組込み製品の品質を高めるための再利用ソフトウェア資産抽出方法

The method of extracting the reusable software assets for improving the quality in embedded software

主　査　飯泉 紀子（株)日立ハイテクノロジーズ
副主査　吉澤 智美 NEC エレクトロニクス(株)
　　田所 孝文（株)山武
研究員　伊藤 雅子 富士通(株)
　　酒井 由夫 日本光電工業(株)
　　佐藤 敏徳 オムロンアミューズメント(株)
　　佐藤 雅思 オムロンアミューズメント(株)
　　松本 拓也 三菱コントロールソフトウェア(株)

概要

組込みソフトウェア開発では既存製品のソフトウェア資産を再利用することが多く、高品質の製品を効率よく開発するためには再利用ソフトウェア資産の品質をいかにして向上させるかが課題である。

一般的に、製品に使用するソフトウェア資産のすべてを完璧に仕上げることは困難である。そこで、既存のソフトウェア資産から製品として価値があり、寿命の長い部分を抽出し、その部分の品質を高めることで、効率的に全体の品質を向上させることができると考えた。

本論文では、その抽出方法を提案し、3 つの組込みソフトウェア製品に適用した結果を報告する。

Abstract

In embedded software, the software assets of an existing product are frequently reused. And it is a great challenge to improve the quality of the reusable software assets.

Generally, it is difficult to improve all of the reusable software assets. The software assets have a valuable and long-lasting part. If that part can be extracted and that quality is improved, the whole quality is efficiently improved.

This paper proposes the method of extracting the assets, and describes the result of applying this method to 3 embedded software products.

1. 本研究の選定理由と解決すべき課題

組込みソフトウェアに要求される機能は多機能化している。このため、組込みソフトウェアは従来に比べ複雑になり、規模も増大している。しかし、組込みソフトウェアは常に高い品質が求められている。また、組込み製品の開発では全く新しい仕様の製品を開発することは稀で、ほとんどの開発で従来製品と共通点のある製品を作る。このため、組込みソフトウェアはゼロから作り上げるということはほとんどなく、従来製品のソフトウェア資産の多くを再利用している。

ところが、この再利用ソフトウェア資産で問題が発生している。例えば、実績のあるソフトウェア資産を使ったものの、従来製品では想定していなかった使われ方をされ、テストされていないパスを通り、そのパスにバグがあったために問題が発生するようなケースである。

ソフトウェア資産を再利用するならば、テスト網羅率を上げる、仕様書を充実させる、再利用部分の独立性を高める、ソースコードを分かりやすくするなど十分な検証や妥当性の確認を行い、品質を十分に高めておく必要がある。しかし、開発するソフトウェアのすべてに対して完全と言えるほどのテストを

実施することはコストや開発期間などの制約条件から考えると現実的には難しい。そこで、既存のソフトウェア資産のうち価値が凝縮されており、何世代にも渡って再利用される資産を抽出し、その再利用資産の品質を高めることで、製品全体の品質が効率的に高まると考えられる。再利用資産の品質を高める手法については、これまでに様々な知見が存在するが、すでに存在するソフトウェア資産を分析し再利用資産を抽出する手法についてはまだ確立されていないと考えられる。

2. 本研究の目標

ソフトウェア再利用資産を利用した製品の品質を効率的に高めるために、ソフトウェア再利用資産の抽出方法を導きだし、価値が凝縮されており何世代にも渡って再利用される資産を特定し、組込みソフトウェアの開発効率向上と品質向上への道筋を立てることを目指す。

3. 研究活動の内容

3.1. 研究経過と確立した方法論

本研究では 3 つのソフトウェアシステムの事例について、[文献 1] [文献 2]を参考にドメイン分析を行い、UML(Unified Modeling Language)のパッケージ図（以後、本論文ではこの図を**ドメイン構造図**と呼ぶ）でソフトウェア構造（アーキテクチャ）を可視化することから始めた（図 1 の手順 1）。**ドメイン構造図**を描くことで、既存製品におけるソフトウェアシステムの静的な機能分割状況と、分割された機能モジュール間の依存関係が明確になる。

当初、我々は**ドメイン構造図**として描いたこの機能モジュール（責務を共有する関数群、または、プログラムソースファイルのセット）の静的な構造を分析することで、再利用資産としての機能モジュールが抽出でき、抽出した機能モジュールの独立性を高め、機能モジュールの信頼性の向上に力を注ぐことで、開発効率向上と品質向上が同時に実現できると考えた。ところが、**ドメイン構造図**を描くことでソフトウェアシステムの静的な構造を可視化することはできたものの、実際のソフトウェア開発担当者であっても再利用すべき資産が何になるのか根拠を持って明確に示すことはできなかった。

そこで、次に考えたのは分割した機能モジュールの重要度を明確にするため、製品を取り巻く環境となる市場要求を分析することである。そして、先にドメイン分析で分離・明確化した機能モジュールが市場要求のどの部分を実現しているのかを可視化することにした。まず [文献 3]にある要求品質展開表の作成手順を使って製品に求められる市場要求を 3 次に展開し、展開した要求に対して[文献 3]の技術展開アプローチとして「お客様要求度」「ビジネス目標」「実現難易度」の 3 つの視点で要求の重要度を分析した。次に、市場要求を実現している機能モジュールを 3 次に展開した要求項目にマッピングすることで、現行のソフトウェアシステムが市場要求をどのように実現しているのかを可視化し、**市場要求・ソフトウェア実装対応表**にまとめた（図 1 の手順 2 と手順 3）。

これらの施策により、現行のソフトウェアシステムの静的構造が**ドメイン構造図**として可視化され、**市場要求・ソフトウェア実装対応表**によって機能モジュールやソフトウェアシステムのアーキテクチャのどの部分が重要な市場要求を実現しているのかが明確になった。そして、**ドメイン構造図**と**市場要求・ソフトウェア実装対応表**をいくつかのパターンで分析することにより、再利用すべきソフトウェア資産を抽出することができた。

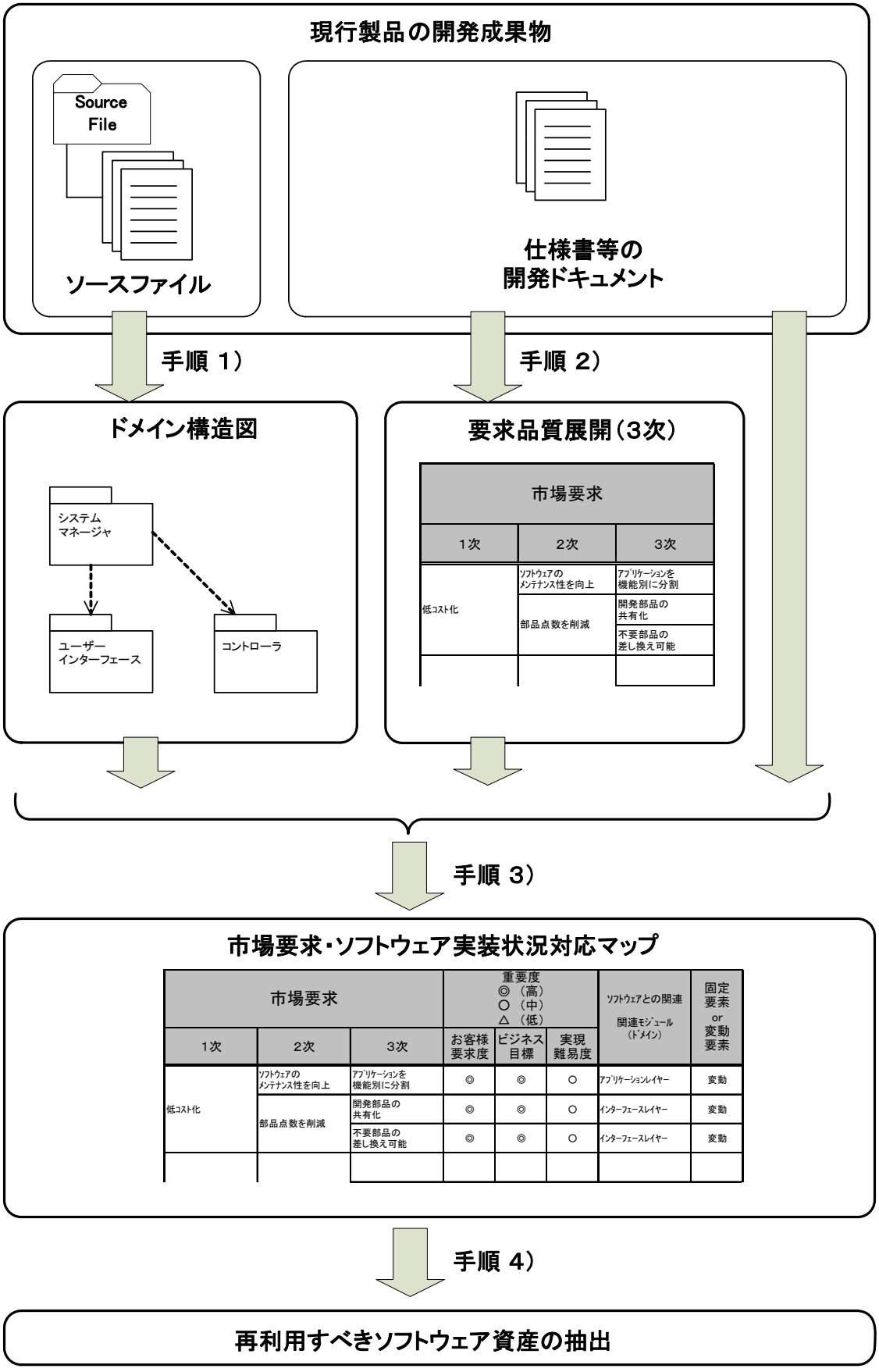


図 1 ソフトウェア再利用資産の抽出手順

3.2. 再利用すべきソフトウェア資産抽出の具体的な手順

以下に、再利用すべきソフトウェア資産抽出の具体的な手段を示す。

【ドメイン構造図の作成】

1) 現行製品のソフトウェアシステムにおいて、ソースファイルのフォルダ構成などから静的な機能モジュール(責務を共有するプログラムソースのセット)に分割し、括りだした機能モジュール(パッケージ)に共有する責務の名前を付け、機能モジュール間の依存関係を UML のパッケージ図を使って可視化する。これにより、ソフトウェアシステムのドメイン構造が明確になる(図 1 の手順 1, 付録 1-1, 2-1, 3-1)。

【市場要求・ソフトウェア実装対応表の作成】

2) 次に、この製品の市場要求を洗い出し、求められる具体的な機能や性能が明確になるように要求を 3 次に展開する(図 1 の手順 2, 付録 1-2, 2-2, 3-2)。

3) 展開した製品の市場要求に対して「お客様要求度」「ビジネス目標」「実現難易度」の視点で 3 段階にランクを付け、2)に追記する。このとき ISO14598-1(JIS X 0133-1 ソフトウェア製品の評価－全体的概観)の手法を用いてランク付けしてもよい。「お客様要求度」のレベルはお客様満足度調査結果、ユーザからのクレーム数、カタログなどのお客様説明資料に明記されている内容、適合する外部規格等のデータや情報から判断する。ビジネス目標は競合他社や市場環境によって組織として実現しなければいけない要求の重要度を記述する。実現難易度はソフトウェアシステムを開発したソフトウェアエンジニアが現行製品の開発経験をもとに記述する。これら 3 つの視点は最終的な再利用資産の抽出時に優先度を付けるために用いる(図 1 の手順 2, 付録 1-2, 2-2, 3-2)。

4) 2)で展開した市場要求に対して、この要求を実現している 1)で括りだした機能モジュールをマッピングし、同時にマッピングした機能モジュールが固定要素なのか変動要素なのかを記述する。固定要素か変動要素かは機能モジュールの設計コンセプトと今後の市場予測から製品を開発したソフトウェア技術者が判断する。機能モジュールではなく、その他の方法で市場要求を実現している場合はその旨をソフトウェアとの関連の欄に記入し、**市場要求・ソフトウェア実装対応表**を完成させる(図 1 の手順 3, 付録 1-2, 2-2, 3-2)。

【再利用資産の抽出】

5) 1)で作成した**ドメイン構造図**と 2)～4)で作成した**市場要求・ソフトウェア実装対応表**から再利用可能な資産を抽出する。お客様要求度及びビジネス目標が高く、要求を実現している機能モジュールが固定要素であれば、その機能モジュール(実際にはソースファイルのセットや仕様書等)自体が再利用すべきコア資産の中心となる。実現難易度はお客様要求度やビジネス目標のレベルが同等だった場合、実現難易度が高い方を優先する。(実現難易度が高い方が容易にまねできないため) また、お客様要求度やビジネス目標の重要度が高く、要求を実現する機能モジュールが変動要素の場合は、プログラムソースそのものではなく、機能モジュールのインターフェース仕様やシミュレーションテスト環境、テストケース、またソフトウェアシステムを実現するアーキテクチャやフレームワークなどが再利用資産にならないかどうかを分析する(3.3.2 ケーススタディ 2 および 3.3.3 ケーススタディ 3 で具体例を説明する)。

3.3. ケーススタディ

本項で再利用資産の抽出の 3 つのケーススタディを示す。

3.3.1. ケーススタディ 1 :再利用すべき機能モジュールが固定要素でかつ、市場要求の重要度が高い場合

このケース(付録 1-1, 1-2)では**市場要求・ソフトウェア実装対応表**(表 1)の中で、お客様重要度とビジネス目標が◎でかつ、関連モジュールが固定要素のものが 2 つ存在している。この 3 つの機能モジュールが重要な再利用資産であるという分析結果はこのソフトウェアシステムを開発したソフトウェア技術者の感覚とも一致している。このように市場要求が高く要求を実現しているモジュールが固定要素の場合は、この機能モジュールを構成しているプログラムソースセット、要求仕様書、機能仕様書、インターフェース仕様書、テスト仕様書、テスト環境などが再利用の対象となる。再利用すべき資産とプログラムソースセットが一致しているため再利用資産としての認知がしやすい。このようなケースでは、図 2 の**ドメイン構造図**の分析結果を使って、再使用資産として抽出された機能モジュールを他の機能モジュールとの依存関係ができるだけ疎になるようにインターフェースを見直すと、機能モジュールの凝集度が増し独立性を高めることができる。独立性の高まったモジュールに対して網羅性の高いテストを実施すれば、再利用資産の信頼性が高まる。

また、固定要素の再利用資産の信頼性を高めることに成功したならば、次に市場要求の重要度が高く、変動要素の機能モジュールの信頼性を高める施策を行う。具体的な方法についてはケーススタディ 2 とケーススタディ 3 に記述する。

表 1 ケーススタディ 1 市場要求・ソフトウェア実装対応表

市場要求			重要度 ◎(高) ○(中) △(低)			ソフトウェア関連モジュール	
1次	2次	3次	お客様要求度	ビジネス目標	実現難易度	名称	固定/変動
利用者の満足度が高い	画面レイアウトが見やすい	画面の明るさを調整できる	△	△	△	User Interface	変動
		各情報の表示レイアウトが見やすい	○	○	△	User Interface	変動
	有益な表示内容	故障情報が表示される	◎	◎	○	User Interface	変動
						DataManager1・HardwareDriver	固定
						Sensor	変動
		接続機器の状態が表示される	◎	◎	○	DataManager1・HardwareDriver	固定
						User Interface	変動
保守が容易	実機、接続機器の診断機能を有する	故障記録機能を有する	◎	○	○	Sensor	変動
						DataManager1・HardwareDriver	固定
		接続機器の自動試験の機能を有する	○	◎	◎	User Interface	変動
						DataManager1・HardwareDriver	固定
						Diagnostician	変動
		操作ログ機能を有する	○	○	○	Application1	固定
		各記録をPCに読み出せる	◎	◎	○	DataTranslator	固定
	S/Wの更新が容易	PCからローディングが行える	△	○	○	DataTranslator	固定
						DataManager1・HardwareDriver	固定

3.3.2. ケーススタディ 2:市場要求として重要度は高いが、その要求を実現する機能モジュールが変動要素の場合

このケース(付録 2-1, 2-2)は**市場要求・ソフトウェア実装対応表**(表 2)の中で「お客様要求度」と「ビジネス目標」の重要度が◎の要求を実現している機能モジュールには固定要素と変動要素が含まれる。このケースではソフトウェアエンジニアは重要な市場要求を実現するために、固定要素と変動要素の機能モジュールを意識的に分割していた。**市場要求・ソフトウェア実装対応表**の結果をもとに市場要求の重要度が高く、固定要素の機能モジュールはケーススタディ1の場合と同様にプログラムソースセットを中心に再利用を考え、変動要素に関しては他の機能モジュールに与える影響が最小限となるようなインターフェース仕様になっているかどうかを確認する。

変動要素と位置づけた機能モジュールに対しては変動する要素が何かを分析し、要求が変化し、改変したソフトウェアが要求仕様に合致しているかどうかを検証することが重要だと判断されたならば、その変動要素の機能モジュールをテストするテスト仕様やテストケース、テスト環境の再利用を検討する。例えば、コストダウンによるハードウェアの変更が予想されたり、外部規格の変更で対応するソフトウェアの修正が余儀なくされたりするような場合は、外部環境の変化を見越して変化した環境下で対象となる機能モジュールが正しく動作しているかどうかを検証するためのテスト仕様書の作成やテスト環境の構築を目指し、変更の頻度の多いプログラムソースそのものではなく、構築したテスト仕様やテスト環境を資源化し再利用する。

ケーススタディ 2 のように、すでに確立されたアーキテクチャを明示化し、プロジェクトメンバがアーキテクチャのコンセプトを再認識し、プロジェクト全体でソフトウェアシステムの構造を洗練することが開発効率と製品品質の組織的な向上につながる。

表 2 ケーススタディ 2 市場要求・ソフトウェア実装対応表

市場要求			重要度 ◎(高) ○(中) △(低)			ソフトウェア関連モジュール	
1次	2次	3次	お客様要求度	ビジネス目標	実現難易度	名称	固定/変動
低コスト化	開発費を削減する	機能を限定する	◎	◎	△	Interface Application 1	固定
	部品数を減らす	—	◎	◎	○	Interface Application 2 Hardware Driver	変動
上位互換	規格を統一する	上位インターフェースを統一する	○	○	○	Interface Application 1	固定
		下位インターフェースを統一する	○	○	○	Interface Application 2 Hardware Driver	変動
メンテナンス性向上	動作状況がわかる	表示機能を追加する	△	△	△	Error Application	変動

3.3.3. ケーススタディ 3:市場要求として重要度は高いが、その要求を実現しているしくみが機能モジュールではない場合

このケース(付録 3-1, 3-2)は、**ドメイン構造図**(図 2)にあるように、ソフトウェアシステムがアプリケーションレイヤ、インターフェースレイヤ、ドライバレイヤという階層構造が重要度の高い市場要求の実現方法として採用されている。ハードウェアの変更に伴いドライバレイヤ内のモジュールに変更があったとしても、アプリケーションレイヤ、インターフェースレイヤのレイヤ間インターフェースを変えないで済むような構

造が取られているケーススタディ 3 では個々のアプリケーションモジュールの市場要求の重要性は均一であり、ユーザ要求に合わせてアプリケーションモジュールを自由に選択できるというソフトウェアシステムの構造自体が重要な資産となっている。したがって、ケーススタディ 3 の再利用すべき資産はレイヤリングのアーキテクチャ、レイヤ間のインターフェース仕様、インターフェースのテスト仕様やテストケース、アプリケーションレイヤ内の個々のアプリケーション機能モジュール群といったものになる。

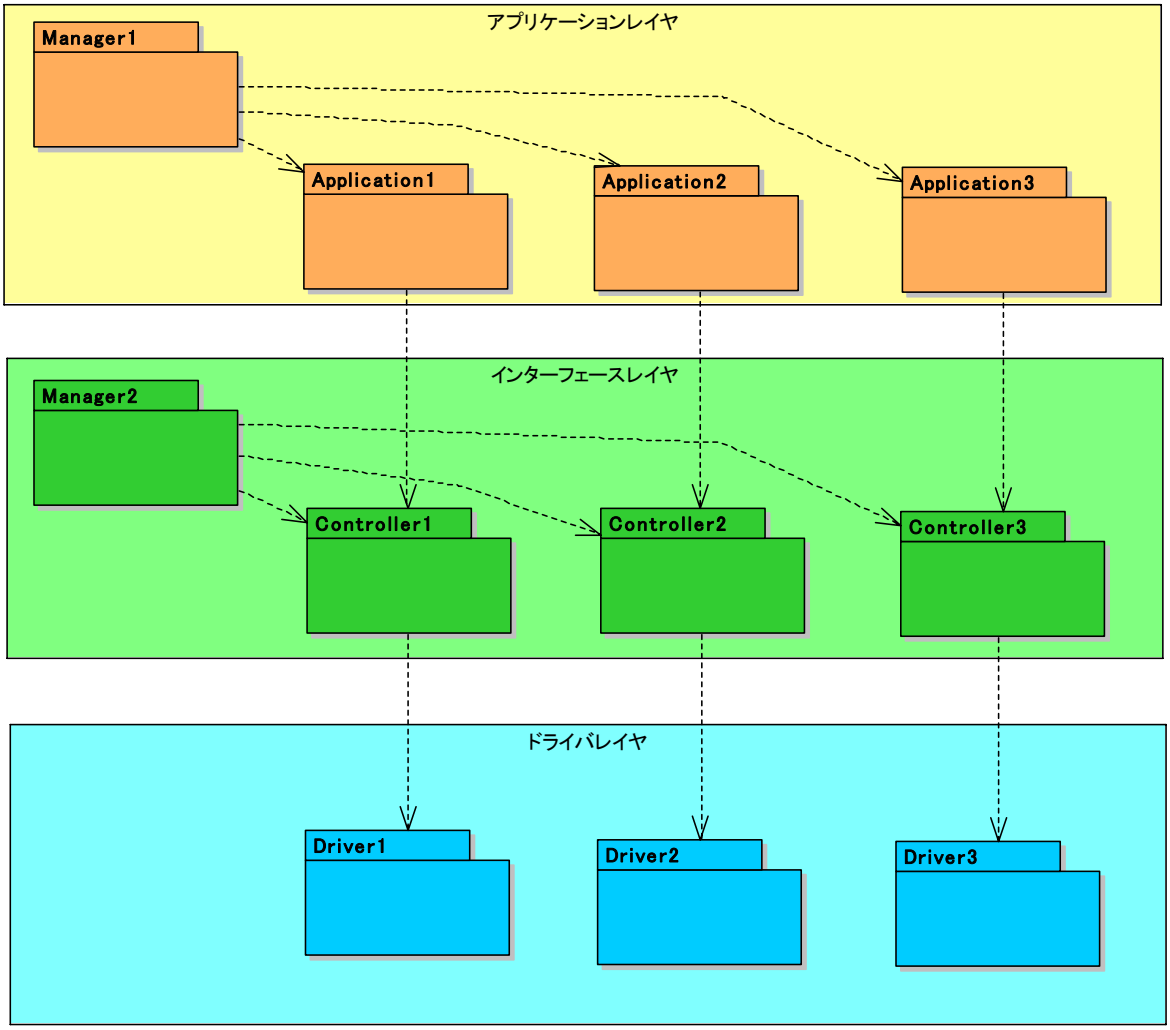


図 2 ケーススタディ 3 ドメイン構造図

4. 研究の成果

3 つの実際に市場投入されている組込みソフトウェアシステムで本研究の方法論を使って分析をした結果、再使用すべきソフトウェア資産の抽出パターンがわかってきた。

現行製品の静的なモジュール構造を**ドメイン構造図**で分析することで、ソフトウェアシステムの構造やアーキテクチャが明確になり、さらに**市場要求・ソフトウェア実装対応表**を作成することで市場要求をどのように機能モジュールやアーキテクチャが実現しているのかが可視化された。そして、**ドメイン構造図**と**市場要求・ソフトウェア実装対応表**を分析することにより、再利用すべき資産が機能モジュール(プログラムソースセットを中心とした情報)そのものなのか、それとも、インターフェース仕様やテスト環境、テストケースなのか、また、システムを実現するしくみとしてのアーキテクチャなのかがわかる。

これにより、同じ市場に組込み機器を投入し続け、製品開発を繰り返す中で構築されていった市場や顧客、プロダクトを製造する組織に最適化したソフトウェアシステムの構造が可視化された。本論文で構築した手法を使うことで、組込みソフトウェアシステムにおける再利用資産の抽出が可能になり、抽出した再利用資産の品質を効果的に高めることにつながると考えられる。

5. 本研究の考察と今後の展開

多くの組込みソフトウェアシステムの場合、歴代の組込み製品で培われてきた市場要求を実現するためのソフトウェアシステムのしくみが存在する。このしくみとしてのアーキテクチャが明示的に可視化されていることは少なく、同じ市場で同じプロジェクトメンバでシステムを構築してきたベテランの組込みソフトエンジニア(アーキテクト)の頭の中に暗黙知として存在していることが多い。このような重要な情報はアーキテクトがそのプロジェクト内に長年留まり、次世代の技術者に継承されていけばよいが、ソフトウェアシステムが巨大化して一人のアーキテクトがシステム全体を把握しきれなくなったり、技術者が入れ替わることによって構築されたソフトウェアシステムのアーキテクチャの設計思想が伝わらなくなったりすると、とたんにソフトウェアの開発効率が下がり品質が劣化する。

本研究で実施・提案した**ドメイン構造図**と**市場要求・ソフトウェア実装対応表**の作成による再利用資産の抽出手法を使うことで、組込みソフトウェアのベテランソフトウェア技術者の頭の中に埋もれていた市場に最適化したソフトウェアシステム実現の構造が可視化され、どこに注力すれば製品群の開発を効率化し、製品群全体の品質を向上できるかが分かるようになる。

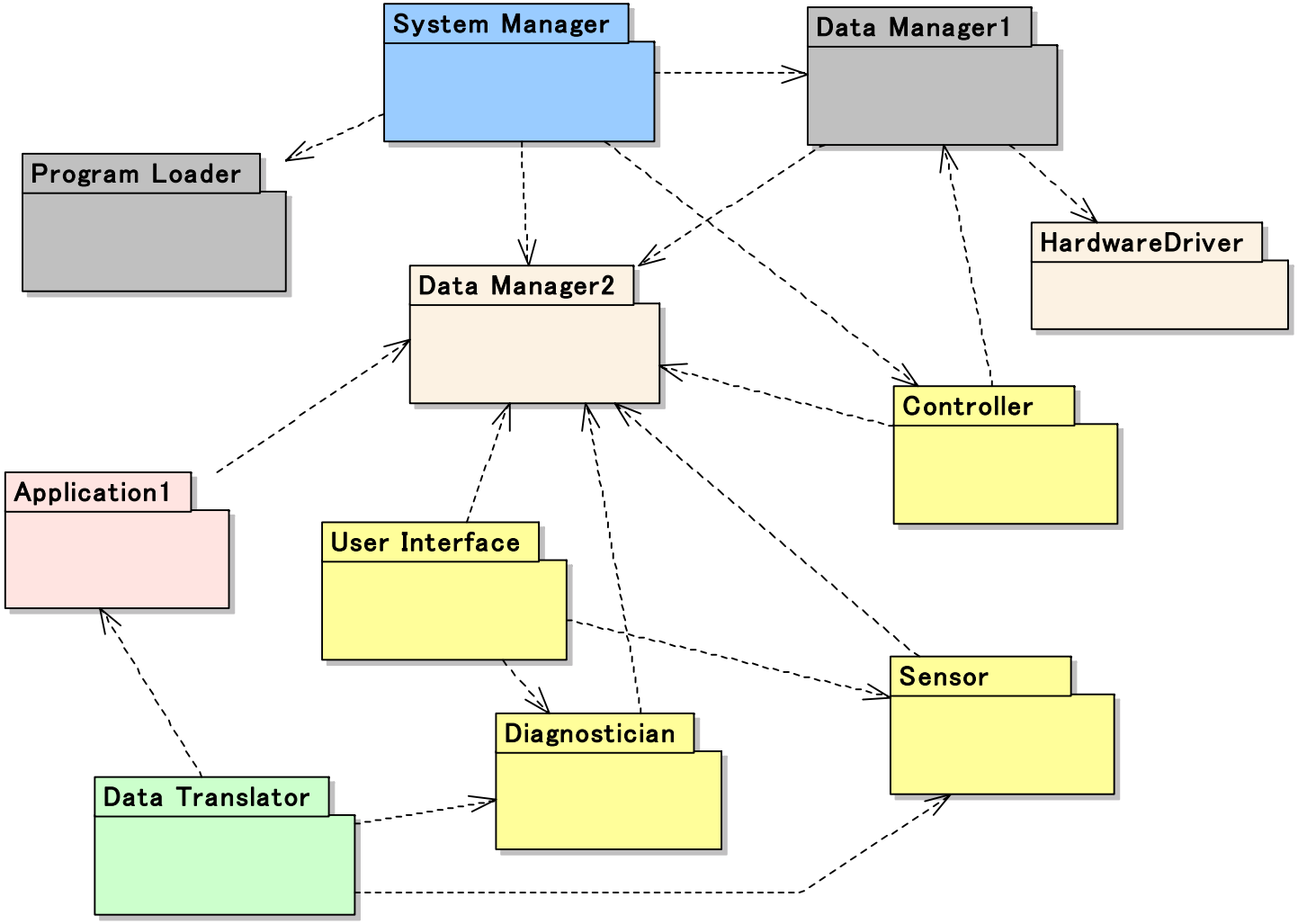
なお、今回の3つのケーススタディでは現行システムが市場要求に最適化されており、アーキテクチャ自体に大きな問題がない場合であった。もしも、従来のソフトウェアに追加要求プログラムの付け足しを続け肥大化してしまったようなシステムであった場合は、現在のソフトウェアシステムの機能分割の構造や実現方法は一度壊して再構築しなければならない場合もあると考えられる。また、市場要求の急激な変化により、これまでの市場要求を実現してきたアーキテクチャが要求に整合せず、かえって効率を落としてしまう場合もあるかもしれない。このような場合は、現在のソフトウェアシステムの**ドメイン構造図**と**市場要求・ソフトウェア実装対応表**を作成し、変化が見込まれる市場要求を新たに分析し、その要求を満たすために最適と考えられるソフトウェアシステムの構造と比較することで、現在から未来へどのようにソフトウェアシステムのアーキテクチャをシフトすればよいかが見えてくると考えられる。

また、本研究では現行製品のソフトウェアシステムの静的な構造を**ドメイン構造図**で可視化したが、**ドメイン構造図**では製品に求められる非機能要件としての性能要求や、システムの動的な振る舞いを表すタイミング、状態遷移、シーケンスなどは明示していない。組込みソフトウェアシステムのアーキテクチャは静的な構造だけで表されるものではないため、今後の展開としては現行のソフトウェアシステムのタイミング図、ステートチャート図、シーケンス図などを UML などの表記法で描いて市場要求の実現方法を可視化し、それらの情報も利用することで再利用すべき資産の抽出がさらに進むと予想される。

参考文献

[1] Will Tracz, ソフトウェア再利用の神話, ピアソンエデュケーション(2001)
[2] 渡辺博之, 渡辺政彦, 堀松和人, 渡守武和記, 組み込み UML, 翔泳社(2002)
[3] 赤尾洋二, 品質展開入門, 日科技連出版社(1990)
[4] ISO14598-1(JIS X 0133-1 ソフトウェア製品の評価－全体的概観)

＜付録 1-1 ケーススタディ 1　ドメイン構造図＞

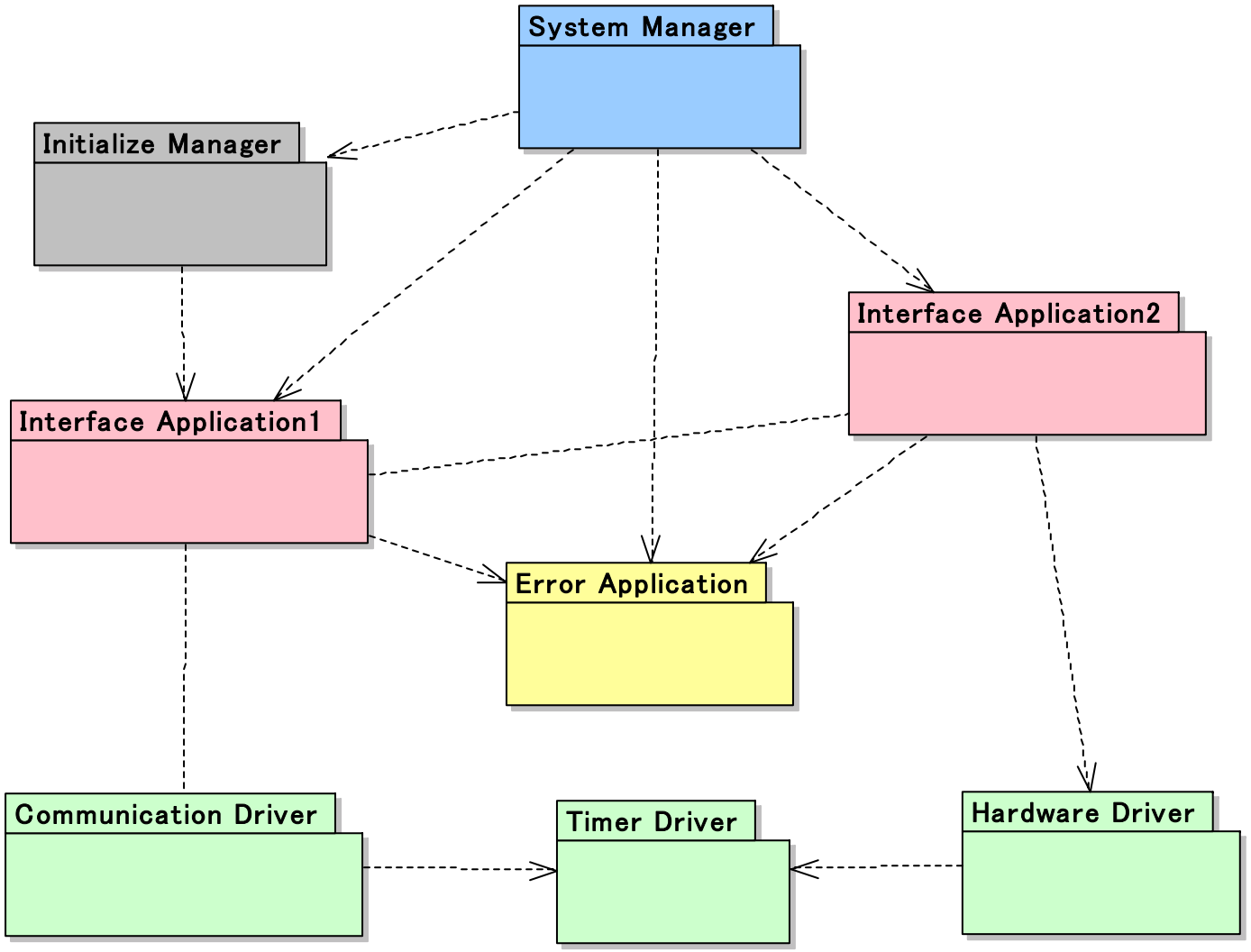


＜付録 1-2 ケーススタディ 1　市場要求・ソフトウェア実装対応表＞

市場要求			重要度 ◎(高) ○(中) △(低)			ソフトウェア関連モジュール	
1次	2次	3次	お客様要求度	ビジネス目標	実現難易度	名称	固定/変動
利用者の満足度が高い	画面レイアウトが見やすい	画面の明るさを調整できる	△	△	△	User Interface	変動
		各情報の表示レイアウトが見やすい	○	○	△	User Interface	変動
	有益な表示内容	故障情報が表示される	◎	◎	○	User Interface	変動
						DataManager1・HardwareDriver	固定
						Sensor	変動
		接続機器の状態が表示される	◎	◎	○	DataManager1・HardwareDriver	固定
						User Interface	変動
保守が容易	実機、接続機器の診断機能を有する	故障記録機能を有する	◎	○	○	Sensor	変動
						DataManager1・HardwareDriver	固定
		接続機器の自動試験の機能を有する	○	◎	◎	User Interface	変動
						DataManager1・HardwareDriver	固定
						Diagnostician	変動
		操作ログ機能を有する	○	○	○	Application1	固定
		各記録をPCに読み出せる	◎	◎	○	DataTranslator	固定
	S/Wの更新が容易	PCからローディングが行える	△	○	○	DataTranslator	固定
						DataManager1・HardwareDriver	固定

※　枠で囲った部分はお客様要求度が高い項目を示す

＜付録 2-1 ケーススタディ 2　ドメイン構造図＞

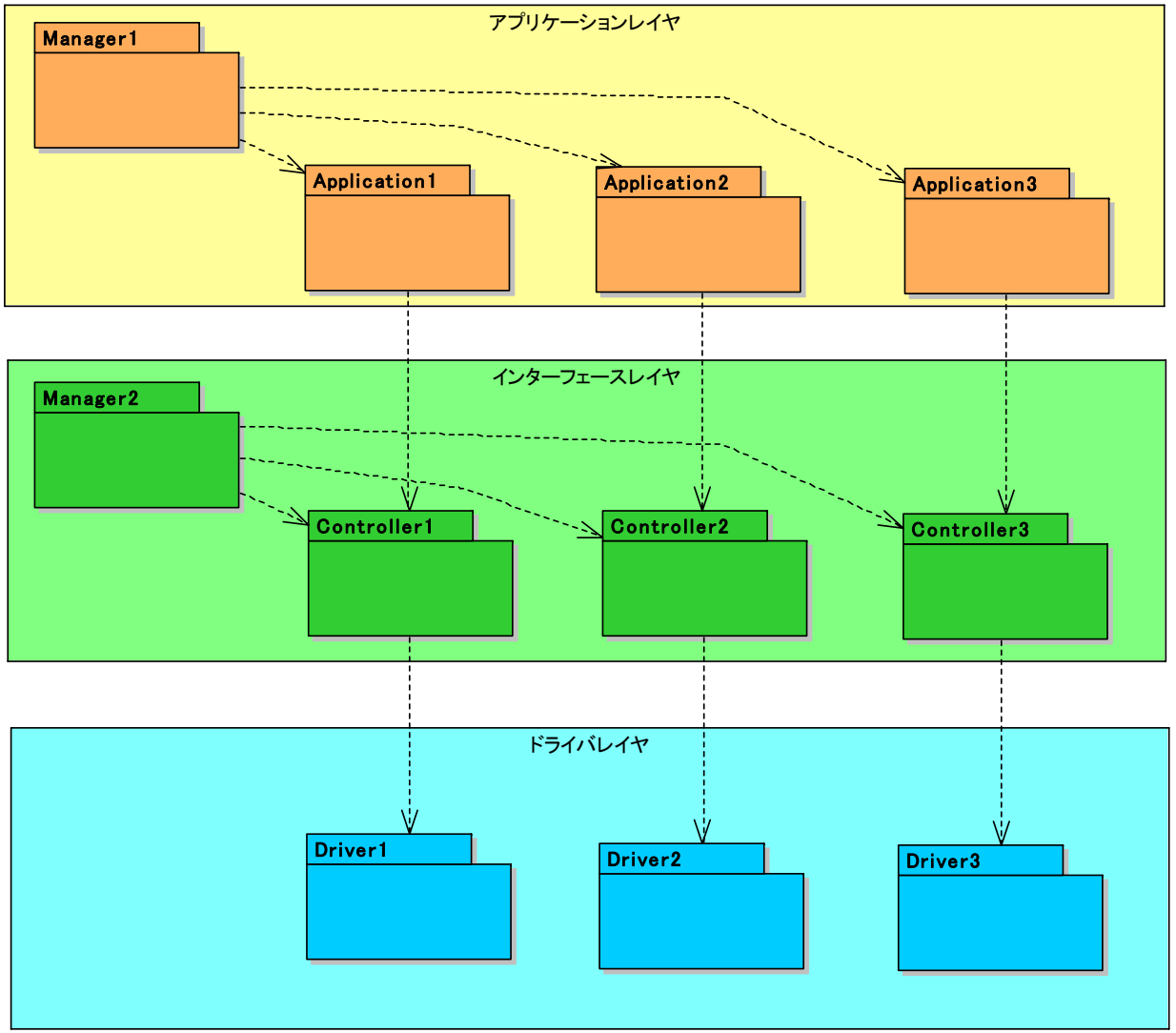


＜付録 2-2 ケーススタディ 2　市場要求・ソフトウェア実装対応表＞

市場要求			重要度 ◎(高) ○(中) △(低)			ソフトウェア関連モジュール	
1次	2次	3次	お客様要求度	ビジネス目標	実現難易度	名称	固定/変動
低コスト化	開発費を削減する	機能を限定する	◎	◎	△	Interface Application 1	固定
	部品数を減らす	―	◎	◎	○	Interface Application 2 Hardware Driver	変動
上位互換	規格を統一する	上位インターフェースを統一する	○	○	○	Interface Application 1	固定
		下位インターフェースを統一する	○	○	○	Interface Application 2 Hardware Driver	変動
メンテナンス性向上	動作状況がわかる	表示機能を追加する	△	△	△	Error Application	変動

※　枠で囲った部分はお客様要求度が高い項目を示す

＜付録 3-1 ケーススタディ 3　ドメイン構造図＞



＜付録 3-2 ケーススタディ 3　市場要求・ソフトウェア実装対応表＞

市場要求			重要度 ◎(高) ○(中) △(低)			ソフトウェア関連モジュール	
1次	2次	3次	お客様要求度	ビジネス目標	実現難易度	名称	固定/変動
低コスト化	ソフトウェアのメンテナンス性を向上	アプリケーションを機能別に分割	◎	◎	○	アプリケーションレイヤー	変動
	部品点数を削減	開発部品の共有化	◎	◎	○	インターフェースレイヤー	変動
		不要部品の差し換え可能	◎	◎	○	インターフェースレイヤー	変動
利用者の使用満足度を向上	店員の操作性を向上	媒体補充の作業回数を削減	○	○	○	Application1 Driver1	変動
		タッチパネル入力による機器操作	○	○	△	Driver3	固定
		エラー対処方法を詳細表示	○	○	△	Driver3	固定
	客の操作性を向上	取引時間の短縮	○	○	△	Application2 Driver2	固定
機器のメンテナンス性を向上	故障原因の特定	ログ機能	△	△	△	Application3	固定

※　枠で囲った部分はお客様要求度が高い項目を示す