類似製品の並行派生開発における共通仕様の検出と共有方法

Methodology of Detecting and Sharing Common Specifications to Stake Holders in Developing Concurrent Derivative Similar Systems

ウイングアーク１ｓｔ株式会社 開発本部 品質統括部
WingArc1st Inc. Quality Management Dept.
○内藤 史郎 1)　星野 充史 2)
○Shiro Naito　Atsushi Hoshino

Abstract

In modification type software developments which develop concurrent multi-systems that have same existing code, there is high possibility that leaks or misunderstandings about common specifications will affect other systems. So it has severe risks.

In previous research, there is a method of establishing expert divisions that organize common specifications which are provided by the consumers of each systems to manage concurrent multi-system developments and to mitigate the risks, but in small-sized concurrent multi-system developments, there were severe cost restrictions, so we could not adopt the method.

So the authors have devised a new methodology of which common specifications can be detected and shared between stake holders and it can be applied to small-sized modification types of concurrent multi-system developments instead of establishing expert divisions. And we named it R^2SPL.

As a result, common specifications can be centralized and specifications can be transferred correctly. So more than half of software anomalies caused by leaks and misunderstandings about common specifications have been mitigated.

1. はじめに

急速な市場の変動や IT 技術の進展にともない，顧客の経営環境も刻々と変化してきた．その結果，競合他社にない優位性を持つ製品をいち早く開発し市場投入することが求められるようになり，競争がいっそう激しさを増してきている．この潮流の中で，競合他社に無い優位性を実現するために求められる顧客からの要求も高度化し，開発規模は増加傾向にあるが，開発期間は短縮される傾向にある．また，この限られた期間内に品質も保証しなければならない．

この熾烈な競争に勝ち残るため，製品の市場投入を短縮するさまざまな取り組みが行われている．複数の類似製品やシリーズ製品を開発する組織におけるこれらの取り組みの一つとして，複数製品のソフトウェアを一元化して一つの実行オブジェクトを生成し，環境変数を

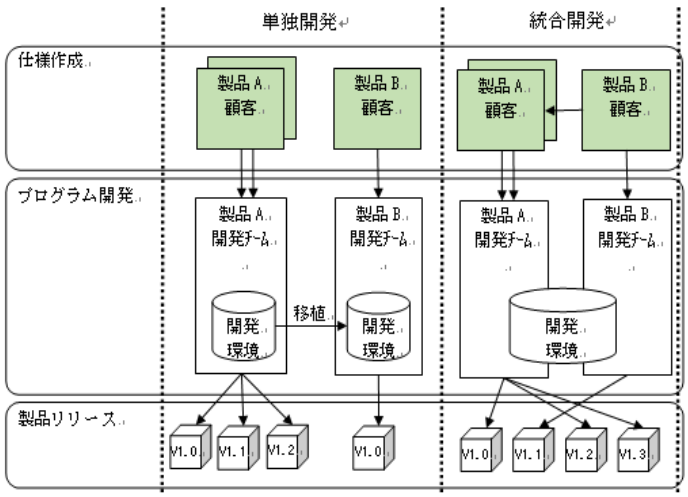

図 1 単独開発と統合開発の開発環境の違い

1) ウイングアーク１ｓｔ株式会社
Quality Management Dept., WingArc1st Inc.
東京都渋谷区桜丘町 20−1 渋谷インフォスタワー Tel: 03-5962-7402　e-mail:naitoh.s@wingarc.com
20-1, Sakuragaokacho, Shibuya, Tokyo Japan
2)アンリツエンジニアリング株式会社
Network System Dept., Anritsu Engineering Inc.

変えると特定製品として動作する開発方法（以下，統合開発という）を採用することがある．各製品を独立した開発環境（以下，単独開発という）で開発するよりも効率がよいからである．筆者らの組織も，このような統合開発を導入した．それぞれの開発環境の違いを図１に示す．

図１のような統合開発において，ある製品の変更がその他の製品へ意図しない影響を与える新たな問題を引き起こすことがある．同じ母体の複数製品に対し，相互への影響を適切に把握しないまま変更を加えることがあるからである．この問題は，仕様提示者と獲得者が多対多の複雑な構図で非同期・不定期に仕様提示が行われることと，担当製品以外の知識の欠如により，類似製品で同じ機能に対する仕様[1]（以下，共通仕様という）に気付きにくいこと，顧客から提示された仕様の解釈を開発チーム間で等しく共有できる工夫の欠如に起因することが多い．

そこで，共通仕様を一元化して製品知識の欠如を補完し，各製品の開発チームが等しい仕様を共有し，さらにはその成果を製品開発計画へフィードバックを可能にする仕組み，R²SPL(Requirement Repository for Software Product Line)を考案した．この適用により，共通仕様漏れや共通仕様伝達ミスによる品質リスクを低減することができた．本論はこの R²SPL 適用結果について述べる．

2. 複雑な仕様提示者と仕様獲得者の構図と統合開発

2.1 統合開発における開発背景

本論で対象とする統合開発と単独開発の特徴を表１にまとめる．

表１ 単独開発と統合開発の特徴

	仕様提示者	仕様提示方法	変更依頼文書	開発規模	開発工程
単独開発	当該製品の顧客（1～2 人）	まとめて提示	当該製品開発で決めた文書	単独開発（0.5～24 人月）	仕様受け入れ 設計 実装 テスト
統合開発	全製品の顧客（最大 8 人）	非同期・不定期	さまざまな文書	最大 4 プロジェクトの並行開発（0.5～40 人月）	同上

表１が示すように，単独開発は顧客と開発者が当該製品の中で閉じる関係であったが，統合開発は製品の枠を超えて仕様提示者と獲得者が多対多になり，不定期・非同期に変更依頼が発生する複雑な構図となった．このような複雑な構図においても，単独開発では各製品で開発環境が独立し，それぞれの開発チームが顧客から提示された仕様に対して開発するため，変更による直接的な影響を他製品に与えなかった．しかしながら，統合開発で単独開発と同様に仕様の受け入れを各製品で行った結果，ある製品に対する変更依頼が，他製品に与える影響を十分に把握できないことがあった．また，変更依頼が他製品へ影響することに気付いても，さまざまな方法で各製品の開発チームへ伝達するため情報が正しく伝わらない問題も発生した．

これらのことから，統合開発では以下を実現する必要があるといえる．

(1) 他の製品に与える共通仕様を検知する

(2) 共通仕様を正確に共有する

2.2 不具合事例の傾向

前項で述べた統合開発における課題について，実際の不具合事例からその関係性を分析した．統合開発の総合テスト以降に発見された不具合事例 318 件を対象に，不具合混入工程・仕様の分類・不具合要因別の順にふるいに掛け，その結果を表２にまとめた．

---

[1] 共通仕様は，複数の類似製品に存在する同一の機能を意味する．ただし，製品のハードウェア構成や扱うデータの違いから詳細仕様が異なる場合がある．

表 2　総合テスト・評価で発見された不具合の要因別内訳

総合テスト・評価における不具合件数	不具合混入工程	仕様の分類	不具合要因別
318 件	設計以降：　224 件	--	--
	仕様受入：　94 件	個別仕様[2]：　28 件	--
		共通仕様：　66 件	仕様記述漏れ[3]：　37 件 仕様不明瞭ミス[4]：　29 件

表 2 が示すように，複数製品に影響を及ぼした共通仕様の不具合は 66 件で，全体の約 20%に相当し，その内訳は，仕様記述漏れと仕様不明瞭ミスによるものであった．そこで，共通仕様に起因する問題の主要因を把握するため，66 件の不具合原因の傾向を分類するとともに，どのような情報が不足して発生したのかを整理し，その結果を表 3 にまとめた．

表 3　共通仕様に起因する不具合原因の傾向

ID	不具合原因の傾向	原因分類	不足した情報
#1	変更依頼に共通仕様の記述が漏れているため共通仕様に気付かない場合．	仕様記述漏れ	・共通仕様の記述 ・類似製品の知見
#2	開発時期の違いで共通仕様に気付かない場合．	仕様記述漏れ	・共通仕様の記述
#3	変更依頼の表現方法の問題による仕様の解釈ミス．	仕様不明瞭ミス	・等しく共有できる表現で記述した共通仕様
#4	他製品の開発チームへ伝達する際の共通仕様の表現方法の問題による仕様の解釈ミス．	仕様不明瞭ミス	・等しく共有できる表現で記述した共通仕様

表 3 が示すように，共通仕様の記述漏れは変更依頼に仕様記述が漏れることと，共通仕様であるという情報が不足して発生していた．変更依頼の仕様記述漏れは単独開発においても発生していたが，当該製品の開発経験豊富な開発者が仕様漏れに気付いて対処することが多かった．つまり，統合開発の背景には類似製品の知識に乏しいという現状がある．このため，仕様の受け入れ工程で共通仕様か個別仕様かを判断するとき，当該製品の知識で仕様を捉えるため個別仕様と判断されやすい．つまり，並行開発する製品と“共通仕様かもしれない”，または後発の製品開発が立ち上がったときに“共通仕様になるかもしれない”という状態を初期判断時に残していないため，共通仕様の漏れが起きやすいといえる．

2.3　先行研究における方法論

このような類似製品の並行開発に関する先行研究として次の 2 つがある．

まず，専門組織を構築して要件定義・開発・現地試験まで複数の開発チームをマネジメントし，仕様やプログラムコードの一元管理を実現する手法である[1]．この手法が効果的なプロジェクトの特徴は，「類似製品の並行開発が 2，3 以上で，全体開発期間が 1 製品分の開発期間より若干長い程度」と論じている．この研究は，最大 10 のプロジェクトが，それぞれ 400～500 人月の類似製品を新規に並行開発する規模である．これに対し，統合開発は最大 4 のプロジェクトが，0.5～40 人月の類似製品を並行して派生開発する規模で，全体の開発期間が 1 製品分の開発期間より大幅に長い場合が多い．また，同様の手法で専門組織を構築するには，製品知識を有する組織を新たに育成するか，または開発チームから開発者を引き抜く必要があり，現組織において新しい

[2] 個別仕様に起因する不具合は，当該製品にのみ存在する機能の変更により発生したものを意味する．

[3] 仕様記述漏れは，複数製品で共通の機能仕様が漏れる場合と，機能に含まれる一部の仕様が漏れる場合，および性能に関わる仕様が漏れる場合を意味する．

[4] 仕様不明瞭ミスは，仕様の文章表現の問題による開発者の解釈の誤りを意味する．

組織や開発チームを柔軟に構築／解散することはコスト上難しい．このため，この手法をそのまま適用することが難しいと判断した．

次に，変更依頼に対する関連システムへの影響を事前に検知する手法がある[2]．これには，顧客からの当該システムに対する変更依頼の背景に，他システムの変更依頼が含まれていることに気付かない問題に対し，専門組織を構築せずに解決する手法が論じられている．この手法は，相互に通信するシステムにおけるインタフェースの相互影響を検知する手法であり，独立したシステムにおける共通資産の相互影響を検知できない．このため，統合開発へそのまま適用することが難しいと判断した．しかしながら，専門組織の代替となる手法で顧客から提示された仕様の影響範囲を検知している点は利用可能であると考えた．

3. 制約条件の中で共通仕様を検知し展開する方法論

2.1 で述べたように，共通の母体で類似製品を並行開発する場合，他製品に影響を与えやすい共通仕様を検知すること，その情報を正しく伝達し，相互に分かり合えることが重要である．そして，2.3 で述べた制約条件の範囲内でなければならない．

そこで，筆者らは専門組織の代替となる新しい方法を考案した．この方法は，共通仕様に気付きやすくする仕組みと，それを共通の伝達手段で展開する仕組みから構成される．この仕組みを PFD（Process Flow Diagram）[5]で示したものが図 2 である．なお，図中の網掛けしたプロセス 2 と 3 は，作業を自動化するプロセスを示す．本章は，これらの仕組みについて解説する．

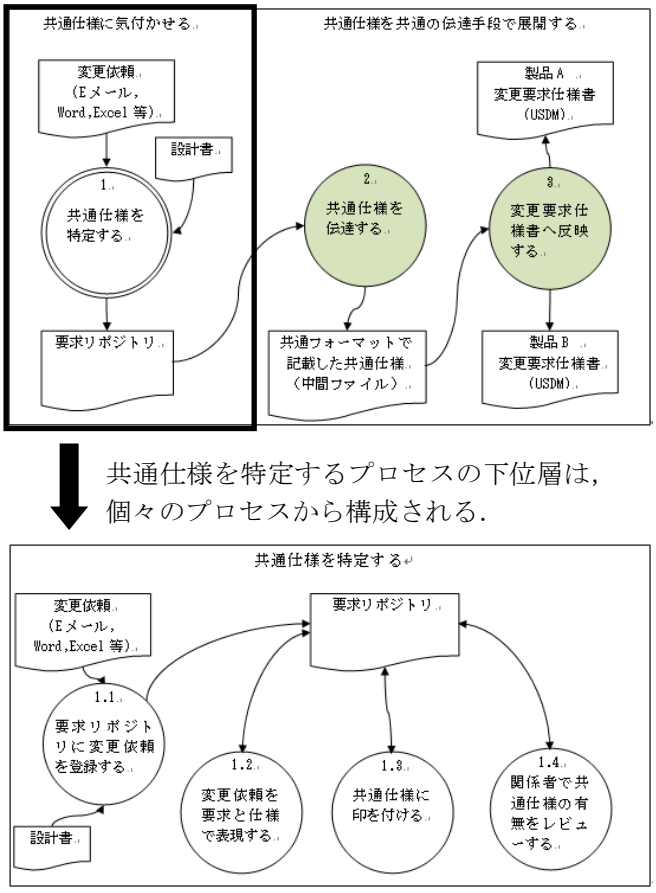

図 2 2 つの仕組みによる課題解決の PFD

3.1 共通仕様に気づきやすくする仕組み

3.1.1 変更依頼の一元化

共通仕様を検出するには，まず各製品の顧客と開発チームにおける多対多の複雑な構図を一元化し，単純にする必要がある．多対多の複雑な構図で非同期に情報通信が行われると，その管理も複雑になり他の製品の適切な情報を得ることが難しくなるからである．先に述べたように，専門組織によって共通仕様の一元管理は行えるが，コスト上の制限で適用できない．そこで，これと同等の機能を持たせるため，専門組織の代わりに「要求リポジトリ」という仕組みを考案した．要求リポジトリは，不定期で非同期にコミットされるソースコードを一元管理するよくあるリポジトリと同様に，複数製品を跨いだ共通仕様を含む変更依頼をファイルとして一元化する．また，要求リポジトリは，最初に開発を始めた開発チームが作成し，変更依頼の度に更新する．そして，途中で他製品の開発が開始された際も同じファイルを共有し，並行開発されるすべての製品開発が完了するまで使用する．これにより，変更依頼の一元化は行えるようになると考えた．

[5] Data Flow Diagram をベースに作られたもので，主に開発プロセスをモデリングするためのモデル図である．

3.1.2　共通仕様と個別仕様，共通仕様の可能性がある仕様の分類

各製品の開発チームが要求リポジトリを参照した際，記載された仕様が共通仕様であるか個別仕様であるかの判別を容易にするには，それぞれの仕様に識別子があるとよい．ただし，要求リポジトリへ仕様記述を行う開発者は，必ずしも他製品の仕様に精通しているとは限らないため，記述しようとしている仕様が個別仕様であることに確証を持てない場合もある．このような場合に誤って個別仕様と判断しないようにするため，共通仕様と個別仕様の他に共通仕様の可能性がある仕様を識別できる必要があると考えた．そこで，要求リポジトリへの仕様記述時に以下に該当する一意の識別子を付与することをルールとした．

(1) 共通仕様であることが明らかであるもの

(2) 「(1)」以外の個別仕様であることが明らかではないもの（以下，保留仕様という）

これにより，各製品の開発チームは要求リポジトリを参照することで，担当製品における実装工程前に共通仕様と保留仕様を識別できると考えた．

3.1.3　保留仕様から共通仕様を抽出

保留仕様には共通仕様を含むことがある．これを特定しないまま実装すると，他の製品へ意図しない影響を及ぼすことがある．したがって，実装工程が開始される前に保留仕様は共通仕様と個別仕様に確実に分類されている必要がある．開発チームは他の製品の仕様に精通しているとは限らないため，この分類には他製品の開発者によるレビューを要する．そこで，要求リポジトリへの仕様記述後，保留仕様を共通仕様と個別仕様に分離するため，各製品の開発者によるレビュープロセスを導入した．これにより，開発チームが実装開始前に確実に共通仕様を識別できると考えた．

3.2　共通仕様を共通の伝達手段で展開する仕組み

前項の方法により共通仕様の検出が行われても，開発者に正しく伝達されなければ正しく実装されず，後工程で不具合を来す可能性がある．仕様がさまざまな文書で任意の書式によって記載されていると，さらにそのリスクは高まる．したがって，仕様とそれに該当する要求の関連性が一別できる標準文書に記載するべきである[3]．また，統合開発は短納期の制約がある派生開発である．清水[4]は，「派生開発の最大の制約は『部分理解』のもとで作業しなければならないこと」と述べている．このため，この開発者の思い込みや勘違いを始めとする部分理解の罠に陥らないような標準文書であることが望ましい．さらに，それを作成するためのコストや手順が少ないとよい．そこで，図 3 のように，これらの特徴に有効とされる USDM(Universal Specification Describing Manner)を採用し，要求リポジトリのデータを入力として，これを自動生成するツールを開発することにした．これにより，要求と仕様の表現が鮮明になり，仕様表現の均質化ができると考えた．また，USDM を自動生成し各製品の開発者がレビューすることで，開発者の負担軽減と部分理解の罠を回避できると考えた．

筆者らは，この一連の仕組みにより，要求リポジトリに蓄積させた共通仕様が，SPL(Software Product Line)開発[5]におけるコア資産[6]の導出を可能にすると考え，R²SPL と命名した．

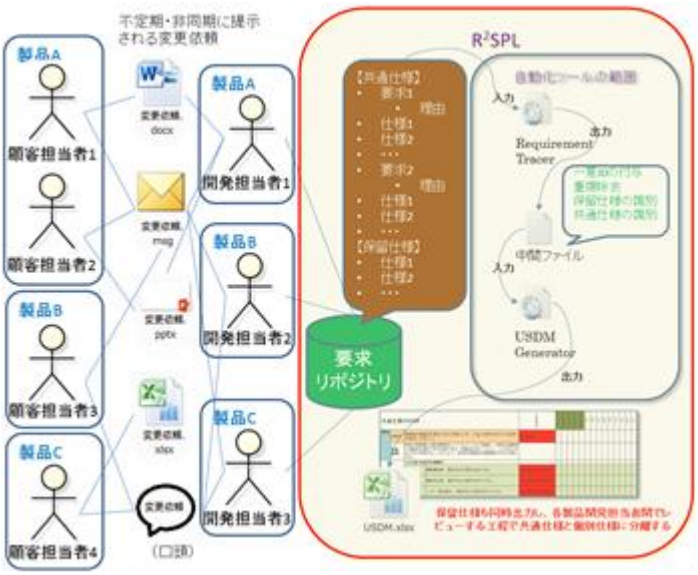

図 3 R²SPL の俯瞰図

[6] 過去の開発経験などを通じて蓄積され，再利用のために整備されたソフトウェア資産．

4. R²SPL の検証

4.1 検証方法

2.2 で述べた過去の並行派生開発事例（以下プロジェクト X という）に対するシミュレーションによる適用，および 2 つの類似製品の並行派生開発（以下プロジェクト Y という）に対する実践適用にて効果を測定した．具体的な検証方法は以下の通りである．

適用手順

(1) 要求リポジトリを共有し，不定期・非同期に提示される変更依頼を登録する．このとき，共通仕様を示す識別子と変更依頼の開発対象の製品識別子を付ける．
(2) 仕様受け入れ工程のレビューは，各製品の開発者がレビューアとして参加する．
(3) レビュー指摘修正後，ツールを使って要求リポジトリから「要求」「仕様」「理由や背景」を共通仕様識別子と各製品識別子に分類して出力し，その情報を各開発チームへ伝達する．
(4) 各開発チームは伝達された情報を，ツールを使って USDM へ入力する．

プロジェクト X に対する効果測定方法

(1) 総合テスト結果から，仕様受け入れ工程で摘出できた共通仕様の漏れと，摘出できなかった共通仕様の漏れの件数を測定する．
(2) 摘出できた共通仕様の伝達ミスによる仕様解釈ミスと，摘出できなかった共通仕様の伝達ミスによる仕様解釈ミスの件数を測定する．
(3) (1)と(2)のうち，摘出できたものを R²SPL の構成要素による効果に分類する．

プロジェクト Y に対する効果測定方法

(1) 総合テストで発見された，共通仕様に関連する不具合を測定する．
(2) 開発完了後に開発コストデータを収集し，R²SPL の適用によるコストの増減を比較する．

4.2 検証結果

前項で述べたプロジェクト X への R²SPL の適用結果を表 4 に示す．また，摘出した不具合 62 件について，R²SPL の 3 つの構成要素のうちどの構成要素により摘出できたかを表 5 に示す．

表 4 プロジェクト X における共通仕様の摘出件数と不具合要因別の摘出率

内訳 不具合要因	不具合事例	手法適用	
	件数	摘出件数	摘出率
合計	107 件	62 件	57.9%
仕様記述漏れ	61 件	44 件	72.1%
仕様不明瞭ミス	46 件	18 件	39.1%

表 5 プロジェクト X における R²SPL の構成要素の摘出件数内訳

R²SPL の仕組み	R²SPL の構成要素	摘出件数
共通仕様に気付きやすくする仕組み	要求リポジトリによる効果	47 件
	保留仕様を許容した効果	6 件
共通の伝達手段で展開する仕組み	USDM の表現で展開した効果	9 件

表 4 が示すように,仕様の受け入れ工程で摘出できなかった共通仕様に起因する不具合の約 58%を摘出できた.特に,仕様記述漏れの摘出は 72%と高い結果が得られた.これらの摘出効果を R²SPL の構成要素で分類すると，表 5 に示したように要求リポジトリによる効果が最も大きいことがわかった.

次に，実践適用したプロジェクト Y の検証結果については，開発規模が 1 人月未満の小規模な派生開発であるものの，総合テストで発見された共通仕様の不具合は 1 件もなく，共通仕様の不具合を未然に防ぐことができた．これは，仕様受け入れ工程のレビューにて，他製品の開発者が共通仕様の漏れを指摘し，この時点で不具合を除去して USDM を展開したためである．また，プロ

ジェクト Y に対する開発コストの増減について分析した結果，開発コストを削減できる効果を得られた．各開発工程における開発コストの増減を表 6 にまとめる．

表 6 プロジェクト Y への手法適用による開発コストの影響

	仕様受け入れ工程	設計工程	実装工程	テスト工程
開発コスト増加	レビュー工数	なし	なし	なし
開発コスト削減	さまざまな伝達文書の作成工数	USDM 作成工数	なし	デバッグ工数

表 6 が示すように，仕様受け入れ工程のレビューでレビューアが増えたため，単独開発と比較してレビュー工数が増加した．一方，要求リポジトリから USDM を自動生成するツールの導入により，設計工程の生産性が向上して開発工数を削減できた．また，テスト工程で不具合が減少したことによる手戻り工数が減少した．工数の増加を 1 として，減少した工数は 2～2.5 であり，開発工数の削減にも効果が見られた．

4.3 考察

共通仕様に気付きやすくする仕組みは，プロジェクト X と Y の両方で仕様受け入れ工程における共通仕様の漏れを検出できることが分かった．特に共通仕様の漏れを防止する効果が高い．共通の伝達手段で展開する仕組みについても，共通仕様解釈ミスによる不具合が低減できたことから，一定の効果があった．これにより，2.2 で述べた統合開発時の共通仕様に起因する不具合の原因を解決し，手戻りを回避できたといえる．

R^2SPL の適用による開発コストの影響は，各製品の開発者によるレビュープロセスという新しい取り組みにより増加したものの，その適用効果で相殺もしくはそれ以上の効果を得られる可能性が高いことがわかった．そして，製品開発のライフサイクルで俯瞰すると，コスト削減効果の蓄積によりツール開発の初期投資も含めて開発コストの削減が可能になる．

このように，R^2SPL の実践適用を展開し始めた最中であるが，その一端として比較的小規模な類似製品の並行派生開発において，ソフトウェア品質と開発コストを両立できる有効な仕組みであるという検証結果が得られつつある．そして，この成果は過去の事例をシミュレーションしたプロジェクト X の結果に実践結果が伴うと期待する．そのためには，今後並行派生開発のプロジェクト数や開発期間の重複パターンを増やして，本検証で明確な効果を得られなかった保留仕様の効果や共通の伝達手段で展開する仕組みの効果を含めて検証することが必要である．

5. まとめ

5.1 組織の弱みを認識し強みに活かす仕組みが達成したこと

昨今のソフトウェア開発は，開発効率を上げるためにさまざまな取り組みがされている．その取り組みのひとつに，統合開発での派生開発を述べた．納期優先の派生開発は，対象領域や既存ソフトウェアの知識が必須であるにも関わらず，部分理解で開発され決められた品質も達成しなければならない．この制約の中で，他製品へ影響を与えやすい共通仕様を正しく認識するには，仕様提示者と仕様獲得者の複雑な構図による影響を排除する仕様受け入れ工程の一元化と，各製品の開発者の知識を集結させる必要がある．また，開発者へ仕様を正確に伝達する必要がある．

その手段として，要求リポジトリによる共通仕様の一元化と USDM による情報伝達を紹介した．要求リポジトリは，仕様提示者と仕様獲得者の多対多の構図の中でも共通仕様を一元化してくれる．また，一時的に保留仕様を許容し開発チーム間でレビューすることで，一人の開発者では気付けなかった共通仕様に気付く機会を与えてくれる．次いで，USDM 自動生成による情報伝達方法について説明した．これは，開発者の思い込みや勘違いによる情報伝達ミスの低減に必ず効果が出る．自動生成による情報伝達のため，作業ミスが発生することもない．

飯泉[6]は，ソフトウェアの品質を作り込むには，自組織の強み・弱みを設計者自身が客観的に把握することが重要だという．一つの製品の設計者では共通仕様の判別が行えない弱みを設計者自身が客観的に把握した上で保留仕様の判断を許容し，他製品の設計者が支え合う強みを持つ組

織を作ることで，高品質であるという付加価値を持つ競争力のある製品を世に出すことができ，組織の持続力に繋がる．

5.2　品質改善の枠組みを超える R²SPL の展望

統合開発を導入することで発生した課題について，2 つの仕組みを組み合わせた手法を考案し品質問題を改善できた．この手法を継続することによって要求リポジトリに共通仕様と個別仕様が蓄積され，これらの仕様と製品の機能との紐付けを行うことができれば，共通のソフトウェア部品の数量や規模が導き出せる．

本論で述べた統合開発は，開発効率を上げるために導入された．その背景にはコストと納期を圧縮したいという組織からの暗黙の要請がある．共通仕様と製品の機能，開発実績との紐付けを行えるよう R²SPL を拡張することで，SPL 開発における要求差分の実現工数を見積もる上での入力情報と成り得る．そして，類似製品におけるコア資産の導出とコア資産からの製品用資産の導出を行い，この資産を再利用する製品サイクルによって，生産性が向上されるであろう．

このように，R²SPL はソフトウェア品質改善の枠組みを超え，予算，規模，計画を始めとする価値のある情報を開発ライフサイクル全体に提供するという発展性を秘めている．そしてこの活動が，組織からの暗黙なコストと納期の要請に応えるための有効な手段になることを期待する．

参考文献

[1]尾崎信之，　石川隆，類似システムの並行開発に適したマネジメント手法の提案，
Journal of the Society of Project Management Vol.8，　No.1，　2006
[2]変更依頼の対応箇所を検討する前に他システムへの影響を検知する方法，
ソフトウェア品質管理研究会 2011 年度 分科会成果報告，日本科学技術連盟
[3]矢口竜太郎，開発ドキュメント　やってはいけない読み手を悩ます文書，
日経 SYSTEMS，日経 BP 社，227，p21-39，2012
[4]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，初版　第 4 刷発行，2012
[5]Paul，K.，Böckle，G. and van der Linden，F.，Software Product Line Engineering，
Springer-Velag (2005) （林　好一，吉村健太郎，今関剛訳，
ソフトウェアプロダクトラインエンジニアリング，エスアイビー・アクセス (2009) ）．
[6]飯泉紀子，設計者自身によるソフトウェア品質作り込みのアプローチ，品質，
日本品質管理学会，42 巻，4 号，p478-485，2012