ソフトウェアの派生開発におけるQCD改善への取り組み

Approach to improve QCD in the derivative development of software

清水信行 * Nobuyuki Shimizu
成井公一 * Hirokazu Narui

概要　先行開発段階におけるソフトウェアの派生開発への取り組みを紹介する。先行開発段階におけるソフトウェア開発は新製品開発のための要素技術開発や性能評価のため多くの機能追加や修正，試行錯誤による仕様変更に対応する必要がある。追加／修正作業など短時間での対応が求められる案件では既存の業務標準プロセスに対して「ドキュメント類の作成に膨大な時間がかかる」，「修正／追加要求に伴うプロセスが定義されていない」，「修正／追加に伴う仕様がソースコードのどこに対応しているのか把握しづらい」などの課題があった。これらの課題を解決するため派生開発に特化したプロセスXDDP（eXtreme Derivative Development Process）を参考に各プロジェクトで派生開発プロセスを定義し運用した。その結果生産性／品質面で改善がみられた。本論文では現在開発中の「鉛バッテリー状態検知センサ」，「車載用近距離レーダ」のソフトウェア開発に派生開発プロセスを適用した結果を紹介する。

1. はじめに

1.1 自動車業界におけるソフトウェア開発動向

車／環境／社会のニーズ多様化により自動車における電子制御化は著しく発展しており，効率的な開発と品質確保に加え機能安全を十分に考慮することが最重要課題となっている。車載用ソフトウェアの効率的な開発手法としてはAUTOSAR（Automotive Open System Architecture）と呼ばれるソフトウェアの共通化・再利用化を目的とした取り組みが活発に行われている。またソフトウェア品質確保のためのプロセス標準としてはAutomotive SPICE（Software Process Improvement and Capability dEtermination）が制定されている。これに加えて，近年最も話題に上がっている自動車向け機能安全規格も正式発行されたことを受け，自動車業界ではこれらの標準に準拠した開発への取り組みが積極的に行われている。これらの動向を踏まえて日本のOEMメーカーやサプライヤは製品開発におけるQCD（Quality：品質，Cost：費用，Delivery：納期）の向上とともに機能安全規格，Automotive SPICE，AUTOSARなどに準拠した開発体制／開発手法／開発プロセスが必要であり，当社においてもこれまで開発プロセスの標準化が進められてきた。

当社ではこれらの流れに沿った取り組みとして自動車用ソフトウェア開発プロセス標準「自動車用ファームウェア開発指針書」が発行された。本指針書では開発工程における成果物となるドキュメントや参照されるドキュメントを明確化した手順（フロー），作業ポイント，コーディング規約，設計ノウハウなどが記載されている。当社では本指針に則りソフトウェア開発を実施している。

1.2 QCD改善への取り組み

1.2.1 ソフトウェア開発の現状

とりわけ先行開発におけるソフトウェア開発は新製品開発のための要素技術開発／性能評価のため多くの機能修正や追加，試行錯誤による仕様変更に対応する必要がある。また仕向け先への試作品の提供や評価／試験のため，対応したソフトウェアを多数開発して提供する必要がある。これらのソフトウェアは機能修正／追加作業ということもあり短期かつ小人数で開発される場合が多い。そのため時間的制約から既存の業務標準のプロセスが省略されることもあり，開発途中においてはドキュメント作成作業が省略されることがあった。またいくつかの案件においては不具合の検出が遅れて下流工程から上流工程への手戻りが発生することがあった。

1.2.2 ソフトウェア開発における課題

短期での対応が求められる機能修正／追加案件には既存の業務標準プロセスに対して以下のような課題があると考えている。

(1) ドキュメントの作成に時間がかかる。
(2) 機能修正／追加に伴うプロセスが定義されていない。
(3) 機能修正／追加に伴う仕様がソースコードのどこに対応しているのか把握しづらい。
(4) (1)～(3)の課題を解決するために時間も工数もあまりかけることができない。

1.2.3 解決手段

これらの機能修正／追加案件におけるソフトウェア開発の課題を解決するため派生開発に特化したプロセスXDDPを参考

* 研究開発本部　自動車電装技術研究所

に各プロジェクトで派生開発プロセスを定義し運用した。派生開発プロセスを導入するメリットを以下に示す。

(1) 開発現場における頻発する要求に対応しやすいプロセスである。
(2) 変更要求仕様書を作成することで変更／修正箇所のBefore／Afterが把握しやすい。
(3) トレーサビリティマトリクス(TM：Traceability Matrix)を作成することで修正／追加仕様がソースコードのどこに対応しているのか把握できる。
(4) 作成した資料はレビュー時に仕様の間違いに気付くきっかけとなる。
(5) 変更／修正箇所のドキュメントを確実に残すことができる。

本論文では「鉛バッテリー状態検知センサのソフトウェア」，「車載用近距離レーダの評価ツール」に派生開発プロセスを適用した結果を紹介する。

2. 手段／方法

2.1 派生開発プロセス

派生開発プロセスとは新しく開発された機能の追加や新たな要求への対応に伴うソフトウェアの修正／追加に特化した開発プロセスである。本プロセスはソフトウェア業界では2007年頃から自動車業界においても2010年頃から脚光を浴びており，自動車部品サプライヤにおいても導入の事例がある[1)]。派生開発プロセスでは下記に示すドキュメントを作成／運用することが推奨されている[2)]。

(1) 変更要求仕様書(USDM：Universal Specification Describing Manner)
(2) TM
(3) 変更設計書

プロセスに関しては修正／追加の元となるソフトウェアの成果物(ソースコード，設計書など)の状況により任意に設定可能であるが，上記3点のドキュメントを作成した後にソースコードを修正することが大原則とされており，同様に作成した各ドキュメントは関係者へのレビューも必要とされている。これらのルールに従うことにより生産性の向上，ソフトウェア品質の向上が期待できる。以下に本取り組みにおける各ドキュメント，レビュー方法の一例を紹介する。

2.2 変更要求仕様書

図1に示すUSDMは，システムの変更／追加要求と要求に対応する仕様について関連する項目ごとに階層分けして記述する方法(書式)であり，ソフトウェア開発方法論のオブジェクト指向や構造化手法との親和性も良いとされている。また変更要求仕様書は「何を(What)」変更するのかを明確に記載し，変更要求に対する理由「なぜ(Why)」を記載することも一つの特徴としている。要求に対する理由を明確にすることは要求内容についての誤解を防ぐことができ，更には要求の変化の方向も予想することができるようになる[3)]。実際に本ドキュメントを作成したところ各階層での記述の詳細度の決定が困難であることがわかった。1つの対策例として図1における第一階層に「要求」，第二階層に「遷移状態レベルの仕様」，第三階求に「関数レベルの仕様」といった基準を設け案件ごとに記述を試みた。

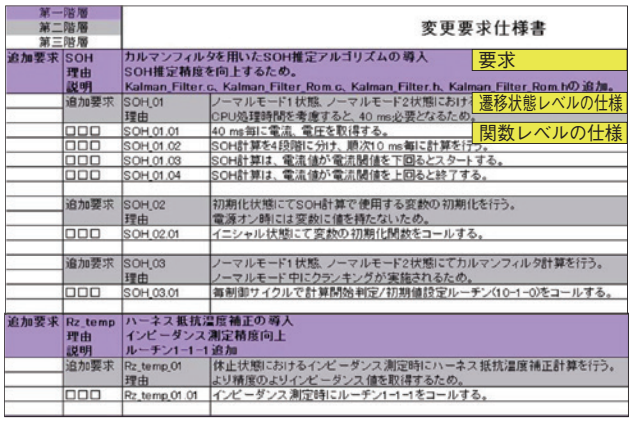

図1　変更要求仕様書
Change requirement specification.

2.3 トレーサビリティマトリクス

TMはUSDMと既存ソフトウェアのソース／ヘッダファイルとを関連付ける資料でありソフトウェア品質の「保守性(機能を改良する際の容易さ)」向上に繋がる。またソフトウェア修正を行う際「どこを(Where)」変更するかが本ドキュメントにより明確になる。自動車用ファームウェア開発指針書において定義されていたドキュメントだけでは機能変更やバグ修正時にコード修正部を特定することは困難であり，修正に伴い新たなバグが混入する可能性もあった。本ドキュメントはこれらの問題に効果的であり直接的にソフトウェア品質の向上に繋がると考えられる。今回の取り組みにおいて画面が存在するものについてはソースコードだけではなくGUI(Graphical User Interface)部も記述し，より第3者へわかりやすい資料となるようにした。

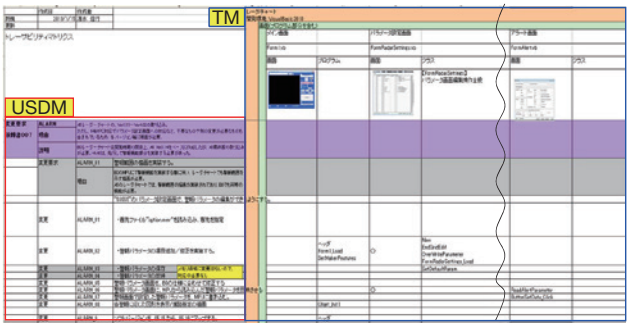

図2　トレーサビリティマトリクス
Traceability matrix.

2.4 変更設計書

変更設計書では関数などで具体的に変更する際の変更方法を文章で記述することを特徴とする。ここでは「誰が(Who)」，「いつ(When)」，「どのように(How)」変更するかを記述し予想される作業工数や実際の修正に要した工数なども可能な限り詳細に記載する(**図3**)。以下に紹介する事例においては本ドキュメントと従来から作成しているフローチャート，データフローダイアグラム(Data Flow Diagram)，状態遷移図，状態遷移表などを元にコーディング作業を実施した。

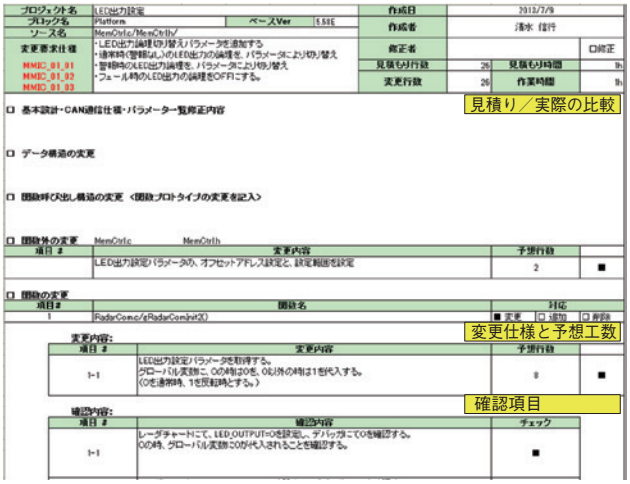

図3　変更設計書
Change design document.

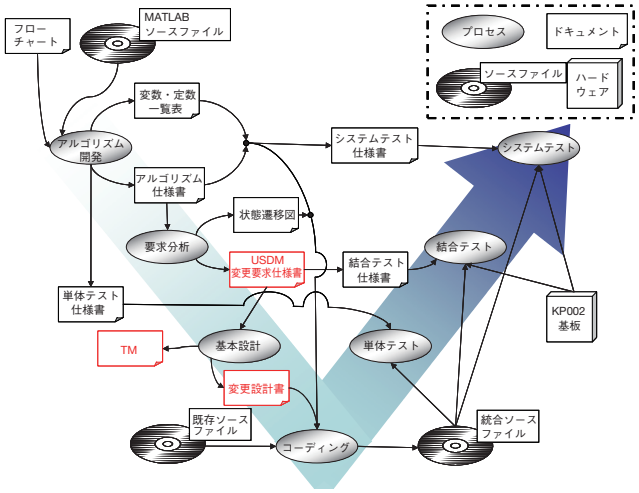

図4　BSS 派生開発プロセスフローダイアグラム
Process flow diagram of BSS derivative development.

2.5 レビュー

派生開発を行うにあたり従来通り成果物レビューやソフトウェアレビューが必要となる。レビューでは関係者がレビューアとして参加し，成果物に潜むバグを事前に取り除くことを目的としている。派生開発において必要となる観点は下記のとおりである。

①顧客から依頼された変更要求をどのように認識したか？
②ソース上の変更箇所をどのように認識したか？
③関数レベルでの変更方法をどのように考えたか？

特に②は重要でありスペックアウト資料と呼ばれる機能追加／修正前のソフトウェアの構造や設計の意図をドキュメント化する作業が必要となるケースもある。

3.　適用結果

3.1 状態検知センサへの適用

3.1.1 派生開発プロセスの導入

派生開発プロセスを状態検知センサ（BSS）の開発に適用した事例を以下に報告する。派生開発プロセスの導入は本取り組みが自動車研において初めてであったため①USDMの作成，②TMの作成，③基本設計所の作成を既存のV 字開発プロセスに組み込むという形で開発を行った。本プロジェクトでは**図4**に示すようなプロセスを定義して開発を行った。図4より新たにドキュメントの作成が必要な工程は「要求分析」，「基本設計」である。これらの工程の変更がソフトウェアの品質と生産性にどのように影響したかを以下に記載する。

3.1.2 派生開発プロセスの適用結果

今回の開発では研究段階のアルゴリズムを既存のソースコードに追加実装するにあたりベースとなるソースコード（約7 kstep）に対し約1 kstepの新規コード開発と，約1 kstepのコード移植を行った。BSSにおける派生開発結果（実装ステップ数，開発工数，検出バグ数，全テスト件数）とこれより算出したソフトウェア開発指標（生産性，バグ密度，テスト密度）を**表1**に示す。本プロジェクトではバグの定義を前工程への手戻り要因としている。

表1　BSS 派生開発における結果と開発指標
Development results and indicators in the BSS derivative development.

開発結果		ソフトウェア開発指標	
実装ステップ数［kstep］	2	生産性［人月／kstep］	2.5
開発工数［人日］	100	テスト密度［件／kstep］※	84
検出バグ数［件］	21	バグ密度［件／kstep］	10.5
全テスト件数［件］	84		

※新規開発コード1 kstepに対するテスト密度

(1) ソフトウェアの品質について

ソフトウェア品質に関する指標として「テスト密度」は単位ステップ数あたりのテスト件数で求められ，IPA（独立行政法人　情報処理推進機構）の基準では50 ～ 100件／ kstepが高い品質や信頼性が求められるシステムにおいて必要であるとされている[4]。今回の開発では新規開発コードに関して84件／kstep（移植部に関しては過去にテスト実施済）の基準内に収まる件数のテストを実施した。「バグ密度」は単位ステップ数あたりの検出バグ数であり，統計的には10 ～ 30件／ kstepがテスト工程前の段階で潜んでいるとBeizerによる調査（1990）の結果として報告されている[5]。「バグ密度」に関しても10.5件／kstepという結果となり，今回の開発において開発スピードの向上とソフトウェア品質の担保という改善効果が確認できた。

(2) ソフトウェアの生産性について

ソフトウェア開発指標の一つである「生産性」は単位ステップ数あたりの開発工数で表現され，過去の自動車研における開発（新規開発含む）では約4 人月／ kstepであった。今回の開発ではアルゴリズム開発からバグ対応までを含め2.5人月／kstepの生産性で開発を完了し，派生開発プロセスの導入による一定の効果が得られたと考えられる。

3.1.3 考察・まとめ

今回検出されたバグ総数は21件となりバグが混入した工程の内訳は「アルゴリズム開発」:11 件，「要求分析」:4 件，「コーディング」:6件であった。またバグを検出した工程は「アルゴリズム開発」:4 件，「コーディング」:1 件，「単体テスト」:5 件，「結合テスト」:3件，「システムテスト」:8件であった（**図5**）。

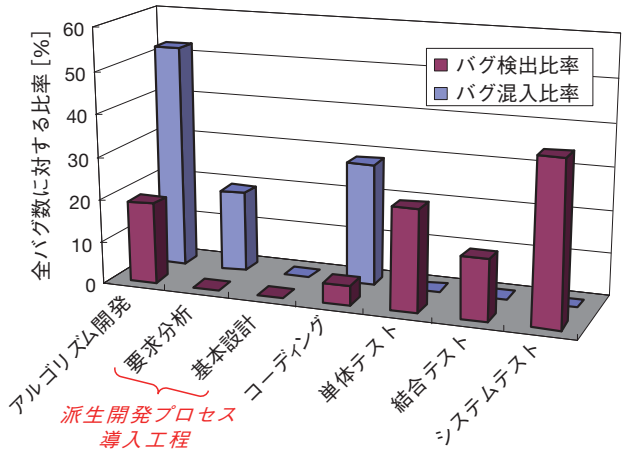

図5　各プロセスにおけるバグ検出比率とバグ混入比率
Bug detection rate and bug mixing rate in each process.

結果よりバグの半数以上は上流工程で混入し，バグの半数以上は下流工程で検出されたことが分かる。バグを検出した後のバグ修正／検証という手戻りを考えると上流の開発工程もしくは上流のテスト工程でバグが検出されることが望ましく，早期にバグを検出するためにレビュー方法の改善やテスト手法の改善，更には上流工程における検証方法の改善の必要性が示唆された。

一方派生開発プロセスの導入という観点では「要求分析」におけるUSDMの作成，「基本設計」におけるTMと変更設計書の作成により要求と仕様の紐付けが明示的に表現でき，ヌケ・モレといったバグ要因の排除に効果的であった。これに伴い「要求分析」と「基本設計」工程におけるバグの混入比率は全体の20%以下となりバグの排除に効果的であった。

3.2 車載用近距離レーダへの適用

3.2.1 派生開発プロセスの導入

派生開発プロセスを車載用近距離レーダに適用した事例を以下に報告する。車載用近距離レーダに関するソフトウェアはレーダ本体のマイコンに組み込まれるファームウェアとパソコンにてレーダ性能を評価するレーダ評価ツールがある。どちらのソフトウェアもレーダの機能追加により短納期かつソフトウェア品質も求められている。それらの要求に対応するため派生開発プロセスを適用した。派生開発プロセス導入により各種ドキュメント作成，レビューを実施することによりQCDを向上させることを目標とした。レーダにおける派生開発プロセスは**図6**のように定義した。

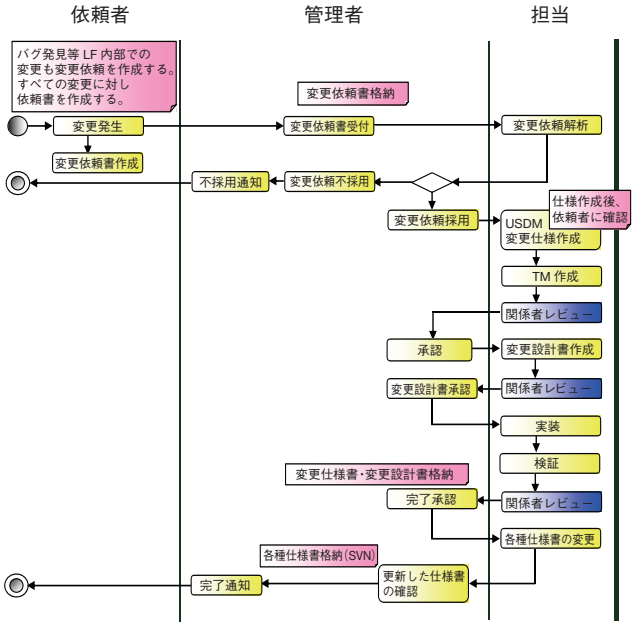

図6　レーダ派生開発プロセス
Radar derivative development process.

3.2.2 派生開発プロセスの適用結果

(1) ソフトウェアの品質について

表2に従来開発と3件の派生開発プロセスにおける各工程の工数割合，**図7**に各工程の工数割合を示す。表2と図7から派生開発プロセスにおける設計工程の工数割合が増加したことがわかる。すなわちコーディング／テストを実施する下流工程から設計を実施する上流工程に工数割合がシフトしたことがわかる。上流工程にて工数割合が増加した要因はドキュメントの作成とレビューを実施したためである。従って従来開発と比較し上流工程にて品質を作りこむ環境が整ったと言える。

一方で派生開発プロセスを適用したことにより見積りよりも工数がオーバーする結果になった。原因は各種ドキュメントを作成するにあたり不慣れだったためと要求仕様における詳細度の検討，ドキュメントのフォーマット改善に時間がかかってしまったためである。

表2　工数割合比較
Man-hour rate comparison.

テーマ名	変更ステップ数［step］	設計工数［h］				コーディング工数［h］				テスト工数［h］	
		見積り	実際	差	割合［%］	見積り	実際	差	割合［%］	実際	割合［%］
従来開発	225	–	5	–	12	–	21	–	51	15	37
派生プロセス1	143	6	10	4	53	3	3	0	16	6	32
派生プロセス2	2132	40	48	8	41	43	51	8	44	17	15
派生プロセス3	199	16	20	4	65	5	5	0	16	6	19

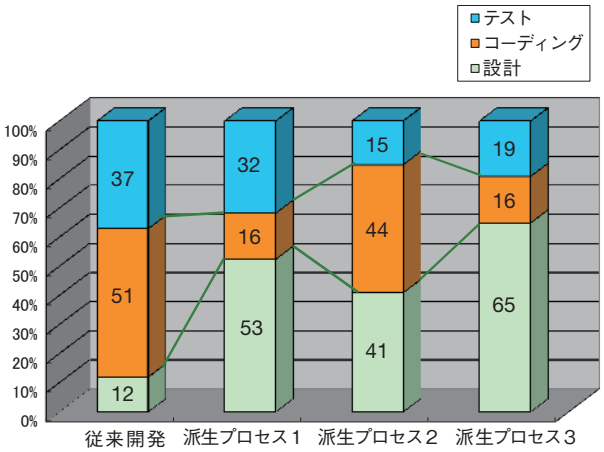

図7　工数割合の推移
Trends in man-hour rate.

また**表3**に従来開発と派生開発プロセス適用案件のバグ密度を示す。バグ密度は，結合／機能テスト時の障害件数と変更ステップ数（LOC）を除算することにより算出した。派生開発プロセスを適用した3件はいずれも結合／機能テスト時の障害密度が減少し品質面で向上したと言える。

表3　バグ密度の比較
Bug density comparison.

テーマ名	変更ステップ数［step］	結合／機能テスト	
		バグ件数	バグ密度
従来開発	225	3	13.3
派生プロセス1	143	1	**7.0**
派生プロセス2	2132	6	**2.8**
派生プロセス3	199	1	**5.0**

（2）ソフトウェアの生産性について

表4に従来開発と派生開発プロセスとのコーディングにおける生産性の比較を示す。設計工程における工数の割合は約4割増加したが一方でコーディング工程における生産性は約3倍に上昇した。したがって従来コーディング工程に割り当てられていた工数をほぼ設計工程にシフトできたことになりコーディング工程における手戻りが減少し効率的に作業が実施できたと言える。

表4　生産性の比較
Productivity comparison.

テーマ名	変更ステップ数［step］	設計工数［h］		コーディング工数［h］			テスト工数［h］
			割合［%］		割合［%］	生産性［LOC/h］	
従来開発	225	5	12	21	51	11	15
派生プロセス1	143	10	**53**	3	**16**	**48**	6
派生プロセス2	2132	48	**41**	51	**44**	**42**	17
派生プロセス3	199	20	**65**	5	**16**	**40**	6

3.2.3　考察・まとめ

レーダの派生開発に派生開発プロセスを導入した結果従来と比較して全体の工数に占める上流工程の割合が増加した。しかしプロジェクト全体の工数はほぼ変わらないため開発の重心が下流から上流工程へシフトでき，その結果結合／機能テストにおける障害密度が減少し品質向上する結果につながった。更にコーディング工程における手戻りが減少してコーディングの生産性が大幅に改善する結果になった。

派生開発プロセスを導入した当初は慣れないドキュメントの作成やレビューの実施に予定以上の工数が発生し導入の効果を実感できずにいた。しかし開発を重ねるにつれドキュメントのフォーマット，開発プロセスが改善されて生産性，品質の改善を実感できるようになった。またレーダ開発に適合した開発プロセスを定義できた意義は大きいと考えている。

4.　おわりに

派生開発プロセスの導入を試みた結果，生産性／品質面で一定の改善がみられた。一方以下のような課題も見えてきた。

①要求仕様書，TM，スペックアウト資料の，要求の詳細度をどの程度にすればレビューに効果的であるか。

②レビューを如何に短時間かつ効果的に実施できるか。

③修正／追加作業を同じソフトで繰り返すと差分の資料が増えていくため差分管理が大変となる。

今後はこれらの課題を解決するための検討を実施していく。さらに定期的にプロセス改善の検討，ドキュメント類のフォーマット見直しについても実施していく。

本取り組みを進めるにあたり派生開発推進協議会で紹介された事例なども参考にさせて頂いた。今後当社における派生開発の課題を解決するべく他のプロジェクトについても派生開発プロセスの導入を推進していく予定である。

参考文献

1）財団法人 日本科学技術連盟：第5回ソフトウェア品質会議論文集，（2012），27.
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），307.
3）清水吉男：要求を仕様化する技術・表現する技術，技術評論社，（2005），165.
4）独立行政法人　情報処理推進機構　ソフトウェア・エンジニアリング・センター：組込みソフトウェア開発向け品質作り込みガイド，翔泳社，（2008），101.
5）Cem Kaner，Jack Falk，Hung Quoc Nguyen：基礎から学ぶソフトウェアテスト，日経BP社（2001），23.
6）宮浦 直人：デジタル複写機のネットワーク開発へのXDDP適用事例，派生開発カンファレンス2013 発表資料，リコーITソリューションズ株式会社，（2013），5.