CPE3243 วิศวกรรมซอฟต์แวร์
(Software Engineering)

Piyavit Laung-Aram

Major of Computer Engineering

Faculty of Engineering

Ramkhamhaeng University, Thailand

การจัดการซอฟต์แวร์

(Software Management)

โครงการคืออะไร? (What is Project?)

- โครงการคือกลุ่มของงานที่ต้องทำเพื่อให้ได้ผลลัพธ์ที่ชัดเจน
- โครงการยังกำหนดเป็นชุดของอินพุตและเอาต์พุตที่จำเป็นเพื่อให้บรรลุเป้าหมาย
- โครงการสามารถเปลี่ยนแปลงได้ตั้งแต่ง่ายไปยาก และสามารถดำเนินการได้คนเดียวหรือร้อยคน
- โครงการมักจะอธิบายและอนุมัติโดย ผู้จัดการโครงการหรือทีมงานบริหารโครงการจะดำเนินการได้ตามวัตถุประสงค์จะต้องขึ้นกับอยู่กับทีมงานที่จะจัดการด้านต่าง ๆ เช่น ด้านลอจิสติกส์และด้านการดำเนินการโครงการ เพื่อให้เสร็จตรงเวลา
- สำหรับการพัฒนาโครงการที่ดี บางทีมแบ่งโครงการออกเป็นงานเฉพาะย่อย ๆ เพื่อให้สามารถบริหารจัดการและใช้จุดแข็งของทีมงานได้อย่างเต็มที่

การจัดการโครงการซอฟต์แวร์คืออะไร?

(What is software project management?)

- การจัดการโครงการซอฟต์แวร์เป็นศิลปะ แนวทางและระเบียบที่เกี่ยวข้อง ในการวางแผนและกำกับดูแลโครงการซอฟต์แวร์ การจัดการโครงการซอฟต์แวร์มีกิจกรรมย่อยต่าง ๆ ที่เกิดขึ้น เช่น การวางแผน การดำเนินการ การตรวจสอบ และการควบคุมโครงการ
- เป็นขั้นตอนในการจัดการ จัดสรร และกำหนดเวลา การจัดการทรัพยากรเพื่อพัฒนาซอฟต์แวร์คอมพิวเตอร์ให้ตรงตามข้อกำหนดต่าง ๆ ซึ่งการจัดการโครงการซอฟต์แวร์ จะช่วยให้ลูกค้าและนักพัฒนาทราบระยะเวลาในการดำเนินการโครงการและต้นทุนของโครงการ ที่ชัดเจน

ความต้องการสำหรับการจัดการโครงการซอฟต์แวร์?

มีความต้องการสามประการสำหรับการจัดการโครงการซอฟต์แวร์คือ

- เวลา (Time)
- ต้นทุนหรือค่าใช้จ่าย (Cost)
- คุณภาพ (Quality)

เป็นส่วนสำคัญขององค์กรที่พัฒนาซอฟต์แวร์ในการส่งมอบผลิตภัณฑ์ที่มีคุณภาพ โดยรักษาต้นทุนให้อยู่ในงบประมาณของลูกค้า และส่งมอบโครงการตามกำหนดการ มีหลายปัจจัยทั้งภายนอกและภายใน ซึ่งอาจส่งผลกระทบต่อปัจจัย ทั้งนี้ปัจจัยสามประการใด ๆ อาจส่งผลกระทบอย่างรุนแรงต่ออีกสองปัจจัยที่เหลือ

ผู้จัดการโครงการ (Project Manager)

ผู้จัดการโครงการคือตัวละครที่มีหน้าที่รับผิดชอบโดยรวมในการวางแผน ออกแบบ ดำเนินการ ตรวจสอบ ควบคุม และปิดโครงการ ผู้จัดการโครงการแสดงถึงมีบทบาทสำคัญเป็นอย่างมากในการที่จำทำให้โครงการบรรลุผลสำเร็จ ผู้จัดการโครงการเป็นตัวละครที่รับผิดชอบในการตัดสินใจทั้งโครงการขนาดใหญ่และขนาดเล็ก ผู้จัดการโครงการใช้เพื่อจัดการความเสี่ยงและลดความไม่แน่นอน ทุกการตัดสินใจของผู้จัดการโครงการจะต้องสร้างผลกำไรให้กับโครงการโดยตรง

บทบาทของผู้จัดการโครงการ (Role of a Project Manager)

1. ผู้นำ(Leader): ผู้จัดการโครงการต้องเป็นผู้นำทีมและควรให้คำแนะนำเพื่อให้เข้าใจถึงสิ่งที่คาดหวังจากผู้ดำเนินการในโครงการทั้งหมด

2. ตัวกลางหรือสื่อกลาง(Medium): ผู้จัดการโครงการเป็นตัวกลางหรือสื่อกลางระหว่างลูกค้าและทีมงานของเขา เขาต้องประสานงานและถ่ายโอนข้อมูลที่เหมาะสมทั้งหมดจากลูกค้าไปยังทีมงานของเขาและรายงานต่อผู้บริหารระดับสูง

3. พี่เลี้ยง(Mentor): ผู้จัดการโครงการจะเป็นผู้ให้คำแนะนำทีมของเขาในแต่ละขั้นตอนและทำให้แน่ใจว่าทีมของเขามีสิ่งที่จำเป็นในการดำเนินการโครงการ นอกจากนี้ผู้จัดการโครงการจะเป็นผู้ให้คำแนะนำแก่ทีมของเขาและชี้ให้พวกเขาไปในทิศทางที่ถูกต้อง

ความรับผิดชอบของผู้จัดการโครงการ

- การจัดการความเสี่ยงและปัญหา
- สร้างทีมงานสำหรับโครงการและมอบหมายงานให้กับสมาชิกในทีม
- การวางแผนกิจกรรมและการจัดลำดับการทำงาน
- ติดตามและรายงานความคืบหน้า
- ปรับเปลี่ยนแผนโครงการเพื่อให้สอดคล้องกับสถานการณ์

กิจกรรม (Activities) ต่าง ๆ ใน Software Project Management

Software Project Management ประกอบด้วยกิจกรรมต่างๆ มากมาย ซึ่งรวมถึงการวางแผนโครงการ การกำหนดขอบเขตของผลิตภัณฑ์ การประมาณราคาในแง่ต่างๆ การจัดกำหนดการงาน ฯลฯ

รายการกิจกรรมมีดังนี้

- Project planning and Tracking-การวางแผนโครงการและการติดตาม
- Project Resource Management-การบริหารจัดการทรัพยากรสำหรับโครงการ
- Scope Management-การบริหารจัดการขอบเขตของโครงการ
- Estimation Management-การบริหารจัดการประมาณการ
- Project Risk Management-การบริหารความเสี่ยงโครงการ
- Scheduling Management-การบริหารจัดการกำหนดการ
- Project Communication Management-การบริหารจัดการการสื่อสารในโครงการ
- Configuration Management-การบริหารจัดการคุณลักษณะของผลิตภัณฑ์หรือโครงการ

Project planning and Tracking-การวางแผนโครงการและการติดตาม

มันคือชุดของกระบวนการต่างๆ หรือเราสามารถพูดได้ว่ามันเป็นงานที่ดำเนินการก่อนการสร้างผลิตภัณฑ์จะเริ่มขึ้น

Project Resource Management-การบริหารจัดการทรัพยากรสำหรับโครงการ

ในการพัฒนาซอฟต์แวร์ องค์ประกอบทั้งหมดจะเรียกว่าทรัพยากรสำหรับโครงการ อาจเป็นทรัพยากรบุคคล เครื่องมือที่มีประสิทธิภาพ และไลบรารีต่าง ๆ

การจัดการทรัพยากรรวมถึง:

- สร้างทีมโครงการและมอบหมายความรับผิดชอบให้กับสมาชิกในทีมทุกคน
- การพัฒนาแผนทรัพยากรมาจากแผนโครงการ
- การปรับการจัดการทรัพยากรในโครงการ

Scope Management-การบริหารจัดการขอบเขตของโครงการ

อธิบายขอบเขตของโครงการ การจัดการขอบเขตมีความสำคัญเนื่องจากขอบเขตจะกำหนดไว้อย่างชัดเจนว่าอะไรจะทำอะไรและไม่ควรทำ การจัดการขอบเขตในการสร้างโครงการ เพื่อเป็นการจำกัดทั้งคุณภาพและปริมาณของงาน ซึ่งอาจเป็นเพียงการจัดทำเป็นเอกสารและหลีกเลี่ยงการต้นทุนในการดำเนินการโครงการเกินกว่าต้นทุนและระยะเวลาที่กำหนด

Estimation Management-การบริหารจัดการประมาณการ

- นี่ไม่ใช่แค่การประมาณราคาเท่านั้น เพราะเมื่อใดก็ตามที่เราเริ่มพัฒนาซอฟต์แวร์ แต่เรายังคำนวณขนาด (บรรทัดของโค้ด) ความพยายาม เวลา และต้นทุนด้วย
- ถ้าเราพูดถึงขนาด บรรทัดของโค้ดจะขึ้นอยู่กับความต้องการของผู้ใช้หรือซอฟต์แวร์
- ถ้าเราพูดถึงความพยายาม เราควรรู้เกี่ยวกับขนาดของซอฟต์แวร์ เพราะเมื่อพิจารณาจากขนาดแล้ว เราสามารถประเมินได้อย่างรวดเร็วว่าทีมใหญ่ๆ จำเป็นแค่ไหนในการผลิตซอฟต์แวร์
- ถ้าเราพูดถึงเวลา เมื่อประมาณขนาดและความพยายาม เวลาที่ใช้ในการพัฒนาซอฟต์แวร์สามารถกำหนดได้อย่างง่ายดาย

และถ้าพูดถึงเรื่องต้นทุน ก็มีองค์ประกอบทั้งหมดเช่น:

Estimation Management-การบริหารจัดการประมาณการ (ต่อ)

ถ้าพูดถึงเรื่องต้นทุน ก็มีองค์ประกอบทั้งหมดเช่น:

- ขนาดของซอฟต์แวร์
- คุณภาพ
- ฮาร์ดแวร์
- การสื่อสาร
- การฝึกอบรม
- ซอฟต์แวร์และเครื่องมือเพิ่มเติม
- ฝีมือแรงงาน

Project Risk Management-การบริหารความเสี่ยงโครงการ

การจัดการความเสี่ยงประกอบด้วยกิจกรรมทั้งหมด เช่น การระบุ การวิเคราะห์ และการเตรียมแผนสำหรับความเสี่ยงที่คาดการณ์ได้และคาดเดาไม่ได้ในโครงการ

หลายจุดแสดงความเสี่ยงในโครงการ:

- ทีมผู้มีประสบการณ์ออกจากโครงการ และทีมใหม่เข้าร่วมโครงการ
- การเปลี่ยนแปลงความต้องการ
- การเปลี่ยนแปลงของเทคโนโลยีและสิ่งแวดล้อม
- การแข่งขันทางการตลาด

Scheduling Management-การบริหารจัดการกำหนดการ

การจัดการกำหนดการในซอฟต์แวร์หมายถึงกิจกรรมทั้งหมดที่ต้องดำเนินการตามลำดับที่ระบุและภายในระยะเวลาที่กำหนดสำหรับแต่ละกิจกรรม ผู้จัดการโครงการกำหนดงานหลายอย่างและจัดการโดยคำนึงถึงปัจจัยต่างๆ

งานที่ต้องคำนึงสำหรับการวางกำหนดการ

- ค้นหางานต่าง ๆ และเชื่อมโยงงานเหล่านั้นเข้าด้วยกัน
- แบ่งหน่วยการทำงานเป็นหน่วยย่อย ๆ
- กำหนดจำนวนหน่วยงานสำหรับแต่ละงาน
- คำนวณเวลาทั้งหมดตั้งแต่ต้นจนจบ
- แบ่งโครงการออกเป็นโมดูล

Project Communication Management-การบริหารจัดการการสื่อสารในโครงการ

การสื่อสารเป็นปัจจัยสำคัญต่อความสำเร็จของโครงการ เป็นสะพานเชื่อมระหว่างลูกค้า องค์กร สมาชิกในทีม และผู้มีส่วนได้ส่วนเสียอื่นๆ ของโครงการ เช่น ซัพพลายเออร์ ฮาร์ดแวร์ตั้งแต่การวางแผนจนถึงการปิด

การสื่อสารมีบทบาทสำคัญ ในทุกขั้นตอนการสื่อสารจะต้องชัดเจนและเข้าใจ การสื่อสารที่ผิดพลาดสามารถสร้างความผิดพลาดครั้งใหญ่ในโครงการได้

Configuration Management-การบริหารจัดการคุณลักษณะของผลิตภัณฑ์หรือโครงการ

เกี่ยวกับการควบคุมการเปลี่ยนแปลงในซอฟต์แวร์ เช่น ข้อกำหนด การออกแบบ และการพัฒนาผลิตภัณฑ์ เป้าหมายหลักคือการเพิ่มประสิทธิภาพการทำงานโดยมีข้อผิดพลาดน้อยลง

เหตุผลบางประการแสดงให้เห็นถึงความจำเป็นในการบริหารจัดการคุณลักษณะของผลิตภัณฑ์หรือโครงการ

- ผู้คนทำงานกับซอฟต์แวร์ที่มีการอัพเดทอย่างต่อเนื่อง
- ช่วยสร้างการประสานงานระหว่างซัพพลายเออร์
- การเปลี่ยนแปลงความต้องการ งบประมาณ กำหนดการที่ต้องรองรับ
- ซอฟต์แวร์ควรทำงานบนหลายระบบ

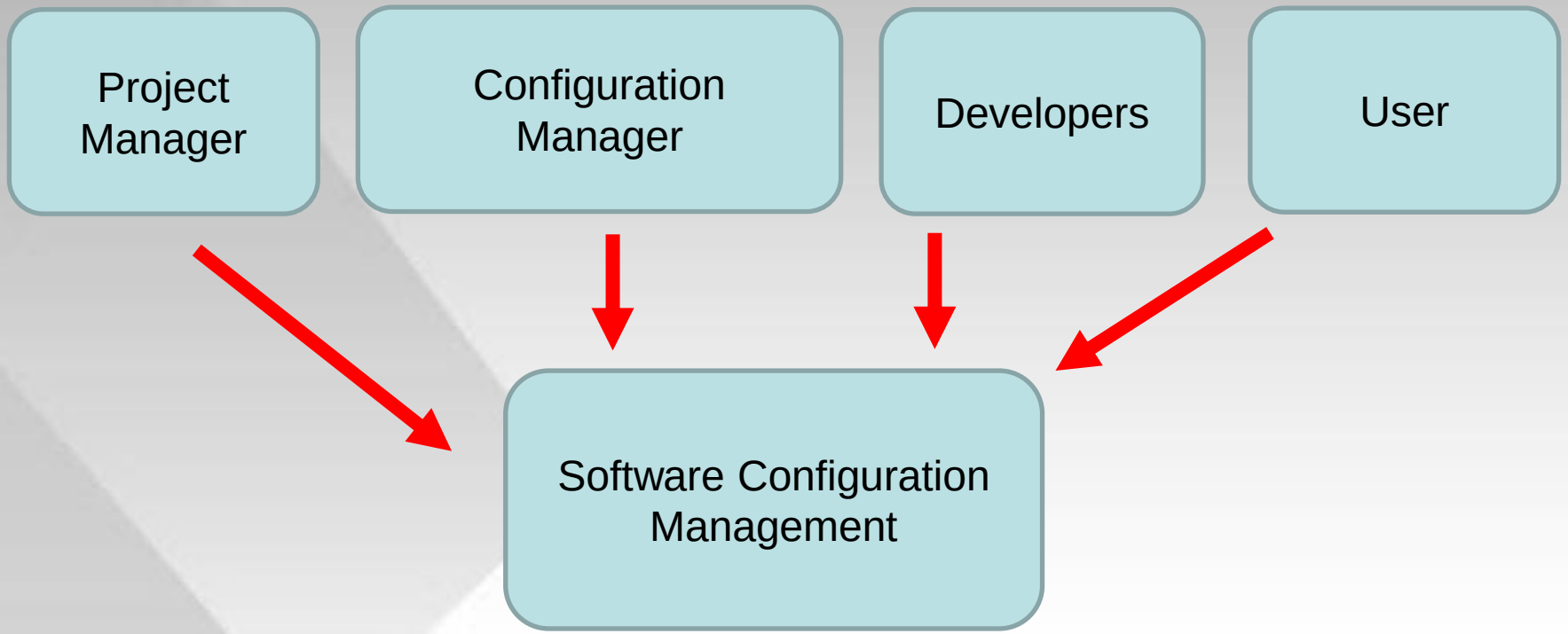
Configuration Management-การบริหารจัดการคุณลักษณะของผลิตภัณฑ์หรือโครงการ(ต่อ)

งานที่ดำเนินการในการบริหารจัดการคุณลักษณะของผลิตภัณฑ์หรือโครงการ :

- การระบุคุณลักษณะของผลิตภัณฑ์หรือโครงการ
- การวางคุณลักษณะพื้นฐานของผลิตภัณฑ์หรือโครงการ
- การเปลี่ยนแปลงการควบคุมตามคุณลักษณะของผลิตภัณฑ์หรือโครงการ
- การปรับสถานะทางบัญชีให้สอดคล้องกับคุณลักษณะของผลิตภัณฑ์หรือโครงการ
- การปรับการตรวจสอบและการทวนสอบคุณลักษณะของผลิตภัณฑ์หรือโครงการ

Configuration Management-การบริหารจัดการคุณลักษณะของผลิตภัณฑ์หรือโครงการ(ต่อ)

ผู้ที่เกี่ยวข้องกับการบริหารจัดการลักษณะโครงการ :



เครื่องมือที่ใช้ในการจัดการโครงการ

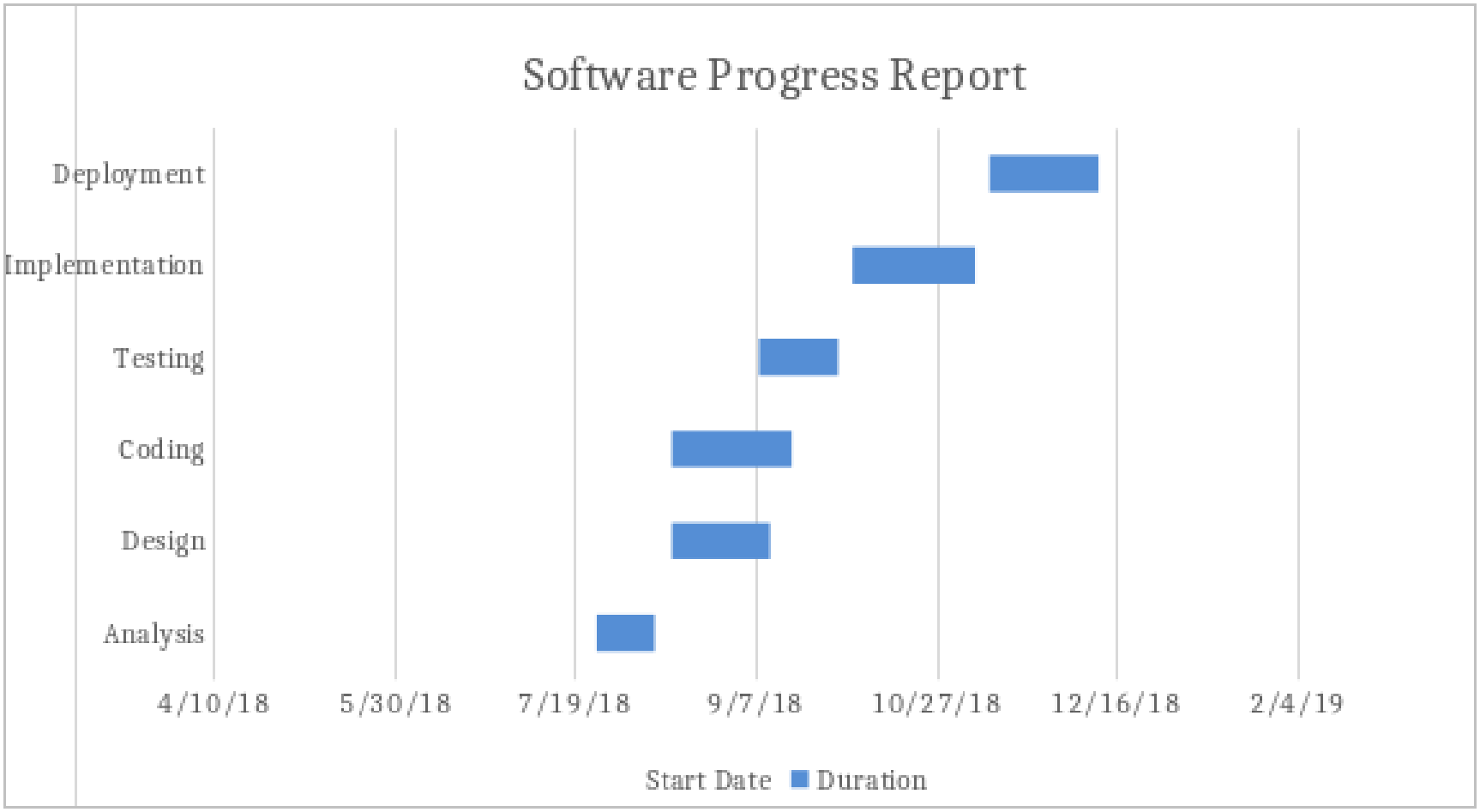
เพื่อให้เราสามารถบริหารจัดการโครงการได้อย่างมีประสิทธิภาพ เราจำเป็นต้องใช้เครื่องมือมาช่วยในการบริหารจัดการโครงการ อาทิเช่น

- Gantt chart
- PERT chart
- Logic Network
- Product Breakdown Structure
- Work Breakdown Structure
- Resource Histogram
- Critical Path Analysis

Gantt chart

Gantt chart พัฒนาขึ้นครั้งแรกโดยเฮนรี แกนต์ ในปี พ.ศ. 2460 แผนภูมิแกนต์มักใช้ในการจัดการโครงการ และเป็นหนึ่งในวิธีที่ได้รับความนิยมและเป็นประโยชน์ที่สุดในการแสดงกิจกรรมที่แสดงตามเวลา แต่ละกิจกรรมแสดงโดยแถบ Gantt chart เป็นเครื่องมือที่มีประโยชน์เมื่อคุณต้องการดูภาพรวมของโครงการหนึ่งโครงการหรือหลายโครงการ ช่วยให้คุณดูได้ว่างานใดต้องพึ่งพากันและกันและเหตุการณ์ใดที่กำลังจะเกิดขึ้น

Gantt chart (ต่อ)

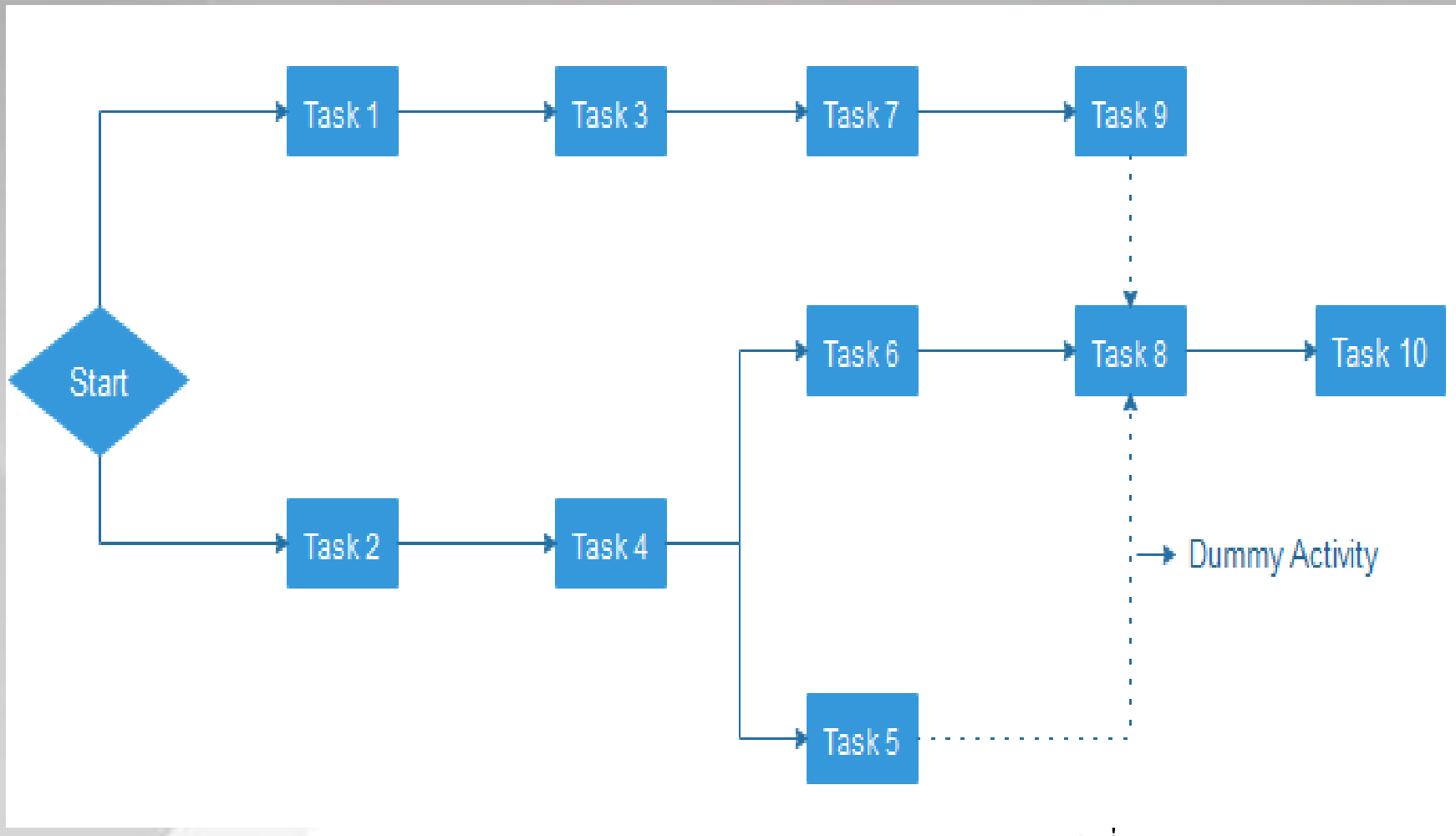


แหล่งที่มา https://www.javatpoint.com/

PERT chart

PERT เป็นตัวย่อของเทคนิคการทบทวนการประเมินผลโครงการ ในปี 1950 กองทัพเรือสหรัฐฯ พัฒนาขึ้นเพื่อจัดการกับโครงการขีปนาวุธใต้น้ำ Polaris ในการจัดการโครงการ แผนภูมิ PERT แสดงเป็นไดอะแกรมเครือข่ายเกี่ยวกับจำนวนโหนด ซึ่งแสดงถึงเหตุการณ์ทิศทางของเส้นแสดงถึงลำดับของงาน ในตัวอย่างข้างต้น งานระหว่าง "งานที่ 1 ถึงงาน 9" ต้องเสร็จสิ้น และสิ่งเหล่านี้เรียกว่างานที่ต้องพึ่งพาหรืองานต่อเนื่อง ระหว่างภารกิจที่ 4 และ 5 และภารกิจที่ 4 และ 6 โหนดทั้งสองจะไม่ขึ้นต่อกันและสามารถทำงานได้พร้อมกัน สิ่งเหล่านี้เรียกว่างานแบบขนานหรือพร้อมกัน หากไม่มีทรัพยากรหรือเวลาที่เสร็จสิ้น งานต้องเสร็จสิ้นในลำดับซึ่งถือเป็นการพึ่งพาเหตุการณ์ และสิ่งเหล่านี้เรียกว่ากิจกรรม Dummy และแสดงด้วยเส้นประ

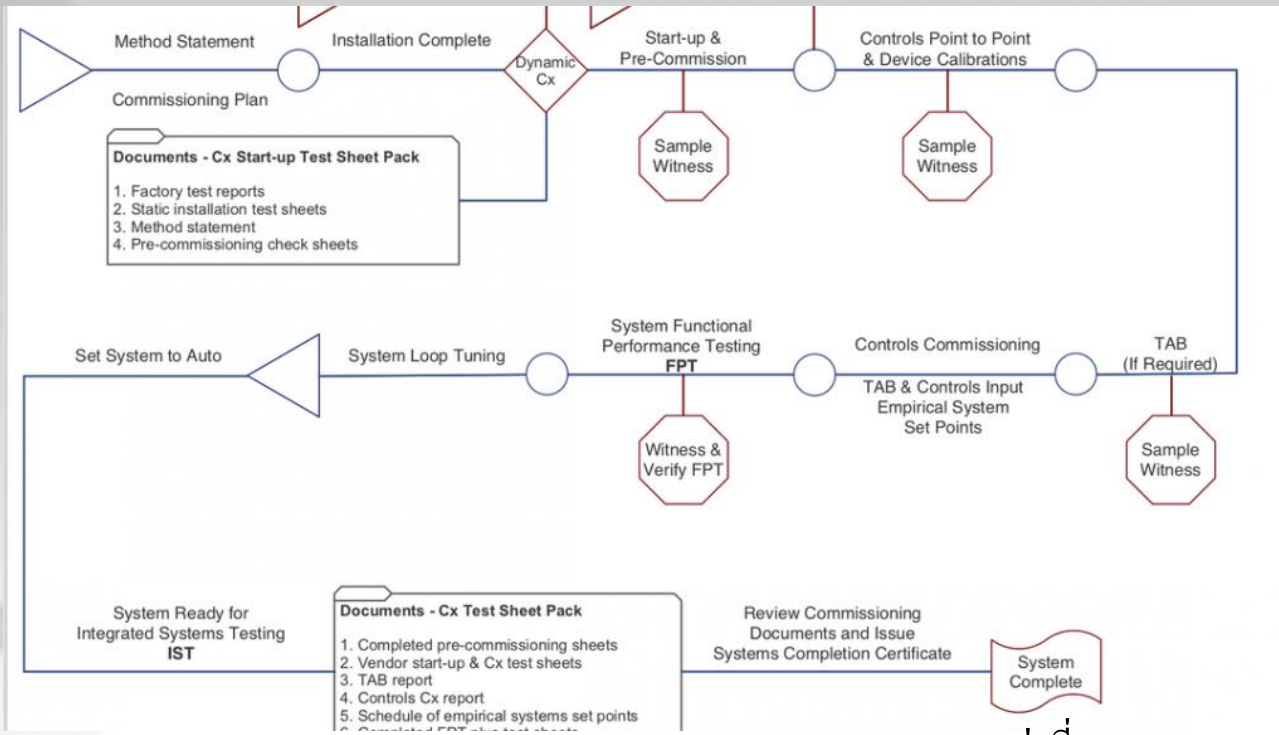
PERT chart (ต่อ)



แหล่งที่มา https://www.javatpoint.com/

Logic Network

Logic Network แสดงลำดับของกิจกรรมในช่วงเวลาหนึ่ง แสดงลำดับของกิจกรรมที่จะทำ การแยกแยะเหตุการณ์และการตรึงโครงการเป็นการใช้งานหลักสองประการ นอกจากนี้ ยังช่วยให้เข้าใจการขึ้นต่อกันของงาน ช่วงเวลา และเวิร์กโฟลว์ของโครงการโดยรวม

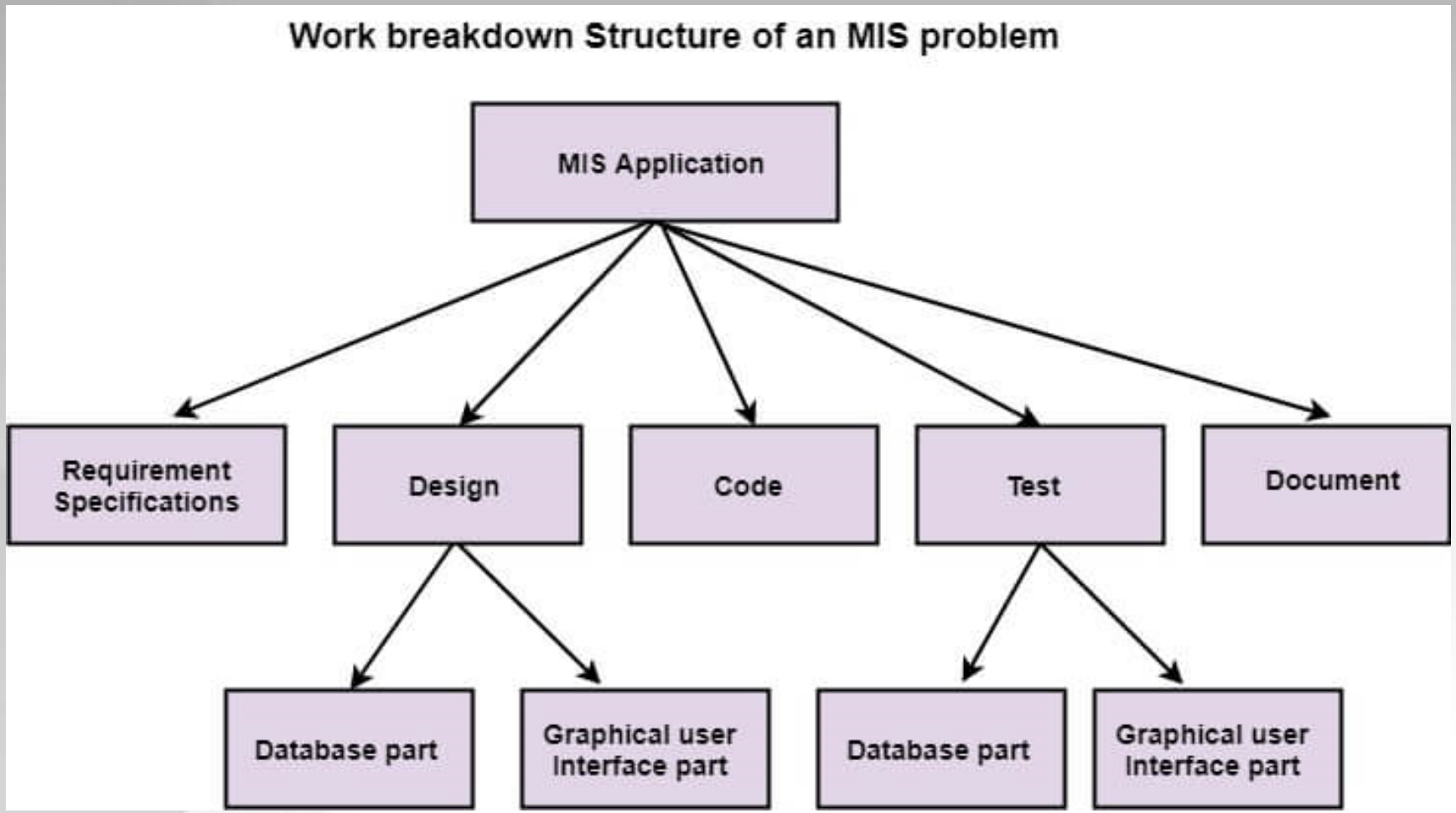


แหล่งที่มา https://bldwhisperer.com/

Product Breakdown Structure

Product Breakdown Structure (PBS) เป็นเครื่องมือในการจัดการและเป็นส่วนหนึ่งของการออกแบบโครงการ เป็นระบบที่เน้นงานสำหรับการแบ่งโปรเจ็กต์ออกเป็นส่วนๆ ของผลิตภัณฑ์ โครงสร้างการแบ่งผลิตภัณฑ์อธิบายงานย่อยหรือแพ็คเกจงาน และแสดงถึงการเชื่อมต่อระหว่างแพ็คเกจงาน ภายในโครงสร้างการแจกแจงผลิตภัณฑ์ งานโครงการได้แสดงภาพแบบไดอะแกรมพร้อมรายการประเภทต่างๆ โครงสร้างการแบ่งผลิตภัณฑ์ก็เหมือนกับโครงสร้างการแบ่งงาน-Work Breakdown Structure (WBS)

Product Breakdown Structure (ต่อ)



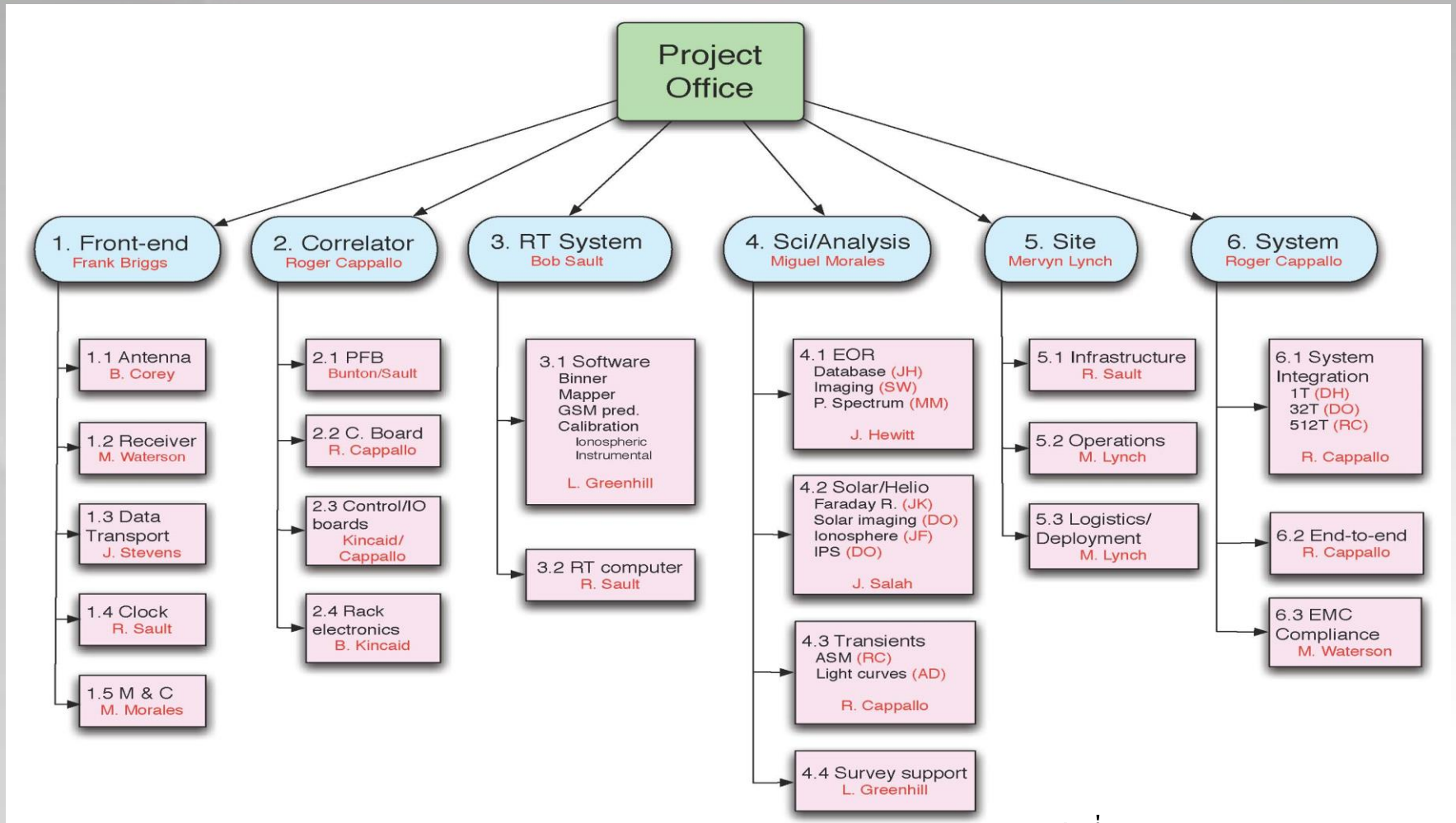
แหล่งที่มา https://www.javatpoint.com/

Work Breakdown Structure

เป็นเครื่องมือในการจัดประเภทงานของทีมออกเป็นส่วนๆ ที่ยืดหยุ่นได้ "Project Management Body of Knowledge (PMBOK)" เป็นกลุ่มคำศัพท์ที่อธิบายโครงสร้างการแบ่งงานว่าเป็น มีสองวิธีในการสร้างโครงสร้างการแบ่งงาน ? จากบนลงล่างและวิธีการจากล่างขึ้นบน

- ในแนวทางจากบนลงล่าง WBS ได้มาจากการบี้โปรเจ็กต์โดยรวมให้เป็นโปรเจ็กต์ย่อยหรืองานระดับล่าง
- แนวทางจากล่างขึ้นบนจะเหมือนกับการฝึกระดมความคิดโดยขอให้สมาชิกในทีมเขียนรายการงานระดับต่ำซึ่งจำเป็นต่อการทำโครงงานให้เสร็จ

Work Breakdown Structure (ต่อ)

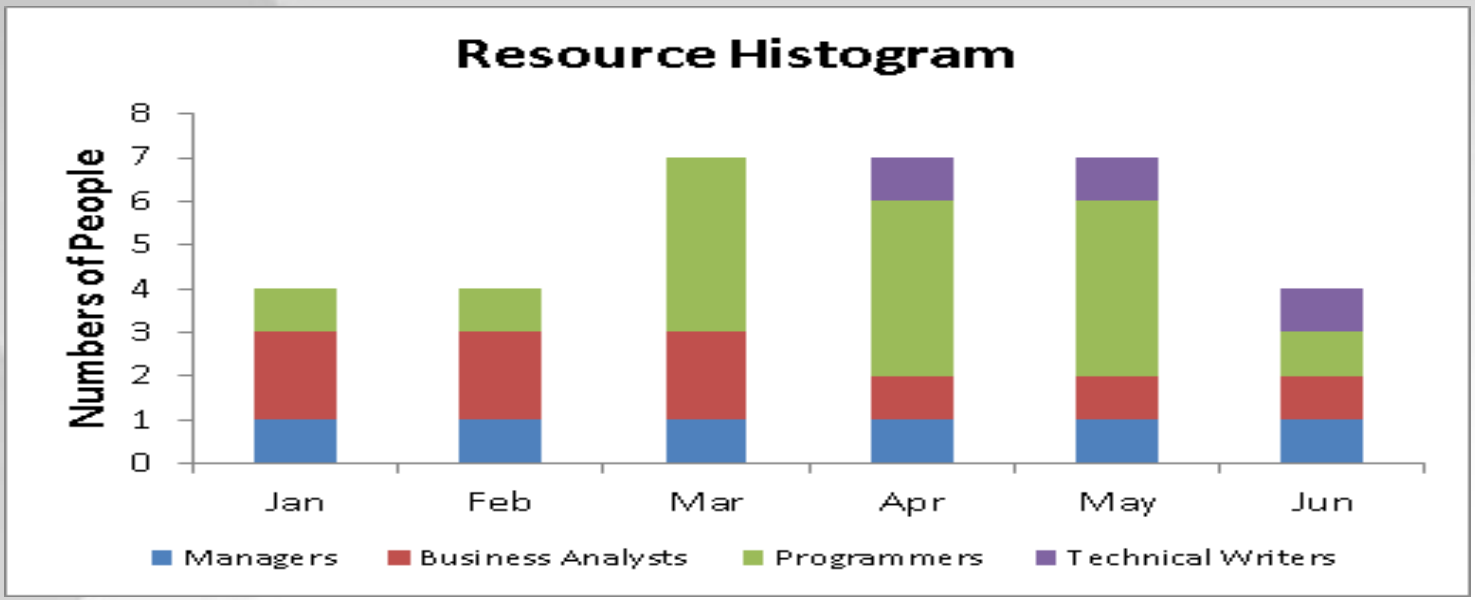


แหล่งที่มา https://tex.stackexchange.com/

Resource Histogram

Resource Histogram คือ แผนภูมิแท่งที่ใช้สำหรับแสดงระยะเวลาที่ทรัพยากรถูกระบุการใช้งานในช่วงเวลาที่กำหนดไว้ล่วงหน้า

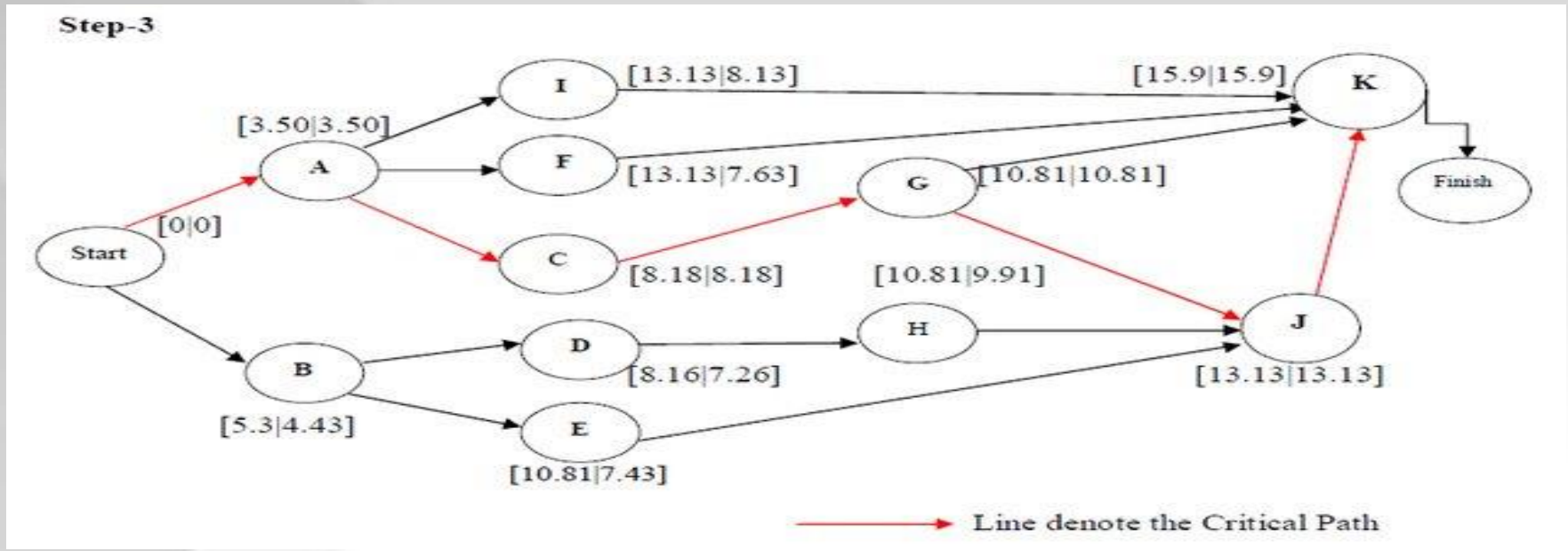
Resource Histogram ยังสามารถระบุถึงความพร้อมในการใช้งานของทรัพยากร ซึ่งใช้สำหรับเปรียบเทียบการใช้งานตามวัตถุประสงค์ที่แตกต่างกัน



แหล่งที่มา https://excelmaster.co/

Critical Path Analysis

การวิเคราะห์เส้นทางวิกฤติ-Critical Path Analysis เป็นเทคนิคที่ใช้ในการจัดหมวดหมู่กิจกรรมที่จำเป็นในการทำงานให้เสร็จ รวมทั้งจัดประเภทเวลาที่จำเป็นในการเสร็จสิ้นแต่ละกิจกรรมและความสัมพันธ์ระหว่างกิจกรรม เรียกอีกอย่างว่าวิธีเส้นทางวิกฤติ CPA ช่วยในการคาดการณ์ว่าโครงการจะเสร็จสิ้นตรงเวลาที่กำหนดไว้หรือไม่



แหล่งที่มา https://www.researchgate.net/

ซอฟต์แวร์เมตริกหรือตัวชี้วัดซอฟต์แวร์

(Software Metrics)

ซอฟต์แวร์เมตริก

ซอฟต์แวร์เมตริก คือ การวัดคุณลักษณะของซอฟต์แวร์ที่สามารถวัดได้หรือนับได้ ซอฟต์แวร์เมตริกมีคุณค่าหลายประการ รวมถึงการวัดประสิทธิภาพของซอฟต์แวร์ การวางแผนรายการต่าง ๆ ของงาน การวัดประสิทธิภาพการทำงาน และการใช้งานอื่นๆ อีกมากมายภายในกระบวนการพัฒนาซอฟต์แวร์ เมตริกจำนวนมากเชื่อมต่อกันทั้งหมด ซอฟต์แวร์เมตริกคล้ายกับฟังก์ชันการจัดการทั้งสี่ คือ การวางแผน องค์กร การควบคุม หรือการปรับปรุง

การจำแนกประเภทของซอฟต์แวร์เมตริก

เมตริกซอฟต์แวร์สามารถจำแนกได้เป็น 2 ประเภทดังนี้

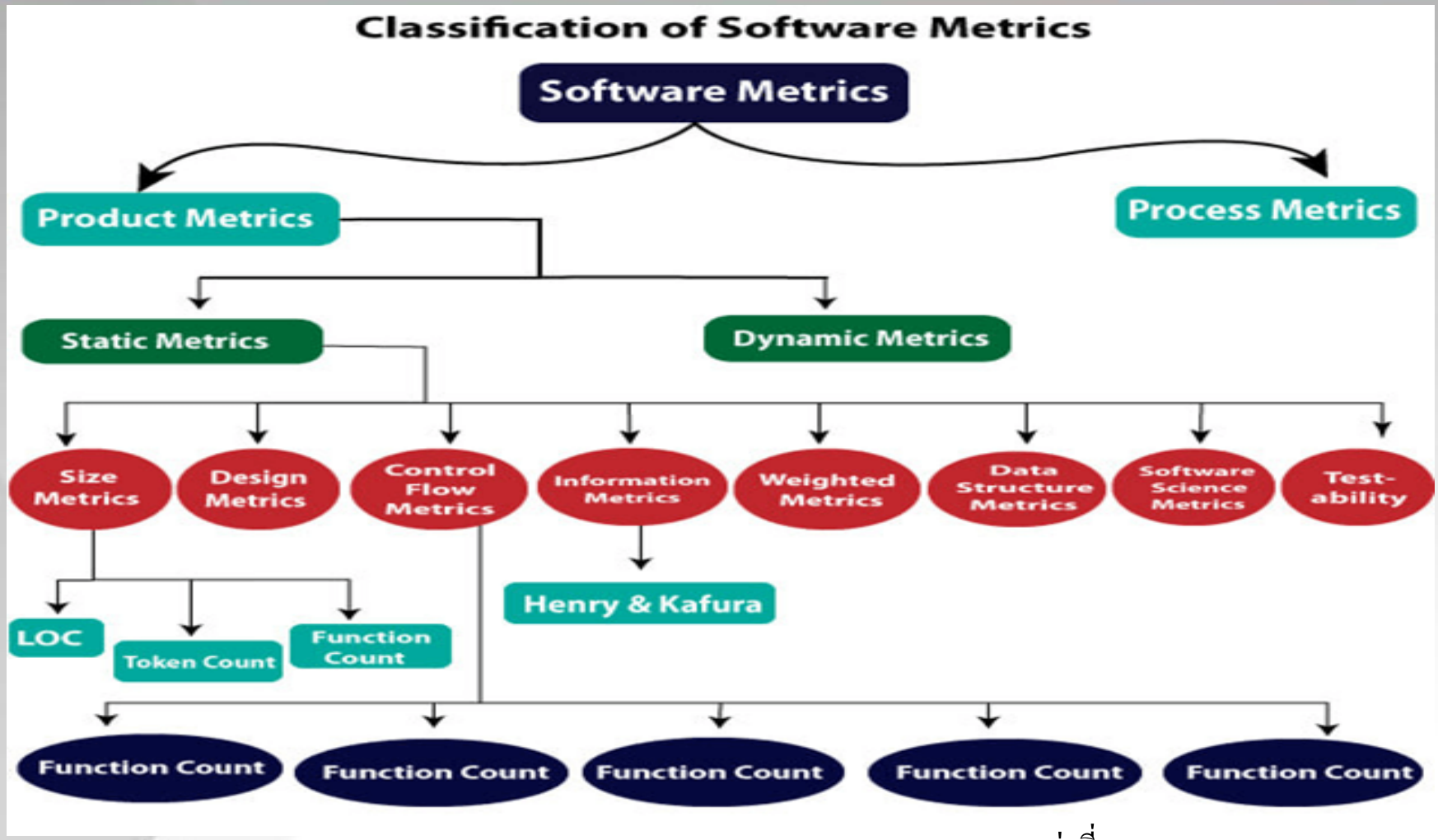
1. Product Metrics: เป็นการวัดคุณสมบัติต่างๆ ของผลิตภัณฑ์ซอฟต์แวร์ คุณสมบัติของซอฟต์แวร์ที่สำคัญสองประการคือ:

- ขนาดและความซับซ้อนของซอฟต์แวร์
- คุณภาพและความน่าเชื่อถือของซอฟต์แวร์

เมตริกเหล่านี้สามารถคำนวณได้ในขั้นตอนต่างๆ ของ SDLC

2. Process Metrics: เป็นการวัดลักษณะต่างๆ ของกระบวนการพัฒนาซอฟต์แวร์ ตัวอย่างเช่น ประสิทธิภาพของการตรวจจับข้อบกพร่อง ใช้สำหรับวัดคุณสมบัติของวิธีการ เทคนิค และเครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์

การจำแนกประเภทของซอฟต์แวร์เมตริก (ต่อ)



แหล่งที่มา https://www.javatpoint.com/

ประเภทของเมตริก

- เมตริกภายใน-Internal metrics : เมตริกภายในคือตัววัดที่ใช้วัดคุณสมบัติที่มีความสำคัญมากสำหรับนักพัฒนาซอฟต์แวร์ ตัวอย่างเช่น การวัด Lines of Code (LOC)
- เมตริกภายนอก:เมตริกภายนอกคือตัววัดที่ใช้สำหรับวัดคุณสมบัติที่มีความสำคัญต่อผู้ใช้มากกว่า เช่น การพกพา ความน่าเชื่อถือ ฟังก์ชัน การใช้งาน ฯลฯ
- เมตริกแบบผสม:เมตริกแบบผสมคือตัววัดที่รวมตัววัดผลิตภัณฑ์ กระบวนการ และทรัพยากร ตัวอย่างเช่น ราคาต่อ FP โดยที่ FP ย่อมาจาก Function Point Metric
- เมตริกโครงการ: เมตริกโครงการคือตัววัดที่ใช้เพื่อตรวจสอบความคืบหน้าของโครงการเพื่อรวบรวมตัวชี้วัดต่างๆ เช่น เวลาและต้นทุนในขณะที่โครงการดำเนินไป ผู้จัดการโครงการจะตรวจสอบความคืบหน้าเป็นระยะๆ และจะเปรียบเทียบความพยายาม ต้นทุน และเวลาใหม่กับของเดิม เพื่อลดต้นทุนในการพัฒนา ซึ่งจะปรับปรุงคุณภาพของโครงการได้ เมื่อคุณภาพดีขึ้น จำนวนข้อผิดพลาดและเวลาตลอดจนต้นทุนที่ต้องการก็ลดลงด้วย

ข้อดีของซอฟต์แวร์เมตริก

ใช้ในการศึกษาเปรียบเทียบวิธีการออกแบบต่างๆ ของระบบซอฟต์แวร์ ดังนี้

- ใช้สำหรับการวิเคราะห์ การเปรียบเทียบ และการศึกษาเชิงวิพากษ์ของภาษาโปรแกรมต่างๆ ที่เกี่ยวข้องกับคุณลักษณะ
- ใช้ในการเปรียบเทียบและประเมินความสามารถและประสิทธิผลของบุคลากรที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์
- ใช้ในการจัดทำข้อกำหนดคุณภาพซอฟต์แวร์
- ใช้ในการตรวจสอบการปฏิบัติตามข้อกำหนดและข้อกำหนดของระบบซอฟต์แวร์
- ใช้ในการอนุมานเกี่ยวกับความพยายามในการออกแบบและพัฒนาระบบซอฟต์แวร์
- ใช้ในการรับแนวคิดเกี่ยวกับความซับซ้อนของโค้ด

ข้อดีของซอฟต์แวร์เมตริก (ต่อ)

- ใช้ในการตัดสินใจเกี่ยวกับการแบ่งโมดูลที่ซับซ้อนว่าต่อไปจะต้องทำหรือไม่
- ใช้ในการแนะนำตัวจัดการทรัพยากรเพื่อการใช้งานที่เหมาะสม
- ใช้ในการเปรียบเทียบและแลกเปลี่ยนการออกแบบระหว่างการพัฒนาซอฟต์แวร์และค่าบำรุงรักษา
- ใช้ในการให้ข้อเสนอแนะแก่ผู้จัดการซอฟต์แวร์เกี่ยวกับความก้าวหน้าและคุณภาพในช่วงต่างๆ ของวงจรชีวิตการพัฒนาซอฟต์แวร์
- ใช้ในการจัดสรรทรัพยากรการทดสอบสำหรับการทดสอบรหัส

ข้อเสียของซอฟต์แวร์เมตริก

- การประยุกต์ใช้ซอฟต์แวร์เมตริก ไม่ใช่เรื่องง่ายเสมอไป และในบางกรณีก็ยากและมีค่าใช้จ่ายสูง
- การตรวจสอบและให้เหตุผลของซอฟต์แวร์เมตริก ขึ้นอยู่กับข้อมูลในอดีต/เชิงประจักษ์ซึ่งความถูกต้องนั้นยากต่อการตรวจสอบ
- สิ่งเหล่านี้มีประโยชน์สำหรับการจัดการผลิตภัณฑ์ซอฟต์แวร์ แต่ไม่ใช่สำหรับการประเมินประสิทธิภาพของเจ้าหน้าที่ด้านเทคนิค
- คำจำกัดความและที่มาของซอฟต์แวร์เมตริก มักจะขึ้นอยู่กับการสมมติที่ไม่ได้มาตรฐาน และอาจขึ้นอยู่กับเครื่องมือที่มีอยู่และสภาพแวดล้อมการทำงาน
- แบบจำลองการทำนายส่วนใหญ่อาศัยการประมาณการของตัวแปรบางตัวซึ่งมักไม่ทราบแน่ชัด

เมตริกมุ่งเน้นที่ขนาด

(Size Oriented Metrics)

แอลโอซีเมตริก (LOC Metrics)

เป็นหนึ่งในตัวชี้วัดที่เร็วและง่ายสำหรับการคำนวณขนาดของโปรแกรมคอมพิวเตอร์ โดยทั่วไปจะใช้ในการคำนวณและเปรียบเทียบประสิทธิภาพการทำงานของโปรแกรมเมอร์ ตัวชี้วัดเหล่านี้ได้มาจากการทำให้การวัดคุณภาพและผลผลิตเป็นมาตรฐานโดยพิจารณาจากขนาดของผลิตภัณฑ์เป็นตัวชี้วัด

LOC (Line Of Code)

ประเด็นเกี่ยวกับการวัด LOC

ในการวัดตามขนาด LOC ถือเป็นค่านอร์มัลไลซ์เซชัน

- เป็นวิธีที่เก่ากว่าซึ่งพัฒนาขึ้นเมื่อโปรแกรม FORTRAN และ COBOL ได้รับความนิยมอย่างมาก
- ผลผลิตถูกกำหนดเป็น KLOC / ความพยายาม ซึ่งวัดความพยายามในคนต่อเดือน
- ตัวชี้วัดตามขนาดขึ้นอยู่กับภาษาการเขียนโปรแกรมที่ใช้
- เนื่องจากประสิทธิภาพการทำงานขึ้นอยู่กับ KLOC ดังนั้นโค้ดภาษาแอสเซมบลีจะมีประสิทธิผลมากขึ้น
- การวัด LOC ต้องมีระดับของรายละเอียดซึ่งอาจไม่สามารถทำได้ในทางปฏิบัติ
- ยิ่งแสดงภาษาโปรแกรมได้ชัดเจนเท่าไร ประสิทธิภาพการทำงานก็จะยิ่งต่ำลงเท่านั้น

ประเด็นเกี่ยวกับการวัด LOC(ต่อ)

- วิธีการวัด LOC ใช้ไม่ได้กับโครงการที่เกี่ยวข้องกับการเขียนโปรแกรมด้วยภาพ (GUI-based) ตามที่อธิบายไว้แล้ว Graphical User Interfaces (GUI) ใช้แบบฟอร์มโดยพื้นฐาน เมตริก LOC เมตริก ใช้ไม่ได้ในกรณีนี้
- กำหนดให้ทุกองค์กรต้องใช้วิธีการเดียวกันในการนับ LOC ที่เป็นเช่นนี้เพราะบางองค์กรใช้เฉพาะคำสั่งที่เรียกใช้งานได้ ความคิดเห็นที่เป็นประโยชน์บางอย่าง และบางองค์กรไม่ทำ จึงมีความจำเป็นที่จะต้องมีการกำหนดเป็นมาตรฐานไว้ใช้อ้างอิง
- เมตริกเหล่านี้ไม่เป็นที่ยอมรับในระดับสากล

การวัดขนาดซอฟต์แวร์จาก LOC/KLOC ยังมีเมตริกอื่นๆ ที่สามารถคำนวณได้เช่นกัน

- ข้อผิดพลาด/KLOC.
- $/ KLOC.
- ข้อบกพร่อง/KLOC
- หน้าเอกสาร/KLOC.
- ข้อผิดพลาด/PM.
- ผลผลิต = KLOC/PM (ความพยายามวัดเป็นบุคคล-เดือน)
- $/ หน้าเอกสาร.

ข้อดีของ LOC

วัดง่าย

ข้อเสียของ LOC

- มันถูกกำหนดไว้ในรหัส เช่น ไม่สามารถวัดขนาดของข้อกำหนดได้
- มีลักษณะเฉพาะของขนาดเดียว กล่าวคือ ความยาว ไม่คำนึงถึงการทำงานหรือความซับซ้อน
- การออกแบบซอฟต์แวร์ที่ไม่ดีอาจทำให้โค้ดมีจำนวนมากเกินไป
- มันขึ้นอยู่กับภาษา
- ผู้ใช้ไม่สามารถเข้าใจได้ง่าย

ซอฟต์แวร์เมตริกของ Halstead

(Size Oriented Metrics)

ซอฟต์แวร์เมตริกของ Halstead

ตามคำกล่าวของ Halstead "โปรแกรมคอมพิวเตอร์คือการใช้งานอัลกอริธึมที่ถือว่าเป็นชุดของโทเค็น ซึ่งสามารถจำแนกได้เป็นโอเปอเรเตอร์หรือตัวถูกดำเนินการ“

(โทเค็น (Token) คือ ชุดข้อมูลเสมือนที่ถูกเข้ารหัสโดยการสุ่มเพื่อใช้แทนข้อมูลที่ต้องการความปลอดภัยสูงและหลีกเลี่ยงการแลกเปลี่ยนข้อมูลนั้นโดยตรง เช่น เลขบัตรเครดิต กระบวนการแปลงข้อมูลกลายเป็นโทเค็นนี้เรียกว่า Tokenization แนวคิดด้าน Tokenization ถูกนำไปประยุกต์ใช้งานในหลายบริบท รวมไปถึงการสร้างนวัตกรรมทางการเงิน)

Token Count

ในตัวชี้วัดเหล่านี้ โปรแกรมคอมพิวเตอร์ถือเป็นชุดของโทเค็น ซึ่งอาจจัดประเภทเป็นโอเปอเรเตอร์หรือตัวถูกดำเนินการ เมตริกวิทยาศาสตร์ซอฟต์แวร์ทั้งหมดสามารถกำหนดได้ในแง่ของสัญลักษณ์พื้นฐานเหล่านี้ สัญลักษณ์เหล่านี้เรียกว่าเป็นโทเค็น

มาตรการพื้นฐานคือ

n1 = จำนวนตัวดำเนินการที่ไม่ซ้ำ

n2 = จำนวนตัวถูกดำเนินการเฉพาะ

N1 = จำนวนครั้งของโอเปอเรเตอร์ทั้งหมด

N2 = จำนวนการเกิดขึ้นทั้งหมดของตัวถูกดำเนินการ

ในแง่ของจำนวนโทเค็นที่ใช้ทั้งหมด ขนาดของโปรแกรมสามารถแสดงเป็น N = N1 + N2

เมตริก Halstead

ปริมาณโปรแกรม- Program Volume (V)

หน่วยในการวัดปริมาณโปรแกรมมีหน่วยเป็น “bit" ซึ่งเป็นขนาดที่แท้จริงของโปรแกรมที่อยู่ในรูปแบบการเข้ารหัสเลขฐานสองหรือรหัสไบนารี

$$V=N*\log_2 n$$

ระดับโปรแกรม- Program Level (L)

ค่าของ L อยู่ระหว่างศูนย์ถึงหนึ่ง โดย L=1 แสดงถึงโปรแกรมที่เขียนในระดับสูงสุดที่เป็นไปได้ (เช่น มีขนาดต่ำสุด)

$$L=V^*/V$$

เมตริก Halstead (ต่อ)

ความยากของโปรแกรม- Program Difficulty

ระดับความยากหรือความโน้มเอียงของข้อผิดพลาด (D) ของโปรแกรมเป็นสัดส่วนกับจำนวนตัวดำเนินการเฉพาะในโปรแกรม

$$D= (n1/2) * (N2/n2)$$

ความพยายามในการเขียนโปรแกรม- Programming Effort (E)

หน่วยวัดของ E คือ หน่วยเบื้องต้นในการปฏิบัติงาน

$$E=V/L=D*V$$

เมตริก Halstead (ต่อ)

ความยาวโปรแกรมโดยประมาณ- Estimated Program Length

ตามคำกล่าวของ Halstead สมมติฐานแรกของวิทยาศาสตร์ซอฟต์แวร์คือความยาวของโปรแกรมที่มีโครงสร้างดีนั้นเป็นเพียงฟังก์ชันของจำนวนตัวดำเนินการและตัวถูกดำเนินการที่ไม่ซ้ำกันเท่านั้น

N=N1+N2

และความยาวของโปรแกรมโดยประมาณแสดงด้วย N^

N^ = n1log2n1 + n2log2n2

มีการเผยแพร่นิพจน์ทางเลือกต่อไปนี้เพื่อประมาณความยาวของโปรแกรม:

NJ = log2 (n1!) + log2 (n2!)

NB = n1 * log2n2 + n2 * log2n1

NC = n1 * sqrt(n1) + n2 * sqrt(n2)

NS = (n * log2n) / 2

เมตริก Halstead (ต่อ)

ปริมาณขั้นต่ำที่เป็นไปได้-Potential Minimum Volume

ปริมาณขั้นต่ำที่เป็นไปได้ V* ถูกกำหนดให้เป็นปริมาณของโปรแกรมที่สั้นที่สุดซึ่งสามารถเขียนรหัสปัญหาได้

$$V^* = (2 + n2^*) * \log2\ (2 + n2^*)$$

ที่นี่ n2* คือจำนวนพารามิเตอร์อินพุตและเอาต์พุตที่ไม่ซ้ำกัน

เมตริก Halstead (ต่อ)

ขนาดของคำศัพท์-Size of Vocabulary (n)

ขนาดของคำศัพท์ของโปรแกรม ซึ่งประกอบด้วยจำนวนของโทเค็นที่ไม่ซ้ำกันที่ใช้ในการสร้างโปรแกรม ถูกกำหนดเป็น:

$n=n1+n2$

ที่ไหน

n=คำศัพท์ของโปรแกรม

n1=จำนวนตัวดำเนินการที่ไม่ซ้ำ

n2=จำนวนของตัวถูกดำเนินการเฉพาะ

เมตริก Halstead (ต่อ)

ระดับภาษา - แสดงระดับภาษาของโปรแกรมการนำอัลกอริธึมไปใช้ อัลกอริทึมเดียวกันนี้ต้องการความพยายามเพิ่มเติมหากเขียนด้วยภาษาโปรแกรมระดับต่ำ ตัวอย่างเช่น การเขียนโปรแกรมในภาษา Pascal ง่ายกว่าใน Assembler

$$L' = V / D / D$$

แลมบ์ดา = L * V* = L2 * V

เมตริก Halstead (ต่อ)

Language levels

Language	Language level λ	Variance σ
PL/1	1.53	0.92
ALGOL	1.21	0.74
FORTRAN	1.14	0.81
CDC Assembly	0.88	0.42
PASCAL	2.54	-
APL	2.42	-
C	0.857	0.445

เมตริก Halstead (ต่อ)

กฎการนับภาษาซี

- ความคิดเห็นจะไม่ได้รับการพิจารณา // หรือ /*…*/
- ไม่พิจารณาการประกาศตัวระบุและฟังก์ชัน
- ตัวแปรและค่าคงที่ทั้งหมดถือเป็นตัวถูกดำเนินการ
- ตัวแปรโกลบอลที่ใช้ในโมดูลต่างๆ ของโปรแกรมเดียวกันจะนับเป็นการเกิดขึ้นของตัวแปรเดียวกันหลายครั้ง
- ตัวแปรโลคัลที่มีชื่อเดียวกันในฟังก์ชันต่างๆ จะถูกนับเป็นตัวถูกดำเนินการที่ไม่ซ้ำกัน
- การเรียกใช้ฟังก์ชันถือเป็นตัวดำเนินการ
- คำสั่งวนซ้ำทั้งหมด เช่น ทำ {...} while ( ) ในขณะที่ ( ) {...} สำหรับ ( ) {...} คำสั่งควบคุมทั้งหมด เช่น if ( ) {...} if ( ) {...} อื่นๆ {...} เป็นต้น ถือเป็นตัวดำเนินการ
- ในสวิตช์สร้างการควบคุม ( ) {กรณี:...} สวิตช์และคำสั่งกรณีทั้งหมดถือเป็นโอเปอเรเตอร์

เมตริก Halstead (ต่อ)

- คำสงวนเช่น return, default, continue, break, sizeof เป็นต้น ถือเป็นโอเปอเรเตอร์
- วงเล็บ เครื่องหมายจุลภาค และเทอร์มิเนเตอร์ทั้งหมดถือเป็นตัวดำเนินการ
- GOTO ถูกนับเป็นโอเปอเรเตอร์ และเลเบลจะถูกนับเป็นตัวถูกดำเนินการ
- การเกิดขึ้นของเอกนารีและไบนารีของ "+" และ "-" ถูกจัดการแยกกัน ในทำนองเดียวกัน "*" (ตัวดำเนินการการคูณ) จะถูกแจกแยกกัน
- ในตัวแปรอาร์เรย์ เช่น "array-name [index]" "array-name" และ "index" ถือเป็นตัวถูกดำเนินการ และ [ ] ถือเป็นโอเปอเรเตอร์
- ในตัวแปรโครงสร้างเช่น "struct-name, member-name" หรือ "struct-name -> member-name" struct-name, member-name ถือเป็นตัวถูกดำเนินการและ '.', '->' จะถูกนำมาเป็น ผู้ประกอบการ ชื่อขององค์ประกอบสมาชิกบางชื่อในตัวแปรโครงสร้างที่แตกต่างกันจะถูกนับเป็นตัวถูกดำเนินการเฉพาะ
- คำสั่งแฮชทั้งหมดจะถูกละเว้น

สรุป

Metric	Meaning	Mathematical Representation
n	Vocabulary	n1 + n2
N	Size	N1 + N2
V	Volume	Length * Log2 Vocabulary
D	Difficulty	(n1/2) * (N1/n2)
E	Efforts	Difficulty * Volume
B	Errors	Volume / 3000
T	Testing time	Time = Efforts / S, where S=18 seconds.

เมตริก Halstead (ต่อ)-ตัวอย่างการคำนวณ

ตัวอย่าง: พิจารณาโปรแกรมการเรียงลำดับดังแสดงในรูป: แสดงรายการตัวดำเนินการและตัวถูกดำเนินการ และคำนวณค่าของการวัดทางวิทยาศาสตร์ซอฟต์แวร์ด้วย เช่น n, N, V, E, λ เป็นต้นวิธีแก้ไข:

รายการตัวดำเนินการและตัวถูกดำเนินการแสดงอยู่ในตาราง

เมตริก Halstead (ต่อ)-ตัวอย่างการคำนวณ

Operators	Occurrences	Operands	Occurrences
int	4	SORT	1
()	5	x	7
,	4	n	3
[]	7	i	8
if	2	j	7
<	2	save	3
;	11	im1	3
for	2	2	2
=	6	1	3
-	1	0	1
<=	2	-	-
++	2	-	-
return	2	-	-
{}	3	-	-
n1=14	N1=53	n2=10	N2=38

เมตริก Halstead (ต่อ)-ตัวอย่างการคำนวณ

ที่นี่ N1=53 และ N2=38 ความยาวของโปรแกรม N=N1+N2=53+38=91

Vocabulary of the program n=n1+n2=14+10=24

Volume V= N * log2N=91 x log2 24=417 บิต

estimate program length N ของโปรแกรม

= 14 log214+10 log2)10

= 14 * 3.81+10 * 3.32

= 53.34+33.2=86.45

พารามิเตอร์อินพุตและเอาต์พุตที่ไม่ซ้ำกันตามแนวคิดจะแสดงด้วย n2*

n2*=3 {x: array ถือจำนวนเต็มที่จะจัดเรียง ใช้เป็นทั้งอินพุตและเอาต์พุต}

{N: ขนาดของอาร์เรย์ที่จะจัดเรียง}

เมตริก Halstead (ต่อ)-ตัวอย่างการคำนวณ

The Potential Volume V*=5log25=11.6

ตั้งต้น L=V*/V

$$= \frac{11.6}{417} = 0.027$$

D=I/L

$$= \frac{1}{0.027} = 37.03$$

ประมาณการ Program Level

$$L^{\wedge} = \frac{2}{n1} X \frac{n2}{N2} = \frac{2}{14} X \frac{10}{38} = 0.038$$

ซึ่งเราอาจใช้สมการอื่น

V^=V x L^= 417 x 0.038=15.67

E^=V/L^=D^ x V

$$= \frac{417}{0.038} = 10973.68$$

เมตริก Halstead (ต่อ)-ตัวอย่างการคำนวณ

10974 เป็นค่า Effort หรือค่าความพยายามในการเขียนโปรแกรม

$$T = \frac{E}{\beta} = \frac{10974}{18} = 610 \text{ seconds} = 10 \text{ minutes}$$

นี่อาจเป็นเวลาที่เหมาะสมในการผลิตโปรแกรม ซึ่งง่ายมาก

เมตริก Halstead (ต่อ)-ตัวอย่างการคำนวณ

```
int sort (int x[ ], int n)

{
    int i, j, save, im1;
    /*This function sorts array x in ascending order */
    If (n< 2) return 1;
    for (i=2; i< =n; i++)
    {
        im1=i-1;
        for (j=1; j< =im1; j++)
            if (x[i] < x[j])
            {
                Save = x[i];
                x[i] = x[j];
                x[j] = save;
            }
    }
    return 0;
}
```

operators	occurrences	operands	occurrences
int	4	sort	1
()	5	x	7
,	4	n	3
[]	7	i	8
if	2	j	7
<	2	save	3
;	11	im1	3
for	2	2	2
=	6	1	3
-	1	0	1
<=	2	-	-
++	2	-	-
return	2	-	-
{}	3	-	-
n1=14	N1=53	n2=10	N2=38

เมตริก Halstead (ต่อ)-ตัวอย่างการคำนวณ

ดังนั้น,

N = 91

n = 24

V = 417.23 บิต

N^ = 86.51

n2* = 3 (x:อาร์เรย์ถือจำนวนเต็มที่จะจัดเรียง ใช้ได้ทั้งเป็นอินพุตและเอาต์พุต)

V* = 11.6

L = 0.027

D = 37.03

L^ = 0.038

T = 610 วินาที

เมตริก Halstead (ต่อ)-ตัวอย่างการคำนวณ

ภาษา C

```
main()
{
  int a, b, c, avg;
  scanf("%d %d %d", &a, &b, &c);
  avg = (a+b+c)/3;
  printf("avg = %d", avg);
}
```

The unique operators are: main , () , {} , int , scanf , & , = , + , / , printf , , , ;

The unique operands are: a , b , c , avg , "%d %d %d" , 3 , "avg = %d"

- $\eta_1 = 12, \eta_2 = 7, \eta = 19$
- $N_1 = 27, N_2 = 15, N = 42$
- Calculated Estimated Program Length: $\hat{N} = 12 \times log_2 12 + 7 \times log_2 7 = 62.67$
- Volume: $V = 42 \times log_2 19 = 178.4$
- Difficulty: $D = \frac{12}{2} \times \frac{15}{7} = 12.85$
- Effort: $E = 12.85 \times 178.4 = 2292.44$
- Time required to program: $T = \frac{2292.44}{18} = 127.357$ seconds
- Number of delivered bugs: $B = \frac{2292.44^{\frac{2}{3}}}{3000} = 0.05$

การวิเคราะห์แบบ Functional Point (FP)

(Functional Point (FP) Analysis)

การวิเคราะห์แบบ Functional Point (FP)

Allan J. Albrecht เริ่มพัฒนา function Point Analysis ในปี 1979 ที่ IBM และได้รับการแก้ไขเพิ่มเติมโดย International Function Point Users Group (IFPUG) FPA ใช้เพื่อประเมินโครงการซอฟต์แวร์ รวมถึงการทดสอบในแง่ของฟังก์ชันการทำงานหรือขนาดฟังก์ชันของผลิตภัณฑ์ซอฟต์แวร์ อย่างไรก็ตาม อาจใช้การวิเคราะห์จุดทำงานสำหรับการประมาณการทดสอบของผลิตภัณฑ์ ขนาดฟังก์ชันของผลิตภัณฑ์วัดจากจุดฟังก์ชัน ซึ่งเป็นมาตรฐานในการวัดเพื่อวัดการใช้งานซอฟต์แวร์

การวิเคราะห์แบบ Functional Point (FP)

วัตถุประสงค์ของ FPA

วัตถุประสงค์พื้นฐานและเบื้องต้นของการวิเคราะห์จุดทำงานคือการวัดและจัดเตรียมขนาดฟังก์ชันของแอพพลิเคชั่นซอฟต์แวร์ให้กับลูกค้า ลูกค้า และผู้มีส่วนได้ส่วนเสียตามคำขอของพวกเขา นอกจากนี้ยังใช้เพื่อวัดการพัฒนาโครงการซอฟต์แวร์พร้อมกับการบำรุงรักษาอย่างต่อเนื่องตลอดโครงการ โดยไม่คำนึงถึงเครื่องมือและเทคโนโลยี

การวิเคราะห์แบบ Functional Point (FP)

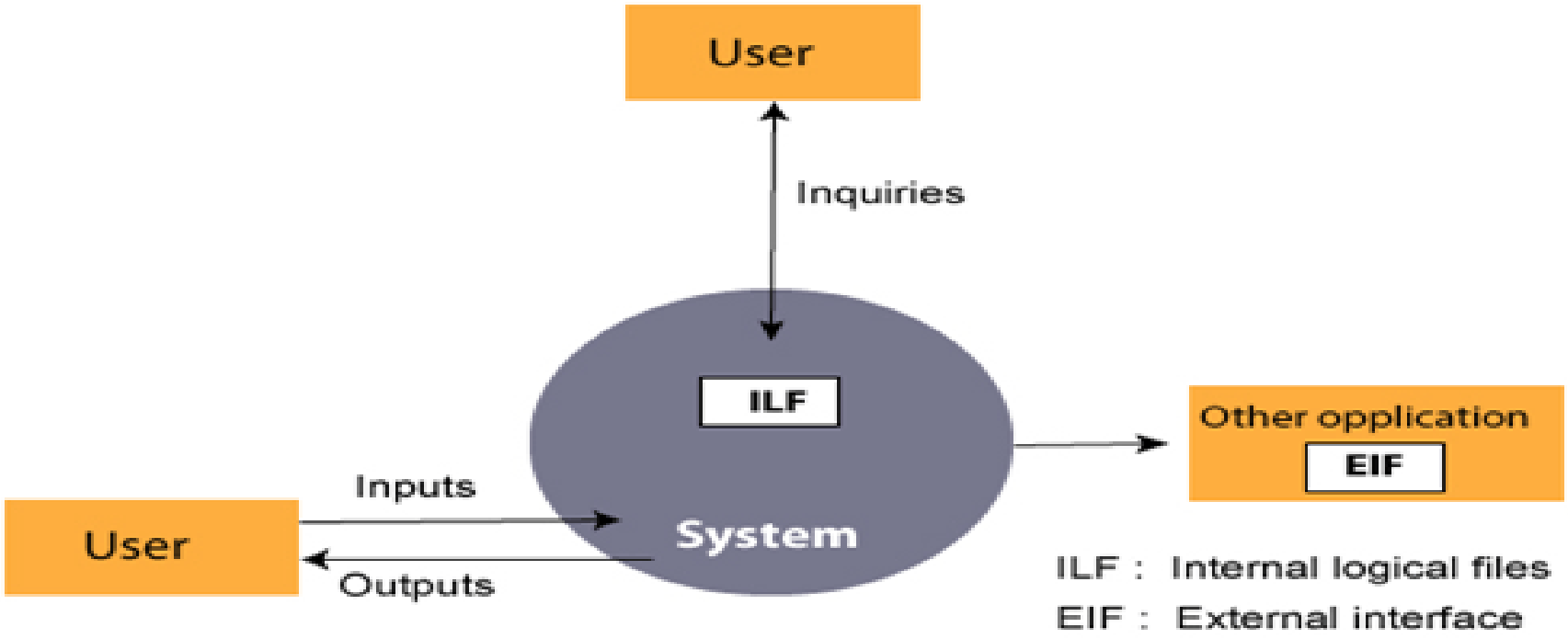
ประเด็นที่พิจารณาเกี่ยวกับ FPs

1. ค้นหา FP ของแอปพลิเคชันโดยการนับจำนวนและประเภทของฟังก์ชันที่ใช้ในแอปพลิเคชัน ฟังก์ชันต่าง ๆ ที่ใช้ในแอปพลิเคชันสามารถใส่ได้ 5 ประเภทดังแสดงในตาราง:

Measurements Parameters	Examples
1.Number of External Inputs(EI)	Input screen and tables
2. Number of External Output (EO)	Output screens and reports
3. Number of external inquiries (EQ)	Prompts and interrupts.
4. Number of internal files (ILF)	Databases and directories
5. Number of external interfaces (EIF)	Shared databases and shared routines.

การวิเคราะห์แบบ Functional Point (FP)

ประเด็นที่พิจารณาเกี่ยวกับ FPs



FPAs Functional Units System

แหล่งที่มา https://www.javatpoint.com/

การวิเคราะห์แบบ Functional Point (FP)

ประเด็นที่พิจารณาเกี่ยวกับ FPs

2. FP แสดงถึงความซับซ้อนของระบบซอฟต์แวร์ จึงสามารถใช้เพื่ออธิบายเวลาของโครงการและความต้องการกำลังคนได้
3. ความพยายามที่จำเป็นในการพัฒนาโครงการขึ้นอยู่กับสิ่งที่ซอฟต์แวร์ทำ
4. FP เป็นภาษาโปรแกรมอิสระ
5. วิธี FP ใช้สำหรับระบบประมวลผลข้อมูล ระบบธุรกิจ เช่น ระบบสารสนเทศ
6. พารามิเตอร์ทั้งห้าที่กล่าวถึงข้างต้นเรียกอีกอย่างว่าคุณลักษณะของโดเมนข้อมูล

การวิเคราะห์แบบ Functional Point (FP)

ประเด็นที่พิจารณาเกี่ยวกับ FPs

7. พารามิเตอร์ทั้งหมดที่กล่าวถึงข้างต้นได้รับการกำหนดน้ำหนักบางตัวที่ได้รับการพิจารณาจากการทดลองและแสดงในตาราง

Weights of 5-FP Attributes

Measurement Parameter	Low	Average	High
1. Number of external inputs (EI)	7	10	15
2. Number of external outputs (EO)	5	7	10
3. Number of external inquiries (EQ)	3	4	6
4. Number of internal files (ILF)	4	5	7
5. Number of external interfaces (EIF)	3	4	6

การวิเคราะห์แบบ Functional Point (FP)

ประเด็นที่พิจารณาเกี่ยวกับ FPs

Function Point Computations (IFPUG)

PARAMETER	simple	medium	complex	
Ext. inputs *EI* X	☐3	+ ☐4	+ ...6 =	☐
Ext. outputs *EO* X	☐4	+ ☐5	+ ...7 =	☐
Ext. inquiries *EIN* X	☐3	+ ☐4	+ ... 6 =	☐
Int. logical files *ILF* X	...7	+ ☐10	+ ...15 =	☐
Ext. logical files *ELF* X	...5	+ ☐7	+ ...10 =	☐
			Count Total	☐

การวิเคราะห์แบบ Functional Point (FP)

ประเด็นที่พิจารณาเกี่ยวกับ FPs

จะได้ค่า Unadjusted Function Point

		Simple		Medium		Complex		Sub-	Total
		count	factor	count	factor	count	factor	totals	
Ext. inputs		1	3	1	4	1	6	13	
comments:		Name		Ready move		Qualities			
Ext. outputs		0	4	0	5	0	7	0	
Ext. inquiries		0	3	0	4	0	6	0	25
Int. logical files		1	7	0	10	0	15	7	
comments:		Data about the user's character							
Ext. interface files		1	5	0	7	0	10	5	
comments:		Data about the user's character							

การวิเคราะห์แบบ Functional Point (FP)

ประเด็นที่พิจารณาเกี่ยวกับ FPs

คำนวณค่าคุณลักษณะซอฟต์แวร์

Total General Characteristics

ในขั้นตอนต่อไปท่านจะต้องคำนวณค่าคุณลักษณะทั่วไปของซอฟต์แวร์ของท่าน ซึ่งได้มาจากปัจจัยที่มีการปรับเปลี่ยนได้ของซอฟต์แวร์ของท่าน

โดยปัจจัยที่มีการปรับเปลี่ยนได้นี้ IEEE ได้กำหนดเอาไว้ 14 ข้อด้วยกันคือ

รายการ	คะแนนที่ให้
1. ซอฟต์แวร์ของท่านต้องการสำรองข้อมูลและกู้ข้อมูล Requires backup/recovery?	.....
2. ซอฟต์แวร์ของท่านต้องการมีการสื่อสารข้อมูลระหว่างกัน Data communications required?	.....
3. ความต้องการมีฟังก์ชันการประมวลผลแบบกระจาย Distributed processing functions?	.....
4. ความต้องการมีการวิเคราะห์ศักยภาพในการทำงาน Performance critical?	.....
5. ต้องการให้รองรับทำงานในสภาวะที่มีการใช้งานสูง Run on existing heavily utilized environment?	.....
6. ต้องการให้มีการรับส่งข้อมูลแบบออนไลน์ Requires on-line data entry?	.....
7. สามารถเปิดหน้าจออินพุตได้ทีละหลาย ๆ หน้า Multiple screens for input?	.....
8. มีการปรับปรุงข้อมูลในแบบออนไลน์ Master fields updated on-line?	.....
9. อินพุต เอาร์พุต การร้องขอเกี่ยวไฟล์ไฟล์มีความซับซ้อน Inputs, outputs, inquiries of files complex?	.....
10. กระบวนการภายในมรความซับซ้อน Internal processing complex?	.....
11. การเขียนโค้ดถูกออกแบบมาให้สามารถนำกลับมาใช้งานได้อีก Code designed for re-use?	.....
12. ซอฟต์แวร์สามารถที่จะถูกติดตั้งและแก้ไขคุณลักษณะต่าง ๆ ได้ Conversion and installation included?	.....
13. สามารถนำไปใช้งานในหลาย ๆ หน่วยงาน Multiple installation in different orgs.?	.....
14. รองรับการเปลี่ยนแปลงในปัจจัยเกื้อหนุนและง่ายในการใช้งาน Must facilitate change & ease-of-use by user?	.....
ผลรวมค่าคุณลักษณะทั่วไปของซอฟต์แวร์ total general characteristics	.....

การวิเคราะห์แบบ Functional Point (FP)

ประเด็นที่พิจารณาเกี่ยวกับ FPs

FP(function point) = [unadjusted function points] x [0.65 + 0.01 x (total general characteristics)]

จากการคำนวณก่อนหน้านี้ที่ได้มา

unadjusted function points = 41

total general characteristics = 24 - 41

FP(function point) = [41] x [0.65 + (0.01 x (24 ถึง 41))] = 36 ถึง 43 หน่วยเป็น FP

ตัวชี้วัดโครงสร้างข้อมูล-Data Structure Metrics

โดยพื้นฐานแล้วความจำเป็นในการพัฒนาซอฟต์แวร์และกิจกรรมอื่นๆ คือการประมวลผลข้อมูล ข้อมูลบางส่วนถูกป้อนเข้าสู่ระบบ โปรแกรม หรือโมดูล ข้อมูลบางอย่างอาจถูกใช้ภายใน และข้อมูลบางส่วนเป็นผลลัพธ์จากระบบ โปรแกรม หรือโมดูล

นั่นเป็นเหตุผลที่ชุดเมตริกที่สำคัญซึ่งจับปริมาณข้อมูลเข้า ประมวลผลในซอฟต์แวร์ รูปแบบผลลัพธ์ การนับโครงสร้างข้อมูลนี้เรียกว่า Data Structured Metrics ในความเข้มข้นเหล่านี้อยู่ที่ตัวแปร (และให้ค่าคงที่) ภายในแต่ละโมดูล & ละเว้นการพึ่งพาอินพุต-เอาท์พุต

ตัวชี้วัดโครงสร้างข้อมูล-Data Structure Metrics (ต่อ)

มีเมตริกโครงสร้างข้อมูลบางตัวในการคำนวณความพยายามและเวลาที่จำเป็นในการดำเนินการโครงการให้เสร็จสิ้น มีตัวชี้วัดคือ:

1. ปริมาณข้อมูล
2. การใช้ข้อมูลภายในโมดูล-
3. จุดอ่อนของโปรแกรม
4. การแบ่งปันข้อมูลระหว่างโมดูล

ตัวชี้วัดโครงสร้างข้อมูล-Data Structure Metrics (ต่อ)

1. ปริมาณข้อมูล: ในการวัดปริมาณข้อมูล มีตัวชี้วัดอื่นอีกมากมาย ซึ่งได้แก่

จำนวนตัวแปร (VARS): ในเมตริกนี้ จะนับจำนวนตัวแปรที่ใช้ในโปรแกรม

จำนวนตัวถูกดำเนินการ ($\eta 2$): ในเมตริกนี้ จะนับจำนวนตัวถูกดำเนินการที่ใช้ในโปรแกรม

$\eta 2$ = VARS + ค่าคงที่ + ป้ายกำกับ

จำนวนการเกิดของตัวแปรทั้งหมด (N2): ในเมตริกนี้ จำนวนรวมของการเกิดตัวแปรจะถูกคำนวณ

ตัวชี้วัดโครงสร้างข้อมูล-Data Structure Metrics (ต่อ)

2. การใช้ข้อมูลภายในโมดูล: การวัดเมตริกนี้ ตัวเลขเฉลี่ยของตัวแปรสดจะถูกคำนวณ ตัวแปรใช้งานได้ตั้งแต่แรกจนถึงการอ้างอิงสุดท้ายภายในโพรซีเดอร์

$$\text{Average no of Live variables (LV)} = \frac{\text{Sum of count live variables}}{\text{Sum of count of executable statements}}$$

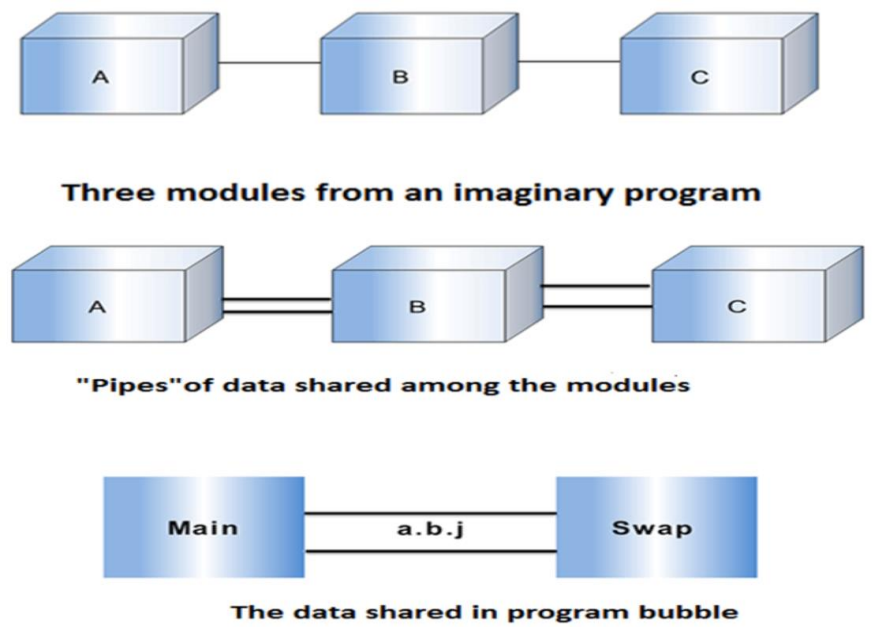
ตัวชี้วัดโครงสร้างข้อมูล-Data Structure Metrics (ต่อ)

3. จุดอ่อนของโปรแกรม: จุดอ่อนของโปรแกรมขึ้นอยู่กับจุดอ่อนของโมดูล หากโมดูลอ่อนแอ (เหนียวน้อยกว่า) จะเป็นการเพิ่มตัวชี้วัดความพยายามและเวลาที่จำเป็นในการดำเนินการโครงการให้เสร็จสิ้น

$$\text{Average life of variables } (\gamma) = \frac{\text{Sum of count live variables}}{\text{Sum of count of executable statements}}$$

ตัวชี้วัดโครงสร้างข้อมูล-Data Structure Metrics (ต่อ)

4. การแบ่งปันข้อมูลระหว่างโมดูล: เมื่อการแบ่งปันข้อมูลระหว่างโมดูลเพิ่มขึ้น (การมีเพศสัมพันธ์ที่สูงขึ้น) ไม่มีพารามิเตอร์ที่ส่งผ่านระหว่างโมดูลเพิ่มขึ้นด้วย ด้วยเหตุนี้จึงต้องใช้ความพยายามและเวลามากขึ้นในการทำโครงการให้เสร็จ ดังนั้นการแบ่งปันข้อมูลระหว่างโมดูลจึงเป็นตัวชี้วัดที่สำคัญในการคำนวณความพยายามและเวลา



แหล่งที่มา https://www.javatpoint.com/

ตัวชี้วัดการไหลของข้อมูล (Information Flow Metrics)

ชุดเมตริกอื่นๆ ที่เราจะพิจารณาเรียกว่า Information Flow Metrics พื้นฐานของการวัดการไหลของข้อมูลพบได้ตามแนวคิดต่อไปนี้ ระบบที่ง่ายที่สุดประกอบด้วยส่วนประกอบ และมันเป็นงานที่ส่วนประกอบเหล่านี้ทำและวิธีการประกอบเข้าด้วยกันเพื่อระบุความซับซ้อนของระบบ ต่อไปนี้เป็นข้อกำหนดการทำงานที่ใช้ในโฟลว์ข้อมูล:

- **Component**: องค์ประกอบใดๆ ที่ระบุโดยการแยกระบบ (ซอฟต์แวร์) ออกเป็นส่วนๆ ขององค์ประกอบ
- **Cohesion**: ระดับที่ส่วนประกอบร่วมทำงานในฟังก์ชันเดียวกัน
- **Coupling**: คำที่ใช้อธิบายระดับความเชื่อมโยงระหว่างส่วนประกอบหนึ่งกับส่วนประกอบอื่นๆ ในระบบเดียวกัน

ตัวชี้วัดการไหลของข้อมูล (Information Flow Metrics) (ต่อ)

ตัวชี้วัดการไหลของข้อมูลจัดการกับความซับซ้อนประเภทนี้โดยสังเกตการไหลของข้อมูลระหว่างส่วนประกอบหรือโมดูลของระบบ เมตริกนี้มอบให้โดย Henry และ Kafura ดังนั้นจึงเป็นที่รู้จักในชื่อ Henry and Kafura's Metric

ตัวชี้วัดนี้ขึ้นอยู่กับการวัดการไหลของข้อมูลระหว่างโมดูลระบบ มีความอ่อนไหวต่อความซับซ้อนเนื่องจากการเชื่อมต่อระหว่างส่วนประกอบของระบบ มาตรการนี้รวมถึงความซับซ้อนของโมดูลซอฟต์แวร์ที่กำหนดเป็นผลรวมของความซับซ้อนของขั้นตอนที่รวมอยู่ในโมดูล กระบวนการก่อให้เกิดความซับซ้อนเนื่องจากสองปัจจัยต่อไปนี้

- ความซับซ้อนของรหัสขั้นตอน
- ความซับซ้อนเนื่องจากการเชื่อมต่อกับสภาพแวดล้อมของขั้นตอน ผลกระทบของปัจจัยแรกได้รับการรวมผ่านมาตรการ LOC (บรรทัดของรหัส) สำหรับการหาปริมาณของปัจจัยที่สอง Henry และ Kafura ได้กำหนดคำศัพท์สองคำคือ FAN-IN และ FAN-OUT

Procedure Complexity = Length * (FAN-IN * FANOUT)**2

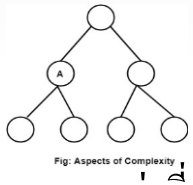


Fig: Aspects of Complexity

แหล่งที่มา https://www.javatpoint.com/

Cyclomatic Complexity

Cyclomatic Complexity-ความซับซ้อนของวัฏจักร เป็นตัวชี้วัดซอฟต์แวร์ที่ใช้ในการวัดความซับซ้อนของโปรแกรม Thomas J. McCabe พัฒนาตัวชี้วัดนี้ในปี 1976 McCabe ตีความโปรแกรมคอมพิวเตอร์เป็นชุดของกราฟกำกับที่เชื่อมโยงอย่างแน่นหนา โหนดเป็นตัวแทนของส่วนต่างๆ ของซอร์สโค้ดที่ไม่มีสาขา และส่วนโค้งแสดงถึงการถ่ายโอนโฟลว์การควบคุมที่เป็นไปได้ระหว่างการทำงานของโปรแกรม มีการใช้แนวคิดของกราฟโปรแกรมสำหรับการวัดนี้ และใช้เพื่อวัดและควบคุมจำนวนเส้นทางผ่านโปรแกรม ความซับซ้อนของโปรแกรมคอมพิวเตอร์สามารถสัมพันธ์กับความซับซ้อนเชิงทอพอโลยีของกราฟได้

วิธีการคำนวณความซับซ้อนของวัฏจักร-Cyclomatic Complexity

McCabe เสนอเลขวัฏจักร V (G) ของทฤษฎีกราฟเป็นตัวบ่งชี้ความซับซ้อนของซอฟต์แวร์ ตัวเลขไซโคลมาติกเท่ากับจำนวนเส้นทางอิสระเชิงเส้นผ่านโปรแกรมในการแสดงกราฟ สำหรับกราฟควบคุมโปรแกรม G หมายเลขไซโคลแมติก V (G) ถูกกำหนดเป็น:

$$V(G) = E - N + 2 * P$$

E = จำนวนขอบในกราฟ G

N = จำนวนโหนดในกราฟ

GP = จำนวนองค์ประกอบที่เชื่อมต่อในกราฟ G

ตัวอย่าง

คุณสมบัติของความซับซ้อนของวัฏจักร:

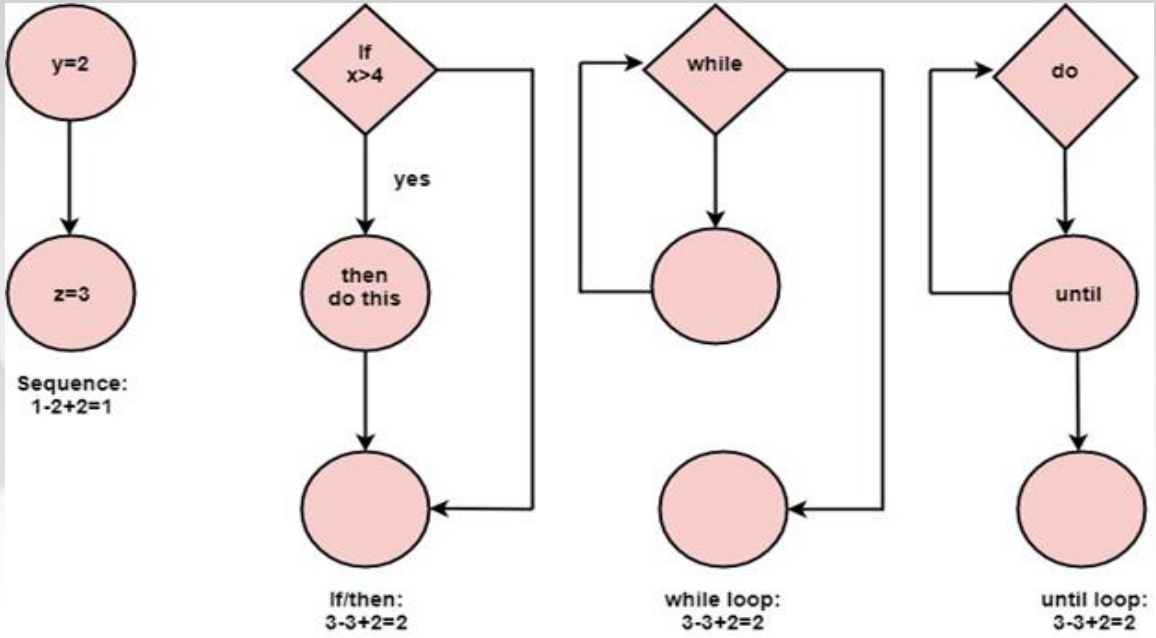
ต่อไปนี้เป็นคุณสมบัติของความซับซ้อนของวัฏจักร:

V (G) คือจำนวนเส้นทางอิสระสูงสุดในกราฟ

วี (ช) >=1

G จะมีหนึ่งเส้นทางถ้า V (G) = 1

ลดความซับซ้อนให้เหลือ 10



แหล่งที่มา https://www.javatpoint.com/

Case Tools สำหรับ Software Metrics

CASE TOOL (เครื่องมือช่วยของวิศวกรรมซอฟต์แวร์) มีอยู่มากมายสำหรับซอฟต์แวร์การวัด เป็นทั้งโอเพ่นซอร์สหรือเป็นเครื่องมือที่ต้องชำระเงิน ตัวอย่างมีดังนี้

- เครื่องมือ Analyst4j ใช้แพลตฟอร์ม Eclipse และพร้อมใช้งานเป็น Rich Client Application แบบสแตนด์อะโลนหรือเป็นปลั๊กอิน Eclipse IDE มีคุณลักษณะการค้นหา ตัวชี้วัด การวิเคราะห์คุณภาพ และการสร้างรายงานสำหรับโปรแกรม Java
- CCCC เป็นเครื่องมือบรรทัดคำสั่งโอเพ่นซอร์ส มันจะวิเคราะห์บรรทัด C++ และ Java และสร้างรายงานเกี่ยวกับตัวชี้วัดต่างๆ รวมถึง Lines of Code และตัวชี้วัดที่เสนอโดย Chidamber & Kemerer และ Henry & Kafura
- Chidamber & Kemerer Java Metrics เป็นเครื่องมือบรรทัดคำสั่งโอเพ่นซอร์ส จะคำนวณเมตริกเชิงวัตถุ C&K โดยการประมวลผลโค้ดไบต์ของ Java ที่คอมไพล์แล้ว

Case Tools สำหรับ Software Metrics (ต่อ)

- Dependency Finder เป็นโอเพ่นซอร์ส เป็นชุดเครื่องมือสำหรับวิเคราะห์โค้ด Java ที่คอมไพล์แล้ว แก่นของมันคือแอปพลิเคชั่นวิเคราะห์การพึ่งพาที่แยกกราฟการพึ่งพาและขุดข้อมูลเหล่านี้เพื่อรับข้อมูลที่เป็นประโยชน์ แอปพลิเคชันนี้มาพร้อมกับเครื่องมือบรรทัดคำสั่ง แอปพลิเคชันที่ใช้ Swing และเว็บแอปพลิเคชัน
- ปลั๊กอิน Eclipse Metrics 1.3.6 โดย Frank Sauer เป็นปลั๊กอินการคำนวณเมตริกโอเพนซอร์สและตัววิเคราะห์การพึ่งพาสำหรับ Eclipse IDE มันวัดตัวชี้วัดต่าง ๆ และตรวจจับรอบในการขึ้นต่อกันของแพ็คเกจและประเภท
- Eclipse Metrics Plug-in 3.4 โดย Lance Walton เป็นโอเพ่นซอร์ส จะคำนวณเมตริกต่างๆ ระหว่างรอบการสร้างและเตือน "การละเมิดช่วง" ของเมตริกผ่านมุมมองปัญหา
- OOMeter เป็นเครื่องมือเมตริกซอฟต์แวร์ทดลองที่พัฒนาโดย Alghamdi ยอมรับซอร์สโค้ด Java/C# และโมเดล UML ใน XMI และคำนวณเมตริกต่างๆ
- Semmle เป็นปลั๊กอิน Eclipse มันมี SQL เช่นภาษาการสืบค้นสำหรับโค้ดเชิงวัตถุ ซึ่งช่วยให้สามารถค้นหาจุดบกพร่อง วัดตัววัดรหัส ฯลฯ