F-032

オントロジーを利用したクラス図設計支援とモデル駆動開発への適用
Applying an Ontology-based Class Diagram Design Support to Model-driven Development

福田 直樹 [1)]　上野 真由美 [2)]　和泉 憲明 [3)]　山口 高平 [4)]
Naoki Fukuta　Mayumi Ueno　Noriaki Izumi　Takahira Yamaguchi

1. はじめに

近年，サービス指向アーキテクチャの発展と共に，システムの外部にあるシステム（Web アプリケーションなど）をコンポーネントとして利用し，利用者の要求に応える高度なアプリケーションを迅速に開発するアプローチが取られつつある．この種のシステム開発では，外部コンポーネントの利用により，システム内部の機構が大幅に簡略化され，システムの機能設計が重要度を増しつつある．本論文でのシステムの機能設計とは，システムが持つべき機能（仕様）を満たすような，外部コンポーネントの組み合わせを適切に設計することである．

モデル駆動アーキテクチャ(MDA，Model-Driven Architecture)では，ソフトウェアのモデルを設計の主体として，実装コードに展開していくパラダイムを提案している．MDA を利用するためには，その入力となるクラス図等をモデルとして設計していく必要がある．従来のクラス図設計支援手法では，ロバストネス分析[1]によりクラス図をユースケースから作成する方法，できあがったクラス図の設計上の適切さを評価するための尺度[2]等が提案されているものの，システム内部で閉じたモデルの設計ではなく，Web アプリケーションの入出力などの，事前に与えられた要素をかならず含むようなクラス図等を適切に設計していくための方法論の検討は課題となっている．

本研究では，Web アプリケーションを外部コンポーネントとして利用した出張業務支援システム開発をケーススタディとして，モデル駆動開発の適用に必要なクラス図の設計支援手法を提案[5]するとともに，そこで設計したクラス図を用いて，モデル駆動開発によるモデル設計時の問題と対処方法の類型化を試みる．

2. MDA の適用による機能設計支援

MDA の利点は，ソフトウェアのモデルを実装プラットフォームからうまく切り離すことで独立性を高め，モデルの再利用性を非常に高くできる点である．同時に，モデルからの自動的なコード生成により，コーディングの負担軽減およびコードの信頼性向上にも寄与する．MDA がソフトウェア設計の上流工程で利用可能となれば，単にモデルを論理的な側面から検証するだけでなく，実際に動作するコードをもとに設計を検討することができ，発注者に仕様の確認を行う際の手助けにもなると考えられる．

実際には，MDA でコードの生成を適切に行うためには，厳密に記述された，ある程度完成されたモデルが必要となるため，MDA を設計の上流工程で利用することは容易ではない．特に，アルゴリズム等の動的振る舞いを記述するためのアクションセマンティクスは XMI レベルで定義されるものの，ＵＭＬ上での図としての表現は用意されず，依然としてコードの記述の必要性が残ったままである．

Web 上で提供されるサービスの増加，また Web を通じてサービスを提供するための枠組みとしての Web サービスの普及により，アプリケーション統合の分野を中心として，外部コンポーネントを利用して迅速にアプリケーションの構築を行う種類の開発が現実味を帯びてきた．外部コンポーネントを用いた開発では，キーとなるアルゴリズムのほとんどが外部コンポーネントから提供されるため，従来の MDA の弱点であったアクションセマンティクスの記述が最小限で済み，MDA の設計上流工程への適用可能性が見えてくる．

表１は，後述する出張業務支援システムの設計に MDA を適用した際に，モデルの不具合で生じる例外的な（想定外の）振る舞いを類型化したものである．モデル設計時には，設計を簡潔にするために，一般的な状況に基づいた設計が行われ，例外的な状況についての記述は意図的に十分になされない場合が多い．しかし，どこまで例外的な状況を設計に含めておくべきかどうかについては，モデル設計時には十分に検証できない場合が多い．MDA をモデル設計時に利用し，例外的な状況の頻度等を検証可能にすることにより，モデルに含めるべき例外的な状況を適切に判断するための支援が可能になると考えられる．表１では，システム構築が完成した際に最終的に考慮した例外のうち半分程度が，MDA を用いることで事前に検証可能であったことを示している．

このような状況で MDA によるモデル設計の検証を行う際に課題となるのが，外部コンポーネントを多数利用する場合でのクラス図の設計である．外部コンポーネントはそれぞれ入出力を扱うための固有なデータ構造を持っており，それらの間での整合性は十分に考慮されない．外部コンポーネントの入出力データ型をうまく包含するようなクラス図の設計には，従来の手法がそのままでは適用できず，新たなアプローチが必要になると考えられる．以降では，属性の包含関係とオントロジーとの比較により，クラス図の設計を支援する方法について述べる．

表１　モデルの不具合で生じる例外の類型化

類型化された例外の種類と数		総数	対応した数
システムの内部が原因で発生する例外	必要な情報が無くて発生する例外	2	2
	ユーザが処理不能な値を入力することによる例外	1	1
	あるプロセスに対する入力情報が不正	1	1
	ユーザ自身の勘違いによる例外	1	0
	Webサービスからの想定外の出力による例外	6	3
システム外部が原因で発生する例外	ユーザが想定外の行動をとりたがる	2	1
	Webサービスへの入力が不正	1	1

[1)] 静岡大学情報学部　[2)] 静岡大学大学院情報学研究科
[3)] 産総研ｻｲﾊﾞｰｱｼｽﾄ研究ｾﾝﾀｰ　[4)] 慶應義塾大学理工学部管理工学科

3. クラス図設計支援

クラス図の設計支援手法としては，表層的な情報(Syntactic な情報)からの支援と，意味レベルの情報を使った支援の 2 つを提案する．表層的な情報からの支援としては，Lattice(束)を利用した．Lattice を利用して支援をするにあたり，要素間の距離や深さを支援するための指標として利用できないかを検討した．

ここでは，Lattice の左側が一番浅く，Lattice の右側が一番深い．深さは属性数と同じである．あるクラスが持つ属性それぞれへの深さを調べることで，深さに大きな差異がある場合には，それらの属性を 1 つのクラスにまとめることが間違っているのではないかと考えた．しかし，深さは属性数と同じなので，深さの差異は属性数の差異であると考えられ，深さを指標として使えないと考えた．

要素間の距離は，クラス間の距離が近ければクラス間に関連がある可能性が高いと考えた．複数の共通属性があり，かつ親子関係にあって距離が近ければ，クラス間に関連があるのではないかと考えた．この指標に基づき，包含関係や継承関係を発見するための支援ができるのではないかと考えた．

意味レベルの情報を使った支援には，オントロジーを利用する．オントロジーとは概念関係の定義を表すものである．今回は，DAML[3]や WordNet[4]で提供されているものを利用し，オントロジーを作成した．オントロジーと Lattice を対応づけることにより，意味レベルでの間違いを指摘できると考えた．

本研究では，Web アプリケーションを組み込んだソフトウェア開発のためのクラス図設計支援手法のアプローチとして， a). 所与の Web アプリケーション(サービス)のポート情報を初期クラス図とする，b).領域オントロジーの概念定義構造をクラス図へ適用する，の２つのアプローチを用いる．

3.1 アプローチ A

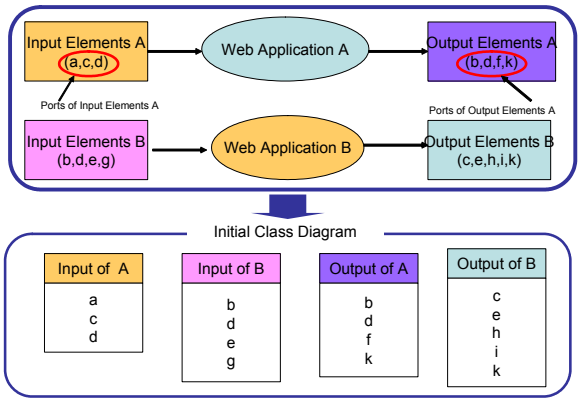

図 1　アプローチ A

アプローチ A では，図 1 のように，複数の Web アプリケーションがあり，その入出力ポートをそれぞれクラスとし，初期クラス図とする．入出力ポートとは，入出力に必要な要素の集合である．例えば，図 1 の Web アプリケーション A の入力ポートは(a, c, d)，出力ポートは(b, d, f, k)であり，それぞれを入力 A クラスと出力 A クラスとする．すべての Web アプリケーションについて，入力と出力に必要なポートを属性とし，それぞれクラスを設計する．これら全てのクラスと，Web アプリケーションの入出力に必要な属性の値を得るために必要な属性集合を 1 つクラスにしたものを初期クラス図とする．本研究での Web アプリケーションとは，Web 上で提供されているサービスであり，1 つのサイトで公開されているサービスが複数の機能を持っている場合も 1 つの Web アプリケーションとする．例えば，旅の窓口はホテルを検索する機能とホテルを予約する機能という 2 つの機能を持っているが，これは 1 つの Web アプリケーションであるとみなす．

アプローチ A の実現方法として，Lattice を用いる．本論文で扱う Lattice は，空集合から要素の漸増により全集合に至るまでの一空間であると定義する．Web アプリケーションの入出力ポートを用いて，Lattice を構成する．あるクラスの Lattice の作成は，以下のとおり行った．

1.クラスの属性を Lattice の要素として追加する．
2.クラスの属性の中に，型として他のクラスを参照している場合は，参照されているクラスの属性も追加する．
3.クラスに含まれるすべての属性がアトムになるまで 2 を繰り返す．
4. Lattice 上の，各 Web アプリケーション入出力，およびそのサブセットの部分をマークする．

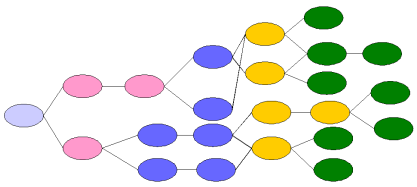

図 2　マークされた Lattice と Sub Lattice

ポートを Lattice の要素とすることによって，全ポートを一空間で管理できるという利点がある．

一般に，クラス図にある全属性数は数十以上になるので，全属性を Lattice の要素として利用する場合，Lattice をすべて計算した上で利用すると計算が不可能になってしまう．本研究では，Lattice をクラス図の洗練度を示す表現方法としてとらえ，そのために必要な部分のみに計算を限定している．ここでは，初期 Lattice に存在する共通親ノードから、４レベル以内までで抽出される Sub Lattice（図２)を新規クラス候補とした．ここで，レベルとは，分岐のない不要中間マークを取り除いた時の距離である．図２から不要中間マークを取り除くと，図３のようになる．

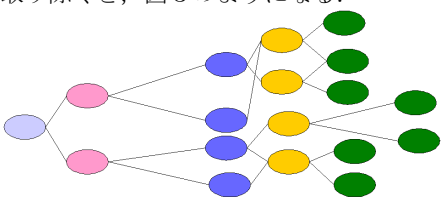

図 3　中間不要マーク削除後の Lattice

3.2 アプローチ B

アプローチ B では，アプローチ A で作成した Lattice(図４上部)とオントロジー(図４下部)を比較する．例えば，図４の下側のオントロジーでは，Y の要素のほうが X の要素より上位にあるが，図４の上側の Lattice では，X の要素のほうが Y の要素より上位概念に相当する位置関係になっている．クラス図とオントロジーの比較によって，概念定義が間違っているかラベルの付け間違いであるということを，モデル設計者が考えるきっかけを与えることができる．クラス図とオントロジーとの比較がクラス図の洗練に有用であると考えられる．

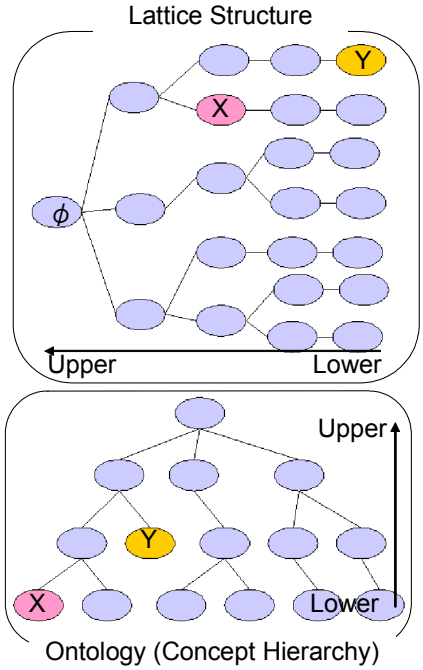

図４　オントロジーとの比較

4. クラス図設計支援のケーススタディ

業務による出張の際には，交通手段・乗り継ぎの検索，ホテル予約，および社内決済等，複数の作業を行う必要がある．これらの作業で用いる情報には相互に関連しており，たとえば列車乗り継ぎ検索で得られた乗り継ぎ情報や運賃を出張旅費の社内決済書類に自動で転記する等，必要な情報を相互に連携させることで，業務上の作業負担を軽減できると考えられる．一方で，出張業務に利用可能な Web アプリケーションは，列車乗り継ぎ検索，特急券予約，ホテル予約等が存在し，社内決済等のイントラネットアプリケーションも Web アプリケーションされている場合が多い．これらの Web アプリケーションを連携させながら一連の出張関連業務の遂行を支援するシステムとして本研究室で開発したのが，ケーススタディとして用いる出張業務支援システムである．

出張業務支援システムの初期クラス図として，Web アプリケーションの入力と出力のそれぞれを 1 つのクラスとした．前節で述べた提案手法をこの初期クラス図に適用し，洗練されたクラス図を生成できるかを検証する．

出張業務支援システムを開発する際に，必要となるそれぞれの Web アプリケーションごとに入出力属性を抽出し，そこから初期クラス図を作成した（図５）．

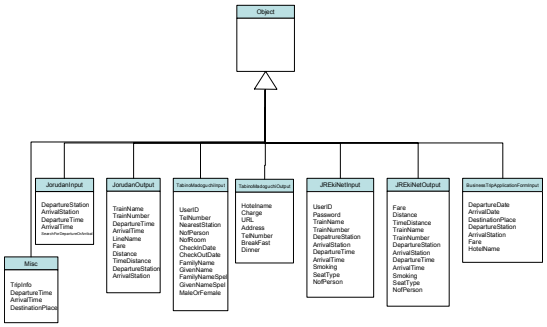

図５　初期クラス図

上記の Web アプリケーションの入出力属性と Web アプリケーションの入力属性値を得るための属性は，35 個あった．この 35 個の要素で Lattice を作成した．そして，Web アプリケーションごとに，入力に必要な属性集合を 1 つのクラスとし，出力に必要な属性集合を 1 つのクラスとして，最初のクラス図とした．これのクラスが Lattice のどの位置にあるのかを図６に示す．○で表されているのが入力で，△で表されているのが出力である．この Lattice を見ると，全体的に，Lattice の左側にクラスがあることに気づく．

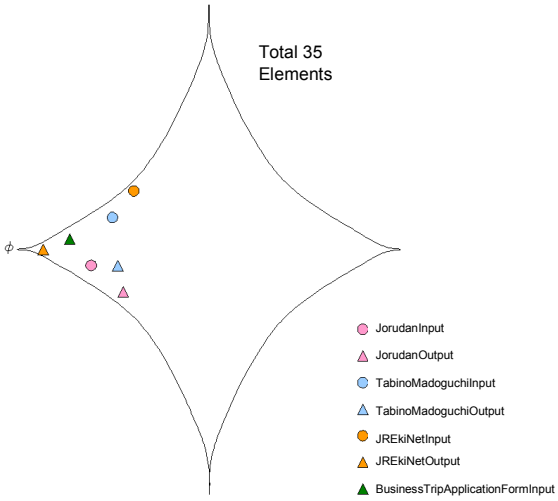

図６　Lattice 上の入出力ポートの配置

次に，本手法を用いずに，出張業務支援システムの最終段階でのクラス図に含まれている属性を要素として用いた Lattice を図 7 に示す．□で表されているのが入力で，◇で表されているのが出力である．図７の丸で囲まれている部分が，最初にクラス図として設定したクラスである．図７の丸で囲まれていないクラスは，出張業務支援システムを開発した時に設計した最終のクラス図の中から，Web アプリケーションの入出力に関係するクラスを抽出し，Lattice のどこに対応するかを示している．場合によっては，ある Web アプリケーションの入力や出力の属性を複数のクラスで持っているので，1 つのサービスの入力や出力が 2 つ以上になっていることもある．

最初のクラスの集合(図 7 の丸で囲まれている部分)と最後のクラスの集合(図 7 の丸で囲まれていない部分)を比較すると，前者のほうが 1 クラス当たりの属性数が少ないので，左側に偏っていることが分かる．つまり，最初のクラスと比較して最後のクラスは右側に移動していると言える．図 7 から，最初のクラスに属性が追加されることによって，最後のクラスが作成されたことが分かる．最初のクラスが，Web アプリケーションを利用するために最低限必要なクラスの集合であるため，最後のクラス図に洗練する時に属性(要素)が削除されることはない．もし，中間時のクラス図についても同じように Lattice 上に配置すれば，初期から中間時に洗練する際に作成された属性(要素)が最終のクラスに洗練する時に必要がないと判断し，属性(要素)が削減されることも考えられる．しかし，ここでは最初のクラスから最後のクラス図に洗練する際の変化についてのみ扱うこととする．

ジョルダンの出力情報において，最初のクラス図から最後のクラス図に洗練した際に，構造化を行うことで新たな要素が追加された．他の Web アプリケーションである旅の窓口や JR えきねっとの出力に関しても，構造化を行ったために新たな要素が追加されたと言える．例えば，列車クラスを作成したが，これは次の JR えきねっとの入力となっているためコーディングする際に，1 つのクラスを操作すればよいので，扱いやすくなったと考えられる．また，出力をタイプ別に見ることは，保守性や拡張性がよくなるということが考えられる．以上のことから，各 Web アプリケーションの出力情報に追加された要素は，構造化したことにより発生し，それにより，実装効率を高めるということが言える．

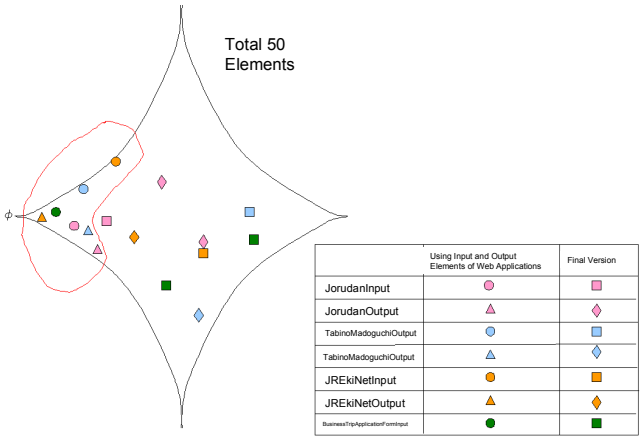

図 7 洗練による Lattice 上での動き

最初のクラス図(図 5)を Lattice で表現する. 前述した提案手法「共通親ノードから 4 レベル以内までで抽出される Sub Lattice を新規クラス候補とする」に基づき，共通親ノードから Sub Lattice を抽出したのが図8である.

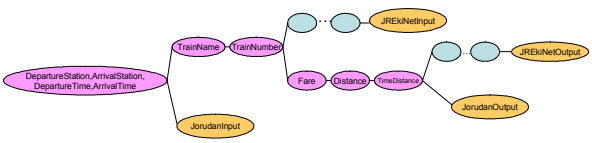

図 8 共通親ノードから4レベル以内の Sub Lattice

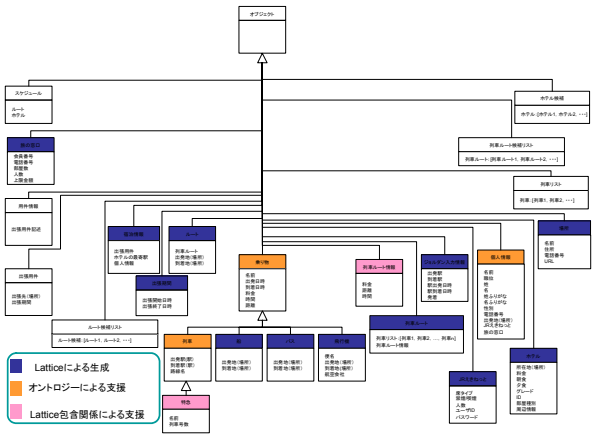

図 9 開発時のクラス図への本手法の適用

出張業務支援システム開発時での，最終のクラス図との比較の全体像を図 9 に示す．図 9 では，大きさの都合でクラス図の詳細は見えないが，クラス名の部分に，Web アプリの入出力要素の Lattice 上への配置により得られた初期クラス，Lattice 上での包含関係による支援の結果生成されたクラス，オントロジーとの比較により洗練されたクラスをそれぞれマークしてあり，本手法が適応できた割合がわかるようになっている．本手法で支援されなかったクラスは，ユーザインタフェースに固有に使われるクラス，および，多重度を扱うためのプラットフォーム固有のラッパークラスであった．以上より，提案手法によって，包含関係や継承関係の発見，他と共通して使われている属性集合や意味的に近い属性集合を分離することが支援可能であると考えられる．

5. おわりに

本論文では，モデル設計の中で，特にクラス図に焦点を当て，外部コンポーネントの入出力を起点にクラス図を設計するためのクラス図設計支援手法を提案した．ケーススタディを通じて，Lattice 上の位置関係と関連オントロジーを注意深く比較することにより，初期クラス図の洗練を支援可能であることを示した．MDA に基づくシステムの機能設計時にシステムの挙動確認を容易にすることで，設計したモデルが持つ不具合を予想可能かどうか調査した．

謝辞

本研究は，文部科学省科学研究費補助金(15300043)「オントロジーとモデル駆動型アーキテクチャの統合」の助成による．

参考文献

[1] G.Booch.: “Object-Oriented Analysis and Design with Applications.”, Benjamin Commings, 1994.
[2] Shyam R.Chidamber and Chris F.Kemerer: A Metrics Suite for Object Oriented Design, IEEE transactionon Software Engineering, Vol.20, no6, June1994.
[3] DAML http://www.daml.org/
[4] WordNet http://www.cogsci.princeton.edu/wn/
[5] 上野真由美: UML モデリングにおけるクラスダイアグラム設計支援, 静岡大学大学院情報学研究科修士論文,2003.