クラウドサービスにおけるバリエーションモデルの提案

2007MI102　勝　崇　　2007MI118　黒野　望

指導教員　青山　幹雄

1. はじめに

近年クラウドサービスの利用が増加している．クラウドサービスでは，利用者(テナント)毎の差異を構成メタデータで表現しており，メタデータ API を用いて構成メタデータの内容が変更されることがある．しかし，構成メタデータの一般的なモデルは存在しない．

本稿では，メタデータ API より特定できる構成メタデータの変動のモデルをバリエーションモデルとして提案する．

2. クラウドサービス

クラウドサービスにおいてテナント毎の差異を定義するために次の技術がある．

2.1. シングルインスタンス/マルチテナント

クラウドサービスの運用形態の一つに単一のインスタンスを複数のテナントが利用するシングルインスタンス/マルチテナント方式がある．

2.2. 構成メタデータ

構成メタデータは，クラウドサービスにおけるテナント情報を管理するプロファイルデータである．アプリケーションにテナント毎の構成メタデータを付加することで，シングルインスタンス/マルチテナント方式を実現する．

2.3. メタデータ API

テナントが構成メタデータを編集するインタフェースとしてメタデータ API がある．メタデータ API を使用することにより，外部アプリケーションから SaaS(Software as a Service)の構成メタデータへのアクセスなど他アプリケーションから SaaS の構成メタデータへのアクセスが可能になる(図 1).

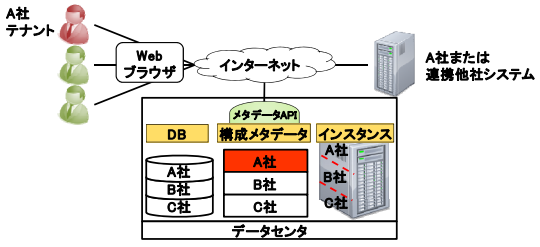

図 1　構成メタデータへのアクセス方法

2.4. CRUD-based コール

メタデータ API のコール方式に Create, Read, Update, Delete の基本的な機能を命令する CRUD-based コールが存在する．CRUD-based コールは実行するコールを明示し，構成メタデータ内の情報を直接編集するコールの中で，最も単純なコールである．

3. 可変性

3.1. 可変性の概念

可変性とは，PLSE (Product Line Software Engineering)においてソフトウェアの変動する可能性である．

3.2. モデル化手法

可変性を表すモデル化手法としてユースケースモデル，フィーチャモデル，メッセージシーケンスモデル，クラス図，OVM (Orthogonal Variability Model) などがある．本稿では OVM とクラス図を用いて可変性をモデル化する．

3.3. OVM

OVM とは，可変性を表現するモデル化手法である．OVM の可変性表現は共通部を省略し，可変部のみを記述することができる．OVM で作成したモデルは，クラス図などの他のソフトウェア開発モデルと関連づけができ，可変性の横断的なビューを提供できる．

4. 研究課題

テナントの操作できる範囲の構成メタデータの変動構造を示すことが課題である．サービスプロバイダが異なる企業にサービスを提供する際に，構成メタデータはテナント毎に異なる．しかし，一般的な構造を示すモデルがないため構成メタデータの差異が把握しづらい．そのため，テナントがアプリケーションにどの程度バリエーションがあるのか把握しづらくなっている．

テナントの操作できる範囲の構成メタデータの変動構造を示すことで，テナント毎におきるバリエーションの範囲を明確にすることが研究課題である．

5. 関連研究

5.1. SaaS における可変性モデルを用いた開発[2]

SaaS アプリケーションの可変性モデル開発を提案している．しかし，表現の抽象度が高く SaaS アプリケーションの作成におけるテナント毎の違いを吸収している構成メタデータに関する可変性は定義していない．

5.2. 可変性メタモデル

可変性メタモデルとは，可変性における所定の問題領域でのモデリングに適用可能な規則や制限を示したモデル

である. 本稿における可変性の規則として文献[3]の可変性メタモデルを用いている.

6. アプローチ

本稿では, メタデータ API に着目し, メタデータ API の操作により構成メタデータの変動部分を表現することで, 構成メタデータの構造を示す.

メタデータ API は, 構成メタデータのデータ構造に影響を与える. よって, メタデータ API に可変性の概念を導入し, メタデータ API によって操作できる範囲を表すことで構成メタデータのデータ構造を推定する.

7. 提案内容

本稿では, メタデータ API より構成メタデータのバリエーションモデルを提案する. バリエーションモデルによりテナント操作による構成メタデータの変動の範囲を示す.

7.1. 提案内容の枠組み

バリエーションモデルの作成プロセスを図 2 に示す.

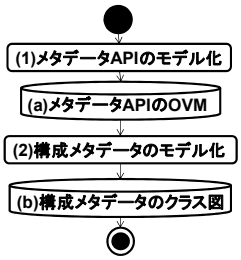

図 2 バリエーションモデルの作成プロセス

(1) メタデータ API のモデル化:メタデータ API の操作により変動する部分に OVM を用いてモデル化する.
(2) 構成メタデータのモデル化:メタデータ API で作成した OVM を基に構成メタデータで変動が起きる場所の構造をモデル化する.
(a) メタデータ API の OVM:OVM でメタデータの可変性を表現したモデル
(b) 構成メタデータのクラス図:構成メタデータで変動部の構造を表現したクラス図

7.2. 前提条件

本稿で提案するモデルの前提条件を表 1 に示す.

表 1 前提条件

前提条件項目	要素
クラウドの運用形態	シングルインスタンス/マルチテナント方式
サービスの種類	CRMサービス
メタデータAPIの種類	CRUD-basedコール
対象のアプリケーション	顧客管理, ワークフロー

7.3. メタデータ API の OVM 作成

メタデータ API で作成するアプリケーションのモデルとして, 顧客管理とワークフローのモデルを作成する. メタデータ API の要素を MVC (Model View Control) に基づいて規定する. 構成メタデータの一般的な要素を定義したいため一般的なソフトウェアアーキテクチャパターンである MVC に基づいてメタデータ API の要素を規定する.

モデル化手法としてメタデータ API での操作による変動部分を表すため OVM を用いて表現する(図 3). 可変性のメタモデルとして文献[3]を用いる.

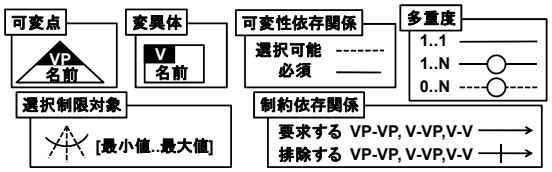

図 3 可変性における OVM での図式表記法

顧客管理はテナント毎に差異を持たせるため, オブジェクトで新規顧客登録や売上データ作成などの機能を定義し, テナント毎に使用するオブジェクトを変更する. 以降, このオブジェクトを機能オブジェクトと呼ぶ. 機能オブジェクトのメタデータ API の要素を MVC に基づいて定義する.

Model に相当する要素を識別子とデータ項目とする. 識別子は機能オブジェクトを一意に識別するデータである. データ項目は機能オブジェクトの中で扱うデータの範囲を決定する要素である.

View に相当する要素を表示ラベルとボタンと画面構成とする. 表示ラベルは, 機能オブジェクトのテナントに見える名前である. ボタンは, データ項目の要素を編集や新規作成させるなどの一般的なボタンの名前の指定である. ボタンの機能自体は一般的なボタンであるため, 機能オブジェクトであらかじめ生成されると考えた. 画面構成はデータ項目などの表示項目の配置を決める.

Control に相当する要素を画面構成と状態とする. 画面構成は View にも存在したが, データ項目などを指定して表示させるため Control の役割も担う. 状態は, 機能オブジェクトが開発中であるのか, すでにリリースしたのかの状態を指定するために必要である.

定義した要素に OVM を用いてモデル化し, 図 4 に示す. 顧客管理の他の要素やワークフローに関してもモデルの作成を行ったが, 今回は割愛する.

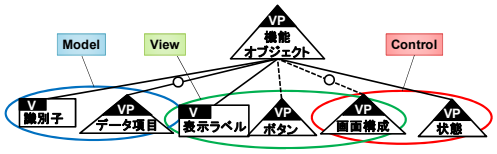

図 4 顧客管理におけるメタデータ API の OVM

7.4. 構成メタデータの基本構造

構成メタデータの基本構造を文献[5]の基本要素を基に作成した(図 5).

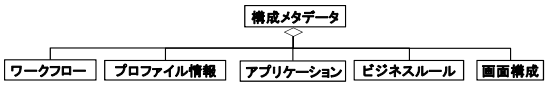

図 5 構成メタデータの基本構造

木構造の最上位階層の要素として，ワークフロー，企業ごとのプロファイル情報，顧客管理などのアプリケーション，企業毎のビジネスルールがあると考えられる．

構成メタデータのクラス図を作成する際に，図5の基本構造を基に顧客管理のメタデータ API 情報を拡張してクラス図を作成する．メタデータ API より判断した要素の位置には，矢印を用いて対応付けする．

7.5. バリエーションモデル

機能オブジェクトのメタデータ API より構成メタデータのモデルを図 6 に示す．

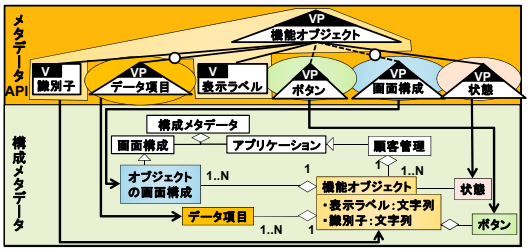

図 6 構成メタデータのモデル

顧客管理アプリケーションは機能部分である機能オブジェクトを複数組み合わせてアプリケーションを構成するため，機能オブジェクトは集約関係になる．

機能オブジェクトのメタデータ API より編集できる構成メタデータの要素は以下の五つの要素になる．

(1) 識別子，表示ラベル：機能オブジェクトの要素であるため，機能オブジェクト内に記述する．

(2) データ項目：機能オブジェクトの構成要素の一部であると考えられるため，集約関係にある．

(3) ボタン：あらかじめ決められた要素であり，名前が変更可能なため，参照して使用していると考えられる．よって，ボタンは機能オブジェクトと集約関係にある．

(4) 画面構成：機能オブジェクトの構成要素の一部であると考えられるため，集約関係にある．また画面構成はオブジェクトの画面表示を担うため，画面構成から継承関係にあると考えられる．ここで機能オブジェクトの画面構成と木構造の最上位にある画面構成は異なるため，機能オブジェクトの画面構成は「オブジェクトの画面構成」とする．

(5) 状態：機能オブジェクト自体に要素として追加しており，メタデータで管理はしていないと考えるので機能オブジェクトの要素になる．

今回表記した要素は一部分で，機能オブジェクト内のデータ項目や画面構成などの要素にもそれぞれバリエーションモデルを作成した．

8. Salesforce.com への適用

提案したモデルの要素を文献[4]の Salesforce.com のメタデータ API の要素と比較し，モデルの要素の妥当性を検証した．要素の比較をするため文献[1]の出張申請アプリケーションの機能を抽出し CRUD 分析を行った．

8.1. Salesforce.com のアプリケーション例

文献[1]から出張申請アプリケーションを作成した．開発プロセスを図 7 に示し，それぞれのプロセスの説明を表 2 に示す．出張申請アプリケーションへの申請ワークフローの追加については分析を行ったが，ここでは割愛する．

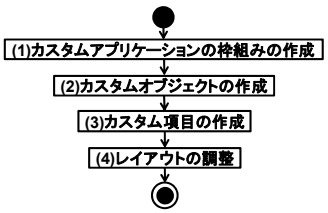

図 7 出張申請アプリケーション開発プロセス

表 2 各プロセスの詳細

開発プロセス	説明
カスタムアプリケーション枠組みの作成	出張申請アプリケーションの枠組みの作成． 呼び出すオブジェクトの指定や配置を設定．
カスタムオブジェクトの作成	オブジェクトの作成．MVCのModelに相当する．
カスタム項目の作成	カスタムオブジェクトに項目を追加． 追加する項目のデータ型や項目名を設定．
レイアウトの調整	項目の配置を調整． 項目の配置変更でアプリケーションに表示される項目を設定．

8.2. 適用例からの機能抽出

図 7 の開発プロセスより以下のように機能抽出を行う．

(1) オブジェクト作成機能の抽出

(2) データ項目追加機能の抽出

(3) データ項目修正機能の抽出

(4) データ項目参照機能の抽出

(5) 画面編集機能の抽出

8.3. CRUD 分析

モデルの要素の妥当性を検証するために CRUD 分析を行う．CRUD 分析に必要な Salesforce.com のメタデータ API 要素を表 3 に示す．

表 3 Salesforce.com のメタデータ API の要素

要素	説明
actionOverrides	オブジェクトで使用するボタンを編集
deploymentStatus	デプロイ状態
fields	データ項目
nameField	カスタムオブジェクトが持つべきデータ項目
fullName	カスタムオブジェクトの識別子

モデルの要素に対応するメタデータ API の要素をまとめ，CRUD 分析を行った(表 4)．

表 4 各要素に対する CRUD 分析

		アプリケーション作成				
		オブジェクト作成	データ項目作成	データ項目修正	データ項目参照	画面構成
機能オブジェクト Custom Object	識別子	C	R	R		R
	fullName	C	R	R		R
	データ項目	C	C	U	R	R
	nameField	C		U	R	R
	fields		C	U	R	R
	ボタン					R
	actionOverrides					R
	画面構成	C				U
	状態	C				
	deploymentStatus	C				

オブジェクト作成は識別子，データ項目，画面構成，状態が Create される．データ項目作成は識別子が Read され，データ項目が Create される．データ項目修正では識別子が Read され，データ項目が Update される．データ項目参照は，データ項目が Read される．画面構成は画面構成が Update され，識別子，データ項目，ボタンが Read される．

9. 提案モデルの評価

9.1. CRUD 分析に基づいた評価

分析結果よりモデルの識別子，ボタン，状態においてはメタデータ API の要素と CRUD が一致した．データ項目では nameField と fields の要素を表現している．画面構成は Custom Object で表現していない要素であるが，他の要素で表現しているため必要な要素と推定する．

よって，一部差異はあるが提案したモデルで CRM サービスにおけるアプリケーション作成に必要な要素は定義できたと考えられる．

9.2. バリエーション表現の評価

(1) 構成メタデータの構造：定義した構成メタデータの基本構造のうち，提案したモデルでは機能オブジェクトのバリエーションを定義できた．機能オブジェクトでは構成メタデータのアプリケーション作成を表現している．構成メタデータにおけるテナント毎の差異を表現でき，CRM サービスの必要な要素を満たしている．そのため，モデルによって SaaS 間の連携などで必要な構成メタデータの構造を知ることができる．

(2) MVC の意義：モデルのバリエーションの設定は MVC に基づいて要素を Model, View, Control に分けて作成した．MVC で要素定義したため，主要なモデル要素を機能，入力，出力の三つに分離できた．入力と出力の両方に属し，V と C を明確に分離できない要素も存在した．しかし，MVC によってメタデータ API の要素を定義したため，可変点や変異体が MVC に分離された．そのため，メタデータ API のモデルによる構成メタデータの要素の特定が容易になった．

(3) 構成メタデータのバリエーション範囲：メタデータ API の CRUD-based コールの操作可能範囲で表現した．CRUD-based コールでは，編集要素を明確に指定するため，構成メタデータの編集可能要素をモデルで表現できた．しかし，構成メタデータのメタデータ API から操作できない要素は，要素間の関係や要素の情報が確認できないため表現できない．メタデータ API からモデル作成する限界であると考えられる．

9.3. バリエーションの表現方法の評価

OVM を用いることにより，以下の 2 点を評価する．

(1) クラス図との対応付け：メタデータ API の可変性を表現できたため，メタデータ API から構成メタデータに影響を及ぼす箇所に対応付けできた．

(2) 共通部と可変部：オブジェクトを作る際に必要な要素の識別子やデータ項目が共通部と表現できた．また，要素内に可変要素を含むデータ項目を可変部と表現できた．このように共通部と可変部を特定できた．しかし，データ項目のような共通部であり，可変部でもある要素があるため，共通部と可変部を明確に分離はできなかった．

10. 今後の課題

今回提案したモデルは，CRUD-based コールのみに絞ったバリエーションモデルである．そのため，他のコール方式は本稿のモデルでは考慮していない．よって，他のコール方式も考慮にいれる必要がある．

また，行った分析がモデルの要素に対する CRUD 分析のみで，モデルのバリエーションに対する分析や妥当性検証などを行っていない．他の分析も行う必要がある．

11. まとめ

本稿では，クラウドサービスにおけるテナント毎の差異をモデル化するために，構成メタデータにおけるバリエーションモデルを提案した．

MVC に基づいてメタデータ API の要素を作成し，メタデータ API が影響を与える構成メタデータの部分を定義し，構成メタデータのテナント毎に起きる可変部分を示した．そして，作成したバリエーションモデルを Salesforce.com のアプリケーションと比較し，SaaS の CRM サービスとして必要な要素を満たしていることを示した．

参考文献

[1] 阿部 友暁, 小堀 貴生, 小林 洋介, Force.com クラウドアプリケーション開発, インプレスジャパン, 2010.

[2] R. Mietzner, et al., Variability Modeling to Support Customization and Deployment of Multi Tenant-Aware Software as a Service Application, Proc. of PESOS, May 2009, pp. 18-25.

[3] K. Pohl, et al., Software Product Line Engineering: Foundations, Principles and Techniques, Springer, 2005.

[4] Salesforce.com, http://www.salesforce.com/jp

[5] 山谷 正己, 図解でわかる SaaS の全て, オーム社, 2009.