

DFEAM: 動的フィーチャ指向消費電力適応型モデリング

田中 文也^{1,a)} 久住 憲嗣^{2,3,b)} 福田 晃^{2,3,c)}

概要: 組み込みシステムには電力制限と多機能化による稼働時間短縮の背景から消費電力削減の要求がある。一方で消費電力と QoS(Quality of Service) にはトレードオフの関係があり、組み込みシステム開発ではこの両立が求められる。本論文では、ソフトウェアの消費電力と QoS の両立を目的とした自己適応型ソフトウェアをモデル駆動開発によって開発するモデリング手法 (DFEAM; Dynamic Feature-oriented Energy-aware Adaptive Modeling) を提案する。本手法ではアプリケーションの変異性を記述したフィーチャモデルと xtUML による振る舞いの記述を関連づけることで、消費電力状況に応じた振る舞いをアプリケーション自身が決定することが可能である。また、可変点の数に応じて複雑化するバリエーションの QoS について、QoS 値を定量化するモデルを作成し比較の指標として最適バリエーションの探索に使用する。複数の可変点を持つアプリケーションを用いてケーススタディを行い、GQM モデルを用いて評価した。評価の結果、電力制限下で可能な限り最大のソフトウェア品質を提供するように適切な適応がなされ、DFEAM の有用性であることが示された。

キーワード: 組み込みシステム, 自己適応ソフトウェア, モデル駆動開発, フィーチャ指向, xtUML

DFEAM: Dynamic Feature-oriented Energy-aware Adaptive Modeling

TANAKA FUMIYA^{1,a)} HISAZUMI KENJI^{2,3,b)} FUKUDA AKIRA^{2,3,c)}

Abstract: There is an increasing demand for reducing the power consumption in the field of embedded-system development. A development methodology, which can change software's power consumption according to the power consumption of the hardware, can help fulfill this requirement. However, there will be a trade-off between the power consumption and service quality, which must be balanced for efficient operation. In this paper, we propose dynamic feature-oriented energy-aware adaptive modeling (DFEAM), which develops self-adaptive software through model-driven development for achieving a proper balance between the power consumption and quality of service (QoS). In this method, the application itself decides its behavior, according to the power-consumption situation, by linking the feature model describing the variability of the application with the description of its behavior, using the executable and translatable unified modeling language (xtUML). For achieving a satisfactory QoS for variations that are complex and dependent on variable points, a model is created to quantify the QoS values, which is then used as an index of comparison for finding the optimum variation. We conducted case studies on applications with multiple variable points, and evaluated them using the GQM model. The results of the evaluation showed that the adaptation incorporated provided the maximum software quality under the given power limitations, thus verifying the usefulness of the proposed DFEAM method.

Keywords: Embedded System, Self-adaptive Software, Model-Driven Development, Feature-oriented, xtUML

1. はじめに

組み込みシステムはハードウェアの大きさや製造コストに制限がある。バッテリー駆動の場合、バッテリーサイズ

¹ 九州大学大学院システム情報科学府
Graduate School and Faculty of Information Science and
Electrical Engineering, Kyushu University, Motooka 774,
Nishi-ku, Fukuoka 819-0395, Japan
² 九州大学システム LSI 研究センター
System LSI Research Center, Kyushu University, Motooka
774, Nishi-ku, Fukuoka 819-0395, Japan
³ 九州大学大学院システム情報科学研究院
Graduate School and Faculty of Information Science and
Electrical Engineering, Kyushu University, Motooka 774,

Nishi-ku, Fukuoka 819-0395, Japan
a) tanaka@f.ait.kyushu-u.ac.jp
b) nel@slrc.kyushu-u.ac.jp
c) fukuda@f.ait.kyushu-u.ac.jp

が制限されると蓄電容量が小さくなるため稼働時間が短くなる。また、機能の多様化にともなうタスク実行シナリオの多様化により組み込みシステムの使用頻度は上昇しており、稼働時間の短縮が問題である。これらの背景からユーザが要求する稼働時間を実現するためにソフトウェアの消費電力を削減する必要がある、消費電力制限が存在する。一方で、消費電力と QoS (Quality of Service) にはトレードオフが存在する。一般に高品質な機能を持つソフトウェアは消費電力も大きくなり、逆に消費電力削減には品質低下が伴う。組み込みソフトウェア開発ではソフトウェアの消費電力削減と QoS 向上の両立が要求される。

ソフトウェア工学の手法に自己適応型ソフトウェア [8], [9], [10] が存在する。これは周囲の状況に応じて振る舞いを変えることができるソフトウェアで、複雑かつ動的化するソフトウェアを扱う効果的な手法として広く認識されている。組み込みソフトウェアは設計時に想定されていないシナリオで実行された場合に組み込みシステムの動作を保証することができないため、実行時のシナリオに応じたタスク実行の機能が期待される。実行時のシナリオは常に変化する。そのため状況に応じた最適な振る舞いは設計時には記述することが難しく、変化する実行時の状況に応じてソフトウェアは最適な振る舞いをするように動的に判断を下す必要がある。

本研究では消費電力削減と QoS 維持を両立する自己適応型ソフトウェアをモデル駆動開発によって実現するモデリング手法 DFEAM (Dynamic Feature-oriented Energy-aware Adaptive Modeling) を提案する。提案手法は 2 つの要素に分けることができる。1 つは SPLE (Software Product Line Engineering) におけるフィーチャモデルと xtUML (Executable and Translatable UML) の関連づけによる、消費電力に関して適応する実行可能モデルの作成である。設計時に可変フィーチャモデルと関連づけた状態マシン図を作成し、その上で消費電力の推定値情報を付加することで、実行時の消費電力に応じた機能構成を実現する。もう 1 つは、最適バリエーションの探索である。最適バリエーションを決定するためにバリエーションごとの推定消費電力と QoS を比較するためのモデルを作成する。ここで、バリエーションとは各可変フィーチャをバインドすることによって得られるフィーチャ構成である。QoS モデルはフィーチャモデルに QoS に関する情報を追加して作成し、モデルフィッティングによって作成した消費電力モデルと併用しシンプルな方法によって最適バリエーションを決定する。本手法はすでに提案されている消費電力自己適応型ソフトウェアの開発方法論 [1] をベースとし、適応コンセプトを QoS について考慮できるように拡張したものである。ケーススタディでは可変点を複数持つアプリケーションを提案手法を適用して開発する。GQM モデルを用いて DFEAM によって適切な適応が実現可能かを評価し、

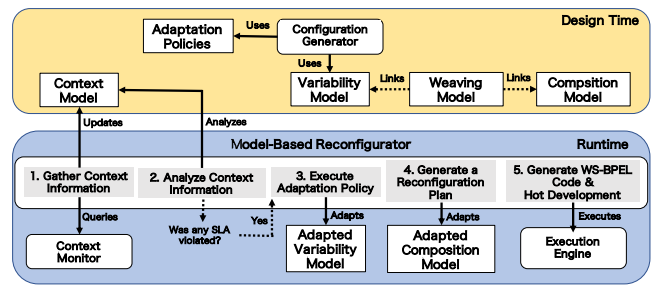


図 1 MoRE-WS の全体像

Fig. 1 An overview of MoRE-WS

xtUML を使用した自己適応アプローチの有用性を示す。

以下に本論文の構成を述べる。モデルを用いた自己適応についての関連研究について第 2 章で述べる。第 3 章では提案手法についてメタモデルを用いて説明する。ケーススタディの内容とその評価について第 4 章で説明し、最後に本研究の概要を第 5 章にまとめる。

2. 関連研究

モデルを用いた自己適応システム開発方法の関連研究を挙げる。

文献 [2] では可変モデルを含む複数のモデルを組み合わせて Web サービスの動的適応を行うフレームワーク MoRE-WS が提案されている。フレームワークは設計時の適応モデルのモデリングのサポートと、実行時の適応実行の両方で利用される。図 1 に MoRE-WS の設計時のサポートと実行時の適応フローを含む全体像を示す。図 1 上部は設計時のフレームワークのサポートを示す。MoRE-WS は設計時に可変性を含むフィーチャモデルおよび Web サービス構成モデルのモデリングや適応規則の生成をサポートする。可変フィーチャモデルは Web サービス構成モデルにマッピングされておりフィーチャモデルの構成に応じてサービス構成が変更できる。実行時には設計時にモデリングしたモデルを適応規則や Context Monitor が検知したコンテキストの変化に応じて適応を行う。コンテキストには Web サービス操作のプロパティが扱われている。図 1 下部は適応フローを示す。コンテキストの変化に応じて可変フィーチャモデルの再構成案を生成し WS-BPEL のコードに反映することで適応を実現する。

モデルを扱う適応の中でも、特に UML (Unified Modeling Language) を用いた適応についての研究がなされている。文献 [7] ではユースケースを拡張した適応型システム向けのドメイン特化モデリング Adapt Case を提案している。Adapt Case は適応性を明示的に示すために UML のユースケースを使用し、設計時の段階で適応性を集約することができる。設計の早い段階で適応性を認識することで要求分析と技術設計の間のギャップの解消に有用である。この手法では、要求分析、論理設計、技術設計の 3 ステップでの開発が想定されている。要求分析ではソフトウェア

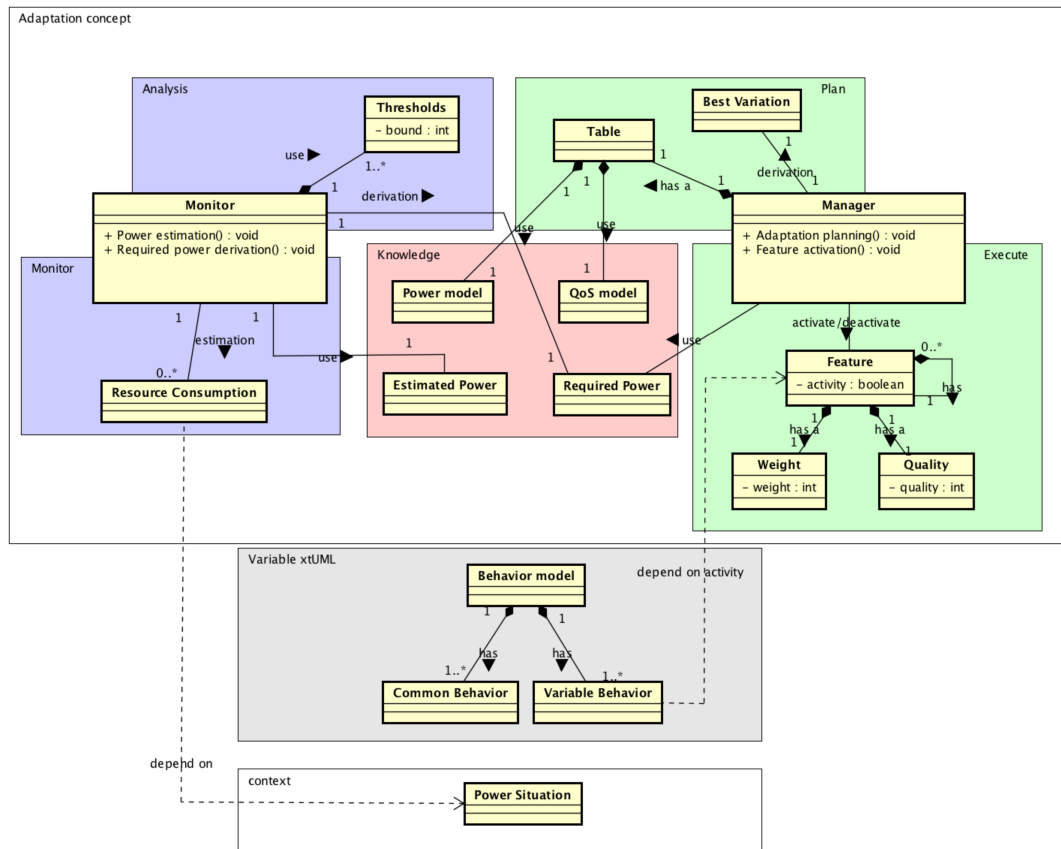


図 2 DFEAM のシステム構成
Fig. 2 System composition of DFEAM

のゴールや要求を分析する。論理設計時にそれらの要求をより具体的なユースケースに落とし込む。標準的なアプリケーションロジックはユースケースに記述され、適応を必要とするイベントやエラーに対するイベントは Adapt Case に記述する。さらに、技術設計時に Adapt Case をソフトウェアコンポーネントに適用する。コンポーネントへの適用は OCL (Object Constraint Language) に Adapt Case を変換することで実現する。ユースケースの拡張は UML プロファイルによって実装される。UML プロファイルを使用することでモデル駆動開発のアプローチへの容易な統合を保証することができる。

ここまで挙げた関連研究では、MoRE-WS では Web サービスの性能に関して適応を行い、Adapt Case では一般的な仕様言語である UML を用いることで採用しやすい適応コンセプトが提案されている。どちらもモデルベースの自己適応の例であるが、組み込みシステム開発における課題である電力制限と品質のトレードオフを解決する自己適応方法論はまだ確立されていない。そこで、我々は消費電力とサービス品質のトレードオフを最適化する振る舞いを行う自己適応手法を提案する。本手法は xtUML を用いたモデル駆動開発と SPLE に基づく手法である。本提案手法を DFEAM (Dynamic Feature-oriented Energy-aware Adaptive Modeling) と名付けた。

3. DFEAM

消費電力とソフトウェア品質に関する自己適応を xtUML を用いて実現する手法を提案する。xtUML はモデル駆動アーキテクチャに基づき、モデル駆動開発へのアプローチをサポートする UML の拡張で、プログラムのように実行可能なシステム仕様を記述することが可能である。設計段階でのテストやパフォーマンス測定が可能でありモデル駆動開発を強力にサポートし、xtUML で記述したシステム仕様はソースコードに変換することができる。

xtUML を用いたモデル駆動開発による消費電力自己適応型ソフトウェアの開発方法論 [1] がすでに提案されている。この手法ではアプリケーションの可変性を記述したフィーチャモデルと xtUML による振る舞いの記述を関連づけることで、消費電力状況に応じた振る舞いをアプリケーション自身が決定することが可能である。しかし先行手法では、複雑なバリエーション間の品質を比較し消費電力と品質のトレードオフを解決するバリエーションを決めることができなかった。例としてモノクロまたはカラー出力の可変点と低速または高速通信の可変点を持つアプリケーションを挙げる。それぞれの可変点内では単一可変点での比較と同様にどちらを選択した方が QoS が高いかを比較できる。しかし、アプリケーション全体で見たとき (モ

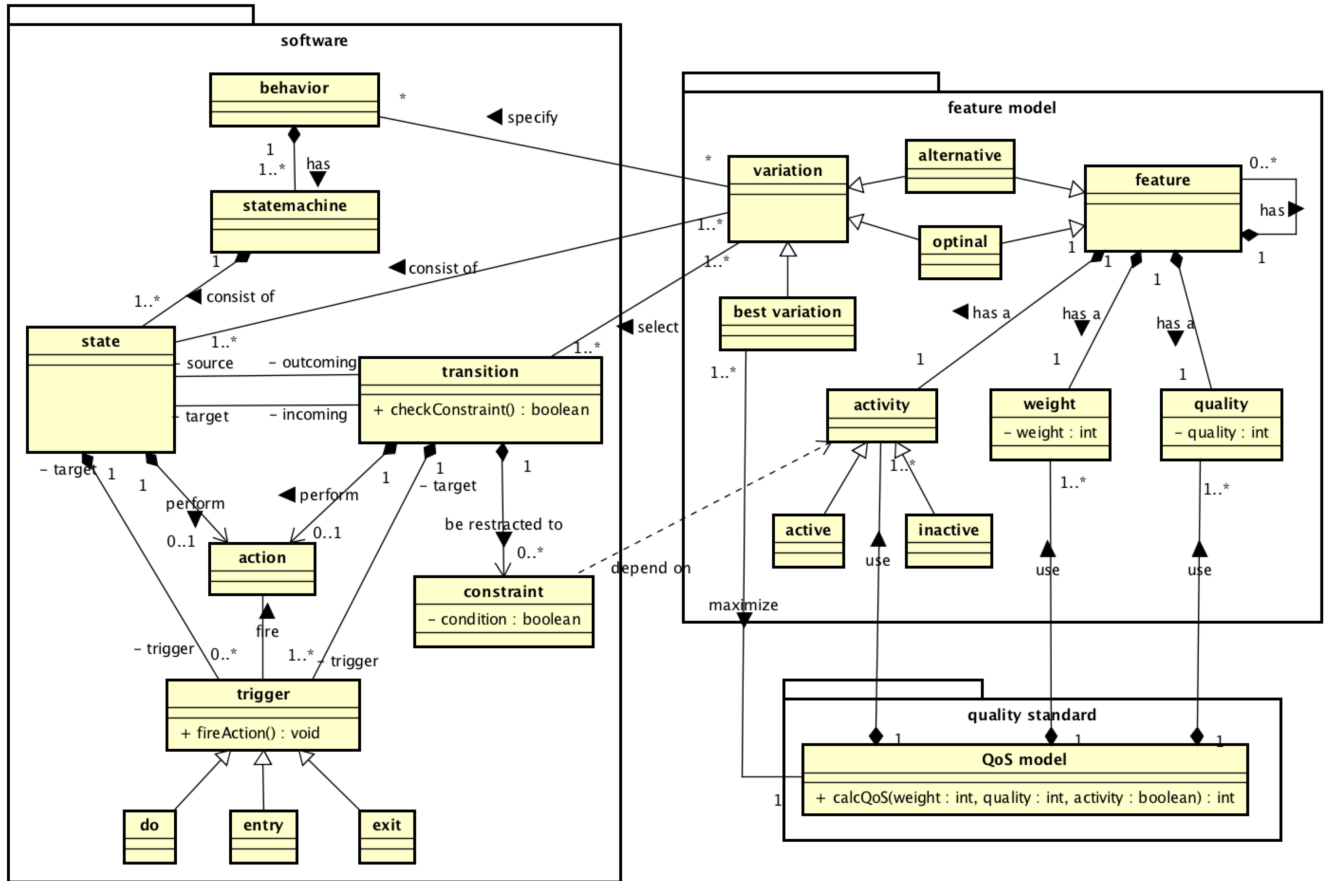


図 3 アプリケーションと評価基準の関係のメタモデル

Fig. 3 A metamodel of relation between application and quality standard

ノクロ、高速)の組のバリエーションと、(カラー、低速)の組のバリエーションはどちらも QoS が高いか単純に比較できない。そこで本提案手法では、バリエーション間の品質を比較するための定量的な QoS モデルを作成して先行研究を拡張し自己適応に QoS に関する考慮を追加する。

図 2 に提案手法全体のシステム構成を示す。提案手法は SPLE の手法で分析した可変性を関連づけた xtUML による振る舞いモデルと、MAPE-K [6] のコンセプトに基づいた自己適応コンセプトの 2 つからなる。MAPE-K は自己適応行動のためのコントロールループモデルである。このモデルはコンテキストおよびシステムを監視するモニタと、モニタ対象を分析するアナライザ、適切な適応を計画するプランニングコンポーネント、計画した適応を実行しシステムに反映する実行コンポーネントを含む。DFEAM の自己適応コンセプトでは主に Monitor がモニタとアナライザの役割を持ち、Manager がプランニングコンポーネントと実行コンポーネントの役割を持つ。図 3 はシステム構成要素を品質評価への影響の側面から表したメタモデルである。本項では図 2 の構成コンポーネントについて詳細を述べることで DFEAM について説明する。

3.1 可変性 xtUML モデル

アプリケーションの可変性を分析し可変性を表すフィーチャモデルと xtUML を関連づけることで可変性を持つ xtUML を記述する。

Behavior model(振る舞いモデル): 振る舞いのモデルはフィーチャモデルと xtUML のステートマシン図からなる。フィーチャモデルではフィーチャをソフトウェアの機能に割り当て、実行時に可変性のある機能を可変フィーチャとして記述する。ステートマシン図にはバリエーションによって異なる振る舞いを異なるステート遷移のパスで記述する。

フィーチャとステートの関連付けには可変点つきステートマシン図を用いる。可変点つきステートマシン図はステート遷移にガード条件を設けることができ、ガード条件によって活性フィーチャに応じたステート遷移を管理することが可能である。実行時には状況に応じたフィーチャモデルの再構成を行いバリエーションを変更することで自己適応行動を実現する。

Common or Variable Behavior(共通または可変な振る舞い): 振る舞いモデルは共通な振る舞いと可変な振る

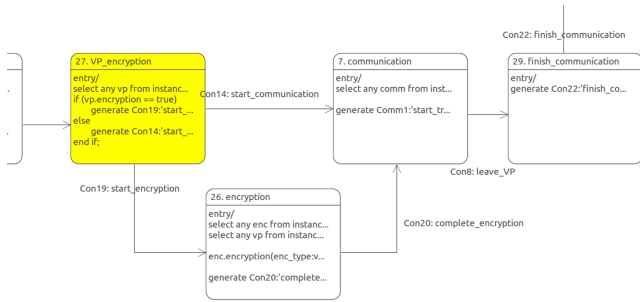


図 4 可変点におけるステートマシン図の記述

Fig. 4 A state machine description at variable point

舞いに分類できる。それぞれの振る舞いは先に述べたソフトウェア機能を割り当てたフィーチャモデルのフィーチャに対応している。共通フィーチャはバリエーションに寄らない共通の振る舞い、可変フィーチャはコンテキストに応じて可変性が求められる振る舞いである。図 4 に後述するケーススタディで設計した xtUML の可変な振る舞い部分のステートマシン記述を示す。可変な振る舞い部分では可変点付きステートマシン図を xtUML で実現するために、可変点ごとに新たに遷移を管理する状態を設置する。遷移管理状態には条件分岐を記述し、ガード条件と同様に活性フィーチャに応じた状態遷移管理を実現する。

3.2 自己適応コンセプト

図 2 のシステム構成では MAPE-K コンセプトの 5 つのコンポーネントに DFEAM の適応コンセプトのコンポーネントをマッピングしている。

Monitor: 実行時に選択するバリエーションはデバイスの消費電力に応じて決定する。そのために動的な消費電力を取得する機構が必要である。Monitor は実行時の消費電力を取得しソフトウェアの要求最大消費電力を導出する役割を持つ。xtUML のステートマシン図で記述された周期的な振る舞いによってある期間中の消費電力を推定する。

Estimated Power: 動的な消費電力を取得するためにモデルベースの消費エネルギー解析手法 [11] を用いる。この手法では設計時にデバイスのリソース消費量をパラメータとした消費電力推定モデルを作成し、実行時の動的な消費電力をこのモデルを用いて見積もることができる。

Resource Consumption: Monitor は消費電力を推定するためのパラメータとしてリソース消費量を使用する。これはリソース消費量がデバイスの消費電力に対応したパラメータであるからである。

Thresholds: 推定消費電力から消費電力状況を把握した Monitor は、Thresholds を参照して実行時の要求消費電力を導出する。Thresholds はバリエーションを変更すべき推定消費電力の閾値であり、閾値はバリエーションごとの消費電力を [11] の手法によって推定し決定する。

Required Power: リソース消費量の監視によって導出される Monitor の成果物が Required Power である。本手法では Required Power に応じて適応ポリシーにしたがったバリエーション変更を行う。

Manager: 実行時に求められる最適なバリエーションはコンテキストに応じて変化する。そのために動的な消費電力から最適なバリエーションを決定する機構が必要である。Manager の役割は Monitor によって導出された要求消費電力を基に最適なバリエーションを決定することである。Manager は選択バリエーションに対応するフィーチャの活性・不活性を行いフィーチャモデルを再構成する。

Feature: ソフトウェアの機能を割り当てた要素で、可変フィーチャは機能の重要度と品質に与える影響の関数を持つ。バリエーションマネージャが決定した最適なバリエーションは Feature の活性または不活性に誘導され、Feature の活性状況に依存する可変点付きステートマシン図のガード条件の可否によってソフトウェアに反映する。

Weight: 可変点に対応する可変機能がアプリケーションの機能としてどれだけ重要であることを示す。可変フィーチャ間での重要度の順番を表現するものであり、その差の大小には意味がない。

Quality: Quality は可変点が QoS に与える影響を表現する関数である。可変性の変化によってソフトウェアの品質に与える影響を 0 から 1 で正規化する。可変性が品質に与える影響は特定の値を境に意味のある大きになることが考えられる。あるいは特定の値以上では可変性が変化しても与える影響に差がなくなる。したがって、Quality は多くのソフトウェアでステップ関数になると考えられる。

QoS model: 各バリエーションの QoS を定量的に示すためのモデルである。QoS モデルの作成によってバリエーションごとの品質を共通の尺度で比較することで、自己適応に QoS に関する考慮を加えることができるようになった点が本手法における主な新規性である。

図 3 に示すように、QoS モデルは各フィーチャの重みと品質関数、またそのフィーチャの活性状況によって構成される。したがって、本手法におけるソフトウェアの品質は可変フィーチャに設定した重みと品質関数によって決定される。バリエーション間の QoS 値を比較するために 0 から最大 QoS 値で正規化される。本手法では、文献 [5] において Web サービス構成の QoS 最適化のために使用されたサービス構成要素の QoS の重み付け合計を参考に、ソフトウェア機能の重み付け平均を QoS モデルとして使用する。

Power model: 各バリエーションの推定消費電力を示すために、各可変点の活性状況を入力としその時の推定消費電力を出力とするモデルである。ある 1 つのバリエーションに対して消費電力を推定することは [11] の手法によって可能であった。しかし、複雑化する可変性はフィーチャの組み合わせを爆発的に増加させる。無数のバリエーション

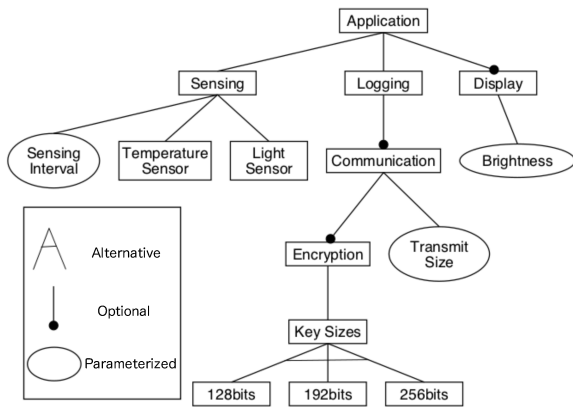


図 5 ケーススタディのフィーチャモデル
Fig. 5 Feature model in case study

に対して包括的に消費電力を推定するために、現実的な数の推定消費電力を重回帰分析によってモデルフィッティングする。

Table: 最適なバリエーションを決定するために参照されるのが対応表である。対応表の作成には *Power model* と *QoS model* を使用する。最適バリエーションの決定にあたってバリエーションを推定消費電力でグループ分けする。1つのグループにはある範囲の推定消費電力を持つバリエーションが属する。範囲は実行時に要求される消費電力に依存する。各グループ内で最大の QoS 値を持つバリエーションを制限消費電力下で最大の QoS を提供しトレードオフ関係を解決する *Best Variation* として採用する。採用されたバリエーションは要求消費電力を満たすバリエーションのうち、開発者が定義した重要度においては最も品質がいいバリエーションであると言える。

4. ケーススタディ

DFEAM の有用性を示すためのケーススタディを行う。ケーススタディに使用するアプリケーションのフィーチャモデルを図 5 に示す。Alternative フィーチャはいずれか 1 つを選択するフィーチャ、Optional フィーチャは有効・無効を選択するフィーチャである。Parameterized フィーチャは連続値を可変点として持つフィーチャである。

ケーススタディにおける可変性 xtUML のモデリングを BridgePoint で行った。BridgePoint は xtUML からコードの自動生成を行うことができモデル駆動開発におけるツールとして活用されている。BridgePoint ではクラス図を記述し特に振る舞いが存在するクラスにステートマシンを設置できる。さらに、ステートマシン図のステート内で振る舞いがあるものは専用のアクション言語を用いて詳細な振る舞いを記述できる。BridgePoint で記述するモデルは Bridge や Function を用いて外部のコードと接続することが可能である。本ケーススタディではアクション言語や Bridge, Function を用いて完全なモデル駆動開発を行うこ

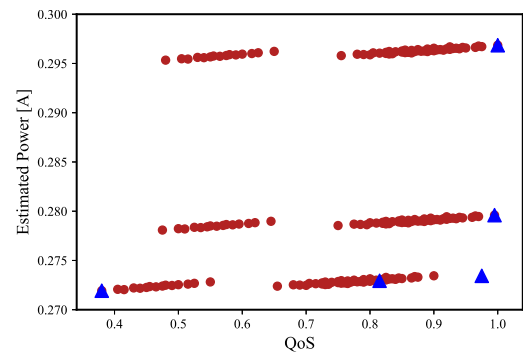


図 6 QoS と推定消費電力の散布図
Fig. 6 Scatter diagram of QoS and estimated power

とが可能であると考え BridgePoint を開発ツールとして採用した。

4.1 ケーススタディデザイン

ケーススタディとして簡単な環境情報の取得・保存を行うアプリケーションを設計した。以下にアプリケーションの概要を示す。

バッテリー駆動のデバイス上で動作する温度センサ・照度センサを用いて環境情報を取得する。両センサのセンシング間隔は同じである。取得データを温度と照度に変換しタイムスタンプを付加したものをある時刻での環境情報とする。環境情報はデバイスのローカルディスクに蓄積される。一定回数センシングすると蓄積した未送信の環境情報をサーバに送信する。サーバではデバイスから送信されたデータが保持されている。サーバではリアルタイムな環境情報を使用したサービスが提供される。サーバは複数のデバイスと通信することが考えられる。

デバイスが最も電力を消費するのは環境情報を送信するときであり、稼働時間を維持するためには送信回数を減らす必要がある。そのため、取得した環境情報は蓄積され一定量のデータとして送信される。しかし、取得した環境情報はリアルタイム性のあるサービスに使用されるため、サーバが保持する最新の情報が古くなる前に十分短い期間で送信される必要がある。加えて、温度・照度のデータは変化が緩やかであるため、データの有用性の観点ではある程度の時間的幅のあるデータであることが好ましい。また、デバイスとサーバの通信の安全性が保たれることや、デバイスの設置場所でサーバを介さずに目視で環境情報を確認できることも求められる。

以上の要件からアプリケーションの可変フィーチャに重みと品質関数を設定した。表 1 に各フィーチャに設定した重みと品質関数を示す。

品質関数に関して、Optional フィーチャである Dis-

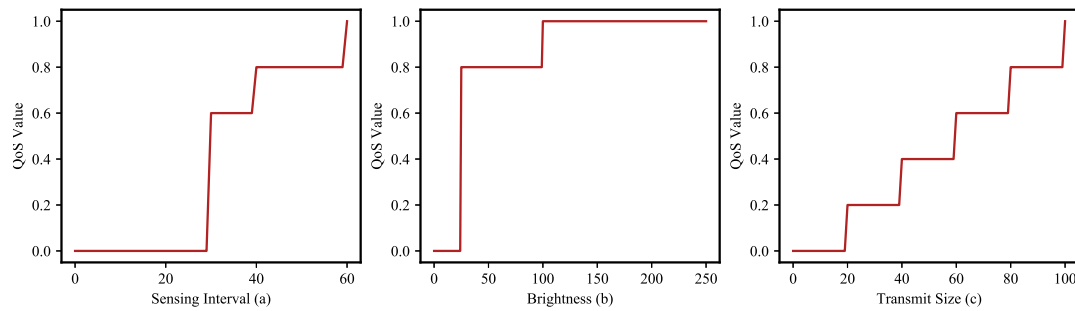


図 7 品質関数

Fig. 7 Quality function

表 1 フィーチャの重みと品質関数

Table 1 Weight and quality function

	Communication	Encryption	SensingInterval	KeySizes	TransmitSize	Display	Brightness
Weight	10	8	7	6	5	3	1
Quality	0/1	0/1	Fig. 7(a)	0.5/0.8/1	Fig. 7(c)	0/1	Fig. 7(b)

play, Communication, Encryption はフィーチャの有無に応じた 0/1 である。Alternative フィーチャである Key-Sizes は (KeySizes, QoS) = (128bits, 0.5), (192bits, 0.8), (256bits, 1.0) と定義する。Parameterized フィーチャである SensingInterval, TransmitSize, Brightness については図 7 の (a)~(c) に示すステップ関数で定義する。

上記の情報から QoS モデルを決定する。Power モデルは可変点の値をランダムに変更して得られた 80 種のバリエーションを使用してモデルフィッティングを行う。本ケーススタディではアプリケーションの動作に伴って変化する電流値のピーク値をモデル化し最大消費電力の削減を試みる。QoS モデルと Power モデルを使用してトレードオフを解決するバリエーションを 0.001A ごとに 26 個グループを探索する。全バリエーションの推定消費電力と QoS 値を図 6 に示す。図中の点は可変点を変更した時に起こりうる全バリエーションであり、5 つの三角の点は探索の結果得られた 5 つのバリエーションである。得られた 5 つのバリエーションを表 2 に示す。

4.2 評価

ケーススタディで得られた 5 つのバリエーション (V1~V5) に対して GQM モデル [3] を用いて DFEAM の有用性を評価する。GQM モデルは明確な目標を置き、目標を達成するために必要なメトリクスを対応づけるゴール思考の測定フレームワークである。GQM モデルを用いたソフトウェアの測定はこれまで数多くなされており [2], [4], 本研究の評価にも適していると判断した。GQM モデルが提供するガイドラインに従い評価を行う。表 3 に評価に使用した GQM モデルを示す。Q1 は選択した 5 つのバリエーション間で消費電力と QoS のトレードオフが存在し、トレードオフ最適化の前提が満たされているかを確認する

ためのものである。また、これを確認することで本手法で使用した恣意的な QoS モデルが妥当なものであることを示すことができる。Q1 を評価するために各バリエーションの推定消費電力 (M1) と QoS 値 (M2) を使用する。Q2 はある制限電力下で最大の QoS を提供するソフトウェアを開発するという要求を満たしているかを確認するためのものである。Q2 を評価するために各バリエーションの QoS 値 (M3) を使用する。

Q1 に関して、図 6 中の選択された 5 つのバリエーションを見ると、より消費電力の大きなバリエーションほど QoS 値が高いことがわかり、選択した 5 つのバリエーションには消費電力とサービス品質のトレードオフ関係が崩れることなく表れている。したがって、本手法が目的とするトレードオフ関係最適化の前提条件が満たされることとなり選択したバリエーション群が妥当な群であると言える。さらに Q2 に関して、選択バリエーション群にはトレードオフ関係があることが認められたので、消費電力がより高いバリエーションは QoS 値もより高い。したがって、制限電力下で最大限の消費電力を使用する 1 つのバリエーションを選択した時 QoS も最大であると言える。以上の結果から、Q1-2 は達成されるため適切な適応ができていると言え、本手法がトレードオフを最適化する自己適応方法論として有用であることを示すことができた。

本ケーススタディでは本手法を用いて決定される実行バリエーションは、特定の制限電力下で定義された優先度において最大の QoS を提供することを示している。しかし、消費電力削減の要求に対してどの程度の効果が認められるかについては示すことができていない。また、本手法は開発者がアプリケーションのユースケースを考慮して機能の重要度を恣意的に決定するものであり、選択されるバリエーション群は開発者やユースケースの制約を受ける。

表 2 選択された最適バリエーション
Table 2 Selected best variations

	Estimated Power[A]	QoS Value	Communication	Encryption	Sensing Interval	Key Sizes	Transmit Size	Display	Brightness
V1	0.2719	0.380	True	False	30	None	20	False	None
V2	0.2729	0.815	True	True	40	256bits	60	False	None
V3	0.2734	0.975	True	True	60	256bits	100	False	None
V4	0.2796	0.995	True	True	60	256bits	100	True	25
V5	0.2968	1.0	True	True	60	256bits	100	True	100

表 3 GQM モデル
Table 3 GQM model

Goal	Question	Metrics
適切に適応できている	選択されたバリエーション群にトレードオフが存在しているか (Q1)	推定消費電力 (M1)
		QoS 値 (M2)
	制限電力下で最大の QoS を提供しているか (Q2)	QoS 値 (M3)

5. おわりに

はじめに、組み込みシステムにはハードウェアの制限にともなって稼働時間の要求を満たすため消費電力削減の必要があるという背景を述べた。しかし、消費電力とサービス品質にはトレードオフがありこれらの解決の要求がソフトウェア開発に存在する。我々はトレードオフを解決するソフトウェア構成となるように自己適応を行うソフトウェアを xtUML を用いたモデル駆動開発で実現する自己適応モデリング手法 (DFEAM) を提案した。本手法では MAPE-K のコンセプトを取り入れた自己適応コンセプトと定量的に QoS を示すモデルを用いた最適バリエーションの探索によって、トレードオフを解決する自己適応行動を行う実行可能モデルをモデリングする。先行研究として消費電力に応じて自己適応を行うソフトウェア開発方法論が存在する。本研究はフィーチャモデル構成で品質が決定される QoS モデルを作成し、電力に関して自己適応を行う先行研究に QoS についての考慮を加えた点で新規性がある。ケーススタディでは可変点を複数持つアプリケーションに対して本手法を適用し GQM モデルを用いて評価した。評価の結果、電力制限下で最大のソフトウェア品質を提供するように適切な適応がなされていることが示され、DFEAM が消費電力と QoS のトレードオフを最適化する自己適応手法として有用であると言える。しかし、本手法は開発者による恣意的な優先度設定やユースケースに依存する点で制約が存在する。今後の課題としてフィーチャ間に依存関係のあるフィーチャ構成での QoS モデルの作成方法の確立や、モデルフィッティングコストを削減した時の自己適応能力への影響の評価が考えられる。

謝辞 本研究は、科研費（課題番号：15H05708）の助成を受けて行っているものです。

参考文献

- [1] F. Tanaka, K. Hisazumi, S. Ishida, and A. Fukuda, "A Methodology to Develop Energy Adaptive Software Using Model-Driven Development," IEEE TENCON 2017, pp.769-774, 2017
- [2] G. H. Alfred, V. Pelechano, R. Mazo, C. Salinesi, and D. Diaz, "Dynamic adaption of service compositions with variability models," the Journal of Systems and Software, pp.24-47, 2014.
- [3] V. R. Basili, G. Caldiera, and H. D. Rombach, "Goal question metric paradigm," Encyclopedia of Software Engineering 1, pp.528-532, 1994.
- [4] T. Chen, B. H. Far, and Y. Wang, "Development of an Intelligent Agent-Based GQM Software Measurement System," Proceedings of the ATS 2003 Conference, pp.188-197, 2003.
- [5] Angus F. M. Huang, Ci-Wei Lan, and Stephen J. H. Yang, "An optimal QoS-based Web service selection scheme," Information Sciences: an International Journal, Volume 179 Issue 19, pp.3309-3322, 2009.
- [6] J.O. Kephart and D.M. Chess, "The vision of autonomic computing," IEEE Computer Society, Volume 36, Issue 1, pp.41-50, 2003.
- [7] M. Luckey, B. Nagel, C. Gerth and G. Engels, "Adapt Cases: Extending Use Cases for Adaptive Systems," Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp.30-39, 2011.
- [8] F. D. Macías-Escrivá, R. Haber, R. del Toro, and V. Hernandez, "Self-adaptive systems: A survey of current approaches, research challenges and applications," Expert Systems with Applications 40. 18, pp.7267-7279, 2013.
- [9] K. Mens, R. Capilla, H. Hartmann, and T. Kropf, "Modeling and Managing Context-Aware Systems' Variability," IEEE Software, Volume 34, Issue 6, pp.58-63, 2017
- [10] D. Weyns, M. U. Iftikhar, D. Gil, and T. Ahmad, "A survey of formal methods in self-adaptive systems," C3S2E, pp. 67-79, 2012.
- [11] R. Yoshimoto, T. Kadono, K. Hisazumi, and A. Fukuda, "A Software Energy Analysis Method Using ExecutableUML," IEEE TENCON 2016, pp.218-221, 2016.