็จสิ้นลง (เรียกว่า sprint review) ซึ่งอาจมีการกล่าวถึงประเด็นด้านคุณภาพและปัญหาต่างๆ
- ใน Extreme Programming การเขียนโปรแกรมแบบคู่ (pair programming) ช่วยให้มั่นใจได้ว่าโค้ดจะถูกตรวจสอบและรีวิวโดยสมาชิกคนอื่นในทีมอยู่เสมอ

Pair programming

- ในวิธีการนี้ สมาชิก 2 คนในทีม มีหน้าที่ในการพัฒนาโค้ดและทำงานร่วมกันเพื่อให้บรรลุเป้าหมาย
- รหัสที่พัฒนาขึ้นโดยสมาชิกจึงถูกตรวจสอบและตรวจทานโดยสมาชิกอีกคนในทีมอย่างต่อเนื่อง
- การเขียนโปรแกรมคู่จะนำไปสู่ความรู้ที่ลึกซึ้งของโปรแกรม เนื่องจากโปรแกรมเมอร์ทั้งสองต้องเข้าใจในรายละเอียดของโปรแกรมเพื่อพัฒนาต่อ
- การเขียนโปรแกรมคู่สามารถหาข้อบกพร่องที่ไม่สามารถค้นพบได้ในการรีวิวอย่างเป็นทางการ

Pair programming weaknesses

- ทั้งคู่อาจทำความเข้าใจข้อกำหนดของระบบผิดพลาดเหมือนกัน ยิ่งคุยกันมากขึ้นก็อาจจะยิ่งเพิ่มความผิดพลาดเหล่านั้น
- ทั้งคู่อาจจะลังเลที่จะมองหาข้อผิดพลาด เนื่องจากไม่ต้องการชะลอความคืบหน้าของโครงการ
- ความสามารถในการค้นพบข้อบกพร่องของคู่นี้อาจจะถูกทำให้แย่ลง เนื่องจากความสัมพันธ์ในการทำงานอย่างใกล้ชิดมักจะนำไปสู่การไม่เต็มใจที่จะวิพากษ์วิจารณ์เพื่อนร่วมงาน

Agile QM and large systems

- เมื่อมีการพัฒนาระบบขนาดใหญ่ แนวทาง agile ที่อาศัยจัดการด้านคุณภาพด้วยเอกสารอย่างน้อยนิด อาจไม่สามารถเป็นไปได้ในทางปฏิบัติ
  - ถ้าลูกค้าเป็นบริษัทขนาดใหญ่ อาจมีกระบวนการบริหารจัดการคุณภาพของตนเอง และคาดว่าบริษัทพัฒนาซอฟต์แวร์จะสามารถรายงานความคืบหน้าในทางที่เข้ากันได้
  - ในกรณีที่มีทีมงานที่กระจายตัวทางภูมิศาสตร์หลายแห่ง นักพัฒนาอาจมาจากบริษัทอื่นที่ใช้ภาษาถิ่นแตกต่างกัน ดังนั้นการสื่อสารแบบไม่เป็นทางการอาจใช้ไม่ได้
  - สำหรับระบบที่มีอายุการใช้งานยาวนาน อาจจะมีการเปลี่ยนแปลงสมาชิกในทีมงานพัฒนา สมาชิกในทีมใหม่อาจไม่สามารถเข้าใจเอกสารเหล่านั้น

Software measurement

Software measurement

- การวัดซอฟต์แวร์ (Software measurement) เกี่ยวข้องกับการหาค่าตัวเลขสำหรับคุณลักษณะของผลิตภัณฑ์ซอฟต์แวร์หรือกระบวนการ
- เพื่อช่วยเปรียบเทียบวัตถุประสงค์ระหว่างเทคนิคและกระบวนการ
- แม้ว่าบางบริษัทได้นำเสนอโปรแกรมการวัดแล้ว แต่องค์กรส่วนใหญ่ยังไม่ใช้การวัดซอฟต์แวร์เป็นระบบ
- มีมาตรฐานเกี่ยวกับเรื่องนี้เป็นจำนวนน้อย

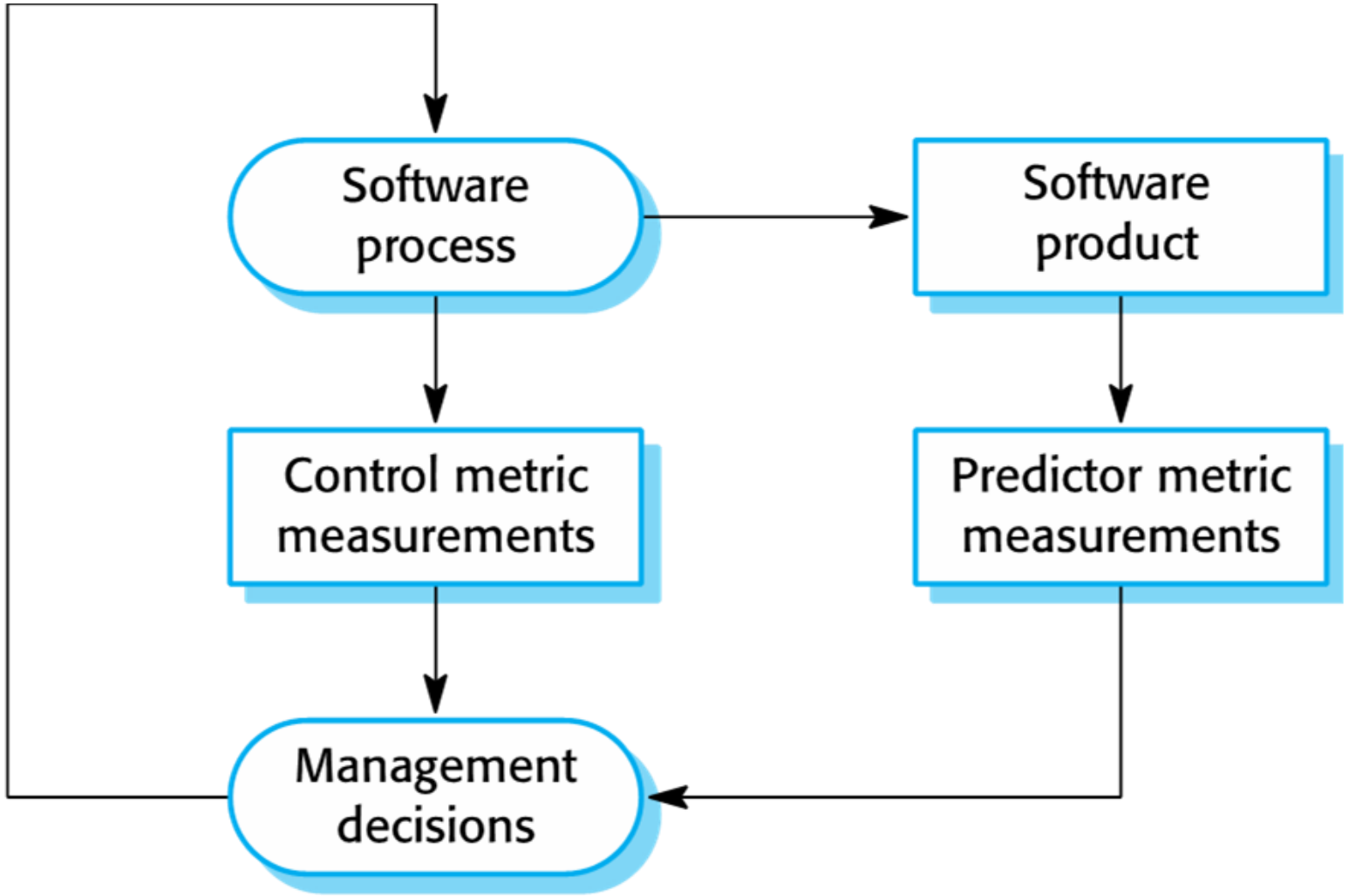
Software metric

- Software metric
- คือการวัดใด ๆ ที่เกี่ยวข้องกับระบบซอฟต์แวร์ กระบวนการ หรือเอกสารที่เกี่ยวข้อง
  - เช่น บรรทัดของรหัสในโปรแกรม, Fox index, จำนวนคน-วันที่จำเป็นในการพัฒนา
- อนุญาตให้ใช้ซอฟต์แวร์และกระบวนการซอฟต์แวร์ในเชิงปริมาณได้
- อาจใช้เพื่อคาดเดาคุณลักษณะของผลิตภัณฑ์หรือเพื่อควบคุมกระบวนการซอฟต์แวร์
- Product metrics สามารถใช้เพื่อการคาดการณ์ทั่วไปหรือระบุส่วนประกอบที่ไม่เป็นไปตามข้อกำหนด

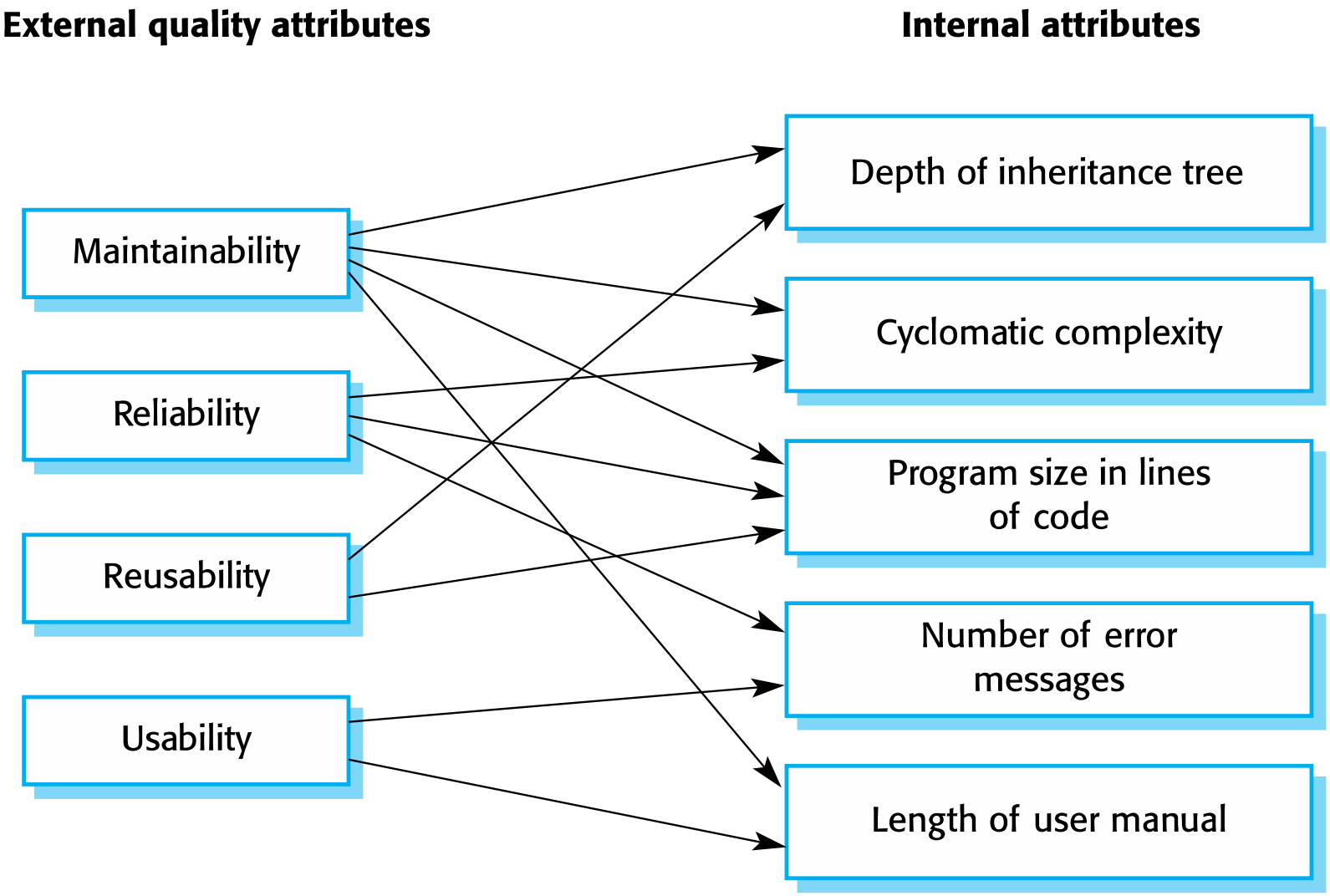
Types of process metric

- เวลาที่ใช้สำหรับกระบวนการเฉพาะที่จะแล้วเสร็จ
  - อาจเป็นเวลารวมในการดำเนินการ เวลาในปฏิทิน เวลาในการดำเนินการของวิศวกรเฉพาะราย และอื่น ๆ
- ทรัพยากรที่จำเป็นสำหรับกระบวนการเฉพาะ
  - ทรัพยากรอาจรวมถึง ทรัพยากรคน-วัน ค่าเดินทาง หรือทรัพยากรคอมพิวเตอร์
- จำนวนครั้งที่เกิดขึ้นของเหตุการณ์หนึ่ง ๆ เช่น
  - จำนวนข้อบกพร่องที่ค้นพบในระหว่างการตรวจสอบโค้ด
  - จำนวนของข้อกำหนดที่ต้องการเปลี่ยนแปลง
  - จำนวนรายงานข้อบกพร่องในระบบที่ส่งมอบ
  - จำนวนบรรทัดโดยเฉลี่ยของรหัสที่แก้ไข (ในการตอบสนองต่อความต้องการเปลี่ยนแปลง)

Predictor and control measurements



Relationships between internal and external software



Problems with measurement in industry

- เป็นไปไม่ได้ที่จะหาจำนวนผลตอบแทนจากการลงทุนในการใช้งานโปรแกรมเมตริกขององค์กร
- ไม่มีมาตรฐานสำหรับซอฟต์แวร์เมตริกหรือกระบวนการมาตรฐานสำหรับการวัดและการวิเคราะห์
- ในหลายบริษัท มีกระบวนการซอฟต์แวร์ที่ไม่ได้มาตรฐาน มีการกำหนดและควบคุมไม่ดี
- งานส่วนใหญ่ในการวัดซอฟต์แวร์ มุ่งเน้นไปที่เมตริกที่ใช้โค้ดและกระบวนการพัฒนาแบบ plan driven
  - อย่างไรก็ตามขณะนี้มีการพัฒนาให้สามารถใช้ได้กับ ERP หรือ COTS
- การนำการวัดซอฟต์แวร์มาใช้ จะเพิ่มค่าใช้จ่ายให้กับกระบวนการต่างๆ

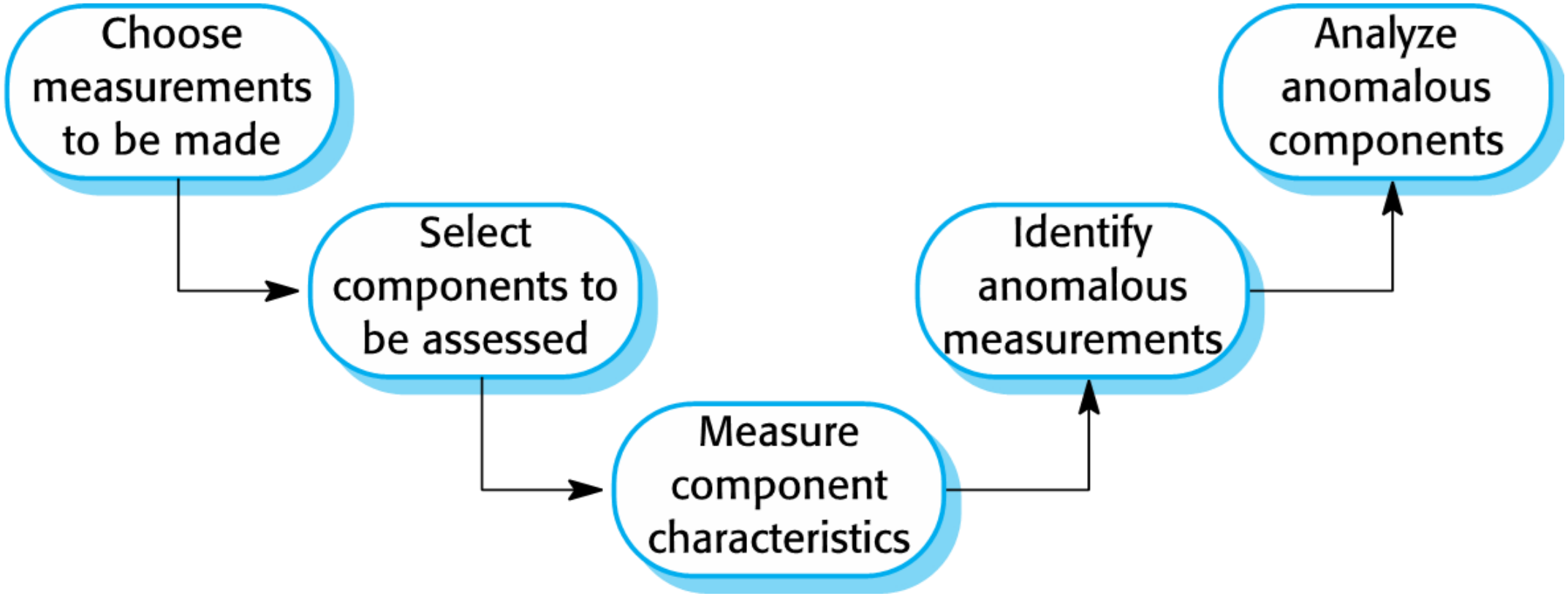
Empirical software engineering

- Software measurement และ software metrics เป็นพื้นฐานของวิศวกรรมซอฟต์แวร์เชิงประจักษ์
- ขอบเขตการวิจัยซึ่งการทดลองเกี่ยวกับระบบซอฟต์แวร์ และการรวบรวมข้อมูลเกี่ยวกับโครงการจริง ได้ถูกใช้เพื่อสร้างและตรวจสอบสมมติฐานเกี่ยวกับวิธีการและเทคนิคด้านวิศวกรรมซอฟต์แวร์
- การวิจัยเกี่ยวกับวิศวกรรมซอฟต์แวร์เชิงประจักษ์นี้ ไม่ได้ส่งผลกระทบอย่างมากนักต่อการปฏิบัติด้านวิศวกรรมซอฟต์แวร์
- เป็นการยากที่จะเชื่อมโยงการวิจัยเข้ากับโครงการที่แตกต่างจากงานวิจัย

Software component analysis

- ส่วนประกอบต่าง ๆ ของระบบ สามารถวิเคราะห์แยกกันโดยใช้เมตริกต่าง ๆ
- ค่าของเมตริกเหล่านี้อาจเทียบกับคอมโพเนนต์ที่แตกต่างกัน และอาจมีข้อมูลประวัติการวัดที่รวบรวมจากโครงการก่อนหน้า

The process of product measurement



Measurement ambiguity

- เมื่อรวบรวมข้อมูลเชิงปริมาณเกี่ยวกับซอฟต์แวร์และกระบวนการซอฟต์แวร์ เราต้องวิเคราะห์ข้อมูลนั้นเพื่อทำความเข้าใจกับความหมายของข้อมูล
- การตีความผิดข้อมูลและการอนุมานที่ไม่ถูกต้องเป็นเรื่องง่าย
- เราไม่สามารถดูข้อมูลด้วยตัวเองอย่างง่าย ๆ ต้องพิจารณาบริบทที่รวบรวมข้อมูลด้วย

Measurement surprises

- การลดจำนวนข้อผิดพลาดในโปรแกรม อาจทำให้จำนวนการโทรติดต่อเจ้าหน้าที่ช่วยเหลือเพิ่มขึ้น
  - โปรแกรมที่มีความน่าเชื่อถือมากและมีตลาดที่มีความหลากหลายมากขึ้นนั้น อาจมีเปอร์เซ็นต์ของผู้ใช้ที่โทรติดต่อฝ่ายบริการความช่วยเหลือลดลง แต่อาจมีจำนวนเพิ่มขึ้น
  - ระบบที่เชื่อถือได้มากขึ้น จะยิ่งถูกนำไปใช้ในลักษณะที่แตกต่างกัน โดยผู้ใช้ที่หลากหลาย เมื่อไม่ตรงตามที่ผู้ใช้คาดหวัง อาจนำไปสู่การโทรติดต่อฝ่ายบริการความช่วยเหลือเพิ่มเติม

Key points

- การจัดการคุณภาพของซอฟต์แวร์นั้นเกี่ยวข้องกับ
  - การทำให้มั่นใจได้ว่า ซอฟต์แวร์มีข้อบกพร่องน้อย
  - มีมาตรฐานในการบำรุงรักษา
  - มีความน่าเชื่อถือ
  - มีความสามารถด้าน portability
- มาตรฐานซอฟต์แวร์มีความสำคัญต่อการประกันคุณภาพซอฟต์แวร์ เนื่องจาก
  - มีการระบุแนวทางปฏิบัติที่ดีที่สุด
  - มาตรฐานเป็นพื้นฐานที่มั่นคงในการสร้างซอฟต์แวร์ที่มีคุณภาพดี

Key points

- การรีวิวกระบวนการซอฟต์แวร์ เกี่ยวข้องกับทีมผู้ตรวจสอบ โดยมีการตรวจสอบว่ามีการปฏิบัติตามมาตรฐานคุณภาพหรือไม่
  - รีวิวเป็นเทคนิคที่ใช้กันอย่างแพร่หลายในการประเมินคุณภาพ
- ในการตรวจสอบโปรแกรม (program inspection) หรือการตรวจสอบแบบ peer review ทีมเล็ก ๆ จะตรวจสอบโค้ดอย่างเป็นระบบ
  - ทีมจะอ่านรหัส (ลงรายละเอียด) และมองหาทั้งข้อผิดพลาดและสิ่งที่อาจจะขาดหายไป
  - ปัญหาที่ตรวจพบจะถูกกล่าวถึงในที่ประชุมทบทวนรหัส

Key points

- การจัดการคุณภาพ Agile ขึ้นอยู่กับการสร้างวัฒนธรรมที่มีคุณภาพ ซึ่งทีมพัฒนาทำงานร่วมกันเพื่อปรับปรุงคุณภาพซอฟต์แวร์
- การวัดซอฟต์แวร์สามารถใช้เพื่อรวบรวมข้อมูลเชิงปริมาณเกี่ยวกับซอฟต์แวร์และกระบวนการซอฟต์แวร์
- เราอาจสามารถใช้ค่าของเมตริกซอฟต์แวร์ เพื่อทำข้อสรุปเกี่ยวกับคุณภาพผลิตภัณฑ์และกระบวนการ
- เมตริกคุณภาพของผลิตภัณฑ์มีประโยชน์อย่างยิ่งสำหรับการคัดแยกส่วนประกอบที่ผิดปกติซึ่งอาจมีปัญหาด้านคุณภาพ
  - ส่วนประกอบเหล่านี้ควรได้รับการวิเคราะห์ในรายละเอียดเพิ่มเติม

Measurement ambiguity

- เมื่อรวบรวมข้อมูลเชิงปริมาณเกี่ยวกับซอฟต์แวร์และกระบวนการซอฟต์แวร์ เราต้องวิเคราะห์ข้อมูลนั้นเพื่อทำความเข้าใจกับความหมายของข้อมูล
- การตีความผิดข้อมูลและการอนุมานที่ไม่ถูกต้องเป็นเรื่องง่าย
- เราไม่สามารถดูข้อมูลด้วยตัวเองอย่างง่าย ๆ ต้องพิจารณาบริบทที่รวบรวมข้อมูลด้วย

คำถาม???