エコマネーWeb プラットフォームのドメイン・モデルの設計

The Design of Domain Model for an Eco-Money Web Platform

小久保 温†，柏谷 至†，石橋 修‡，櫛引 素夫†，坂井 雄介†，佐々木 てる†，田中 志子†
†青森大学，‡八戸学院大学

Abstract

The 'eco-money' is a kind of local currency. We have developed a web platform for the experiments of eco-money at Koubata region of Aomori City in 2015. Eco-money users can trade services or goods with eco-money by using the platform. In this article, we discuss design of domain model for our eco-money platform. We analyzed Entity-Relationship of the domain and designed the main-constitutes of the domain model are 'user', 'service and good' and 'trade'. The sub-constitutes are 'payment', 'following relationship', 'message' and 'notification'. The subclasses of 'service and good' are 'want' and 'offering'. The subclasses of 'message' are 'talk' and 'negotiation'. These subclasses implemented with the 'Single Table Inheritance' architecture pattern. The 'trade' has five states, 'to be contracted', 'canceled', 'to be performed', 'to be paid' and 'finish'. These states internal-represented by date-time fields of the related actions.

Keywords: eco-money, local currency, domain model, object-relational mapping, web service

1 エコマネーとは

「エコマネー」は「地域通貨」の一種である．

地域通貨とは，なんらかのコミュニティの内部で流通する通貨の一種であり，法定通貨でないものを指す．近代的な地域通貨としては，マイケル・リントンが 1983 年にカナダではじめた LETS(Local Exchange Trading System)，エドガー・カーンが 1985 年にアメリカではじめたタイム・ダラー(Time Dollar)などが知られている．日本では，その機能や運用形態は，紙幣類似証券取締法により制限されている．

「エコマネー」は，加藤敏春が 1997 年に提唱したもの(加藤 2001)で「エコロジー」「エコノミー」「コミュニティ」「マネー」を組み合わせた造語であり，コミュニティの再生を目指し「環境」「介護」「福祉」「コミュニティ」「教育」「文化」などに関するものを取り引きするものとされる．具体的には，サービスや物品をエコマネーという通貨を介して取引する．エコマネーの現状については，山﨑(2013)などがある．

エコマネーの具体的なしくみを，図 1 に示した．まず，①エコマネーの運営団体を設立し，②参加したい人は運営団体に参加登録する．そのときに連絡先などを知らせる．③運営団体は，登録した人に所定の初期額を記入した「通帳」を渡す．「通帳」はたとえば紙のノートである．④参加者は，取引したい役務や物品を列挙したものを作って，運営団体に提出する．取引したい役務や物品とは，自分が「お願いしたいこと・もの」「提供できること・もの」である．⑤運営団体は，全参加者の取引したい役務や物品をまとめてリストを作り，連絡先とともに配布する．⑥参加者はそれらのリストを見て，自分たちで相手と交渉し，役務や物品を取引し，対価をエコマネーで支払うというものである．

取引では，a.相手と取引の条件(日時や価格など)を交渉し，b-1.交渉がまとまると実施を契約(約束)する．c.契約したら，請負者はそれを実施する．d.実施されたら，依頼者は支払をする．なお，b-2.交渉がまとまらなかった場合はキャンセルする．

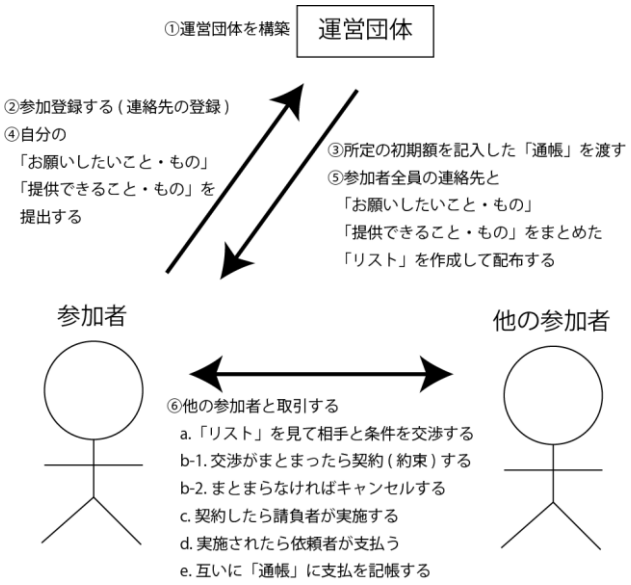

図 1: 典型的なエコマネーのしくみ

2 プラットフォームの目的と要件

平成 27 年度，青森大学周辺の青森市幸畑地区で，地域の住民と大学生の交流を意図とする，エコマネーの実証実験をすることになり，そのためのシステムを開発することになった．

地域通貨を運用する際に，参加者に通帳として紙のノートなどを配布することもあるが，これは準備が容易な一方で，①運営側で取引の実態を把握することができない，②通帳を持ち歩く必要がある，③通帳を紛失する危険性があり，紛失すると復元できないなどの欠点がある．そこで，運営側で取引の状況を把握できるように，Web アプリケーションとしてエコマネーのプラットフォームを開発し，利用することにした．ターゲットとするクライアントとして，近年普及しているスマートフォンを対象とすることにした．なお，地域の方々のうち，高齢者はスマートフォンを所持していない割合が高いことが想像されるが，別途運用を検討してカバーすることにした．

プラットフォームでは，役務や物品の授受など現実世界でのアクションが必要なもの以外，エコマネーに関わるすべての運用が完結するように検討し，以下の要件を定めた．

1.エコマネーの参加者は，プラットフォームに登録する．
登録すると所定の額の電子エコマネーを渡される．
2.参加者は，取引したい役務や物品(「お願いしたいこと・もの」「提供できるもの・こと」)を公開できる．
3.参加者は，他の参加者と役務や物品の取引に関わる検索や交渉や決済ができる．

また，エコマネーのやりとりを便利にするために，以下の要件を定めた．

4.参加者は，プロフィールを公開できる．
5.参加者は，他の参加者をお気に入りに登録できる．
6.参加者は，他の参加者と文字で会話をすること(メッセージのやりとり)ができる．
7.他の参加者が自分に対してアクションを起こすと，お知らせが送られてくる．

3 ドメイン・モデルの設計

要件を元にドメイン・モデルを検討して設計した．ドメイン・モデルとは，システム化する対象領域(ドメイン)のモデルである．

今回開発するエコマネー･プラットフォームは，しくみとしては物品を出品し販売するネットショップやオークションに似ている．ただし，役務の提供を含むため，実施日程や支払額などには交渉が発生するため，取引の状態の取り扱いは複雑になった．

以下, ドメイン･モデルの設計について論じる．

3.1 設計の前提

今回開発するプラットフォームは，開発期間を短縮するために Web アプリケーション･フレームワークを使用し，ユーザー数の増加に容易に対応できるクラウドを使用して稼働させることにした．具体的には，Ruby on Rails フレームワーク(Hansson, 2009)とクラウド PaaS(Platform as a Service)の Heroku(“salesforce.com, inc.”., n. d.)である．また，データの永続化には Heroku で利用可能なものの一つである関係データベースの PostgreSQL を使用することにする．

また，今回はドメイン・モデルについて論じ，画面遷移などについては論じないが，プラットフ

ォームは原則として RESTful アーキテクチャー(Fielding, 2000)に準拠し，それぞれのエンティティのうち公開されるものに対応した URL を用意する．

3.2 主要なエンティティ

最初に主要なエンティティを抽出した．プラットフォームの要件 1〜3 に沿って, エコマネーのやりとりを整理すると,「ユーザー」による「役務・物品」の「取引」であり，これら 3 つが主要なエンティティとなる.「取引」には「支払」が付帯し，これも説明上，主要なエンティティとした方がわかりやすいので，本論文ではそのように扱う．

また，エコマネーのやりとりを便利にするために定めた要件 4〜7 に沿い，「お気に入り」「会話」「お知らせ」なども主要なエンティティとなる．

3.3 ユーザーのエンティティ

ユーザーは，システムへの登録と認証ができる必要がある．また，円滑に利用するためにプロフィールが公開できる必要がある．

このうち，登録と認証は開発の工数を減らすために，devise(Plataformatec, 2009)というプラグインを使用することにした．devise では，e-mail アドレスとパスワードで認証し，ログイン状態を保持することができる．また，メールでの登録確認，パスワードの回復，ログイン失敗を繰り返すとアカウントをロックするなど，アカウントを利用するときの利便性とセキュリティを高める機能があり，これらの機能で使用する属性があらかじめ用意されている．

更に，プロフィールとして名前と自己紹介を公開できるように属性を用意した．

devise のデフォルトの設定では，退会するとユーザーのデータを削除してしまう．しかし，ユーザーのデータを削除すると，関連する取引記録などから参照できなくなる．そこで，退会時にデータを削除せず，退会日時を記録し，退会日時のデータが存在するユーザーを退会したとみなすことにした．

また，システムの管理上，管理者ユーザーを用意した方が便利なので，真偽値の管理者属性を用意することにした．

以上を ER(Entity-Relationship)図の記法を用いて図 2 にまとめた．

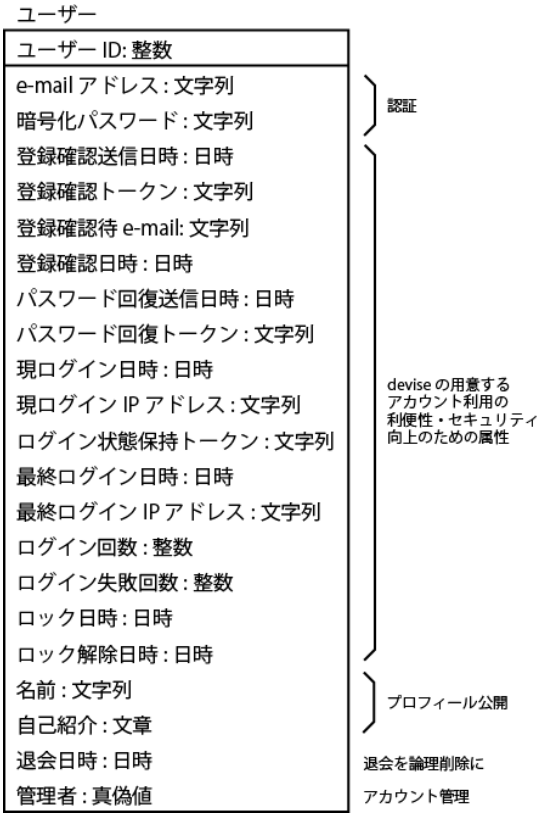

図 2: ユーザーのエンティティ

3.4 役務・物品のエンティティ

取引したい役務・物品のエンティティは，登録するユーザーが提供する場合と受納する場合の 2 種類があり，これらを構成する要素はユーザーの役割(提供・受納)以外は同一である．

このような場合，オブジェクト指向言語では継承を利用すると設計を整理することができる．しかし，関係データベースの理論的背景となっている関係データモデルには，継承を表現する方法がない．

オブジェクト指向言語と関係データベースを対応させ，継承を実現する実装方法はいくつかあるが，今回は設計を単純に保つため，ファウラー(2003)による「単一テーブル継承(STI: Single Table Inheritance)」アーキテクチャ・パターンを採用することにした．STI は，データベースに継承の親クラスに対応した表を用意し，そこに子クラスの種類を示す属性を設けるという継承の実現方法である．インスタンス化できる子クラスとしては「お願いしたいこと・もの」と「提供できる

こと・もの」の 2 つを用意し，それらの親クラスは「役務・物品」とした．

役務･物品を表現する親クラスに必要な属性は，カテゴリー，ユーザー，タイトル，詳しい説明，価格の記述とすることにした．このうち，カテゴリーは独自に増やせないように，独立したエンティティを用意して参照することにした．カテゴリーのエンティティは名称の属性のみを持つ．ユーザーは，ユーザーのエンティティを参照する．価格については，少なくとも取引の交渉前の段階では，「3 ポイント」や「時給 3 ポイント」など，多様な提示の仕方を許容することにして，自由記述可能な文字列とした．

また，下書きの状態で保存できた方が便利なので，公開に関する属性を真偽値で用意した．以上をまとめたものが図 3 である．

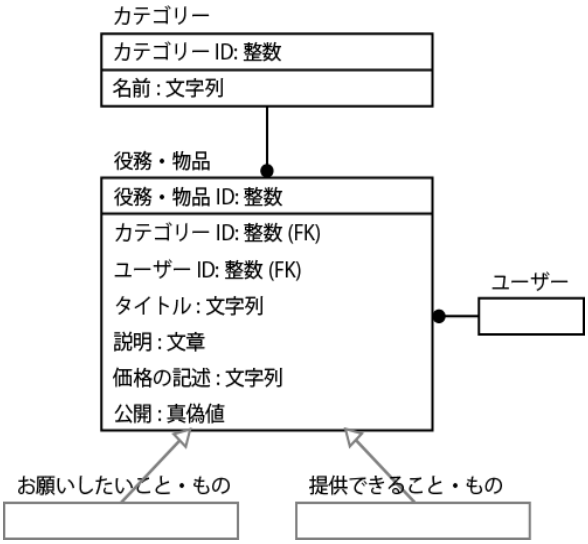

図 3: 役務・物品のエンティティ

3.5 取引に関わるエンティティ

取引を行うには，取引に関わるユーザー，取引内容と状態，支払の大きく 3 つのエンティティが必要と考えた．支払については 3.6 に節を改めて論じる．

まず，取引するユーザーは，依頼者と請負者の 2 者があり，これは 3.3 で論じたユーザーを参照する．依頼者と請負者は，役務・物品を提供・受納するユーザーか取引を開始するユーザーのいずれかであり，対応を表 1 に示した．

取引内容と状態は，取引する役務・物品の説明などの情報，価格などの条件，取引がどの段階(たとえば，支払待ちの状態など)まで進んでいるかの情報から構成されると考えた．

役務・物品の情報は，役務・物品のエンティティと同じである．ただし，これを参照するよりも，取引を開始した段階で役務・物品の情報を複製して保持することにした．なぜなら，役務・物品にはたとえば「犬の散歩」など長期間に渡って公開され，何回も取引される可能性があるものがある．そして，取引が繰り返されるうちに，詳細な記述をよりよく書き換えたくなるのかもしれない．書き換わる可能性がある情報を参照するよりは，取引開始時の情報を複製して保持しておいた方が，記録としては適正と考えた．

取引を行うために実施予定の日時，支払予定額を合意しておく必要があり，これを取引の条件とした．条件を調整するためには，連絡を取り合う必要があり，交渉というエンティティを別途用意することにした．交渉については 3.8 で説明する．

取引の状態の属性を設計するため，取引を分析して状態を抽出し状態機械図(図 4)を作成した．

状態は「条件調整中」「キャンセル」「成約」「実施完了」「支払完了」の 5 つである．

双方が同じ条件で合意すれば「成約」，成約後に実施すると「実施完了」，実施後に支払うと「支払完了」とした．また，実施前であれば，依頼者・請負者のいずれかがキャンセルでき，「キャンセル」状態になるものとした．

双方が同じ条件で合意すれば「成約」になるという状態遷移を実現するために，取引の条件の他に「依頼者による条件提示日時」「請負者による条件提示日時」を属性として持たせることにした．この属性を利用した「条件調整中」→「成約」の状態遷移の様子を表にしたものが表 2 である．「依頼者による条件提示日時」「請負者による条件提示日時」の両方のデータが NULL でなくなれば，成約したと見なすという設計である．

たとえば，「提供できるもの・こと」の取引をはじめるとき，依頼者が条件(実施予定日時と支払額)を提示する．このとき，条件と，依頼者が条件を提示(取引を開始)した日時を記録する．次に，取引相手(請負者)がその条件と同じ条件を提示した場

表 1: 役務・物品の種類と依頼者・請負者

役務・物品の種類	依頼者	請負者
お願いしたいこと・もの	役務・物品を受納するユーザー	取引を開始したユーザー
提供できること・もの	取引を開始したユーザー	役務･物品を提供するユーザー

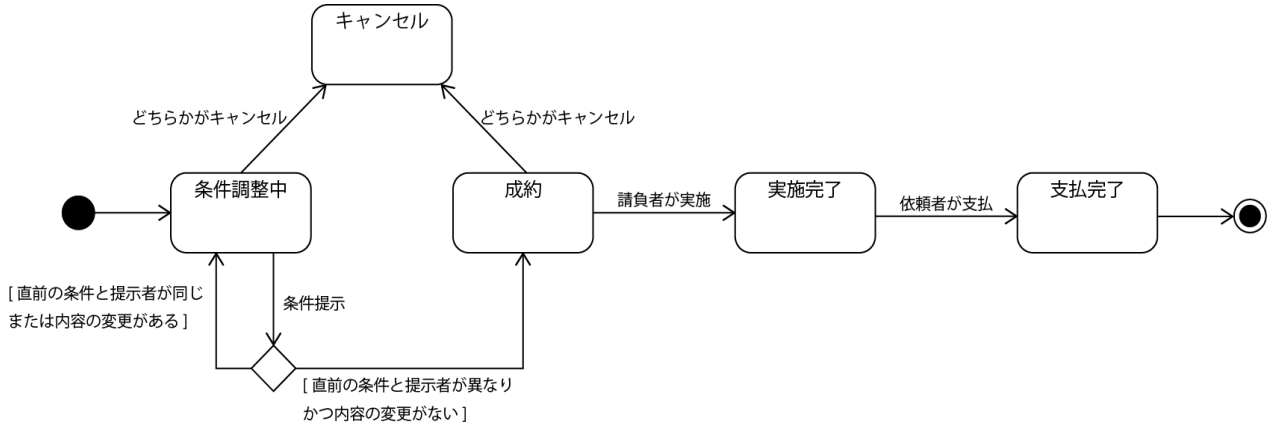

図 4: 取引の状態機械図

表 2: 「条件調整中」→「成約」の状態遷移表

状態		状態識別子	属性の値		依頼者のアクション		請負者のアクション	
					同じ条件を提示	異なる条件を提示	同じ条件を提示	異なる条件を提示
条件調整中	条件提示中 依頼者	S1	依頼者条件提示日時	該当日時	-	S1(遷移なし) ・依頼者条件提示日時を更新	S3 へ遷移 ・請負者条件提示日時を記録	S2 へ遷移 ・依頼者条件提示日時を NULL に ・請負者条件提示日時を記録
			請負者条件提示日時	NULL				
	条件提示中 請負者	S2	依頼者条件提示日時	NULL	S3 へ遷移 ・依頼者条件提示日時を記録	S1 へ遷移 ・依頼者条件提示日時を記録 ・請負者条件提示日時を NULL に	-	S2(遷移なし) ・請負者条件提示日時を更新
			請負者条件提示日時	該当日時				
成約		S3	依頼者条件提示日時	該当日時	-	-	-	-
			請負者条件提示日時	該当日時				

合は,「成約」になり，請負者が条件を提示した日時を記録する．異なる条件を提示した場合は，条件の記録を更新し，請負者が条件を提示した日時を記録する．同時に依頼者が条件を提示した日付のデータを NULL にする．

「実施完了」と「支払完了」への状態遷移は，それぞれ「実施日時」と「支払日時」を記録することで実現する．「キャンセル」への状態遷移は，キャンセルしたのが依頼者か請負者か区別するために，それぞれのキャンセル日時を記録することにした．片方でも記録されると,「キャンセル」へ状態遷移する．

以上の取引の内容や状態に関するエンティティの設計を図 5 に示す．

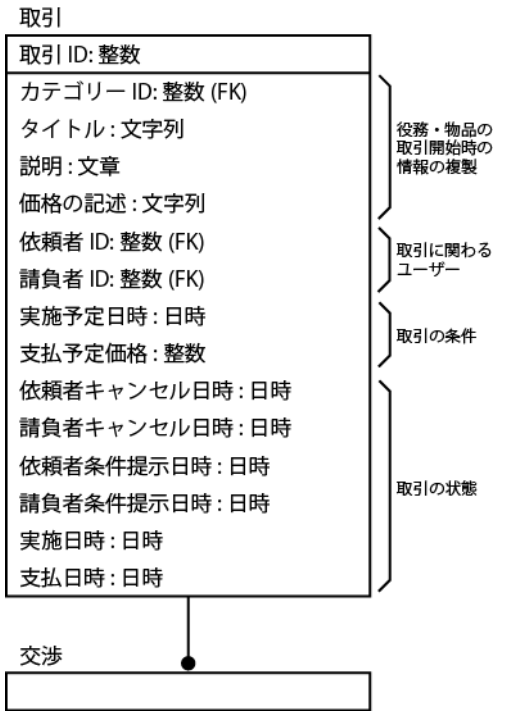

図 5: 取引の内容や状態に関するエンティティ

また，状態を管理するには，エンティティの属性値を直接参照して処理するよりは，ある操作が可能か否かを返す抽象化したメソッドを用意した方がわかりやすいプログラムを書くことができる．そこで，取引にはユーザーを引数に操作が可能か否かを返す以下の機能を持ったメソッドを用意した．

・条件提示可能か？
・実施可能か？
・支払可能か？
・キャンセル可能か？

3.6 支払のエンティティ

支払に関するエンティティは，口座間で金銭を移動するという現実世界の決済を参考にして検討した．エンティティを抽出すると，口座，支払(送金)となる．

口座は残高を持ち，ユーザーを参照する．支払は，支払者口座と受領者口座への参照，金額，件名，備考の情報を持たせることにした．また，取引に結びついた振込ができるように，取引を参照する．さらにシステムの動作を後日監査するために，支払・受領者の口座の取引前後の残高を記録しておくことにした．これらをまとめたのが図 6 である．

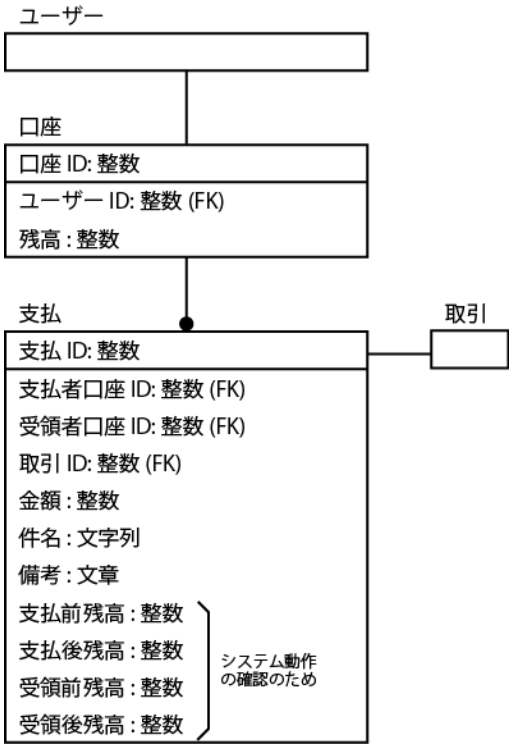

図 6: 支払に関するエンティティ

なお，口座には，口座間で送受金できるメソッドを用意することにした．同時に送受金要求が発行されても適正に処理できるように，メソッドの内容はトランザクションとして実行し，データベースのレコードには実行時にロックをかけること

にした.

3.7 フォロー関係(お気に入り)に関わるエンティティ

お気に入りは，Twitter のフォロー機能を参考に設計した．Twitter は，ユーザーの一部となる大学生たちが親しんでいるサービスである．Twitter のフォロー機能は，フォローしたい相手に対し一方向リンクを生成するものである．フォロー機能を実現するために，図 7 のようにフォローする側とフォローされる側のユーザーを参照する属性を持ったフォロー関係というエンティティを作ることにした.

3.8 会話，交渉のエンティティ

会話と交渉では，相互にメッセージを送信しあう．そこで，送信・受信者，メッセージの本文を持ったエンティティを考えた．送信・受信者はユーザーへの参照である．交渉については，更に関係する取引への参照も持つ．会話と交渉はほぼ構造としては同じなので，STI アーキテクチャ・パターンを採用し，メッセージという親クラスを用意し，継承させることにした．これらを示したのが図 8 である.

3.9 お知らせのエンティティ

「お知らせ」とは，たとえば誰かが自分の公開している「提供できること・もの」について，提供を依頼してきたら，それを連絡してくれる機能である. Twitter, Facebook など近年の SNS では，普通に見られる機能である.

今回開発するプラットフォームでは，お知らせの機能とは，あるユーザーに対して他のユーザーやシステムが何らかのアクションを起こしたときに，通知するものと考えた．より具体的には，通知されるのは，プラットフォームの何らかの RESTful な URL と考えた．URL で示されたページに，誰が何をしたのかがわかる情報が書かれている．そして，通知されたユーザーがその URL を閲覧すると，お知らせは既読になるというしくみである.

そこで，図 9 のように，お知らせのエンティティを構成する属性は，受信者，参照 URL，お知らせの本文，閲覧日時とした.

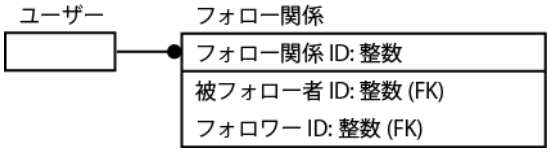

図 7: フォロー関係のエンティティ

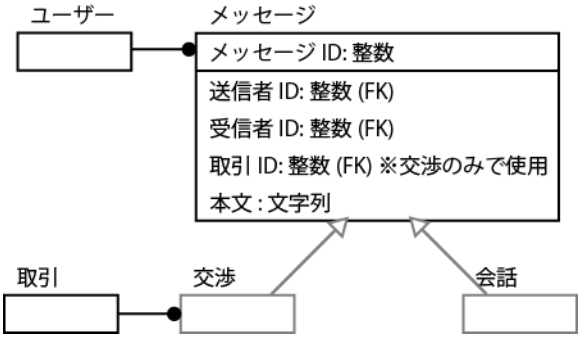

図 8: 会話，交渉のエンティティ

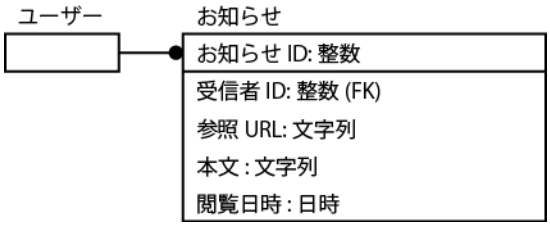

図 9: お知らせのエンティティ

たとえば，自分が「犬の散歩」という「提供できるもの・こと」を公開しているとする．A さんが，これに依頼すると，対応した「取引」のデータが生成され，ページが生成される．すると，自分に「A さんが「犬の散歩」を依頼しています」という本文を持ったお知らせと，「取引」のページの URL が通知されるというわけである.

お知らせは「ユーザーによる他のユーザーへのアクション」が引き金となって発生する．これを実現するには，取引，会話など個々のエンティティにお知らせを送るメソッドを実装するのではなく，ユーザーに他のユーザーへのアクションのメソッドを一括して用意すると，プログラムの見通しがよくなる．そこで，以下のような他のユーザーに対するアクションをユーザーのメソッドとして用意することにした.

・役務・物品の取引を開始する
・取引の条件を提示する
・取引をキャンセルする
・取引を実施する
・支払う
・他のユーザーをフォローする
・フォローを解除する
・話しかける

3.10 ドメイン・モデルのエンティティの全体像

ここまでで設計したドメイン・モデルのエンティティの全体像を図 10 に示す.

4 まとめ

2015 年に青森大学周辺の青森市幸畑地区で, エコマネーに関する実証実験を行うことになっている. そして, そのためのスマートフォン向けの Web プラットフォームを開発することになった.

本論文では, そのプラットフォームのドメイン・モデルの設計を論じた. 設計したドメイン・モデルでは, 役務・物品をエコマネーで交換する機能を実現することができた. 設計は, オブジェクト指向言語を用いた Web アプリケーション・フレームワークと関係データベースを使用する前提で行った. 設計では, 要件をまとめ, システム化する対象領域(ドメイン)のエンティティと属性を抽出し, 必要に応じてメソッドを検討した.

謝辞

本研究開発は, 科研費基盤研究(C)「電子エコマネーを活用したボランティア・コーディネート支援ツールの開発」(研究代表者: 石橋修)の助成を受けたものである.

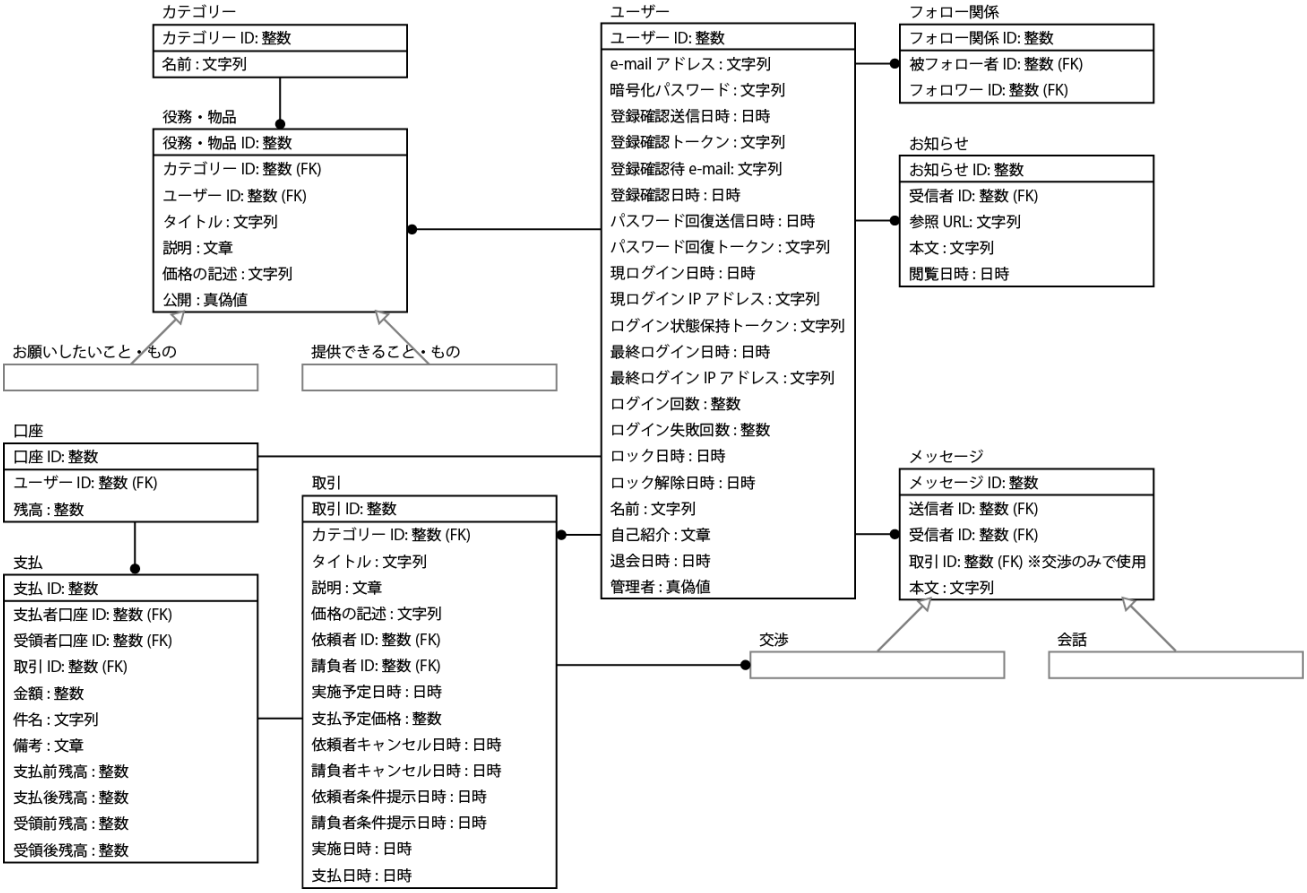

図 10: ドメイン・モデルの全体像

文献

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* (Doctoral dissertation, University of California).

Fowler, M. (2002). *Patterns of enterprise application architecture*. Addison-Wesley Longman Publishing Co., Inc..
(長瀬 嘉秀 監訳. 株式会社テクノロジックアート 訳. (2005). 『エンタープライズアプリケーションアーキテクチャパターン』. 翔泳社.)

Hansson, D. H. (2009). Ruby on rails. *Website. Project site: http://www.rubyonrails.org*.

加藤 敏春. (2001). 『エコマネーの新世紀』. 勁草書房.

Plataformatec. (2009). Devise. *Ruby gem. Project site: https://github.com/plataformatec/devise.*

"salesforce.com, inc.". (n. d.). Heroku. *Cloud platform as a service. Official site: http://www.heroku.com*.

山﨑 茂. (2013). 『地域再生の手段としての地域通貨: 「エコマネー」の可能性と限界に注目して』. 大阪公立大学共同出版会.

The Design of Domain Model for an Eco-Money Web Platform

Atsushi KOKUBO[†], Itaru KASHIWAYA[†], Osamu ISHIBASHI[‡], Motoo KUSHIBIKI[†], Yuusuke SAKAI[†], Teru SASAKI[†] and Sachiko TANAKA[†]
[†]Aomori University, [‡]Hachinone Gakuin University

「エコマネー」は，地域通貨の一種である．平成 27 年度に青森市幸畑地区で，エコマネーの実証実験を実施することになり，そのための Web プラットフォームを開発することになった．エコマネーの利用者は，プラットフォームを利用して，役務や物品をエコマネーを媒介として取り引きする．本論文では，エコマネー・プラットフォームのドメイン・モデルの設計を論じる．われわれはドメインの実体-関係を分析し，主要な構成要素を「ユーザー」「役務・物品」「取引」，副次的な構成要素を「支払」「フォロー関係」「メッセージ」「お知らせ」として設計した．「役務・物品」は「お願いしたいこと・もの」と「提供できること・もの」，「メッセージ」は「会話」と「交渉」をサブクラスとして持つ．これらのサブクラスは「単一テーブル継承」アーキテクチャー・パターンにより実現される．「取引」は，「条件調整中」「キャンセル」「制約」「実施完了」「支払完了」の 5 つの状態を持ち，これらは関連する行動の日付により内部的に表現されている．

キーワード: エコマネー，地域通貨，ドメイン・モデル，オブジェクト関係マッピング，ウェブ・サービス