メタデータ駆動クラウドサービスアーキテクチャの可変性制約依存関係パターンの提案と評価

M2011MM045　黒野　望

指導教員　青山　幹雄

1. はじめに

クラウドサービスでは，マルチテナントが支援されている．マルチテナントとはリソースを共有しながら，複数のテナントにサービスを提供する仕組みである．マルチテナントを実現するアーキテクチャとしてメタデータ駆動アーキテクチャが注目されているが，その設計手法が確立されていない．

本研究では，メタデータとアプリケーション間の連動する可変性に着目し，メタデータ駆動型アーキテクチャにおける可変性の制約依存関係をパターン化する．

2. メタデータ駆動アーキテクチャ

メタデータ駆動アーキテクチャは，メタデータによりインスタンスを制御するアーキテクチャスタイル[5]である(図 1)．メタデータはテナント情報を定義するプロファイルデータである．メタデータを基に動的なインスタンス制御を行うことで，テナント毎の多様な要求へ柔軟に対応する．

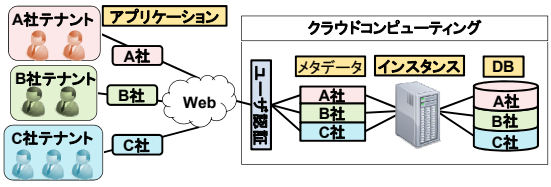


図 1 メタデータ駆動アーキテクチャの構造

3. 研究の課題

メタデータ駆動アーキテクチャでは，メタデータを基にインスタンスを制御している．本研究では，そのアーキテクチャを実現するための課題として次の 2 点を対象とする．

3.1. メタデータの変動箇所が不明確

メタデータのデータ構造の違いがテナント毎のカスタマイズ性に影響を与える．しかし，テナント毎に発生する変動箇所が不明確であり，テナント毎の変動箇所を管理，表現することは困難である．

3.2. メタデータとアプリケーション間の依存関係が不明確

メタデータの変更によってインスタンスを制御し，アプリケーションを生成している．その際，メタデータ変更によるアプリケーションへの影響範囲が不明確で，依存関係も明確にされていない．そのため，メタレベルの開発からベースレベルの制御を設計することが困難である．

4. 関連研究

4.1. OVM(Orthogonal Variability Model)

OVM は，プロダクトラインの可変性を表現するモデルである[3]．OVM では，機能の表現を省略し，可変性のみ表現できる．

4.2. マルチテナントの可変性モデリング

マルチテナントの可変性モデルが提案されている[2]．可変性を OVM を用いてモデル化しているが，表現抽象度が高く，マルチテナントの制御を適切に表現できていない．

5. アプローチ

メタデータはソフトウェアプロダクトラインの可変性を表現できる．ソフトウェアプロダクトライン手法の中で次の 3 点に着目し，制約依存関係をパターン化する(図 2)．

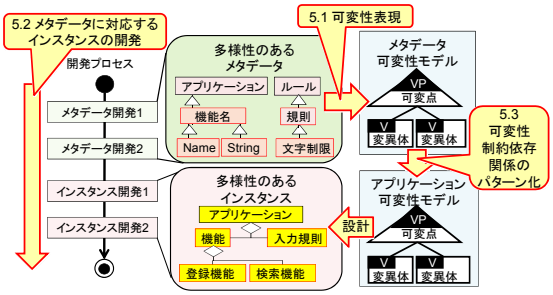


図 2 メタデータ駆動における可変性記述

5.1. 可変性記述

メタデータとアプリケーションは，プロダクトライン手法を適用できることから，テナント毎に発生するメタデータとアプリケーションの変更箇所をそれぞれ可変性として表現する．可変性で表現することで，「何が変異するのか」と「どのように変異していくのか」が表現できるようになりメタデータ駆動アーキテクチャにおける変更箇所の管理が容易になる．

5.2. メタデータに対応するインスタンスの開発

メタデータ駆動アーキテクチャの開発方法として，テナント毎の差異を表現するメタデータよりアプリケーションを生成している．そのため，メタデータの開発からそのメタデータの可変構造に対応できるように，コア部分であるインスタンスの可変性を考慮する開発方法が必要である．

5.3. 可変性制約依存関係のパターン化

メタデータの可変性がアプリケーションの可変性に影響

与えていることに着目して，メタデータの可変性とアプリケーションの可変性の制約依存関係をパターン化する．「どのように変異していくのか」と「どの人を目的とするときにその変異がおこるか」が明確になり，メタデータとアプリケーション間の協調関係が明確になる．よって，メタデータ駆動アーキテクチャのモデル化が容易になる．

6. 可変性制約依存関係パターンの提案

可変性制約依存関係パターンを作成する手順として，メタデータとアプリケーションの可変箇所を MVC(Model View Controller)に基づき分類する．次に，メタデータ駆動アーキテクチャにおける可変性依存関係を表現できるように，OVM の表現を拡張する．最後に，関心事毎の可変性制約依存関係をパターン化し，可変性制約依存関係パターンを提案する．

6.1. MVC に基づく可変性の分類

メタデータ駆動アーキテクチャでは，要求の可変性がメタデータの可変性に影響し，メタデータの可変性がアプリケーションの可変性に影響する(図 3)．本研究では，各レベルの可変性は Model，View，Controller に関心事を分割できるため，可変性を MVC より分類する．

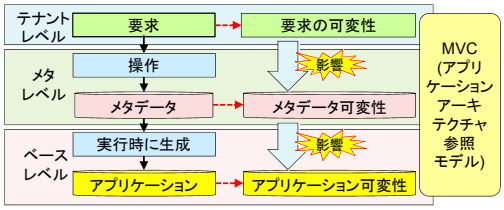


図 3 メタデータとアプリケーションの可変性の関係

6.2. OVM の制約依存関係の表現拡張

OVM を用いてメタデータ駆動アーキテクチャにおける可変性とその制約依存関係を表現する．従来の OVM では，要求と排除のみで，メタデータ駆動アーキテクチャにおける可変性の制約依存関係を適切に表現できない．本研究では，OVM の制約依存関係の表現方法を拡張する．

6.2.1. 制約依存関係の分類

メタデータ駆動アーキテクチャの制約依存関係は，次の 3 つに分類できる．

(1) メタデータ間の制約依存関係(M-M)

メタデータの可変性が他のメタデータの可変性を制限もしくは依存する関係を示す．

(2) メタデータとアプリケーション間の制約依存関係(M-A)

メタデータの可変性が生成されるアプリケーションの可変性を制限もしくは依存する関係を示す．

(3) アプリケーション間の制約依存関係(A-A)

アプリケーションの可変性が他のアプリケーションの可変性を制限もしくは依存する関係を示す．

6.2.2. 拡張制約依存関係

本研究ではメタデータ間の制約依存関係，メタデータからアプリケーションへの制約依存関係，アプリケーション間の制約依存関係は，それぞれデザインパターンの目的である構造，生成，振舞いに対応すると考える．それらの視点で制約依存関係の拡張を行った(図 4)．

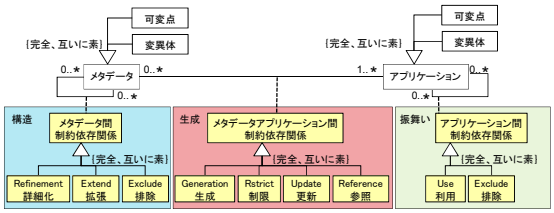


図 4 可変性の制約依存関係モデル

Refinement，Generation，Restrict，Update，Reference の 5 つの制約依存関係を拡張した．Refinement では，メタデータの可変部により，親子関係が構成されるため，依存関係を拡張した．Generation，Restrict，Update，Reference では，メタデータの変更により，ユーザが利用できるアプリケーションの変異体を生成するため，制約依存関係を拡張した．

6.3. 可変性制約依存関係のパターン化

メタデータとアプリケーションの可変性を MVC より分類したので，M(Model)，V(View)，C(Controller)で制約依存関係をパターン化した(図 5)．

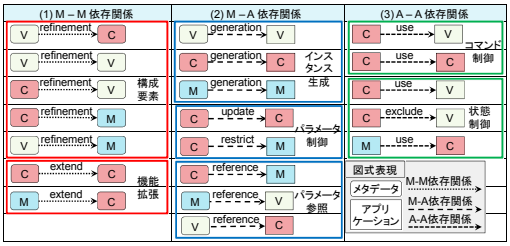


図 5 可変性制約依存関係パターン

制約依存関係パターンを作成する際に，M-M を構造の視点で，M-A を生成の視点で，A-A を振舞いの視点で制約依存関係をパターン化した．

(1) M-M 制約依存関係(構造)

構造の視点から構成要素と機能拡張として依存関係をパターン化した．構成要素では，メタデータの変更によって子要素として持つべき要素が変更されるため，依存関係でパターン化した．機能拡張では，メタデータの変更によって他の機能を拡張する場合に，依存関係でパターン化する．

(2) M-A 制約依存関係(生成)

生成の視点からインスタンス生成，パラメータ制御，パラメータ参照として依存関係をパターン化した．インスタンス生成では，メタデータの変更によってユーザが利用するアプリケーションの可変性の変異体が決定するため，依存関係としてパターン化した．パラメータ制御では，メタデータ

の変更によって，他のインスタンスを制御するインスタンスを生成するため，パラメータ制御の制約依存関係をパターン化した．パラメータ参照では，メタデータの入力値によって，アプリケーションとして新しく変異体が追加されるため，依存関係をパターン化した．

(3) A-A 制約依存関係(振舞い)

振舞いの視点からコマンド制御と状態制御の依存関係をパターン化した．コマンド制御では，ユーザがアプリケーションの変異体を選択した際に，他の可変部を要求する場合を考慮し，パターン化した．状態制御では，ユーザの利用状態により，可変部を要求や制限するため，制約依存関係としてパターン化した．

以上より生成されたパターンの一例として構成要素のパターンを図 6 に示す．パターンとして意図，構造と利用例を規定し，パターン化した．

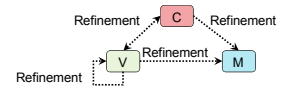
パターン名	構成要素
パターンの概要	アプリケーション自体の構成要素として依存関係を持つためのパターンである．
パターン化の意図	メタデータでは，アプリケーションの枠組みや構成を担っていると考え，メタデータ間の制約依存関係では，アプリケーションの構成要素の視点で可変性の依存関係がパターン化した．
パターンの構造	
例	・VからV，VからC，VからM，VからMのRefinement依存関係： レイアウトを構成するために，他の制御内容やデータを出力する場合 ・VからVのみのRefinment依存関係： ビューの変更の際に他のビューを要素として埋め込む場合

図 6 構成要素パターン

6.4. 提案パターンの適用方法

メタデータ開発プロセスに提案したパターンを適用する際に，構造，生成，振舞いの順に段階的にモデルを詳細化する．段階的に詳細化することで，可変性の依存関係を示す際に，過不足を減らすことが可能になる．本研究の提案パターンの適用プロセスを図 7 に示す．

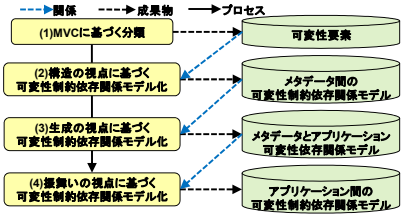


図 7 提案パターンの適用方法

(1) MVC に基づく分類

アプリケーション開発プロセスにおける各プロセスで作成したい機能を MVC に基づいて可変性を分類する．

(2) 構造の視点に基づく制約依存関係モデル化

各プロセス間の関係より，構造に影響を与える可変性制約依存関係をモデル化する．

(3) 生成の視点に基づく制約依存関係モデル化

各プロセスにおいてメタデータよりアプリケーションの可変性を定義する可変性依存関係をモデル化する．

(4) 振舞いの視点に基づく制約依存関係モデル化

生成されたアプリケーションがユーザの状態や入力によって，アプリケーションの可変性に影響する可変性制約依存関係をモデル化する．

7. 例題へのパターンの適用

7.1. 適用対象と目的

Force.com[4]上でのアンケート管理アプリケーション[1]の作成プロセス(図8)にパターンを適用することで，メタデータ作成プロセスからテナント毎の可変性の表記可能性，可変性による影響範囲の導出可能性を検証する．

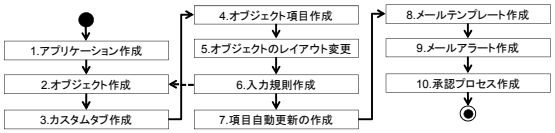


図 8 アンケート管理アプリケーション作成プロセス

7.2. 適用方法

6.4 節で述べたプロセスを基に可変性制約依存関係をモデル化する．アンケート管理アプリケーション作成プロセスから可変性を抽出し，可変性をMVCに分類する．分類した可変性を構造，生成，振舞いの順に制約依存関係をモデル化する．

7.3. 構造の視点での制約依存関係

構造の視点より制約依存関係をモデル化した(図 9)．アプリケーションはタブを保持する．また，タブの選択よりタブはオブジェクトを持つように，構造の依存関係を示した．

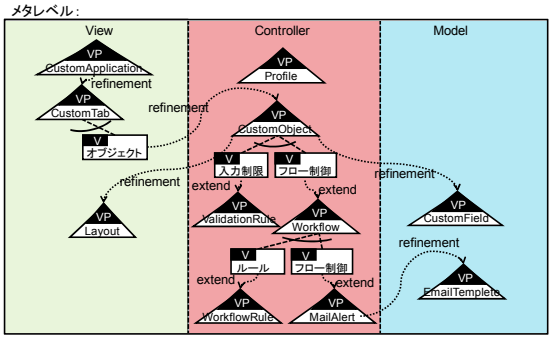


図 9 構造の視点における制約依存関係

7.4. 生成の視点での制約依存関係

生成の視点より制約依存関係をモデル化した(図 10)．構造からアプリケーションを生成することで，アプリケーションにおいてユーザが利用できる機能やインスタンスの可変性を生成する依存関係を示した．

7.5. 振舞いの視点での制約依存関係

振舞いの視点より制約依存関係をモデル化した(図 11)．生成されたアプリケーションの可変性より，ユーザがアプリケーションを利用した際に，利用される機能やレイアウトの制約依存関係を示した．

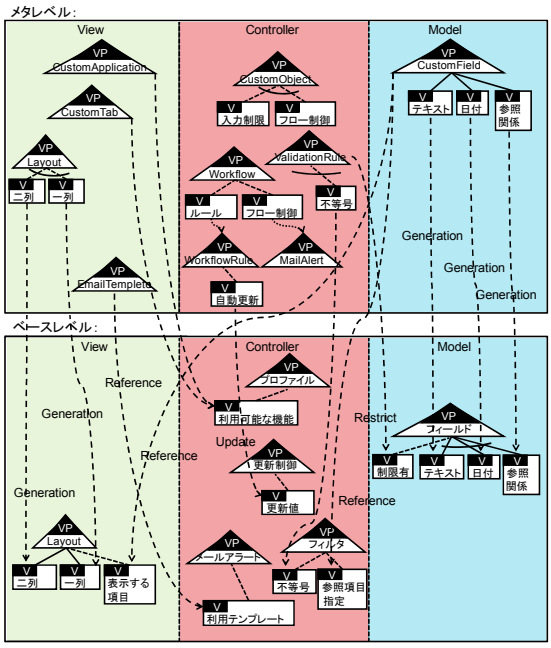


図 10 生成の視点における制約依存関係

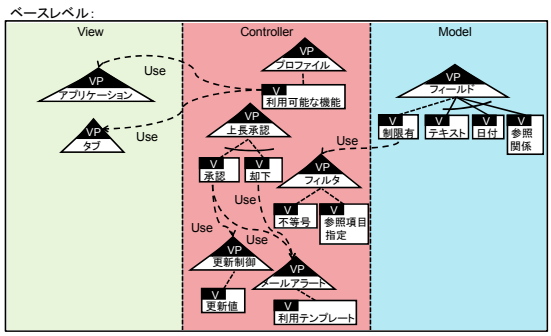


図 11 振舞いの視点における制約依存関係

8. 評価と考察

提案パターンを例題に適用した結果を基に次の 3 つの観点から評価する.

(1) 拡張 OVM に関する評価

従来の OVM では表現できない M-M と M-A の制約依存関係を拡張した. OVM を拡張したことにより, 可変性の概念である「どのように変異していくのか」,「どの人を目的とするときにその変異がおこるか」の表現が可能になった. よって, メタデータ駆動アーキテクチャの可変性モデル作成のために OVM を拡張することは妥当である.

(2) 可変性制約依存関係パターンの評価

例題より, MVC の可変性制約依存関係は提案したパターンの挙動で表現できることが確認できた. そのため, メタデータとアプリケーションの協調関係として, メタデータ間ではテナントレベルで考慮でき, アプリケーション間ではユーザレベルで考慮できることが分かった. また, メタデータとアプリケーション間ではテナントレベルとユーザレベルの可変性の関係付けの役割を担うものと考慮できる.

(3) 可変性による影響範囲導出に関する評価

例題より, メタデータからアプリケーションとアプリケーションからアプリケーションの可変性を導出できた. この結果, 可変性制約依存関係のモデル化プロセスは, 段階的にシステムの可変性を詳細化することができる. さらに, デザインパターンの目的毎に可変性を抽出するため, 可変性による影響範囲の特定が容易になる.

上記の評価結果から, 提案パターンをアプリケーション開発における各プロセスに適用することによって, 設計者にメタデータ駆動アーキテクチャにおける可変性とその制約依存関係の理解を促進できる. この結果, メタデータ可変性とその可変性から特定できるインスタンスの可変性への導出依存関係が理解でき, メタデータ駆動アーキテクチャの設計が支援できる.

9. 今後の課題

(1) 外部可変性と内部可変性

メタデータ可変性の影響は, 外部と内部の両方の可変性に影響を与えるため, さらにパターン化する必要がある.

(2) パターンの詳細化

各 M, V, C の関係から次の可変要素を推測できるようにパターンを詳細化する必要がある.

10. まとめ

本研究では, メタデータの可変性とアプリケーションの可変性が連動していることに着目し, メタデータとアプリケーションの可変性依存関係パターンを提案した. パターン化する際に, デザインパターンの目的である構造, 生成, 振舞いの 3 つに分類し, 制約依存関係をパターン化した. また, 従来の OVM では表現できない制約依存関係の表現方法を拡張した. 提案したパターンをアプリケーション開発のプロセスに適用し, モデル化することで評価した.

参考文献

[1] 今岡 純二, Force.com のすべて, 日経 BP, 2011.
[2] R. Mietzner, et al., Variability Modeling to Support Customization and Deployment of Multi-Tenant-Aware Software as a Service Applications, Proc. PESOS '09, 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] C. D. Weissman, et al., The Design of the Force.com Multitenant Internet Application Development Platform, Proc. of SIGMOD, 09, Jun. 2009, pp. 889-896.