9 WAYS TO SPLIT USER STORIES

เก้าวิธีช่วยแบ่ง USER STORIES ให้เล็กลง

โดย

ปิโยรส ปิยจันทร์
https://medium.com/@piyorot

หนังสือเล่มนี้เป็นการรวมรวบบทความจากบล็อกส่วนตัวของผมที่เขียนไว้ใน https://medium.com/@piyorot ถ้าสนใจก็ติดตามผลงานกันได้ครับ เขียนทุกวันครับ

piyorot

ผมสนใจเรื่อง Project Management and Software Development process ครับ ก็เลยเขียนบล็อกนี้ครับ http://t.co/h2JWWI562d

3 FOLLOWING

23 FOLLOWERS

เรื่องราวสมมติข้างล่างเป็นบทสนทนาระหว่าง

ศิริพงษ์ – Project Manager ผู้มีใจรักใน Agile Development และมีความเชี่ยวชาญพอสมควรในเรื่องของการเขียน User Story … กับ

เอกภพ – Product Owner มือใหม่ที่ถึงแม้จะเริ่มเข้าใจว่า User Story และ INVEST คืออะไร แต่เค้ายังมีข้อสงสัยถึงเทคนิคในการแบ่งย่อยให้ User Story มีขนาดเล็กลง

ศิริพงษ์ไม่รอช้า … นี่คือเก้าวิธีที่จะช่วยเอกภพได้

แบ่ง User Story ด้วยเทคนิค CRUD

Splitting a User Story Using ‘CRUD’ Technique

หลังจากศิริพงษ์เล่าถึงการเขียน User Story ที่ดีด้วยหลัก

- Independent
- Negotiable
- Value
- Estimatable
- Small
- Testable

จบแล้ว เอกภพมีความสงสัยในเทคนิคต่างๆที่ใช้ในการแบ่ง User Story ใหญ่ๆให้เล็กลง ศิริพงษ์จึงขยายความต่อ

ศิริพงษ์: “การแบ่ง User Story นั้นมีเทคนิคเยอะครับ พี่พูดถึง Data Boundaryไปแล้วงั้นขอแนะนำเทคนิคถัดไปที่เรียกว่า CRUD ครับ”

CRUD

ศิริพงษ์: “CRUD เป็นคำที่ประกอบขึ้นจาก Database Operations ที่เราใช้งานบ่อยนั่นคือ C = CREATE (หรือ INSERT), R = READ (หรือ QUERY/SELECT), U = UPDATE, และ D = DELETE ก็คือคำสั่งใน SQL แหละครับ

การใช้ CRUD จะเหมาะกับ User Story ที่ผูกกับ Object หรือ Entity ในระบบ เช่น ถ้าเรากำลังทำเวปจัดหางาน Object ที่มีก็จะเป็น User, Employer, Resume, Job Function, Job Position, Application, และอื่นๆ

ถ้าเรากำหนด User Story ให้กับ Object แต่ละตัวมีความเป็นไปได้ที่ User Story นั้นจะมีขนาดใหญ่ เช่น

> *As a user, I want to manage my resume, so that …*

ด้วยเทคนิค CRUD เราจะแบ่ง User Story นี้ตาม Database Operations แบบนี้ครับ

- CREATE: As a user, I want to create my resume, so that …
- READ: As a user, I want to read my resume, so that …
- UPDATE: As a user, I want to update my resume, so that …
- DELETE: As a user, I want to delete my resume, so that …

ไม่ยากใช่มั้ยครับ?”

เอกภพ: “ไม่ยากครับ แต่ผมลืมแล้วว่า Data Boundary เป็นยังไงอะครับ อธิบายอีกรอบได้ป่าว ฮ่าๆ? _/_”

แบ่ง User Story ด้วยเทคนิค Data Boundary

Splitting a User Story Using 'Data Boundary' Technique

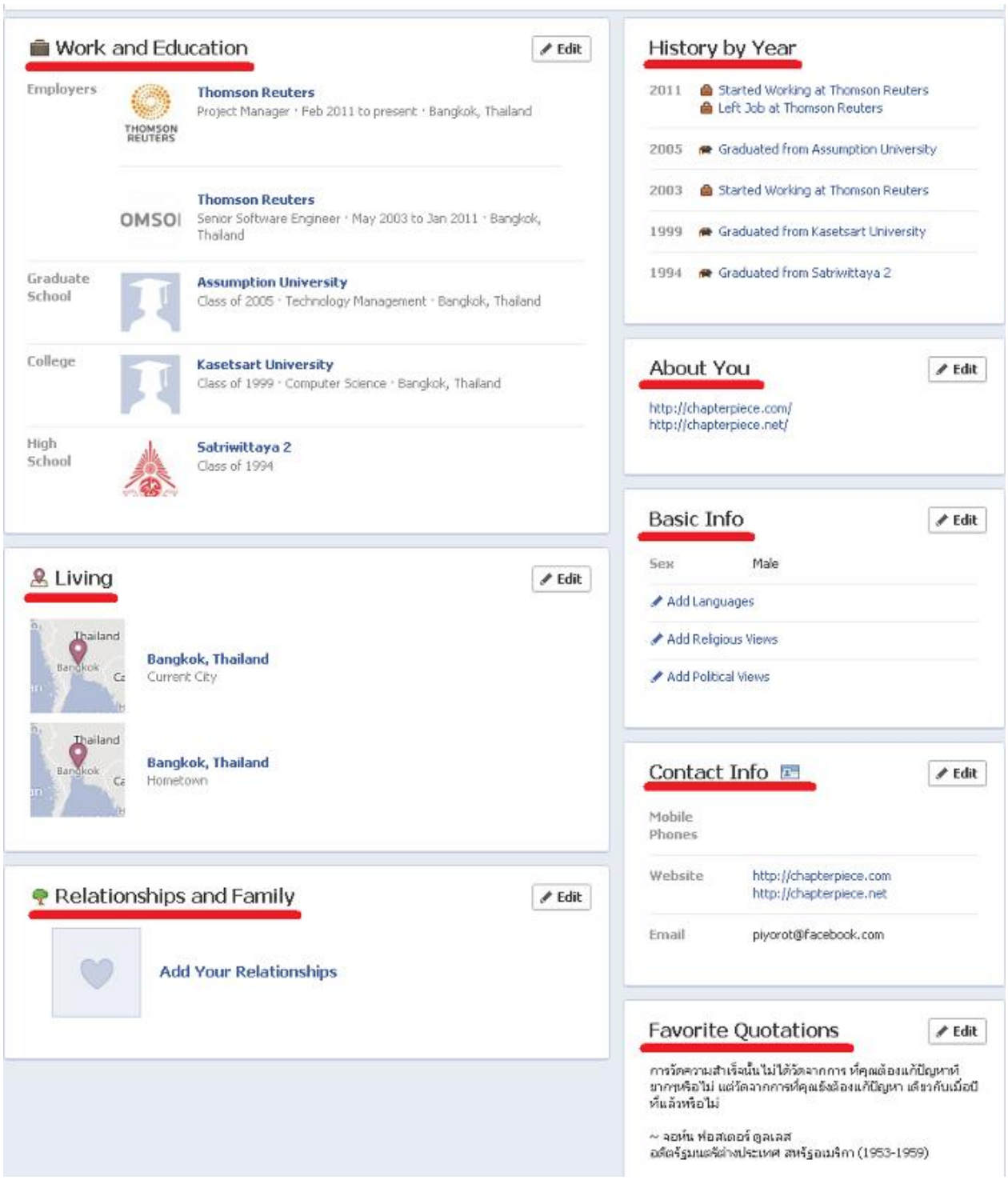
ศิริพงษ์: "ได้ครับ ... การแบ่ง User Story ด้วยเทคนิค CRUD และ Data Boundary สามารถเอามาใช้ร่วมกันได้ดีมากเลย งั้นอธิบาย Data Boundary ก่อน" ศิริพงษ์ยินดีทำตามคำขอของเอกภพ

Data Boundary

ศิริพงษ์: "Data Boundary แปลตรงตัวคือ 'ขอบเขตของข้อมูล' เพราะฉะนั้นเมื่อเอามาใช้กับการแบ่ง User Story มันก็คือ 'การแบ่งตามขอบเขตข้อมูล' ฮ่าๆ กำปั้นทุบดินมาก ยกตัวอย่างเลยแล้วกันครับ

รูปนี้แสดง Profile ของ User ถ้าเรามองว่า User เป็น Object หนึ่งของระบบเอกจะเห็นว่ามันแบ่งขอบเขตข้อมูลที่ชัดเจนเลยว่ามี

- Work and Education
- Living
- Relationship and Family
- History by Year
- About You
- Basic Info
- Contact Info
- Favorite Quotes



เวลาต้องการแบ่ง User Story ย่อยๆของ User Object เราก็พิจารณาที่ Data Boundary ของมันได้ครับ เช่นแทนที่จะเขียนว่า I want to see my User Profile ซึ่งมีขนาดใหญ่ เราแบ่งเป็น I want to see my Work and Education, I want to see my Basic Info เป็นต้น”

CRUD + Data Boundary

ศิริพงษ์: "ยิ่งเอามารวมกับ CRUD จะยิ่งมันส์เข้าไปใหญ่ แบบนี้ ...

- I want to create (CRUD) my Profile that includes Basic Info and Contact Info (Data Boundary).
- I want to add (CRUD) my Living (Data Boundary) into my Profile.
- I want to read (CRUD) my Profile that includes Basic Info, Contact Info and Living (Data Boundary).
- I want to edit (CRUD) my Basic Info (Data Boundary).
- ..."

เอกภพ: "เจ่งครับพี่ มีอีกมั้ย?" เอกภพตาลุกวาวด้วยความอึ้งและทึ่ง

ศิริพงษ์: "มีอีกเยอะครับ แนวทางการแบ่ง User Story ซอบหละซิ ฮ่าๆ"

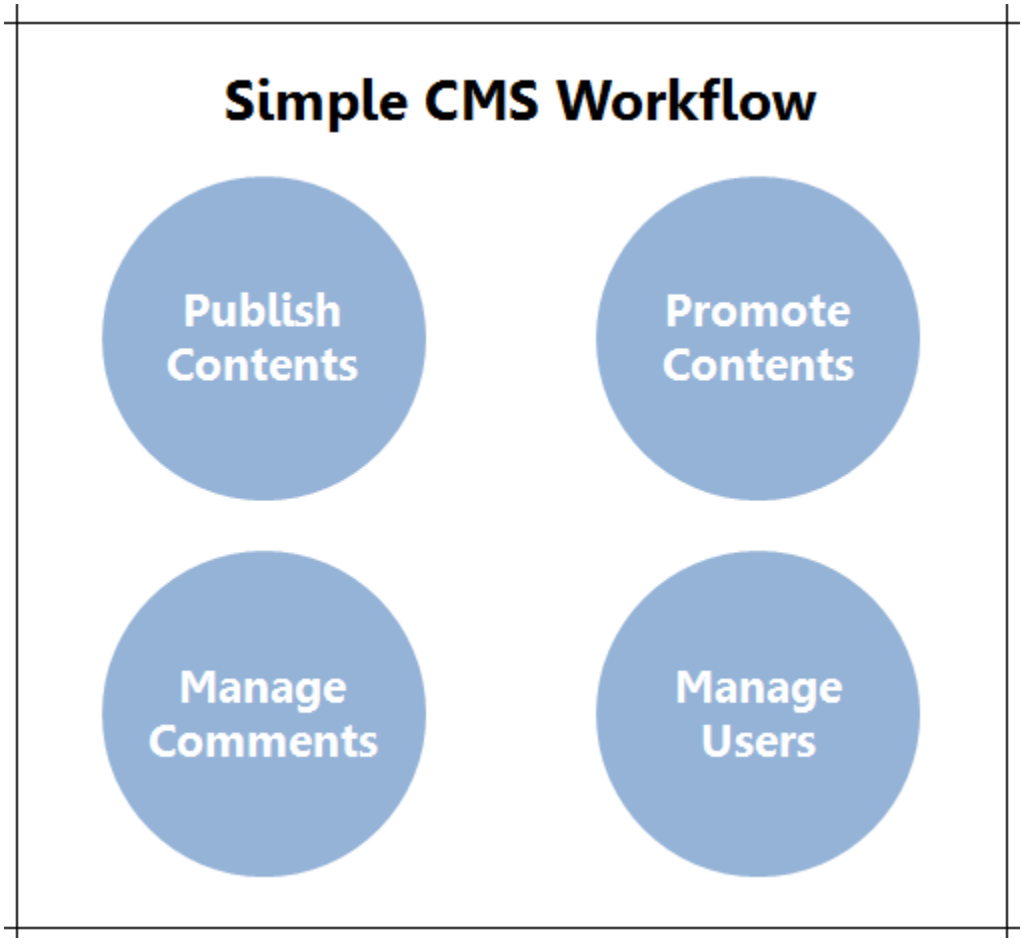
แบ่ง User Story ด้วย Workflow

Splitting a User Story Using Workflow

ศิริพงษ์: "เทคนิคต่อมาคือการแบ่ง User Story โดยใช้ Workflow ครับ"

Workflow

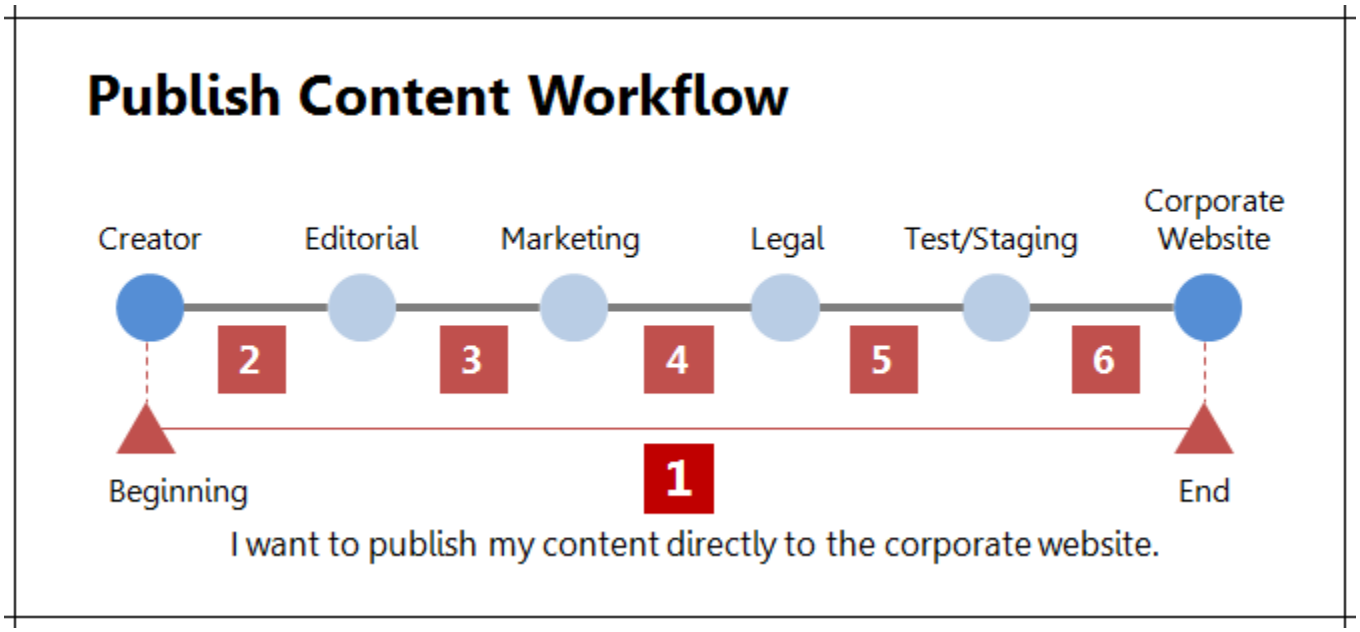
ศิริพงษ์: "นี่ก็เป็นเทคนิคที่ใช้ได้กับแทบทุกระบบเลย เช่น ระบบ Content Management System (CMS) ของเอกเมื่อไล่ดูคร่าวๆมันคือการเชื่อมต่อกันของ Workflow ต่างๆตามรูปซึ่งเมื่อเราวิเคราะห์ได้แบบนี้เราก็สามารถตั้งต้นเขียน User Story ชุดแรกออกมาได้แล้วครับ หลักการสำคัญคือ



เขียน User Story จากจุดเริ่มต้น (Beginning) ถึงจุดสุดท้าย (End) ก่อน จากนั้นพิจารณาเพิ่มขั้นตอนระหว่างกลาง (Middle Step) และกรณีพิเศษ (Special Case) ภายหลัง

ลองดูจาก Content Publication Workflow นะครับ User Story แรกเลยคือ

> *As a content creator, I want to publish my content to the corporate website, so that …*



แต่จากรูปบอกเราว่าก่อนที่จะเอา Content ไป Publish ที่ Website ได้ต้องผ่านการรีวิวจากกองบรรณาธิการ ฝ่ายการตลาด และฝ่ายกฎหมายก่อน รวมถึงต้องลองทดสอบ Publish ไปที่เครื่อง Test หรือ Staging ก่อนด้วย ขั้นตอนมากแบบนี้เลยทำให้ User Story ข้างบนมีขนาดใหญ่เกินไป เราต้องย่อยลงด้วยการแบ่งตาม Workflow Step ดังนี้ครับ

1. I want to publish my content directly to the corporate website.
2. I want to send my content to editorial team for review.
3. I want to send my content to marketing team for review.
4. I want to send my content to legal team for review.
5. I want to publish my content to test/staging machine.

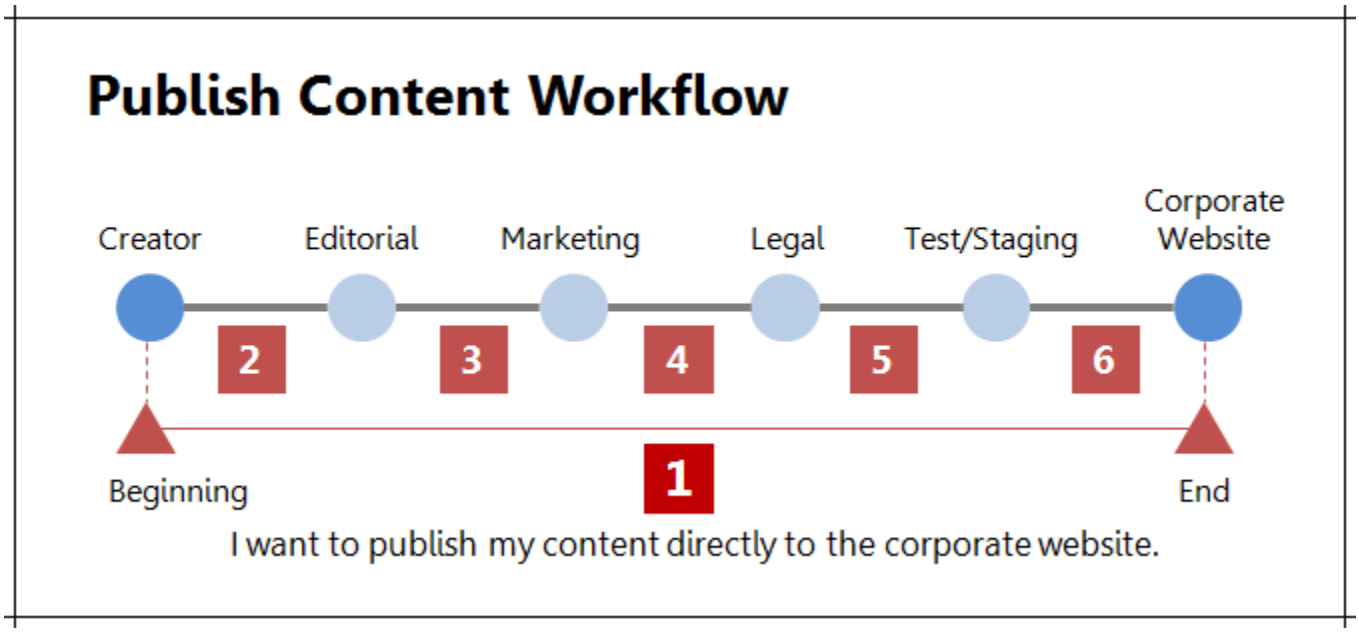
6. I want to publish my content from test/staging machine to the corporate website.
7. I want to save my drafted content.
8. …

เหตุผลที่กำหนดให้ User Story ที่ 1 มีความสำคัญสูงสุดเพราะมันเป็น User Story ที่มีคุณค่ามากที่สุด … ถ้า Content ไม่อยู่บน Corporate Website ก็ไม่มีคนอ่าน … เราจะเลือกทำอันนี้ก่อนให้เห็นภาพจากจุดเริ่มต้นถึงจุดสุดท้าย (End-To-End) ของ Workflow จากนั้นเราค่อยเพิ่มขั้นตอนหรือรายละเอียดระหว่างกลางลงไปเรื่อยๆครับ”

แบ่ง User Story ด้วย Roles

Splitting a User Story Using Roles

ศิริพงษ์: "โดยส่วนมากแล้วการแบ่ง User Story ด้วย Workflow จะมาคู่กับการแบ่งด้วย Roles หรือประเภทของผู้ใช้ระบบซึ่งสามารถมองเห็นภาพได้จากการวิเคราะห์ Workflow ครับ ลองดูรูปเดิมนะ"



Publish Content Workflow for CMS

Roles

ศิริพงษ์: "จากรูปเราเห็นว่ามีประเภทของผู้ใช้อยู่อย่างน้อยห้าประเภทคือ (1) Creator, (2) Editorial, (3) Marketing, (4) Legal และ (5) Reader/Viewer จากนั้นเราก็มาลงรายละเอียดของสิ่งที่แต่ละคนต้องการ

Role	User Story
Creator	-I want to publish my content to corporate website. -I want to send my content to editorial, marketing, legal team to review. -I want to publish my content to test machine. -I want to save my drafted content. -I want to add image/audio/video into my content.
Editorial	-I want to approve content. -I want to print out content for review. -I want to share content with my team.
Reader/Viewer	-I want to read content from corporate website. -I want to comment content. -I want to add content to read-later list. -...

Splitting User Stories Using Roles

ถ้ามี User Story ไหนใหญ่เกินไปเราก็แบ่งย่อยมันด้วยเทคนิคอื่นได้อีกครับ เช่นแบ่งด้วย Workflow อีกครั้งสำหรับ User Story ของ Editorial ที่ว่า I want to approve content

- I want to send approval message back to a creator.
- I want to review content.
- I want to make changes to content.
- I want to save my change.
- I want to continue adding my change.
- I want to have 'track changes' feature.
- I want to have multiple reviewers.

ประมาณนี้ครับ"

แบ่ง User Story ด้วย Business Rules

Splitting a User Story Using Business Rules

ศิริพงษ์: "หลายครั้งเราจะเจอ User Story มีขนาดใหญ่เพราะมันประกอบขึ้นจาก User Story เล็กๆหลายอันหรือที่เรียกว่า Compound Story และหนึ่งในปัจจัยที่สร้าง Compound Story ก็คือกฎเกณฑ์ทางธุรกิจ (Business Rules) ที่มีมากมาย"

Business Rules

ศิริพงษ์: "ยกตัวอย่างเป็นกรณีการสร้างรายงานแล้วกันนะ สมมติว่า Product Owner เขียน User Story มาแบบนี้ครับ

As a user, I want to generate a revenue report with flexible dates.

เห็นได้ชัดเลยครับว่ามันไม่ใช่ User Story เล็กๆเพราะคำว่า "flexible dates" นั้นกำกวมและอาจจะซ่อน User Story เล็กๆไว้มากมาย หลังจาก Development Team พูดคุยกับ Product Owner แล้วจึงรู้ว่า flexible dates มี Business Rules ที่ผูกติดอยู่หลายข้อ เช่น As a user, I want to generate a revenue report

- for 'today'.
- for 'this week'.
- for 'this month'.
- for 'this quarter'.
- for 'this year'.
- for 'week to date'.
- for 'month to date'.
- for 'year to date'.
- for 'all time'.
- starting from 'x' to 'y'.

หลังจากที่ได้ Revenue Report ออกมาแล้ว Product Owner มี User Story มาเพิ่มอีกหนึ่งอันคือ

As a user, I want to filter results in a revenue report by flexible criteria.

นี่ก็เป็น Compound Story เหมือนกันครับ Development Team ก็ต้องถามยืนยันกับ Product Owner ว่า "flexible criteria" มีอะไรบ้าง คำตอบที่ได้อาจจะเป็นดังนี้ As a user, I want to filter results in a revenue report

- by 'sources of revenue - single, multiple, and all'.
- by 'target vs. actual'.
- by 'tax included/excluded'.

การแบ่งย่อยแบบนี้ทำให้เราได้ User Story ชุดใหม่ที่มีขนาดเล็กลงจาก User Story ตั้งต้น (Epic) แต่มันจะมีขนาดเล็กพอแล้วหรือเล็กไปหรือไม่เราก็ต้องพิจารณาดูอีกทีครับ"

แบ่ง User Story ด้วย Data Entry Methods

Splitting a User Story Using Data Entry Methods

ศิริพงษ์กำลังเมามันส่วนเอกภพกำลังเมาหมัดจากหลักการแบ่ง User Story ที่มีมากมาย ศิริพงษ์คิดว่าไหนๆก็ไหนๆแล้วโซโลให้มันจบในวันเดียวไปเลยดีกว่า ว่าแล้วเค้าก็จัดหนักต่อไป

ศิริพงษ์: "เอกรู้อะไรปะ บางครั้งที่ User Story มันซับซ้อนหรือมีขนาดใหญ่ไม่ใช่เพราะการทำงานภายในของมันแต่เป็นเพราะวิธีการใช้งานต่างหาก ที่เห็นได้ชัดเลยคือหลายครั้งมันยากเพราะการมี User Interface เข้ามา แน่นอนมันทำให้ผู้ใช้ใช้งาน Feature นี้ได้ง่ายขึ้นแต่ความง่ายในการใช้งานมักจะมากับความยากของการ Implement ฮ่าๆ"

Data Entry Methods

ศิริพงษ์: "ถ้าเราเจอสถานการณ์แบบนี้ เราเลือกที่จะแบ่งงานให้เล็กลงและง่ายขึ้นด้วยหลักการที่เรียกว่า Data Entry Methods ครับ นั่นคือแยกระหว่างวิธีการที่ง่ายที่สุดในการนำข้อมูลเข้าระบบ (Data Entry) แล้วจากนั้นค่อยเพิ่มลูกเล่นเข้าไป ยกตัวอย่าง เราทำระบบ To-Do-List มี User Story ง่ายๆมาอันนึงคือ

> *As a user, I want to create my first to-do-list.*

เรารู้สึกว่าหน้า UI มันยากสุดๆ เราจะทำไงดี? เอาแบบนี้ไปเลย

- As a user, I want to create my first to-do-list using Command Line.
- As a user, I want to create my first to-do-list using simple UI.

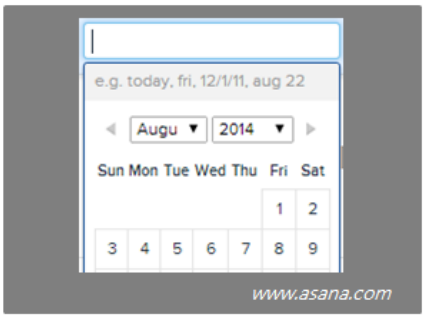
ฮ่าๆ มันอาจจะเวอร์ไปนิด พี่แค่ยกตัวอย่างให้เห็นภาพนะครับ ทั้งสอง User Story มีเป้าหมายสูงสุดเหมือนกันคือ 'สร้าง to-do-list' ให้ลูกค้าแต่อันแรกจะง่ายกว่าในแง่ Implementation เพราะตัดส่วนที่เป็น UI ออกไปเลย

ถามว่ามีคุณค่าสูงสุดกับลูกค้าหรือไม่ก็คงไม่เพราะลูกค้าส่วนใหญ่คงไม่ใช้ Command Line แต่ถ้าเรามองว่านี่เป็นกรณีที่ลูกค้าจะ Migrate to-do-list ที่เค้ามีเป็นสิบเป็นร้อยอันจากระบบอื่นมาระบบเราหละ ... การ Import ข้อมูลเยอะๆผ่าน Command Line ก็เป็นอะไรที่สมเหตุสมผลใช่ปะ? อยู่ที่สถานการณ์ครับเรื่องแบบนี้"

เอกภพ: "พี่ครับ แต่ขั้นที่สองที่ว่า 'simple UI' ถ้ามันยังยากอยู่หละครับ?"

ศิริพงษ์: "แน่นอนครับ มันยังยากอยู่ แต่เราก็ใช้วิธีการเดียวกันนี่แหละแบ่งให้มันเล็กลงและง่ายขึ้น พี่

ยกตัวอย่างแบบนี้ครับ โดยทั่วไป to-do-list ต้องมี Due Date ด้วย และเมื่อพูดถึง Date หรือวันที่เราก็จะนึกถึง Calendar ซึ่งตรงนี้เราสามารถทำให้ง่ายก็ได้ยากก็ได้แบบนี้

	Evolution of Calendar UI		
1	Due Date:	12 August 2014	พิมพ์วันที่เองทั้งหมด
2	Due Date:	12 ▼ 8 ▼ 14 ▼	เลือกวัน เดือน ปี จาก Dropdown Box
3	Due Date:		เลือกวันที่จากปฏิทิน
4	Due Date:		เลือกวันที่จากปฏิทินหรือพิมพ์วันที่ด้านบนก็ได้ เช่น Today, Fri, ...

Evolution of Calendar UI

วิวัฒนาการของ Calendar UI ก็อยู่ในรูปแบบ User Story ที่เราเลือกจะทำได้ครับ.

แบ่ง User Story ด้วย Major Effort

Splitting a User Story Using Major Effort

ศิริพงษ์: “มี Compound Story บางประเภทที่ถึงแม้เราจะแบ่งให้มันเล็กลงแล้วแต่ก็ยังไม่เป็นอิสระต่อกัน (Independent) อยู่ดี เช่น

> *As a user, I want to pay for my items using VISA, MasterCard, or American Express.*

สำหรับ User Story นี้เราแบ่งมันได้เป็นสาม User Story ย่อยๆ ดังนี้

- As a user, I want to pay for my items using VISA.
- As a user, I want to pay for my items using MasterCard.
- As a user, I want to pay for my items using American Express.

ดูแล้วก็เหมือนจะดีนะ แต่ความเป็นจริงเมื่อเริ่มทำงานคือเราต้องสร้าง Credit Card Processing Infrastructure ให้กับ VISA ถ้าเราต้องการเพิ่ม MasterCard หรือ American Express ภายหลังก็จะเป็นเรื่องง่ายเลยเพราะมี Infrastructure อยู่แล้ว

ถ้าเจอสถานการณ์แบบนี้เราจำเป็นต้องประเมินขนาดของงานให้ VISA ใหญ่กว่า MasterCard และ American Express ปัญหาที่จะตามมาคือเราต้องไม่ลืมปรับขนาดของงานถ้า Product Owner เปลี่ยนความสำคัญของงาน เช่นจากตอนแรก VISA -> MasterCard -> American Express มาเป็น MasterCard -> American Express -> VISA ก็ยุ่งยากพอสมควร แต่เรามีทางเลือกอีกแบบในการเขียน User Story ครับ”

Major Effort

ศิริพงษ์: “Major Effort คือแบ่งด้วยระยะเวลาที่ต้องใช้ จากตัวอย่างข้างบนเรารู้เลยว่า Credit Card ใบแรกที่เราทำจะมีขนาดใหญ่มาเพราะต้องเตรียม Credit Card Processing Infrastructure ส่วนอีกสองใบที่ตามมาจะมีขนาดเล็กกว่าอย่างชัดเจนแต่เราไม่แน่ใจว่าจะเริ่มทำที่ Credit Card ประเภทไหนก่อน เราควรเขียน User Story แบบนี้ครับ

- As a user, I want to pay for my items with one credit card type (VISA, MasterCard, American Express). -> **Major Effort**
- As a user, I want to pay for my items with the rest of credit card types (VISA, MasterCard, American Express)(given that one card type already implemented).

User Story ทั้งสองอันนี้ถึงแม้จะไม่ได้เป็นอิสระต่อกันอย่าง 100% เพราะ User Story แรกต้องทำก่อนเสมอแต่ความสัมพันธ์ในเรื่อง Implementation จะชัดเจนกว่าเขียน User Story แยก Credit Card แต่ละประเภทเพราะเรารู้ทันทีว่า Credit Card Processing Infrastructure จะรวมอยู่ใน User Story แรก (Major Effort) เสมอครับ”

แบ่ง User Story ด้วย Error Handling Method

Splitting a User Story Using Error Handling Method

ศิริพงษ์: "เคยได้ยินคำว่า Positive Case (Happy Path) กับ Negative Case มั้ยครับ? Feature ที่เราทำทั้งหมดมีสองคำนี้เป็นส่วนประกอบ คำว่า Positive Case ก็คือการที่ผู้ใช้ใช้งาน Feature นี้ได้อย่างถูกต้องและบรรลุวัตถุประสงค์ที่เค้าต้องการ ส่วน Negative Case ก็ตรงกันข้ามถ้าเค้าจะพยายามใช้ Feature ด้วยวิธีการที่ไม่ถูกต้องซึ่งระบบเราก็ต้องป้องกันไม่ให้ Case เหล่านี้เกิดขึ้นหรือส่งผลกระทบต่อระบบโดยรวม ... อันนี้แหละที่เรียกว่า Error Handling ครับ"

ศิริพงษ์: "เราสามารถแบ่ง User Story โดยพิจารณาที่ Error Handling Method ได้"

Error Handling Method

ศิริพงษ์: "พี่ไม่ได้สนับสนุนให้ปล่อยปะละเลยนะแต่เราเลือกวิธีการจัดการ Error ได้ครับ เช่น ระบบเราจองตั๋วหนัง ถ้ามองงานนี้ให้เป็น Workflow เราควรต้องเขียน Workflow หลักขึ้นมาก่อนนั่นคือ

> *As a user, I want to book movie tickets, ...*

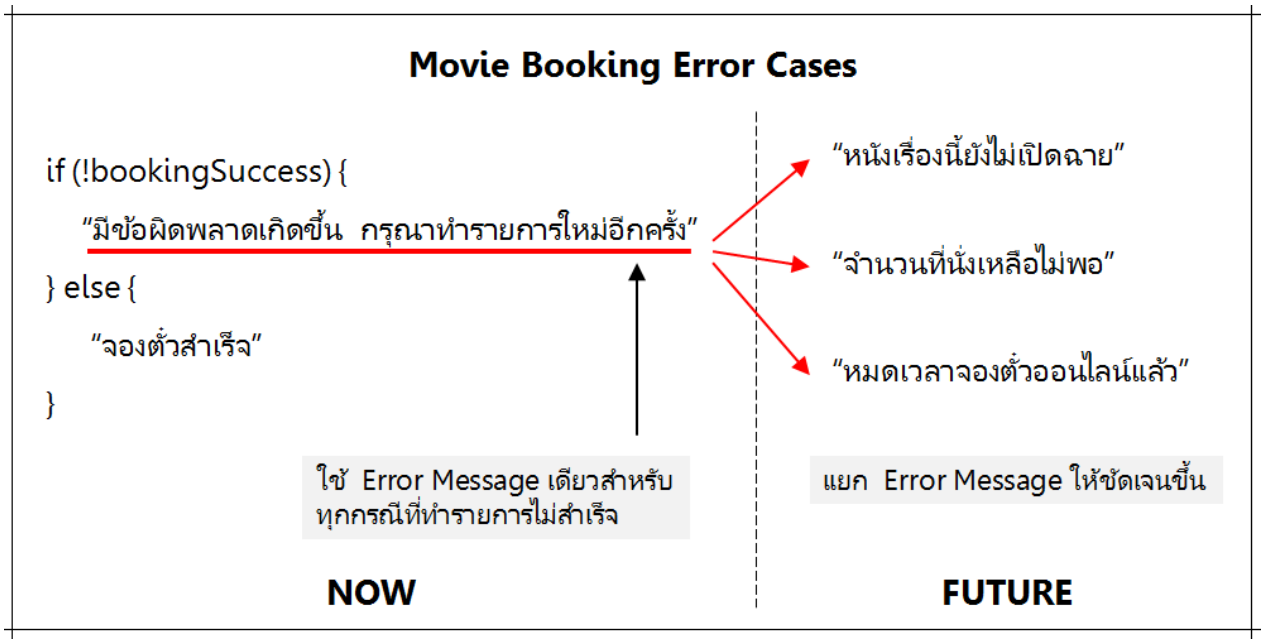
แต่เรารู้แล้วหละว่าการที่จะให้ลูกค้าจองตั๋วหนังเรื่องใดหรือรอบใดได้ต้องมีBusiness Rules แบบนี้

1. หนังต้องเปิดฉายแล้ว
2. รอบที่ต้องการดูมีที่นั่งเหลือเพียงพอ
3. จองก่อนหนังฉาย 20 นาที

ซึ่งถ้าทำทั้งส่วนที่เป็น Positive Case (จองตั๋วสำเร็จ) และ Negative Case (จองตั๋วไม่สำเร็จ) มันก็จะเป็น User Story ที่ใหญ่เกินไป ดังนั้นเราแบ่งมันได้แบบนี้ครับ

- As a user, I want to successfully book movie tickets, ...
- As a user, I want to be notified if something bad happens while I am booking movie tickets, ...

ที่น่าสนใจคือ User Story ที่สอง ถ้าการที่จะแจ้งเตือนในรายละเอียดทุกขั้นตอนมันอาจจะเยอะไป เราเลือกที่จะแจ้งเตือนเป็นข้อมูลกลางๆที่ตัดรายละเอียดของ Error นั้นออกไปก่อน



Movie Booking Error Cases

หลังจากนั้นค่อยมาทำให้ Error Handling มันฉลาดขึ้น เช่น

1. ระบุรายละเอียดใน Error Message ให้มากขึ้น
2. อาศัยการ Enable/Disable ปุ่ม ช่องว่าง หรือส่วนประกอบอื่นๆบนหน้าเวปไปเลย
3. เพิ่มการเขียน Error Message และรายละเอียดทางเทคนิคอื่นๆลงใน Log File ครับ"

แบ่ง User Story ด้วย ‘Spike’ Technique

Splitting a User Story Using ‘Spike’ Technique

ศิริพงษ์: “พี่เคยอธิบายเรื่อง Spike ไปแล้วก่อนหน้านี้ ตอนนี้ขอขยายความเพิ่มเติมอีกนิดนะ ... อันนี้เทคนิคสุดท้ายแล้ว”

ศิริพงษ์: “ถ้าเราเจอสถานการณ์ที่เราไม่เข้าใจว่า User Story นี้ต้องทำอะไร หรือเรารู้สึกว่ามีความเสี่ยง (Risk) สิ่งที่ไม่รู้ (Unknown) และความซับซ้อน (Complexity) อยู่เยอะ เราสามารถใช้เทคนิคที่เรียกว่า “Spike” มาช่วยได้

Spike

Spike คือการใช้เวลาทำ Technical Investigation เพื่อตอบคำถามที่มีใน User Story นั้นๆ มันมีประโยชน์หลายอย่างครับ

1. ถ้าทีมเราไม่มีความรู้เฉพาะทาง (Domain Knowledge) Spike จะใช้เพื่อทำความคุ้นเคยกับเทคโนโลยีเหล่านั้น เช่น AngularJS คืออะไรวะ? ขอศึกษาหน่อย
2. ถ้าเราคิดว่า User Story นี้มันใหญ่ไปไม่สามารถ Estimate ได้ เราจะใช้ Spike เพื่อหาข้อมูลเชิงลึกเพิ่มเติมเพื่อช่วยในการแบ่ง User Story ให้เล็กลง
3. ถ้าเรารู้สึกยังไม่มั่นใจว่า Technical Solution นี้จะเวิร์คจริงๆ เราจะใช้ Spike เพื่อลองทำ Research หรือ Prototype ดูก่อน อันนี้จะช่วยให้เรามีความมั่นใจมากขึ้น
4. บางครั้งถึงแม้เราจะเข้าใจวัตถุประสงค์ของ User Story นี้แต่มันไม่ชัดเจนว่าลูกค้าจะใช้งานหรือ Interact กับระบบยังไง (อันนี้เรียกว่า Functional Risk) เราก็จะใช้ Spike เพื่อพิสูจน์สมมติฐานก่อน

โดยสรุป Spike แบ่งได้สองรูปแบบครับ

Technical Spike: ใช้สำหรับตอบคำถามเชิงเทคนิคล้วนๆ เช่น วิธีการนี้จะช่วยแก้ปัญหา Performance และ Load ได้มั้ย? การเลือกใช้เทคโนโลยีตัวนี้จะช่วยตอบโจทย์ของงานนี้หรือไม่?

Functional Spike: ใช้เพื่อสร้างความชัดเจนในเรื่องการใช้งานหรือความสัมพันธ์ของระบบกับผู้ใช้ ส่วนใหญ่ Functional Spike จะออกมาในรูปแบบของ Prototype, Mock-Up, Wireframe เป็นต้น

และหลายครั้ง User Story เดียวสามารถแบ่งออกมาได้ทั้ง Technical Spike และ Functional Spike ครับ เช่น สำหรับ Mobile Wallet App

As a user, I want to see my daily, weekly and monthly expenses displayed in pie chart so that I can review and understand my expense level quickly.

มาแบบนี้เราแบ่งได้สอง User Story ครับ

1. To research what technique we can use to display pie chart in iOS, check the level of battery this technique uses, and check app loading time. -> **Technical Spike**

2. To create prototype that displays daily, weekly, and monthly expense in pie chart and get feedback from users. -> **Functional Spike**

ครบแล้วครับเก้าเทคนิคที่ใช้ในการแบ่ง User Story” ☺