試験に役立つ
モデリングの紹介

サイボウズ 株式会社
渡邉 豊幹

自己紹介

渡邉 豊幹

サイボウズ株式会社
開発本部

2011年新卒入社、QA
現在は販売管理システムを担当

製品紹介

中小企業向けグループウェア

サイボウズ Office

業務アプリ構築クラウド

kintone

大企業向けグループウェア

Garoon

メール共有システム

メールワイズ

お話しすること

- モデルを作成し、試験に活用した話
- メリットと作成時のポイント

アジェンダ

- 担当プロダクト/チームの紹介と抱えていた問題
- 実施した解決策「モデルセットを作成する」
- 作成時のポイントとコストについて
- 今後の展望

担当プロダクト

- 米国市場向けの販売管理システム
- ユーザー

1. 製品のエンドユーザー
   - 試用申し込みや契約 など
2. 社内のセールスチーム
   - 特別価格での発注やライセンス管理、キャンペーンの設定 など

チームの紹介

チーム構成

開発スタイル

状況

チームの紹介

チーム構成

開発スタイル

状況

- 開発チームは6名（うち QAは3人）
- セキュリティ検証は社内専門チームに依頼

チームの紹介

チーム構成

開発スタイル

状況

試験後に
QA がリリース

- スクラム開発（1スプリント1週間）
- リリース頻度は約2回/週

チームの紹介

チーム構成

開発スタイル

状況

事前に全てを
設計しない

実装前に全て設計することや、内部の詳細な
ドキュメントを作成することはせず、
段階的に開発を行いリリースしている

チームの紹介

チーム構成

開発スタイル

状況

個々の
機能仕様書は
あるけど…

1. 開発が進むにつれて処理が複雑になる
2. 各機能の仕様書はあるが、全体像を把握できるドキュメントが無い

開発上の問題点

試験の影響範囲が把握しにくい

新規メンバー/他チームへの教育コストが高い

開発上の問題点

試験の影響範囲が把握しにくい

新規メンバー/他チームへの教育コストが高い

⬇

連携している外部サービスとのシステム境界が複雑になり、試験設計の難易度が上がる

開発上の問題点

試験の影響範囲が把握しにくい

新規メンバー/他チームへの教育コストが高い

↓

個々の機能の詳細な仕様書はあるが、
全体像を把握するための資料が無い

開発上の問題点

試験の影響範囲が把握しにくい

新規メンバー/他チームへの教育コストが高い

基本的なケースのモデルを
作成して解決できないか

なぜモデルが必要?

- 文書よりイメージの方が概要をうまく掴める
- 細かい点だけに着目すると全体像を掴みにくい

モデルに着目したきっかけ

平鍋健児さんを招いて開催された社内勉強会

- アジャイルにおいて必要十分な設計とは？
- 一貫性を保って開発するには？
- 効果的な対話のために必要なものは？

→ シンプルなモデルセット(Keeps)を作成する、というアイディアを知る

Keeps?

モデル

Keeps

モデルとして残すもの
・安定している
・寿命が長い

Temps

残さないもの
・実装中に変わる
・一度共有して捨てる

Keeps?

モデル

Keeps

モデルとして残すもの
・安定している
・寿命が長い

共通理解を促す
シンプルな
モデルセット

残さ
・実装中に変わる
・一度共有して捨てる

作成したKeepsの紹介

Keeps

Architecture

システム全体の構造を表すもの

Domain Model

業務で扱う対象とそれらの繋がりを表したもの

Key Use Cases

ユーザーから見たシステムの代表的な使い方

作成したKeepsの紹介

Keeps

Architecture

システム全体の構造を表すもの

Domain Model

業務で扱う対象とそれらの繋がりを表したもの

Key Use Cases

ユーザーから見たシステムの代表的な使い方

作成したKeepsの紹介

Keeps

Architecture

システム全体の構造を表すもの

Domain Model

業務で扱う対象とそれらの繋がりを表したもの

Key Use Cases

ユーザーから見たシステムの代表的な使い方

作成したKeepsの紹介

Keeps

Architecture

システム全体の構造を表すもの

Domain Model

業務で扱う対象とそれらの繋がりを表したもの

Key Use Cases

ユーザーから見たシステムの代表的な使い方

作成したKeepsの紹介

Keeps

Architecture

システム全体の構造を表すもの

Domain Model

業務で扱う対象とそ

ユースケース図
コミュニケーション図

Key Use Cases

ユーザーから見たシステムの代表的な使い方

ユースケース図

※一部秘密情報のため、実際の表現を変えています

ユースケース図

※一部秘密情報のため、実際の表現を変えています

コミュニケーション図

※一部秘密情報のため、実際の表現を変えています

改善できた点

試験の影響範囲が把握しにくい

新規メンバー/他チームへの教育コストが高い

改善できた点

試験の影響範囲が把握しにくい

新規メンバー/他チームへの教育コストが高い

1. 連携サービスとの境界や、サービスが実行されるタイミングが分かりやすくなった
2. 仕様書では細かい挙動に注目しがちだが、俯瞰して影響範囲を見ることができる

改善できた点

試験の影響範囲が把握しにくい

新規メンバー/他チームへの教育コストが高い

1. システム全体で各ユーザーがどのような操作を行うのか、俯瞰して見ることができる
2. 各ユースケースごとのシステムの振る舞いについても概要が把握しやすい

改善できた点

試験の影響範囲が把握しにくい

新規メンバー/他チームへの教育コストが高い

その他

1. モニタリング対象の API を洗い出す際に活用
2. E2E のテストの改善で活用

作成時のポイント

- 目的を明確にする
- チーム内で合意をとる
- ツールを活用する

作成時のポイント

目的を明確にする

チーム内で合意をとる

ツールを活用する

1. スコープを明確にし、作りすぎを防ぐ
2. 作成するケースを最小限に抑える

作成時のポイント

目的を明確にする

チーム内で合意をとる

ツールを活用する

1. メンテナンスコスト含めて検討する
2. 更新すべき情報について共有する

作成時のポイント

目的を明確にする

チーム内で合意をとる

ツールを活用する

PlantUML を使用し作成コストを抑えた

作成時のポイント

PlantUML
コードで UML を作成するツール

1. 誰でも同じような図を作成できる
2. コードなので git でバージョン管理できる

見た目の微調整が難しいが、厳密な表現より
概要を把握できることを優先して使用

メンテナンスについて

ルール

作成コスト

メンテナンスコスト

メンテナンスについて

ルール

作成コスト

メンテナンスコスト

既存のプロセスに
合わせる形で開始

- 開発時（仕様変更時）に PG/QA が修正
- 上記ルールは他の機能仕様書などと同様

メンテナンスについて

ルール

作成コスト

メンテナンスコスト

クラス図 ×1
オブジェクト図×1
ER図×1
コミュニケーション図×1
ユースケース図×7

- 10人日程度
- 計11枚の図と説明を作成

メンテナンスについて

ルール

作成コスト

メンテナンスコスト

- 2020年6月の運用開始後、誤りを一度修正
- 仕様変更による修正はまだ無し

今後の展望

チーム内用語を記載

ADR の記載

今後の展望

チーム内用語を記載

ADR の記載

チーム特有の用語を定義し記載することで
認識に齟齬が生じることを防ぐ

今後の展望

チーム内用語を記載

ADR の記載

Architecture Decision Records と呼ばれる、
アーキテクチャ上重要な意思決定の
背景を記載し、暗黙知が増えることを防ぐ

まとめ

- 下記問題を解決するために Keeps を作成した
  - 試験の影響範囲が把握しにくい
  - 新規メンバー/他チームの教育コストが高い

- 作成ポイント
  - 目的を明確にする
  - チーム内で作成について合意をとる
  - 楽に作成できるツールを活用する
  - メンテナンスのルールを決める

- 今後の展望
  - 用語の定義と ADR を記載する

参考情報

- 平鍋健児さんの資料
「Modeling in the Agile Age」
https://www.slideshare.net/hiranabe/modeling-in-the-agile-age-228224347

- PlantUML
https://plantuml.com/