15-A-10

XDDP におけるデグレード防止効果を高めるための手法 [1]

～『気づきナビ』の考案～

1. 適用した技術や手法の概要

既存のソフトウェアに変更・追加等を行い、別のソフトウェアを開発する派生開発では、既存ソフトウェアの仕様書や設計書が更新されない、設計書とソースコードの乖離が次第に大きくなるといった問題がある。そのため、非熟練技術者の場合、派生開発での仕様変更の影響箇所の見極めを誤り、変更箇所の故障や変更が影響して起きる既存箇所の故障（デグレード）が繰り返し発生している。

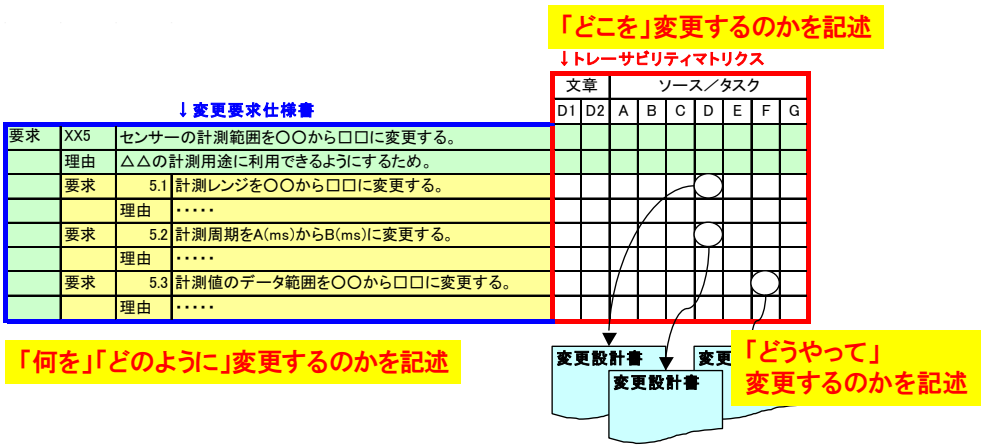
派生開発におけるこれらの問題を解決するため、XDDP（eXtreme Derivative Development Process）手法の導入を検討したが、XDDP 手法においても、非熟練技術者の場合､デグレードに対しては XDDP の導入効果が少ないことが現状分析（シミュレーション）で明らかになった。

本編では、XDDP の導入にあたり、技術者の熟練度に依存しなくても変更箇所の故障やデグレードを低減するなどの効果を得られる手法について紹介する。

本事例で導入した XDDP は派生開発に特化したプロセスモデルであり、 変更を表現する視点から３点セットと呼ばれる３種類の成果物を定義している（図 15-A-10-1）。具体的には、「何を」「どのように」変更するかを記述する変更要求仕様書､「どこを」変更するかを記述するトレーサビリティ・マトリクス､「どうやって」変更するかを記述する変更設計書の３種類である。

---

[1] 事例提供:
アズビル株式会社 ビルシステムカンパニー 開発本部　関野 浩之 氏
株式会社インテック コンサルティング事業部　大坪 智治 氏
キヤノン IT ソリューションズ株式会社 IT サービス事業本部　外谷地 茂 氏



担当者の思い込みや勘違いを低減するレビューの効果を引き出し、不具合の作り込みを防ぐ手法

図 15-A-10-1　XDDP 3 点セットの成果物

2. 適用した技術や手法の導入に踏み切った理由や経緯

2.1. 背景と『気づきナビ』について

派生開発では仕様変更が頻繁に発生し、それらは短納期・低コストを要求される場合が大多数であるため、仕様書や設計書の更新が後回しになったり、行われなかったりしている。また、繰り返される派生開発の中で、設計書とソースコードの乖離が次第に大きくなり、設計書が使い物にならなくなるなど、派生開発を取り巻く環境は劣悪な状態にある。このような環境の中で品質を確保するためには、仕様変更による影響箇所の正確な見極めが欠かせないことから、現場では、派生開発に必要な知識・スキル及び経験を十分に持った熟練技術者を頼りに仕様変更による影響箇所の見極めを行うことが望ましい。

しかしながら常に熟練技術者を割り当てることが難しく、その結果、影響箇所の見極めを誤り、変更箇所の故障やデグレードが繰り返し発生している。

そこで、XDDP を導入することによって、熟練技術者に依存しなくても変更箇所の故障やデグレードを低減できないかの検討を行った。はじめに XDDP の導入効果をシミュレーションにより確認した。その結果判明したことは、XDDP は変更箇所の故障の防止には有効であるが、デグレードの防止についての効果は熟練技術者が持つ開発経験から得られる知識・スキルの有無に大きく影響を受けることであった。そこで熟練技術者に頼ることなく XDDP によるデグレード防止効果を向上させる工夫を検討した。

熟練技術者に頼ることなく非熟練技術者でも XDDP によるデグレード防止効果を向上させる工夫として、『気づきナビ』という独自の手法を考案した。『気づきナビ』は、ある規則で作られた２種類の表を組み合わせて使用することで、派生開発の中で発生する変更内容（変更仕様）とデグレード防止の教訓を紐づけて可視化する手法である。『気づきナビ』の使用に

より、熟練技術者でなくとも仕様変更に必要な教訓を引き出すことができ、XDDP によるデグレード防止効果の向上が可能となった。

2.2. 現状分析と効果シミュレーション

XDDP を導入するにあたり、その効果を確認するため、現状分析とシミュレーションを実施した。不具合事例の現状分析や不具合防止効果のシミュレーションの詳細を記す。

(1) 不具合事例の現状分析

派生開発の不具合事例を収集・分析した結果、「変更箇所の故障」と「既存箇所の故障（デグレード)」が半々を占めていた。その結果を図 15-A-10-2 に示す。

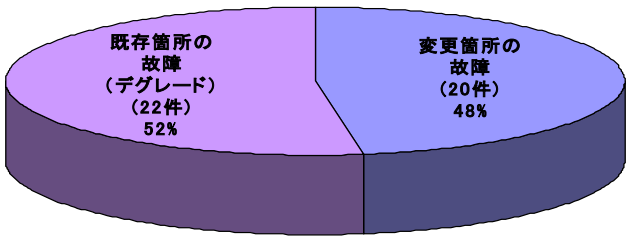


図 15-A-10-2　不具合事例の分析結果

XDDP を適用することで得られる不具合防止効果を評価するために、XDDP を適用しない開発で発生した不具合事例を用いてシミュレーションを行った。XDDP はそれぞれ視点の違った「３点セット（変更要求仕様書、トレーサビリティ・マトリクス、変更設計書)」の成果物の相互作用によって、レビューの効果を引き出し、担当者の思い込みや勘違いを低減する方法である。不具合の要因と XDDP による不具合防止のしくみを表 15-A-10-1 に示す。

表 15-A-10-1　XDDP による不具合防止のしくみ

不具合要因	思い込みや勘違いを低減するしくみ
A) 要求や仕様の勘違い	「変更要求仕様書」のレビューで不具合を除去する
B) 仕様レベルでの変更場所の特定の勘違い	「トレーサビリティ・マトリクス」のレビューで不具合を除去する
C) ソースレベルでの変更場所の特定や変更方法の勘違い	「変更設計書」のレビューで不具合を除去する

(2) XDDP による不具合防止効果シミュレーション

XDDP を適用した場合でもそれによる不具合防止の効果は、開発に関わる技術者の知識・スキルレベルに大きく依存することが考えられる（仮説)。そこで知識・スキル

の有無が不具合防止にどのように影響するのかを把握するために、当該製品の開発に必要な知識・スキルを持つ「熟練技術者」と、それらの知識を持たない「非熟練技術者」とを、シミュレーションのパラメータとした。熟練技術者と非熟練技術者の知識・スキルの差を表 15-A-10-2 のように定義した。

表 15-A-10-2　熟練技術者と非熟練技術者の知識・スキル

熟練技術者と非熟練技術者の知識・スキル	熟練 技術者	非熟練 技術者
ITSS/ETSS のレベル	4	1～3
ソースコード読解力（アーキテクチャ読解力）	十分	十分
当該製品の開発経験	十分	不足
ソースコードレベルの製品知識	十分	不足
当該製品を利用する顧客側の業務・運用の知識	十分	不足
当該製品のハードウェア・ソフトウェアの知識	十分	不足

表 15-A-10-1 に示すように XDDP による不具合防止は主にレビューによる。一般にレビューの効果はレビュアーの知識・スキルに左右されるため、非熟練技術者プロジェクトでは XDDP の不具合防止効果は十分でないと仮定した。

また「デグレード」の防止には仕様変更による影響箇所の正確な見極めが必須となるため、「変更箇所の故障」の防止に比べ、知識・スキルへの依存性が高いと仮定した。

不具合事例を表 15-A-10-1 で判定した時の不具合防止効果を表 15-A-10-3 に示す。熟練技術者プロジェクトでは「変更箇所の故障」と「デグレード」の不具合防止効果に大きな差異はなく、どちらも 80%以上となった。

それに対して非熟練技術者プロジェクトでは「変更箇所の故障」の不具合防止効果は 42%、「デグレード」の防止効果は 15%であった。以上から非熟練技術者プロジェクトで発生するデグレードは XDDP を適用した場合でも防止が難しいことがわかった。

表 15-A-10-3　XDDP の不具合防止シミュレーション結果

不具合現象	パラメータ	不具合 総計	XDDP 対象領域			XDDP 対象領域外 (*2)
			不具合 件数	防止可（*1）	防止不可	
変更箇所の故障	熟練技術者	20 件	19 件	17 件(89%)	2 件(11%)	1 件
	非熟練技術者	20 件	19 件	8 件(42%)	11 件(58%)	1 件
既存箇所の故障 （デグレード）	熟練技術者	22 件	20 件	16 件(80%)	4 件(20%)	2 件
	非熟練技術者	22 件	20 件	3 件(15%)	17 件(85%)	2 件

(*1) カッコ内は不具合防止効果（XDDP で防止可の不具合事例数／XDDP 対象領域の不具合事例数×100%）を示す

(*2) 開発工程終了後の追加要件管理ミスなど、XDDP の対象とならない工程に原因があった不具合

(3)　デグレード防止に必要な知識・スキルの分析

（2）のシミュレーション結果より、デグレード防止には熟練技術者が持つ知識やスキルが大きく寄与していることがわかった。次に過去のデグレード事例 20 件を選び、それらの不具合要因に対し、知識・スキルの有無がどう影響するのかを分析した。その結果を表 15-A-10-4 に示す。

表 15-A-10-4 の分析結果より、非熟練技術者のデグレード防止に必要な知識・スキルとして以下の 6 点が挙げられた。

本事例では非熟練技術者がソースコードを読み込むだけでは得られない①～⑥の知識を「ソースコード外知識」と定義する。

① 業務・運用に関する知識（画面操作など）
② スペックアウト方法の知識（調査場所、調査方法など）
③ データに関する知識（データ構造、データ範囲など）
④ 制御に関する知識（関数呼び出し、イベント／タスクの流れ、データの排他制御など）
⑤ 非機能要求の知識（リソース、パフォーマンスなど）
⑥ 動作環境に関する知識（OS、ミドルウェアなど）

①の知識は仕様書などを調査し、ベースソフトウェアの業務・運用を理解することで得られる。②の知識は類似した変更仕様の経験が必要である。③～⑤の知識は設計書などを調査し、ベースソフトウェアのアーキテクチャを理解することで得られる。設計書が整備されていない場合、ソースコードを読み込み、データ構造や処理構造や制御構造を明らかにする。その上で DFD（Data Flow Diagram）やシーケンス図などを活用してベースソフトウェアのアーキテクチャを理解する。⑥は動作環境の観点がなければ気づくことが難しい知識である。

表 15-A-10-4　デグレードの不具合要因とデグレード防止に必要な知識・スキル

知識・スキルへの依存	不具合要因	必要な知識・スキル	熟練技術者(*3)	非熟練技術者(*3)	小計	合計
依存する	A-1)変更による類型的な影響パターンを認識できず、変更場所、変更方法を間違えた。	スペックアウト方法	○	×	1 件	14 件
	A-2)仕様書に記述されていない仕様、設計書に記述されていない実装を認識できず、影響場所の候補にならなかった。	業務・運用に関する知識	○	×	1 件	
		データに関する知識	○	×	2 件	
		制御に関する知識	○	×	1 件	
		動作環境に関する知識	○	×	2 件	
	A-3)仕様書に記述されていない非機能要求を認識できず、影響場所の候補にならなかった。	非機能要求の知識 (パフォーマンスなど)	○	×	2 件	
	A-4)設計書に記述されていない制約を認識できず変更場所、変更方法を間違えた。	データに関する知識	○	×	2 件	
		制御に関する知識	○	×	1 件	
		動作環境に関する知識	○	×	1 件	
	A-5)設計書に記述されていない連携を認識できず影響場所の特定を間違えた。	制御に関する知識 (サービス、タスク、バッチなど)	○	×	1 件	
依存しない	B-1)スリップ（錯誤）→コーディングミスなど		×	×	1 件	6件
	B-2)ラップス（やり忘れ）→網羅性の検討漏れなど		○	○	1 件	
	B-3)プロセス違反→テストしていない、変更管理していないなど		×	×	2 件	
	B-4)ソースコードの読解ミス		○	○	2 件	

(*3)（2）の XDDP の不具合防止シミュレーション結果（○：XDDP で防止可／×：XDDP で防止不可）

（4）　XDDP による不具合防止効果シミュレーション結果のまとめ

表 15-A-10-4 の A-1)～A-5) はいずれも設計書が整備されていない。このような環境に XDDP を適用した場合、スペックアウト方法や変更要求仕様書の記述内容が品質を確保するための鍵となるが、それらは設計者の知識・スキルに大きく依存している。非熟練技術者であれば、スペックアウト方法や変更要求仕様書の記述内容を一様にする手法[4]を適用することで、変更要求仕様書の不備が原因の後戻り作業を防止することができる。

しかし過去に発生したデグレードの要因は「ソースコード外知識」の不足による変更場所や変更方法や影響場所の間違いなので、変更要求仕様書を正確に記述しても、

「ソースコード外知識」を正確に知っていなければ、変更場所や変更方法や影響場所などをトレーサビリティ・マトリクスや変更設計書に反映することはできない。そのため非熟練技術者のデグレードを防止するには、「ソースコード外知識」を効率よく得る手法がポイントとなる。

「ソースコード外知識」を得る手段は設計書であるが、設計書が整備されていなければ、設計書から正確な情報を得ることは難しい。しかし多くの組織には過去の不具合事例から得た教訓を蓄積したチェックリストがある。過去の不具合事例には「ソースコード外知識」の見落としから発生したデグレードも多く、それらへの対策が記述されたチェックリストには「ソースコード外知識」が集積されていると推測される。したがって非熟練技術者がチェックリストから容易に「ソースコード外知識」を引き出す手法を考案すれば、過去に発生したデグレードの発生を低減することが可能と考えた。

3. 適用した技術や手法をスムーズに導入し活動を定着させるための、事前準備や工夫等

XDDP によるデグレード防止効果を高めるための課題はチェックリストの問題である。設計チェックリストを収集・分析した結果、チェックリストの問題点と課題は表 15-A-10-5 のようになった。

表 15-A-10-5　チェックリストの問題点と課題

No.	問題点	課題
1	網羅性を追求する目的からチェックリストが肥大化している。またチェックリストはレビュー対象（設計チェックリストの場合では機能など）ごとに整理されていることが多く、利用シーンや目的・観点によって整理されていない。そのため、莫大なチェックリストの中から必要なチェック項目を探し出すには時間がかかる。	莫大なチェック項目の中から必要なチェック項目を効率的に探し出すしくみをつくる。 ・・・チェックリストの課題①
2	非熟練技術者にとってチェック項目の内容が抽象度の高いものがある。そのため、チェック項目を当該製品に適用し具体化することができない。	何を変更して発生した不具合であるのかがわかる情報をチェック項目に追加する。 ・・・チェックリストの課題②
3	非熟練技術者にとってチェック項目の内容を理解できないものがある。そのため、変更仕様に合致したチェック項目であることを認識できない。	

(1)　解決策のポイント

表 15-A-10-5 で示したチェックリストの課題①を解決するためには、派生開発の中で発生する変更仕様とそれに関係したチェックリストを結び付ける共通情報が必要である。そのような共通情報として「変更特性」を考案した。変更特性とは変更仕様の中から“変更する”という行為を簡潔なキーワード（「何を」「どのように」変更した

のか）で一般化したものである。変更仕様から変更特性を抽出する例を図 15-A-10-3 に示す。

変更特性例

【変更仕様】:「〇〇タスクに××データを読み出し、△△を計算し、□□データを書き出しする.」

変更特性の抽出

【変更特性】:「××データの読み出し処理追加」、「△△計算処理追加」、
「□□データの書き出し処理追加」

図 15-A-10-3　変更特性の抽出例

また既存のチェックリストに「変更特性」を情報として付加することができれば、「何を」変更した時に参照すべきチェック項目であるのかが明確になるため、チェックリストの課題②も解決できる。

(2)　『気づきナビ』

莫大なチェック項目から今回の変更仕様に関係のあるチェック項目だけを引き出すために、変更要求仕様書の列項目に変更特性を付け加えた「変更特性マトリクス」と、既存のチェックリストに変更特性を加え整理したチェックリスト（「変更特性チェックリスト」）とを組み合わせ、変更仕様とデグレード防止の教訓を「変更特性」で紐づけて利用する手法を考案した。本事例ではこれを『気づきナビ』と定義する。

この『気づきナビ』は XDDP の３点セットの１つである変更要求仕様書と組み合わせて利用することがポイントである。変更要求仕様書には以下の特徴がある。

- 要求と仕様を階層関係で表現することで変更仕様をモレなく洗い出すことができる。
- 変更仕様は「変更前（Before）／変更後（After）」形式で変更内容が記述されており、「何が」「どのように」変更されるのかを理解しやすい。

これらの特徴を踏まえた上で、『気づきナビ』の使い方について説明をする。変更特性マトリクスの行項目には変更要求仕様書の変更仕様、列項目には変更特性チェックリストの変更特性が記述されている。まずは行項目の変更仕様ごとに列項目の変更特性を順に確認し、該当する変更特性の欄に〇印をつける。つぎに〇印のついた変更特性の変更特性チェックリストを順に確認し、変更仕様から抽出された変更特性に関係のあるチェック項目だけを確認する。これにより、『気づきナビ』の導入前後で以下の効果が得られる（図 15-A-10-4 『気づきナビ』導入前と導入後）。

- 導入前：変更仕様ごとにチェックリスト上の全チェック項目を確認する。
- 導入後：変更仕様から抽出された変更特性に関係のあるチェック項目だけを確認する。

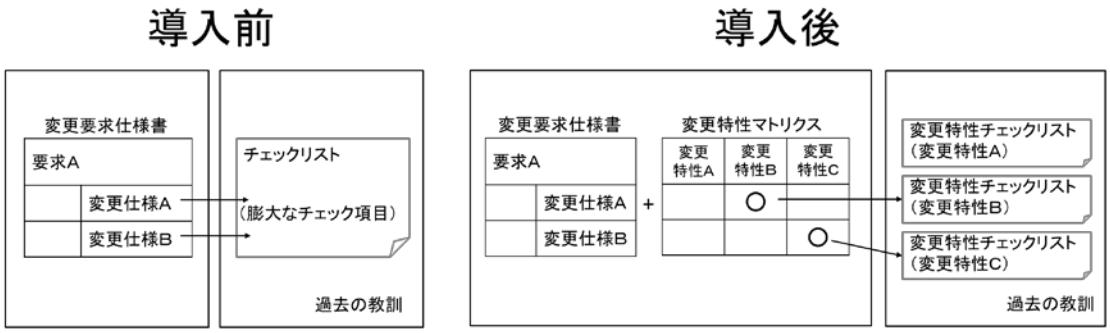


図 15-A-10-4　『気づきナビ』導入前と導入後

この『気づきナビ』をはじめて作成する場合、既存のチェックリストと不具合報告書を利用し、以下の手順で作成する（熟練技術者による作成）。

① 既存のチェックリストの元になった不具合報告書から「変更特性」を抽出し、それを既存のチェックリストに新しいキーワードとして追加する。

② 変更特性チェックリストの「変更特性」（「何を」「どのように」変更したのか）の「何を」が同じものを同じカテゴリにまとめ、変更要求仕様書の列項目に付け加える（「変更特性マトリクス」）。

(3)　変更特性マトリクス

変更特性マトリクスは、変更仕様とその変更仕様に含まれる変更特性の関係をマトリクスで表すことで、変更仕様における変更特性の分析を可能とする。変更特性マトリクスの例を図 15-A-10-5 に示す。

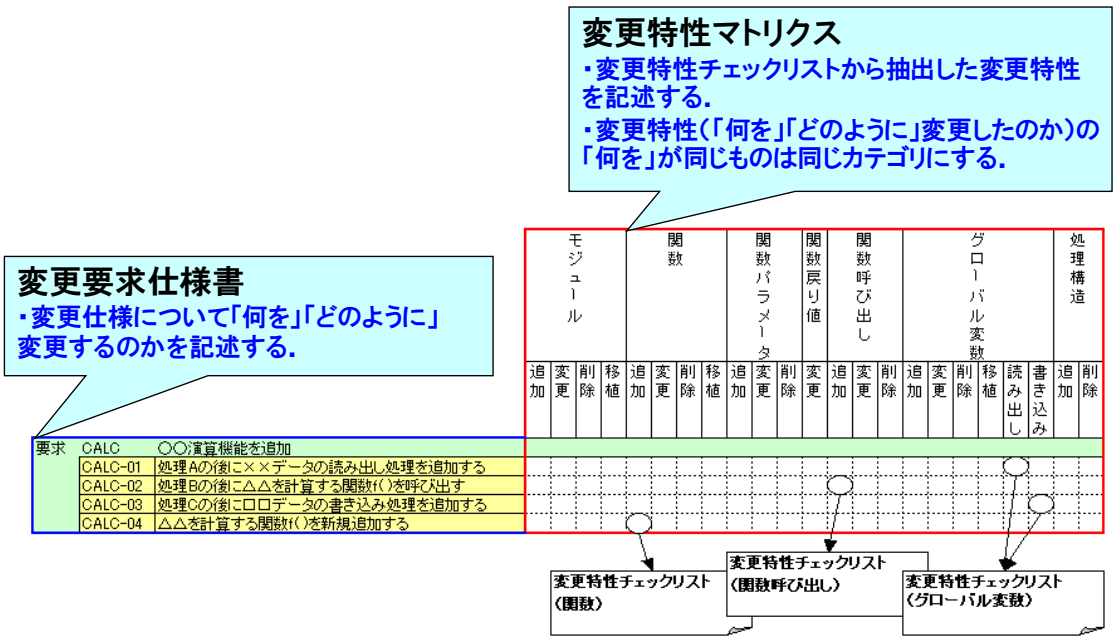


図 15-A-10-5　変更特性マトリクスの例

① 変更特性マトリクスの行項目

XDDP の３点セットの１つである変更要求仕様書によって抽出された変更仕様を記述する。

② 変更特性マトリクスの列項目

変更特性チェックリストから抽出した変更特性を記述する。

変更特性（「何を」「どのように」変更したのか）の「何を」が同じものは同じカテゴリにする。

(4) 変更特性チェックリスト

変更特性チェックリストは既存のチェックリストに「変更特性」の情報を追加したものである。「変更特性」は既存のチェックリストの元になった不具合報告書（表 15-A-10-6）から抜き出し、注意するポイントの抽象度に応じて製品固有のものと製品間で共通のものとを準備する。変更特性チェックリストの例を表 15-A-10-7 に示す。

表 15-A-10-6　不具合報告書の例

項目	内容
不具合事例	B タスクの計測処理が規定時間内に完了しないことがあり、計測値がエラーになる。
作りこみ要因	A タスクに同期待ちのある関数 f( )の呼び出しを追加したところ、A タスク実行中に B タスクが動作できなくなった。
チェック項目	タスクに同期待ち処理がある場合、このタスクよりも優先度の低いタスクが動作できなくなるので注意する。

表 15-A-10-7　変更特性チェックリストの例

抽象度	分類	変更特性	目的	チェック項目
低い ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ 高い	製品固有	製品○○に関数 f( )の呼び出し追加	・割り込み時間が長くなることで、システム動作を保証できなくなることを防止するため。 ・タスク処理の同期待ちが長くなることで、システム動作のパフォーマンスが劣化することを防止するため。	関数 f( )は通信を使って外部データを取得する。外部データを取得するまでは for ループによる待ちがあるため、関数 f( )の呼び出しを追加するとタスクの処理時間は長くなる。このタスクの実行中、このタスクよりも優先度の低いタスクが動作できなくなるので注意する。
		（同期待ちのある）関数呼び出し追加		リアルタイムシステムの場合は、タスクの処理時間が長くなる。このタスクの実行中、このタスクよりも優先度の低いタスクが動作できなくなるので注意する。
	製品間で共通	データの読み出し処理追加		リアルタイムシステムの場合は、データの読み出しの影響でタスクの処理時間が長くなっていないかを確認する。

(5)　『気づきナビ』のメンテナンス

本手法を利用した開発の中で新たな不具合が発生した場合は、不具合の発生した変更仕様を変更要求仕様書上で確認し、対応する変更特性を変更特性マトリクス上で特定する。不具合の分析を行い、再発防止の教訓を変更特性チェックリストに追加する。

また派生開発の中で発生する変更仕様は多種多様なため、新たな変更特性が見つかった場合は、変更特性を変更特性マトリクス、設計上の定石や注意点を変更特性チェックリストに追加する。

上記のようなメンテナンスを繰り返すことで『気づきナビ』をそれぞれの製品にフィットしたもの（ガイド）に成長させることが可能となる（図 15-A-10-6）。

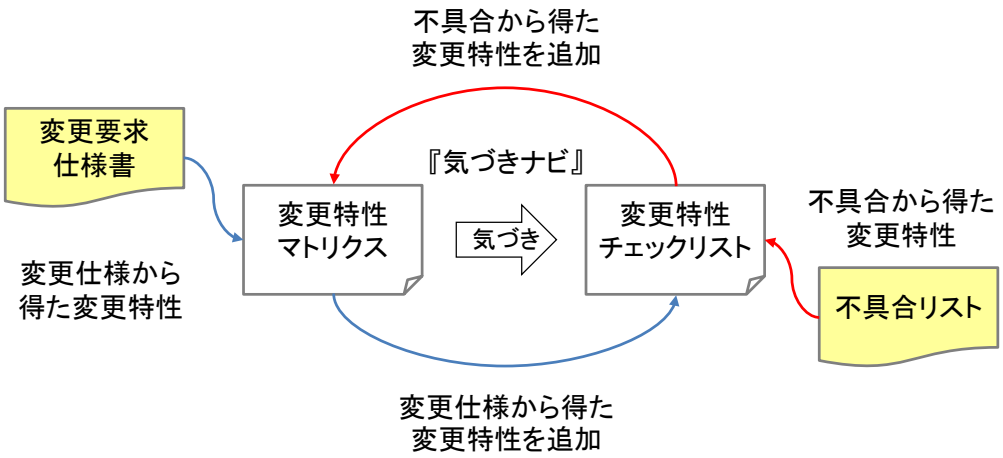


図 15-A-10-6　『気づきナビ』の成長モデル

4. 適用した技術や手法の効果測定の方法とその結果

変更特性マトリクスと変更特性チェックリストからなる『気づきナビ』を使用することにより、非熟練技術者でも、変更仕様に対応するチェック項目を抽出することができ、デグレード防止に有効であることの検証結果を示す。

(1)　検証方法

『気づきナビ』の効果の検証は、表 15-A-10-8 に示すように、過去のデグレード事例に対するシミュレーション検証①と、XDDP を導入したプロジェクト（組込み系開発：変更仕様数：21 件）で行う検証②について実施した。

表 15-A-10-8　『気づきナビ』の効果の検証方法

	対象	目的	方法
検証①	「変更特性チェックリスト」	変更特性を介して変更内容に対応したチェック項目にたどり着き、そこに有効な情報があった場合に、デグレード防止に関与するか否かを検証する。	1)過去のデグレード事例に対し、デグレード防止のための変更特性とチェック項目を作成する。 2)次回以降の派生開発で、作成したチェック項目にたどりつけた場合、デグレード防止に関与するか、否かをシミュレーションする。
検証	「変更特性マトリクス」「変更特性チェックリスト」	非熟練技術者でも、確認が必要なチェック項目の数を少なくできることを検証する。	XDDP のプロジェクトにおいて、変更内容に対するチェック項目を「チェックリスト」で抽出した場合と「変更特性マトリクス」で抽出した場合を比較する。(*4)
		非熟練技術者でも、経験に依存することなくチェック項目を抽出できることを検証する。	XDDP のプロジェクトに「変更特性マトリクス」と「変更特性チェックリスト」を適用し、変更内容に対する熟練技術者のチェック項目と非熟練技術者のチェック項目を比較する。(*4)
		非熟練技術者でも、『気づきナビ』使用前には気づけなかった新たな欠陥に気づけることを検証する。	XDDP のプロジェクトに「変更特性マトリクス」と「変更特性チェックリスト」を適用し、変更内容に対する熟練技術者のチェック項目と非熟練技術者のチェック項目を比較する。(*4)

(*4)　熟練技術者 2 名（経験 10 年以上）、非熟練技術者 2 名（経験 4 年以下）について確認

(2)　検証結果

検証①（過去のデグレード事例のシミュレーション）では収集したデグレード事例 22 件中、非熟練技術者にとっては XDDP を適用してもデグレード防止が難しい 17 事例を検証した結果、13 事例について非熟練技術者がデグレードの回避策や注意事項・確認事項の教訓を取得でき、それらがデグレード防止に効果的であることが確認できた。

検証②では XDDP を導入したプロジェクトにおいて『気づきナビ』に期待される 3 つの効果が得られるかを検証した。検証の結果、３つの効果についていずれも当初期待した結果を得られた（表 15-A-10-9）。

表 15-A-10-9　『気づきナビ』に期待される効果と確認された効果

No.	期待される効果	確認された効果
1	非熟練技術者でも、確認が必要なチェック項目の数を少なくできる。	変更仕様 21 件に対し、チェックリストのチェック項目は 63 件、変更特性チェック項目は 52 件となり、約 17%少なくなった。
2	非熟練技術者でも、経験に依存することなくチェック項目を抽出できる。	変更仕様 21 件に対し、熟練技術者のチェック項目は 52 件、非熟練技術者のチェック項目は 54 件となりほぼ同数になった。また熟練技術者が抽出した 52 件は、非熟練技術者も 100%を抽出できていた。
3	非熟練技術者でも、『気づきナビ』使用前には気づけなかった新たな欠陥に気づける。	変更仕様 21 件、チェック項目総数 54 件に対し、非熟練技術者が見落としがちな関数パラメータ追加時の異常処理の抜け（のべ 5 件）に気づき、確認モレを防止できた。

(3)　考察

検証①（過去のデグレード事例のシミュレーション）の結果では、非熟練技術者にとっては XDDP を適用してもデグレード防止が難しい 17 事例中の 13 事例について非熟練技術者がデグレードの回避策や注意事項、確認事項の教訓を取得でき、それらがデグレード防止に効果的であることが確認できた。

これにより、変更特性マトリクスと変更特性チェックリストからなる『気づきナビ』を使用することにより、非熟練技術者でも、変更仕様に対応するチェック項目を抽出することができ、当初の課題であるデグレード防止に有効であることが確認できた。

しかし、チェックリストの内容によっては、今回の検証結果と同等のデグレード防止に関する有益な情報を得られない可能性もあり、『気づきナビ』で高い効果を得られるためのチェックリストの作成指針を検討することが、今後の課題である。

検証②の結果（表 15-A-10-9）より、「熟練技術者に頼ることなく XDDP によるデグレード防止効果を高める」という目的に対して期待した効果が得られたと考える。表 15-A-10-10 に検証②の各結果に対する考察を記載する。

表 15-A-10-10　『気づきナビ』の検証結果に関する考察

No.	検証結果	検証結果に対する考察
1	確認するチェック項目の数が『気づきナビ』導入前に比べ約 17%少なくなった。	検証結果よりも高い効果を期待していたが、検証で利用したチェックリストには、設計共通のチェック項目が多く、製品固有のチェック項目が含まれていないことから、このような結果になったと考えられる。
2	熟練技術者と非熟練技術者で確認したチェック項目がほぼ同数になった。	変更特性に従って機械的にチェック項目を確認することにより、知識・スキルに依存することなくチェック項目の確認が行えるようになったためと考えられる。
3	非熟練技術者が見落としがちな異常処理に関する抜けに気づいた。	

また、検証を行っていく中で、デグレードの問題に限らず、変更箇所の故障といった他の問題に関しても多くの気づきがあることがわかった。これは、『気づきナビ』の本質が変更特性を抽出して変更仕様とチェック項目（過去の教訓）の紐付けを行うことにあるため、チェック項目の内容がデグレードに関するものでなくてもデグレードの問題と同様の効果が発揮されたものと考えられる。

5. 現場への導入時の具体的な工夫と適用手法

(1)　試行プロジェクト

本手法を適用するにあたり試行プロジェクトによる確認を行った。なぜならば、試行プロジェクトにより効果を確認できれば、他のプロジェクトへ展開する時に、導入障壁を軽減する効果となるためである。試行プロジェクトは、組込みの小規模プロジェクトである。一気に大規模プロジェクトに適用せず、組込みの小規模プロジェクトを選びノウハウの蓄積を図った。

(2)　試行プロジェクトの体制

熟練技術者 2 名（経験 10 年以上)、非熟練技術者 2 名（経験 4 年以下）で試行してみた。試行プロジェクトには 2 つのサンプルを 2 つのパターンで試行確認を実施した。熟練技術者が少ないプロジェクトも可能なことを確認できた。

(3)　『気づきナビ』の普及

考案した『気づきナビ』の効果について、グループ内に展開を図った。

6. 品質・生産性について

品質については「4.（2）検証結果」 に記載した通り、必要なチェック項目を漏らすことなく抽出できたという点で非熟練技術者の作業品質が向上したと判断する。

生産性については定性的ではあるが、品質をあげつつ、生産性もあげるという目標であっ

た。「4.（2）検証結果」 に記載した通り、作業品質が向上しており、かつ後戻りを防止しているという点で、従来よりも工数は削減され生産性が向上したと判断する。

熟練技術者が作成した変更特性チェックリストを非熟練技術者が使用すれば効率の良い開発が可能になる。

7. 導入後の普及改善活動、今後の課題等

（1） 検証への影響

本手法の導入で設計時にバグを作りこむことが減少するので検証時に発生していた後戻り作業は減少すると考えられる。

また、組込みソフトウェアでは設計者と検証者が同一であることが多いが、設計者と検証者が異なる場合でも、変更特性が共有されていれば、変更特性チェックリストからチェック項目を同様に抽出することができるため、設計者と異なる人が検証に携わることが可能となる。

（2） 水平展開状況

グループ内への展開は完了している。他のグループには、抽出すべき変更特性の種類が必ずしも同じにならないため、そのままの展開はできないが、他のグループの特性に合った変更特性を作成すれば適用は可能である。

（3） 今後の課題

『気づきナビ』による不具合防止効果はチェック項目の記述内容に大きく依存している。網羅性の高い、整理された、質の高いチェック項目は不具合防止効果を高めると考えられるが、記述内容が自由である以上、定量的な効果を得ることが難しい。不具合防止効果を高めるための情報をどのように記録するのか、『気づきナビ』に適した不具合報告書の記述方法、チェック項目の記述方法などを検討していくことが今後の課題である。

『気づきナビ』をナレッジマネジメントのツールとして組織的に活用していくためには、知識を引き出すための「変更特性」をわかりやすく体系的に整理することが必要となる。これらの方法を検討することも今後の重要な課題である。

8. まとめ

XDDP を導入しても非熟練技術者プロジェクトで発生するデグレードを防止するために、本事例では、「変更特性マトリクス」と「変更特性チェックリスト」を組み合わせて使用する手法『気づきナビ』を考案し、過去のデグレード事例に対するシミュレーション検証と XDDP を導入したプロジェクトでの検証を比較することによりその効果を確認した。

その結果、『気づきナビ』を用いれば技術者の知識・スキルに依存することなく、変更内容に必要なチェック項目を確認することが可能になった。『気づきナビ』は「熟練技術者に頼る

ことなく XDDP によるデグレード防止効果を高める」という課題に対して一定の成果を上げることができた。

また『気づきナビ』を現場に導入・定着させるために、既存のチェックリストと不具合報告書から「変更特性マトリクス」と「変更特性チェックリスト」を作成し、メンテナンスしていく方法を示した。これらの方法を使うことで「変更特性マトリクス」と「変更特性チェックリスト」は双方の整合性を保ちながらの運用が可能になったと考える。

一方『気づきナビ』は熟練技術者の暗黙知を形式知として蓄積し、非熟練技術者に活用することが可能なので、ナレッジマネジメントのツールとして見ることができる。『気づきナビ』に蓄積されている知識を組織的に利用し、SECI モデルの「共同化（Socialization）」→「表出化（Externalization）」→「連結化（Combination）」→「内面化（Internalization）」のサイクルを繰り返せば、熟練技術者が気づいていない知識が新たに出てきたり、技術者同士の共通の体験が生まれたりするため、さらに効果的なものになると期待される。

参考文献

[1] 関野 浩之、大坪 智治、外谷地 茂、長友 優治：XDDP によるデグレード防止効果の検証とその効果を高めるための方法、ソフトウェア品質（SQiP）シンポジウム、2011

[2] 清水 吉男：「派生開発」を成功させるプロセス改善の技術と極意、技術評論社、2007

[3] 組込み製品の品質を高めるための暗黙知の抽出・利用方法、ソフトウェア品質管理研究会 2007、日本科学技術連盟

[4] XDDP によるデグレード防止効果の検証とその効果を高めるための方法、ソフトウェア品質管理研究会 2010、日本科学技術連盟

[5] 津田 剛宏、田中 聡、古畑 慶次：設計手法を活用した変更要求仕様書の作成手法、ソフトウェア品質（SQiP）シンポジウム、2010

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（IPA/SEC）