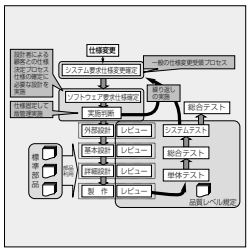


組み込みソフトウェアの高信頼性手法（SPIRIT）の開発

左近　透・松本達治・畑中健一
村瀬　亨

Software Productivity Initiative by Reliable and Innovative Technology (SPIRIT) —— by Toru Sakon, Tatsuji Matsumoto, Kenichi Hatanaka and Toru Murase —— Electronic control units (hereafter abbreviated as ECUs) are being increasingly used in vehicles and their control functions are becoming more complicated. In this trend, the number of troubles caused by failures in software applications running on ECUs is growing. Therefore, development and execution of a highly reliable technology for developing in-vehicle embedded software have become pressing needs among software developers.

The authors conducted an extensive investigation on current processes of software development, and based on the investigation result, developed a highly reliable method for software development that consists of two processes: (1) a function development process for accepting customer's specification changes and enhancing the degree of functional perfection and (2) a quality assurance process for realizing high reliability.

In development work, the two processes are switched to one another at a "milestone" date, which is called a quality assurance expiration date. Until the quality assurance expiration date, the function development process is executed to fit specification changes into the development schedule, realize requested specifications, and conduct an exhaustive test on software functions and structure. After the quality assurance expiration date is a quality assurance period for enhancing software quality. In case the specification changes occur during this period, the date of software completion (release date) is extended so that a sufficient process period can be secured. In the quality assurance period, various tests are conducted to ensure system quality.

Although iterative development is a conventionally proposed development method for embedded software, no specific method for implementing software quality assurance actions has been proposed. The authors think that software quality can be ensured at a high level by setting a quality assurance expiration date and shifting the purpose of development from and to function development and quality assurance before and after that date. This method will be validated in the actual development environments to be established as an effective software development method.

1. 緒　言

現在、エンジン制御を始めとして車のさまざまな機能が電子制御で実現されている。安全性、車内快適性の向上や環境負荷の低減などへの期待の高まりを背景に、電子制御ユニット（Electronic Control Unit. 以下ECU）が果たす役割が増大しており、一台の車に数十個のECUが搭載されている車種も存在する。

従来、これらのECUは、制御対象、例えばエンジン制御やドア制御といった制御対象毎に制御を行うのが通常であったが、最近では車載ネットワークを介して複数のECUが分散ネットワーク・コンピュータシステムを形成し、連携・協調して仕事を行うようになっている。仕事によっては、数十台のECUが相互にデータ通信を行って、車全体で一台のコンピュータのように振舞い、車の状況に応じた高度な制御動作を行っている。

ECU上で稼動し、これらの制御機能を実現する制御ソフトウェアは、装置（ECU）の目的とする仕事（ミッション）に特化して調整されたソフトウェアであることから、PC上で使われる業務用ソフトウェアと区別して組み込みソフトウェアと呼ばれる。最近、車載ECUの組み込みソフトウェアによる不具合が露見することが目立ってきた。それは、単に車一台あたりのECU台数が増えたということだけでなく、上記のような高度な利用が増えて、ソフトウェア機能・構成が複雑化してきていることが要因となっている。

自動車業界においても、ECU組み込みソフトウェアの信頼性向上は課題として捉えており、AUTOSAR（Automotive Open System Architecture）やJASPAR（Japan Automotive Software Platform Architecture）などのコンソーシアム活動での目標のひとつに、組み込みソフトウェアの信頼性の確保を取り上げて取り組んでいる。また、国際的競争力をつけた我が国の基幹産業の死角として、経済産業省が主導して、組み込みソフトウェアの信頼性確保に取り組んでおり、これに呼応する形で、大学等の研究機関や企業等からは組み込みソフ

トウェアが採り上げられるようになっている。

米国では、組み込みソフトウェアは、宇宙航空・軍事技術を支える技術として長い歴史を持っており、ソフトウェア工学の中心的技術として、ソフトウェア開発プロセス（開発モデル）や信頼性テストに関して多くの技術が発表され、また実証結果が報告されている。

自動車産業は、モジュール化技術の典型であるコンピュータ・エレクトロニクス産業と比較して、統合型（インテグラル）産業と分類されている。開発においては、常に部品相互のすり合せが繰り返され、最適な仕様に向けて部品メーカも協力し合って高度な機能を発揮するまで磨き上げていく手法を採用している。その中にあって、車載ECU向けの組み込みソフトウェアは、すり合せされる部品相互のインターフェイス的役割も担っており、それゆえにモジュール化が困難であった。ECU組み込みソフトウェアは、最終アセンブリを行う自動車メーカに持ち込まれてからもすり合せの対象となるため、組み込みソフトウェアの信頼性確保のためには、その開発プロセスにまで踏み込んだ手法が必要となる。

我々は、ECUで稼動する組み込みソフトウェアの開発プロセスの最上流からテスト、リリースまでのプロセスと開発チームの役割分担を明確にし、従来採用されているウォーターフォールモデルに代わる反復法を適用した組み込みソフトウェアの開発モデルを確立し、高信頼性化手法を開発した。

本論文では、その高信頼化技術、具体的には、従来手法の問題点を分析しながら、新しく提唱する、組み込みソフトウェア高信頼性開発手法について述べる。以下、第2章では、従来組み込みソフトウェア手法の現状を分析し、第3章では、本論文の新開発手法の骨子について述べ、第4章では、そのエンジニアリング的側面について考察し、第5章で、他の研究との比較を試みた上で、第6章においてまとめを行う。

2. 組み込みソフトウェア開発方法論の現状

車載組み込みソフトウェアは、第1章で述べたように、PCなどで用いられる業務用ソフトウェアと比べ、次のような特徴を持っている。

①ソフトウェアとして果たすべき仕事（ミッション）は、あらかじめ、限定的に決められており、従って想定されていない機能に対する拡張性はない。また、性能追求等の最適化調整も、開発時に他の車載部品とのすり合せによって決められており、メモリ増設など処理能力的な拡張性も想定していない。

②ECUと一体となって制御に用いられるため、実時間制御（リアルタイム処理）や、定められた時間制約を守って処理結果を出すことが求められる期待目的型処理（ミッションクリティカル処理）である。

③ECUと共に大量に同時に出荷され、出荷された後は余程のことがない限り、プログラムの修正や改良はできない。

従って、車載組み込みソフトウェアの信頼性は、ECU開発メーカ内で組み込みソフトウェアを開発するプロセスの中で確立されている必要がある。しかし、一方では自動車メーカと部品メーカとが、車種開発の最終段階まで調整を行う、自動車産業固有のすり合せ作業の中で組み込みソフトウェアの機能が確定していく側面も持っている。このため、従来から、提唱されているソフトウェアのウォーターフォール（V字型）開発モデルである、定義、実装、確認のサイクルが適用できることはまずない。特に、すり合せ作業を重視する余り、信頼性確保の手法が十分適用できないまま、実現機能の確認に傾斜してきた面が否定できない。

近年、組み込みソフトウェアの信頼性確保のための手法開発は、非常に活発になってきており、EASE（Empirical Approach to Software Engineering）プロジェクトやIPA（Information technology Promotion Agency）のソフトウェア・エンジニアリング・センター、大学などの研究機関からいくつかの提案が見られる。その一つが反復型開発モデルである。これは、小規模なリリースを行い、段階を追って機能実装を進めるアプローチである。プロセス管理手法や機能実装の進め方の違いなどによりイテラティブ、インクリメンタル、Rational Unified Process（RUP）®などの手法がある。しかし、すり合せ型業界における組み込みソフトウェア開発への具体的な適用方法については、いまだ決定的な適用方法は見出されていない。

ここで、従来の組み込みソフトウェアの開発プロセスの分析を行なう。**図1**に、典型的なECU組み込みソフトウェアの開発の流れを示す。試作完了のたびに顧客である自動車メーカに、評価と確認を求める作業が入っている。これがすり合せである。その結果を得て、改良のための試作が行われる。また、ECU組み込みソフトウェアの典型的な開発モデルを**表1**に示す。

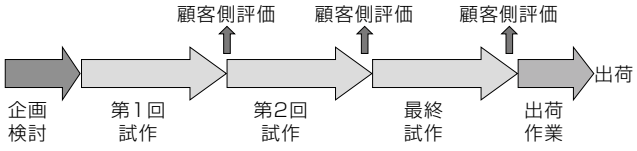


図1　ECUの開発の進行パターン

表1　開発体制の概要

開発規模（バイナリサイズ）	256KB以下（主に120KB程度）
開発期間	半年から1年（新規開発で2年）
試作回数	2～3回（ただし顧客都合により随時対応）
試作期間	1から2ヶ月
開発チームサイズ	5人程度（規模が大きいもので10人程度）
部門内での同時並行開発数	20程度

ECU組み込みソフトウェア開発で最も普及しているのが、ウォータフォール型モデルであり、**図2**に示す。このモデルでは、ソフトウェアの品質保証は、各仕様書とソースコードに対するレビューおよび、外部仕様、基本仕様、詳細仕様の内容に対応した機能確認するテストにより実施される。こういった開発と確認のサイクルはドキュメント体系としても確立され、それらが確実に実施されていることを開発の仕組みとして検証しているのがCMMであり、あるレベルをクリアしていれば相当の信頼性が確保できるはずである。

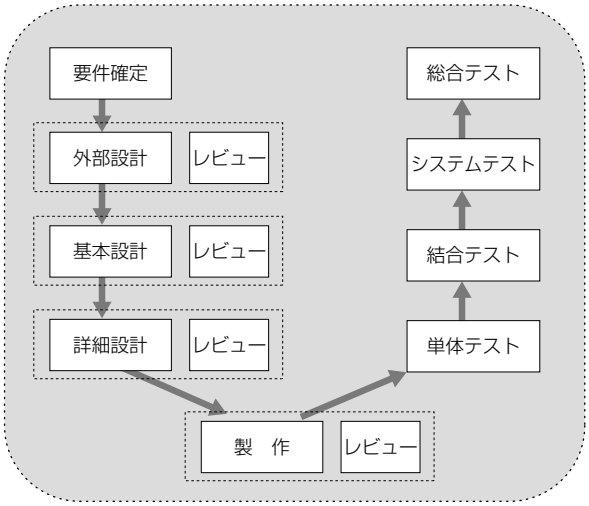


図2　ECU開発での標準の開発プロセス

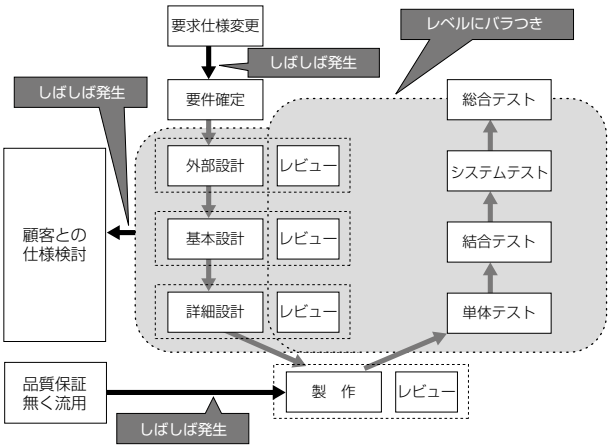


図3　開発成果物分析による実態調査結果

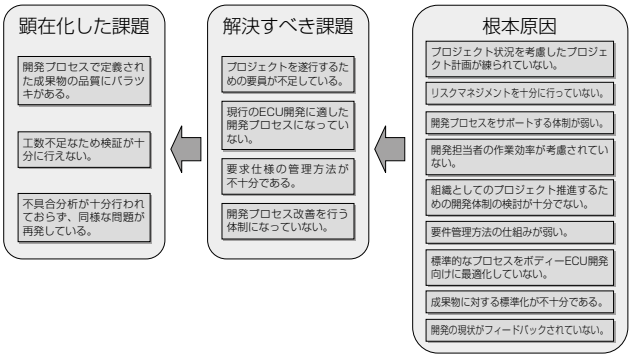


図4　開発要員ヒヤリングによる実態調査結果

しかし、これまで見てきたとおり、すり合せ型産業でのECU開発ではソフトウェアに起因する不具合の発生の根絶には到っていない。その原因として、以下の3点が課題となっている。

①要求仕様変更の問題

ウォータフォール型モデルは、前提として、開発工程の早い段階で仕様を仔細に固め、しっかりテストし、工程遅れや品質を損なう要因は、前倒し出しつくす、という考えに基づいている。それも各部門をモジュールとしてとらえ、モジュール毎に信頼性を確保していけば、システム全体の信頼性も高まる、との前提条件を置いている。これは、すり合せ型組み込みソフトウェア開発にはそのまま適用することは困難である。理由は、すり合せ型開発では、各モジュールなどの“部分”の仕様は、“全体”の仕様固めと連動し、開発工程後半に確定し、一旦確定した仕様も見直しが入るためである。従って、一般のウォータフォールモデル型で重視する、各ステップを固めて確認というサイクルが常に乱されているともいえ、これに対応した開発モデルが必要となる。

②品質保証工程の問題

①のすり合せ型開発では、実現すべきシステムの機能・性能は格段に向上するが、開発ステップに、明確な品質保証工程が必要となる。開発における品質保証の考え方は、システム全体の品質保証は、開発者による地道な仕様書・ドキュメントの作成により、ドキュメントレビューによって行われる。また、最終的にはシステム完成時に、開発し終えたシステムを直接様々な品質基準に基づいた品質保証テストすることによって確認する。

ところが、①で述べたすり合わせ活動によって、最終的にシステムの仕様が変更を受けてしまい、部分間で不整合を起こしたまま、部分毎の確認で問題なし、となる要素が入り込む。この不具合を摘出しシステム全体の品質保証を行うためには、やはり最終的にシステム全体の品質保証テストを実施することが必要である。しかし、多くの場合、他の部品と同様に、ECU組み込みソフトウェアの品質は、ECU開発メーカ側で対応することが前提となっている。

また、車メーカ、部品メーカ共に、すり合わせを追求するあまり、システム仕様決定期限を越え、品質保証期限限界を圧迫しかねない状況に陥りがちである。従って、すり合わせを採り入れた反復法の中に、品質保証工程を明示的に導入することが重要である。

③ソフトウェア部品の課題

ソフトウェアの部品を用意しておき、信頼性の高い部品を再利用することで、ソフトウェアの信頼性を確保するという考え方は、大方のところは異論がない。

実際の開発現場では汎用プラットフォーム化を推進し、ハードウェアや通信部分、そのほか汎用的に使われる低レベル機能を部品化し、それらを利用して開発を実施し、ある程度の効果をあげている。しかし、一般には部品の粒度が大きくなるほど、部品化の条件が煩雑となり、部品化をためらうことにつながる。また、部品化を宣言して反復利用に供しても、利用条件が複雑で難しい。特に、すり合せ型開発における組み込みソフトウェアでは、実行時の動的な動作条件は、ドキュメントでは表現が困難である。その結果、部品化は諦め、個々のモジュール開発者が“似たようなコード群”を流用することが行われる。コードの流用は、その品質保証ができていないため、問題が発生しやすい。

3. ECU向け組み込みソフト開発手法

第2章で提示した問題点を解決するために、従来の手法を分析し、以下のソフトウェア開発プロセスを提案する。

①工程を2つに分け、前半を仕様の作りこみの期間、後半を品質検証の期間とする。品質検証の期間は、製品リリース時期から、品質保証に対して必要な工数を逆算して設定する。またこれに関して、品質検証期間を開始するに先立ち、最終的な仕様変更の受付可能日を設定する。これらの設定が存在することは顧客との契約作業時に設定する。具体的な設定は製品開発内容によって規定する。仕様変更受付期限を越えて仕様変更が発生した場合には、対応を顧客と協議する。

②仕様の作りこみの期間では、反復型開発により仕様変更に対応した開発を行う。ただし、仕様変更に際して、ECUとしての仕様確定後、さらにソフトウェアの仕様を確定するために顧客との交渉プロセスを設ける。

③仕様の作りこみの期間では、仕様変更に直接関連した部分のテストを行う。テストの内容は、仕様変更対象部分に限定した機能テスト、境界値テストおよび仕様変更対象部分に関連した機能のリグレッションテストを行う。

④仕様の作りこみの期間では、設計書ならびにソースコードに関するレビューを実施する。レビューは設計に関係する少数参加者で実施する。また、構文解析ツールやスケジュール管理システムを用いた効率の良い実施を目指す。

⑤品質保証期間では、各機能に対してはレグレッションテストとして、作りこみの過程で実施したテストを、また品質保証のために、システムの状態および入力に関する複合条件テストを実施する。

4. 車載組み込みソフトウェア開発への反復型開発手法の実施モデル

3章で述べた実施モデルの詳細を検証する。

4－1 組み込みソフトウェア開発での信頼性の確保 組み込みソフトウェアの信頼性は、信頼性設計と信頼性テストを工程の中に組み入れることによって実現される。

しかし、既に述べたように、組み込みソフトウェアの開発工程においては、組み込みソフトウェアの部分的な仕様変更が相次ぐため、通常のウォータフォール型開発モデルであれば、テスト工程が織り込まれているのに対して、信頼性テストの実施の機会を逸してしまうことが多い。

そこで、筆者らは、反復型モデルと呼ばれる開発モデルをもとに、さらに信頼性テストを確実に実施することを図った、高信頼性反復型モデルを開発した。基本的な考え方は、組み込みシステム固有の仕様変更を信頼性保証期限までは受け入れ、反復法と呼ばれる手法で実現機能確認テストをその都度実施し、信頼性保証期限後は仕様変更を原則受け付けず、信頼性テストを実施する、という考え方である。

高信頼性反復型開発モデルでは、開発プロセスに以下の要素を加えている。

(1) 信頼性期限（デッドライン）の設定
(2) 信頼性期限までの反復法の完全実施
(3) 信頼性期限後の信頼性テストの完全実施

信頼性期限の設定とは、組み込みソフトウェア・システム仕様変更の受付締め切り期限を設けるということである。この期限までは、仕様変更を受け入れてもよいが、この期限を過ぎた場合の仕様変更は、受け付けないか、あるいは、最終納期を変更してもらう必要がある。最終の納期の変更理由は、もちろんシステムとしての信頼性確保のためである。信頼性期限は、たとえ仕様変更がなかったとしても、原則として早めることはできない。仮に早めて信頼性テストに入っていた場合、仕様変更は受け入れなければならないので、途中まで実施した信頼性テスト作業は、棄却しなければならない。また、信頼性期限後に、そのシステムの信頼性テストは、完全に実施しきらねばならない。そのため、この期限の設定は開発チームの開発実績に基づき、反復法で度々の仕様変更を機能・性能上反映しきる開発力と、その後出荷までの信頼性テストを実施しきる開発力があってのことになる。また、仕様変更は、社外顧客から発せられることが多いため、顧客との信頼がなければ、期限を区切ることは実質困難となる。これらの基本的考え方を元に、高信頼性反復型開発モデルの考え方を述べる。

4－2 仕様変更定義プロセスの定義と開発作業プロセスへの組み込み 分析で述べたように、ソフトウェアの仕様を固定するには、ソフトウェアの実装の立場から、顧客との仕様検討実施が必要である。この作業をプロセス上で明確に定義するため、開発側のPDCサイクルに仕様変更確定作業を定義する。仕様変更の確定作業は開発のPDCサイクルのP（Plan）の考えに従って実施する（**図5**）。Planは仕様確定作業→実施判断→実施計画の順で進む。

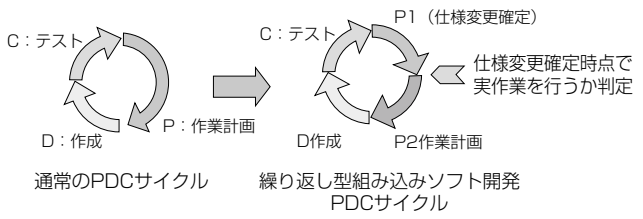


図５　仕様決定の開発プロセスへの組み込み

仕様変更の確定プロセスは顧客との対話を中心としたプロセスとなる。すなわち、顧客からの仕様変更を受け取った段階で、その内容を確認する。確認後、開発部隊で検討を進め、関連する仕様などについて分析し、再度顧客に確認を求める。このプロセスを繰り返し実施することで、仕様に関する誤解を可能な限り防ぐと同時に、関連する仕様変更も同時に決定し類似の仕様変更が現れるのを防ぐ（**図６**）。

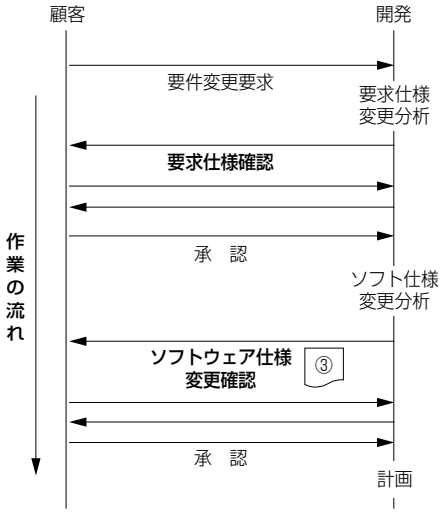


図６　仕様変更確定における顧客との対話プロセス

実際のソフトウェア作成作業の開始判断は、イベントや予め決められたスケジュールなどの外部的な要因と開発人員の投入可能時期など内部的な要因を検討して判断する。

これに合わせて、成果物の構成管理も仕様変更を中心とした管理とする。すなわち、固定された仕様毎にバージョンを定義する一方、不具合修正等による更新をリビジョンと定義する。品質の確認はバージョン単位で実施する。このような管理は、商用またはオープンソースの構成管理ツールで実施可能である。

４－３　部品化の推進　一般にソフトウェアの部品化設計は難易度が高く、設計時に再利用を考慮することは設計コストが高くなる。特に粒度の大きい高機能の部品は汎用的な部品設計はきわめて困難である。従って、部品開発プロセスをソフトウェア開発プロセス中で位置づけるには、既に存在する開発成果物の中から部品を開発するプロセスを定義すること、および部品の構成を実際の開発の進め方にあわせたものにすることが必要である。

そこで、仕様変更に対する開発が完了するたびにソースコードと関連する仕様書の改変内容を分析、その結果をソフトウェア部品に反映し部品の品質保証を実施するプロセスを定義する。なお、このプロセスは製品開発に対して横断的におこなうために、部品化担当チームを別途定義する（**図７**）。初めて部品化を行う場合には、過去開発成果物を分析した上で実施する。この作業は、ハードウェアやOS（Operating System）のインターフェイスに代表される低レベル汎用機能部品と、製品としての機能単位で定義される高機能部品とに分けて進める。

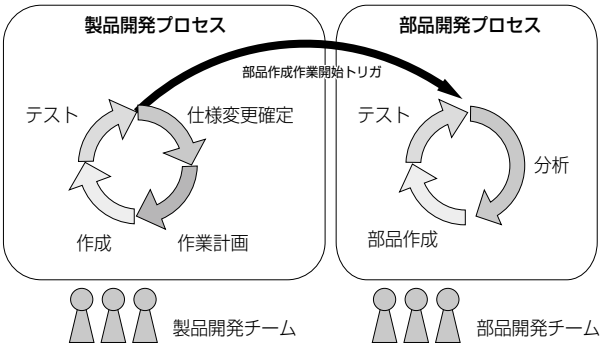


図７　部品化プロセス

低レベル汎用機能部品は、機能仕様、インターフェイス定義およびソフトウェア部品本体と品質保証テスト結果から構成される。また、高機能部品では、機能仕様、インターフェイス定義、仕様書のテンプレート、過去の仕様実装のパタン例を含む機能開発手順書、ソフトウェア部品本体および品質保証テスト結果から構成される。いずれの場

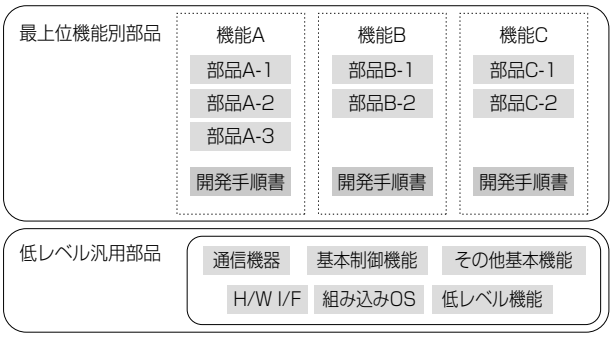


図８　部品構成

合でも、品質保証テストは、部品化担当チームで実施、品質保証を行う。 製品開発の際には、製品機能ごとに定義されたソフトウェア部品群から部品を選択して開発を進めてゆく。(**図8**)

ただし、部品の制作に際して、ソースコードの中で共通部分、汎用部分を切り出すことは人手のみで行うことは限界がある。そこで、部品化の手法として、コードクローン検出技術の利用を実施する。コードクローンとは、ソースコード流用によりプログラム中または複数のソフトウェア間で存在する複数の類似コードのことである。ECU開発では、問題はあるものの流用が有効に使われているため、本方法による部品の切り出しが有効と考えられる。また、同時に製品ごとに変更を受けた部分も切り出せることから、部品ごとのインターフェイス作成も容易であろうと考える。(**図9**)

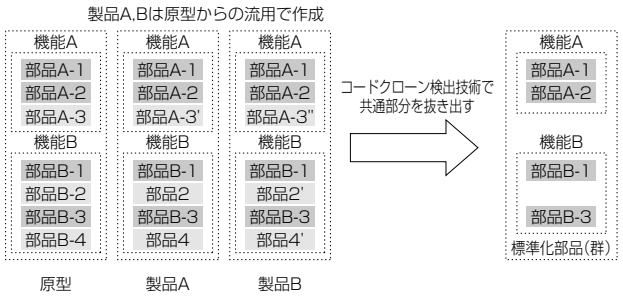


図9　コードクローン検索による部品化

このような部品校正の開発の進め方は、はじめに製品機能の選択時に高機能部品を選択し、その部品群に付随する製品開発手順に従って作業を実施するものとなる。新規に開発される機能の追加は主に低レベル汎用部品を利用して開発される。一方、部品化チームは開発成果物の分析を並行して実施し部品品質の強化に努める。

4－4　品質保証（テスト）の実効性向上　反復において繰り返されるテストを効率化し、実施負荷の低減を積極的に図ることは、以下の2つの手段で実施する。まず、テストの実施内容をレベル付けし、開発時期に応じて計画的な品質積み上げが可能なように規定することにより、各開発段階で適切な品質保証を実施し、投入工数の適正化を図る。つまり、反復での品質の積み上げ時、最終ソフト出荷時、および製品出荷時におこなうテストの内容を明確に区別する。たとえば、開発の反復段階ではシステムの網羅的なテストは費用対効果が悪く、一回の反復にかかるコストも大きい。しかし、最終ソフト出荷段階では、事実上、不具合修正の最後の機会であるため、品質の信頼性を確認する網羅的なテストは必須である。製品出荷時には、すでに重大な不具合は取り除かれているものとして機能動作中心のテストを実施する。テストの内容については、近年、業界ごとに品質の基準となるレベル分けの規定が作られており、車関連においてはMISRA（Motor Industry Software Reliability Association）開発ガイドラインによるテストレベル（**表2**）が存在する。したがって、その基準に沿った品質保証計画およびテスト内容規定を作成することが有効と考える。

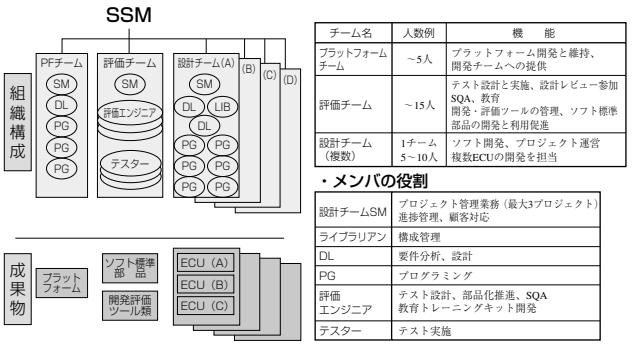
表2　MISRA Integrity Level

Development Process	Integrity Level				
	0	1	2	3	4
Testing	—	Show fitness for purpose. Test all safety requirements. Repeatable test plan.	Black box testing.	White box module testing --- delined coverage. Stress testing against deadlock. Syntactic stalic analysis.	100% white box module testing. 100% requirements testing. Semantic static analysis.
Verification and validation	—	Show tests: are suitable; have been performed; are acceptable; exercise safety features. Traceable correction	Structured program review. Show no new faults after corrections.	Automated static analysis. Proof (argument) of safety properties. Analysis for lack of deadlock. Justify test coverage. Show tests have been suitable.	All tools to be formally validated (when available). Proof (argument) of code against specification. Proof (argument) for lack of deadlock. Show object code reflects source code.

さらに、テスト自動化の導入により実施負荷の低減を図る。特に最終ソフト出荷に伴う網羅テストは、テスト項目が膨大となるため、繰り返し段階で実施したテストの入力パラメータと出力結果を元に自動化ツールで実施するという形を推進する。

4－5　品質保証（レビュー）の実効性向上　反復開発で効率よく反復を行うためにはレビューの効率向上を実現することも必要である。そのためにはテストと同様、レビューにおいても人手をかけてレビューしなければならない部分に限定して人員の時間投入を行う一方、レビューそのものの効率的な実施方法を導入する。具体的には、①コーディング規約等の表現や大域変数の初期化などの確認はツールを利用し、その一方、②レビューにおいて、参加者の知識や立場などが重大な影響を及ぼすことを考慮に入れ、レビュー参加者を下流工程担当者やテスト担当者など関連工程の担当者などと規定してレビューの有効性を高める方策を実施する。このようなレビューワの能力および開発における立場に着目して参加者を選択すること、および少人数でのレビューが有効であることは（Parnas 1985）などで指摘されており、有効性が高いと考えられる。

また、実際の運用に際してはレビューのスケジューリングが困難になることが容易に想定されるため、レビュー実施優先時間帯を作成し、実施の便を図るなどの工夫が有効であると考える。



チーム名	人数例	機　能
プラットフォームチーム	～5人	プラットフォーム開発と維持、開発チームへの提供
評価チーム	～15人	テスト設計と実施、設計レビュー参加 SQA、教育 開発・評価ツールの管理、ソフト標準部品の開発と利用促進
設計チーム（複数）	1チーム 5～10人	ソフト開発、プロジェクト運営 複数ECUの開発を担当

・メンバの役割

設計チームSM	プロジェクト管理業務（最大3プロジェクト） 進捗管理、顧客対応
ライブラリアン	構成管理
DL	要件分析、設計
PG	プログラミング
評価エンジニア	テスト設計、部品化推進、SQA 教育トレーニングキット開発
テスター	テスト実施

図10　ECUにおける反復型開発体制例

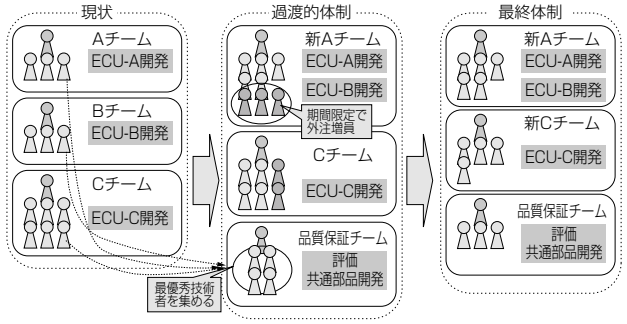


図11　体制の移行

5.　関連研究

組み込みソフトウェア開発手法に関する研究は数多く存在する。しかしほとんどの研究は要求定義や部品化など開発プロセス内部での個別の技法に関するものが多い。また、アジャイル開発モデル（eXtream Programming やLeanソフトウェア開発など）に代表される開発モデルに関する研究でも、組み込みソフトウェア開発の適用方法についてはいまだ方法論が確立していないのが現状である。特に、①顧客が動作を確認することが難しい組み込みシステムでどのようにして顧客との対話を進めるか、②どの段階で、どのようなテスト、品質や仕様の実現内容の確認を実施すべきかについて、開発プロセスの進行と関連づけて議論されていないうえに、組み込みシステムで品質保証を最終的にいかに確保するかについては、明確な回答が与えられていない。

本論文では、開発工程全体を2段階に分けるソフトウェアサイクルとして定義する。前期は製品の機能を作りこむための機能開発期間である。後期は、品質保証を確実に行うための検証期間である。そして、それぞれの時期で必要なテスト内容を目的と合致させることを一つの特徴としている。また、顧客との対話をソフトウェア開発プロセスで規定している点が他に提案されているモデルとの違いである。

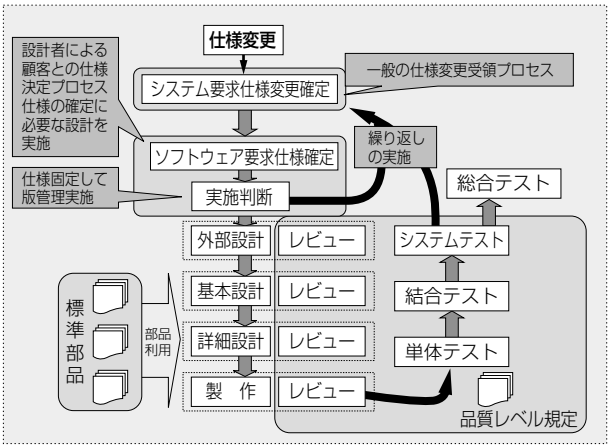


図12　車載組み込みソフト向け開発プロセス

なお、プロセス内部の個別の最適化については制限を設けていない。したがって、レビュー効率化や部品化のためとりえる方策が別途あれば、それを取りこむことになんら障害となるものではない。

6.　結　　言

車載組み込みシステムは高機能化と開発期間の短期化が進行する一方、きわめて高い信頼性が要求されている。本論文では、高信頼性車載組み込みソフトウェア開発を実現する手法を提示した。この手法の要点は、以下の4点である。

①開発工程を前期と後期に分ける。前期は製品の機能を作りこむための機能開発期間である。後期は、品質保証を確実に行うための検証期間である。

②開発工程後期で品質保証を行うために必要な日数から、仕様変更の最終受付期日を顧客との合意の上で決定する。それ以降に仕様変更が発生した場合には顧客と対策を検討した上、工程の変更等の対策を実施する。また、十分な品質を保証するための検証方式および確証の内容についても合意の上決定する。

③開発工程前期では、組み込みシステムに特有の仕様変更の頻発と動作確認の困難さに対処するため、反復法による開発プロセスを実施する。このとき、1回の反復の開発を効率よく進めるために、ソフトウェアの部品化およびレビュー強化、テスト強化を実施する。

④反復の時点で行われるテストは、仕様変更対象となった部分が仕様どおりに実装され、かつ動作するかを確認する（Verificationテスト）。品質保証の段階で行われるテストは、レグレッションテストとしての各機能テスト、複合条件テストを実施する。

今回の報告は手法の提案に留まっているが、今後、本開発手法を現場に適用し、より品質の高い組み込みソフトウェア開発に貢献したいと考えている。

参 考 文 献

(1) 石井康夫他編「日科技連ソフトウェア品質管理シリーズ」東京、日科技連、6冊（1986）

(2) 前川徹「ソフトウェア最前線―日本の情報サービス産業界に革新をもたらす7つの真実」東京、アスペクト、262p.（ISBN 4-75721-064-7）（2004）

(3) Glass,R.L.「ソフトウェア開発55の真実と10のウソ」東京、日経BP社、328p.（ISBN 4-82228-190-6）（2004）

(4) Heldman,K.「早わかりプロジェクトマネジメント」東京、コンピュータ・エージ社、366p.（ISBN 4-87566-292-0）（2004）

(5) Wiegers,K.E.「ピアレビュー高品質ソフトウェア開発のために」東京、日経BPソフトプレス、272p.（ISBN 4-89100-388-X）（2004）

(6) Kaner,C.Nguyen,H.C,Falk,J.「基本から学ぶソフトウェアテストーテストの「プロ」を目指す人のために」東京、日経BP社、471p.（ISBN 4-82228-113-2）（2001）

(7) Craig,R.Jaskiel,S.P.「体系的ソフトウェアテスト入門」東京、日経BP出版センター、297p.（ISBN 4-82228-207-4）（2004）

(8) Kruchten,P.「ラショナル統一プロセス入門.東京,ピアソン・エデュケーション」293p.（ISBN 4-89471-342-X）（2001）

(9) 経済産業省「2004年版組み込みソフトウェア産業実態調査報告書」

(10) 神谷年洋「コードクローンとは、コードクローンが引き起こす問題、その対策の現状」電子情報通信学会誌 Vol. 87, No. 9, pp. 791-797（2004.9）

(11) Brooks, Frederick P., Jr. 1987 "No Silver Bullet-Essence and Accidents of Software Engineering." IEEE, Computer,

(12) Grady, Robert. Successful Software Process Improve-ment, Prentice Hall, 350p.（ISBN 0-1 3626-623-1）（1997）

(13) Parnas, D. L, Weiss, D. M. Active Design Reviews：Principles and Practices, Proc. Eighth International Conference on Software Engineering, August（1985）

執 筆 者

左近　透：住友電工情報システム㈱　参事

松本　達治：㈱オートネットワーク技術研究所　主席

畑中　健一：自動車技術研究所　制御・通信ネットワーク研究部　プロジェクトリーダー

村瀬　亨：自動車技術研究所　所長（工学博士）