1G3-03

オントロジーとモデル駆動アーキテクチャに基づく機能設計の支援

Function Design Support with Ontology and Model-driven Architecture

見置 孝昌[*1] MIOKI Takamasa　上野 真由美[*1] UENO Mayumi　福田 直樹[*2] FUKUTA Naoki　和泉 憲明[*3] IZUMI Noriaki　山口高平[*4] YAMAGUCHI Takahira

[*1] 静岡大学大学院情報学研究科 Graduate School of Informatics, Shizuoka University

[*2] 静岡大学情報学部 Faculty of Information, Shizuoka University

[*3] 産業技術総合研究所サイバーアシスト研究センター Cyber Assist Research Center, AIST

[*4] 慶應義塾大学理工学部管理工学科 Department of Administration Engineering, Keio University

In this paper, we propose a support mechanism to design the functionalities of software by using ontology and model driven architecture. Ontology is used to make an initial model and identify some unbalanced parts in the class diagram. Model driven architecture is employed to overcome the function modeling process by formulating bug patterns and the respective re-design strategies. In the case study of developing business trip support system, we analyzes how initial class diagram could be refined. A model-driven development tool is applied to formulate their initial bug patterns by using model-to-code conversion techniques. In our preliminary evaluation, we found that there are typical error patterns in the initial model.

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．この主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)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ダイアグラム等をモデルとして設計していく必要がある．従来のクラスダイアグラム設計支援手法では，ロバストネス分析[1]によりクラスダイアグラム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ダイアグラム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ダイアグラム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ダイアグラムを用いて，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. クラスダイアグラム設計支援

クラスダイアグラム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 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．

2.1 モデル設計支援手法

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ダイアグラム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ダイアグラムとする，b).領域オントロジーの概念定義構造をクラスダイアグラムへ適用する，の2つのアプローチを用いる．

連絡先：見置孝昌 静岡大学情報学部 〒432-8011 浜松市城北3－5－1，053-478-1478 mioki@ks.cs.inf.shizuoka.ac.jp

2.2 アプローチA

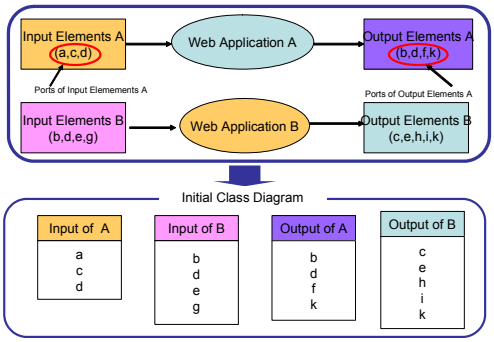

図 1 アプローチ A

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，初期クラスダイアグラムとする．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ダイアグラムとする．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という機能とホテルを予約するという機能という 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(束)を用いる．本論文で扱う Lattice(束)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．

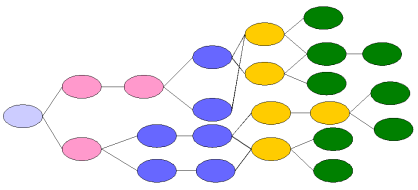

図 2 Lattice と Sub Lattice

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，クラスダイアグラム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 をクラスダイアグラム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概念を取り除いた時の距離である．図 2 から不要中間概念を取り除くと，図 3 のようになる．

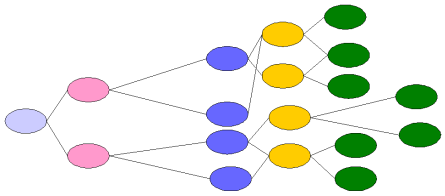

図 3 中間不要ノード削除後の Lattice

2.3 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，ピンク色の要素のほうがオレンジ色の要素より上位にあるが，図 4 の上側のオントロジーでは，オレンジ色の要素のほうがピンク色の要素より上位概念になっている．クラスダイアグラム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．クラスダイアグラムとオントロジーとの比較がクラスダイアグラムの洗練に有用であると考えられる．

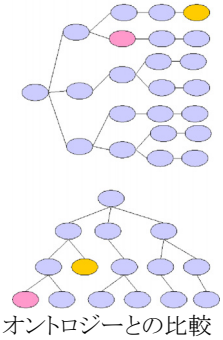

図 4 オントロジーとの比較

3. ケーススタディ

3.1 モデル設計支援手法の適用

本研究室で開発した出張業務支援システムの初期クラスダイアグラムとして，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．第 3 章で述べた提案手法をこの初期クラスダイアグラムに適用し，洗練されたクラスダイアグラム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ダイアグラムを作成した(図5)．

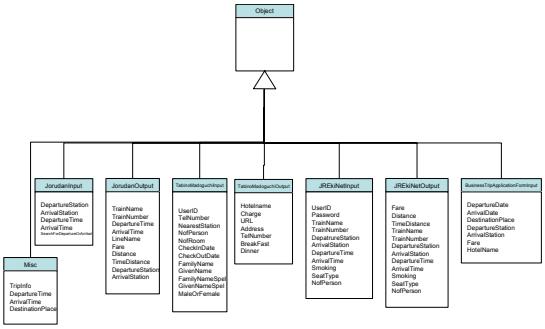

図 5 初期クラスダイアグラム

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，最初のクラスダイアグラムとした．これのクラスが Lattice のどの位置にあるのかを図 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．ピンク色はジョルダン，水色は旅の窓口，オレンジは JR えきねっと，緑は出張申請システムである．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．

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ダイアグラ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．ピンク色はジョルダン，水色は旅の窓口，オレンジは JR えきねっと，緑は出張申請システムである．赤い丸で囲まれている部分が，最初にクラスダイアグラムとして設定したクラスである．赤い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ダイアグラムの中から，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(赤い丸で囲まれている部分)と最後のクラスの集合(赤い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 6 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，最後のクラスダイアグラムに洗練する時に属性(要素)が削除されることはない．もし，中間時のクラスダイアグラム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ダイアグラム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，最初のクラスダイアグラムから最後のクラスダイアグラムに洗練した際に，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．

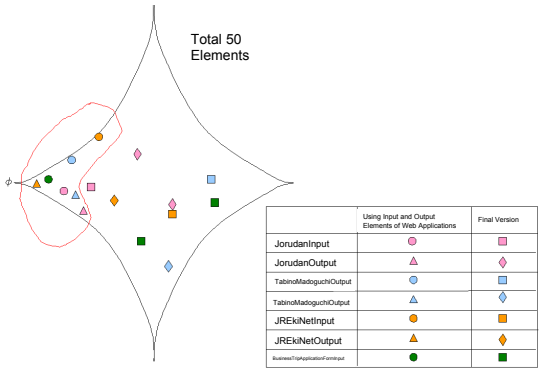

図 6 Lattice 上の入出力ポートの配置

最初のクラスダイアグラム(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 9 である．

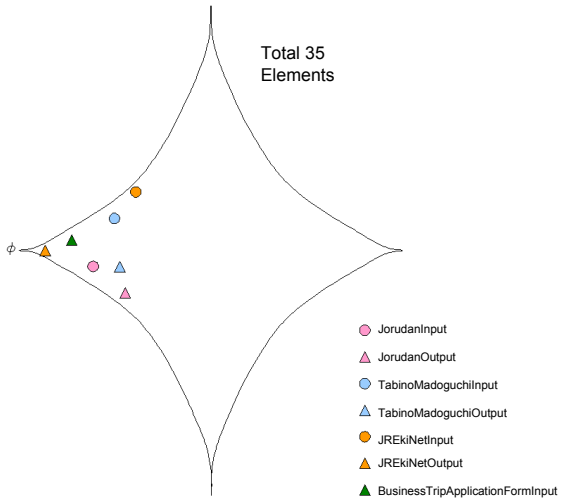

図 7 洗練による Lattice 上での動き

ピンク色のノードは共通ノードで，オレンジ色のノードはクラスである．出発駅，到着駅，駅出発日時，駅到着日時，料金，距離，時間，路線名がジョルダン出力クラスであるが，このクラスは列車を表しているクラスだと考えられる．実際システムを開発した時には，最終的には，特急クラスと列車クラスは分離していた．この時は，実際には列車は列車号数や名前を持っていて，特急は路線名を持っているが，検索や予約の時に特に重要でないと判断したため，削除した．図 7 を見ると，列車名と列車号数は 1 つの共通ノードになっており，人間が判断すれば別クラスとすることも可能であるので，列車クラスを継承して特急クラス(列車名，列車号数)とした．料金，時間，距離については，列車のルート間についての情報を表しているので，これらを 1 つのクラスとして分離できると考えた．結果，ジョルダン出力クラスが列車

クラスと特急クラスになり，JR えきねっと出力が特急クラスと関連を持つというようにクラスダイアグラムを変更した．JR えきねっと入力についても，図 8 を見ると分かるように，特急クラスの属性の大部分を属性として持っているので，これを特急クラスとして置き換えた．

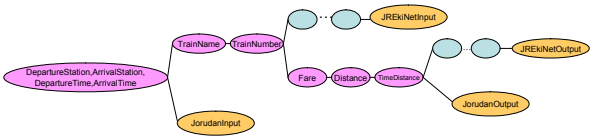

図 8 共通親ノードから4レベル以内の Sub Lattice

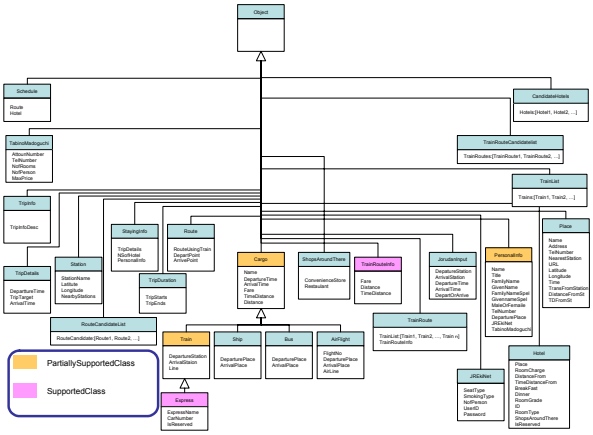

図 9 開発時のクラスダイアグラムへの本手法の適用

出張業務支援システム開発時での，最終のクラスダイアグラムとの比較を図 9 に示す．図 9 で，ピンク色で示したクラスは，提案手法によって支援できたクラス，黄色で示したクラスは提案手法によって一部支援ができたクラスである．以上のことから考えると，提案手法によって支援できることは，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である．

4. MDAの適用による機能設計支援

図 10 は出張業務支援システムのアクティビティ図である．図 10 をモデル駆動アーキテクチャ(MDA)に基づ開発支援ツール ArcStyler の入力とし，システムを配置(deploy)する際に必要となるファイルが自動生成されることを確認した．

MDA ツールへ入力モデルは Web アプリケーションに特化した構造はないが，MDA 支援ツールによって生成されたコードおよびモデルは Web アプリケーションとして動作するモデルとなっている．MDA 支援ツールが作成したモデルを Web アプリケーションの構造に適合した形にモデルを展開する作業が支援された．

次に，モデルから生成されたコードを実行することで得られる振る舞いをどのようにモデルの洗練に役立てるのかを検討する．

出張業務支援システムを設計，または実装する際に，考えられたバグの類型化を行った．本システムを構築するにあたり，考えられたバグの数は全てで 18 挙げられた，これらを類型化して表 1 を得た．

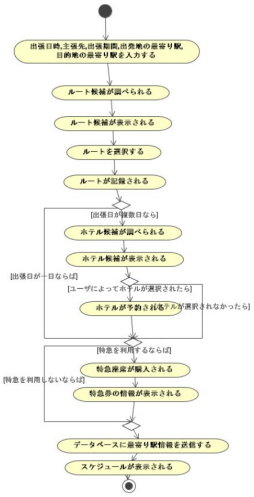

図 10 出張業務支援システムのアクティビティ図

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

5. Conclusion

本論文では，モデル設計の中で，特にクラスダイアグラムに焦点を当て，外部コンポーネントの入出力を起点にクラスダイアグラムを設計するためのクラスダイアグラム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，初期クラスダイアグラムの洗練を支援可能であることを示した．MDA に基づく開発支援ツールを適用し，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．モデルが持つ不具合とシステムの挙動との関連を類型化した．システムの挙動からモデルの持つ不具合とその修正方法の特定の支援は，今後の課題である．

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.

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．