派生開発における協力会社への前提知識の伝達方法

主査　　：飯泉　紀子　（株式会社日立ハイテクノロジーズ）
副主査　　：足立　久美　（株式会社デンソー）
アドバイザー　：清水　吉男　（株式会社システムクリエイツ）
リーダー　　：伊藤　敏也　（株式会社インテック）
研究員　　：羽渕　喜英　（株式会社インテック）
　　鎌田　朋洋　（アンリツエンジニアリング株式会社）
　　山本　貴之　（アンリツエンジニアリング株式会社）

研究概要
派生開発において，開発案件の増加やコスト削減，納期短縮に対応するために，経験豊富な協力会社だけでなく今まで付き合いのない新規の協力会社との協業も必要となってきた．しかしながら，新規の協力会社に委託したソフトウェアの品質が悪く，手戻り工数が発生し，さらに次の案件への着手が遅れるなど悪循環を招いている．そこで，知識・経験の浅い新規の協力会社に対して，製品開発に必要な知識を過不足なく伝え，ソフトウェア品質を安定させる手法「機能一覧 Chips 紐付け表」を考案した．この考案を新規の協力会社が関係した不具合事例に適用した結果，不具合の低減（手戻り防止）に効果があることがわかった．

1.　はじめに

派生開発の現場では，ベンダーが顧客から改造の要求事項をとりまとめ，協力会社にプログラム設計・実装・単体テストを依頼し，最後にベンダーが受け入れテストを実施するビジネスモデルで開発を進めている．以前は，同一の協力会社に同一製品の派生開発を委託していたため開発は順調であったが，最近は開発案件の増加やコスト削減，納期短縮に対応するために経験豊富な協力会社だけでなく，今までに付き合いのない新規の協力会社との協業を余儀なくされている．また，協力会社の選定にはコスト面での制約があり，ベンダーの開発担当者には選定権限がない状況である．

この状況で問題となっていることのひとつが新規の協力会社起因による不具合の多さである．特に顧客へ製品を納品した後に不具合が発生するとベンダーはその対応を行うため，現在着手している案件を中断せざるを得ない状況となり，さらにコストが増えるといった負の連鎖が続いてしまっている．新規の協力会社の選定には購買部門が調査を行った上で一定の実績がある協力会社に限定しているが，選定基準をクリアした場合でも前述したように不具合が多く発生している．

そこで,新規の協力会社との協業においてどのような問題が発生しているかを持ち寄り，分析を行った．その結果，新規の協力会社に委託した場合の不具合には，用語の理解不足など既存製品派生開発時の前提知識に起因する不具合が多く，さらに分析を進めると，協力会社に提供した仕様書やガイドラインの記載不備および情報過多が原因で，新規の協力会社に正しく知識を伝達できていないとわかった．

知識不足起因の不具合を防ぐという観点では，仕様書やガイドラインに必要な情報が漏れなくすべて記載されていれば既存製品派生開発時の前提知識を新規の協力会社に伝達できるのではと考えた．しかし，十数年単位で保守を行っているソフトウェアの場合，プログラム設計レベルの仕様書までメンテナンスできていない．さらに，知識がすべて網羅された仕様書があったとしても担当者が改造に必要な情報を閲覧できるとは限らない．実際

に不具合を分析する中で，情報過多により必要な知識を見つけられずに不具合を埋め込んでしまったケースがある．そこで，発生した不具合から開発時に必要な知識を導き出し，協力会社に必要な知識を過不足なく伝える「機能一覧 Chips 紐付け表」を考案した．

以降，2 章の現状分析では協力会社との協業で発生している不具合の分析を行い，先行研究の有無を示す．3 章の解決策では，不具合から導き出した知識をドキュメントに整理し，協力会社に過不足なく知識を伝達する仕組みを示す．4 章の検証結果では，3 章で示した解決策を用いることで，不具合の混入をどの程度防止できるかを検証した結果を示す．5 章にはまとめと今後の課題を示す．

2.　現状分析

2.1　協力会社の不具合分析とその原因

2.1.1 不具合分析

新規の協力会社との開発でどのような不具合が発生しているか把握するため，依頼しているプログラム設計・実装・単体テスト工程を原因とする不具合を抽出し，分析した．

抽出した不具合(全 60 件)について原因の分類と件数を分析した結果を図 1 に示す．図 1 における分類は，「知識不足（調査・設計といった各作業の前提知識が欠落した)」，「設計不備（前提知識を持つが要求された追加・変更機能のプログラム設計において設計誤りやパターン漏れ･変更漏れが発生した)」，「影響調査不足（前提知識を持つが要求された追加・変更機能の影響を受けて想定外の機能で不具合を起こした)」，「実装誤り（単純ミスや実装技術不足によりプログラム設計の通りに実装できていない)」,「その他」の 5 つである．

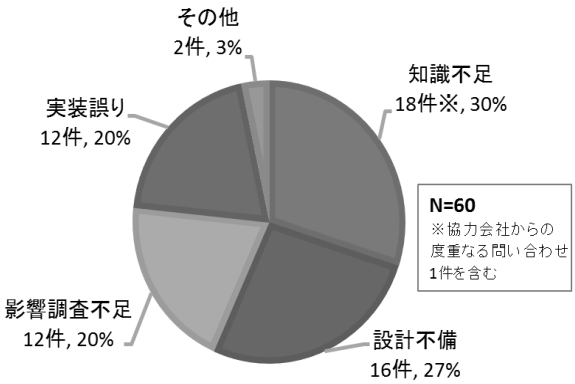

図 1　不具合の原因

分析した結果，不具合の原因件数について,「その他」以外は大きな差はない．不具合への対応策を検討すると,「知識不足」は，作業プロセスの見直しやスキルアップを要する「設計不備」･「影響調査不足」･「実装誤り」と比べ，新規の協力会社に不足している前提知識を伝達するだけで不具合の発生を防ぐことができ，比較的容易に実施可能かつ，短期間で品質の改善（不具合の発生防止）につながると考えられる．また，顧客視点で考えると，新規開発から製品の開発をベンダーに任しているため知識の継承は出来て当たり前のことであり，知識不足起因の不具合といった初歩的なミスを犯すことは顧客からの信頼を失うため，ここについては特に対策を打ちたい．

以上のことから，本研究では新規の協力会社の知識不足に起因する不具合に着目した．

2.1.2 協力会社に不足していた知識の分析

分析した不具合事例において新規の協力会社に不足していた知識を分類した結果を表 1 に示す．知識とそれに対応する不具合の事例を挙げると，例えば「用語」では,「ある製品でアンテナというと論理アンテナと物理アンテナの 2 つの解釈があり，大半は論理アンテナの意味で用いられている．派生開発で他製品に移植する際に物理アンテナとして読み違えてしてしまい，不具合が発生した」という事例がある．また,「開発環境の使い方」では,「テストで一部の機能を実行する際には，既存プログラムをテスト用に変更しなければならない．このことを知らなかったため，テストを正しく行うことができなかった」という事例がある．これらの事例の知識は，既存の協力会社であればこれまでの開発経験から補っていると考えられる．

一方，新規の協力会社は改造する製品の知識を持っておらず，膨大な既存資料から読み取る・ベンダーに問い合わせるといった方法で知識を補わなければならないが，派生開発の限られた時間の中，以上のことを行うのは難しい．そこで，新規の協力会社に不足している知識を効率良く補う事で，新規の協力会社の開発品質を既存の協力会社の品質に近づけることを期待できると考えた．

表 1　新規の協力会社で不足した知識

不足していた知識	件数
用語	3
性能要件	2
テーブル構造（データの持ち方）	5
プログラム設計ルール	4
コーディングルール	3
開発環境の使い方	1

2.2　課題

2.1 章で分析した通り，知識不足に起因する不具合は知識が正しく伝達できていれば発生を防ぐことができた不具合である．そこで知識の伝達方法について調査を行った．新規の協力会社への知識の伝達では，ドキュメント類（プロセス系，仕様書系，ガイドライン系）の提供を行い，それを補足するために，口伝による教育（OJT，Q&A，レビュー，ソフトウェア構造の説明会）もあわせて行っていた．そこで，知識不足に起因する不具合（18 件）について不足した前提知識のドキュメントへの記載状況を調査した結果，表 2 の結果を得た．

表 2 より，協力会社に伝わらなかった前提知識の多くは仕様書・ガイドに「記載がない」ことが原因と分かった．この理由は，用語や非機能要件など仕様に至る背景の記載先として適切なドキュメントが存在しないためであった．記載がない場合は，口伝によって補われていた場合もあるが，補うことを失念すると不具合として表出してしまう．「記載はあるが見つけづらい」については，膨大なドキュメントの中から今回の改修に必要な知識を適切に発見できなかったことで発生している．新規開発とは異なり設計を通じて全体理解はできず，派生開発の限られた時間の中で新規開発と同様の体系の大量ドキュメントから必要な知識を漏れなく発見することは難しいと考えられる．「わかりにくい」については，日本語の書き方や表現，読み手の経験に起因すると思われる．

表 2　伝わらなかった前提知識の記載状況

不足していた前提知識のドキュメント記載状況	件数
記載がない（欠落）	10
記載はあるが見つけづらい（過多）	4
記載内容がわかりにくい	4

そこで本研究では，新規の協力会社に開発を依頼する際に品質を安定させるため，派生開発に必要な知識を新規の協力会社に漏れなく情報過多にならないように伝え，知識の伝達不備に起因する不具合の発生を防ぐことを課題とした．

2.3　先行研究

本研究の課題の解決策として，新規の担当者に知識伝達する方法を扱っている先行研究が適用できないかを調査した．

まず，熟練者の頭の中にある仕様や設計の理由を抽出する手法の研究がある[1]．この研究は，熟練者の頭の中だけに存在し，後任者へ引き継がれることのない設計や仕様の理由等の既存製品派生開発時の前提知識を派生開発のプロセスの中で機械的に抽出しているが，ドキュメント記載の不足を補い，読み手が改造に必要な情報を見つけ易くしようとする本研究とは異なる．

次に，ドメイン知識を既存のドキュメントや熟練者から抽出・再構築し，メンバにドメイン知識（製品の機能や開発背景など）を伝達する手法を提案する研究がある[2]．この手法は，大量のドキュメントや熟練者が持つドメイン知識を抽出し漏れなくドキュメント化することを実現するが，プログラム設計や実装の知識を対象としない点で本研究と異なる．

最後に，開発者全員が製品に対して無知見である場合に，調査プロセスを定義し変更漏れや誤りを防ぐ研究[3]があるが，ベンダーの知見を効率良く伝えることのできる本研究と異なる．

上記の結果から新規の協力会社に開発を依頼する際，既存製品派生開発時の前提知識を協力会社に過不足なく伝え，品質を安定させる方法を提案する．

3. 解決策

3.1 解決の方針

2 章で分析したとおり，新規の協力会社との協業で品質の安定を図るには，既存製品派生開発時の前提知識を漏れなくかつ情報過多にならないよう適切に伝達する必要がある．以下に，その解決方針を示す．

2 章で示した「記載がない」・「わかりにくい」といった原因に対しては，前提知識をまとめたドキュメント(以降， Chips と呼称）を新しく作成，または既存の Chips を改善することで対応できる．しかし,「記載はあるが見つけづらい」という原因に対しては改善を行うことができない．協力会社に不足している知識を補うために新しい Chips を次々に作成した場合，参照すべき Chips の全体量が増え，どれを参照すべきかわからず,「記載はあるが見つけづらい」といったことが原因の不具合が増えてしまうことが考えられる．

そこで，どの Chips を参照すべきかをベンダーが明示する仕組みとして，本研究では,「機能一覧 Chips 紐付け表」を考案した．この表を利用することで，協力会社が発生させた「記載はあるが見つけづらい」が原因の不具合に対しては改善を行うことができる.Chips の伝達方法について，3.2 章で提示する．続いて 3.3 章にて，機能一覧 Chips 紐付け表のメンテナンス方法を提示する．メンテナンスをしていくことで,「記載がない」知識をドキュメントに落とし込むことができる．最後に 3.4 章にて，機能一覧 Chips 紐付け表による知識伝達方法の改善手順を紹介する．

3.2 「機能一覧 Chips 紐付け表」の概要

新規開発や派生開発を行っていく中で，ベンダーはその製品の派生開発に必要な Chips をガイドラインや仕様書という形で管理している．その情報をそのまますべて新規の協力会社に渡しても情報が多く，協力会社が派生開発の際に必要な情報を参照することは難しいため,「記載はあるが見つけづらい」といった事象が発生してしまう.そこで本研究では，ベンダーが派生開発に必要な Chips をテーラリングして，新規の協力会社に提供する必要があると考えた．その方法として機能一覧 Chips 紐付け表を提案する．

機能	Chips					
	テーブル関連図.xlsx	DB参照規約（コーディング標準書.xlsx 4.1章）	DB更新規約（コーディング標準書.xlsx 4.2章）	Web画面作成規約（コーディング標準書.xlsx 3章）	…	…
ログイン		○		○		
条件検索	○	○		○		
検索結果一覧	○	○		○		
商品情報詳細	○	○		○		
商品情報ダウンロード	○	○		○		
商品情報Webメンテナンス		○	○	○		
商品情報ファイルアップロード			○	○		
画像登録ポップアップ			○	○		
メーリングリスト登録			○	○		
…						
…						

図 2 機能一覧 Chips 紐付け表（マスター）

機能一覧 Chips 紐付け表は，2 種類作成する．機能ごとに必要な Chips を紐付けたマスターとなる機能一覧 Chips 紐付け表（マスター）と協力会社に依頼する派生開発案件用にテーラリングを行った機能一覧 Chips 紐付け表（テーラリング済み）の 2 種類である．

最初にベンダーは，機能一覧 Chips 紐付け表（マスター）の作成を行う．図 2 のように

既存ドキュメントとして整備されている機能の一覧表に Chips 一覧を追記する．そして，各機能の改修時に必要な Chips と機能の交点に〇印を記載する.この表を参照することで，新規の協力会社でも各機能の派生開発時に参照すべき Chips を簡単に知ることができる．

ただし，案件によっては，必ずしもすべての Chips を使用する必要はない．例えば，条件検索の Chips には，Web 画面作成規約と DB 更新規約があったとする．今回の改修要望は Web デザインの変更だった場合，DB 更新規約を参照する必要はない．機能一覧 Chips 紐付け表（マスター）をテーラリングし，本当に必要な Chips を協力会社に提供する必要がある．そこで，図 3 のようなテーラリングを行った機能一覧 Chips 紐付け表を作成する．

機能	派生開発要求			Chips					
	表示項目の入れ替え	参照機能のHTML5対応	商品登録チェック修正	テーブル関連図.xlsx	DB参照規約（コーディング標準書.xlsx 4.1章）	DB更新規約（コーディング標準書.xlsx 4.2章）	Web画面作成規約（コーディング標準書.xlsx 3章）	…	…
ログイン		〇			〇		◎		
条件検索		〇		〇	〇		◎		
検索結果一覧		〇		〇	〇		◎		
商品情報詳細	〇	〇		〇	◎		◎		
商品情報ダウンロード				〇	〇		〇		
商品情報Webメンテナンス	〇		〇		◎	◎	◎		
商品情報ファイルアップロード			〇			◎	〇		
画像登録ポップアップ			〇			◎	〇		
メーリングリスト登録						〇	〇		
…									
…									

図 3　機能一覧 Chips 紐付け表（テーラリング済み）

機能一覧 Chips 紐付け表（テーラリング済み）は，まず機能一覧 Chips 紐付け表（マスター）に今回の派生開発の要求一覧を追記する．記載する要求は，各ステークホルダと合意が取れドキュメントにまで落としこむ必要がある．そして，要求ごとに対象となる機能の交点に〇印を記載する．それにより，機能ごとにどのような派生開発の要求があるか明示でき，今回の派生開発に必要な Chips の判断が容易となる．例えば，要求が Web デザインの改修だった場合，Web 画面作成規約の Chips は必要だが，DB 更新規約の Chips は不要となるといった判断が容易になる．要求を加味し，今回の派生開発に必要な Chips を〇印から◎印に変更することで，機能一覧 Chips 紐付け表（テーラリング済み）は完成する．

従来，協力会社は派生開発に必要な知識を全ての Chips を参照したり，ベンダーに問い合わせたりすることで取得していた．機能一覧 Chips 紐付け表を横列に見ることで，機能ごとの改修に必要な知識をベンダーに問い合わせる必要なく，また全ての既存ドキュメントを参照することなく，的確に取得することができる．◎印となっている Chips は，今回の派生開発に必須な前提知識，〇印のままの Chips は，今回の派生開発に直接必要ではないが，機能把握の為に必要な Chips である．これにより，協力会社の学習の容易化を図ることができ,ベンダーにとっても協力会社からの問い合わせが減ることはメリットになる．

機能一覧 Chips 紐付け表を使用することで,「記載はあるが見つけづらい」という問題は解決できる．しかし,「記載がない」,「わかりにくい」といった問題は解決することができない．この問題は，機能一覧 Chips 紐付け表をメンテナンスすることで解決できる．

3.3　「機能一覧 Chips 紐付け表」のメンテナンス方法

「記載がない」,「わかりにくい」といった問題を解決するには，機能一覧 Chips 紐付け表（マスター）をメンテナンスする必要がある．そのタイミングとメンテナンス方法を以下に記載する．

<1>[機能の追加時]新しい機能を作成した際に発生する．新しい機能を機能一覧 Chips 紐付け表（マスター）に追記し，それに紐づく Chips に〇印を記載する．

<2>[機能の改修時]既存機能に改修があった場合に発生する．改修された機能に紐づく Chips を新たに追加，削除を行う．たとえば，今までは DB 参照しかない機能だったが，新

しく DB 登録の機能が追加された場合，DB への登録規約が新しく参照すべき Chips として追加される．

〈3〉[不具合の分析時]ベンダーが受け入れテストを実施し，不具合が発覚した際に発生する．その不具合の原因を以下の 3 パターンに分析し，それぞれ対応を行う．

〈3-1〉「記載がない」

新たに記載がない派生開発の前提知識を Chips に書き起こす．その Chips を，機能一覧 Chips 紐付け表に追記し，機能との紐付けを行う．

〈3-2〉「記載はあるが見つけづらい」

機能一覧 Chips 紐付け表に Chips が記載されていなければ追記して，機能との紐付けを行う．Chips が記載されている場合は，機能との紐付けに不備があるので，再度ベンダーにて紐付けの確認を行う．

〈3-3〉「わかりにくい」

既存のドキュメントを修正し，協力会社にわかり易くなったか確認した上で他に同様の記述がないか横展開チェックを行って Chips を改善する．

〈4〉[協力会社からの問い合わせ発生時]協力会社から問い合わせがあった場合も不具合発生時と同様に分析を行い，問い合わせ内容に応じて〈3-1〉～〈3-3〉と同様の対応を行う．

3.4 「機能一覧 Chips 紐付け表」による知識伝達方法の改善手順

協力会社に過不足なく知識を伝達するために，機能一覧 Chips 紐付け表の運用手順について説明する．提案手法は，開発案件を実施する度に図 4 に示す手順を繰り返す．各手順の詳細を下記に示す．

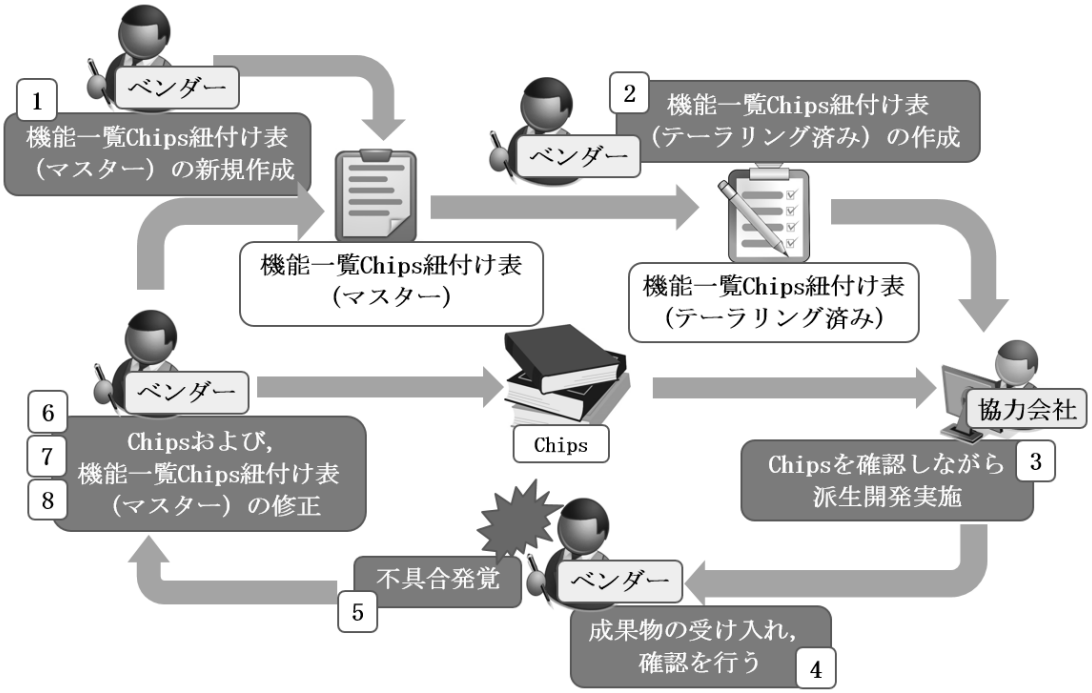

図 4 「機能一覧 Chips 紐付け表」の使用による改善手順

1. ベンダーが，機能一覧 Chips 紐付け表（マスター）の作成を行う．(3.3 章の〈1〉，〈2〉を実施する．)
2. 案件発生時にベンダーは，機能一覧 Chips 紐付け表（マスター）から今回の派生開発に必要な Chips をテーラリングした，機能一覧 Chips 紐付け表（テーラリング済み）を作成する．協力会社に派生開発対象プログラム，仕様書やガイドライン等の派生開発に必要なドキュメントと機能一覧 Chips 紐付け表(テーラリング済み)を提供する．

3. 協力会社は機能一覧 Chips 紐付け表（テーラリング済み）を参照し，派生開発を行うに当たり，必要な前提知識を確認しながら開発を行う．
4. ベンダーが協力会社の成果物を受け入れ，確認を行う．
5. 受け入れとリリース以降に不具合が発生．原因分析を行った結果，協力会社が派生開発に必要な前提知識を正しく認識していないことが原因と判明．
6. 上記不具合を「記載がない」,「記載はあるが見つけづらい」,「わかりにくい」に分類し，それぞれ対応を行う．(3.3 章の<3>を実施する.)
7. 協力会社から QA としてあがってきた問い合わせに対しても，不具合同様に機能一覧 Chips 紐付け表（マスター）のメンテナンスを行う．(3.3 章の<4>を実施する.)
8. 機能および，Chips の追加・変更・削除があった場合，機能一覧 Chips 紐付け表（マスター）のメンテナンスを行う．(3.3 章の<1><2>を実施する.)

再度，協力会社に派生開発を依頼するときは，2～8 を繰り返し実施する．このサイクルを繰り返すことで，機能一覧 Chips 紐付け表の品質が上がり，新規の協力会社にもベンダーが持っている派生開発に必要な前提知識を過不足なく提供することができる．

4. 解決策の検証

4.1 検証方法

既存製品派生開発時の前提知識をベンダーから新規の協力会社へ十分に伝達できていないため，既存の協力会社では起こりえない不具合が発生していた．この問題の解決策として機能一覧 Chips 紐付け表を考案し，そのメンテナンス方法と使用手順を 3 章で示した．

そこで，この解決策が有効であるか判断するために，すでに分析済みの派生開発の不具合(全 60 件)に対して効果があるか,次の点についてシミュレーションを行い,検証した．

(1) 協力会社起因の不具合は減るか．
(2) 機能一覧 Chips 紐付け表の作成・メンテナンス工数はどれくらいか．

4.2 検証結果

(1) 不具合の削減

機能一覧 Chips 紐付け表（テーラリング済み）を協力会社に提供することで，今回抽出した協力会社起因の不具合(全 60 件)が防げるか，協力会社に確認を取った．その結果「記載はあるが見つけづらい（過多)」が原因の不具合の発生を防ぐことができ,「わかりにくい」「記載がない」が原因の不具合は，一度目の発生を防ぐことはできないが，二度目の発生は防げるとわかった．数値として 18 件(30%)の不具合を防ぐことができた．

(2) 機能一覧 Chips 紐付け表の作成・メンテナンス工数

機能一覧 Chips 紐付け表の作成･メンテナンス工数は,担当するシステムや Chips の数,発生した不具合の件数により変化するため,信頼性の高い数値を提示することはできない．そのため，研究員が担当するシステムでのシミュレーションの一例を提示する．

（機能数：228　Chips 数：50　派生開発要求件数：15　）

機能一覧 Chips 紐付け表(マスター)の初期作成工数：20 時間
機能一覧 Chips 紐付け表(マスター)のメンテナンス工数：5 分　(1 機能追加ごとに)
機能一覧 Chips 紐付け表(マスター)のメンテナンス工数：25 分(1Chips 追加ごとに)
機能一覧 Chips 紐付け表(テーラリング済み)の作成工数：10 時間
(【内訳】　機能と要求の紐付け：6 時間 Chips の紐付け：4 時間)
※Chips の作成工数は含めない

4.3 考察

検証の結果，既存製品派生開発時の前提知識の Chips を機能一覧 Chips 紐付け表に集約し，変更依頼要求があった機能の変更に対して必要な Chips を一覧に整理して明確化することで，協力会社の学習の容易化を図ることができ，協力会社の設計および開発工程にお

いて品質と生産性向上に有益である事がわかった．また Chips は過去の不具合を元に作成するので新たな協力会社に過去と同じ不具合を再発させない効果がある．

まず，品質面については，機能一覧 Chips 紐付け表により現状発生している知識不足の不具合をすべて防げるとわかった．次に，生産性の面について考察する．シミュレーションを行ったプロジェクトにおいて，運用･保守段階で検出されたバグ対応の手戻り時間は 1 件あたり 20 時間というデータがあり，このプロジェクトでは 1 案件で 1 件程度の Chips 伝達不備の不具合が発生していた．よって，このプロジェクトに提案手法を初めて導入するときは，機能一覧 Chips 紐付け表の作成（20 時間）とテーラリング（10 時間）で合計 30 時間かかるため，初回は 10 時間のコスト増となる．しかし，2 回目以降の派生開発ではテーラリングの 10 時間で済みコスト減となる．ただし，Chips の作成工数は精査できておらず，作り方によってはコスト増となる可能性がある．そのため，Chips 作成時は，ベンダーの担当者や関係者の知恵を出し合い，既存の協力会社の開発担当者を投入するなどして，Chips 作成工数の洗い出しに時間をかけない工夫が必要である．

別の観点から考察すると，システムに必要な仕様書を完全に整備して知識を補う解決策も考えられる．しかし，シミュレーションを行ったシステムにおいてプログラム設計レベルの仕様書を整備するには，30 人月程度の工数を要するという見積結果があり，不具合の傾向をふまえた機能一覧 Chips 紐付け表の方が安価に効果を出せる．さらに，プログラム設計レベルの仕様書を完全に整備しても，情報過多のため新規の協力会社が必要な情報にたどりつけない可能性もある．以上より，過去の不具合から得た教訓とベンダーの経験を Chips･機能一覧 Chips 紐付け表に織り込むことを継続できれば，機能一覧 Chips 紐付け表は，新規の協力会社が引き起こす前提知識不足起因の不具合を防止するのに有効である．

5. まとめ

5.1 結論

機能一覧 Chips 紐付け表を利用することで，「見つけづらい」に起因する不具合を解決できる．初期の開発サイクルでは，過去の不具合に基づいた Chips が存在しないため「わかりにくい」「記載がない」といった知識不足をカバーすることは難しい．しかし機能一覧 Chips 紐付け表を作成，利用しながら開発サイクルをまわしていくことで，派生開発に必要な Chips を過不足なく伝えることができ，協力会社に依頼した成果物の品質向上につながり，ベンダーと新規の協力会社の知識の差を埋められることがわかった．

5.2 今後の課題

派生開発において新規の協力会社に開発を依頼する際に品質を安定させる目的で本研究を進めて来た．新規協力会社の不具合原因で，既存製品派生開発時の前提知識の伝達不備の次に,件数が多かった設計不備についても今後分析し問題解決を図る必要がある.また，既存製品派生開発時の前提知識の元となる Chips の抽出方法をはじめとし，今回の施策が発注するベンダーの効率を損なわずに運用でき，コストの面でも効果が得られるかを実際のプロジェクトで確認する必要がある．

参考文献

[1] 関野浩之，大坪智治，大内智之，後任担当者視点を取り入れた設計背景の形式知化による派生開発の品質向上策，SQiP 研究会分科会報告書，2011

[2] Noriko Iizumi，Tomoko Tomiyama，Atsuko Koizumi，Atsushi Takahashi，Utilization of Domain-Specific Knowledge for Quality Software Design，5th World Congress for Software Quality，2011

[3] 中井栄次，間瀬研二，古畑慶次，無知見プロジェクトに対する XDDP の適用，SQiP シンポジウム，2009