2007－EVA－23 (3)
2007／11／19

プロダクトライン開発におけるコアアセット変更コストの見積り

高祖 靖臣[†], 中西 恒夫[††], 北須賀 輝明[‡],
田頭 茂明[††], 福田 晃[††]

[†] 九州大学大学院システム情報科学府
[††] 九州大学大学院システム情報科学研究院
[‡] 熊本大学大学院自然科学研究科

プロダクトライン開発では，変化する市場の動向に応じて，再利用資産であるコアアセットに変更が加えられる．コアアセットの変更を計画するにあたり，その変更コストの見積りが不可欠である．コアアセットには設計文書などのコード以外の資産も含まれるが，既存の手法はコード行数を基に変更コストを見積るものが多い．既存の手法より正確な見積りを得ることを狙い，本提案手法では，コアアセットの資産種別ごとに変更コストを求め，足し合わせることでコアアセット全体の変更コストを見積る．資産種別ごとの変更コストは，フィーチャの変更が各クラスに及ぼす影響の大きさを分析し，変更するクラス・メソッドの数，及びコード行数に基づいて求める．

Estimating Core Assets Modification Cost in a Software Product Line

Yasuomi KOUSO[†], Tsuneo NAKANISHI[††], Teruaki KITASUKA [‡],
Shigeaki TAGASHIRA[††], and Akira FUKUDA[††]

[†] Graduate School of Information Science and Electrical Engineering, Kyushu University
[††] Faculty of Information Science and Electrical Engineering, Kyushu University
[‡] Graduate School of Science and Technology, Kumamoto University

In software product line engineering, change of market requirements causes modification of reusable artifacts, namely core assets. It is essential to estimate core assets modification cost on planning the modification. Core assets include not only codes but also various artifacts such as software design documents. However, most of existing cost models estimate core assets modification cost based on LOC. Our method evaluates a modification cost for each type of core assets and estimates core assets modification cost by summing up those costs in order to get more accurate estimation than existing methods. The method analyzes impact on classes, which is caused by modification of features. The modification cost for each type of core assets is evaluated based on the number of features, classes and LOC to be modified.

1 はじめに

ソフトウェアプロダクトライン開発方法論 (SPLE: Software Product Line Engineering) は，同一の製品分野に属する製品群を包括的に開発する再利用技術である．SPLE では製品間の機能や特徴に関する共通性・相違性を分析した上で，コアアセットと呼ばれる再利用可能な開発資産を，各機能・特徴と関連付けて戦略的に構築する．コアアセットとは，各個別製品で共有される開発資産であり，分析・設計資産やコード，テストケース，ツールなどを含む．製品開発時は，各個別製品の機能や特徴をもとにコアアセットから必要な要素を選定し，それらを前もって定められた方法で組み合わせることで製品を導出する（文献[1]）．

市場が要求する製品機能の変化により，新規機能の追加や既存機能の変更がしばしば発生する．製品機能の追加・変更は，コアアセットに対し変更を加えることによって実現できる．コアアセットの変更を計画するにあたり，その変更コストを見積ることが必要である．なぜなら，投資対効果の視点において，コアアセットの変更が有益であるかどうかを判断しなければならないからである．

米国の Boehm らやドイツの Böckle らにより，コアアセット構築後，および製品導出後の機能変更に要するコストの見積りモデルが提案されている（文献[2] [3] [4] [5] [6]）．これらのモデルは，プロダ

クトライン開発で複数製品を開発するのに要するコストと，製品間で部品の再利用をすることなく各製品を別個独立に開発するのに要するコストを比較するために考案された．これらのモデルでは，設計資産やテスト資産などの各資産種別ごとに変更コストを見積ることは行っていないが，各資産種別ごとに変更コストを求めることで，より正確な見積りが得られると考えられる．

そこで本稿では，各資産種別ごとに変更コストを求め，それを基にコアアセット変更コストを計算する見積り方法を提案する．各章の構成は以下のとおりである．第二章では，SPLE，および変更コスト見積りについて述べる．第三章では，Boehm らと Böckle らによるコスト見積り手法について説明する．第四章では，提案する変更コスト見積り手法を示す．第五章では，まとめと今後の課題について述べる．

2 SPLE，変更コスト見積り

2.1 SPLE

特定の市場を対象とした組込み製品は，製品間で機能や特徴に共通性・相違性が存在する複数の製品を開発することが多い．そのため，機能や特徴の製品間共通性・相違性を分析した上で，計画的に構築した開発資産を再利用することで各個別製品の開発を効率化できる．こうした計画的な再利用により，同一製品分野に属す複数製品を効率的に開発する方法論を，ソフトウェアプロダクトライン開発方法論（SPLE: Software Product Line Engineering）という．

SPLE は，製品間で共有利用する開発資産を構築するドメイン開発と，共有資産を用いて各個別製品の開発を行うアプリケーション開発という二つの活動から成る．ドメイン開発では，まず市場の動向を調査し，対象製品分野における機能・特徴に関する製品間共通性・相違性を分析する．その分析結果をもとに，コアアセットと呼ばれる共有開発資産を計画的に構築する．コアアセットは，アーキテクチャ，UML 分析・設計資産，コード，テストケース，ツールなどが含まれる．コアアセットは，製品の各機能・特徴と関連付けて管理する．また，各要素の組み合わせ方や，目的の機能・特徴を持つ製品の構成手順は事前に定めておく．アプリケーション開発では，開発対象製品に求められる機能・特徴をもとに必要なコアアセット要素を選定し，ドメイン開発で定められた構成方法に従い対象製品を導出する．

2.2 変更コスト見積り

市場が要求する製品機能の変化により，新規機能の追加や既存機能の変更が発生する．製品機能の追加・変更は，コアアセットに対し変更を加えることによって実現できる．コアアセットの変更を計画するにあたり，その変更コストを見積ることが必要である．なぜなら，投資対効果の視点において，コアアセットの変更が有益であるかどうかを判断しなければならないからである．また，コアアセットの変更に要する人員や器材の調達計画を立てる上で，変更規模・変更コストを参考にする必要もある．

コアアセットの変更コスト見積り手法・計算モデルとして，次章で述べる Boehm らや Böckle らなどによるものが提案されている．

3 関連研究

3.1 COPLIMO

Boehm らが文献[2) 3)]で提案する COPLIMO (COnstructive Product Line Investment MOdel) は，従来の非プロダクトライン型開発を対象としたコスト見積りモデル COCOMO II（文献[7)]）を，プロダクトライン開発のコスト見積りに対応できるよう拡張したモデルである．コアアセットの構築コスト，およびコアアセット変更コストを，追加・変更コード行数を基に見積る．変更ソースコード行数は，変更前のコード行数に変更割合を掛けることにより算出する．

コアアセット変更コスト C[人月] は，下記の式 1 で計算する．

$$C = a \cdot (AMSIZE)^b \cdot \Pi(EM) \qquad (1)$$

各項の意味は以下のとおり．

- a および b: プロジェクトごと，もしくは開発組織ごとに設定される定数．過去の開発データを基に設定する．
- $AMSIZE$: 補正済み変更コード行数．単位はキロ行．変更するコードの行数を，プログラマの習熟度やコードの理解容易性に応じて補正したもの．
- $\Pi(EM)$: COCOMO II のコスト補正項．求められる製品品質の高さや開発環境の充実度などに応じて値が定まる．

3.2 Böckle らによる見積りモデル

Böckle らは，文献[4) 5)]において，開発組織立ち上げコストや製品導出コストなどのコスト要素

に基づく，コアアセット構築コスト・製品導出コストの見積りモデルを提案している．コアアセット構築とN個の製品を導出するのに要するコストC[人年]は，下記の式2で求める．

$$C = C_{org} + C_{cab} + \sum_{i=1}^{N}(C_{unique}(P_i) + C_{reuse}(P_i)) \quad (2)$$

各項の意味は以下のとおり．全ての項の単位は人月である．

- C_{org}：プロダクトライン開発を導入するにあたり，開発組織を仕立て直すのにかかるコスト．組織の再構成，メンバのトレーニング，プロセス改善にかかるコストを含む．
- C_{cab}：コアアセットを構築するのに要するコスト．製品間の共通部・相違部の分析，アーキテクチャの構築，設計・実装，文書化，テスト資産構築にかかるコストから成る．
- $C_{unique}(P_i)$：i個目の製品（P_i）に固有であり，コアアセットから導出することができないソフトウェア部品を開発するのに要するコスト．
- $C_{reuse}(P_i)$：i個目の製品（P_i）用の資産を，コアアセットから導出するのにかかるコスト．コアアセットから必要な資産を見つけ出すコスト，資産内の可変部分を目的の製品向けに固定・設定するコスト，導出した資産に対しシステムテストを行うコストから成る．

Böckleらは，プロダクトライン開発で複数製品を開発するのに要するコストを見積る上で，上記の各項を考えなければならないとしている．ただし各項の値は，専門家の経験や既存の見積り手法を応用することにより求められるとしているのみで，具体的な計算方法を示していない．

3.3 Ganesanらによる見積りモデル

Ganesanらは，文献[6]において式2のコアアセット構築コスト・製品導出コストの見積りモデルを拡張し，コアアセット構築後に変更を加えた場合の累積コストを見積る計算モデルを考案している．その計算モデルは，Böckleらによる計算モデルにコアアセット変更コストの項を1つ設けたものである．すなわち，M回変更を行ったときの累積コストは下記の式3の形で表される．

$$C = C_{org} + C_{cab} + \sum_{i=0}^{M}(C_{reinv_i}) + \sum_{i=1}^{N}(C_{unique}(P_i) + C_{reuse}(P_i)) \quad (3)$$

ただし，C_{reinv_i}は，コアアセットのi回目の変更に要するコストである．

文献[6]では，$0.5 \cdot C_{cab} < C_{reinv_i} < 2 \cdot C_{cab}$の範囲でモンテカルロ分析を行い，コアアセット変更コストの総和：$\sum_{i=0}^{M}(C_{reinv_i})$を決定している．なお，$C_{cab}$はCOCOMO IIを用いて求めている．

4 変更コスト見積り

前章で述べたBoehmらやBöckleらが提案するコアアセットの変更コスト見積り手法・計算モデルは，プロダクトライン開発で複数製品を開発するのに要するコストと，製品間で部品の再利用をすることなく各製品を別個独立に開発するのに要するコストを比較するために考案されたモデルである．これらのモデルでは，設計資産やテスト資産などの各資産種別ごとに変更コストを見積ることは行っていないが，各資産種別ごとに変更コストを求めることで，より正確な見積りが得られると考えられる．そこで本提案手法では，各種資産要素ごとに変更コストを求め，そこからコアアセット変更コストC_{change}を見積る．

4.1 コストモデル

プロダクトライン開発では，フィーチャという製品の機能や特徴を表す概念を用いて，製品間の共通性・相違性を表現する．そのフィーチャ間の関係をまとめたモデルがフィーチャ図[8)9)]である．本提案手法は，オブジェクト指向を用い，フィーチャを中心とした設計・実装を行っていることを前提としている．コアアセットに含まれる各開発資産はフィーチャと関連付けて管理される．

市場が要求する製品機能の変化は，フィーチャ図への新たなフィーチャの追加や既存フィーチャの変更で表現することができる．フィーチャの変更に続いて，変更フィーチャと関連するコアアセット資産，すなわちクラス図・シーケンス図などの設計資産，コード，テストケース，開発支援ツールに対して変更を反映する必要がある．本稿では，設計資産変更コスト，コード資産変更コスト，テスト資産変更コスト，ツール資産変更コストをそれぞれC_{design}，C_{code}，C_{test}，C_{tool}と記述することにする．

コアアセット変更コスト C_{change} を以下のように定める．

$$C_{change} = C_{design} + C_{code} + C_{test} + C_{tool} \quad (4)$$

- 設計資産変更コスト： C_{design}
  市場の要求の変化により，分析・設計資産を変更するのに要するコストのこと．クラス図やシーケンス図，状態遷移図などの分析・設計文書を書き換えるのに要するコストや，分析・設計文書のレビュー・検証に要するコストを含む．

- コード資産変更コスト： C_{code}
  変更を加えた分析・設計資産に従って，コードを変更するのに要するコストのこと．コードを書き換えるコストや，コードレビューに要するコスト，分析・設計文書とコードとの整合性チェックに要するコスト，単体テストに要するコストを含む．

- テスト資産変更コスト： C_{test}
  変更を加えた分析・設計資産とコード資産に応じて，テスト資産を変更し，テストを実行，システムテストが完了するまでに要するコストのこと．ただし，単体テストに要するコストは含まないものとする．テストケースを書き換えるコストや，各種テストを実行し，コードを修正するコストを含む．

- ツール資産変更コスト： C_{tool}
  開発で利用するツールに変更を加えるたり，新たに導入するのに要するコストのこと．ツールの変更・導入開始から実際に利用可能になるまでのコスト．

以下では，設計資産変更コスト C_{design} とコード資産変更コスト C_{code} を見積る方法について詳細に説明する．

4.2 見積り方法

設計およびコードを変更するのに要するコストの見積りは以下の流れに従って行う．

1. 変更フィーチャの分析：新たに追加するフィーチャや既存フィーチャの変更内容を分析する．
2. クラスへの影響の分析： フィーチャの変更に伴って影響を受ける既存クラスや，新規に作成するクラスを分析する．
3. 設計資産変更コスト C_{design} の見積り： 変更を加える既存クラス数，および新規作成するクラス数から C_{design} を見積る．
4. コード資産変更コスト C_{code} の見積り： 変更するコード行数，および新規作成するコード行数から C_{code} を見積る．

以下で各フェーズについて詳しく説明する．

変更フィーチャの分析： 市場の要求の変化によって，新たに追加するフィーチャや，内容を書き換えたり，削除する既存フィーチャを分析する．分析の際には，市場の要求を満たす目的で追加・変更するフィーチャと相互作用を持つがために，変更を加える必要があるフィーチャも分析する．新規追加フィーチャの集合を $Feature_{new}$ とし，内容を書き換えたり，削除する既存フィーチャの集合を $Feature_{edit}$ とする．

例として，図1のフィーチャを持つ電子ポットプロダクトラインを考える．各長方形はフィーチャを表す．長方形に何もついていないフィーチャは製品間で共通のフィーチャ（mandatory フィーチャ）であり，長方形の上部に丸がついているフィーチャは製品ごとに選択可能なフィーチャ（optional フィーチャ）である．各フィーチャの意味は以下のとおりとする．

- 沸騰機能（98 ℃）： ポット内の水を 98 ℃になるまで熱する機能．
- 保温機能（90 ℃）： ポット内の水の温度を 90 ℃に保つ機能．
- タイマ機能： ユーザがタイマ開始を通知してから 3 分後にブザーを鳴らす機能．
- 自動ロック機能： ポットのふたを閉めた時，自動的にふたのロックをかける機能．

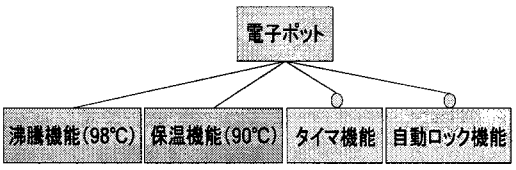

図 1: 変更前の電子ポットフィーチャ図

このフィーチャに，新たな optional フィーチャ「ミルク温め機能」フィーチャを加え，「タイマ機能」

フィーチャの振る舞いを以下のように変えることを考える．

- ミルク温め機能： ポット内のミルクを 60 ℃まで熱し，保温する機能．
- タイマ機能： ユーザがタイマ時間を 1 分刻みで最大 60 分まで指定し，指定時間経過後にブザーを鳴らす機能．

この場合，$Feature_{new} = \{$ミルク温め機能$\}$, $Feature_{edit} = \{$タイマ機能$\}$ となる．変更後のフィーチャ図を図 2 に示す．

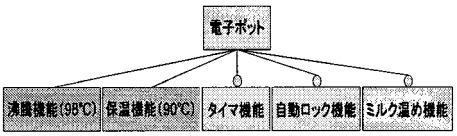

図 2: 変更後の電子ポットフィーチャ図

クラスへの影響の分析: 表 1 に表すような，フィーチャとそれを実現するのに必要となるクラスとの対応表（フィーチャ/クラス対応表）を用い，フィーチャの変更に伴って影響を受ける既存クラスや，新規に作成するクラスを特定する．$Feature_{new}$，および $Feature_{edit}$ と対応関係のあるクラスを抜き出し，それらがフィーチャの変更により影響を受けるクラスとなる．

「ミルク温め機能」フィーチャを追加し，「タイマ機能」フィーチャを変更する場合，表 1 から既存の「ポット制御」クラス，「タイマ制御」クラス，「ブザー」クラス，「ボタン」クラスに変更を加え，「ヒータ制御方式」クラスと「保温メニュー」クラスを新たに作成する必要がある．

既存フィーチャと既存クラスの対応関係はコアアセット構築時に定義されているので，その情報を用いることができる．一方，フィーチャの追加や変更によって新たに作成するクラスおよびフィーチャとの対応関係は，見積りを行う技術者が決定し，上記のフィーチャ/クラス対応表に反映しなければならない．

「ミルク温め機能」フィーチャを追加し，「タイマ機能」フィーチャを変更する場合，「ヒータ制御方式」クラスと「保温メニュー」クラスを新たに作成する必要があると考えられる．新規追加クラスとフィーチャとの対応関係は，表 1 の通りとなる．

表 1: フィーチャ／クラス対応表

フィーチャ／クラス対応表				変更フィーチャ		新規追加フィーチャ
		沸騰機能	保温機能	タイマ機能	自動ロック機能	ミルク温め機能
既存クラス	ポット制御	×	×	×	×	×
	温度制御	×	×			×
	ヒータ	×	×			×
	温度計	×	×			×
	タイマ制御			×		
	ブザー			×		
	ボタン			×		
	ふたロック				×	
新クラス	ヒータ制御方式	×	×			×
	保温メニュー		×			×

設計資産変更コスト C_{design} の見積り： 変更を加える設計資産として，クラス図，シーケンス図／コミュニケーション図，状態遷移図を考える．

これらの設計資産の変更コストは，以下の要素に関連していると考えられる．

- 変更を加える既存メソッドの数
- 既存クラスに追加するメソッドの数
- 新規作成クラスのメソッドの数
- 変更・追加するクラスと対応関係にある，optional フィーチャや alternative フィーチャ（選択可能フィーチャ）の数

追加・変更メソッドの数が C_{design} に影響を与えると考える理由は，メソッドを追加・変更することで，オブジェクト間に新たなメッセージ送受信が発生したり，既存メッセージ送受信を変更することが考えられるからである．メッセージの送受信の流れが変化した場合，シーケンス図／コミュニケーション図だけでなく状態遷移図にも反映する必要がある．そのため，追加・変更メソッドの数が C_{design} に影響を与えると考える．

電子ポットプロダクトラインの UML 設計図の例を用いて説明する．図 4 は，図 3 の設計 UML 図に対し，「ミルク温め」フィーチャを追加した時の設計 UML 図である．「ミルク温め」フィーチャを実現するため，図 4 の「ポット制御」クラスに「ミルク温め ()」を，「ヒータ制御方式」クラスに「方式設定 ()」などの新たなメソッドを追加している．また，「ヒータ」クラスの「ON ()」メソッドを「ON（加熱強度）」に変更するなど，既存のメソッドの定義変更も行っている．これらのメソッド変更に対応して，シーケンス図や状態遷移図に対して変更が加わっている．

変更・追加するクラスと対応関係にある選択可能フィーチャの数が，C_{design} に影響を与えると考える理由は，フィーチャの選択状況に応じて，メッセージ送受信の流れや状態遷移を切り替えることができるように設計する必要があるからである．メッセージ送受信の流れや状態遷移を切り替えを考慮しなければならない場合の設計は，切り替えを意識する必要のない場合の設計よりも複雑であり，より多くのコストを要すると考えられる．そのため，変更・追加するクラスと対応関係にある選択可能フィーチャの数が，C_{design} に影響を与えると考える．

製品ごとに「ミルク温め」フィーチャの有無を選択できるように，図 4 のシーケンス図（沸騰シナリオ）の例では，「ヒータ制御方式」オブジェクトの取捨が選択でき，メッセージの流れも「ミルク温め」フィーチャの有無に応じて変化するように記述している．一方，「ミルク温め」フィーチャの有無を製品ごとに選択せず，すべての製品が「ミルク温め」フィーチャを有するように定めた場合，シーケンスに参加するオブジェクトやメッセージの流れを切り替えるような機構を用意する必要はない．以上のことから，変更・追加するクラスと対応関係にある選択可能フィーチャの数が，C_{design} に影響を与えると考える．

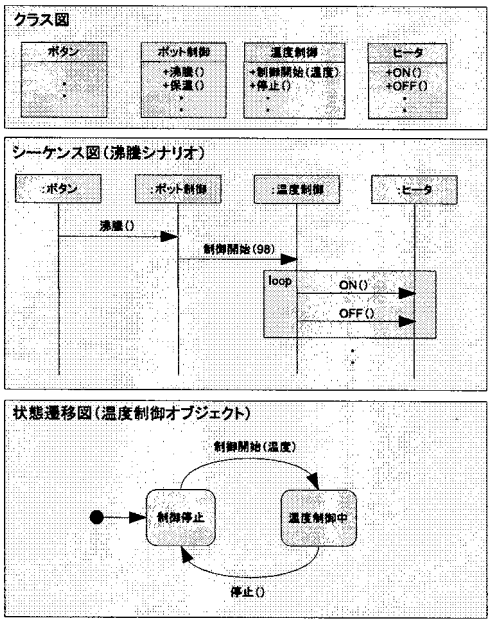

図 3: 変更前の電子ポット UML 設計図の例

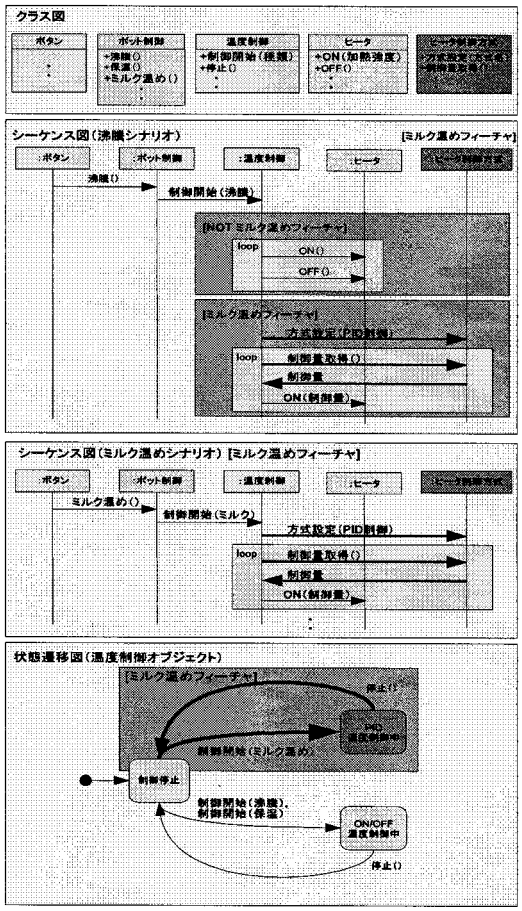

図 4: 変更後の電子ポット UML 設計図の例

以上のことを踏まえ，変更・追加メソッドの数，および対応関係にある選択可能フィーチャの数から C_{design} を次式のように見積る．

$$C_{design} = a \cdot NOM^b \quad (5)$$
$$NOM = \sum_{i=1}^{N_{class}} (M(Class_i) \cdot W(Class_i))$$

ただし，

- a および b：プロジェクトごと，もしくは開発組織ごとに設定する定数．コアアセット構築時のデータから求めるものとする．
- $M(Class_i)$：クラス $Class_i$ に対して，追加・変更するメソッドの数．
- $W(Class_i)$：クラス $Class_i$ と対応関係を持つ選択可能フィーチャの数に応じて，$M(Class_i)$ を補正する値．
- N_{class}：クラスの数．

電子ポットの例で，$M(Class_i)$，$W(Class_i)$ が表2の値をとる場合，$NOM = 12+3.6+3.6+0+4.8+0+1.2+9.6+3.7 = 38.5$[メソッド] となる．$a = 0.2$，$b = 1.1$ とすると，$C_{design} = 0.2 \cdot 38.5^{1.1} = 11.1$[人月] となる．ただし，

$$W(Class_i) = \begin{cases} \text{選択可能フィーチャ数} \times 1.2 \\ \qquad (\text{選択可能フィーチャ数} > 0) \\ 1 \\ \qquad (\text{選択可能フィーチャ数} = 0) \end{cases}$$

と仮定して計算している．

コード資産変更コスト C_{code} の見積り： 変更する既存コード行数，および新規作成するコード行数から C_{code} を次式のように見積る．

$$C_{code} = a \cdot (LOCSIZE)^b \quad (6)$$
$$LOCSIZE = \sum_{i=1}^{N_{class}} (LOC_{new}(Class_i) + LOC_{edit}(Class_i) \cdot F(Class_i))$$

ただし，

- a および b：プロジェクトごと，もしくは開発組織ごとに設定する定数．コアアセット構築時のデータから求めるものとする．

表 3: コード変更コスト見積りのためのパラメータ

		LOC_{edit}	F	LOC_{new}	LOC_i
既存クラス	ポット制御	1.2	0.4	0.1	0.58
	温度制御	0.8	0.2	0.3	0.46
	ヒータ	0.3	0.1	0	0.03
	温度計	0.2	0	0	0
	タイマ制御	0.3	0.8	0.2	0.44
	ブザー	0.2	0	0	0
	ボタン	0.2	0.3	0	0.06
	ふたロック	-	-	-	-
新クラス	ヒータ制御方式	0	0	0.6	0.6
	保温メニュー	0	0	0.8	0.8

- $LOC_{new}(Class_i)$：クラス $Class_i$ に対して，新規追加するコード行数．
- $LOC_{edit}(Class_i)$：クラス $Class_i$ の変更前のコード行数．
- $F(Class_i)$：クラス $Class_i$ の変更前コード行数のうち，変更を加えるコード行数の割合．
- N_{class}：クラスの数．

電子ポットの例で，$LOC_{new}(Class_i)$，$LOC_{edit}(Class_i)$，$F(Class_i)$ の各値が表 3 のようになる場合，$LOCSIZE = 0.58 + 0.46 + 0.03 + 0 + 0.44 + 0 + 0.06 + 0.6 + 0.8 = 2.97$[キロ行] となる．$a = 4.3$，$b = 1.2$ とすると，$C_{code} = 4.3 \cdot 2.97^{1.2} = 15.9$[人月] となる．

5 まとめと今後の課題

プロダクトライン開発は，コアアセットと呼ばれるソフトウェア資産を戦略的に構築，再利用することにより，多品種製品の開発を効率化する．本稿では，コアアセットの資産のうち，設計資産とコードについてその変更コストを求める方法を提案した．変更を加えるフィーチャを分析し，フィーチャ／クラス対応表を用いてクラスに対する影響分析を行った．変更の影響が及ぶクラスごとに変更を加えるメソッド数と選択可能フィーチャの数を見積もり，それを基に設計変更コストを求めた．コード変更コストは，各クラスの変更行数を基に求めた．

今回，設計資産変更コストおよびコード資産変更コストを見積る方法について提案したが，資産全体の変更コストを求めるためには，テスト資産およびツール資産変更コストの見積り方法を示す

表 2: 設計変更コスト見積りのためのパラメータ

		追加メソッド数	削除メソッド数	変更メソッド数	$M(Class_i)$	選択可能フィーチャ数	$W(Class_i)$	M*W
既存クラス	ポット制御	2	0	3	5	2	2.4	12
	温度制御	1	0	2	3	1	1.2	3.6
	ヒータ	0	0	3	3	1	1.2	3.6
	温度計	0	0	0	0	1	1.2	0
	タイマ制御	2	1	1	4	1	1.2	4.8
	ブザー	0	0	0	0	1	1.2	0
	ボタン	1	0	0	1	1	1.2	1.2
	ふたロック	-	-	-	-	-	-	-
新クラス	ヒータ制御方式	8	0	0	0	1	1.2	9.6
	保温メニュー	12	0	0	0	1	1.2	3.7

必要がある．また，今回提案した見積り手法の実験的評価を行う予定である．

参考文献

1) Paul Clements and Linda Northrop, "Software Product Lines : Practices and Patterns," Addison-Wesley, August 2001.

2) Barry Boehm, A. Winsor Brown, Ray Madachy, and Ye Yang, "A Software Product Line Life Cycle Cost Estimation Model," *Proc. of the 2004 International Symposium on Empirical Software Engineering*, pp.156–164, August 2004.

3) Hoh Peter In, Jongmoon Baik, Sangsoo Kim, Ye Yang, and Barry Boehm, "A Quality-based Cost Estimation Model for the Product Line Life Cycle," *ACM Communication*, Vol.49, No.12, pp.85–88, 2006.

4) Günter Böckle, Paul C. Clements, John D. McGregor, Dirk Muthig, and Klaus Schmid "Calculating ROI for Software Product Lines," *IEEE Software*, Vol.21, No.3, pp.23–31, 2004.

5) Klaus Pohl, Günter Böckle, and Frank van der Linden, "Software Product Line Engineering - Foundations, Principles, and Techniques -," Splinger, September 2005.

6) Dharmalingam Ganesan, Dirk Muthig, and Kentaro Yoshimura, "Predicting Return-on-Investment for Product Line Generations," *Proc. of the Software Pruduct Line Conf. (SPLC) 2006*, pp.13–22, August 2006.

7) "COCOMO II Model Manual," *ftp://ftp.usc.edu/pub/soft_engineering/COCOMOII/cocomo99.0/modelman.pdf.*

8) K. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh, "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures," *Annals of Software Engineering*, pp.143–168, 1998.

9) K. Kang, J. Lee, and P. Donohoe, "Feature-Oriented Product Line Engineering," *IEEE Software*, Vol.19, No.4, pp.58–65, 2002.