製造工程に対する工程完了判定の導入
「何をつくるか」から「どうつくるか」へ品質保証の拡大

Implement process completion judgment for process of development phase
～Expansion of quality assurance from what to make to how to make～

藤井和弘
Kazuhiro.fujii.dd@ins.orix.jp
オリックス生命保険株式会社　IT 業務管理部

発表要旨：
当社では、ウォーターフォール型のシステム開発案件の品質保証のため、2020 年 1 月より工程完了判定を導入し、上流工程の要件定義書、設計書への品質保証を開始することで、成果として「対前年度比での障害件数の削減目標」を達成した。

ただし、課題として、製造工程は完了判定対象外としており、製造工程で埋め込む欠陥数の削減や、ソースコードに起因する本番稼働開始後の保守性への対応は、積み残しの課題となっていた。
このため、引き続き製造工程にも完了判定を導入し、ソースコードの品質の向上にも取り組むこととした。

工程完了判定は、多くの企業で導入されており、オーソドックスな品質保証の活動であるが、当社はユーザー企業であり、製造工程を外部委託する場合も多いため、判定を導入するにあたり、担当者のスキルや保有する設計情報にばらつきがあるなど考慮すべき制約があった。

私たちと同じような課題や制約を抱えている参加者に対して、導入で工夫したポイント、苦労した内容および、見込んでいる導入効果を説明する。

キーワード：

製造工程の完了判定

想定している聴衆

ユーザー企業の品質保証担当

発表者の紹介（全角１００文字）：

オリックス生命保険株式会社にてソフトウェアの品質管理を担当しております藤井です。
初めて本シンポジウムで経験発表をさせていただきます。よろしくお願いします。

製造工程に対する工程完了判定の導入
「何をつくるか」から「どうつくるか」へ品質保証の拡大

オリックス生命保険株式会社
IT業務管理部
藤井　和弘
kazuhiro.fujii.dd@ins.orix.jp

ほかにはない
アンサーを。

目次

1. 背景と課題 – I. 背景

当社では、2020年1月から、要件定義、外部設計、単体テスト、結合テスト、システムテスト工程の完了時に品質判定を導入し、ソフトウェアの品質向上に取り組んでいる。

ただし、製造工程の完了判定は、制約により実施できておらず、ソースコードの品質管理は未対応となっていた。

制約条件

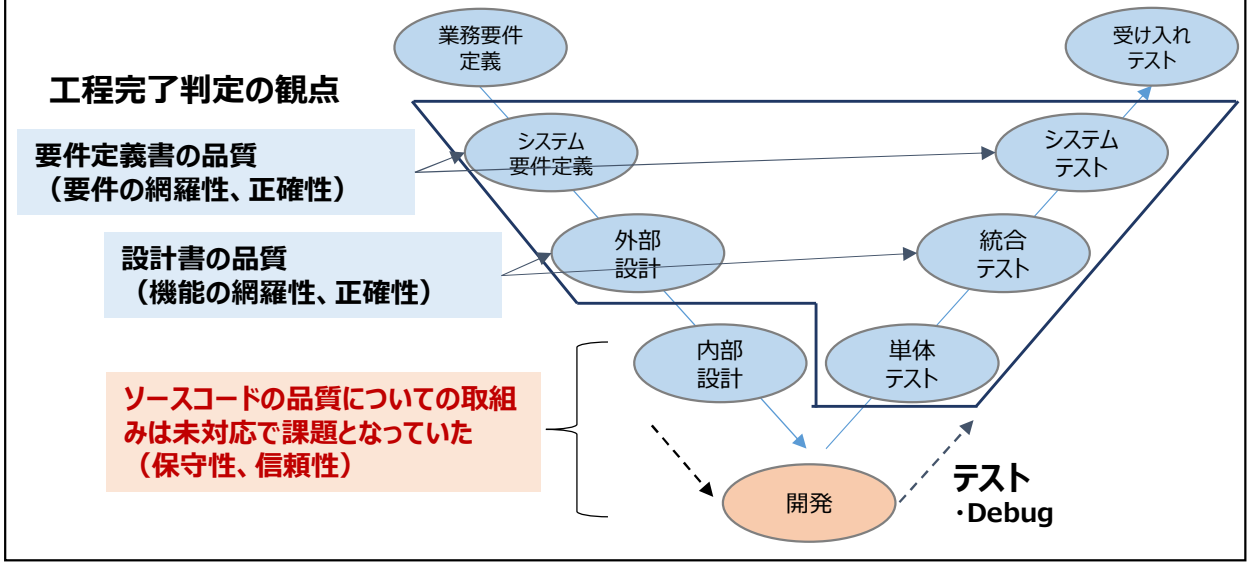
スキルの制約

ソースコードの品質を作り込むための考え方や観点が、開発担当者間でばらつきがある

設計情報の制約

委託開発やパッケージ利用などで内部設計情報やソースコードが開示されていないソフトウェアがあり、積極的にソースコードの品質の作り込みに関与できていない場合がある

現行の工程完了判定の対象工程



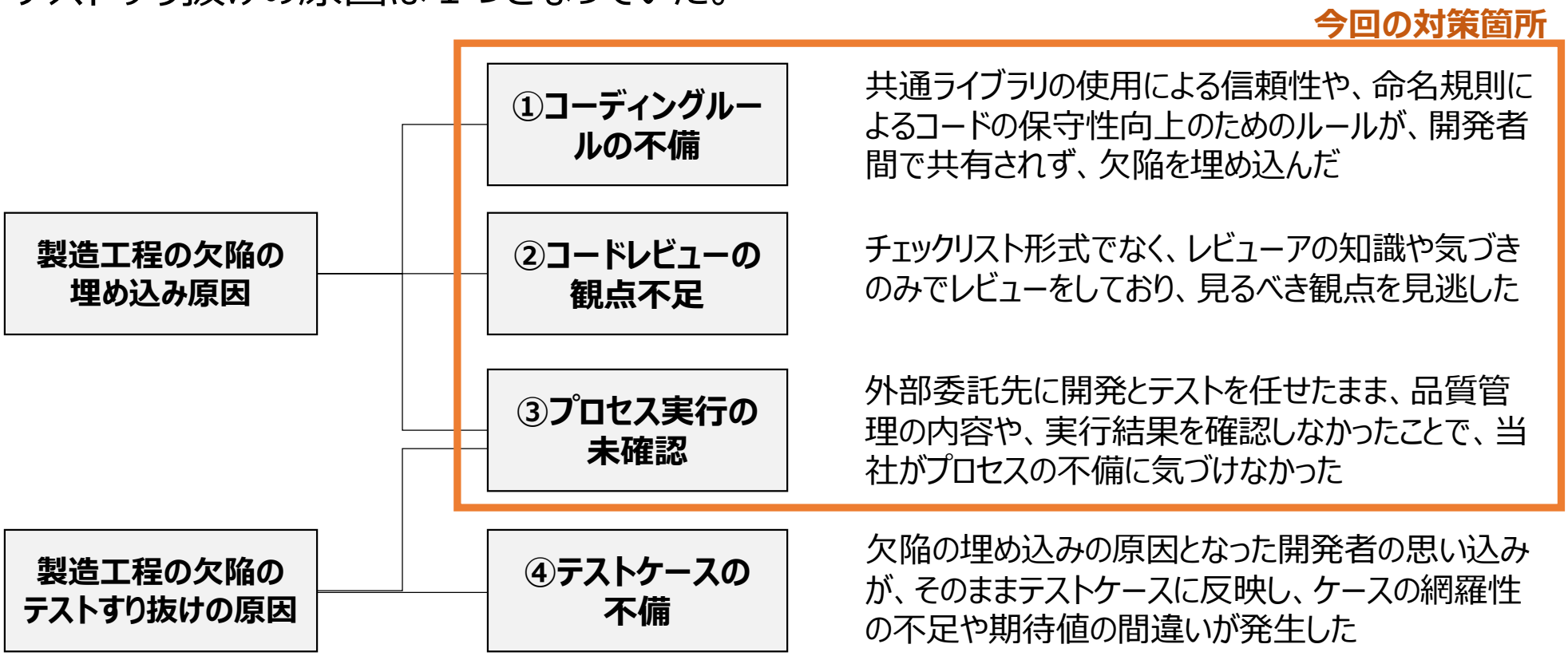
1. 背景と課題 – II. ソースコードの品質向上についてのこれまでの取組み

- ✓ 当社では、委託開発と内製化の２種類の開発パターンがあり、委託開発案件ではソースコードの品質に関する取組みは、案件ごとに異なっている。
- ✓ 案件の種類に関わらず、求める品質を確保するための取組みを必要としていた。

	取組みを継続		取組みの見える化が必要
ソースコードの品質向上の取組み	内製化案件		委託開発案件
コーディングルールの整備	◯ 対応済み	⇔	△開発委託先の対応内容に準じる
ソースコードレビュー	◯ チェックリスト形式によるレビュー	⇔	
静的解析ツールによるコード解析	◯ 実行中	⇔	

1. 背景と課題 – III. 製造工程での欠陥の埋め込みとテストすり抜けの代表的な原因

✓ 当社での過去の障害を確認すると、製造工程で欠陥を埋め込む代表的な原因は３つ、テストすり抜けの原因は１つとなっていた。



2. 取組みの内容と目的

✓ 要件定義、設計に引き続き製造工程においても完了判定を導入し、ソースコードの品質向上に取り組むことで、欠陥の埋め込みによる信頼性の低下やソースコードの保守性の低下を防ぐこととした

目的

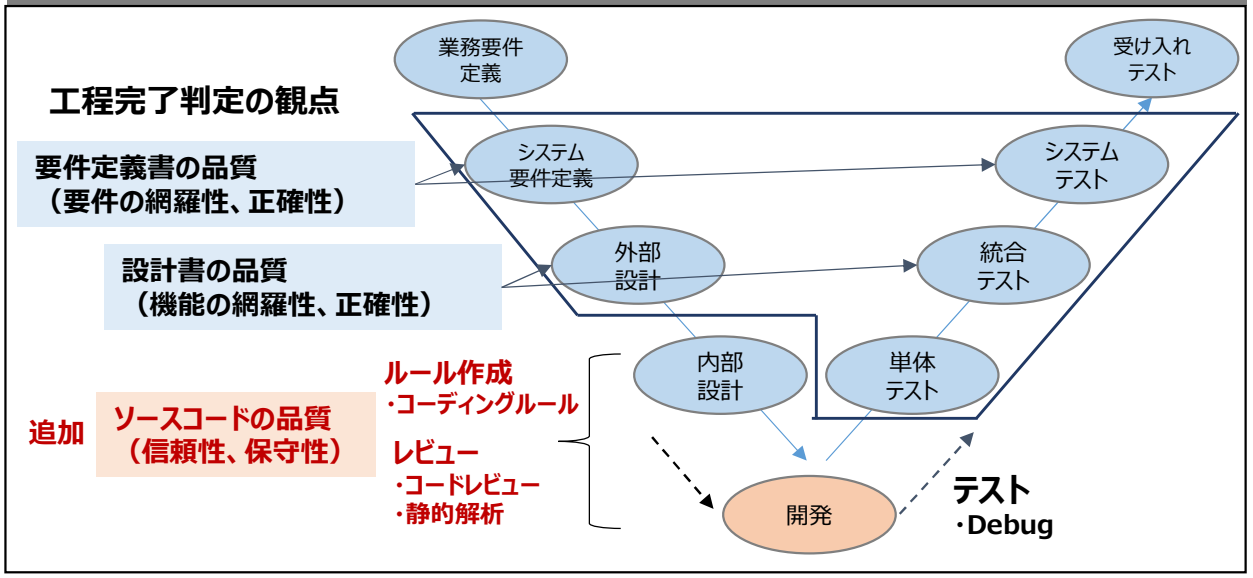
欠陥の埋め込みによる信頼性の低下を防ぐ

欠陥の早期発見とすり抜けによるテスト工程での手戻りを予防する

技術的負債による保守性の低下を防ぐ

作成者以外が読み取れなかったり、書き方に一貫性のないソースコードの作り込みを防ぎ、テスト工程では解消できない保守性の低下を予防する

追加する工程完了判定の対象工程

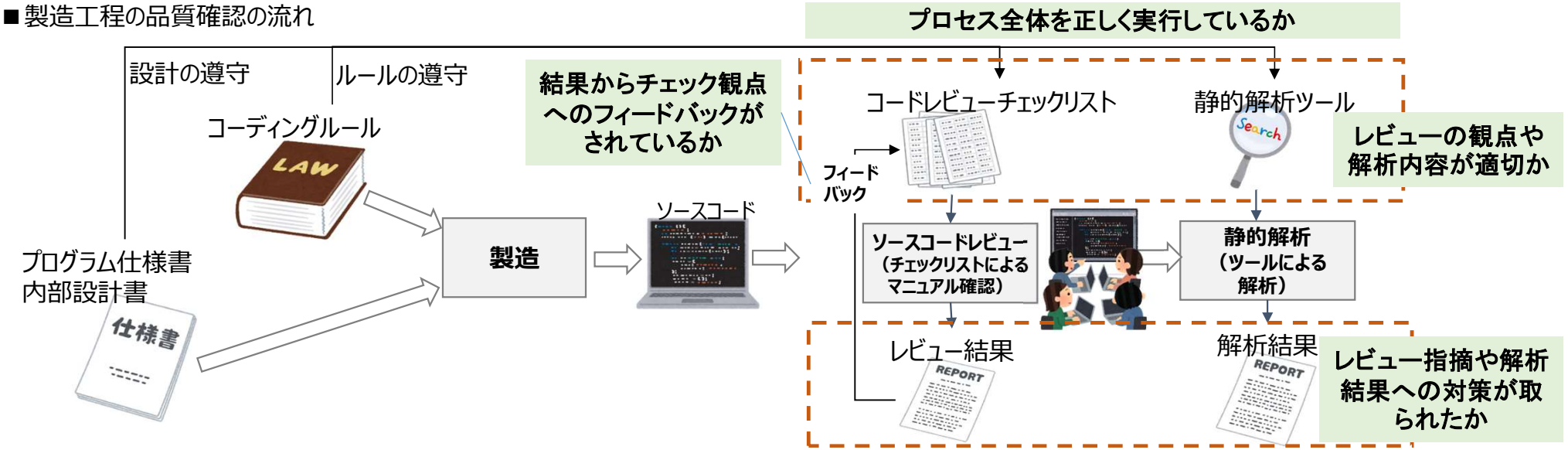


3. 実施概要－Ⅰ. 判定の観点

開発担当者は、自己評価結果と作成した成果物をを品質保証部門に所属する判定者へ提出する。判定者は、次の観点を確認し、判定結果を開発担当者へ連絡する。

- ✓ **コーディングルールが作成され、チェックリスト形式によるソースコードレビューが実施されているか、ツールによる静的解析が正しく実施されているか**（プロセスの確認：プロセス実行未確認への対応、コーディングルールの不備）
- ✓ **ソースコードレビューでの指摘やツールによる静的解析結果へ対策が取られているか、新たに発見したチェック観点を次のレビューに横展開できているか**（プロセスの確認：プロセス実行未確認への対応、コードレビューの観点不足への対応）
- ✓ **ソースコードレビューのチェック観点や静的解析の解析内容が適切か**（プロダクトの確認：観点不足への対応）

■製造工程の品質確認の流れ



3. 実施概要 – II. 判定の形式と観点の詳細

✓ 開発担当者は、**チェックリストへの回答**で自己評価を行う。チェックリストは**プロセスの確認**と**プロダクトの確認**の観点に分かれる

■プロセス確認用チェックリストの観点の一部を抜粋

	No	評価基準	自己評価欄	
			チェック	コメント
工程完了判定チェックシート ※チェックリストから一部引用	1	（製造工程の品質を理解するための前提条件：品質を作り込む対象を把握する） □製造するモジュール（コンポーネント）一覧とその役割や機能を確認していること ※工程完了判定用のフォルダに、上記が分かる資料を格納していること	適	
	2	（製造工程の品質を理解するための前提条件：モジュール（コンポーネント）の品質リスクを把握し、効率的なレビューを行う） □製造するモジュール（コンポーネント）の流用度合いを確認していること 「①全く新しく作るもの」、「②他社実績があったり、委託先の共通ライブラリなど既存のプログラムを一部修正して流用するもの」、「③他社実績があったり、委託先の共通ライブラリなど既存のプログラムを完全流用するもの」を分類し、品質リスクが高いコンポーネントを識別していること	適	
	3	（製造工程の品質を理解するための前提条件：品質を作り込む対象を把握する） □アプリケーションの論理構成を確認していること	適	
	4	（製造工程の品質を作り込むためのルールとプロセス コーディング規約） □コーディング規約やルールを作成していること ※工程完了判定用のフォルダに、上記が分かる資料を格納していること	適	
	5	（製造工程の品質を作り込むためのルールとプロセス ソースコードレビュー） □コードレビューチェックリストを作成していること ※工程完了判定用のフォルダに、上記が分かる資料を格納していること	適	
	6	（製造工程の品質を作り込むためのルールとプロセス ソースコードレビュー） □過去の本番障害事例からコードレビューチェックリストの改善を図っていること	適	
	7	（製造工程の品質を作り込むためのルールとプロセス ソースコードレビュー） □ソースコードレビューを計画した全モジュール（コンポーネント）に対して実施していること	適	
	8	（製造工程の品質を作り込むためのルールとプロセス 静的解析） □アプリケーションのレイヤや確認観点別に静的解析で使用するツールを選択していること 例：JavaScript – ESlint、Java – Spotbugs、Checkstyle 管理ツール–SonarQube など	対象外	規模が小さい為、ツールによる解析は実施しない
	9	（製造工程の品質を作り込むためのルールとプロセス 静的解析） □静的解析をどのタイミングで実行するか確認できていること	適	
	10	（製造工程の品質を作り込むためのルールとプロセス 静的解析） □プログラムの品質を保つために、ソースコードレビューで検証する観点、静的解析ツールで検証する観点が切り分けられており、明確になっていること	適	
	11	（製造工程の品質を作り込むためのルールとプロセス 静的解析） □実施を予定していた静的解析が完了していること	適	

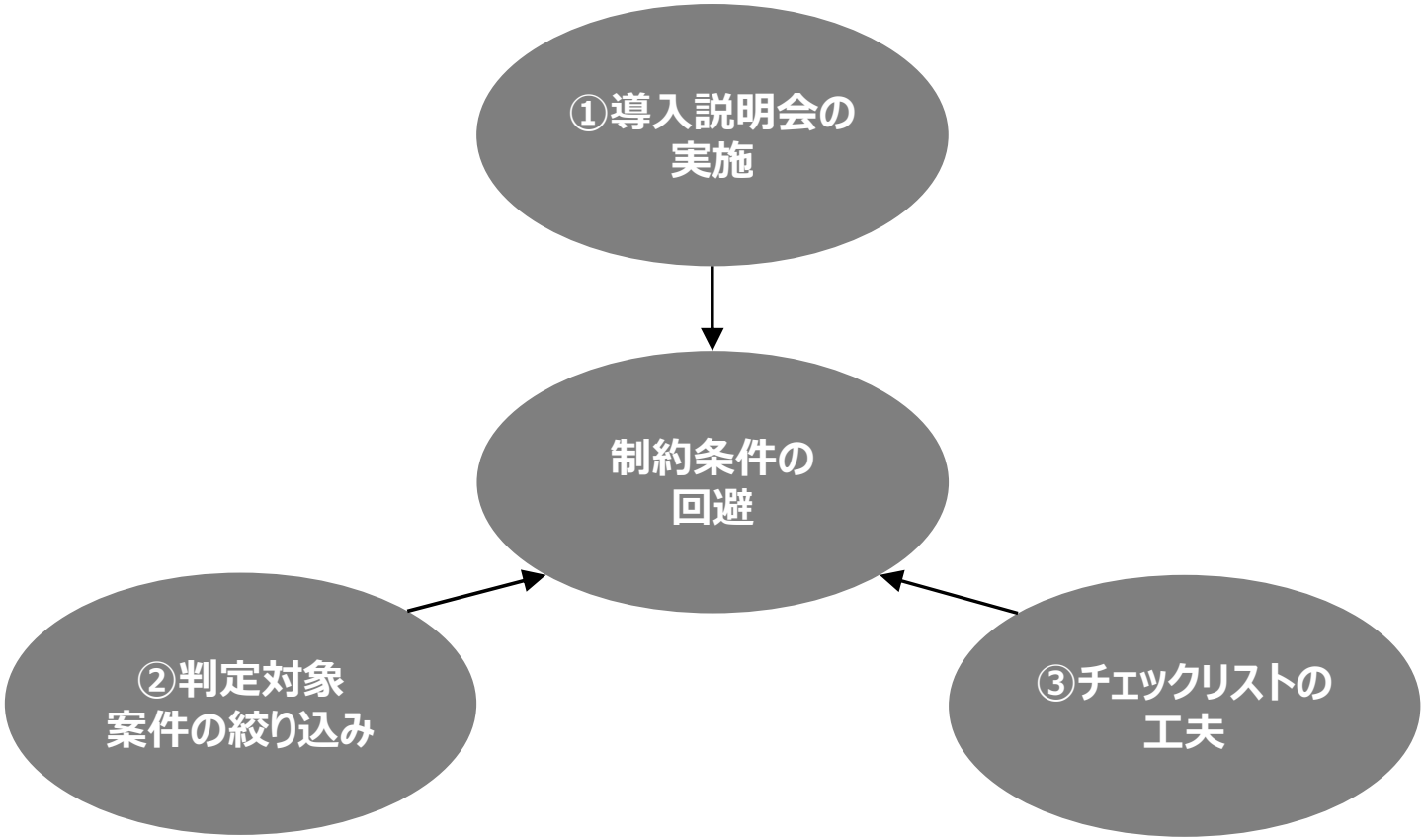
3. 実施概要 – II. 判定の形式と観点の詳細

■プロダクトの確認用チェックリストの観点の一部を抜粋

	分類		No	評価基準	自己評価欄	
					チェック	コメント
ITサービス品質チェックシート ※チェックリストから一部引用	構造	ソースコードの信頼性	1	・コードは、設計を完全に正しく実装しているか？	適	
		ソースコードの保守性	2	・コードは、適切に構造化され、スタイルと形式が規約や標準に従った一貫性をもっているか？	適	
			3	・ コード内にスタブやテストルーチンが残っていないか？	適	
			4	・ 外部の再利用可能なモジュール（コンポーネント）またはライブラリ関数を呼ぶことで置換できるコードは存在しないか？	適	
			5	・ １つのプロシージャにまとめられる反復的なコードブロックは存在しないか？	適	
			6	・ 定数または文字列定数は、ハードコーディングせずに、シンボルを使用しているか？（マジックナンバー）	適	
	文書化	ソースコードの保守性	7	・コードは、開発者以外が保守しやすいように、明確かつ適切にコメントを記載しているか？	適	
			8	・すべてのコメントは、規約や標準に従い、一定のルールで一貫性をもって記載されているか？	適	
	変数	ソースコードの信頼性	9	変数のライフサイクルでの不正を検出し、初期化忘れ、破棄忘れ、未使用などを検出しているか（データフロー解析） define： 変数を宣言、定義、または初期化する。 use：計算または判定述語で、変数を使用すなわち読み込む。 kill： 変数を削除、破棄する、または変数がスコープ外になる。	適	
		ソースコードの保守性	10	・すべての変数は、規約や標準に従い、意味のある一貫した明確な名前を使用して、適切に定義されているか？（命名規約）	適	
	算術演算	ソースコードの信頼性	11	・算術演算について下記のような誤差を予防する確認をしているか コードは、浮動小数点数の等号による比較を避けているか？ コードは、体系的に丸め誤差を防いでいるか？ コードは、桁数が大きく異なる数字の加算および減算を避けているか？ 除数がゼロまたは無効値のテストをしているか？	適	
	ループとブランチ	ソースコードの信頼性	12	・すべてのループ、ブランチ、および論理構造は、完全で、正しくて、また適切にネストしているか？（制御フロー解析）	適	
			13	・IF-ELSEIF チェーンの最初で分岐するのは最も頻度の高いパターンとなっているか？	適	
			14	・ループの終了条件は明白で必ず成立するか？	適	
		ソースコードの保守性	15	・ループ内のステートメントで、ループ外に出せるものはないか？	適	
		ソースコードの信頼性	16	・外部デバイスのアクセスで、タイムアウトまたはエラートラップを使用しているか？	適	
			17	・ファイルにアクセスする前に、存在していることをチェックしているか？	適	
			18	・プログラムの終了時に、すべてのファイルとデバイスを正しい状態にしているか？	適	
	セキュリティ	セキュリティ	19	機密情報の取扱いについて、要件定義を受けてコーディングにあたってのルールが決まっており、、コードレビューで実装を確認しているか　例）機密情報はプレーンテキストで格納しない、プログラムで使い終わった機密情報はディスクに保管しない、再利用可能な動的メモリに格納された機密情報は消去する等	適	

3. 実施概要 – III.導入にあたり工夫したこと

✓ 製造工程の完了判定を導入するにあたり、制約条件を回避するための対応を実施した。



3. 実施概要 – III.導入にあたり工夫したこと

① 導入説明会の実施 （スキルの制約への対応）1/2

➢ 実施目的

• **開発担当者へのスキル面のフォローアップ**	スキルの制約への対応。開発の経験が少ない担当者向けに疑問点を解消するため
• **効果的な判定のための意見交換**	開発担当者の考えをヒアリングし、対象案件の範囲やチェックリストの観点のレベル感などを検討するため

➢ 説明内容

• **製造工程の品質作り込みの一般的な方法**	チェックリストによるソースコードレビュー、ツールによる静的解析の説明
• **製造工程の品質保証の目的や期待する成果、信頼性、保守性の向上の効果**	製造工程の完了判定を導入することによる開発担当者にとっての利点の説明
• **開発担当者に過度に負担をかけない工夫**	効果的に判定するための対象案件の絞り込みやチェック観点の内容について、説明会時点でのアイデアを説明

3. 実施概要 – III.導入にあたり工夫したこと

① 導入説明会の実施（スキルの制約への対応）2/2

➢ 開発担当者からの質問

スキルの制約や設計情報の制約に関する質問が集中した

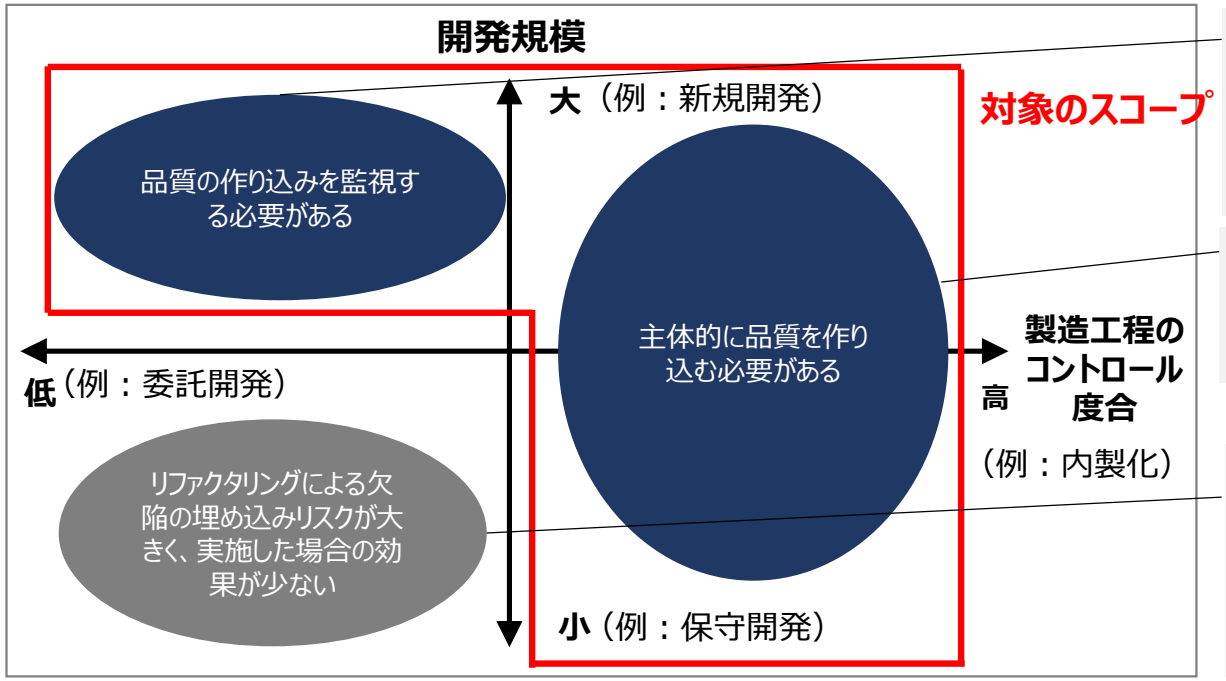
• **設計情報の不備について（設計情報の制約）**	稼働中のソフトウェアでは、委託先との請負契約で内部設計情報が開示されていないものがあり、そもそも工程完了判定のために必要な情報が得られない。
• **チェック観点のレベル感について（スキルの制約への対応）**	ソースコードの品質を作り込むために必要なことやコーディングルールでチェックすべき観点を理解していない。活動を推進するために品質の作り込み方やチェックの観点の解説についてフォローアップしてほしい。
• **既存システムに対する影響について**	稼働中のソフトウェアに対する静的解析ツールの実行結果やソースコードレビューの結果をうけて、コードを改修しても、得られる成果より、新たな欠陥を埋め込むリスクのほうが大きいのではないか。また、現状、安定稼働しているソフトウェアに対してこのような改修をするコストを捻出するのは容易ではない。

3. 実施概要 – III.導入にあたり工夫したこと

② 判定対象案件の絞り込み（既存システムに対する影響や設計情報の不備への対応）

製造工程の完了判定の対象は、新規開発案件と内製化案件を対象とした

製造工程の工程完了判定の対象



対象とした理由

- **新規開発案件**

開発規模が大きくソースコードレビューやツールによる静的解析の必要性があり、はじめから無理なく実行結果をコードにフィードバック可能な「新規開発案件」を対象とした

- **内製化案件**

製造工程のコントロール度合いが高く、自ら品質を作り込む必要がある「内製化案件」を対象とした

対象外とした理由

- **その他（稼働中の内製化以外のシステム）**

ソースコードレビューや静的解析結果の対応に必要なコストおよび、リファクタリングで欠陥を埋め込むリスクが保守性の向上で得られる欠陥予防のメリットを上回ると判断した

3. 実施概要 – III.導入にあたり工夫したこと

③ 判定対象案件の絞り込み（チェック観点のレベル感への対応）1/2

判定のチェック項目は、データフローや制御フローおよび命名規則など、**開発言語に共通する抽象度の高い項目を採用した。**

プロダクトの確認チェック項目例 ※チェックリストから一部引用

分類		No	評価基準
構造	ソースコードの信頼性	1	・コードは、設計を完全に正しく実装しているか？
	ソースコードの保守性	2	・コードは、適切に構造化され、スタイルと形式が規約や標準に従った一貫性をもっているか？
		3	・コード内にスタブやテストルーチンが残っていないか？
		4	・外部の再利用可能なモジュール（コンポーネント）またはライブラリ関数を呼ぶことで置換できるコードは存在しないか？
		5	・1つのプロシージャにまとめられる反復的なコードブロックは存在しないか？
		6	・定数または文字列定数は、ハードコーディングせずに、シンボルを使用しているか？（マジックナンバー）
文書化	ソースコードの保守性	7	・コードは、開発者以外が保守しやすいように、明確かつ適切にコメントを記載しているか？
		8	・すべてのコメントは、規約や標準に従い、一定のルールで一貫性をもって記載されているか？
変数	ソースコードの信頼性	9	変数のライフサイクルでの不正を検出し、初期化忘れ、破棄忘れ、未使用などを検出しているか（データフロー解析） define：変数を宣言、定義、または初期化する。 use：計算または判定述語で、変数を使用すなわち読み込む。 kill：変数を削除、破棄する、または変数がスコープ外になる。
	ソースコードの保守性	10	・すべての変数は、規約や標準に従い、意味のある一貫した明確な名前を使用して、適切に定義されているか？（命名規約）
算術演算	ソースコードの信頼性	11	・算術演算について下記のような誤差を予防する確認をしているか コードは、浮動小数点数の等号による比較を避けているか？ コードは、体系的に丸め誤差を防いでいるか？ コードは、桁数が大きく異なる数字の加算および減算を避けているか？ 除数がゼロまたは無効値のテストをしているか？
ループとブランチ	ソースコードの信頼性	12	・すべてのループ、ブランチ、および論理構造は、完全で、正しくて、また適切にネストしているか？（制御フロー解析）
		13	・IF-ELSEIF チェーンの最初で分岐するのは最も頻度の高いパターンとなっているか？
		14	・ループの終了条件は明白で必ず成立するか？
	ソースコードの保守性	15	・ループ内のステートメントで、ループ外に出せるものはないか？
	ソースコードの信頼性	16	・外部デバイスのアクセスで、タイムアウトまたはエラートラップを使用しているか？
		17	・ファイルにアクセスする前に、存在していることをチェックしているか？
		18	・プログラムの終了時に、すべてのファイルとデバイスを正しい状態にしているか？
セキュリティ	セキュリティ	19	機密情報の取扱いについて、要件定義を受けて、コーディングにあたってのルールが決まっており、、コードレビューで実装を確認していること 例）機密情報はプレーンテキストで格納しない、プログラムで使い終わった機密情報はディスクに保管しない、再利用可能な動的メモリに格納された機密情報は消去する等
		20	バッファーオーバーフローや不正なコマンドを実行する信頼できない可能性のあるデータ入力を防ぐ入力チェックや、文字列操作のライブラリを実装するルールが決まっており、コードレビューで確認していること

3. 実施概要 – III.導入にあたり工夫したこと

③ 判定対象案件の絞り込み（チェック観点のレベル感への対応）2/2

チェック項目には、**それぞれ解説を付ける**ことで、開発担当者が、チェックする内容とその目的を理解できるようにした。

分類		No	評価基準	評価基準の解説
構造	ソースコードの信頼性	1	・コードは、設計を完全に正しく実装しているか？	–
	ソースコードの保守性	2	・コードは、適切に構造化され、スタイルと形式が規約や標準に従った一貫性をもっているか？	コーディングルールに従い、一貫したルールでコード書くことで、コードの可読性が高まります。 これにより、保守で既存のコードを派生開発する場合に欠陥を埋め込みにくくなります。 また、正しく動作していたとしても、コードの保守性が低い場合もあり、コードの保守性は、通常のテスト工程では発見できません。したがってコーディングルールの設定と製造工程での作り込みが非常に重要になります。
		3	・ コード内にスタブやテストルーチンが残っていないか？	–
		4	・ 外部の再利用可能なモジュール（コンポーネント）またはライブラリ関数を呼ぶことで置換できるコードは存在しないか？	1つのモジュール（コンポーネント）や関数を再利用することで1つの処理を1カ所で定義します。そうすることで、機能の凝集度が上がり、保守で既存のコードを修正する場合の影響範囲の特定漏れや改修漏れを予防できます。
		5	・ １つのプロシージャにまとめられる反復的なコードブロックは存在しないか？	
		6	・ 定数または文字列定数は、ハードコーディングせずに、シンボルを使用しているか？（マジックナンバー）	数値を必ず変数として定義することで、コードの可読性が高まります。 保守で既存のコードを修正する場合に、下記の理由で埋め込まれる欠陥を予防できます ・変数名などの名前を持たないため、その数値の持つ意味がわかりづらい ・数値を変更する場合に、複数の箇所を変更しなければならない場合の見落とし
文書化	ソースコードの保守性	7	・コードは、開発者以外が保守しやすいように、明確かつ適切にコメントを記載しているか？	コーディングルールに従い、一貫したルールでコード書くことで、コードの可読性が高まります。 これにより、保守で既存のコードを派生開発する場合に欠陥を埋め込みにくくなります。 コメントに記載する内容を作成者以外が読んで理解できるものになっているかの確認です。
		8	・すべてのコメントは、規約や標準に従い、一定のルールで一貫性をもって記載されているか？	コーディングルールに従い、一貫したルールでコード書くことで、コードの可読性が高まります。 これにより、保守で既存のコードを派生開発する場合に欠陥を埋め込みにくくなります。 コメントに記載する項目と内容がルールを守っているかどうかの確認です。

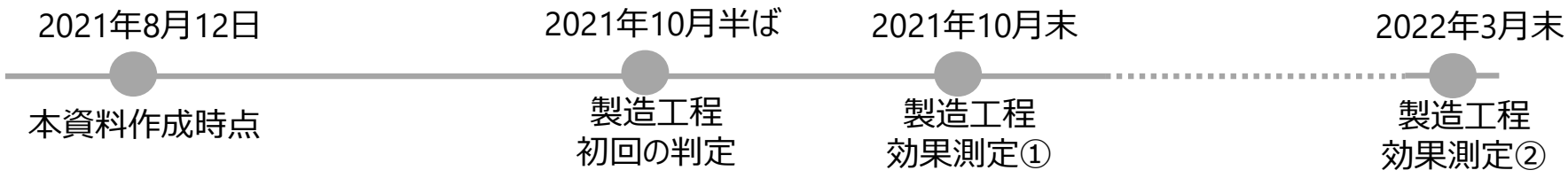
4. 効果測定について

重点目標（KPI）

2022年3月末における重点目標（KPI）	製造工程での埋め込み本番障害件数 10%以上削減（対前年度比）

今後の効果測定スケジュール

- ✓ 2021年5月から展開しているが、現時点（2021年8月12日）で適用対象の案件が発生しておらず効果測定は実施していない。初回の判定は2021年10月半ばを予定している。
- ✓ 2021年10月末、2022年3月末の計2回のタイミングで効果測定を実施する予定である。
- ✓ 上記タイミングで、工程完了判定における開発担当者への勧告事項の分析を行い、製造工程の品質作り込みにおける現状の弱点とその理由を考察したうえで、対応策を検討する。



5. 今後の展開

開発担当者自身で品質を作り込むことができるノウハウの整理を推進する

➢ 開発担当者のスキルの向上

判定結果とともに、改善すべき勧告事項を判定者からフィードバックすることで、判定実績の積み上げが、開発担当者のスキル向上に役立つようにする。

➢ ツールの整理

調達
提案依頼書（RFP）の改善

委託開発案件での、品質管理プロセスの実行を着実に行うため、RFPに品質要求として製造工程での品質管理活動を明記する。

製造
開発標準へのフィードバック

コーディングルールや、ソースコードレビューのチェックリスト、静的解析ツールなどを製造工程おける当社の開発標準として整理する。

判定
チェックリストの改善

発生した障害の原因分析や効果測定などで新たに抽出した観点をチェックリストにフィードバックし、横展開する。

➢ 形骸化防止の取組み

発生障害の分析、製造工程での欠陥の傾向分析を定期的に継続し、形骸化による障害の発生がないかを確認する

ご清聴ありがとうございました

本発表を通して、同様の取組みをされている皆さまとの情報交換、意見交換に繋がればと考えています。

本発表に関するお問い合わせ先

オリックス生命保険株式会社
IT業務管理部
藤井　和弘
kazuhiro.fujii.dd@ins.orix.jp