全員参加型レビューのススメ

デザイナーがいないチームでUXを向上するための数年に渡る取り組み

株式会社インテック　長谷川真裕

今日は
「デザイナーがいないチーム
でUXを向上するための
数年に渡る取り組み」
についての話をします。

今日は
「デザイナーがいないチームでUXを向上するための数年に渡る取り組み」
についての話をします。

ISO 9241-210において

「製品、システム、サービスを使用した、および／または、使用を予期したことに起因する人の知覚（認知）や反応」と定義される

つまり

「ユーザーが製品・サービスを通じて得られる体験」

わたし × UX

今日は
「デザイナーがいないチームでUXを向上するための
数年に渡る取り組み」
についての話をします。

計画

ビジネス戦略

業務仕様

市場分析

用件定義

開発

DB設計

API設計

UI設計

ビジュアルデザイン

データ設計

レビュー

リリース

移行

教育

テスト

運用維持

チューニング

評価

計画

ビジネス戦略

業務仕様

市場分析

用件定義

開発

DB設計

API設計

UI設計

ビジュアルデザイン

データ設計

レビュー

UXは経営的なレベルで語られる

UXはUIレベルや製品・サービスで語られる

経営者
UX専門コンサル

PM
エンジニア
デザイナー
マーケティング
営業

組織・文化
ビジネスシステム

製品・サービス・機能(UIなど)

計画

ビジネス戦略

業務仕様

市場分析

用件定義

開発

DB設計

API設計

UI設計

ビジュアルデザイン

データ設計

レビュー

UXは経営的なレベルで語られる

UXはUIレベルや製品・サービスで語られる

経営者
UX専門コンサル

PM
エンジニア
デザイナー
マーケティング
営業

組織・文化
ビジネスシステム

製品・サービス・
機能(UIなど)

この発表中ではUXを

「エンドユーザーがソフトウェアを使用してすでにある業務を円滑におこなうことができること」≒「チームやUIで向上できるユーザー体験」

という観点で使用しています。

もう少し具体的に言うと、
「派生開発において
使いやすさを維持するための
品質向上プロセス」
についての話です。

！注意！

この発表では以下の話については都合上割愛しています。

1. 何を以て使いやすいと言えるか、といった基準の話

2. 検出率や生産性についての数値的な話

目次

1

「使いやすさ」という品質

ソフトウェア品質？

クオリティ・ゲート

マネジメントシステム

ソフトウェア品質

リファクタリング

ITIL

ISO

要求仕様

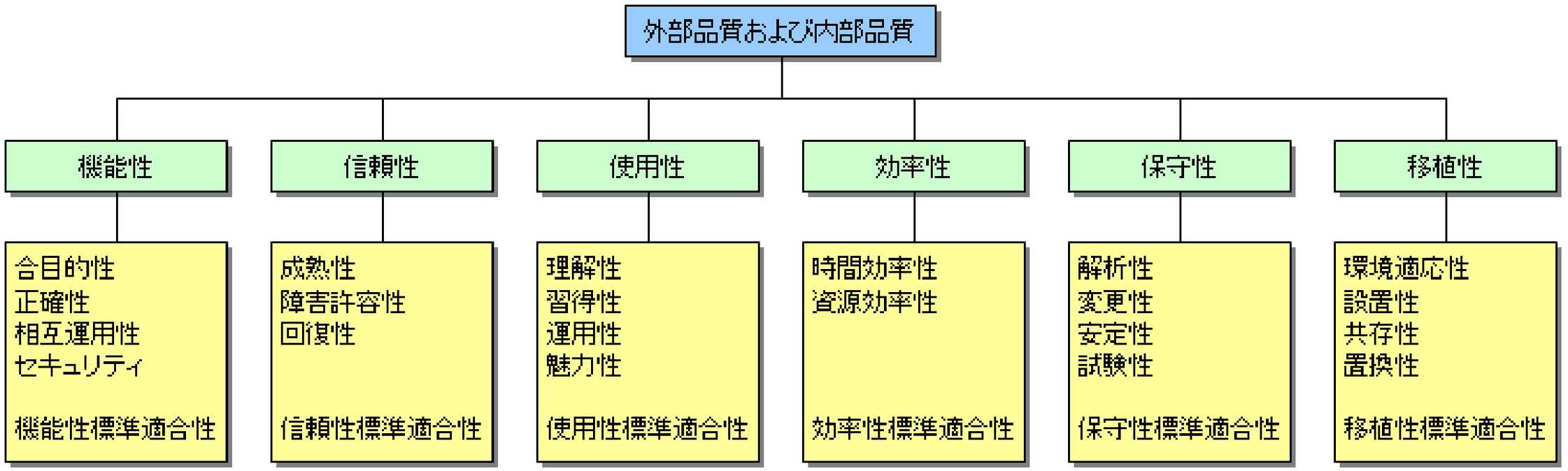
「使いやすさ」⊆「品質」？

ウィキペディアによると…

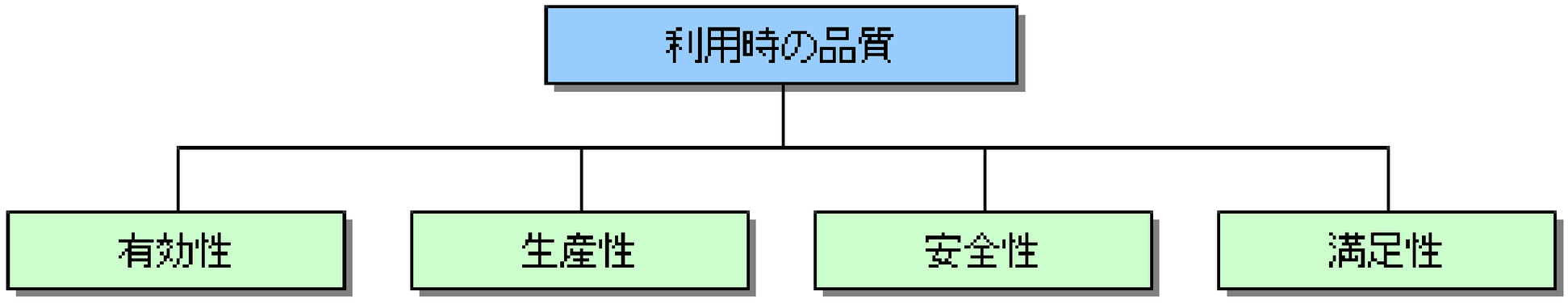
ソフトウェア品質（ソフトウェアひんしつ、英: Software quality）は、ソフトウェアの品質を指し、プログラマの観点からはソースコードの品質、エンドユーザーの観点からはアプリケーションソフトウェアの品質を意味する。

https://ja.wikipedia.org/wiki/%E3%82%BD%E3%83%95%E3%83%88%E3%82%A6%E3%82%A7%E3%82%A2%E5%93%81%E8%B3%AA

JIS X 0129-1 によると…



ISO 9126-1(JIS X 0129-1)から引用



ISO 9126-1(JIS X 0129-1)から引用

http://homepage3.nifty.com/kaku-chan/q_and_p/what_quality.html

利用時品質検討ワーキング・グループ」を発足

ソフトウェア高信頼化

「利用時品質検討ワーキング・グループ」を発足
～IoT時代の開発時に考慮すべき「利用時の品質」のポイントをとりまとめ～

2016年9月27日公開
独立行政法人情報処理推進機構
技術本部 ソフトウェア高信頼化センター

近年、自動車や家電などのさまざまなモノがインターネットに接続し、モノ同士が相互に接続する「IoT社会」が到来しています。IoT (※1) を活用することで、これまでにない付加価値の高い新たなサービスの提供が期待されています。一方で、製品が"つながる"ことによって、利用者や開発者が想定しない不具合や事故が発生するリスクもあります。"つながる"ことを想定した製品の安全・安心の確保は、IoT社会では最重要課題です。

このような背景を踏まえ、今後IoT製品やシステムを開発していくためには、利用者の目的や特性および利用環境を考慮した高い信頼性を確保できること（＝利用時の品質を考慮すること）が重要となります。そこで、IPA/SECは、これからのIoT製品やシステムの開発における利用時の品質の考え方・設計への反映プロセス・評価手法などを整理するとともに、安全・安心なIoT製品やシステムの実現に向けた開発の方向性を検討する活動を開始します。

利用時品質検討ワーキング・グループでは、「利用時の品質」に関する研究活動を行っている特定非営利活動法人人間中心設計推進機構（HCD-Net)とIPAが協業し、異なる立場の各業界の有識者からの意見を集約することで、IoT製品・サービスの開発に当たって分野横断的に活用できる観点として取りまとめていきます。

(*1) Internet of Things：モノのインターネット

ソフトウェア高信頼化
› 組織概要・委員会活動
› 事業内容のご案内
› 事業内容のご案内
› 事業に関連するお知らせ
› 報告書・出版物・ツール
› セミナー・イベント
› SEC journal
› メールマガジン・Twitter
› SECの公開情報を検索する（SWE iPedia）
› 利用者登録
› よくある質問と回答

「使いやすさ」⊆「品質」！

派生開発で
「使いやすさ」を
維持するためには
関係者全員で
レビューすることが大事！

全員でレビューするとは？

関係者全員でレビューを回すこと

- 追加機能が要求を満たしているだけではなく、使いやすさを損なっていないかどうかをレビューする
- 多様な視点でレビューすることで抜け、漏れを防ぐ
- 机上ではなく実物を操作することで開発者自身が使いやすさを実感する
- ソースコードの正当性や効率性については別途おこなうべき

全員でレビューするとは？

関係者全員でレビューを回すこと

この発表で紹介する
全員参加型レビューの次のステップとして
現在「GitHub Flow」に取り組み中

- 机上では[illegible]操作することで開発者自身が使いやすさを実感する

- ソースコードの正当性や効率性については別途おこなうべき

2
結人・束人開発の現状と問題点

結人・束人開発のいま

- 2008年から開発がスタートし、2016年で8年目
- 新機能を追加したり、既存機能を改善したり、いわゆる「派生開発」を継続中
- 参画メンバーの入れ替わりなどによりシステムの全貌をとらえにくくなっている
- 結人・束人を構成するコンポーネントは3つから7つに増加し、機能数も増え複雑なシステムに…

機能追加や機能変更するのが辛い…

不安なレビュー

- これで本当に大丈夫なのか？！
  - 機能追加による他の機能への影響を把握しずらくなってきた
- 形骸化しているかもしれない？！
  - チームの成熟か慣れによるものか突っ込んだ指摘が減ってきた

今までのレビュー方法

1. サブリーダーによるレビュー
2. 定期的な集合型レビュー
3. モンキーテスト大会
4. GUI自動テスト

サブリーダーによるレビュー

メリット

トップダウン的なレビューができる

デメリット

一部のメンバーの負荷が高い

単一的な視点でのレビューに陥りやすい

定期的な集合型レビュー

メリット

全員で同じテーマを話し合える

デメリット

メンバーの招集がうまくいかないことがある

人数が多いと時間がかかる

控えめな人の意見が聞きずらい

モンキーテスト大会

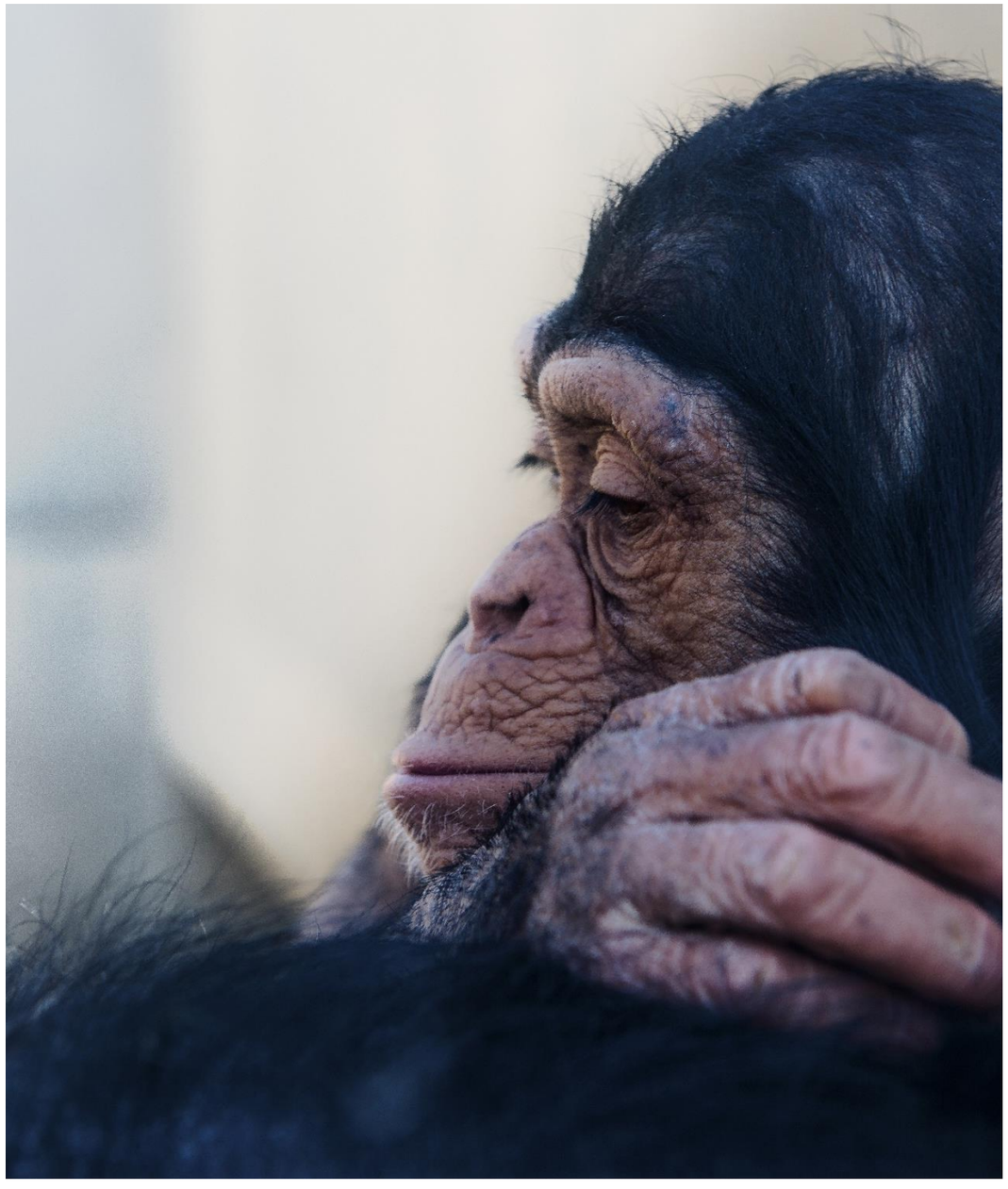
メリット

ユーザーの意図しない操作などの影響を検出しやすい

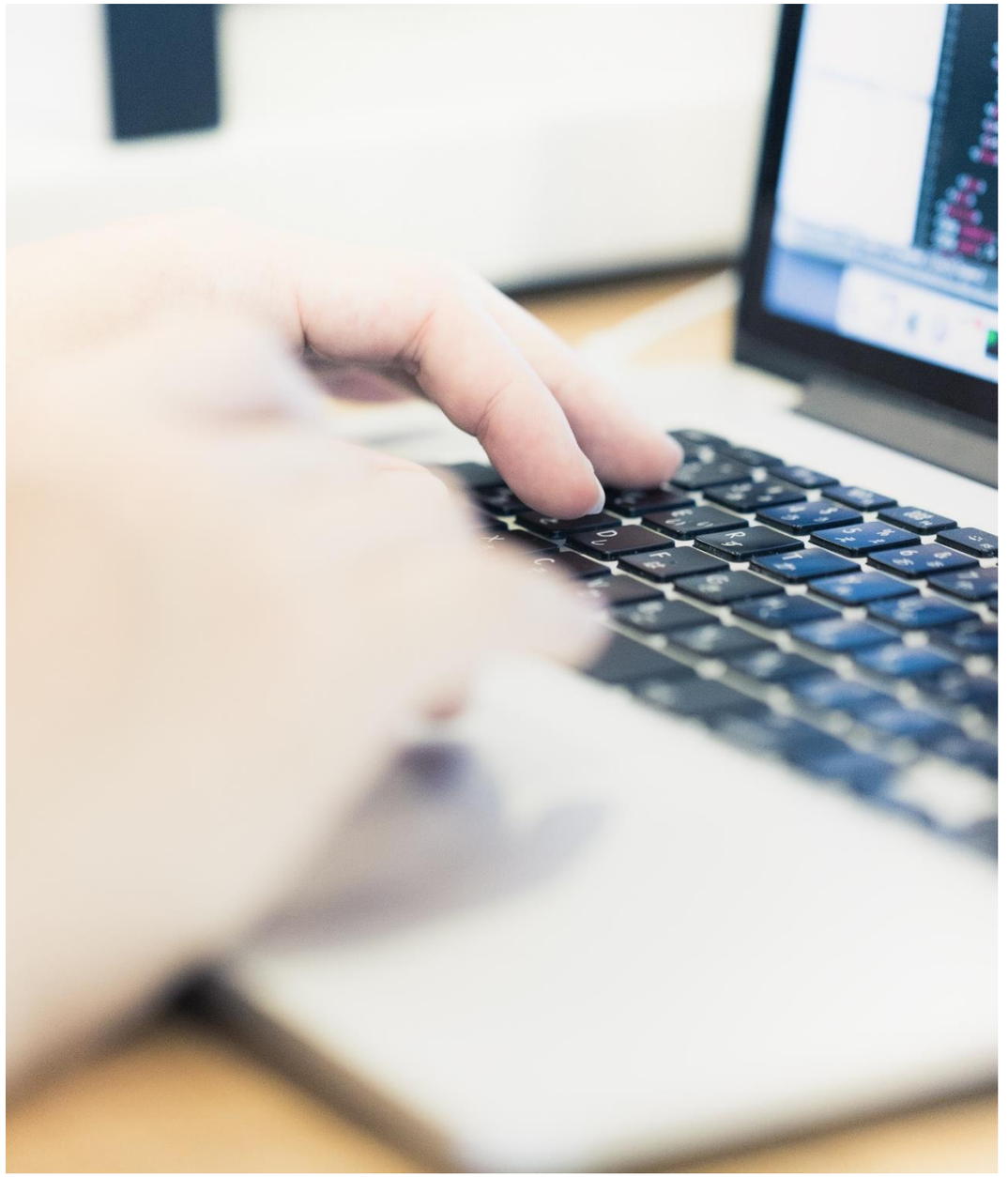
既存機能への影響が検出できる

デメリット

機能追加や変更に伴う妥当性は確認しずらい



GUI自動テスト



メリット

既存機能への影響が検出されやすい

デメリット

バージョンごとの管理が煩雑すぎる

どれも一長一短

クリアすべき課題

1. **機能の妥当性をレビューできること**

   - 既存の機能に影響がないか、使い勝手に変わりがないかを確認できることが求められる

2. **一部のメンバーにレビュー負荷がかかりすぎないこと**

   - チームが成熟し個々のメンバーの力が上がったにもかかわらず旧来のレビュー方法に戻るのは好ましくない

3. **既存の開発プロセスと親和性の高いこと**

   - 過去にはマインドマップやExcelなどで課題を管理したこともあったが年月の経過とともに活用されなくなってしまった

3

全員参加型レビューの紹介

前提・環境

- いつでもだれでもアクセスできるステージング環境が整備されている

- TracやRedmineなどのプロジェクト管理ツールを使用している

- Webアプリケーションの場合、標準ブラウザ(IEなど)を決めていること

手順

1. 担当者はTracなどのすでにあるプロジェクト管理システムのチケットに従ってタスクを実施

2. 担当者はチケットにレビューチェックシートを作成する

3. 朝会などでレビュー者をアサイン

   - 指名・立候補などなんでもOK

4. レビュー者はステージング環境を利用して成果物を実際に使ってみてレビューする

5. 担当者、レビュー者間でフィードバック＆改善..(以下繰り返し)

レビューチェックシート

- 基本的にはフリーフォーマットだが必要に応じてテンプレートを準備
  - テンプレートの内容は関係者全員で合意
- 以下の項目を含める
  - 追加された、あるいは変更された機能
  - 既存の機能に影響がないか
  - 操作手順やデータ

レビューチェックシート例（Trac）

```
    }
}
```

◆確認事項

- WFリクエストがページネーションされずにすべて表示されること
- 条件を満たすWFリクエストがある場合、
  - 一括承認ボタンが表示されること
    - 一括承認チェックボックスにチェックが入っている場合のみ押下可能
  - 一括承認対象選択用のカラム、およびチェックボックスが表示されること
  - 全選択用チェックボックス（ヘッダ行）をクリックするとすべてのリクエストにチェックがされること
- 条件を満たすWFリクエストがない場合、
  - 一括承認ボタンは表示されないこと
  - 一括承認対象選択用のカラムおよび、チェックボックスは表示されないこと
- 条件を満たすWFリクエストのみ承認できること
  - 承認された申請番号が実行順（申請番号の降順）に画面に表示されること
- 画面を表示した際に表示されていた承認ステップに対してのみ行えること
  - 画面を表示してから一括承認を行うまでの間に、他の人により承認されて認できないとする
- 正しくStorageに「承認」アクションが登録されていること

◆関連チケット

担当者(ビュー依頼者)は
確認すべき点を列挙

レビュー者がどのような状況を
見てOKと判断したか
後から確認できる

更新者: 999000_099064 (11ヵ月前)

レビューOK

以下確認

- WFリクエストがページネーションされずにすべて表示されること →○
- 条件を満たすWFリクエストがある場合、
  - 一括承認ボタンが表示されること→○（上下に表示）
    - 一括承認チェックボックスにチェックが入っている場合のみ押下可能　→○
  - 一括承認対象選択用のカラム、およびチェックボックスが表示されること→○
  - 全選択用チェックボックス（ヘッダ行）をクリックするとすべてのリクエストにチェされること→○
- 条件を満たすWFリクエストがない場合、
  - 一括承認ボタンは表示されないこと　→○（データ0件、1件で確認）
  - 一括承認対象選択用のカラムおよび、チェックボックスは表示されないこと　→○
- 条件を満たすWFリクエストのみ承認できること　→○
  - 承認された申請番号が実行順（申請番号の降順）に画面に表示されること　→○
- 画面を表示した際に表示されていた承認ステップに対してのみ行えること　→○
  - 画面を表示してから一括承認を行うまでの間に、他の人により承認されている場合を承認できないとする→○（エラー確認）
- 正しくStorageに「承認」アクションが登録されていること　→　○　（履歴で確認）

その他：課題

- 画面上でのソートについて考える
- 対象がある場合とない場合でレイアウトが違うのは違和感がないのか考える

レビューチェックシート例（GitBucket）

999000_105119 commented on 10 Dec

ゴール

- WebEngineで定義されているモデルをPDF出力コード中で呼び出せるようにする
- WebEngineはCIサーバに構築されたgemサーバから取得する
- Storageに接続するためのAPIキーはDesignerのものを借用する
  - 最終的には[redacted]PDFツール用のAPIキーを設定する
- StorageのURLはIDM_HOMEの設定から取得する

注意点（Macで実行する場合）

- jrubyのproxy読んでしまう不具合を回避するバッチファイルをpdf_generator/config/initializerlに配置する
  - http://trac.mms.intec.co.jp/gitbucket/999000_110101/jruby_proxy_patch

バッチプログラムの場合は
実行手順・環境を提示

レビュー者の確認手順が
妥当かどうかも
あとから確認できる

999000_102004 commented on 10 Dec

確認しました。

確認手順

- JRubyインストール(済み)
- トピックブランチチェックアウト
- bundle install
- IDM_HOME設定
  - Storage接続先確認
  - ApiClient(パスワード)確認
  - license.json配置確認
- `jruby -S rake pdf:kansa`

確認結果＆コメント

- [x] PDF出ました。
- [x] Storageから取得した値を出力できました。
- [x] WebEngine自体で依存解決できていない分を別途Gemfilelに定義している件、了解しました。

FAQ

よくある質問

新人でもレビューできるの？

できます。

必要なスキルはほんの少しの「業務知識とITスキル」です。

新人のピュアな観点でレビューすることにより、担当者の思い込みや新たな顧客ニーズの気づきにつながることがあります。また、レビューすることにより業務への理解が深まります。

不安な場合は他メンバーにてダブルチェックを実施します。

どんなプロジェクトで実践できるの？

画面操作を含むWebアプリケーションやコマンドラインで実行するバッチプログラムなどどんなプロジェクトにも適用できます。

実際に以下のプロジェクトで実践しました。

- 結人・束人
- R様向けPDFレポート出力ツール

すべての指摘事項を受け入れるべき？

いいえ。

万人が満足するような完全なものを作ることはできませんし、作ろうとすべきではありません。

影響度や優先度などプロジェクトの状況に応じてやるやらないを判断することが好ましいです。(最終的には顧客が決めること)

効果

Effect

レビューノウハウを共有しながらレビューできる

⇒レビュー時のやり取りや手順を記録として蓄積できる

相談や質問など、成果物を指さしながらコミュニケーションできる

⇒担当者、レビュー者自身が使いやすさを体験できる

多様な視点でレビューできる

⇒担当者の思い込み、顧客ニーズへの気づきにつながる

4

「使いやすさを維持するための品質向上プロセス」への第一歩

事業を継続するためにも
私たちにとって顧客に
製品、サービス、システムを
長く使い続けてもらうことはとても大事

派生開発

顧客の要望、市場の変化、時代の変化に応じて開発し続ける必要がある
（＝新しい価値を提供し続ける）

ユーザー視点に立って品質を作り込むことができれば、
顧客満足につながる
↓
顧客の私たちへの信頼が上がる
↓
私たちの製品・サービス・システムを長く使ってもらえる

まとめ

- ✓ **全員参加型レビューは多様な視点でレビューできる！**
  - 問題点の早期発見につながり、品質・生産性向上
  - 新たな顧客ニーズの気づきにつながり、顧客満足度向上
- ✓ **メンバー全員が満遍なく機能に触れることができる！**
  - メンバー全体のスキル向上

ご清聴ありがとうございました。

補足(レビューの心得)

- 誰もが平等
- 全員参加
- 問題解決を避ける
  - 創造的思考
  - 分析的思考→まずはこっち(情報を取り込んで比較)
- 決定を急がない

「みんなではじめるデザイン批評」
https://www.amazon.co.jp/dp/4802510209

補足(人はシステムを使うときメンタルモデルを作る)

- メンタルモデル＝ある物事が機能している仕組みをその人がどう理解しているかを表現したもの。

- システムがメンタルモデルにない反応をすると「**使いづらい**」と感じる

- 人は常にメンタルモデルを持っている

- 人は過去の経験に基いてメンタルモデルを構築する

- 全ての人が同じメンタルモデルを持つわけではない

インタフェースデザインの心理学
https://www.amazon.co.jp/dp/4873115574