PG・UT作業の完了条件チェックリスト

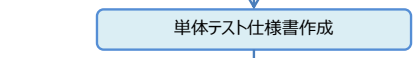
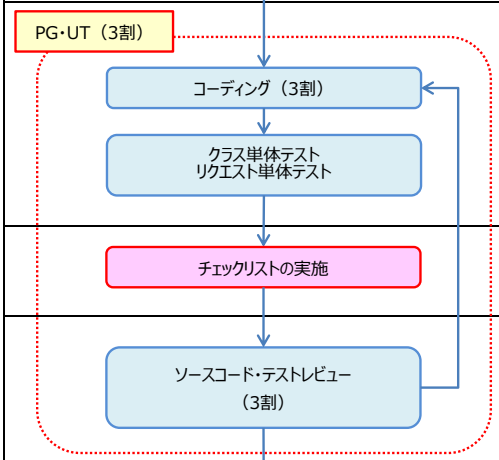
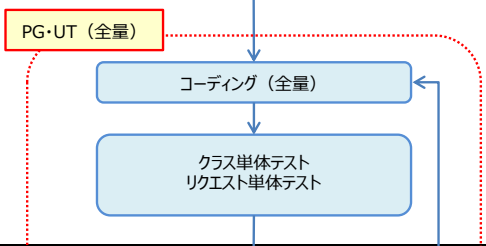
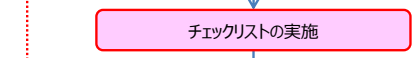
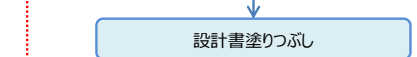
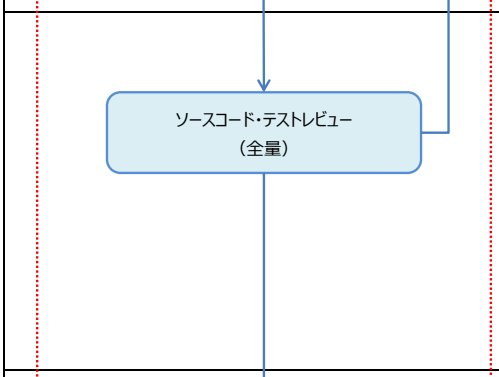
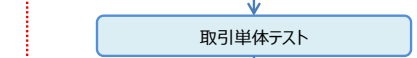
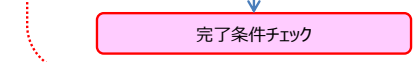
第1.0版

2019年3月26日

【導入事例】

「PG･UT作業の完了条件チェックリスト」や「プログラマー向け成果物セルフチェックリスト」を導入した開発プロセスの事例を以下に示す。

◎前提
- オフショアに開発委託する案件のため、委託の際の条件として、「PG･UT作業の完了条件チェックリスト」を完了条件とした。
- Nablarch未経験者やJavaスキルが高くないメンバもいたため、そのメンバに対しては３割完成時点でレビューを実施した。
- 設計書の内容を漏れなく･誤りなくコーディングできていることを確認するために、コーディング後に設計書を塗りつぶして網羅的にチェックをした。

作業フロー	作業内容と作業観点	担当
	担当者が機能･取引の把握を行う。	担当者
	単体テスト標準に準拠したケースであること。 設計書の内容を網羅するテストケースが作成されていること。	担当者
	読込んだ設計書を元に数リクエスト分のコーディングと単体テストを行う。 どこまでで3割レビューを依頼するか、事前に担当者と技術統括間ですり合わせをしておくこと。 PG･UT（3割）は取引の難易度や、担当者のスキルによっては省略可とする。	担当者
	コーディングと単体テストの成果物に対して以下のチェックリストを実施する。 **･プログラマー向け成果物セルフチェックリスト**	担当者
	取引の3割の完成度時点でソースコード･テストのレビューを行う。 全量完成前にレビューを行うことで、大きな手戻りを防止する。 例：取引に初期表示･一覧検索･登録･確認･完了の5リクエストがあれば、初期表示と一覧検索完了時点でのレビュー、等。	技術統括
	残りのコーディングと単体テストを行う。	担当者
	コーディングと単体テストの成果物に対して以下のチェックリストを実施する。 **･プログラマー向け成果物セルフチェックリスト**	担当者
	ソースコードレビュー前に設計書の塗りつぶしチェックを行う。 設計書に記載されている仕様（画面レイアウト含む）が、漏れなく･間違いなく実装されているか確認する。	担当者
	■技術観点レビュー 以下の技術観点例を参考に、成果物のレビューを実施する。 ･各種標準への準拠（Javaコーディング規約、UI標準、テスト標準） ･よくあるコーディングミス（BigDecimal、NullPoint、添付ファイル） ･共通化の妥当性 ･ソースコードの可読性 ･アプリケーションフレームワーク機能の利用方法 ･アプリケーション性能／セキュリティ観点の考慮 ･テストの過不足 ■業務観点レビュー 設計書どおり実装されているかの観点で、設計書とソースコードの突合チェックを行う。レビューアの工数が確保できない場合は別担当者が実施(突合チェックは第3者の視点で行うことが重要であるため。)	技術統括
	アプリケーションをアプリケーションサーバにデプロイし、画面打鍵を実施する。 ･表示制御、入力制御、js、取引内の画面遷移を確認。	担当者
	プロジェクトで定めている、PG･UT完了条件をチェックする。 **･PG･UT作業の完了条件チェックリスト**	担当者

PG･UT作業の完了条件チェックリスト

取引ID：
取引名：

1. 目視によるチェック

No	チェック項目	チェック内容	チェック方法	評価指標	指標値	確認者	確認日	確認結果	コメント
1	コーディング・単体テスト	開発で起きやすいコーディングミスがないこと。	「プログラマー向け成果物セルフチェックリスト」を使用してチェックする。	NG件数	0件				
2	実装漏れ	外部設計書と内部設計書のすべての記載が漏れなく実装できていること。	外部設計書と内部設計書に対して、ソースコードに実装済みの箇所を塗りつぶすことでチェックする。	塗りつぶされていない箇所	0個				

2. ツールによるチェック

No	チェック項目	チェック内容	チェック方法	評価指標	指標値	確認者	確認日	確認結果	コメント
1	TODOコメント	ソースコードと設定ファイルにTODOコメントが残っていないこと。	ソースコードと設定ファイルにTODOコメントが残っていないことをチェックする。 ・Java ・jsp、sql、xml、config、properties、js	残件数	0件				
2	Javaソースコード	一般的に推奨されるJavaソースコードの記述方法に違反していないこと。	静的解析ツールやIDE（統合開発環境）の機能でチェックする。	エラー数・警告数	0件				
3	Javaコンパイル	Javaソースコードがコンパイルできること。	JDKのコンパイル機能やIDEの機能を使用してチェックする。	エラー数・警告数	0件				
4	@Ignoreアノテーション （@Ignoreは単体テスト実行を無視するJUnitのアノテーション）	@Ignoreアノテーションを使用していないこと。	@Ignoreの付与されたテストが残っていないことをチェックする。	使用箇所	0件				
5	自動単体テスト	テストコードがすべて合格すること。	IDEの機能でテストコードを実行してチェックする。	NG件数	0件				
6	自動単体テストのカバレッジ	テストコードがテスト対象のソースコードをすべてカバーしていること。	IDEの機能でテストカバレッジを測定してチェックする。	テストカバレッジ（C0とC1）	100% ※１				

※１　try-with-resources を使った実装やSQLExceptionが発生した場合の処理など、自動テストで実施が難しい箇所は対象外とする。

3. レビュー指摘対応のチェック

No	チェック項目	チェック内容	チェック方法	評価指標	指標値	確認者	確認日	確認結果	コメント
1	レビュー完了率	レビュー対象成果物の全量がレビューされていること。	レビュー対象成果物のレビュー状況をチェックする。	完了率	100%				
2	ソースコードレビュー指摘	レビュー指摘にすべて対応できていること。	レビュー指摘の対応状況をチェックする。	残件数	0件				
3	単体テスト仕様＆結果レビュー指摘	レビュー指摘にすべて対応できていること。	レビュー指摘の対応状況をチェックする。	残件数	0件				

4. その他のチェック

No	チェック項目	チェック内容	チェック方法	評価指標	指標値	確認者	確認日	確認結果	コメント
1	開発残（積み残し）	開発残（積み残し）がないこと。	上記1～3のチェック内容を確認し、チェックする。 積み残しが存在する場合は、後続工程の担当チームに連絡できていること。 ①積み残し内容 ②積み残しによる制限事項 ③対処予定日	残件数	0件				