SoPL における動的再構成の実現に関する研究
—代替サービス検索のための枠組みの提案—

M2012MM010 岩田 繁
指導教員：野呂昌満

1　はじめに

個々の顧客の要求や，環境の変化などに対応したシステム構成の動的な変更の要求が，様々な分野で増加している[3]．システムの実行時まで要求が固まらない製品のための解決策として Dynamic Software Product Line(以下，DSPL)[4] がある．DSPL は事前定義された情報をもとに製品系列の可変部分を動的に決定し，要求や状況に応じて製品構成を変更する考え方である．SOA を用いた DSPL の実現方法として Service-oriented Product Line(以下，SoPL)[2] がある．SoPL の分野ではサービスの再利用や動的再構成，サービスが利用できなくなった場合のために，動的にサービスを検索する枠組みが必要となる．

サービスの検索は一般にサービスとオペレーションの名前の一致により行なわれている．その際に，開発者がサービスについての記述を確認することで代替可能なサービスであるかを判断している．名前に全ての意味を集約してサービスの検索を自動化した場合，誤ったサービスを利用してしまう．精度の高い動的なサービス検索を実現するためには，サービスの動的，並びに静的な意味を記述し，それらを検索に使うことが必要になると考えた．

本研究の目的は，サービスの意味を考慮して，代替可能サービスを検索するための枠組みの提案である．そのために，代替可能なサービス仕様の条件と実行可能な確認方法の提案を行なう．精度の高いサービス検索を実現することで，信頼性の高い SoPL を実現するとともに動的再構成の進化の基盤を構築する．

本研究では，仕様はサービスの持つ機能に特徴付けられるという前提にたつ．サービス仕様としては，サービスが持つオペレーションの仕様とオペレーションの実行順序を規定するサービスの仕様があると考えた．サービス仕様を状態遷移機械を用いて定義することで，仕様記述の軽量化と，代替可能性の推論の単純化を目指す．

本研究では，サービス仕様を状態遷移機械を用いて形式的に定義する方法を提案した．代替可能性の確認を可能とするために，オペレーションの仕様とサービスの仕様について，それぞれの代替可能な条件を定義した．定義した条件に基づいてサービスの代替可能性を確認する方法を提案した．代替サービス検索のための枠組みを実現するためのアーキテクチャを提案した．事例を用いてサービスの代替可能性を確認する例を提示した．事例には同様の機能を持つことから，一般に公開されている SNS 系のサービスを用いた．

本研究の成果として，代替可能なサービスの条件の定義を明確にした．精度の高いサービス検索の方法を提案し，信頼性の高い SoPL を実現するとともに，動的再構成の進化の基盤を構築した．

2　The Capability Matching of Web Services

Gao ら [5] は，サービスの機能記述言語を提案し，記述言語における仕様の一致を定義している．その際に，機能記述言語の主要な目的について分析している．主要な目的とその説明を次に示す．

軽量： 読み書き・理解が容易

表現力： 具体的な振舞いや実行結果の情報を表現可能

型と制約を表現する能力： 一致の推論のための型や制約の十分な表現能力が必要

Gao らの提案する機能記述言語は，サービスの機能をアトミックなものとして定義している．我々は経験と分析から，サービスは複数のオペレーションを持ち，それらの実行順序に依存関係があることが分かっている．本研究では，このようなサービスの意味を考慮することで，精度の高いサービス検索を実現することを目指す．

3　代替可能なサービス

本章では，代替可能なサービスを定義する．図 1 に，代替可能なサービス仕様の条件の概略を示す．

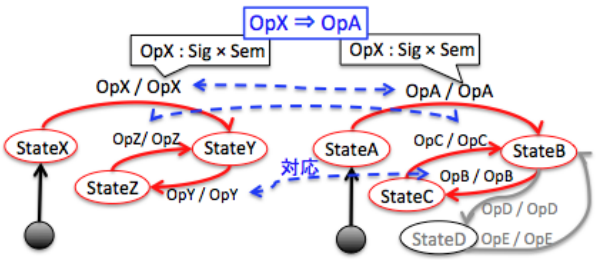

図 1　代替可能なサービス仕様の条件の概略

状態遷移機械を用いたサービス仕様の形式的な記述方法と，代替可能な仕様であるための条件を定義する．

3.1　サービス仕様の定義

本節では，サービス仕様の定義方法を提案する．図 2 に，サービス仕様の定義の概略を示す．サービスの仕様は，状態遷移機械を用いて定義する．オペレーションの仕様は，Signature と Semantics の観点から定義する．

サービスの仕様の定義方法を定義する．一般にコンポーネントの仕様は，Z や VDM 等の仕様記述言語を用いて形式的に定義される．本研究では，サービスの仕様はミーリ型の状態遷移機械を用いて定義するものとする．それにより図式表現で記述することが可能となり，定義方法の軽量化を図ることができる．代替可能性を確認する際に，状態遷移機械の同型問題ととらえることが可能となる．

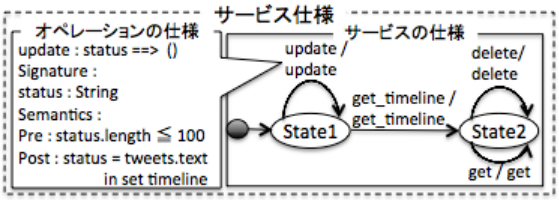

図 2 サービス仕様の定義の概略

サービスの仕様の形式的な定義方法を示す．サービス X の状態遷移機械は 次のように定義する．

サービス X MX = [S, E, O, Fs, Fo]

S は状態の集合，E はイベントの集合，O はオペレーションの仕様の集合を表す．Fs は状態とイベントから次の状態を返す関数を表し，Fo は状態とイベントから実行するオペレーションの仕様を返す関数を表す．

3.2 代替可能なサービス仕様の条件

本節では，3.1 章で定義したサービス仕様の定義方法をもとに，代替可能な仕様であるための条件を定義する．代替可能なサービス仕様の条件は，次の二点を満たすものとする．

- サービスの仕様が包含関係
- 対応遷移のオペレーションの仕様がマッチ

サービスの仕様の包含関係の定義とオペレーションの仕様のマッチの条件について順に説明する．

3.2.1 サービスの仕様の包含関係の定義

サービスの仕様の包含関係とはサービスの仕様を定義した状態遷移機械の形が準同型であることを指す．サービスの仕様の包含関係の定義では，状態名や遷移名の一致は考慮せず，状態遷移機械の形のみを対象とする．これにより，オペレーション名を除外したサービスの代替可能性の確認が可能となる．

Gao ら [5] の定義する状態遷移機械の準同型の定義をもとに，サービスの仕様の包含関係を定義する．二つの状態遷移機械 X, Y を考えた場合の Gao らの準同型の定義を示す．任意の s ∈ Sx や e ∈ Ex に対して，次の式を考える．式 (1),(2) における状態の対応関係の関数 g(Sx) → Sy が単射な場合を準同型と定義している．この定義では，イベント名が同じであることを前提としている．

$$g(Fsx(s,e)) = Fsy(g(s),e) \quad (1)$$

$$Fox(s,e) = Foy(g(s),e) \quad (2)$$

サービスの仕様の包含関係を形式的に定義する．本研究では，オペレーション名の一致は考慮しない．よってイベントの対応関係についても明確にしなければならない．上記の内容をもとに，サービスの仕様の包含関係の形式的な定義を行なう．任意の s ∈ Sx や e ∈ Ex に対して，次の式を考える．式 (3),(4) における状態の対応関係の関数 g(Sx) → Sy とイベントの対応関係の関数 h(Ex) → Ey が単射な場合を準同型と定義する．

$$g(Fsx(s,e)) = Fsy(g(s),h(e)) \quad (3)$$

$$Fox(s,e) = Foy(g(s),h(e)) \quad (4)$$

3.2.2 オペレーションの仕様のマッチの条件

オペレーションの仕様のマッチの条件は，Signature と Semantics が関連する．Signature は，入出力の型が一致しているものと定義する．Semantics は，事前・事後条件をもとに機能の同一性を確認する方法である，Zaremski ら [1] の Pre / Post Match を利用する．

Zaremski らは，関数の一致にはいくつかの種類があることを述べている．一致の種類の中で，代替可能であるものは Exact pre/post と Plug-in の二つである．Exact pre/post は Plug-in の特殊系なので Plug-in であれば代替可能であるといえる．Zaremski らの定義を受け入れて Semantics の代替可能な条件に利用する．二つの関数 Q, S が存在し，事前条件を pre, 事後条件を post と記述する場合の，Semantics の代替可能な条件の形式的な定義を式 (5) に示す．

$$(Qpre \Rightarrow Spre) \wedge (Spost \Rightarrow Qpost) \quad (5)$$

4 代替サービス検索のための枠組みの提案

本章では，3 章で定義した代替可能なサービスの定義をもとに，代替サービス検索のための枠組みを提案する．代替サービス検索を実行可能にするために，定義に基づいたサービスの代替可能性の確認方法を提案する．代替サービス検索を自動化するためのアーキテクチャを提案する．

4.1 サービスの代替可能性の確認方法

本節では，3.2 節で定義した代替可能なサービス仕様の条件の確認方法を提案する．確認をするための工程には，オペレーションの仕様のマッチの確認とサービスの仕様の包含関係の確認がある．各プロセスを行なうための確認方法を順に記述する．

4.1.1 オペレーションの仕様のマッチの確認方法

オペレーションの仕様の比較は，3.2.2 項で定義した条件の充足を確認する．条件判定の推論の自動化には，Prototype Verification System や Calculus of Constructions などの自動定理証明器が利用可能であると考える．これ以上は，本研究の範囲外なので言及しない．

4.1.2 サービスの仕様の包含関係の確認方法

3.2.1 項で定義したサービスの仕様の包含関係の定義をもとに，確認する方法を示す．サービスの仕様の包含関係の定義における，二つの関数 g，h が単射で定義可能であることを確認する．確認する際の前提として，オペレーションの比較は終えているものとする．前提から，イベントの対応関係の関数 h(Ex) → Ey が単射であることがいえる．

状態の対応関係の関数 g(Sx) → Sy が単射であることを確認する方法を示す．各オペレーションは，一回以上

状態遷移機械に表れる．全てのオペレーションに対して，事前状態と事後状態の対応関係を考えることで関数 g を定義することが可能となる．定義した関数 g が単射である場合に，サービスの仕様の包含関係が確認できる．

4.2　自動化のためのアーキテクチャの提案

本節では，動的に代替可能サービスを検索するための枠組みを実現するためのアーキテクチャを提案する．4.1 節で提案したサービスの代替可能性の確認方法をもとに，アーキテクチャを構築する．図 3 に，代替サービス検索の自動化のためのアーキテクチャを記述する．

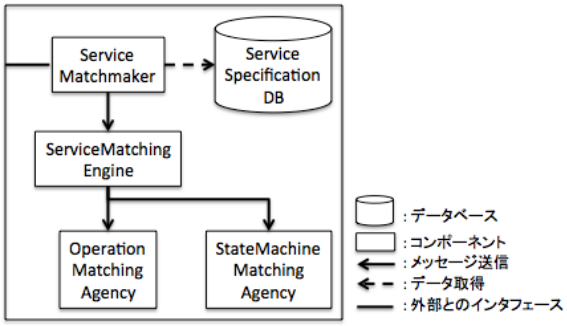

図 3　自動化のためのアーキテクチャ

各構成要素とその役割を次に記述する．ServiceMatchmaker はサービス仕様を入力として取得し，ServiceSpecificationDB で管理されているサービス仕様の中から代替可能なサービスの検索を行なう．ServiceMatchingEngine は，二つのサービス仕様をもとに代替可能性を確認する．OperationMatchingAgency は，サービスの持つオペレーション群の仕様のマッチを確認し，対応関係を明確にする．StateMachineMatchingAgency は，二つのサービスの仕様の包含関係を確認する．外部とのインタフェースを通じてサービス仕様を入力することで，代替サービスの検索を実行する．

5　事例検証

本章では，事例を用いてサービスの代替可能性を確認する例を提示する．事例には，一般に公開されている SNS 系のサービスである Twitter のつぶやきサービスを用いる．検索元のサービスの仕様とつぶやき投稿のオペレーションの仕様は図 2 に記述したものを利用する．図 4 に Twitter のサービスの仕様とつぶやき投稿のオペレーションの仕様を示す．代替可能性の確認の各プロセスの内容を順に示す．

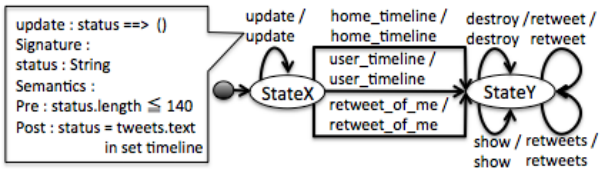

図 4　Twitter のつぶやきサービスの仕様

5.1　オペレーションの仕様のマッチの確認

検索元と Twitter のつぶやき投稿のオペレーションの仕様の比較を行なう．Signature は，一つの String 型の引数であることから一致していることが確認できる．事前条件は，検索元の仕様が文字数が 100 以下であることが条件であるのに対して，Twitter の仕様は 140 以下であれば良いので条件を満たす．事後条件は，同一の記述であることから条件を満たす．以上のことからオペレーションの仕様のマッチの条件を満たしていることが確認できた．

他のオペレーションについても同様に代替可能性の確認を行なった．その結果 get_timeline は user_timeline と，delete は destroy と，get は show と対応することが分かった．この対応関係をもとに，サービスの仕様の包含関係の確認を行なう．

5.2　サービスの仕様の包含関係の確認

検索元と Twitter のサービスの仕様の比較を行なう．オペレーションの対応関係をもとに各イベントの事前状態と事後状態の関係を考える．表 1 に各イベントの事前状態と事後状態の関係を整理した．

表 1　事前・事後状態の関係表

イベント	検索元の事前状態	Twitter の事前状態	検索元の事後状態	Twitter の事後状態
update	State1	StateX	State1	StateX
get_timeline	State1	StateX	State2	StateY
delete	State2	StateY	State2	StateY
get	State2	StateY	State2	StateY

表 1 から状態の対応関係を考察した結果，StateX は State1 と，StateY は State2 と対応することが分かる．よって状態の対応関係の関数 g を定義でき，サービスの仕様が包含関係であることが確認できた．以上のことから，Twitter サービスを代替サービスとして利用することが可能であることが確認できた．

6　考察

本章では，サービス仕様の定義方法の妥当性，代替可能なサービス仕様の条件の妥当性，動的再構成の進化の三つの観点から考察する．

6.1　サービス仕様の定義方法の妥当性

提案したサービス仕様の定義方法について考察する．Gao らの分析した主要な目的である軽量，表現力，型と制約を表現する能力の三つの観点から考察する．

軽量の観点から考察する．本研究では，サービスの仕様とオペレーションの仕様を明確に区別してそれぞれの形式的な定義方法を提案した．これにより，利用者やシステムが記述された仕様を容易に理解可能であるといえる．サービスの仕様は，状態遷移機械を用いて定義することを提案した．状態遷移機械を用いることで，図式表現で記述することが可能となり，開発者による読み書きが容易に行なうことが可能となった．

表現力の観点から考察する．サービスの仕様に状態遷移機械を用いることで，オペレーション間の依存関係を記述する．オペレーションの仕様の記述を事前・事後条件としたことで，抽象的に機能を記述する．この記述によって，各オペレーションの具体的な振舞いや実行結果が表現できる．二つの仕様を総括することで，サービスの具体的な振舞いや実行結果を記述することができる．

型と制約を表現する能力の観点から考察する．オペレーションの仕様の定義によって，各オペレーションの型と制約について記述できる．サービスの仕様の定義によって，オペレーションの実行順序の制約について記述できる．サービスの仕様の比較を状態遷移機械の同型問題と捉えることができ，推論が単純化できた．

Gao らの分析した主要な目的の面でサービス仕様の定義方法は妥当であるといえる．今後の課題として，事例検証を重ねることによる妥当性の確認が必要である．

6.2 代替可能なサービス仕様の条件の妥当性

本研究では代替可能なサービス仕様の条件を定義した．条件としては，サービスの仕様が包含関係であり，対応遷移のオペレーションの仕様がマッチであるものと定義した．事例検証として，一般的なつぶやきサービスの仕様を考え，Twitter のつぶやきサービスとの比較を行なった．その他に mixi のつぶやきサービスとの比較をおこなった．

オペレーションの仕様のマッチの条件について考察する．つぶやき投稿のオペレーションは，同様の機能を持つので各サービス間で互換性があると仮定する．実際に互換性があるものに対して適用可能であればマッチの条件が妥当であるといえる．事例検証の結果，二つの例に対してオペレーションが代替可能な条件を充足していることが確認できた．よって，オペレーションの仕様のマッチの条件は妥当であるといえる．

サービスの仕様の包含関係の定義について考察する．本研究ではサービスの仕様を記述した二つの状態遷移機械が準同型であれば，サービスの仕様が包含関係であると定義した．状態遷移機械は，サービスの持つオペレーションの実行順序の依存関係を表現している．状態遷移機械が準同型であることは，オペレーションとその実行順序の依存関係が対応していることを意味している．よって，サービスの仕様の包含関係を状態遷移機械の準同型として定義することは妥当であるといえる．以上のことから代替可能なサービス仕様の条件は妥当であるといえる．

6.3 オペレーションの仕様のマッチの応用

オペレーションの仕様のマッチの応用について考察する．本研究では，オペレーションの Signature の比較は，同一のものを対象とした．しかし，入出力の情報量が同じであれば，代替可能である場合が考えられる．この場合，小林らの提案するタグによる引数の意味付け [6] が利用可能であると考えた．

小林らは，タグを用いて各引数に意味付けを行ない，タグの名前の一致によって Signature の意味の同一性を確認する手法を提案した．この手法を利用することによって，オペレーションの入出力の違いを考慮したオペレーションのマッチの確認が可能となる．

6.4 動的再構成の進化

動的再構成の進化について考察する．動的再構成では，事前定義をもとに動的にコンポーネントを入れ替える．コンポーネントがあらかじめ用意されていることが前提となるので，実質的には新機能の追加を行なうことはできない．本研究では，サービス仕様をもとに代替可能なサービスを検索する枠組みを提案した．アプリケーションに，環境情報を分析してサービス仕様を生成する仕組みを用意することを考える．動的にサービス仕様を生成することが可能となれば，動的に新機能を追加することが可能となる．これは PLSE におけるドメインエンジニアリングを動的に行なうことが可能であることを意味する．よって，動的再構成の進化の基盤を構築することができたといえる．

7 おわりに

本研究の目的は，サービスの意味を考慮した代替サービス検索のための枠組みを提案することである．そのために代替可能なサービス仕様の条件と実行可能な確認方法の提案を行なった．これにより精度の高いサービス検索を実現し，より信頼性の高い SoPL の実現が可能となった．動的再構成の進化の基盤を構築することができた．今後の課題として，事例検証を重ねることによるサービス仕様の定義方法の妥当性の確認がある．代替サービス検索のための枠組みの実装を行なうことがある．枠組みを利用した動的再構成可能な範囲の拡大を行なうことがある．

参考文献

[1] A. M. Zaremski, and J. M. Wing, “ Specification matching of software components, ” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 6, pp 333-369, 1997.

[2] G. Kotonya, and D. Robinson,“ Engineering Service-Based Dynamic Software Product Lines, ”*Computer*, vol. 45, no. 10, pp. 49-55, 2012.

[3] J. Lee and K. C. Kang, “ A Feature-Oriented Approach for Developing Dynamically Reconfigurable Products in Product Line Engineering, ” *10th International Software Product Line Conference (SPLC 2006)*, pp. 131-140, 2006.

[4] S. Hallsteinsen, M. Hinchey, S. Park, and K. Schmid, “ Dynamic Software Product Lines, ” *Computer*, pp. 93-95, 2008.

[5] X. Gao, J. Yang, M. P. Papazoglou, ”The Capability Matching of Web Services, ” *Multimedia Software Engineering*, pp 56-63, 2002.

[6] 小林利誌, 富塚祐太, “ SOA アプリケーションプラットフォームのプロダクトライン化に関する研究ー再利用コンポーネントの定義，設計，実装ー,“ 情報理工学部ソフトウェア工学科，2014.