B4-5　要求仕様のパターン獲得

徳田英修　　白岩政太郎　　三浦孝夫

法政大学大学院工学研究科電気工学専攻

概 要

ソフトウェア設計とデータベース設計は上流過程において、必要な情報の確定、またその情報の操作をするところなどが、類似しているといえる。現在、要求を確定する方法は漠然としており、その体系的な方法が必要である。我々は要求分析時の出力である要求仕様書からパターンを獲得し、ノウハウの蓄積をする方法を提案する。抽象度の低い要求仕様書から、段階的に固有情報を削除、一般化していくことにより、階層的な抽象度の持つパターンが獲得できる。本稿では例を用いてこの流れを提案し、その有効性を述べる。

1　目的

ソフトウェア開発過程初期の要求分析においてオブジェクト指向分析の図形が果たした役割は大きい。文章だけでなく図形を利用することにより、視覚的にシステムを理解することができ、意思の疎通が容易となる。

要求分析を支援する様々なCASEツールがあるが、これらは本質的には図面ツールで、オブジェクト指向分析のグラフィカルな部分をサポートするものである。手で描かれた図を変更するより図面ツールで構築された図を修正するほうが容易であり、この手作業の部分を支援することを目的としている。いくつかのCASEツールには、基礎となるモデルへの変更があると、自動的に全ての影響を受けた図形に反映するものもあるしかしこれらのCASEツールは、分析の思考を支援しているわけではなく、あくまで思考する際の余分な労力を削減するためのものであるといえる。

本稿では、繰り返し行われる要求分析の中からパターンを獲得する手法を論じる。獲得されたパターンはノウハウとして蓄積することができ、要求分析を再利用することによる要求分析の支援が可能となる。

本論文の構成として、まず第二章で要求パターンを定義し、第三章でパターンを獲得するための問題点と解決策を挙げる。続く第四章で具体例を挙げてパターン獲得の流れを説明し、第五章で評価、第六章で結びとする。

2　要求パターン

デザインパターンとはソフトウェア設計の経験をまとめたもので、ソフトウェア設計の際、何回も起きる問題に対する解法を示すものである。これは設計上の様々な状況における問題の解決に適用され、すでに成功した設計を他者が再利用することが可能となる [3]。要求パターンはデザインパターンより上流部分のパターン化であり、デザインパターンと同様に、繰り返し出てくる要求をパターン化したものと定義する。

ソフトウェア開発過程における設計フェーズでは、仕様書とともに要求分析に使用した文書も利用される。その中にはクラス図やコラボレーション図、シーケンス図等

が含まれている。類似した製品を作る時、これらの再利用をする可能性がある。我々はこれら要求分析時に生じる UML 図からパターンを獲得する。

オブジェクト指向分析には３つのステップがあり、それぞれ Use-case modeling、Class modeling、Dynamic modeling である [2]。Use-case modeling で様々な結果が製品によってどのように計算されるかを示し、Class modeling で、クラスやそれらの属性およびクラス間の相互関係を決定する。また Dynamic modeling はクラス、サブクラスによって行われるアクションを決定する。つまり、これらの図形からパターンを獲得するには、ユースケース図、もしくはクラス図のパターン化をすることとなる。

ユースケースのパターン化を考えた場合、その意味情報を抽象化すると、ユースケースとしての意味を全くなさなくなってしまう。よって、ユースケースのパターン化は不適であるといえる。それに対し、クラス図はオブジェクト間の関連、相互作用を示したものであり、オブジェクト間の関連を特徴と見てパターン化することが可能である。

3　パターン化と抽象度

要求分析の出力である UML 図からのパターン獲得を考えた場合、抽象度が問題となってくる。例えば「徳田酒屋」の例を考える。「徳田酒屋」の倉庫では毎日数個のビン詰めの酒が入ったコンテナが搬入されてくる。倉庫係はコンテナを受け取り、そのまま倉庫に保管し、積荷表を受付係に渡す。空になったコンテナはすぐ搬出される。受付係は出庫依頼を受け、そのつど倉庫係へ出庫指示書を出す。在庫がないかあるいは数量が不足している場合は、その旨を依頼者に連絡し、在庫不足リストに記入する。そして該当品の積荷が必要量あった時点で不足品の出庫指示をする。また空になるコンテナを倉庫係に知らせる [1]。

この「徳田酒屋」という酒屋の「徳田酒屋在庫管理システム」からパターンを獲得する時、そこから得るべきパターンは人によって違う。欲しいパターンは「酒屋在庫管理システムパターン」かもしれないし、「在庫管理システムパターン」かもしれない。あるいは単純に「管理システムパターン」かもしれない。三浦酒屋が徳田酒屋のシステムをもとに在庫管理システムを作ろうと思ったのなら、必要となるパターンは「酒屋在庫管理システムパターン」である。しかしある本屋が「徳田酒屋在庫管理システム」の在庫の管理の仕方を参考に本の在庫を管理するシステムを作ろうと思ったのなら「在庫管理システムパターン」が必要となる。

このようにシステムから獲得するパターンをひとつに絞ることは不可能である。パターンを獲得するには元となるシステムに固有な情報を削除、一般化し、抽象度を高めていくが、我々はこの過程を段階的にすることを考える。こうすることで抽象度に段階が設けられ、一つのシステムから複数のパターンが獲得されることになる。要求パターンは情報を削除、一般化するごとに抽象度があがり、それとともに具体性が失われていく。これら抽象度によって分けられたパターンの中からユーザは必要な抽象度のパターンを選択し、利用していくことになる。

4　パターン獲得の流れ

この節では例を用いて要求仕様からのパターン獲得の流れを説明する。以下の手順により、階層的な抽象度をもつ要求パターンを獲得する。

１．オブジェクトの分類
２．省略オブジェクトの抽出、追加
３．固有名詞、例に固有な情報の削除、一般化
４．相互作用を起こさないメソッド削除
５．連結度が低いものの削除

図１は「徳田酒屋」の在庫管理システムのクラス図である。

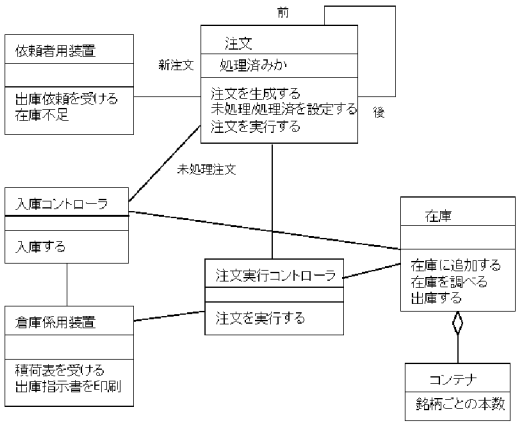

図 1: 酒屋倉庫問題クラス図 [1]

4.1　オブジェクトの分類

最初にクラス図を３つのオブジェクトのタイプに分類する。オブジェクトは境界オブジェクト、制御オブジェクト、実体オブジェクトに分けることができる。境界オブジェクトは直接アクターと相互作用するオブジェクトであり、制御オブジェクトはオブジェクトの動作を制御するオブジェクトである。実体オブジェクトは相互作用を自分から発生しないオブジェクトであり、メッセージがデータ値の設定、更新、参照などを行う操作を起動する。それぞれのオブジェクトの関係は図２のようになっている。

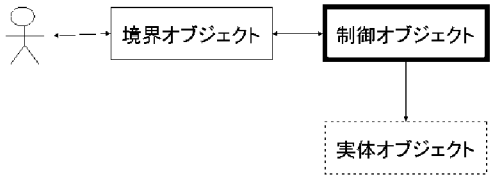

図 2: 基本的なオブジェクトの関係

以上の定義によりオブジェクトの分類を行ったものが図３である。

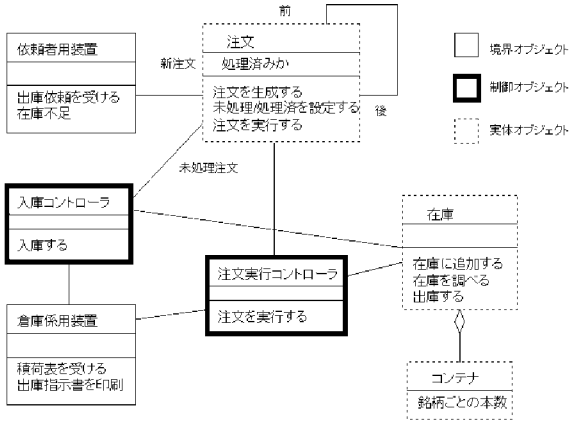

図 3: オブジェクトの識別

4.2 省略オブジェクトの抽出、追加

次にクラス図製作過程において省略されたと思われるオブジェクトの抽出、追加を行う。この例では本来、図 4 のように依頼者用装置と注文オブジェクトの間には出庫依頼コントローラがあったと考えられる。この出庫依頼コントローラは、注文オブジェクトというただひとつの実体オブジェクトと交信するだけであり、特に制御オブジェクトを置く効果がないため、省略されたものと考えられる。しかしこのクラス図を今後再利用することを考えた場合、この出庫依頼コントローラは注文実行コントローラのように複数のオブジェクトの呼び出し制御を行うように利用される可能性がある。こうなるとこの例のように出庫依頼コントローラを省略することはできなくなる。ここでは今後の再利用性を考え、こういった省略されたオブジェクトを見つけ出し、追加する。この状態が全ての情報を含んだ、もっとも抽象度の低い状態である。

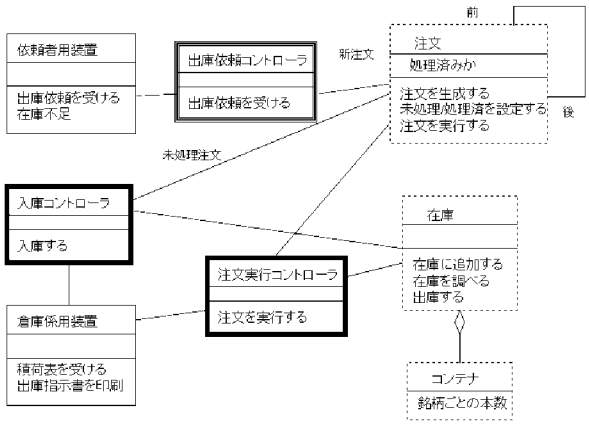

図 4: 省略オブジェクトの抽出、追加

4.3　固有名詞、例に固有な情報の削除、一般化

元となるシステムから削除、一般化する情報はパターンとしての重要度の低い、一般的ではないものから行う。例えば「徳田酒屋在庫管理システム」の「徳田」や「酒屋」といった情報である。この例の場合、在庫から出ている「コンテナ」がこの例に固有な情報である。（図５）パターンとしてみた場合、在庫がコンテナに入っていようが段ボール箱に入っていようが興味はない。ここでは固有な情報の削除、一般化を行う。

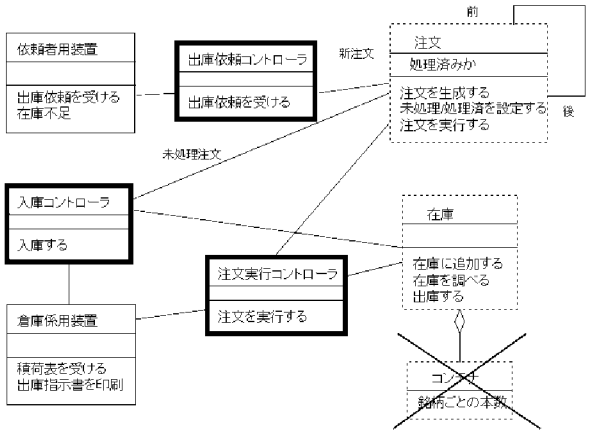

図 5: 固有な情報の削除

4.4　相互作用を起こさないメソッド削除

ここでは他のオブジェクトに相互作用を起こさないメソッドを省略する。この操作により、オブジェクト間の関連のみを強調したパターンを獲得できる。他のオブジェクトと相互作用を起こさないということは、他のオブジェクトに影響を与えていない情報であるということなので、これらの省略により他のオブジェクトに影響を与える情報だけが残り、オブジェクト間の関連のみが強調されることになる。（図６）

4.5　連結度が低いものの削除

この段階では連結度の低いオブジェクトを削除していく。ここでいう連結度とは単純にオブジェクトから伸びている関連を示す線の数であり、この数が少ないものほど他のオブジェクトとの関連が少なく重要度が低いと考え、削除していく。具体的には連結度が１しかないものを削除する。この例では依頼者用装置オブジェクトの連結度が１であり、削除の対象となる。（図７）

ここで得られたパターンは連結度の高い、システムの中枢となるオブジェクトのみが取り出された形となる。

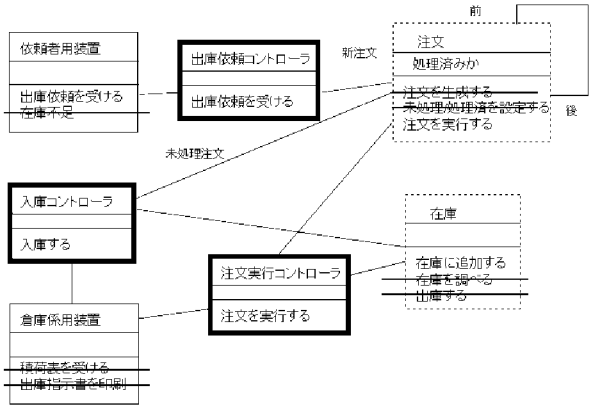

図 6: 相互作用を起こさないメソッドの省略

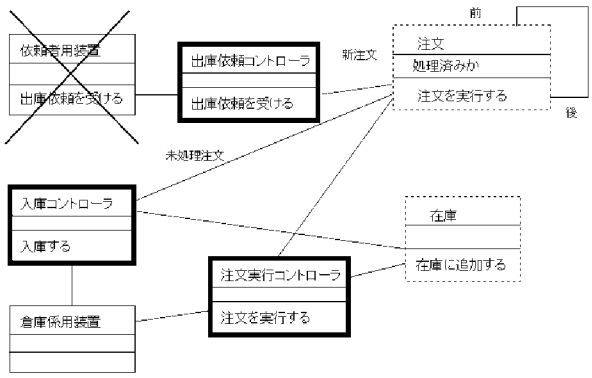

図 7: 連結度の低いオブジェクトの削除

5　評価

得られた複数のパターンは図８のように階層的な抽象度を持つ。この節ではこの抽象度の階層ごとにパターンの評価を行う。

省略オブジェクトの追加をした状態が最も具体性が強く、抽象度が低い。逆に最後に連結度の低いオブジェクトを削除したパターンがもっとも抽象度が高い。

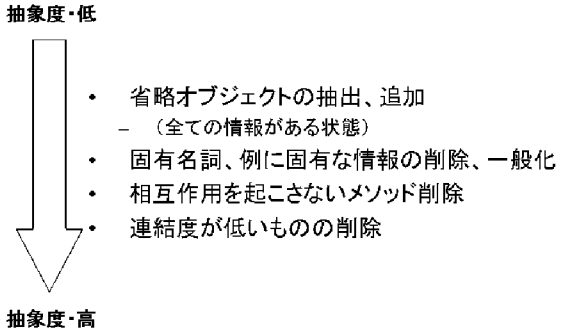

図 8: 抽象度

5.1　省略オブジェクトの抽出、追加

省略したオブジェクトを抽出し、追加することにより要求パターンは全ての情報を保有した状態になる。このパターンはもっとも具体性が強いため、他のシステムに組み込もうとした場合、変更するべき箇所が多くなってしまいコストがかかる。この段階はパターンを獲得するための準備状態である。

5.2　固有名詞、例に固有な情報の削除、一般化

この抽出度の段階のパターンはよく似た状況のシステムでパターンを再利用する時に利用する。

例えば徳田酒屋での在庫管理の仕方をそのまま三浦酒屋に導入するとき、このパターンをそのまま導入できる。しかし、パターンとしては具体性が強いため、他のシステムにパターンの一部分だけを使用したいときなど、変更箇所が多くなり、コストがかかる。倉庫の在庫を確認し、入庫するだけのシステムを作ろうとした場合、注文オブジェクトや依頼者用装置オブジェクトなどは不要であり、これらの削除と削除にあわせたクラス図の再構築が必要となってしまう。

5.3 相互作用を起こさないメソッド削除

この抽出度の段階のパターンはオブジェクト間の関係が強調されたものであり、各オブジェクト個々の内部的な処理は無視されている。

例えば、徳田酒屋では倉庫係が出庫指示書を印刷しているが、出庫指示書を印刷するのが印刷所オブジェクトであった場合、一段階前の抽象度のパターンでは倉庫係用装置オブジェクトを変更するというコストがかかる。しかし相互作用を起こさないメソッドが削除されたこのパターンでは、新たに印刷所オブジェクトを作成すればよいだけである。

5.4 連結度の低いものの削除

この段階のパターンは連結度の低いオブジェクトが排除されたことにより、他のオブジェクトとの結びつきの強い、システムの中枢部のオブジェクトのみが残ったものである。それ以外の情報が削除されてしまったため具体的なシステムとしての意味合いが薄れてしまっているが、具体性が失われたことによって汎用性が高くなっている。

例えば出庫依頼コントローラは依頼者用装置から出庫依頼を受けるが、このパターンだと、そこに実体オブジェクトである「依頼者要望書オブジェクト」をかわりに配置することも可能である。

6 むすび

本研究では繰り返し行われる要求分析のノウハウを蓄積し、再利用するためにそれらのパターン化を考えた。しかし必要となるパターンの抽象度は、再利用する際の状況や、人によって違う。そのため我々は階層的な抽象度をもつ、要求パターンの獲得をする方法を提案した。

要求分析で出力されるクラス図より要求パターンを獲得するために、まずオブジェクトを境界オブジェクト、制御オブジェクト、実体オブジェクトに分類し、そこから省略されたオブジェクトを抽出し、追加した。実際の要求分析では不要として省略されたこれらのオブジェクトは再利用の際、省略できないものとなる可能性がある。再利用性を向上させるためにこれらのオブジェクトを探し出し、加えた。続いてこの全ての情報を持つ最も具体的な状態のクラス図から重要性の低いと考えられる情報を順に削除、一般化していき、抽象度を高めていった。最初に固有名詞等の元となるクラス図に固有な情報を削除、一般化したパターン、そこから他のオブジェクトとの相互作用をおこさないメソッドを省略し、オブジェクト間の関連だけを残したパターンを獲得した。最後に連結度の低いオブジェクトを削除したパターンを獲得した。階層化された抽象度を持つパターンは再利用の際、必要な抽象度のパターンが選択され利用されることになる。

参考文献

[1] 磯田定宏:オブジェクト指向モデリング, コロナ社 (1998)

[2] Stephen R. Schach:Object-Oriented and Classical Software Engineering,McGraw-Hill Professional Publishing(2001)

[3] Erich Gamma:オブジェクト指向における再利用のためのデザインパターン, ソフトバンクパブリッシング (1999)