情報処理学会論文誌 Vol.60 No.8 1350–1359 (Aug. 2019)

組込みシステムのプロダクトライン開発における再利用性を考慮したシステムアーキテクチャ評価手法

大塚 敏史[1,2,a)] 中西 健二[3] 成沢 文雄[3] 土屋 達弘[2]

受付日 2018年11月5日, 採録日 2019年5月9日

概要：近年の制御システムでは，自動運転等の高度な制御を実現するため，ソフトウェアの規模が増大している．そこで開発効率向上のためにソフトウェアの再利用性を向上させ，開発するソフトウェアの量を削減することが有用である．プロダクトライン（製品系列）におけるシステムアーキテクチャを設計する際に，論理機能を複数製品で共通化設計し，それらの機能から構成される論理アーキテクチャを個々の製品の物理アーキテクチャに配置することにより，システム間での論理機能の再利用性が向上する．しかし，論理機能を実現するソフトウェアは，異なる実行環境に実装することにより修整が必要となり，再利用性への影響が発生する．そこで本研究では，システムアーキテクチャの共通化設計を行う際に，論理アーキテクチャの物理アーキテクチャへの統合先に応じた再利用性を評価可能なアーキテクチャ評価手法を提案する．本提案手法を自動運転システムの開発で適用し，多数のシステムアーキテクチャ候補を評価し，それらの中からパレート最適なものを導出することができた．これにより，本手法を用いて組込みソフトウェアの実行環境依存性を考慮した上で最適なアーキテクチャを選定可能であることを確認した．

キーワード：設計手法，組込みソフトウェア，アーキテクチャ評価，共通化，再利用性，パレート最適

A System Architecture Evaluation Method that Reflects Software Reusability in Product Line Development of Embedded Systems

Satoshi Otsuka[1,2,a)] Kenji Nakanishi[3] Fumio Narisawa[3] Tatsuhiro Tsuchiya[2]

Received: November 5, 2018, Accepted: May 9, 2019

Abstract: Recent control systems have been increasing their system and software scale in order to realize advanced control such as automated driving. Therefore, in order to enhance development efficiency, it is becoming increasingly useful to improve reusability of software and to reduce the amount of software to be developed. When designing a system architecture in a product line, by designing logical functions in common with products and synthesizing a logical architecture comprising of some of the logical functions with the physical architecture of each product, it is possible to improve reusability of logical functions in the product line. However, software that implements logical functions needs to be modified when ported to different execution environments, which affects reusability. In this paper, we propose an evaluation method that can be used to assess system architectures in product line development. The method evaluates both hardware cost and commonality of software by considering the mapping of the logical architecture to the physical architecture. We report the results of applying the proposed method to the development process of an automated driving system, where we successfully derived Pareto-optimal architectures. The results demonstrate that the method enables us to select an optimal architecture by considering the execution environment dependency of embedded software.

Keywords: designing method, embedded software, architecture evaluation, commonality, reusability, Pareto optimal

© 2019 Information Processing Society of Japan

1. はじめに

近年，社会インフラにおける制御システムは，制御の高度化や自動化を目的とし，CPS（Cyber Physical Systems）や IoT（Internet of Things）等のサイバー空間との連携や機器間制御による知能化が加速している．その中で組込みシステムはサイバー空間と現実空間の境界に位置し，センシングによる現実空間の情報収集のための演算や，現実空間に対し演算結果を実現させるアクチュエータを制御する重要な役割を担っている．

組込みシステムはそれぞれの環境において最適化されるため，多品種多量化が起こりやすい．たとえば自動車システムでは内部に 1 つまたは複数の演算装置（マイコン等）を有する ECU（Electronic Control Unit）が，1 車種で平均 20 個以上 [1] 実装されており，それぞれ役割と必要な演算性能が異なるため，異なる種別の演算装置により制御を行っている．さらに，自動車の開発においては，複数車種が 1 つのプロダクトライン（製品系列）で開発されるため，プロダクトライン全体で，組込みシステムを効率良く開発する必要がある．そのためのソフトウェア開発技術として，ソフトウェアプロダクトライン [2] があり，システムの共通部および可変部をモデル化し，プロダクトライン全体での共通化設計や差分設計を行ってコア資産の再利用を効率良く行うことにより，プロダクトライン全体の開発効率向上を行う．

また別の方法では，製品開発を効率化するためのシステムズエンジニアリング手法を適用し，たとえば IEEE1220 [3] に記載されているプロセスのように，論理構造を設計した後に，実行環境である物理構造に配置する統合を実施する．本プロセスを用いてシステムを設計し，論理構造と物理構造を分離することで，論理機能の変更や再配置を行うことが容易となる．

しかし，組込みシステムでは，ソフトウェアの実行環境（OS，マイコン等）が異なると，ソフトウェアのマイコンや OS への依存部の修整が必要となる．影響の分離や非依存化の実装方法として，リソースの仮想化技術 [4] により物理構造と論理構造を分離する取組がある．しかし，組込みソフトウェアでは限られたリソースやリアルタイム制約の中で要求された機能を実現するため，仮想化技術を利用できる場合は少なく，通常はマイコンや OS の専用機能を使用することにより，実行環境に対するソフトウェアの依存性が発生する．

従来のアーキテクチャ評価手法では，構築したシステムアーキテクチャ（論理（Functional）アーキテクチャおよび物理（Physical）アーキテクチャ）のコスト，性能，再利用性等の観点でトレードオフを評価し [5]，またそれらパラメータによりシステムアーキテクチャを最適化していたが [6]，実行環境に依存する組込みソフトウェアの再利用性を，その実行環境との組合せを考慮して判定することはできなかった．

そこで本研究では，プロダクトラインにおいて，組込みソフトウェアと実行環境の組合せにより変動するソフトウェアの再利用性を，アーキテクチャ統合時にソフトウェアと実行環境を関連させて評価を行う方法を提案する．この評価手法を用いることで，ソフトウェアの再利用性と物理アーキテクチャのコストとのトレードオフを考慮して，適切なシステムアーキテクチャの選択を実現する．さらに提案手法を，自動車の自動運転システムのプロダクトラインにおける開発事例に対し適用した結果を示す．

本論文の構成は以下のとおりである．2 章では本研究の背景となる自動車システムアーキテクチャ設計の動向と課題，3 章では関連研究，4 章で提案手法の内容，5 章と 6 章で適用事例とその考察について述べ，7 章でまとめを行う．

2. 背景

2.1 自動車システムアーキテクチャ設計の動向

自動車の E/E（Electrical and Electronic）システムは，安全性および快適性の向上や環境負荷低減を目的とした高度な制御を実現するために，その制御範囲を拡大している．特に，自動運転システムの実現に向け，E/E システムの複数の構成要素（センサ，アクチュエータ，コントロールを行う演算装置を有する ECU）の連携により，車両全体の統合制御を行うことが求められるようになっている．

従来のシステムでは，個々のセンサ，アクチュエータは個別に制御されており，それらに割り当てられている ECU 上に，制御を実行するソフトウェアを配置すれば十分であった．しかし，自動運転システム等の近年のシステムでは，複数の ECU を同時に用いて多数のセンサ，アクチュエータを連動させることが必要であり，物理ハードウェアはネットワーク化され，ECU が GW（ゲートウェイ）を介して通信可能となっている（**図 1**）．このようなネットワーク化された物理アーキテクチャにおいては，各機能を実現するソフトウェアは，自動車内の ECU のいずれにも配置される可能性がある．

そのため機能の配置を最適化するためには，システムズエンジニアリングプロセスに従い，システム全体の論理構造を明確化し，その論理構造を実現する物理構造への配置の複数案を検討し，それぞれのトレードオフを比較した上で最適な機能の配置を決定することが必要となる．

[1] 株式会社日立製作所
Hitachi, Ltd., Hitachi, Ibaraki 319–1292, Japan
[2] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[3] 日立オートモティブシステムズ株式会社
Hitachi Automotive Systems, Ltd., Hitachinaka, Ibaraki 312–8503, Japan
[a)] satoshi.otsuka.hk@hitachi.com

© 2019 Information Processing Society of Japan

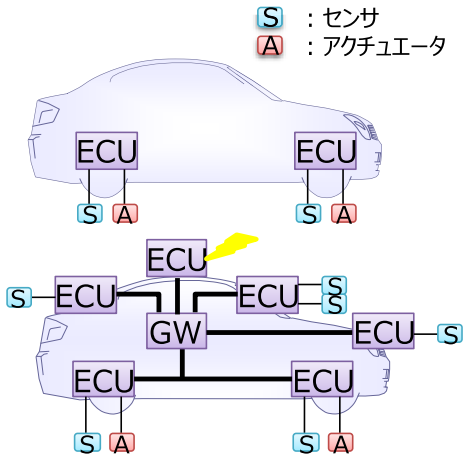


図 1 自動車システムの物理アーキテクチャの変化．（上）従来，（下）現在

Fig. 1 Transition of automotive physical architecture.

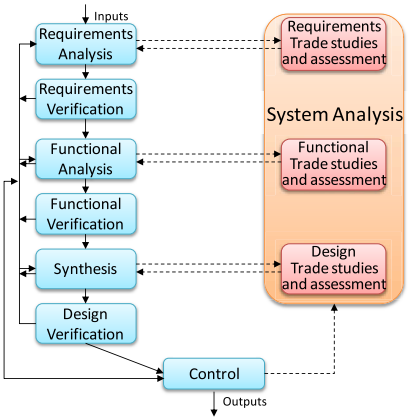


図 2 IEEE1220 システムズエンジニアリングプロセス概要

Fig. 2 IEEE1220 systems engineering process.

2.2 自動車システムアーキテクチャ設計の課題

IEEE1220 で定義されているエンジニアリングプロセスの概要を図 2 に示す．ここで Synthesis プロセスにおいては，論理アーキテクチャを物理アーキテクチャに統合する．この際に代案との比較評価を行い，システムアーキテクチャを決定する．

機能連携が複雑化する自動車システムでは，システムを構成する物理エレメント（演算装置，ネットワーク等）および論理エレメント（論理機能，データリンク等）の数，また論理エレメントを物理エレメントに対し配置するパターン（配置パターン）数が増加し，物理アーキテクチャ決定に必要な評価量が増大する．

たとえば論理エレメント数が m，物理エレメント数が n の場合，配置パターンは n の m 乗となる．具体例として，それぞれ論理および物理のエレメント数が 10 の場合，すべての配置パターン 10 の 10 乗となり，これらを手作業にて順次評価することは非常に困難である．

1 つのシステムを効率的に設計するためには，これら組合せ数が膨大な中で，コストや性能，安全性を考慮してシステムアーキテクチャを最適化する．さらにプロダクトライン全体等複数のシステムで最適化を行う際には，それぞれのシステム間でのソフトウェアの再利用によるソフトウェア開発コスト低減を考慮したうえで物理アーキテクチャを決定することが必要となる．

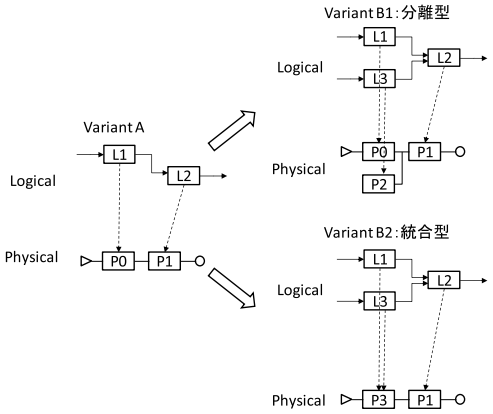


図 3 プロダクトライン開発における物理アーキテクチャのバリアント

Fig. 3 Variants of physical architecture in product-line development.

図 3 に，複数の物理アーキテクチャの候補から物理アーキテクチャを選択する例を示す．ここではプロダクトラインのバリアント（変異体）A と新たなバリアント B1 または B2 を含めて全体最適を検討する例であり，新たなバリアントではバリアント A に対し論理機能 L_3 が追加されている．

たとえばバリアント B1 は，バリアント A との共通性を重視して追加機能を分離して実現する分離型の例である．ここでは，物理エレメント P_0，P_1 と配置される論理エレメント L_1，L_2 の組合せがバリアント A と共通化されており，L_1，L_2 の実行環境がバリアント A，B1 で同一なため，ソフトウェアの再開発が不要となる．一方で L_3 を実行する物理エレメント P_2 を追加しており，ハードウェアコストが比較的高くなる．

一方でバリアント B2 は，ハードウェアコスト低減を考慮して，物理エレメントの数を変えずに追加機能を実現する統合型の例であり，論理エレメント L_1，L_3 が物理エレメント P_2 に配置されている．この場合，P_0 は P_3 に変更されており，これは増加するソフトウェアの実行のためにマイコンの上位種別への変更等が必要となるためである．結果として，論理エレメント L_1 については，バリアント A，B2 とで実行環境の差異が生じるため，実行環境に依存する部分のソフトウェアの再開発が必要になる．一方でバリアント B1 に比べ物理エレメントの個数が少なくなっており，ハードウェアコストを低減することが比較的容易である．

適切なシステムアーキテクチャの選択のためには，このようなハードウェアコストとソフトウェアの開発コスト

© 2019 Information Processing Society of Japan

のトレードオフについて，物理アーキテクチャの候補と配置パターンの組合せごとに，定量的に比較評価する必要がある．

3. 関連研究

プロダクトラインでの共通性に基づくシステムアーキテクチャの評価としては，Crawley ら [7] や Hofstetter ら [8] による研究があり，プロダクトラインでの Commonality について，単純な類似性ではなく，共通化をプロダクトラインのライフサイクル全体での共通化判断時期を含めてメリットとデメリットのトレードオフを考慮した設計を行う手法を提案している．

プロダクトラインにおけるソフトウェアアーキテクチャについて，熊木らは，プロダクトラインから，要求文書に含まれる語の類似性を用いて，コア資産（クラス）の再利用率を考慮してアーキテクチャ設計を行う手法を提案している [9]．

ソフトウェアの再利用性に関連するソフトウェアのメトリクスに関しては，国際標準である SQuaRE（System and software product Quality Requirements and Evaluation）における再利用性，ISO/IEC9126 における移植性がある．これらの定義では，たとえば，ソフトウェア全体に対する変更量の割合をもとに評価することが述べられている．

他の再利用に関するメトリクスとして，Washizaki らは，ブラックボックス型開発のコンポーネントベースのソフトウェア開発の再利用性のメトリクスを提案しており，Understandability，Adaptability，Portability を測定するための 5 つのメトリクスを用いて評価している [10]．

一方で，システムアーキテクチャを最適化するための手法としては，Kazman らの提案する ATAM（Architecture Tradeoff Analysis Method）[5] がある．この手法では，トレードオフポイントを定めステークホルダと議論を行い，再利用性を含めてシステムアーキテクチャを評価する．

システムアーキテクチャの自動最適化を実現するための手法として Adachi らは，HiP-HOPS（Hierarchically Performed Hazard Origin and Propagation Studies）を用いたシステムアーキテクチャおよび故障伝播のモデル化を行い，安全要件を検証し，要求コストと信頼度を満たすパレート最適解の導出を行う手法を提案している [12]．

他の最適解の探索手法として，Li らは，システムモデルを CSP（Constraint Satisfaction Problems）として定式化し，最適解を導出する手法を提案している．この中では各種要件（コスト，スケジューラビリティ，リソース）を制約条件として定式化しソルバにより最適解を導出している [13]．

また Doufene らは，システムズエンジニアリングプロセスの中でアーキテクチャフレームワークによる多目的最適化を実現するため，制約式を関数化し，パレートフロントを導出して最適化のための意思決定をサポートする方式を提案し，EV powertrain の自動化割合の最大化とオーナの維持コスト最小化を目的関数とした適用事例を通して，提案手法を評価している [6]．

また，論理および物理アーキテクチャ統合時の評価演算量削減において，大塚らは安全性の Fail-operational の必要要件を定式化し，制約として用いることで候補となるアーキテクチャを限定することで，物理アーキテクチャ評価の演算量を削減する手法を提案している [14]．

これら研究においては，組込みソフトにおいて重要となる，ソフトウェアの実行環境への依存性により再利用性が異なる点を考慮して物理アーキテクチャを評価する手法については述べられていない．そこで本研究では，ソフトウェアの実行環境への依存性により再利用性が異なる点を考慮した再利用性の評価式と，前記評価式を用いてパレート最適解を用いて物理アーキテクチャをトレードオフ評価して最適化を可能とする手法について提案する．

4. 提案手法

4.1 システムアーキテクチャ評価の全体像

システムアーキテクチャ評価の全体像を図 **4** に示す．この例では，バリアント A と共通のプロダクトラインであるバリアント B のシステムアーキテクチャを最適化する例について示している．

バリアント A とバリアント B は共通のプロダクトラインであり，共通のフィーチャーツリーから可変性を選択することにより，それぞれのバリアントのフィーチャーを決定する．

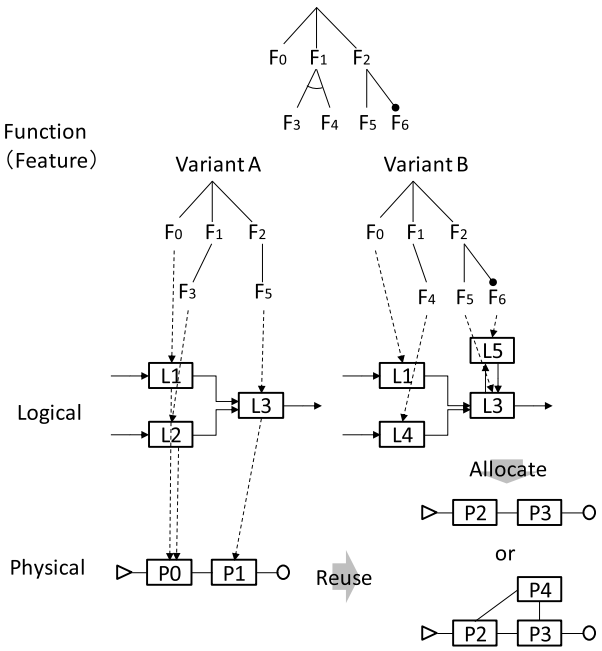


図 4 システムアーキテクチャ評価の全体像
Fig. 4 Summary of system architecture evaluation.

© 2019 Information Processing Society of Japan

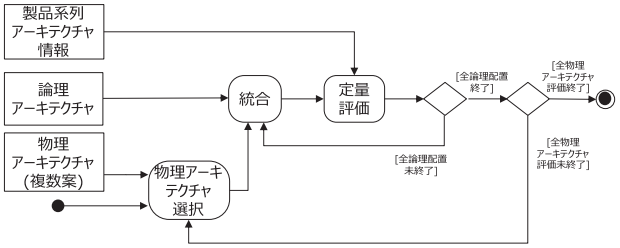


図 5 アーキテクチャ評価手順
Fig. 5 Process of architecture evaluation.

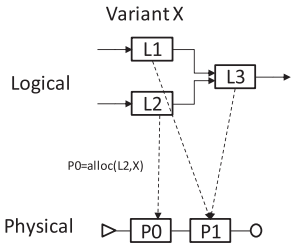


図 6 統合の例
Fig. 6 Example of synthesis.

次に，それぞれ選択したフィーチャーを実現するための論理アーキテクチャを設計し，さらにそれを実行する物理アーキテクチャを設計する．ここではバリアント A の設計は完了しており，その設計結果を再利用してバリアント B を設計する例を示している．

バリアント B の実現には，多くの物理アーキテクチャの候補と，さらにそれぞれの物理アーキテクチャに対する論理アーキテクチャの配置パターンがある．そのため後述する評価手法を用いて最適な物理アーキテクチャと配置パターンの組合せを選択する必要がある．

システムアーキテクチャ評価の手順を**図 5** のアクティビティ図を用いて説明する．最初の物理アーキテクチャ選択では，システム設計者が構築した複数の物理アーキテクチャ案から 1 つの候補を選択する．次の統合においては，論理アーキテクチャの物理アーキテクチャへの統合について，配置パターンを 1 つ選択する．その後定量評価において，プロダクトラインのシステムアーキテクチャ情報を基に評価を行う．これらを順次繰り返し，候補となるすべてのシステムアーキテクチャについて定量的に評価を行う．

物理アーキテクチャについては，システム設計者がコストや演算性能の制約，過去モデルとの共通性等を考慮して，現実的に実現可能な複数の候補を作成する．それらの物理アーキテクチャの候補と，論理エレメントの配置パターンの組合せをすべて網羅して評価することを本評価手法の狙いとしている．

数個の物理案であっても，論理エレメントの配置パターンは 2.2 節で説明のとおり非常に多く存在し，それらのトレードオフを定量的に評価することが重要となる．

4.2 物理アーキテクチャ選択

物理アーキテクチャ選択においては，複数の候補の中から現在の評価対象となる物理アーキテクチャを選択する．すべての物理アーキテクチャの評価が終了するまで，順次物理アーキテクチャを選択する．

4.3 統合

統合においては，論理アーキテクチャのエレメントを物理アーキテクチャに配置する．**図 6** に統合の例を示す．ここではバリアント X の論理アーキテクチャの各エレメントが物理アーキテクチャに配置されている例を示しており，たとえば論理エレメントの L_2 は物理エレメントの P_0 に配置されている．

このようにしてアーキテクチャの統合を行う．また統合においては，複数の配置パターンがある場合（たとえば図 6 の例で，L_1 の配置先を P_0 と P_1 で比較する場合），順次パターンを変更する．

4.4 定量評価

ここでは，物理アーキテクチャに論理アーキテクチャを配置して得られるシステムアーキテクチャを，物理コストとソフトウェアの再利用性の両面から評価する手法を提案する．

評価の対象となるシステムアーキテクチャの物理アーキテクチャは，下記のとおり物理エレメント P_i の集合で表す．

$$\Pi = \{P_1, P_2, \ldots, P_m\}$$

物理エレメント P_i は，属性としてコスト $\mathrm{cost}(P_i)$ を持つ．

また，論理アーキテクチャも同様に論理エレメント L_i の集合で表す．

$$\Lambda = \{L_1, L_2, \ldots, L_n\}$$

論理エレメント L_i は，規模 $sizeof(L_i)$ を属性として持つ．ここで，配置される物理エレメントが変わっても，規模は影響を受けないと仮定する．

統合によって得られる論理エレメントの物理エレメントへの配置は，論理アーキテクチャから物理アーキテクチャへの写像 $Alloc : \Lambda \to \Pi$ として表すことができる．つまり，論理エレメント L_i が物理エレメント P_j に配置されていることは，以下のように表される．

$$P_j = Alloc(L_i)$$

バリアントのシステムアーキテクチャを，論理アーキテクチャ Λ，物理アーキテクチャ Π，論理アーキテクチャから物理アーキテクチャへの写像 $Alloc : \Lambda \to \Pi$ として定義する．

© 2019 Information Processing Society of Japan

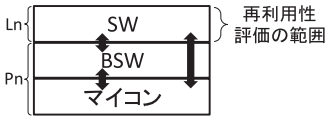


図 7 再利用性評価の範囲
Fig. 7 Focus of reusability evaluation.

$V = (\Lambda, \Pi, Alloc)$

以下，バリアント $V = (\Lambda, \Pi, Alloc)$ を，物理コストとソフトウェアの再利用性から評価する方法を示す．

(a) **物理コスト評価**

物理コストの判定では，評価対象の物理アーキテクチャのエレメントすべてのハードウェアのコストを合算する．評価対象の物理アーキテクチャ Π に対し，物理コストを以下のように定める．

$$\text{物理コスト} = \sum_{Pi \in \Pi} \text{cost}(P_i)$$

(b) **再利用性評価**

再利用性については，論理機能を実現するソフトウェアが再利用できる割合いについて評価を行う．前述のとおり，組込みシステムのソフトウェアは実行環境（BSW（OS 等の基本ソフトウェア），マイコン）に依存するため，バリアント間での実行環境に依存した部分の変更による修整を考慮し，再利用性を判定する（**図 7**）．

物理エレメント P_i, P_j について，再利用性 $Reuse(P_i, P_j)$ $(0 \leq Reuse(P_i, P_j) \leq 1)$ が定まることを仮定する．これは，任意の論理エレメント L_k が P_i へ配置できるように設計されているときに，L_k を P_j に配置できるようにする際に修整不要な再利用できる元の L_k の割合いを示す．これは物理エレメント P_i, P_j のマイコン，および，BSW の類似の度合いをもとに算出する．

L_i が P_j 上で開発済みのとき，それを利用して L_k を P_j 上で開発する際に再利用できるソフトウェアの量を以下のように定める．

$$sizeof(L_i) * Reuse(P_i, P_j)$$

また，ソフトウェア全体の再利用性（Reusability）を以下のように定める．

$$\text{Reusability} = \frac{\text{修整不要となるソフトウェアの規模}}{\text{ソフトウェア全体の規模}}$$

ここで，ソフトウェア全体の規模は，評価対象のバリアントの論理アーキテクチャを Λ とし，下記のとおり定める．

$$\text{ソフトウエア全体の規模} = \sum_{Li \in \Lambda} sizeof(L_i)$$

修整不要となるソフトウェアの規模は，下記のとおり定める．

修整不要となるソフトウエアの規模

$$= \sum_{Li \in \Lambda} sizeof(L_i) * MaxReuse(L_i, Alloc(L_i))$$

$MaxReuse(L_i, P_j)$ は，論理エレメント L_i を物理エレメント P_j に配置できるように設計する際，すでに開発されている全バリアントを利用できる前提で得られる最大の再利用性である．開発済みの全バリアントの集合を M とすると，以下のように定まる．

$$MaxReuse(L_i, P_j) = \max_{(\Lambda,\Pi,Alloc) \in M \ s.t. \ Li \in \Lambda} \{Reuse(Alloc(L_i), P_j)\}$$

本節の例においては，M には，バリアント B の再利用元であるバリアント A が含まれる．

4.5 アーキテクチャ評価結果の比較

4.4 節で提案した評価手法を用いることで，複数の物理アーキテクチャの候補について，論理エレメントの配置パターンについて必要な制約を満たす複数の組合せで評価を実施し，その結果に基づいて，現在開発対象としているバリアントのシステムアーキテクチャを決定する．具体的には，物理コストとソフトウェアの再利用性の両面に基づいた決定を行うために，それら両者を目的関数とし，評価したシステムアーキテクチャ全体を実行可能解の集合とした多目的最適化問題を考える．このとき，パレート最適解の集合であるパレートフロントに属するものを，バリアントとして選択するアーキテクチャの候補とする．それらの候補から，ステークホルダと合意を取りアーキテクチャを決定する．

以上の手順により，ソフトウェアの実行環境に依存する再利用性を考慮したうえで，システムアーキテクチャの物理コストとソフトウェア再利用性とのトレードオフ評価に基づくアーキテクチャの決定が可能となる．

5. 適用事例

5.1 実験

提案手法を，自動運転システムのプロダクトライン開発におけるシステムアーキテクチャ評価に適用した．

本開発では，2 章で例を示した統合と分離のトレードオフの比較を行った．これは機能進化を続ける自動運転制御 ECU に対し，他の ECU を統合することでハードウェアコストを削減するか，現行製品である再利用元との共通化により開発コストを削減するか，そのトレードオフを定量的に比較するためである．

本開発において，自動運転制御 ECU に対して統合の対象となる ECU は 2 つあり，そのそれぞれを自動運転制御 ECU に統合するか否かの 4 つの物理アーキテクチャで比較を行った．

© 2019 Information Processing Society of Japan

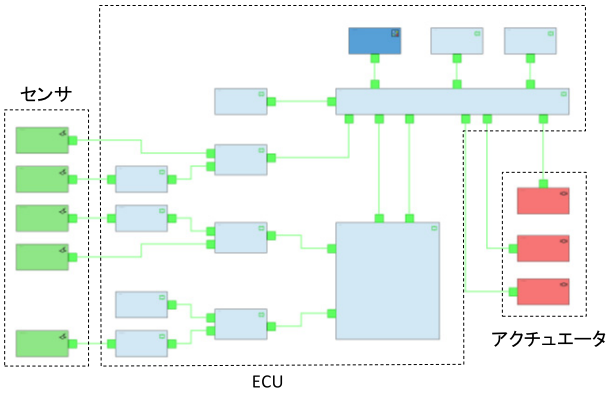


図 8 再利用元の物理アーキテクチャ
Fig. 8 Physical architecture of original variant.

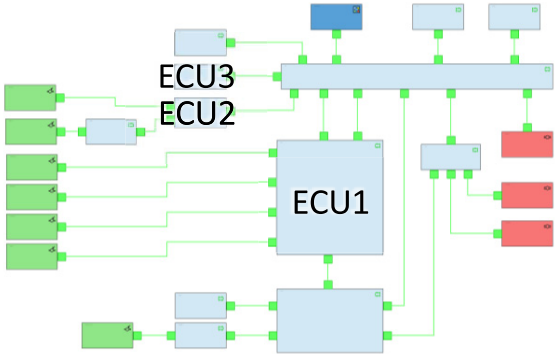


図 9 候補となる物理アーキテクチャの一例
Fig. 9 Candidate for physical architecture.

図 8 はプロダクトライン開発の再利用元となる物理アーキテクチャであり，センサ，アクチュエータ，ECU を含む．論理エレメントはこれら ECU が有する演算装置の実行環境に配置される．

図 9 がプロダクトライン開発で候補となる物理アーキテクチャの一例である．記載している ECU2，ECU3 が統合の対象であり，この物理アーキテクチャではいずれも統合は行われていない．この物理アーキテクチャに加えて，ECU2 を ECU1 に統合，ECU3 を ECU1 に統合，ECU2 と ECU3 を ECU1 に統合，の 4 つの物理アーキテクチャについて評価を行った．

図 10 はこのプロダクトラインの基となるフィーチャーツリーに対応する論理アーキテクチャである．各バリアントの論理エレメントはすべてこのアーキテクチャに含まれている．バリアントを開発する場合は，必要な論理エレメントのみを含む論理アーキテクチャを抽出して用いる．

本開発では，論理機能として，特定の走行条件における認識・判断・制御機能，および特定の認識機能，が ECU1，2，3 の統合に主に関連する機能であり，新たなバリアントにはこれらの機能が追加されている．これら論理機能を含め，該当するバリアントで使用するすべての論理エレメントを物理エレメントに配置し評価を実施した．また，ECU1 には過去に開発のない新規マイコンが搭載される．

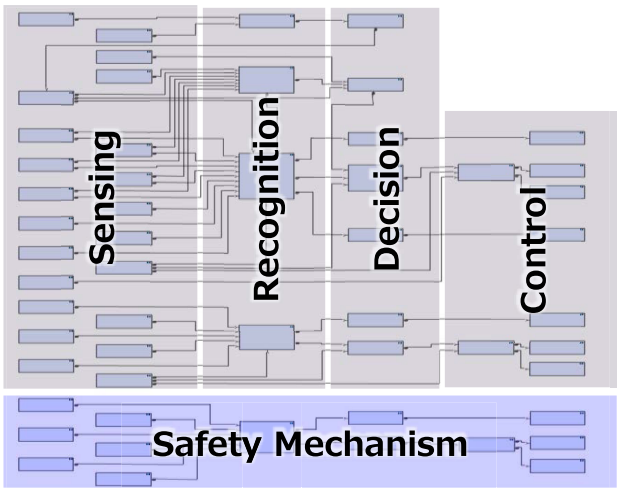


図 10 論理アーキテクチャ
Fig. 10 Logical architecture.

論理アーキテクチャと 4 つの物理アーキテクチャに対し，それぞれの論理エレメントについてはソフトウェアの開発量を，それぞれの物理エレメントについてはマイコンや BSW が共通か否かを判定するための種別，およびハードウェアのコストを割り当てて評価を行った．また，論理アーキテクチャから物理アーキテクチャへの可能な割当てを選別するための定量評価および安全性評価 [14] に用いるため，論理エレメントに演算量/通信量/要求する安全レベル等を，物理エレメントに演算性能/通信帯域/実現可能な安全レベル等を付与した．

ソフトウェアの再利用性については，今回は当該ドメインでの過去開発での経験にならい，マイコンまたは BSW に変更があった場合には一律で 75%を再利用可能，双方に変更がある場合には 50%を再利用可能，変更がない場合は 100%を再利用可能，新規に作成となる論理エレメントは再利用性 0%として評価を行った．

また論理アーキテクチャの物理アーキテクチャへの配置については，システムアーキテクチャ評価手法 [14] を用いた．これにより，すべての配置パターンの中で性能要件の制約（アクチュエータやセンサとの機電一体の機能配置），および安全要件（Fail-operational の必要条件）を充足する 2,500 通り（4 つの物理アーキテクチャ候補での合計）に絞り込み，評価を行った．

5.2 再利用率に関する演算過程

ここで再利用性の計算式の例について説明する．再利用元のシステムで，物理エレメント P_i（たとえば ECU2 のマイコン A，BSW1）に配置されていた論理エレメント L_k（$sizeof(L_k) = 10\,[\mathrm{kLoC}]$）を例にあげる．

この論理エレメント L_k を，ECU2 と ECU1 を統合後，別の物理エレメント P_j（たとえばマイコン B，BSW2）に統合した場合，L_k の再利用率については，

© 2019 Information Processing Society of Japan

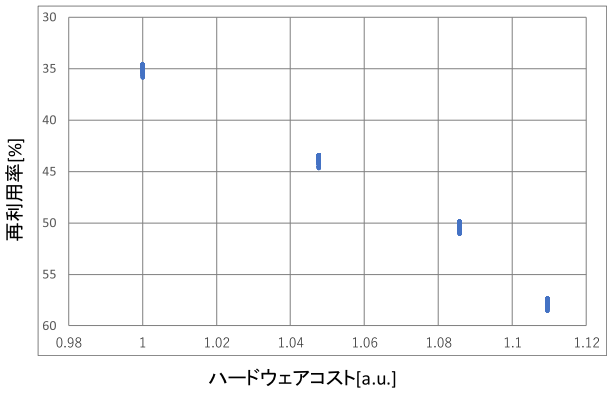


図 11 評価結果
Fig. 11 Evaluation results.

$$sizeof(L_k) * MaxReuse(L_k, P_j) = 10 * 0.5 = 5\,[\mathrm{kLoC}]$$

となる．また ECU2 を統合せず，統合前と同じ物理エレメント P_i に配置した場合には

$$sizeof(L_k) * MaxReuse(L_k, P_i) = 10 * 1.0 = 10\,[\mathrm{kLoC}]$$

となる．マイコンまたは BSW のいずれかが異なる場合には，$MaxReuse(L_k, P_k) = 0.75$ として演算を行う．これら再利用率をすべての論理エレメント L_k に関して演算し，全体の再利用率を導出した．

5.3 結果

評価結果について**図 11** に示す．ここでは横軸がハードウェアのコスト，縦軸が再利用率となっている．左下を最適点とするため，再利用率を上下逆とし，下の点がより再利用率が高いように記載している．ハードウェアコストについては，ここでは 2 つの ECU をすべて統合した物理アーキテクチャのハードウェアコストを 1 として正規化した数値で表現している．

図が示すとおり，今回の例では，ハードウェアコストと再利用性がトレードオフの関係となった．

具体的には，左から，ECU すべてを統合した物理アーキテクチャ，ECU2 のみを統合した物理アーキテクチャ，ECU3 のみを統合した物理アーキテクチャ，統合を行わなかった物理アーキテクチャに対応している．それぞれの物理アーキテクチャに対する点で，最も再利用率の高いものが，パレート最適なシステムアーキテクチャとなっており，このような 4 点がパレートフロントを構成している．

また，ハードウェアコストが同じ複数の点は，物理アーキテクチャが同じで，論理エレメントが異なる物理エレメントに配置され，再利用性が変化する例を示している（**図 12**）．

5.4 結果に対する分析と考察

図 11 に示すとおり，物理アーキテクチャの候補について，既存製品からの再利用開発において，組込みソフトウェアの再利用性を考慮したパレート最適を導出可能となった．ソフトウェア再利用性の数値は，今回の評価では概算見積り（0%/50%/75%/100%）を基にしており，数値を実際の値と比べると誤差が生じるが，それぞれの実行環境依存性を同条件で比較しているため，再利用性の数値が相対的に高いということは，いずれかの論理エレメントをより再利用性が高い実行環境に配置することを示している（たとえば，再利用前と再利用後でマイコンまたは BSW に変更がある実行環境に配置し 75%再利用と判定した論理エレメントを，マイコンおよび BSW が再利用前と再利用後で同じとなる 100%再利用が可能な実行環境に配置する，等）．ハードウェアコストが同じ条件において，再利用性が向上する論理エレメントの物理エレメントへの配置を求めることが可能になると考える．ここから，たとえばハードウェアコストやソフトウェア開発コストの優先順位を考慮し，物理アーキテクチャを選択可能となる．

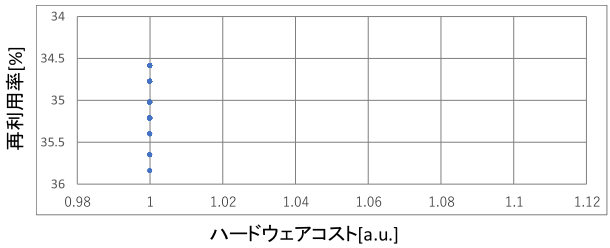


図 12 評価結果（一部拡大）
Fig. 12 Evaluation results (part of the graph).

従来手法では，ハードウェアコストのみの評価を実施した場合，個別の製品でコスト最適となり，製品系列全体での開発コストが増大する可能性がある [6], [13].

一方，再利用性のみを評価した場合，共通化を目的として製品系列全体で同じコントローラ・マイコンを選定し，結果として同様に製品系列全体でのハードウェアコストが大きくなる．文献 [7], [8] ではコストと再利用性を考慮しているが，本研究で課題としている組込みソフトウェアの実行環境依存性を考慮していないため，結果として論理エレメントの再利用が困難となる可能性がある．本提案手法ではハードウェアコストと実行環境依存性を考慮したソフトウェアの再利用性を同時に評価可能となる．

特に統合における配置案は網羅的に評価を行い，再利用性の最適点（ハードウェアコストが同一で，最も再利用性が高い点）を導出可能となった．安全性評価手法 [14] と組み合わせることで，数が大きい配置パターンから，安全性の要件を満たす妥当な組合せで，上記トレードオフと最適な配置を網羅的に評価可能となった．

また本プロジェクトにおいては，再利用率から換算したソフトウェアの開発工数がプロジェクトの許容範囲内であったため，ハードウェアコストを優先し，統合型の物理アーキテクチャを選択した．ここでたとえば再利用率から

© 2019 Information Processing Society of Japan

換算されるソフトウェアの開発工数がプロジェクトとして許容できない，またはハードウェアコストが製品計画上許容できない場合はそれぞれの値で境界を設定し，許容範囲内でのトレードオフ選択による最適化を行うことが可能となると考える．

本節で扱った製品は開発途中であり，提案手法により導出した開発工数と，実際の開発工数との比較が可能な段階には至っていない．しかし，個別の ECU における再利用率については，当該ドメインのエキスパートによるレビューで，おおむねの見積りについては妥当と評価されている．全体の再利用率は個々の ECU について同じ方法で見積りを行った結果の積み重ねであり，同様に妥当であると判断している．

また今回の開発ではプロダクトライン開発を前提とし，特定のバリアントのコア資産を再利用して異なるバリアントの開発を行う例を示したが，バリアント管理を前提としない派生開発においても同様に本手法は適用可能である．その場合は，派生開発元と派生開発先の論理エレメントや物理エレメントの差分や配置について，同様に実行環境依存性に基づき評価を行うことにより，複数の物理アーキテクチャからパレート最適となるアーキテクチャを導出することが可能である．

6. 限界と妥当性への脅威

(a) 内的妥当性への脅威

実行環境への依存性の数値について，今回は経験則を基に一律試算を行った．同種のマイコンの系列間の移植には比較的コストがかからない等があるため，パラメータとして反映することにより，そのような条件を加味した評価が可能となると考える．

(b) 外的妥当性への脅威

IT 系等，実行環境とソフトウェアの分離ができている分野においては，上記実行環境への依存性は考慮不要と考える．組込み分野においても，仮想化技術 [4] の進展により適用が不要になる可能性はあるが，1 章で述べたとおり，組込みシステムはリアルタイム性等の観点で実行環境への依存性が高いため，そのような制約の環境で有用であると考える．

7. おわりに

本研究では，組込みシステムのプロダクトライン開発において，定量的な評価に基づいて適切にシステムアーキテクチャを選択する方法を提案した．まず，論理アーキテクチャを物理アーキテクチャへ統合して得られるシステムアーキテクチャの定量的評価を，組込みソフトウェアの実行環境への依存性を考慮してその再利用性を見積もったうえで，ハードウェアコストとソフトウェア開発量とのトレードオフを数値化することで実現した．この評価手法を，候補となるシステムアーキテクチャを安全性等の制約を考慮した上で絞り込む既存手法と統合することで，最適なアーキテクチャを選択する方法を提案した．さらに，自動運転用システムの開発への提案手法の適用事例を通してその有効性を示した．

今後の課題としては，再利用率の本手法による予測値と実開発での値の比較評価，再利用性以外の評価指標の統合や，要求を最適化する物理アーキテクチャ候補の生成方法等があげられる．

参考文献

[1] 富士キメラ総研：車載電装デバイス&コンポーネンツ総調査 2017 下巻：ECU 関連デバイス編，富士キメラ総研 (2017).

[2] Pohl, K., Günter, B. and van der Linden, F.: *Software product line engineering: Foundations, principles and techniques*, Springer Science & Business Media (2005).

[3] IEEE Std 1220-2005 Systems engineering - Application and management of the systems engineering process (2005).

[4] Sandström, K. et al.: Virtualization technologies in embedded real-time systems, *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation* (*ETFA*), IEEE (2013).

[5] Kazman, R., Klein, M. and Clements, P.: ATAM: Method for architecture evaluation, No. CMU/SEI-2000-TR-004, Carnegie-Mellon University Pittsburgh PA Software Engineering Institute (2000).

[6] Doufene, A., Chalé-Góngora, H.G. and Krob, D.: Complex systems architecture framework: Extension to multi-objective optimization, *Complex Systems Design & Management*, Springer, Berlin, Heidelberg, pp.105–123 (2013).

[7] Crawley, E. et al.: A commonality toolkit focused on NASA applications (2010).

[8] Hofstetter, W.K.: A framework for the architecting of aerospace systems portfolios with commonality, Diss. Massachusetts Institute of Technology (2009).

[9] 熊木健太郎，鷲崎弘宜，深澤良彰：要求・設計資産からのプロダクトラインアーキテクチャ抽出，日本ソフトウェア科学会大会論文集，Vol.28, pp.1–10 (2011).

[10] Washizaki, H., Yamamoto, H. and Fukuzawa, Y.: A metrics suite for measuring reusability of software components, *Proc. 9th International Software Metrics Symposium*, IEEE (2003).

[11] Zubrow, D. and Chastek, G.: *Measures for software product lines*, No. CMU/SEI-2003-TN-031, Carnegie-Mellon University Pittsburgh PA Software Engineering Institute (2003).

[12] Adachi, M. et al.: An approach to optimization of fault tolerant architectures using HiP-HOPS, *Software: Practice and Experience*, Vol.41, No.11, pp.1303–1327 (2011).

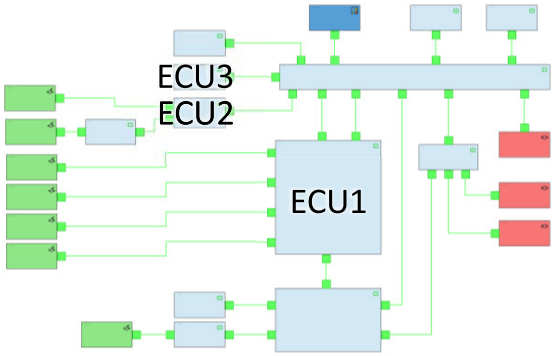
[13] Li, S. and Hemani, A.: Three-dimensional design space exploration for system level synthesis, *Proc. 2014 17th Euromicro Conference on Digital System Design* (*DSD*), IEEE (2014).

[14] 大塚敏史，中西健二，櫻井康平：安全性評価に基づき演算量を削減する Fail-operational E/E アーキテクチャ評価手法，*SEC Journal*, Vol.13, No.4, pp.10–17 (2018).

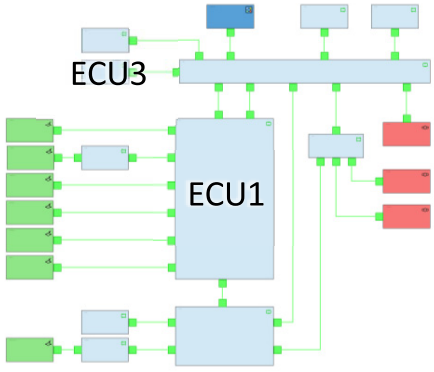
© 2019 Information Processing Society of Japan

付　　録

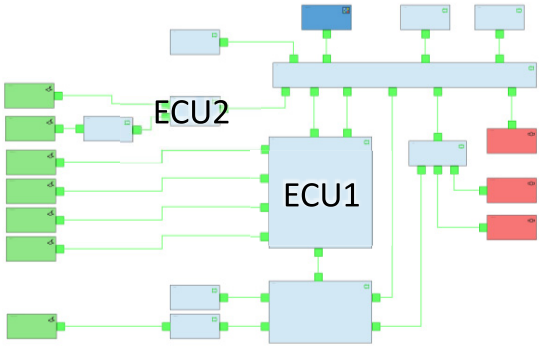
A.1 物理アーキテクチャの候補



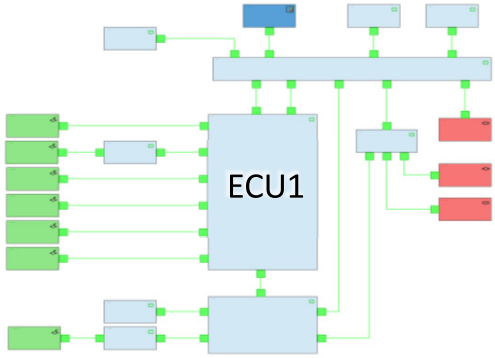
物理アーキテクチャ1：分離型



物理アーキテクチャ2：ECU2統合



物理アーキテクチャ3：ECU3統合



物理アーキテクチャ4：ECU2およびECU3統合

大塚 敏史

2000 年新潟大学工学部電気電子工学科卒業．2002 年同大学大学院自然科学研究科博士前期課程修了．同年（株）日立製作所入社．2018 年大阪大学大学院情報科学研究科博士後期課程入学．2012 年より日立製作所日立研究所で自動車 E/E システムに関する研究開発に従事．

中西 健二

1980 年生．2002 年茨城大学工学部メディア通信工学科卒業．2014 年より日立オートモティブシステムズ（株）．自動運転システムの開発に従事．

成沢 文雄　（正会員）

1970 年生．1994 年東北大学工学部情報工学科卒業．1996 年同大学大学院情報科学研究科情報基礎科学専攻修士課程修了．同年（株）日立製作所日立研究所入社．2016 年より日立オートモティブシステムズ（株）．自動運転システムの開発に従事．自動車技術会会員．

土屋 達弘　（正会員）

1995 年大阪大学大学院基礎工学研究科博士前期課程修了．1996 年同大助手．同大講師，助教授等を経て，2012 年より大阪大学大学院情報科学研究科教授．博士（工学）．ディペンダブルシステム，モデル検査，ソフトウェアテスト等に関する研究に従事．電子情報通信学会，ソフトウェア科学会，IEEE，ACM 各会員．

© 2019 Information Processing Society of Japan