เลขที่ 1214
วันที่ 6 ก.ค. 60
เวลา 6-32

เลขที่ 1195
วันที่ 5/7/60
เวลา 13.54น.

กรค. 555
เลขที่
วันที่ 4 ก.ค. 2560
เวลา 13.23 น.

รายงานสรุปการฝึกอบรม/สัมมนาภายนอก ประจำปี 2560

ฝทท 22/10 ลงวันที่ 3 กรกฎาคม 2560

เรียน รผบ. ผ่าน ผชก (นายสุชินฯ) ผอ.ฝทบ. รผก.(CIO) ผชก.(4) ผอ.ฝทท. ผอ.กรค. หน.พค.

ภพร. 1204
วันที่ - 6 ก.ค. 2560 9.50น.

ฝทท. (ภายใน) 1068
เลขที่
วันที่ 4 ก.ค. 2560
เวลา 14.24 น.

ส่วนที่ 1 ข้อมูลทั่วไป (สำหรับผู้เข้ารับการฝึกอบรม)

1. ข้าพเจ้า นางสาวมนัสนันท์ เพียรกิจรุ่งเรือง
ตำแหน่ง พนักงานวิเคราะห์ระบบ 5 สังกัด (แผนก/กอง/ฝ่าย) พค./กรค./ฝทท.
ได้รับอนุมัติให้เข้ารับการฝึกอบรม/สัมมนา หลักสูตร Managing Non-Functional Requirements and Business Goals จัดโดย สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ (สวทช) ฝทบ.
ระหว่างวันที่ 19-21 มิถุนายน 2560 สถานที่จัด อาคาร Software Park แจ้งวัฒนะ เลขที่ 3361
ค่าลงทะเบียนอบรม/สัมมนา ☑ เสียค่าใช้จ่าย 12,840 บาท ☐ ไม่เสียค่าใช้จ่าย วันที่ 06 ก.ค. 256
เวลา 09:45

รผบ. เลขที่ 1871
วันที่ 7/7/60 เวลา 11:50น.

2. ข้าพเจ้าขอรายงานสรุปการฝึกอบรม/สัมมนา ดังนี้

2.1 สรุปรายละเอียดเนื้อหาของหลักสูตรของ Managing Non-Functional Requirements and Business Goals มีรายละเอียดได้ดังนี้

1. ภาพรวมความต้องการ

1.1 วัตถุประสงค์การบริหารความต้องการ

การบริหารความต้องการเป็นส่วนสำคัญในการสนับสนุนการพัฒนาซอฟต์แวร์ โดยผู้เก็บความต้องการต้องทำให้ทุกฝ่าย (Stakeholder) เข้าใจมิติของระบบ ที่ประกอบด้วย (1) ความสามารถ หรือฟังก์ชั่น (2) กลไกการทำงานภายใน และ (3) คุณภาพ ที่มีความสัมพันธ์ในการพัฒนาระบบและให้ครอบคลุมความต้องการ ซึ่งมีกระบวนการทำงานดังนี้

1) รวบรวมปัญหาและความต้องการ
2) วิเคราะห์ปัญหาและความต้องการ
3) กำหนดระบบที่จะพัฒนาหรือปรับปรุง
4) จัดการกับขอบเขตของงาน
5) ปรับปรุงข้อกำหนดของระบบ
6) สร้างระบบที่ถูกต้องตรงตามความต้องการ

เรียน ผอ.ฝทบ.
- โปรดดำเนินการต่อไป
- เห็นควรปรับปรุงแบบฟอร์มรายงานใหม่ โดยตัดรายการภายนอกหน่วยงานออก
7/7/60

(นายฤทธิกา สุภารัตน์)

1.2 ความต้องการคืออะไร

แนวทางการพัฒนาระบบแบบ Object-Orientation ซึ่งอยู่บนแนวคิด Use Case Driven หรือ Function-Driven จะเรียกความต้องการหลักว่า Use Case ซึ่งก็คือฟังก์ชั่น และเรียกความต้องการทางคุณภาพว่า Non-Functional Requirement หรือ Special Requirement หรือ Supplimentary Requirement แต่แนวทางพัฒนาระบบแบบ Agile เรียกความต้องการว่า Use case (ความต้องการ) หรือ User Story (การอธิบายความต้องการ) ซึ่งความต้องการ มี 2 รูปแบบ ดังนี้

1) แบบเรียบง่าย เป็นการอธิบายสั้นๆ ใช้ภาษาที่สั้นกระชับ เช่น มีปริมาณไม่เกิน 1-3 บรรทัด ต่อ 1 ความต้องการ หรือการพูดบรรยายโดยใช้เวลาอธิบาย 1 ความต้องการสั้นๆ เช่น ไม่เกิน 1-3 นาทีต่อ 1 ความต้องการ (User Story)

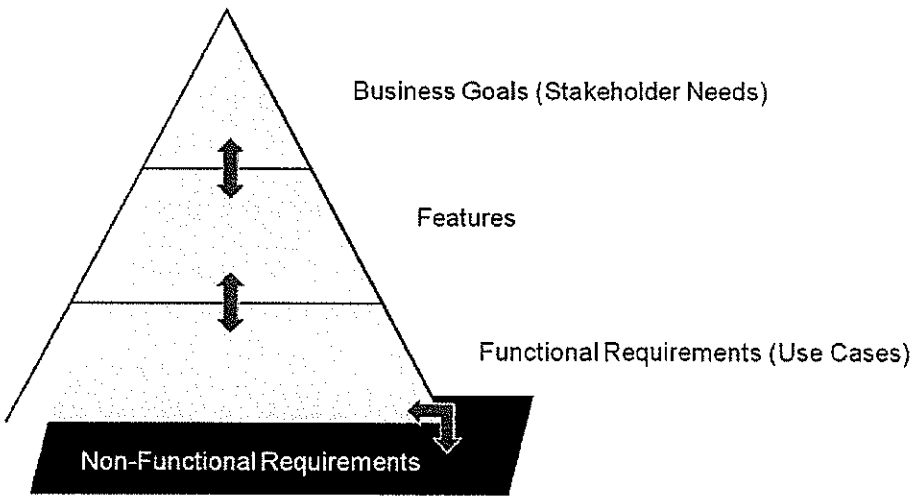
2) แบบละเอียด เป็นการอธิบายรายละเอียดแบบละเอียดๆ ซึ่งมีประเด็นย่อยๆ เช่น การอธิบาย Functional Requirement หรือ Use Case โดยแบ่งเป็นหัวข้อ ชื่อความต้องการ, รหัสความต้องการ, Brief Description, Pre-Condition, Post-Condition, Basic Flow, Alternative Flows, Special Requirements เป็นต้น (Use Case Specification) และ การอธิบาย Non-Functional Requirement แบ่งเป็นหัวข้อ Stimulus, Source of stimulus, Artifact, Environment, Response, Response measure (Quality Scenario ใน Non-Functional Requirement)

1.3 ประเภทความต้องการและความสัมพันธ์

ความต้องการของระบบ มีหลายรูปแบบ ไม่ว่าจะเป็น Functional Requirement และ Non-Function Requirement ตามหัวข้อ 1.3.1 และมีอีกรูปแบบหนึ่ง คือ Functionality กับ Non-Functionality ตามหัวข้อ 1.3.2

1.3.1 Functional Requirement และ Non-Function Requirement

ความต้องการ โดยทั่วไปแบ่งออกเป็น 2 ประเภทได้แก่ Functional Requirement และ Non-Function Requirement แต่ยังมีความต้องการที่สำคัญ และมีผลกระทบต่อการพัฒนาระบบ คือ Business Goals และ Feature โดยมีรายละเอียดดังนี้



1) Functional Requirement ในทาง Object-Orientation เรียกว่า Use Case คือ ฟังก์ชั่นของระบบที่มีความสัมพันธ์กับผู้ใช้ หรือ ฟังก์ชั่นของระบบที่ผู้ใช้เข้ามาเรียกใช้ได้ ซึ่งฟังก์ชั่นนี้สนับสนุนและสนองตอบการทำงานของผู้ใช้โดยตรง หรือ ฟังก์ชั่นระบบที่เอื้อประโยชน์ต่อผู้ใช้โดยตรง

2) Non-Functional Requirement หรือ NFR คือ ความต้องการที่เกี่ยวกับฟังก์ชั่นของระบบที่ไม่ได้เอื้อประโยชน์ต่อผู้ใช้โดยตรง เปรียบเสมือนคุณสมบัติทางคุณภาพทางกายภาพของสถาปัตยกรรมและระบบ ซึ่งตัวอย่างของ Non-Functional Requirement ได้แก่

- Availability คือ ความพร้อมในการให้บริการ
- Reliability คือ ความน่าเชื่อถือ
- Performance คือประสิทธิภาพในการประมวลผลและการใช้ทรัพยากร
- Scalability คือ ความสามารถด้านการรองรับการประมวลผลที่มากขึ้น
- Usability คือ ความสามารถด้านการสร้างคุณค่า คุณประโยชน์ และความง่ายในการใช้งาน
- Testability คือ ความสามารถในการทดสอบได้
- Modifiability คือ ความสามารถในการปรับปรุงแก้ไขได้
- Interoperability คือ ความสามารถในการทำงานร่วมกันได้
- Security คือ ความปลอดภัย

3) Business Goals หรือ Business Requirement/ Business Need, Stakeholder Need คือ ความต้องการเชิงธุรกิจ เป็นความต้องการที่กำหนดโดย Stakeholder ซึ่งเป็นความต้องการในระดับแรกในการเริ่มต้นโครงการ เช่น พัฒนาเสร็จรวดเร็วเพื่อทันต่อการใช้งานหรือขาย, ช่วยเพิ่มกำไรสุทธิได้ 10%, มีความสามารถในการแข่งขันกับคู่แข่งได้, สร้างการรับรู้และเข้าใจในตัวสินค้า (ซอฟต์แวร์) ได้เร็ว

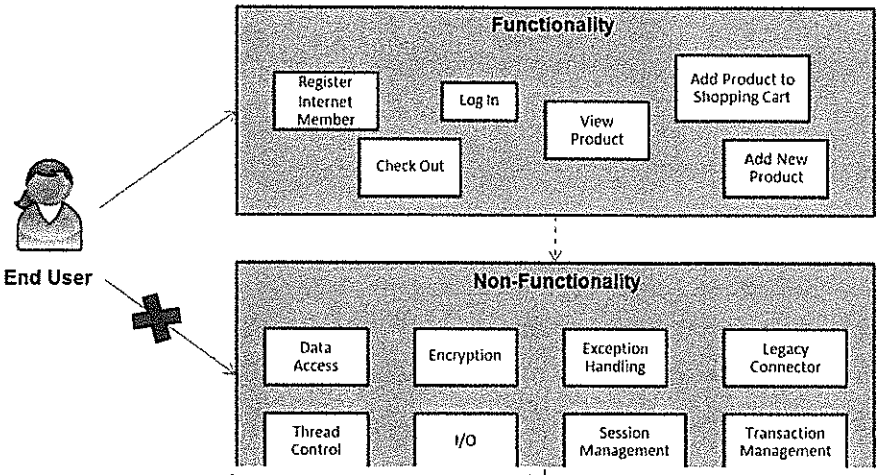
4) Feature คือ ความสามารถของระบบ หรือ คุณลักษณะเฉพาะของระบบที่ถูกพัฒนาขึ้น เพื่อให้ผู้ใช้งานให้ความสนใจ

ทั้งนี้แต่ละความต้องการและแต่ละประเภทควรเชื่อมโยง (trace) หากันได้ เพื่อให้ทราบว่ามีความเกี่ยวข้อง และสามารถเชื่อมโยงความต้องการไปยัง Software Artifact อื่นๆ ได้อีก เช่น Design, Test Case, User Manual, Screen, Source Code เป็นต้น

1.3.2 Functionality และ Non-Functionality

1) Functionality คือ การทำงานของระบบที่ไคลเอ็นต์ (ผู้ใช้ / อุปกรณ์/ ระบบภายนอก) สามารถมีปฏิสัมพันธ์ด้วย หรือเรียกใช้ได้ตรงๆ เช่น ผู้ใช้สามารถเรียกใช้ฟังก์ชั่นต่างๆ ได้โดยตรง เช่น Login, Check Out ส่วนกรณีไคลเอ็นต์เป็นระบบภายนอก จะเป็นการเรียกผ่านเซอร์วิสของระบบ

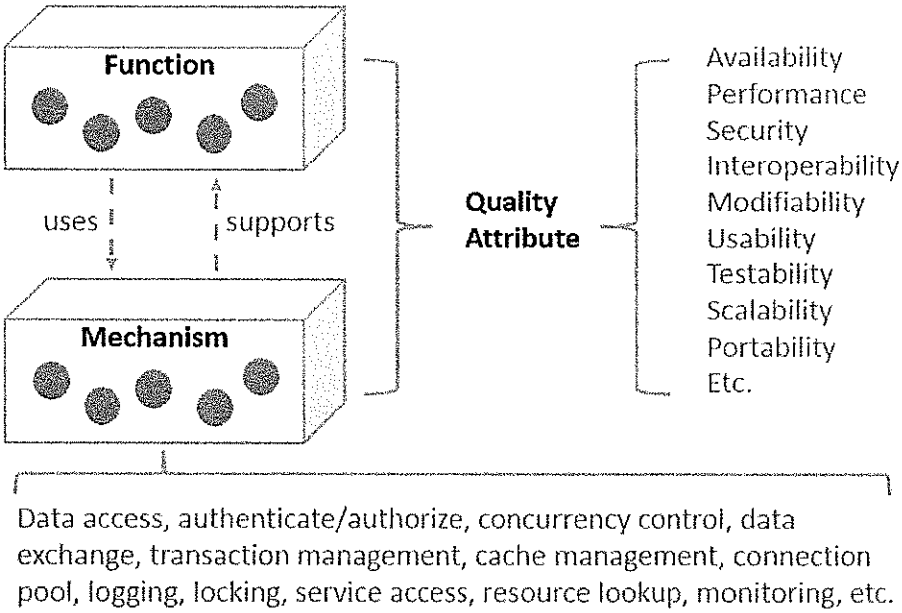
2) Non-Functionality หรือในทางสถาปัตยกรรมเรียกว่า 'กลไกทางสถาปัตยกรรม' (Architectural Mechanism) คือ การทำงานของระบบที่ไคลเอ็นต์ไม่สามารถมีปฏิสัมพันธ์ด้วยโดยตรงได้ หรือเรียกใช้ตรงๆ ไม่ได้ ซึ่งเป็นส่วนการทำงานภายใน เช่น กลไก Data Access, Encryption, Thread Control ซึ่งกลไกฯ นี้สามารถเรียกใช้กันเองได้ และถูกเรียกใช้โดยส่วน functionality ได้ หรือฟังก์ชั่นเรียกใช้กลไกฯ หรือ functionality เรียกใช้ non-functionality ได้



1.3.3 ความสัมพันธ์และผลกระทบระหว่างฟังก์ชั่นกับคุณภาพ

1.3.3.1 ความสัมพันธ์ระหว่างฟังก์ชั่นกับคุณภาพ

ในการสอบถาม วิเคราะห์ และบริหารความต้องการ ในหัวข้อ 1.3.1 และ 1.3.2 ต้องให้ความสำคัญกับ Non-function เนื่องจากมีผลต่อคุณภาพระบบ และมีผลต่อฟังก์ชั่นอื่นๆ เช่น เวลาสอบถามผู้ใช้งานต้องถามเรื่องคุณภาพด้วย หรือ กำหนดคุณสมบัติแอพพลิเคชั่นเซิร์ฟเวอร์ว่าควรมีคุณภาพที่ต้องการและกลไกการทำงานภายในอย่างไร อาทิ transaction management, caching, pooling, concurrency control, login, locking, data mapping, authenticate, authorize, state management เป็นต้น



ทั้งนี้ Quality Attribute คือ คุณสมบัติเชิงคุณภาพของสิ่งต่างๆ ภายในระบบที่มีการทำงานร่วมกัน (สถาปัตยกรรม โมดูล อ็อบเจ็คต์) และรวมถึงคุณภาพของระบบ เช่น ระบบที่มี Input และมี Output เกิดขึ้น ต้องมีการกำหนดคุณสมบัติเชิงคุณภาพไว้ ซึ่งระบบจะมีคุณภาพอะไรขึ้นอยู่กับ Stakeholder เป็นผู้กำหนดว่าต้องการให้ระบบมีศักยภาพเพียงใด

การกำหนด Quality Attribute มาจากการเก็บความต้องการประเภท Non-Functional Requirement ซึ่งแบ่งออกเป็น 3 ประเภท ดังนี้

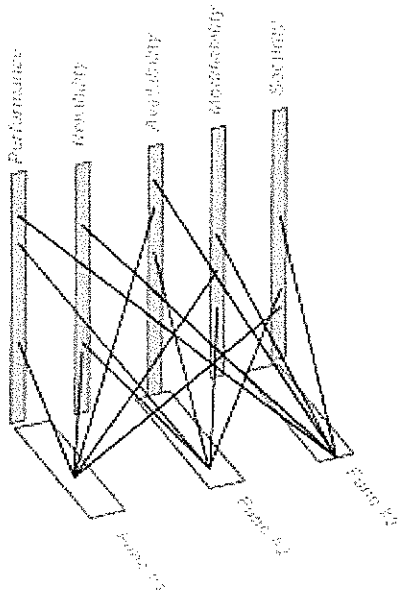
1) System Quality คือ คุณภาพระบบ มีหลักๆ 6 ตัว ได้แก่ Availability, Modifiability, Performance, Security, Testability และ Usability ซึ่ง System Quality *เรียกอีกอย่างว่า "Non-Functional Requirement"*

2) Business Quality คือ คุณภาพเชิงธุรกิจที่ระบบได้ตอบสนองต่อธุรกิจของ Stakeholder เช่น Increase Net Profit, Increase Competitiveness, Short Time to Market, Reduce Paper Work เป็นต้น หรือมองว่าเป็น Business Goal (มองภาพรวมของทั้งระบบหรือโครงการ) ที่มีการมองในระดับลึกและมีรายละเอียดมากกว่า

3) Architecture Quality คือ คุณภาพของสถาปัตยกรรมระบบ ไม่ได้มองที่การทำงานแบบ System Quality แต่มองที่ประโยชน์และการนำไปใช้ เช่น Conceptual Integrity, Buildability, Correctness and Completeness เป็นต้น

1.3.3.2 ผลกระทบระหว่างฟังก์ชั่นกับคุณภาพ

คุณภาพของซอฟต์แวร์ไม่ได้หาจาก Functional Requirement เพียงอย่างเดียว แต่มาจาก Non-Functional Requirement และการบริหารจัดการ (ออกแบบ เขียนโปรแกรม ทดสอบ) ด้าน Non-Functional ซึ่งซอฟต์แวร์ประกอบด้วยหลายฟังก์ชั่น และฟังก์ชั่นอาจต้องการคุณภาพด้านต่างๆ ที่แตกต่างกันไป เช่น ฟังก์ชั่นการบวกเลข 1 + 1 = 2 อาจต้องการคุณภาพด้าน Performance, Reliability



2 การบริหารความต้องการ

2.1 การเก็บ วิเคราะห์ และบริหารความต้องการ

- การเก็บรวบรวมความต้องการ (Requirement Gathering & Discovering) เป็นการรวบรวมปัญหา โดยอาศัยการสัมภาษณ์ การทำแบบสำรวจหรือแบบสอบถาม ซึ่งเริ่มจากการกำหนดกรอบเวลาให้ชัดเจน และเน้นเก็บความต้องการให้มากที่สุด โดยอธิบายแต่ละความต้องการสั้นๆ ประมาณ 1-3 บรรทัด ไม่เน้นทำความเข้าใจความต้องการ แต่เน้นปริมาณ และสิ่งที่บันทึกควรเป็นประเด็นหลัก เช่น ตัวอย่าง คือ Functional Requirement ควรระบุ Who, What, Why เป็นอย่างน้อย และอาจมี When, Where, How ได้

- การวิเคราะห์ความต้องการ (Requirement Analysis) คือ การวิเคราะห์เพื่อทำความเข้าใจความต้องการ โดยบรรยายด้วยข้อความหรือวาดรูปประกอบ เช่น ตัวอย่าง คือ Functional Requirement แบ่งเป็นหัวข้อ Pre-Condition, Post-Condition, Basic Flow, Alternative Flow, Business Rule เป็นต้น เมื่อวิเคราะห์เสร็จต้องส่งให้ User หรือ Stakeholder ช่วยกันตรวจสอบ เมื่อทุกคนเข้าใจตรงกัน จะนำไปวิเคราะห์และออกแบบระบบได้

- การบริหารความต้องการ (Requirement Management) คือ บริหารความต้องการตั้งแต่ต้นจนจบโครงการ เพื่อให้ความต้องการถูก “transform” ไปสู่กระบวนการต่างๆ เช่น วิเคราะห์และออกแบบระบบ เขียนโปรแกรม ทดสอบระบบ เป็นต้น เพราะการบริหารความต้องการ ถือเป็นศูนย์กลางของการบริหารที่สำคัญที่ต้องเคียงคู่ไปกับการบริหารโครงการ เพราะทุกกิจกรรมในกระบวนการพัฒนาซอฟต์แวร์ต้องอิงกับความต้องการ จึงนิยมตั้งเป็นหน่วยงานหรือทีมงานเฉพาะ เช่น หน่วยงาน Program Management Office (ทีมบริหารโครงการ), ทีมบริหารความต้องการ, ทีม Test, ทีม Architect เป็นต้น

2.2 การกำหนด Functional Requirement

การกำหนดหรืออธิบายรายละเอียด Functional Requirement จะมี 2 รูปแบบ คือ

1. แบบเรียบง่าย เหมาะกับแนวการพัฒนาระบบแบบ Agile โดยอธิบายเป็น Use Story คือ อธิบายสั้นๆ เช่น ไม่เกิน 1 สไลด์ใน MS PowerPoint โดยต้องระบุ Who, What, Why เป็นอย่างน้อย อาจมี When, Where และ How ได้

2. แบบละเอียด เหมาะกับแนวการพัฒนาระบบแบบ Object-Orientation ที่เป็นแบบทางการ โดยอธิบายเป็น Use Case Specification หรือ Use Case Description ซึ่งมีรายละเอียดดังนี้

- ชื่อ หมายถึง ชื่อ Use Case ควรใช้คำหรือข้อความสั้นๆ ที่ขึ้นต้นด้วยคำกริยา เพราะเวลาอ่านความต้องการ มักเริ่มอ่านจากไคลเอ็นต์หรือผู้ใช้มาที่ Use Case เช่น บุคคลทั่วไปสมัครสมาชิกทางอินเทอร์เน็ต
- รหัส หมายถึง รหัส Use Case เนื่องจากทุกความต้องการและทุกประเภทควรมีรหัสที่ขึ้นต้นด้วยอักษรย่อประเภทความต้องการ หรือขึ้นต้นด้วยรหัสระบบ แล้วปิดท้ายสุดด้วยลำดับตัวเลขของความต้องการ
- Brief Description หมายถึง ข้อความบรรยายความต้องการสั้นๆ ไม่ควรเกิน 3 บรรทัด
- Flow of Events หมายถึง ขั้นตอนที่ไคลเอ็นต์หรือผู้ใช้กระทำกับระบบ ซึ่งอธิบายในมุมว่าผู้ใช้งานมาใช้ระบบ ไม่ใช่อธิบายในมุมระบบทำอะไร เพราะอธิบายความต้องการที่เป็นกิจกรรมในงาน

Requirement ที่เน้นที่ปัญหาและความต้องการ เพราะหากอธิบายในมุมระบบทำอะไรให้ไคลเอ็นต์หรือผู้ใช้ แสดงว่ามองในมุม Solution ซึ่งคืองาน System Analysis คือ ระบบจะทำอะไร โดย Flow of Events สามารถอธิบายบรรยายเป็นขั้นตอน หรือวาดเป็นรูป Business Process หรือ Flow Chart ได้ แต่แนะนำให้วาดด้วยไดอะแกรม BPMN (Business Model Notation) หรือ UML Activity Diagram เพราะมีรายละเอียด มีสัญลักษณ์ให้เลือกใช้ และได้มาตรฐาน ซึ่ง Flow of Events มี 2 ประเภท ได้แก่

(1) Basic Flow หมายถึง โฟลว์หลักหรือขั้นตอนหลักที่ไคลเอ็นต์หรือผู้ใช้เข้ามาใช้ระบบ โดยไม่มี error เกิดขึ้นเลย และ ทำทุกขั้นตอนผ่านด้วยดีไม่มีติดเงื่อนไข busines rule ใดๆ เลยเช่น การถอนเงินที่ตู้ ATM มีขั้นตอนหลักๆ ได้แก่ 1) เสียบบัตร 2) กดรหัส 3) เลือกเมนู 4) เลือกประเภทบัญชี 5) กดจำนวนเงิน 6) รับเงินและบัตร

(2) Alternative Flow(s) หมายถึง โฟลว์ที่ไม่เป็นไปตาม Basic Flow ซึ่งเกิดได้จาก 2 สาเหตุ ได้แก่ เกิดจากผิดเงื่อนไขใน Basic Flow หรือ เกิด error ใน Basic Flow สำหรับ Use Case ใดจะมี Alternative Flow หรือไม่มีก็ได้ และสามารถมีได้หลายโฟลว์ เช่น เลือกเมนูถอนเงินผิดเป็นเลือกเมนูดูยอดคงเหลือ โฟลว์เลยเปลี่ยนไปหน้าจอแสดงยอดเงินคงเหลือ

- Pre-Condition หมายถึง Action หรือ Use Case ที่ไคลเอ็นต์หรือผู้ใช้ต้องทำก่อนจะมาทำ Use Case นี้ หรือระบุเป็นเงื่อนไขก็ได้ แต่หากไม่ชำนาญระวังอย่าระบุเงื่อนไขมากเกินไป ให้เน้นที่ Action หรือ Use Case ที่ต้องทำก่อนมาทำ Use Case นี้ เพราะจะทำให้คุมโฟลว์ภาพรวมได้ดีเช่น ต้องมีบัตร ATM ก่อนจึงจะถอนเงินที่ตู้ ATM ได้
- Post-Condition หมายถึง Action หรือ Use Case ที่ไคลเอ็นต์หรือผู้ใช้ต้องทำหลังจากทำ Use Case นี้เสร็จแล้ว หรือระบุเป็นเงื่อนไขก็ได้ แต่หากไม่ชำนาญระวังอย่าระบุเงื่อนไขมากเกินไป ให้เน้นที่ Action หรือ Use Case ที่ต้องทำหลังทำ Use Case นี้ดีกว่า เพราะจะทำให้คุมโฟลว์ภาพรวมได้ดีเช่น ถอนเงินเสร็จต้องเลือกว่าจะรับใบสลิปหรือไม่รับ
- Special Requirements หมายถึง Non-Functional Requirement ที่เฉพาะเจาะจงกับฟังก์ชั่นหรือ Use Case นี้เท่านั้น เช่น หน้าจอกดเงินต้องใช้ง่าย การถอนเงินที่ตู้ ATM ต้องมีความปลอดภัย ต้องทำงานรวดเร็ว
- Business Rules หมายถึง เงื่อนไขหรือตรรกะ คำว่า “Business Rule” เป็นคำที่นิยมใช้ในทางธุรกิจ ในไอทีใช้คำว่า “Business Logic” มากกว่า ซึ่งทั้ง 2 คำก็มีความหมายเหมือนๆ กัน ใช้แทนกันได้ แต่ในงาน Requirement นิยมใช้คำว่า Business Rule มากกว่า เช่น กดเงินตั้งแต่หลักร้อยขึ้นไป ห้ามถอนเงินเกินจากยอดคงเหลือ กดรหัสบัตรให้ถูก

ข้อสำคัญในการอธิบาย Runctional Requirement (Use Case)

- อย่าอธิบายเกี่ยวกับหน้าจอ หรือใช้คำที่เฉพาะเจาะจงกับหน้าจอจนเกินไป เพราะหากรายละเอียดบนหน้าจอเปลี่ยน จะทำให้ต้องตามแก้เอกสารความต้องการบ่อยตาม
- ใช้ข้อความสั้นกระชับ อย่าบรรยายเยอะ

- แยก Business Rule ออกมาจากข้อความอธิบายให้ชัดเจน เพราะมีประโยชน์ต่องานออกแบบ เขียนโปรแกรม ทดสอบระบบ
- ชื่อฟังก์ชั่น หรือ Use Case ควรใช้คำหรือข้อความสั้นๆ เพราะช่วยให้จำง่าย
- ระบุ Alternative Flow ให้มากที่สุด เพราะมีสำคัญต่องานออกแบบ เขียนโปรแกรม และ ทดสอบระบบ
- หากออกแบบหน้าจอแล้ว สามารถ Capture ภาพหน้าจอมาแปะใน Use Case Specification ได้ จะช่วยให้ผู้อ่านเห็นภาพขึ้น

ตัวอย่างการอธิบาย Use Case ละเอียดและเป็นทางการ

Use Case Specification : Register Internet Member (UC-001)

Brief Description

สำหรับผู้ที่ต้องการสมัครเป็นสมาชิกประเภท Internet Member เพื่อที่จะสามารถทำการซื้อสินค้าได้ หากผู้สมัครเป็นสมาชิกของทางร้านอยู่แล้วเมื่อสมัครเป็น Internet Member แล้วจะสามารถลงประกาศขายสินค้าผ่านทางเว็บได้

Flow of Events

Basic Flow

1. เลือกหัวข้อ '***สมัครสมาชิก***' บนหน้าเว็บผู้ที่ต้องการสมัครเป็นสมาชิกต้องเข้ามาที่เว็บก่อน โดยในทุกหน้าจะแสดงหัวข้อ '***สมัครสมาชิก***' ซึ่งอยู่ในบริเวณที่สังเกตง่าย
2. ป้อนชื่อ***ผู้ใช้***และ***รหัสผ่าน*** [BR-001], [BR-002], [BR-003]
3. ระบุว่าเป็นสมาชิกของทางร้านอยู่แล้วหรือไม่
4. ป้อนที่อยู่อีเมล์
5. กดปุ่ม '***ตกลง***' เมื่อเสร็จสิ้นจากนั้นระบบจะส่งอีเมล์แจ้งยืนยันการสมัครไปให้ผู้สมัครตามที่อยู่อีเมล์ที่ป้อน

Alternative Flows

A1 **ไม่ได้เป็นสมาชิกร้านค้า** [BR-004]

เกิดขึ้นที่ขั้นตอนที่ 3 ของ Basic Flow

1. เลือกประเภทอาชีพของผู้สมัครจากรายการ
2. ป้อนช่วงรายได้ต่อเดือน
3. ป้อนช่วงรายได้รวมของครอบครัวต่อเดือน
4. ป้อนที่อยู่และเบอร์โทรศัพท์ติดต่อ
5. กลับไปที่ขั้นตอนที่ 4 ของ Basic Flow

A2 **เป็นสมาชิกร้านค้าอยู่แล้ว** [BR-004]

เกิดขึ้นที่ขั้นตอนที่ 3 ของ Basic Flow

1. ป้อนหมายเลขสมาชิกร้านค้า
2. กลับไปที่ขั้นตอนที่ 4 ของ Basic Flow

Special Requirements

Security

การจัดเก็บและการเข้าถึงชื่อผู้ใช้และรหัสผ่านต้องมีความปลอดภัย ซึ่งต้องมีแต่เจ้าของชื่อผู้ใช้คนนั้นเท่านั้นที่เห็นรหัสผ่านได้ [BR-005]

Pre-Conditions

-

Post-Conditions

ตรวจสอบอีเมล์เพื่อ activate

เมื่อสมัครสมาชิกผ่านทางเว็บแล้ว ผู้สมัครต้องตรวจสอบอีเมล์เพื่อคลิ้กลิงค์ในอีเมล์ เพื่อเข้าสู่หน้าจอยืนยันและทำการ activate ชื่อผู้ใช้รายนั้นให้สามารถเริ่มใช้งานได้

Relevant Business Rules

- BR-001 ชื่อผู้ใช้ใช้ภาษาไทยได้
- BR-002 รหัสผ่านต้องป้อนสองครั้งให้เหมือนกัน
- BR-003 รหัสผ่านต้องเป็นตัวเลขและตัวอักษรผสมกันและห้ามเกิน 8 ตัว
- BR-004 ถ้าผู้สมัครเป็นสมาชิกร้านค้าอยู่แล้วให้ป้อนเฉพาะหมายเลขสมาชิก หากไม่ใช่ให้ป้อนรายละเอียดอื่นที่จำเป็นเพิ่มเติม
- BR-005 การจัดเก็บและการเข้าถึงชื่อผู้ใช้และรหัสผ่านต้องมีความปลอดภัย ซึ่งต้องมีแต่เจ้าของชื่อผู้ใช้คนนั้นเท่านั้นที่เห็นรหัสผ่านได้

2.3 การกำหนด Non-Functional Requirement (NFR) หรือ System Quality

การระบุ Non-Functional Requirement (NFR) จะอธิบายในรูปแบบการบรรยายเป็นเรื่องราว ถึงสถานการณ์สำคัญขณะที่ระบบกำลังทำงาน ซึ่งต้องอาศัยจินตนาการและมองระบบไปถึงอนาคต ว่า Stakeholder มีความกังวล (Concern) เกี่ยวกับอะไร ซึ่งเรียกสถานการณ์ว่า 'Scenario' หรือ "Quality Scenario" ซึ่ง Quality Scenario มีรายละเอียดดังนี้

1) Stimulus คือ สิ่งเร้าหรือสิ่งกระตุ้น ในทางไอทีใช้คำว่า "Trigger Event" ที่ทำให้สถานการณ์สำคัญนั้นๆ เกิดขึ้น (ไม่มีอะไรในโลกเกิดขึ้นได้ด้วยตัวมันเอง เมื่อมีสิ่งหนึ่งเกิดขึ้น สิ่งอื่นจึงเกิดขึ้นเป็นลำดับตามๆ กันมา) ให้มองถึง Storyboard หรือ Snapshot เช่น ระบบประมวลผล เพราะผู้ใช้ทำการ submit transaction

2) Source of Stimulus คือ แหล่งที่มาของสิ่งเร้า หรือตัวที่ก่อให้สิ่งเร้ามันเกิดขึ้นมา จะเป็นคนหรืออุปกรณ์ก็ได้

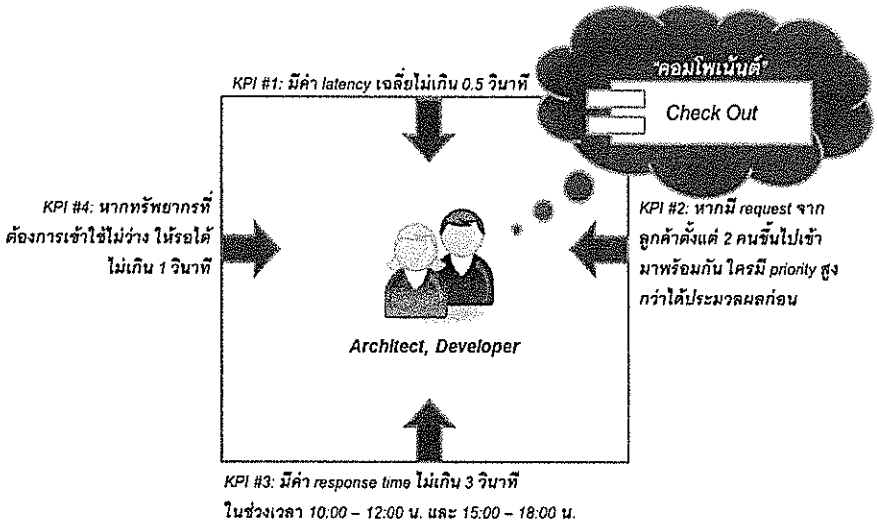
3) Artifact คือ สิ่งต่างๆ ที่เกี่ยวข้องมีผลต่อสถานการณ์สำคัญนั้นๆ เช่น System, Module, Hardware, Transaction, Network, Document เป็นต้น

4) Environment คือ สภาพแวดล้อมขณะระบบกำลังทำงาน เช่น Degraded Mode, Normal Mode เป็นต้น

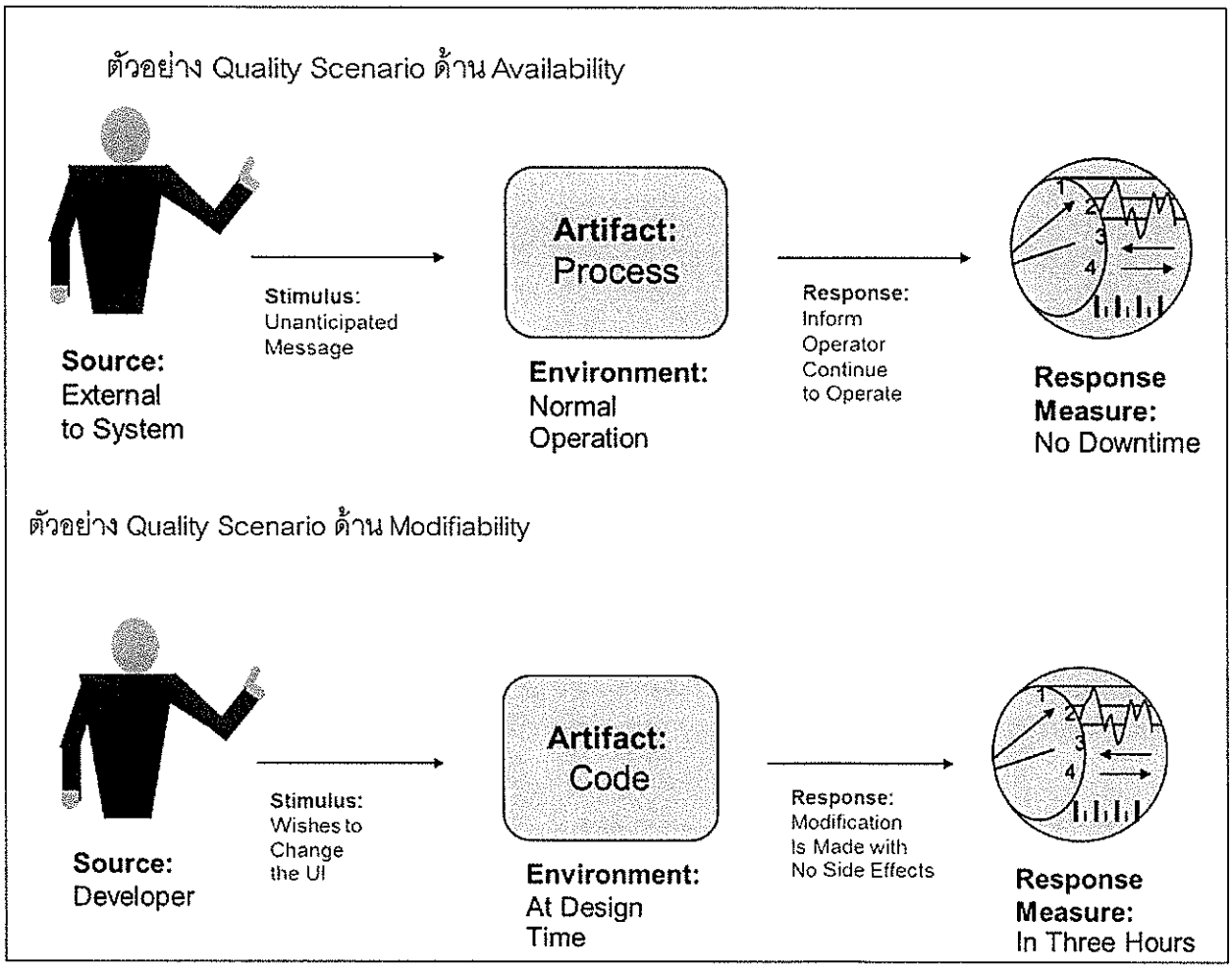
5) Response คือ การตอบสนองของระบบ เมื่อมีสิ่งเร้าเกิดขึ้น ดังนั้น Response จะสัมพันธ์กันกับ Stimulus

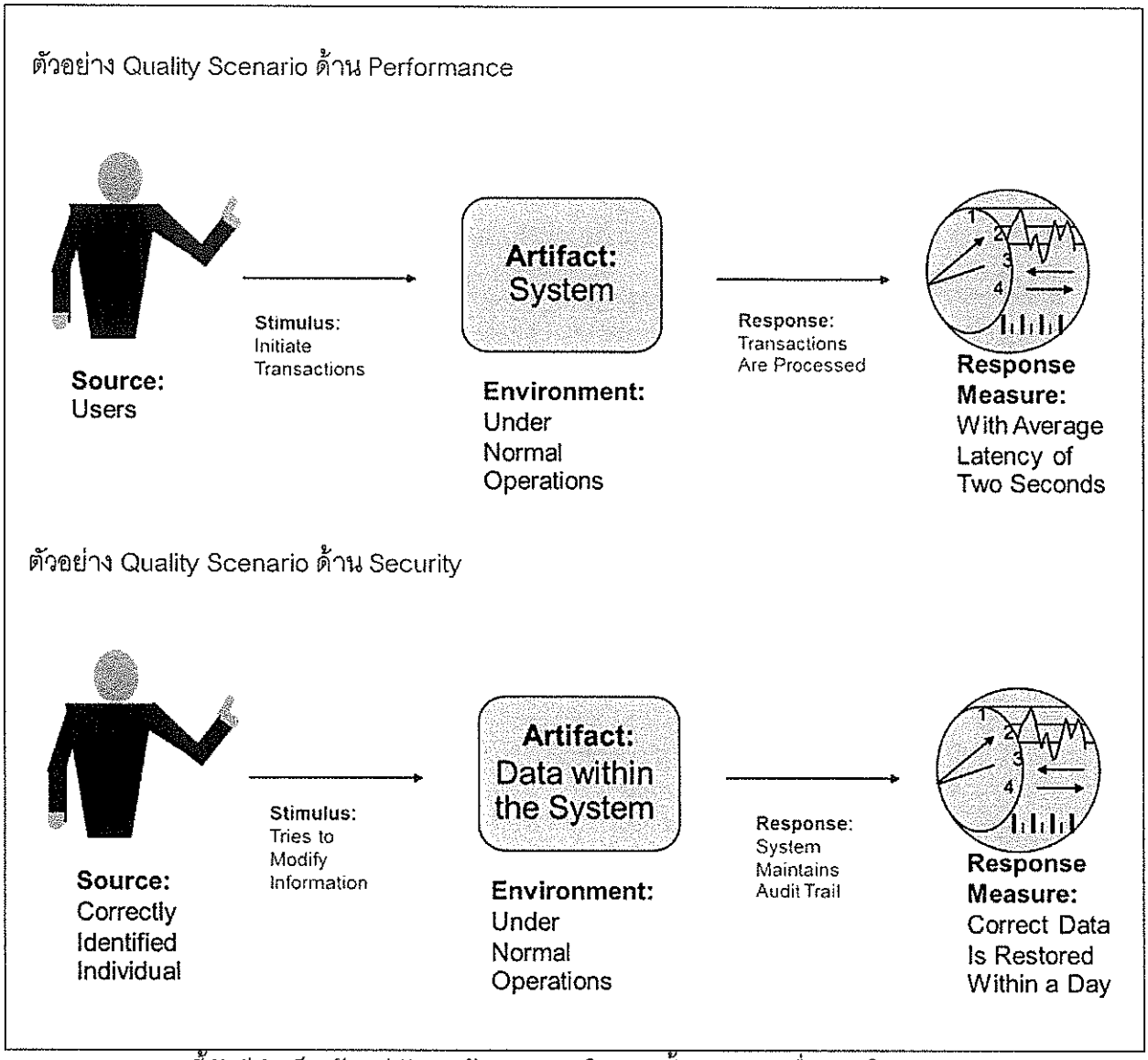
6) Response Measure คือ ตัววัดการตอบสนองของระบบ หรือ ตัวชี้วัด (KPI) คุณภาพ เพราะคุณภาพใดๆ ต้องมีตัวชี้วัดเสมอ ซึ่งจะแตกต่างกันไปตามแต่ละคุณภาพ

ทั้งนี้การกำหนด Non-Functional Requirement (NFR) หรือ Quality Attribute เป็นกรอบหรือ influence ให้กับทีมงาน เพื่อออกแบบและพัฒนาระบบให้ได้คุณภาพตามที่กำหนดได้



ตัวอย่าง Non-Functional Requirement (NFR)





นอกจากนี้ยังมีจำเป็นต้องมีทักษะด้านเทคนิคในการตั้งคำถาม เพื่อระบุในการหา Non-Functional Requirement (NFR) ประเภทต่างๆ เพื่อวิเคราะห์และจำลองสถานการณ์ (Quality Scenario) หรือ วิเคราะห์ด้านธุรกิจว่า Stakeholder มีความกังวล หรือมี Business Goal เกี่ยวกับอะไรบ้าง ซึ่งรายละเอียดอยู่ใน *ภาคผนวก ก.*

2.4 Soft Skill ที่มีความสำคัญต่องาน (Requirement)

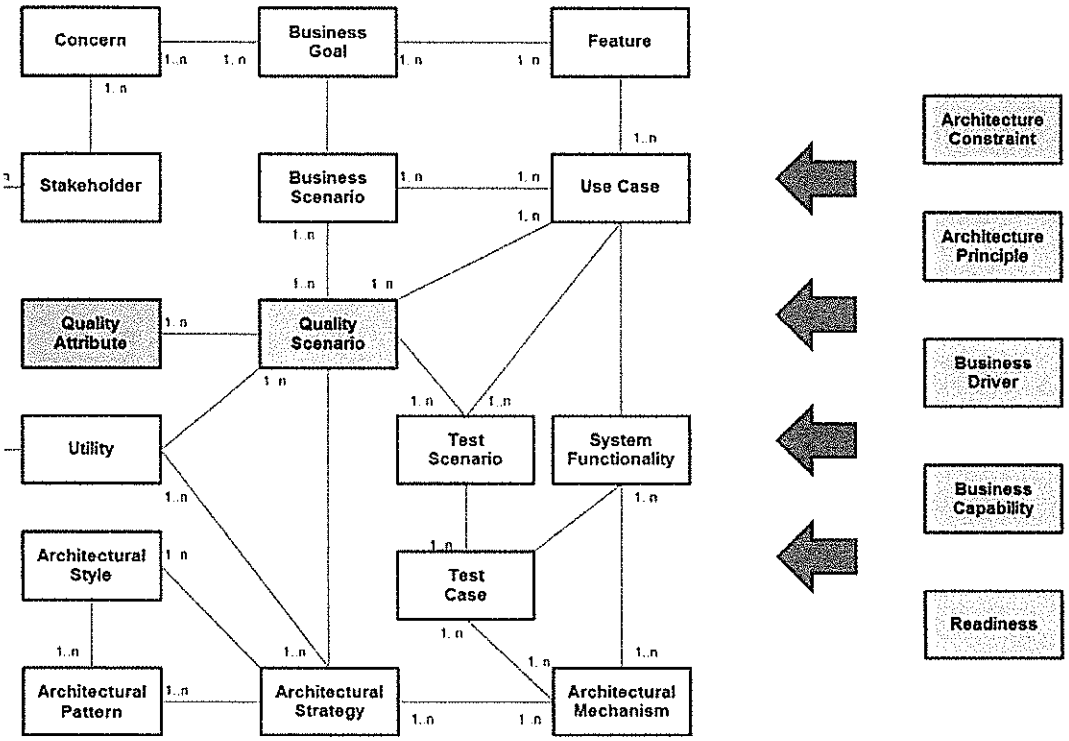
การบริหารความต้องการ นอกจากเข้าใจความรู้และเทคนิคต่างๆ แล้วบุคคลนั้นต้องมีทักษะส่วนบุคคลเพิ่มเติมในการจัดการปัญหาต่างๆ ที่เกี่ยวข้อง ดังนี้

- การรับมือประเด็น สังคม-การเมือง-ทุจริต
- การลดความขัดแย้ง และจัดการการสื่อสารกับลูกค้า
- การลดความกำกวมในการสื่อสาร
- การเข้าใจทัศนคติและความต้องการของ Stakeholder
- การพิสูจน์ปัญหาความต้องการ

3. บริบทและคุณสมบัติความต้องการในสร้างความสัมพันธ์แบบ Matrix

3.1 Architectural Requirement Context

Architectural Requirement Context คือ บริบทสำหรับความต้องการที่มีผลต่อการออกแบบและสร้างสถาปัตยกรรมซอฟต์แวร์ (Software Architecture) ซึ่ง Software Architecture ประกอบด้วยกลไกและการเชื่อมโยงการทำงานภายในระบบ โดยคิดปริมาณเฉลี่ย 20% ของระบบตามทฤษฎีของ Pareto และอีก 80% คือ Requirement ซึ่งประกอบด้วย Business Goals, Non-Functional Requirements และ Functional Requirements โดยมี Architectural Drivers เป็นตัวขับเคลื่อนสำคัญ ทั้งหมดนี้มีผลต่อซอฟต์แวร์โดยรวม



ปัจจัยที่มีต่อการระบุและวิเคราะห์ความต้องการ ซึ่งมีผล (impact) ต่อโครงการทางตรง/ทางอ้อม เช่น

- Architecture Constraint คือ ข้อจำกัดทางสถาปัตยกรรม เช่น ด้านธุรกิจ, สภาพตลาด/เศรษฐกิจ, เทคโนโลยี, ข้อมูล, องค์กร, งบประมาณ, การบริหาร
- Architecture Principle คือ หลักการพื้นฐานทางสถาปัตยกรรม เช่น ด้านธุรกิจ, เทคโนโลยี, ข้อมูล, ความปลอดภัย, แอพพลิเคชั่น
- Business Driver คือ ปัจจัยที่มีผลต่อการดำเนินธุรกิจ (เฉพาะ Business Case ที่สัมพันธ์กับโครงการนี้)
- Business Capability คือ ความสามารถทางธุรกิจ และรวมถึงความสามารถด้านอื่น
- Readiness คือ ความพร้อมในด้านต่างๆ เช่น นโยบาย, การบริหาร, งบประมาณ, บุคลากร

ปัจจัยกลุ่มนี้เรียกว่า Constraint คือ สภาวะภายนอกหรือปัจจัยภายนอกที่เป็นข้อจำกัดและเป็นแรงกดดันทำให้ Stakeholder เกิดความกังวล (Concern) เมื่อปัจจัยภายนอกส่งผลต่อความรู้สึกจึงทำให้เกิดความต้องการและความต้องการในลำดับแรกๆ คือ Business Goal หรือ Business Requirement

ขั้นตอนหลักในการวิเคราะห์ความต้องการ ตาม Architectural Requirement Context

1. ระบุ Stakeholder ของโครงการ และระบุ Concern ของแต่ละ Stakeholder โดยวิเคราะห์แบบ Outside-In คือ วิเคราะห์จาก Constraint ต่างๆ ที่มีผลทำให้ Stakeholder กังวล

- สร้าง Matrix ระหว่าง Stakeholder Concern กับ Constraint

2. วิเคราะห์ Stakeholder Concern แล้วระบุ Business Goal หรือ ปรับปรุง Business Goal หากมีอยู่แล้ว

- สร้าง Matrix ระหว่าง Stakeholder Concern กับ Business Goal

3. วิเคราะห์ด้านธุรกิจในมิติต่างๆ เช่น Business Scenario, Business Case, SWOT, Strategy Map ในระดับ Stakeholder Concern, Business Goal คือ พื้นที่ด้านแก่นปัญหาของระบบ (Problem Domain Space)

- สร้าง Matrix ระหว่าง Business Scenario กับ Business Goal

4. กำหนดแนวทาง (Solution) โดยวิเคราะห์ว่าควรมีระบบ/แอพพลิเคชั่นอะไร และควรมีความสามารถ (Feature) อะไรที่ตอบสนองต่อ Business Goal และ Stakeholder Concern ได้ จากนั้นสร้างในระดับ Feature คือ พื้นที่ด้านแนวทางแก้ไขปัญหาของระบบ (Solution Space)

- สร้าง Matrix ระหว่าง Feature กับ Business goal

5. วิเคราะห์ Feature เพื่อระบุ Functional Requirement หรือ Use Case

- สร้าง Matrix ระหว่าง Use Case กับ Feature

6. วิเคราะห์ Feature และ Business Scenario เพื่อระบุ Quality Scenario ซึ่งหมายถึงสถานการณ์ (ให้มองเป็นช็อตๆ เหมือนช็อตภาพยนตร์ หรือ Snap Shot) ว่าระบบที่กำลังพัฒนานี้ มีสถานการณ์หรือเหตุการณ์สำคัญใดบ้าง โดยให้คำนึงถึง Stakeholder Concern และ Business Goal เป็นเป้าหมายสำคัญ

- สร้าง Matrix ระหว่าง Quality Scenario กับ Business Scenario, Quality Scenario กับ Business Goal แต่หากไม่ได้ทำ Business Scenario ให้สร้างเฉพาะคู่หลังพอ

7. แต่ละ Quality Scenario ต้องระบุด้วยว่าสัมพันธ์กับ Quality Attribute หรือ Non-Functional Requirement ใด

- สร้าง Matrix ระหว่าง Quality Scenario กับ Quality Attribute

8. วิเคราะห์ Requirements ทั้ง Use Case และ Quality Scenario เพื่อระบุ Test Scenario โดยควรแยกเป็น Functional Test Scenario และ Architectural หรือ Non-Functional Test Scenario หรือจะเป็น Test Scenario ที่รวมทั้งด้าน Functional และ Non-functional ก็ได้ แต่ผู้วิเคราะห์ต้องมีทักษะที่ชำนาญพอ และต้องคำนึงถึงกระบวนการด้าน Testing และผู้ทดสอบที่จะ Test ด้วย

- สร้าง Matrix ระหว่าง Test Scenario กับ Quality Scenario และ Test Scenario กับ Use Case

9. วิเคราะห์ Test Scenario โดยเชื่อมโยงกันให้จำแนกและให้สอดคล้องกับ Stakeholder Concern, Business Goal, Feature, Use Case, Quality Scenario เพื่อระบุ, ออกแบบ และสร้าง Test Case ต่อไป โดยควรแยกเป็น Functional Test Case และ Architectural Test Case เสมอ ไม่ควรรวมกัน

- สร้าง Matrix ระหว่าง Functional Test Case กับ Test Scenario และ Functional Test Case กับ Use Case

10. ทำความเข้าใจ Use Case ที่วิเคราะห์แล้ว และ Functional Test Case เพื่อวิเคราะห์และออกแบบระบบในส่วน System Functional หรือ System Functionality โดยในขณะเดียวกันก็ทำความเข้าใจ Quality Scenario ที่วิเคราะห์แล้ว และ Architectural Test Case เพื่อวิเคราะห์และหาแนวทาง (Architectural Strategy)

- สร้าง Matrix ระหว่าง Architectural Test Case กับ Test Scenario และกับ Quality Scenario

11. สำหรับ Architectural Strategy (หรือเรียกว่า Architectural Tactic หมายถึง กลวิธีทางสถาปัตยกรรม) ควรระบุ Architectural Pattern, Architectural Style ที่ใช้ด้วย ซึ่งจริงๆ ควรระบุ Design Pattern ด้วย เพราะ Design Pattern ใช้ประจำ

12. โดย Architectural Strategy ต้องระบุด้วยว่ามีประโยชน์ (Utility) ต่อ Quality Scenario ใดบ้าง ครอบคลุมแค่ไหน จากนั้นก็ประเมินสถาปัตยกรรม (Architecture Evaluation)

- สร้าง Matrix ระหว่าง Architectural Strategy กับ Quality Scenario

13. นำ Architectural Strategy ที่ผ่านการประเมินเบื้องต้นมาวิเคราะห์, ออกแบบ และสร้าง Architectural Mechanism (กลไกทางสถาปัตยกรรม) ซึ่งต้องระบุด้วยว่าสนับสนุน System Functionality ใดบ้าง ในทางกลับกันคือ มี System Functionality ใด และมาเรียกใช้ (Call) Architectural Mechanism ใดบ้าง

- สร้าง matrix ระหว่าง
  - Architectural Strategy กับ Architectural Mechanism
  - Architectural Mechanism กับ Architectural Mechanism
  - Architectural Mechanism กับ System Functionality
  - System Functionality กับ System Functionality

3.2 Requirement Attribute, Traceability และ ตาราง Matrix

Requirement Attribute, Traceability และ ตาราง Matrix มีประโยชน์มากต่อการระบุคุณสมบัติและคุณลักษณะของความต้องการ ทั้งยังใช้ในด้านอื่นและมีประโยชน์ต่อด้านอื่นได้อีกมาก โดยเฉพาะการจัดลำดับความสำคัญของความต้องการ การเชื่อมโยงองค์ประกอบต่างๆ ในโครงการ เช่น เอกสาร, หน้าจอ, ความต้องการ, Test Case, โมดูล เป็นต้น อีกทั้งช่วยสนับสนุนการบริหารการเปลี่ยนแปลงและบริหารความเสี่ยงได้ดี และยังเป็นการช่วยลดการใช้ทัศนคติหรือความรู้สึกส่วนตัวในการพิจารณาความสำคัญของแต่ละความต้องการได้ด้วย

- Requirement Attribute คือ รายละเอียดของความต้องการ เช่น ข้อความบรรยายรายละเอียด , Traceability และแอททริบิวต์อื่นๆ เช่น Priority, Difficulty, Risk, Owner, Status, Customer Needs, Iteration Planned, Architecture Impact, Stability, Benefit, Effort และต้องระบุค่าด้วย เช่น Yes / No,

1-10, High / Medium / Low ซึ่งแต่ละ Requirement มี Requirements Attribute เป็นของตัวเอง และควรเปิดเผยให้ทุกคนในฝ่ายผู้พัฒนาระบบฯ สามารถดูรายละเอียดได้ โดยมีคำอธิบายแอททริบิวต์ดังนี้

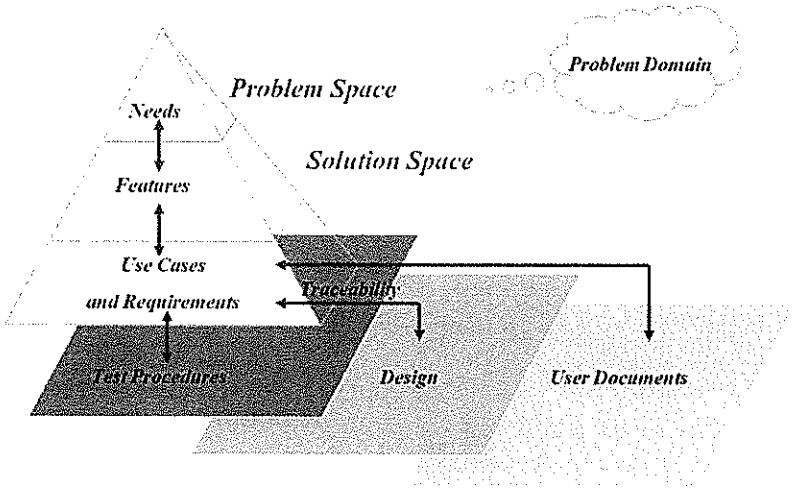
- Priority คือ ระดับความสำคัญของความต้องการ
- Difficulty คือ ระดับความยาก/ง่ายของความต้องการ
- Risk คือ ระดับความเสี่ยงของความต้องการ
- Owner คือ ผู้รับผิดชอบความต้องการตัวนั้นๆ
- Status คือ สถานะของความต้องการ เช่น proposed, approved, accepted, rejected, deleted
- Customer Needs คือ ระดับความต้องการในความต้องการตัวนั้นๆ ในมุมมองลูกค้าหรือผู้ใช้
- Iteration Planned คือ iteration / release / phase ที่จะอิมพลีเม้นต์ความต้องการตัวนั้นๆ
- Architecture Impact คือ ระบุว่าความต้องการตัวนั้นๆ มีผลกระทบต่อหรือเกี่ยวข้องกับสถาปัตยกรรมระบบหรือไม่
- Stability คือ ระดับเสถียรภาพของความต้องการ หรือ ที่มักเรียกกันว่า ความต้องการนิ่งแค่ไหนแล้ว เช่น ถ้านิ่งแล้วก็ให้คะแนน 8/10 หมายถึง ยังพอแก้ไขรายละเอียดเล็กน้อยได้ แต่ส่วนหลักนิ่งแล้ว เป็นต้น
- Benefit คือ ประโยชน์ของความต้องการ
- Effort คือ ทรัพยากรที่จะใช้อิมพลีเม้นต์ความต้องการ นิยมระบุเป็น man day, man hour เป็นต้น

ตัวย่าง Requirement Attribute ที่อธิบายเป็นตาราง Matrix

Features	Benefit	Effort	Risk	Architecture Impact	Stability
FEATURE1: Save and restore sort and filter criteria	Med High	Low	Low	Low	High
FEATURE2: Ability to save a RequisitePro document as a Microsoft® Word® document.	Med High	Low	Low	Low	High
FEATURE3: Ability to see deleted requirements in a view window.	Medium	Med High	Medium	Low	Medium
FEATURE4: Support for Currency datatype attributes.	Medium	Medium	Med Low	Low	Medium

- Traceability คือ ความสามารถในการตรวจสอบย้อนกลับ (Trace) ของอิลิเม้นต์ (Element) ในโครงการกับอิลิเม้นต์อื่นๆ ในโครงการได้ โดยเฉพาะที่เกี่ยวข้องกับ Requirement และอิลิเม้นต์ในโครงการนี้เรียกว่า Traceability Item ซึ่งโดยปกติแล้ว Item นี้ประกอบด้วยประเภทของ

Requirements โมเดลอิลิเม้นต์จากการวิเคราะห์และออกแบบ, อิลิเม้นต์ที่เกี่ยวกับการ Test, เอกสารการอบรม และใช้งานสำหรับ End user



และแต่ละ Traceability Item ต้องมีการระบุเซ็ตของแอททริบิวต์ (ศึกษาในเรื่อง Requirements Attributes) ซึ่งมีประโยชน์ในการติดตามสถานะ, ประโยชน์, ความเสี่ยง ฯลฯ ที่สัมพันธ์กับ Item นั้น ๆ เพราะจุดประสงค์ของ Traceability คือ เพื่อให้เข้าใจที่มาของ Requirements/ บริหารขอบเขตของโครงการ/ บริหารการเปลี่ยนแปลงที่มีต่อ Requirements/ประเมินผลกระทบจากการเปลี่ยนแปลง Requirement/ ประเมินผลกระทบจากความผิดพลาดจากการ Test ที่เกี่ยวข้อง Requirements/ ตรวจสอบให้แน่ใจได้ว่าทุก Requirements ได้ถูกอิมพลีเม้นต์ และ ตรวจสอบให้แน่ใจได้ว่าระบบฯ ทำงานได้ตรงตามสิ่งที่ควรจะเป็น (ตรงตาม Requirements) อีกทั้งTraceability นิยมอธิบายด้วยตาราง Traceability Matrix ซึ่งอาจประยุกต์ MS Excel มาใช้งาน หรือบันทึกลง database

ตัวอย่างตาราง Traceability Matrix 1

ตัวอย่างตาราง Matrix ระหว่าง Use Cases กับ System Qualities (H=High, M=Medium, L=Low)

Use Cases / System Qualities	Register Internet Member	Add New Product	View Product	Add Product to Shopping Cart	Check Out
Availability	H1	H2	H3	H4	H5
Performance			M1	H6	M2
Usability	M3	M4	L1	L2	H7
Modifiability				L3	L4
Security	M5				M6
Reliability			M7		H8
Interoperability	M8	M9	M10		M11
Integrability	M12	M13	M14	M15	M16

ตัวอย่างตาราง Traceability Matrix 2

ตัวอย่างตาราง Matrix ระหว่าง Use Cases กับ Architectural Mechanisms

Use Cases / Architectural Mechanisms	Register Internet Member	Add New Product	View Product	Add Product to Shopping Cart	Check Out
Data Access	H1	M1			H2
TX Management	M2	M3			H3
Exception Handling	L1	L2	L3	L4	M3
Session Management		L5	H4	H5	M4
Logging	L6	L7	L8	L9	L10
Object Pooling Management	H6	M6	H7	H8	H10
Legacy Connector	M7	M8	M9		M10
Isolation Locking Management					H11