Vol.2010-SE-168 No.14
Vol.2010-EMB-17 No.14
2010/6/2

# 事例分析に基づく組込みシステムに適したソフトウェアアーキテクチャの提案

海老原 健一[*1] 満田 成紀[*2]
福安 直樹[*2] 鯵坂 恒夫[*2]

現在組込みシステムは大規模・複雑化しているためモデル中心設計などのソフトウェア開発手法の導入が積極的に進められている．組込みシステムではハードウェアとソフトウェアの連携を考慮したシステム開発が必要であり，ハードウェアの特性を取り込める開発手法が望まれている．

本研究では，2 つの組込みソフトウェア開発事例を分析し，組込みシステムに適したソフトウェアアーキテクチャを提案する．このアーキテクチャは 3 層構造を持ち，ハードウェアを制御する層，アプリケーション機能を実現する層，それらを連絡する層に分けられる．また，提案するアーキテクチャを利用した開発プロセスを提案する．

# Software Architecture Design for Embedded Systems based on Case Study Analisys

Kennichi Ebihara,[*1] Naruki Mitsuda,[*2]
Naoki Fukuyasu[*2] and Tsuneo Ajisaka[*2]

Because embedded systems are becoming large-scale and complex, software design methods, such as model-centered design, are brought in positively for embedded system developments. For embedded systems, the system development method in consideration of cooperation of hardware and software is required.

We analyze two development cases of embedded software, and propose software architecture suitable for embedded systems. This architecture is divided into three layers: the layer which controls hardware, the layer which realizes application functions, and the layer which connects other layers. The development process using this architecture is also proposed.

## 1. はじめに

近年，組込みシステムの大規模・複雑化が著しい．組込みシステムにおいては，不具合発生時の製品回収リスクを避けるため，高品質なソフトウェアが必要である．このため，ソフトウェアのモデルを利用した開発手法の導入がすすめられている[1)]．この時，組込みシステム特有のハードウェア特性や外乱といった環境要素を考慮したモデル中心の開発手法が必要である．

本研究では，組込みシステムに適したモデル中心開発手法の提案を目的として，モデル中心設計を用いた 2 つの組込みソフトウェア開発プロジェクト事例を分析する．分析結果に基づき，環境要素の簡便なハンドリングを可能とするソフトウェアレイヤの存在を示し，これを取り入れたソフトウェアアーキテクチャの提案を行う．

## 2. 組込みシステム開発事例の分析

### 2.1 分析対象とした組込みシステム事例

分析対象として，小型飛行船の自律航行システムと複合機ビジネスアプリケーションの開発事例を利用した．それぞれのアプリケーションの概要について記述する．

#### 2.1.1 飛行船自律航行システム

飛行船自律航行システムの開発は，MDD ロボットチャレンジ 2009[2)] への参加の中で行われた．このシステムは，小型飛行船が地上にある出発地点から上空にある 2 つの風船を経由して着陸地点へ自律飛行するシステムである．

図 1 に示すように，システム構成として飛行船・地上センサ・基地局 PC がある．地上センサは超音波センサのマトリクスとなっており，飛行船から発せられた超音波の受信時刻情報を基地局 PC に送信する．飛行船は駆動系として上昇下降用プロペラ 1 つと旋回用プロペラ 2 つを持っている．その他に方位センサ，高度センサ，Zigbee 通信装置を持ち，方位・高度・電池残量を基地局 PC に送信する．基地局 PC は地上センサと飛行船から受信したデータを解析し，プロペラを回すモータの出力数値を飛行船に送信することで，目的地へ飛行させる．

*1 現在，和歌山大学大学院システム工学研究科
Presently with Graduate School of Systems Engineering, Wakayama University
*2 現在，和歌山大学システム工学部
Presently with Faculty of Systems Engineering, Wakayama University

Vol.2010-SE-168 No.14
Vol.2010-EMB-17 No.14
2010/6/2

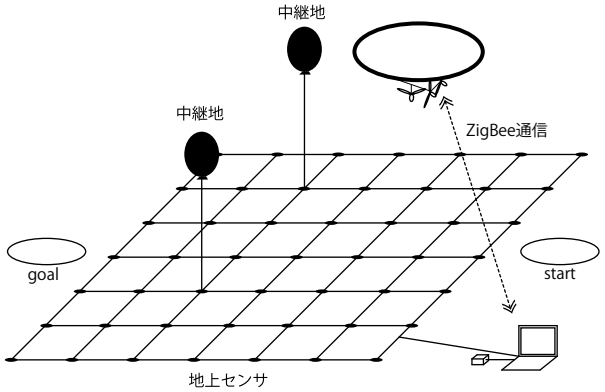


**図 1** 自律航行システムの構成

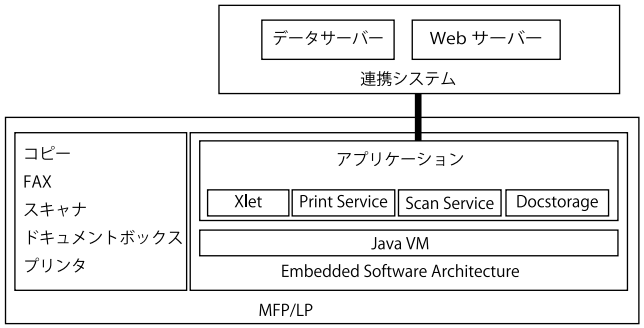


**図 2** 複合機ビジネスアプリケーションの構成

#### 2.1.2 複合機ビジネスアプリケーション

複合機ビジネスアプリケーションの開発は，RICOH & Sun Java challenge 2009[3] への参加の中で行われた．開発対象としたシステムは「ビジネスプロジェクトにおけるデータ収集管理システム」である．このシステムでは，プロジェクトに関連する雑誌等の切り抜きや，手帳に書き込んでいるメモといった様々な「記事」をスキャナから取り込み，そのまま複合機上からシステムに登録する．システム内ではメンバーから提供された記事を全員で共有し，コメント等を交換することで新たなアイデアのきっかけとする．

複合機ビジネスアプリケーションの構成を図 2 に示す．複合機の基本機能としてコピー・FAX・スキャナ・ドキュメントボックス・プリンタがある．また，複合機はネットワークを通じて他のシステムと連携することが可能である．複合機上のソフトウェア開発は Java で実装することとされており，操作パネルを制御する Xlet，スキャンを行う Scan Service，プリントを行う Print Service，複合機中にデータを保存する DocStorage といった API が提供されている．

### 2.2 クラス構成の分析

本研究では，各開発事例のクラス構成に着目し，その役割によって 2 つのグループに分けることができると考えた．

自律航行システムのクラス図を図 3 に，複合機ビジネスアプリケーションのクラス図を図 4 を示す．それぞれのクラス構成を分析した結果，アプリケーション機能を実現するためのクラスと，ハードウェア制御を簡便化するためのクラスに分けることが可能であると考えた．

自律航行システムでは，センサデータの送受信や旋回 PID 制御を行うクラス，飛行船の高度や方位といった各種センサ値を格納するクラスがあり，これらはハードウェア制御を簡便化するためのクラスである．一方，離着着陸動作や目的地への誘導などの飛行戦略にかかわるクラスがアプリケーション機能を実現している．

複合機ビジネスアプリケーションでも，サーバ通信やスキャナ・印刷といった環境が提供している API の利用を簡便化するためのクラスがある．さらにこのアプリケーションでは MVC モデルを基調としたモデリングをしており，そのための抽象クラスである画面や制御クラスも，操作パネルというハードウェアの制御を簡便化するクラスと言える．

### 2.3 組込みシステム開発で考慮すべき要求

分析した結果から，組込みシステム開発では，ソフトウェアの設計時において考慮すべき要求が 2 種類あると考える．

- アプリケーション機能を実現するための要求
- ハードウェア制御を簡便化するための要求

1 つ目は一般的なトップダウンからの要求である．ユースケース等で表記され，ユーザから見たシステムとしての機能を明らかにするものである．

2 つ目はシステムの対象ドメインに依存したボトムアップからの要求である．組込みシステムでは，アプリケーション機能を実現するためには，対象ドメインに含まれる様々なハードウェアとの協調が必要である．しかし，多数のハードウェアを協調させるロジックは複雑であり，直接アプリケーション機能の中で実現しようとすると，複雑なロジックの分散化に

Vol.2010-SE-168 No.14
Vol.2010-EMB-17 No.14
2010/6/2

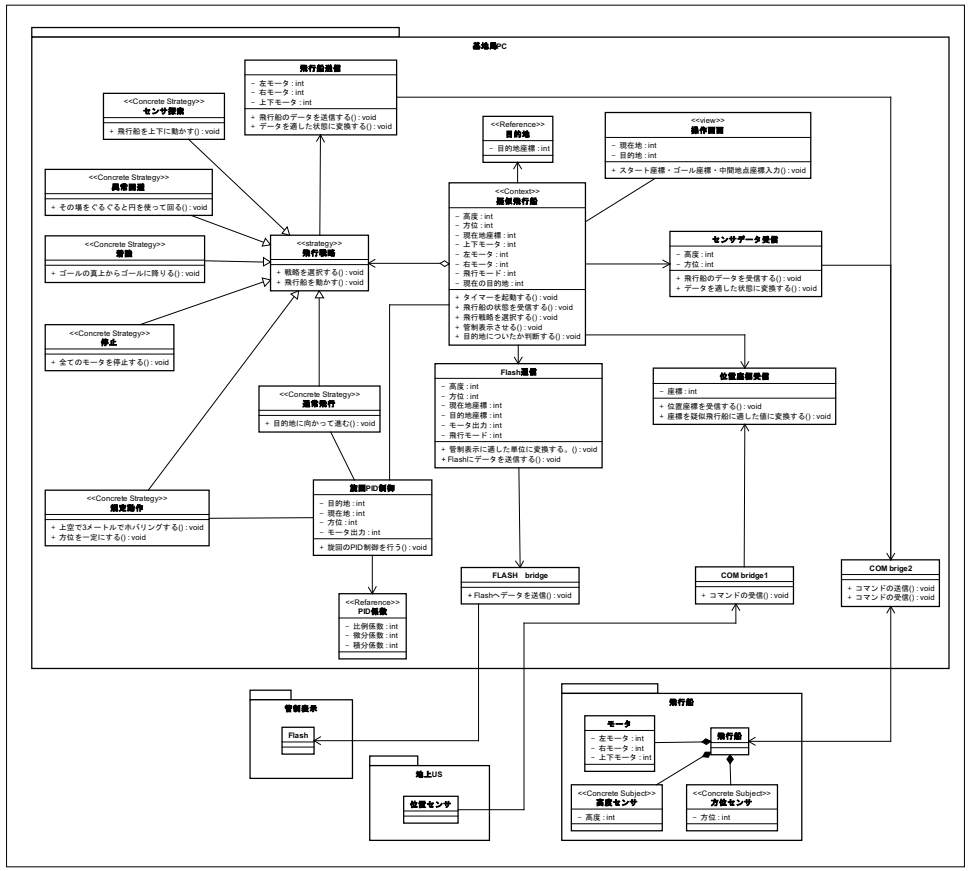


**図 3** 自律航行システムのクラス図

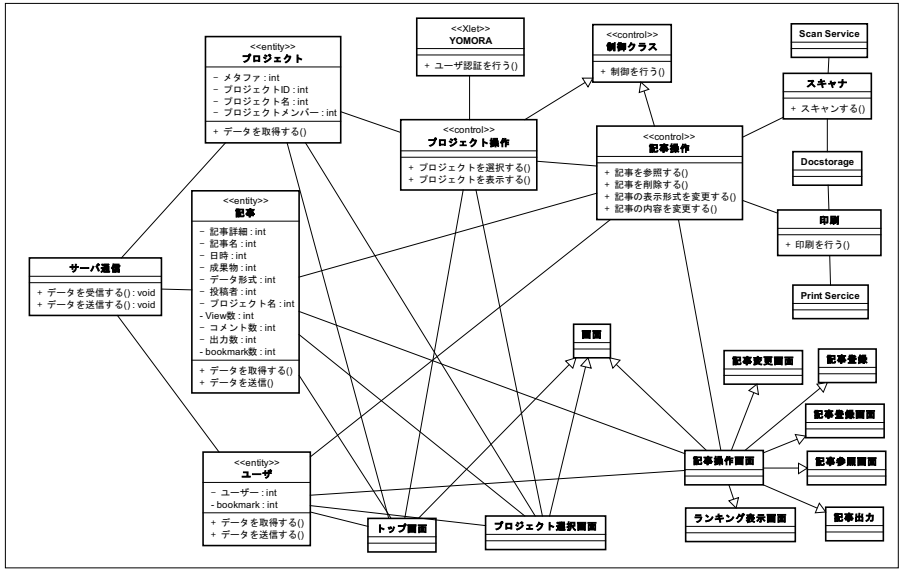


**図 4** 複合機ビジネスアプリケーションのクラス図

よって，見通しの悪いシステムとなってしまう．そこで，ハードウェア間の協調ロジックを隠蔽し，あたかもハードウェアの単体制御の組み合わせてによってアプリケーション機能が実現できるような支援機能が要求される．

## 3. 組込みシステムに適したソフトウェアアーキテクチャ

前節の事例分析により，組込みシステム開発には 2 種類の要求があることがわかった．本節では分析結果を受けて，組込みシステムに適したソフトウェアアーキテクチャを提案する．

### 3.1 組込みソフトウェアの 3 層構造

本研究で提案するソフトウェアアーキテクチャの概念図を図 5 に示す．このアーキテクチャは 3 層構造となっており，最上部のアプリケーション層はユーザ視点の要求であるアプリケーション機能を実現する層である．最下層の制御層は実際にハードウェアを制御する層であり，すでにシステムの実装環境として提供されている API やドライバが存在する層である．間に挟まれた環境構成層が，ドライバや API 類の協調ロジックを隠蔽し，アプリ

Vol.2010-SE-168 No.14
Vol.2010-EMB-17 No.14
2010/6/2

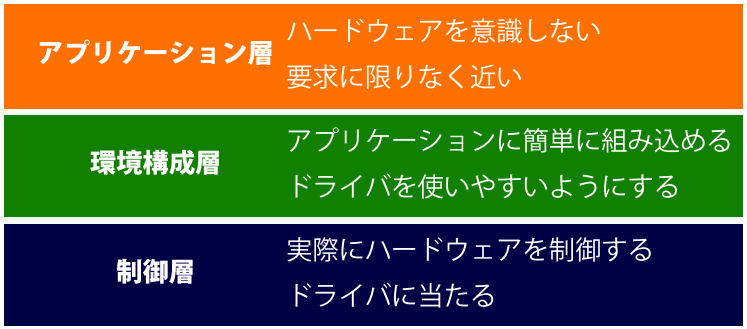


**図 5** 組込みソフトウエアの 3 層構造

ケーション機能から簡便に利用すること可能とすることを目的とした層である．

3 層に分割することでアプリケーション層は少ないドメイン知識での開発が可能となる．つまり，アプリケーション開発者は外部環境やハードウェアの制約や協調といった複雑なロジックをを意識する必要なく開発できる．加えて，アプリケーション層内での作業分担がしやすくなる．

一方，環境構成層はドメイン知識を豊富に持つ開発者が担当することになる．この層は作業分担が難しく，開発者への負荷が大きくなることが考えられるが，アプリケーションへの依存度が低いため，一度開発したソフトウェア部品を他のアプリケーションで再利用することが可能であり，同じドメインの複数アプリケーション全体を見れば負荷が調整できる．

また，環境構成層があることで，ハードウェアの仕様変更等の環境の変化への対応がより柔軟にできることになり，保守性や再利用性の良い高品質なプログラムの開発へとつながる．

### 3.2 分析事例の構造化

分析事例を 3 層構造に当てはめた結果を示す．自律航行システムを図 6 に複合機アプリケーションを図 7 に示す．

自律航行システムでは制御層にハードウェアである位置センサやモータ，方位センサなどの実際のハードウェアが当てはまる．環境構成層は飛行船を操作する際に必要なクラスが含まれている．飛行船の各種値を格納しておく擬似飛行船クラスや，PID 制御クラスなど飛行船を飛ばす上で必要なクラスが該当する．アプリケーション層は飛行方法についてのクラスが含まれている．ハードウェアを細かく規定したクラスではなく，飛行船の飛行戦略が含まれておりユーザーが要求する飛行船の飛行手順が記述されている．

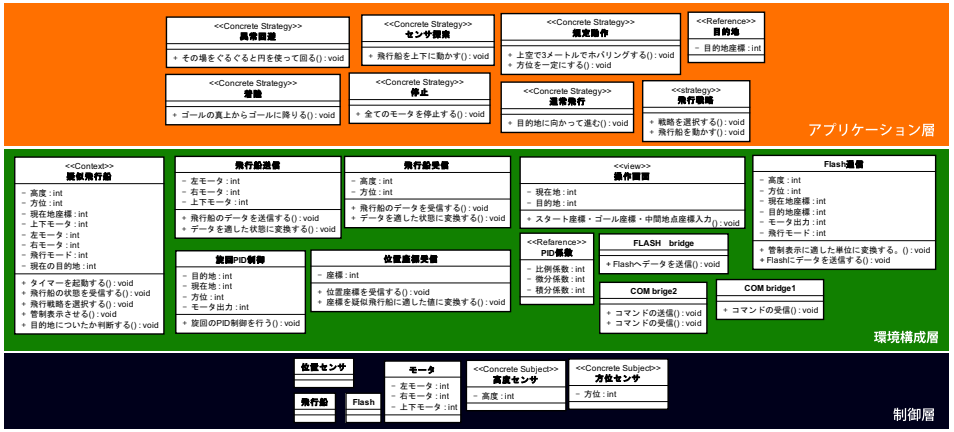


**図 6** 自律航行システムにおける 3 層構造

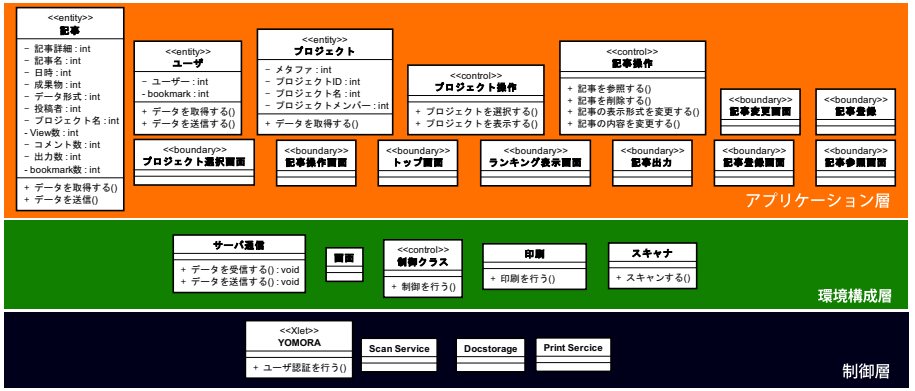


**図 7** 複合機アプリケーションにおける 3 層構造

Vol.2010-SE-168 No.14
Vol.2010-EMB-17 No.14
2010/6/2

複合機アプリケーションでは制御層に Xlet や Scan Service，Docstrage，Print Service が含まれている．これらは複合機独自の API であるため，ハードウェアの特性を理解して利用する必要がある．また，複合機の基本動作を扱う上で必要な API であり直接ハードウェアを制御しているため制御層とした．環境構成層では外部との通信を行うサーバ通信クラスや Scan Service を使い易くした印刷クラスがある．画面や制御クラスといった抽象クラスは，画面のパーツを制御するためのものであり，アプリケーションに非依存であるため環境構成層となる．アプリケーション層は記事収集システムに必要な画面や記事操作クラスが含まれている．

それぞれの環境構成層を比較すると，自律航行システムは複合機アプリケーションよりも環境構成層が充実している．自律航行システムは風や湿度といった環境による外乱と，センサやモータといったハードウェアの制御がある．特にハードウェアを制御するクラスが多くあり，複雑な制御ロジックを隠蔽するために，PID 制御や飛行船送信等のクラスが増えている．複合機アプリケーションは，既に複合機専用の API が提供されている．環境構成層ではこれらの API の利用手順を実装しただけであり，そのためのクラス数は少なくなっている．

## 4. 環境構成層を利用した開発プロセス

本節では，組込みシステムの開発において，環境構成層を利用した場合の開発プロセスについて説明する．開発プロセスを考案するにあたって，以下の事項を前提とした．

- 環境構成層はドメイン依存・アプリケーション非依存である．
- 既にある環境構成層のソフトウェア部品は利用する
- 制御層はハードウェアから与えられたドライバを使用する．

### 4.1 提案する開発プロセス

下記に提案する開発プロセスを記述する．

(1) 要求獲得
(2) 要求分析
(3) 環境分析
(4) 全体設計
(5) 実装

既存の開発プロセスに加え環境分析のフェーズを追加した．要求獲得では既存のソフトウェア開発と同様，ユーザーが求めている要求を獲得することでシステムに必要な機能を理解する．この時，ユースケースができることが望ましい．要求分析では要求獲得で得られたユースケースを木構造にして分析する．環境分析ではその結果からアプリケーション層と環境構成層との連携を行う．全体設計では 3 層構造を考えながら全体設計を行う．その後，詳細設計を経てアプリケーションの実装を行う．

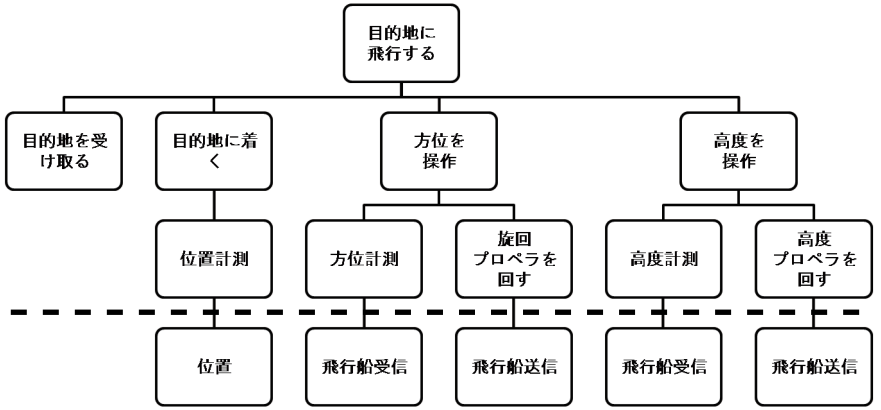


**図 8** 自律航行システムの要求構造

#### 4.1.1 要 求 分 析

要求分析では，要求獲得から得たユースケースを木構造に分解していく構造分析の作業を行う．機能を分解することで要素技術を把握することができる．

自律航行システムによる例を図 8 に示す．複合機アプリケーションによる例を図 9 に示す．

図の破線より上が要求分析により得られた要求構造である．アプリケーション層を意識して作られているため，ハードウェアを制御するためのボトムアップのアプローチはせず，ユースケースを分割していく．

自律航行システムでは初めに目的地へ飛行するため，方位の操作や目的地のデータを受け取るといった機能に分解していく．複合機アプリケーションでも自律航行システムと同様に機能を分解する．

Vol.2010-SE-168 No.14
Vol.2010-EMB-17 No.14
2010/6/2

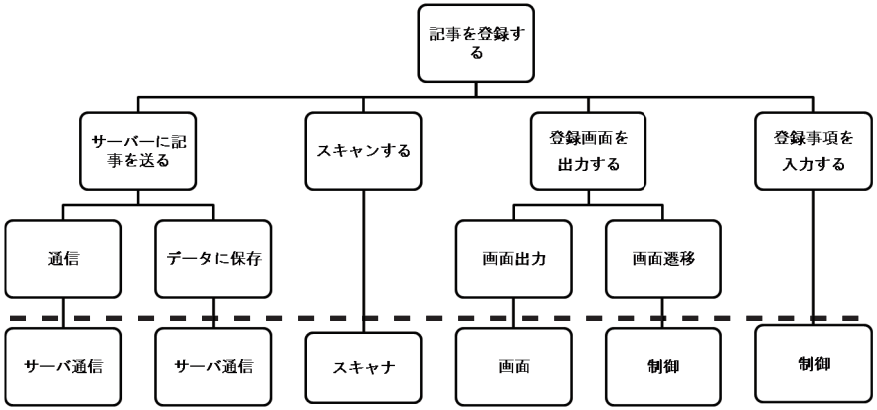


**図 9** 複合機アプリケーションの要求構造

#### 4.1.2 環境分析

ユースケースを分解して適切な要素技術を把握したあと，その下に環境構成層に存在するソフトウェア部品を当てはめていく．既存部品を当てはめていくことで，各要素技術に適切なハードウェア制御のロジックが組み込まれることになる．

図 8，図 9 では破線より下位が環境分析の結果当てはめられるソフトウェア部品である．

自律航行システムはユースケースを分割していった際に「目的地を受け取る」という要求がある．これは目的地データを受け取るというものであるが，当てはまる環境構成層がない．ここではアプリケーション側で実装することが必要となっている．

### 4.2 開発プロセスの特性

既に開発されている環境構成層のソフトウェア部品を利用することで，組込みソフトウェア開発が効率的になる．アプリケーション開発者はハードウェアの制御ロジックを完全に理解しなくとも，ハードウェアの制御ロジックを環境構成層の部品を利用して実装することができる．これによってコスト削減と品質保証が実現される．

しかし，環境構成層に適切な部品がない場合は，新たに開発を行う必要がある．環境構成層を開発する場合は環境構成層はドメインには依存するが，アプリケーションに非依存であることが重要である．また、様々なハードウェアを利用したとしてもアプリケーションが動作できる環境構成層を作る必要がある．一般にこのようなソフトウェア部品を開発するためには豊富な経験とドメイン知識が必要とされている．環境構成層に含まれるソフトウェア部品の開発に必要な経験やドメイン知識を得るための支援方法は今後の課題である．

### 4.3 プロダクトライン開発との連携

プロダクトライン開発[4] のドメインエンジニアリングでは，ハードウェアや実装環境のバリエーションを構造的に分解したフィーチャモデルを利用する．本研究で提案する 3 層構造に照らし合わせると，ドメインエンジニアリングが対象としているのは，主に制御層と考えられる．

本研究が着目している環境構成層には，アプリケーション機能の実現を簡便化することを目的に開発されたソフトウェア部品が含まれる．この層に含まれる部品の開発手法については今後の課題としているが，ここにプロダクトライン開発のフィーチャモデルが活用できると考えている．フィーチャモデルによって明らかとなる環境のバリエーションこそが，環境構成層で隠蔽すべきロジックの対象である．フィーチャモデルにしたがってコアアセットが変更されたとしても，環境構成層がその差異を吸収することで，アプリケーションロジックに影響を及ぼさないようにすることが可能である．

## 5. まとめ

本研究では 2 つの組込みソフトウェア開発事例を分析し，組込みシステムにはアプリケーション機能の実現と，ハードウェア制御の簡便化の 2 種類の要求があることを示した．2 種類の要求をもとに，アプリケーション層，環境構成層，制御層の 3 層構造からなるソフトウェアアーキテクチャを提案した．3 層構造にすることで，ハードウェアの変更を柔軟に対応できる．また，アプリケーション開発者にとってハードウェア制御の知識が少なくても組込みソフトウェアを開発が可能になる．

また，3 層構造に基づいたソフトウェア開発プロセスを提案した．ユーザから要求を獲得し，要求を機能ごとに木構造に分解する．木構造で表現した要求に環境構成層にあるソフトウェア部品を対応させることで，全体設計を行うこととした．

今後の課題として，アプリケーションに適応した環境構成層がない場合の環境構成層の開発方法を明らかにすることが考えられる．この時，アプリケーション層と環境構成層の分割基準が重要である．また，要求分析・環境分析でドメイン知識の少ない開発者も木構造を適切に作成し環境構成層を当てはめることが可能であるか確認することも必要である．

Vol.2010-SE-168 No.14
Vol.2010-EMB-17 No.14
2010/6/2

## 参 考 文 献

1) IPA SEC:組込みソフトウェア開発における品質向上の勧め [設計モデリング編], アイティメディア株式会社, 2006.
2) MDD ロボットチャレンジ 2009 実行委員会 : MDD ロボットチャレンジ 2009. http://sdlab.sys.wakayama-u.ac.jp/mdd2009/.
3) RICOH : RICOH & JAVA Developer Challenge.http://www.ricoh.co.jp/javachallenge/.
4) K.C kang,Jaejoon Lee, and Jaejoon Lee, and Patrick Donohoe : Feature-oriented Product Line Engineering , IEEE software, Vol.9, No.4, pp58-65, July/August 2002.