Case Study of a (not so pleasant) Wizard Project at ABC

Presented by: Zijin Li
Background

About ABC:

- E-commerce
- B2C
  - Buy products from vendors
  - Sell to customers
- Huge number of users
Background

About my team:

- Supply chain technologies
- Goal
  - Improve efficiency
  - Lower costs
  - Make more profits
Background

About infraction, chargeback and dispute:

- Policies for vendors
  - Shipping time
  - Packaging quality
  - Mandatory freight information
  - ...
- Infraction: when vendors do not comply with a policy
- Chargeback: ABC charges vendors for their infractions
- Dispute: vendors can dispute chargebacks issued incorrectly
Problem

- Vendors are not familiar with the policies
- Infractions exist in their supply chain
- Chargebacks are issued by ABC
- Vendors blindly dispute all chargebacks
- ABC receives a huge number of invalid disputes
  - Disputing on correctly placed chargebacks
  - Disputing on incorrectly placed chargebacks, but without appropriate evidence
Project Goal

- Add a feature that can help vendors understand the policies
- Educate them on why they are charged and how to avoid similar charges
- Reduce the number of disputes
Initial Idea

- A wizard for each infraction type
- Launch wizard before vendors dispute
- One question leads to another depending on vendors’ choices
- End node: dispute or reject
RE Change Before Implementation

Configurable wizard:

- Static pages and links are not scalable
- Wizards need to be created and updated without code change
- A system for admin to create and update wizards
- Wizards are stored in DB
- Another system to retrieve wizards and display to vendors
RE Change Before Implementation, Again

Customized question:

- Generic messages are not good enough
- Vendors need to see specific info to better understand the situation
- Using tokens

Infraction table:

Record_1: [“ship_date”: “2017-07-01”...]

In DB:

You shipped on $ship_date, is it true?

What the vendor see:

You shipped on 2017-07-01, is it true?
RE Change Before Implementation, and Again

Auto-completed question:

- No need to ask vendors everything
- Some questions should be automatically completed according to the data we have
- Another backend system to deal with auto-completed questions

Example:

A. \$ship\_date > \$deadline
B. \$ship\_date <= \$deadline
RE Change Before Implementation, and Again...

Required document:

- When the wizard ends we know the vendor’s situation
- We know what evidence the vendor need to upload to make a valid dispute
- Make it mandatory
  - If vendors do not upload required document, do not let them submit the dispute
- Requires me to understand the existing dispute-submit system and make changes to it
RE Change During Implementation

Tracking info:

- Add metrics to track what vendors select
- Can be used to help refine the wizards

Non-dispute mode:

- Allow vendors to run the wizards on expired chargebacks to receive useful information
Observations: Emotions Matter

- Multiple parties related
  - Project manager
  - Infraction owners (people who created the policies)
  - Vendors
  - Vendor experience team (people who make sure vendors have good user experience)

- Forgot about vendor experience team
  - New requirements from them
  - Not quite necessary
Observations: Emotions Matter, Cont’d

- Conflicts between parties
  - Project manager and infraction owners want the wizard to be length
  - Vendors and vendor experience team do not agree

- Have a meeting with representatives from conflicted parties
  - Let the representatives negotiate directly
  - Do not be the middleman
Observations: Requirements Will Change

- Requirements change -> new requirements show up
- New requirements -> more change and new requirements
  - Tokens -> auto-complete questions

Questions to ask:

- (Before implementation) At what point do we start writing code?
- (During implementation) Do we want to deal this new requirement or not?
Observations: Requirements Will Change, Cont’d

- Functional requirements may be unstable
- Not high level project goals and non-functional requirements
- Distinguish essential requirements and additional requirements
  - Essential requirements: contribute to project goals or non-functional requirements
  - Additional requirements: “what you want but not what you need”
  - Example:
    - Goal: Educate vendors
    - Non-functional requirements: Scalability
    - Configurable wizard, customized question, auto-completed question -> essential
    - Required document, non-dispute mode -> additional
Conclusions

- Described a development experience at ABC

Lessons learned:

- Emotions matter
  - Consider all relevant parties
  - Gather requirements from different parties as early as possible to avoid major changes later
  - Deal with conflicts by asking representatives to negotiate directly

- Requirements will change
  - Identify essential requirements and additional requirements
  - Cut off additional requirements or move to next iteration
Thank You

Questions?